IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 October 2022, accepted 21 October 2022, date of publication 26 October 2022, date of current version 3 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3217537

RESEARCH ARTICLE

Onboard Powerline Perception System for UAVs Using mmWave Radar and FPGA-Accelerated Vision

NICOLAJ HAARHØJ MALLE, (Member, IEEE), FREDERIK FALK NYBOE, (Member, IEEE), AND EMAD SAMUEL MALKI EBEID, (Senior Member, IEEE)

UAS Center, DIII Group, University of Southern Denmark, 5230 Odense, Denmark

Corresponding author: Nicolaj Haarhøj Malle (nhma@mmmi.sdu.dk)

This work was supported by the European Union's Horizon 2020 Research and Innovation Programme through Drones4Safety under Grant 861111.

ABSTRACT Autonomous Unmanned Aerial Vehicle (UAV) interactions with powerlines, such as close-up inspections for fault detection or grasping and landing for recharging, require advanced onboard perception capabilities. To solve such tasks, the UAV must be equipped with perception abilities that allow it to navigate between powerlines and safely approach specific cables of interest. A perception system with such capabilities requires state-of-the-art sensor technologies and data processing while still being subject to the limited hardware and energy resources of the UAV. In this paper, we present an advanced embedded system based on the cutting-edge Multiprocessing System-on-Chip (MPSoC) for onboard UAV powerline perception. Our platform consists of a mmWave radar and an RGB camera with data processing carried out on the MPSoC, covering both CPU and Field-Programmable Gate Array (FPGA) computations. Following hardware-software co-design methodology, the heavy image processing tasks are accelerated in the FPGA and fused with computationally light mmWave data on the CPU, facilitating pose-estimation of the power lines. Utilizing the open-source autonomy frameworks PX4 and ROS2, we demonstrate integration of the system with onboard path planning based on the estimated cable positions. The robustness of the detection and pose-estimation methods have been demonstrated in several tests performed both in simulated and real-world powerline environments. The results show that our proposed perception system allows the UAV to safely navigate in close proximity to powerlines, by perceiving more individual cables at longer distances compared to previous work, while remaining lightweight, power-efficient, and low-cost.

INDEX TERMS Autonomous UAV, computer vision, FPGA, hardware acceleration, mmWave radar, powerline, sensor fusion.

I. INTRODUCTION

Autonomous operations of drones have significantly increased in the last few years, spanning from large-scale infrastructure inspections and monitoring to interactions and manipulation of hard-to-reach objects [1]. These drones integrate several technologies and algorithms for accomplishing their tasks (e.g., path planning and scene reconstruction, 3D sensing, localization, exploration and navigation) [2], [3]. These tasks pose many challenges due to the large amount of data that needs to be processed onboard of the drone in real time. Central Processing Units (CPUs) and Graphics Processing Units (GPUs) are commonly used to run novel algorithms and handle a wide range of tasks [4]. In contrast, reconfigurable hardware units, so called FPGAs, typically handle a more narrow set of tasks but are able to process massively parallel computations in real-time while being energy efficient compared to CPUs and GPUs. Utilizing the strengths of both general purpose and flexible software on a CPU as well as rigid and deterministic custom hardware accelerating

The associate editor coordinating the review of this manuscript and approving it for publication was Yizhang Jiang.

FIGURE 1. Timelapse showing UAV positions during autonomous flight.

logic circuits on an FPGA makes a heterogeneous system well suited for performing various dynamic and complex robotic workloads efficiently.

In this work, we utilize the cutting-edge heterogeneous and reconfigurable embedded boards featuring chips named MPSoC [5] to build an advanced system that is capable of fusing large amounts of sensory data and running heavy algorithms onboard the drone in real-time. The developed system is designed to detect and pose-estimate powerline cables to accomplish intelligent autonomous drone operations near overhead energized cables such as inspections or recharging.

This work extends our previous work on drones developed to recharge from, interact with, and inspect powerlines [6], [7] by adding an advanced onboard perception platform to the drone system to increase its onboard capabilities, awareness, and smartness to detect and pose-estimate the powerline cables in real-time.

The contributions of this work are

- a novel mmWave and camera sensor fusion algorithm for multi-cable powerline detection and pose-estimation;
- a heterogeneous compute architecture with FPGA-accelerated high-throughput onboard processing; and
- an open-source, efficient, and lightweight powerline perception system.

The developed software is available on GitHub [8] and a video demonstration on YouTube [9] shows the system's performance. A timelapse of the video demonstration can be seen in Fig. 1 showing the autonomous drone alignment near powerlines.

The rest of the paper is structured as follows: Sec. II outlines related work; Sec III introduces the utilized background technologies; Sec. IV explains the applied methodology and how the system is set up; Sec. V, VI, and VII detail the computer vision, sensor fusion, and autonomy aspects of the system; Sec. VIII presents the experimental setup and results; Sec IX discusses the work and highlights points for future work; and finally, Sec. X concludes on the findings.

II. RELATED WORK

Compared to previous works on autonomous powerline approaching, our system performs similarly or better but with significant improvements in severeal areas. LineDrone [10] is based purely on optical perception of the environment to perform the powerline pose estimation. As the system's tests show, adverse lighting conditions limits accurate powerline depth estimation to less than 2 meters using purely camera based estimation, and less than 3 meters using a LiDAR approach. Additionally, their powerline alignment system only works as an assisting tool to a pilot who is in full control of height at all times. [11] demonstrates powerline approach maneauvers. However, this is achieved based on full state knowledge of both drone and powerline and as such does not focus on the perception aspect of the problem. LOCATOR [12] uses two 8-segment solid-state LiDARs for powerline pose estimation. The system is able to autonomously align itself with the nearest detected powerline, but does not take into considerations other nearby powerlines. The chosen sensors also suffer from relatively low range, with one sensor not able to detect a 38mm cable further than 2m away. Additionally, in 2022, a single 8-segment lidar (of which two is used in LOCATOR) costs more than the entire system proposed in this paper.

Several previous works [13], [14], [15] make contributions towards detection of and perching on pipe-like objects. These use some combination of monocular and stereo cameras to detect the pipe-like structures. However, as shown in [16], the performance of stereo depth perception of thin powerline cables deteriorates heavily within a few meters. Additionally, for a purely monocular approach, the diameters of each powerline cable must be known in order to obtain a depth estimate for the powerlines.

Tab. 1 compares metrics of our system with several previous works. Metrics of special interest include perception range, diameter and number of tracked objects, as well as weight, size, and hardware price. The diameter and number of tracked objects as well as the range at which our system can perceive the objects are crucial to the system's ability to enable safe navigation in close proximity to multiple powerlines. Our system outperforms previous work by detecting more, thinner, and further objects and has been shown to be robust in a real outdoor powerline environment. Additionally, our test platform is significantly more lightweight, compact, and affordable than the platforms used in the compared works thanks to the use of novel sensor and compute technologies.

Powerline detection from aerial vehicles using mmWave radars has previously been explored for detection and

TABLE 1. Quantitative comparison between previous work and our system.

Category \ Work	LineDrone [10]	LOCATOR [12]	Thomas et al. [15]	Ramon-Soria et al. [14]	Ours
Sensors	Camera + 2D LiDAR	2x 8-segment 2D LiDARs	Camera	RGB + stereo camera	RGB camera + mmWave radar
Depth estimation method	Depth sensor + image estimation	Depth sensor	Image estimation	Depth sensor	Depth sensor
Perception range	2.5 m (under challenging lighting)	5 m	-	-	10 m
Type of tracked object	Powerline	Powerline	Pipe	Pipe	Powerline
Object diameter	1.9-3.8 cm	3.8 cm	15 cm [a]	15 cm [a]	1 and 2 cm
Number of tracked objects	1	1	1	1	Multiple
Processing system	CPU + GPU	CPU	CPU	CPU + GPU	CPU + FPGA
Perception update rate	30 Hz	-	75 Hz	16 Hz	100 Hz
Level of autonomy	Semi	Full	Full	Full	Full
Approach object from	Above	Below	Below	Above	Below
Weight	14 kg	1.45 kg	-	3 kg	0.61 kg (no battery)
Size (l × w × h)	110 × 110 × 60 cm	-	54 × 54 × - cm	55 × 55 × - cm	28 × 33 × 18 cm
Hardware price	>5000 US$ [b]	1900 US$	-	>1000 US$ [b]	780 US$

[a] Estimation based on relative size between known drone frame and tracked object.
[b] Estimates based on sensor, computer, and drone frame prices.
– Information not found.

warning systems [17], [18]. However, these systems are mainly targeted for larger aircraft and do not focus on high-fidelity detection. Barrett et al. [19] demonstrate a small, low-power mmWave radar device for measuring distances to overhead powerlines.

Compared to sensors of similar price, weight, and size, our previous work [16] shows that a mmWave radar device outperforms a range of other sensors when detecting multiple powerlines from onboard a UAV. Particularly the signal-to-noise ratio of the produced data and the detection range stand out. Other work with UAV-mounted mmWave radars focuses on general depth perception [20], [21] and detection of other UAVs [22], [23]. Additionally, mmWave radars have been extensively studied in driver assistance and autonomous car research [24], [25], [26]. Recently, mmWave radars alongside cameras have been used on UAVs for general object avoidance [27] and specifically powerline avoidance [28]. These works focus on obstacle detection and avoidance rather than object detection and approach, which requires accurate estimates of the objects' positions and precision maneuvers for cable following and grasping.

Cameras are another popular sensor for powerline detection. Santos et al. [29] present PLineD, an algorithm for extracting powerlines from aerial images that runs at 2 Hz on a dual-core Intel Pentium T4300 CPU with 4GB RAM. Zhou et al. [30] demonstrate an algorithm that can autonomously detect powerlines and use it on a UAV to track powerlines in a real-life setting. Recently, deep learning has also been applied to this problem. Son et al. [31] develop and test a powerline detection system for UAVs powered by a tiny-YOLO variant and show its performance running on an embedded device onboard a drone. Vemula et al. [32] use a YOLACT deep learning algorithm to perform instance segmentation on powerline infrastructure elements and show how it can detect and outline powerlines and other related hardware in the image. While these works perform well at powerline detection, they do not provide any information on their 3D position around the UAV.

Other approaches for powerline detection from UAVs often use LiDAR sensors [33], [34]. In previous work from our group, Iversen et al. [12] use two 2D LiDARs to autonomously align a drone with a cable for grasping. However, the used sensors cannot detect wires further than a few meters. Mirallès et al. [10] present a UAV that semi-autonomously lands on powerlines based on camera and LiDAR data. Their system is not fully autonomous and requires expensive hardware.

Another proven but expensive approach is to use event cameras. These sensors are robust to motion blur and adverse lighting conditions, and Dietsche et al. [35] show how their unique data can be used to track powerlines with a UAV. In [36], Konopka introduces a powerline detection system based on fusion between event camera and magnetometer data. Single line detection is achieved on the ground, but the effects of multiple cables or powered flight is not discussed.

Powerline detection based solely on magnetic field sensing is another area of active research. These sensors are unaffected by otherwise common issues like poor weather or lighting conditions, but rely completely on an emitted measurable magnetic field. Vasiljević et al. [37] show how an array of magnetic sensors can be used to obtain an estimate of a current carrying wire. Martinović et al. [38] demonstrate UAV localization around two parallel magnetic field emitting transmission lines. Wu et al. [39] present a scheme for powerline parameter reconstruction based on magnetic sensors and show how their approach can produce a 3D pose estimate of three lines in a lab setting. However, besides relying on current in the powerlines, these approaches only work well when in close proximity to the powerline and usually do not handle multi-line, multi-phase powerline setups. Additionally, none of the methods have been demonstrated to run onboard a UAV in real-time.

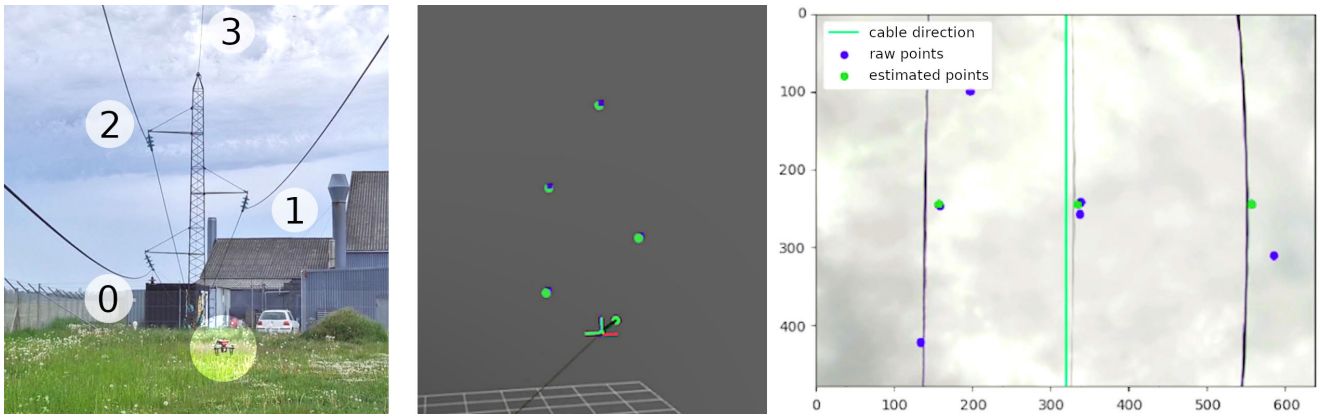


FIGURE 2. Visualization of data generated by powerline perception system. **Left:** Actual drone position in 4-cable powerline test setup with annotated cable numbering. **Middle:** 3D mmWave data and UAV frame as seen in Rviz2. The mmWave radar sensor has a larger field of view than the camera and is able to detect all four cables. **Right:** UAV upward facing camera with 3D mmWave data projected onto image. Given the position of the UAV relative to the cables and the 100° lens of the camera, only three of the four cables are in the camera's field of view. The green cable direction line indicates the perceived powerline direction.

Because of their power efficiency and high throughput capability, FPGAs have been used on UAVs as flight controllers [40], neural network accelerators [41], [42], and generally deployed for their reconfigurability [43], [44]. NASA's Mars UAV [45] uses FPGAs for flight control, fault tolerance, and as IO and communications hubs. FPGAs have also shown great promise for embedded, real-time implementations of advanced algorithms such as linear and nonlinear Model Predictive Control (MPC) [46], [47], [48] and various computer vision algorithms [44], [49]. Ladig et al. [50] show an FPGA-accelerated perception system that assists a pilot in yaw-alignment of a drone relative to detected lines.

III. BACKGROUND

The work builds on several novel and unconventional technologies, such as mmWave radar for sparse point cloud acquisition, as well as hardware acceleration of computationally heavy processing tasks. This section introduces these technologies.

A. ONBOARD COMPANION COMPUTER

For UAVs requiring more autonomy than simply following a set of waypoints, additional onboard processing is usually facilitated by the use of an onboard companion computer. The companion computer may be responsible for sensor interfacing, processing of sensor data, and exchanging information with the flight controller. The type of onboard computer used on a drone platform is typically restricted with respect to its size, weight, and power consumption. Many suitable single board computers are restricted in their computational power which limits the possible complexity of the autonomy and algorithms running on the UAV.

By mixing the use of CPUs and FPGAs, computationally heavy tasks can be accelerated with customized hardware kernels while general tasks run in software. An FPGA is an integrated circuit that in a grid structure features a significant number of various logic gates, lookup tables, memory elements, digital signal processing blocks, and other logic building blocks linked by a configurable array of electrical connections. FPGAs are traditionally programmed in a hardware descriptive language such as VHDL and Verilog that require deep understanding of hardware design.

B. FLIGHT CONTROLLER

Low-level control of the drone is handled by an onboard flight controller. The flight controller ensures the stability and smooth flight of the drone and includes several safety functions such as geo-fencing, return-to-home if radio contact is lost, and automatic landing if the battery is low. Additionally, interfacing with peripherals such as the GPS module and radio and telemetry links as well as the electronic motor speed controllers is handled by the flight controller. Intricate autonomous flight control is achieved by using the so-called offboard mode. In this operational mode, the drone's position, velocity, and acceleration can be controlled by sending control reference trajectory messages to the flight controller from the onboard computer.

C. PERCEPTION SYSTEM

The perception system uses data from several sensors in order to interpret the surroundings of the drone. Distance, inertial, and image sensor data is filtered and fused to enhance this interpretation. Fig. 2 visualizes some of the data captured and produced by the system.

1) mmWave RADAR

Frequency Modulated Continuous Wave (FMCW) radar is a mature technology that has been used in the automotive industry for years. Miniaturization means that these devices are now small and lightweight enough to fit on drones. They provide long-range 3D point measurements, and in the case of powerline detection, they are capable of detecting small

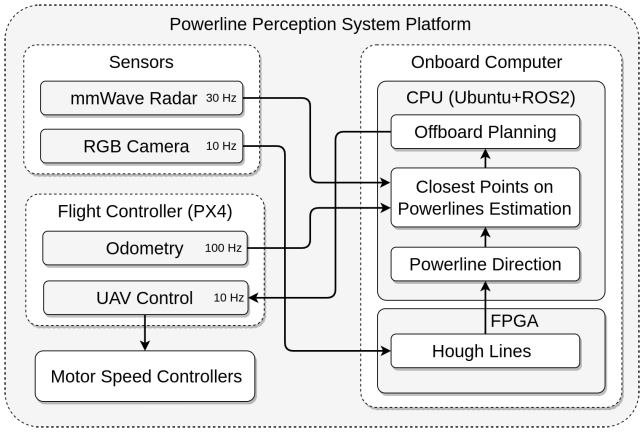


FIGURE 3. High level overview of the powerline perception system dataflow.

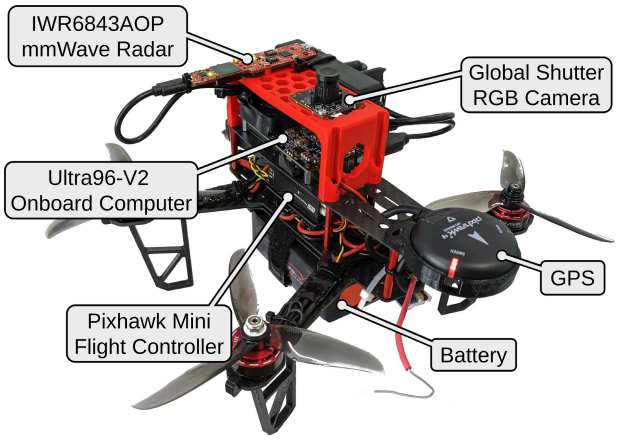


FIGURE 4. Powerline perception system UAV platform based on a QAV250.

cables (1cm diameter) at distances of more than 10 meters. The data produced is very sparse and therefore not an issue to process on embedded platforms. Radar is known to be robust in adverse weather and lighting conditions, and this may become useful when drones are deployed on a large scale.

2) RGB CAMERA

Cameras are extremely versatile sensors that have a huge number of applications. Images contain large amounts of complex information that can be useful in scenarios like powerline detection. However, extracting this information can sometimes be a burden on even large computers, let alone embedded devices. Therefore, the camera is used solely to detect the direction of the powerlines when the mmWave sensor is unable to do so. Unlike mmWave sensors, cameras rely on good lighting conditions to function properly, and weather conditions like fog and heavy rain may obstruct their sensing of the environment.

3) SENSOR FUSION

Our powerline perception system relies on data from several sensors to estimate the positions of surrounding powerlines. The act of creating a single, coherent model or estimate based on multiple sources of data is often referred to as sensor fusion. Each source of data likely provides something that cannot be inferred from the other sources. Therefore, data from all sources must be combined in a way that preserves each source's unique information in order to create the coherent model.

Fig. 3 shows an overview of the system. Various inputs, i.e. mmWave, camera, and odometry data, are processed and fused in several computational nodes in the FPGA and CPU. The resulting coherent powerline estimation can then be used to calculate waypoints for simple autonomous flight.

IV. METHODOLOGY

This section describes the methodology applied for development of the different sub-systems. Fig. 3 shows a high-level overview of the full system. Here, sensor and flight control data is fetched and processed to produce an estimate of each surrounding powerline's position relative to the drone. This is referred to as the perception system. This system relies on an intricate design of software and hardware data processing modules running on the onboard computer. These modules and their underlying dependencies are described as onboard processing. Now, with robust perception capabilities onboard the UAV, autonomy can be implemented to perform tasks such as autonomous powerline alignment.

A. HARDWARE

The platform is based on a Holybro QAV250 racing kit with a maximum payload capacity of 500g. The frame has been modified to place the battery on the bottom instead of on the top and longer legs have been designed to support this. A custom GPS mount is used to place the GPS module in a more convenient position. The Ultra96-V2 is connected to the drone frame via a set of screws, and the sensors are mounted above the onboard computer on a custom sensor mount. A generic USB hub is also attached to the sensor mount and is used to expand the number of available USB ports of the onboard computer to accommodate the three USB devices; mmWave radar, USB camera, and Pixhawk USB-to-serial connection. In our testing, only with FTDI branded USB-to-serial devices were we able to successfully establish a connection between the flight controller and onboard computer. The onboard computer has had its aluminum passive heat sink replaced with an low-profile 12V fan for lower weight active cooling. All USB cables have been cut to length to save weight.

The resulting UAV platform is shown in Fig. 4 with annotations for the most relevant pieces of hardware including the onboard computer, mmWave and camera sensors, and the flight controller.

The entire platform, as shown in Fig. 4 (excluding battery), is assembled from parts worth around US$780 based on manufacturers' suggested retail pricing (MSRP) (2022), see Tab. 2. The battery is excluded because this component's

TABLE 2. Bill of materials of drone platform components.

Name	Price (US$)	Weight (g)	Size (mm)
Holybro QAV250 (Incl. GPS, flight controller & telemetry radio)	288	400	85x276x327
Avnet Ultra96-V2	299	61	26x85x54
Texas Instruments IWR6843AOPEVM	125	12	27x96x8
AR0144 USB camera	58	10	32x32x21
Other (cables, 3D prints, connectors, USB hub)	10	129	-
Total	780	612	183x276x327

specifications varies depending on use-case; a lightweight, cheap, low capacity battery could be used to demonstrate the functionality of this work, while a heavier, more expensive battery with higher capacity would enable longer flights. The specific battery used during test flights is a Turnigy Graphene Professional 4000mAh 4S 15C LiPo Pack that weighs 420g and is available for around 35 US$. The dimensions of the modified platform are 183×276×327 mm and the total weight (without a battery) is 612g.

B. PERCEPTION

The mmWave sensor used in this work is the IWR6843AOP-EVM [51] from Texas Instruments (TI). Its antenna-on-chip design results in a very small and efficient device and a USB connection to the onboard computer makes for easy system integration.

The parameters of the IWR6843 device, such as maximum unambiguous range, range resolution, and update frequency, can be configured through a command-line interface given in a configuration file that has been created with the TI mmWave Demo Visualizer tool [52]. With a Graphical User Interface (GUI) from TI it is possible to fine-tune a configuration to meet the requirements of the use case. Using the GUI, a "Best Range Resolution" preset is fine-tuned to have a maximum unambiguous range of 18.5m, range resolution of 4.5cm, 30 Hz update frequency, and automatic clustering of detected points close to each other. The mmWave device is interfaced via a serial script which we have modified to publish received data as a PointCloud2 topic on the ROS2 network. The script configures the device with the above preset and starts reading data.

The upward facing camera for the powerline perception system has been carefully chosen to be immune to vibrations while still offering a good resolution. Fig. 5 compares images taken with rolling and global shutter - notice the wavy lines in the first three images, resulting from flight induced vibrations and rolling shutter. Such artefacts could pose challenges in the perception pipeline. The AR0144 camera module has a global shutter and offers easy integration via USB in a small and affordable package. The 720p RGB sensor captures enough detail with a 100 degree horizontal field-of-view lens to enable the Hough Lines algorithm to perform as expected. The `usb_cam` ROS2 package is used to publish image data from the camera on the ROS2 network at a rate of 10 Hz.

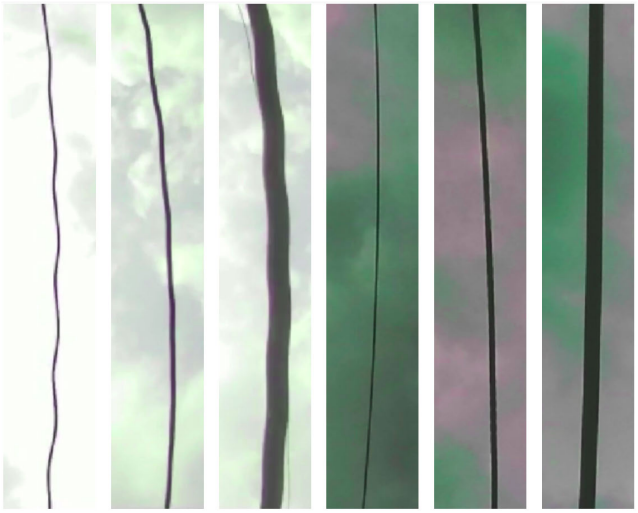


FIGURE 5. Comparison of cable appearance in images taken during flight. a, b, c: Images taken with rolling shutter camera - notice wavy effect. d, e, f: Images taken with global shutter camera.

Most of the image processing takes place on the FPGA to meet throughput requirements without saturating any single CPU core. Ideally, the incoming images should be processed at least as fast as they are received. This means that image processing should take 100 milliseconds or less. Running part of the image processing in a hardware accelerator means that the system can meet the throughput requirements without spending unnecessary CPU resources. The decision to hardware accelerate image processing rather than other parts of the perception system, such as estimation or projection and transformation, was made based on previous observations that the Hough Lines algorithm is demanding on small embedded systems. Additionally, estimation is done using linear low order Kalman filters, which are relatively light to run on the CPU, and the projection and transformation algorithms include equations that are not efficiently executed on FPGAs.

The perception system also relies on vehicle odometry information, and this is fetched from the Pixhawk via a USB-to-serial connection. Information from the flight controller is made available to the onboard computer as a set of topics on the ROS2 network. The UAV inertial state estimate is published to the `/vehicle_odometry` topic at a rate of 100Hz and includes information on the position, velocity, orientation, and angular velocities. These values are derived from sensors internal in the flight controller where their data is also processed and filtered.

C. ONBOARD PROCESSING

The Ultra96-V2 onboard computer [53] features a quad-core application CPU (APU) with 2GB of RAM as well as FPGA fabric and Advanced eXtensible Interface (AXI) bus intra-chip communication infrastructure. Setting up the onboard computer for autonomous drone operations is

non-trivial and time consuming, and the MPSoC4Drones framework [54] exists to ease this process on the Ultra96-V2 single-board computer. The framework targets applications using an external PX4 flight controller and creates bootable images with Ubuntu 20.04, ROS2, PX4 communication, drivers, and custom programmable logic designs. In essence, the framework acts as a wrapper for Xilinx's embedded development tools and lets the developer iteratively develop FPGA circuits for UAV applications. MPSoC4Drones manages the installation of Ubuntu 20.04, drivers, ROS2, PX4 communication dependencies, and custom hardware kernels on the Ultra96-V2 device. The operating system and other developed software runs on the APU, and custom hardware kernels in the FPGA are made available to the operating system as device files (in `/dev`).

The operating system running on the onboard computer is Ubuntu Base 20.04 for Arm architectures that includes a minimal environment suitable for constrained systems. Ubuntu was chosen because most developers are familiar with this distribution and use it on their development workstations. This means less time is spent on solving issues related to porting the developed software from a workstation to the deployment hardware. Version 20.04 is the currently recommended Ubuntu distribution to use with PX4.

For high-level inter-process communication, visualizations, and debugging, ROS2 is used as the middleware of choice. ROS2 is the second major version of the popular set of open-source software libraries and tools that aims to ease the development of robot applications. ROS2, like its predecessor, combines sensor drivers, complex algorithms, and advanced visualizations in an easy-to-use package that is light enough to run on most devices.

For most tasks that are not handled by the operating system, ROS2 Foxy is used as the middleware for ''gluing'' all the computational nodes of the system together. ROS2 was chosen over ROS(1) for its superior networking communication, newer language standards, greater flexibility, and since ROS(1) is nearing end of life and ROS2 represents the future in robotic systems development. ROS2 is used out-of-the-box and no special configurations is applied when running on the onboard computer. ROS2 Foxy is the recommended version to use with PX4 autopilot.

Flight control is handled by an onboard Pixhawk Mini flight controller running the PX4 software stack. PX4 is the most advanced open-source flight-control software on the market. Coupled with a ground control application like QGroundControl, PX4 offers a user-friendly experience for getting started developing applications with drones as fast as possible. Seamless and deep integration with ROS2 enables the development of complex autonomous systems. The PX4 firmware V1.13.0alpha is built with Real Time Publish Subscribe Protocol (RTPS) support, and a microRTPS agent on the onboard computer acts as a bridge between PX4 and ROS2 messages and topics, enabling reading information (e.g. vehicle odometry) from and writing commands (e.g. waypoints) to the flight controller.

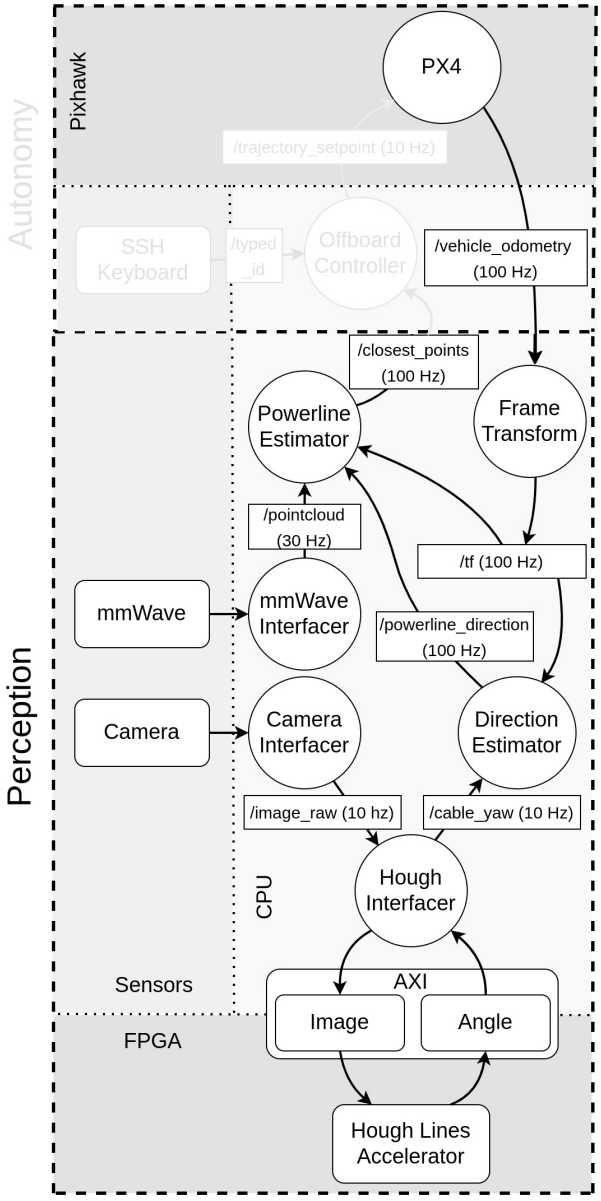


FIGURE 6. Detailed node interaction diagram of the system.

Every piece of software meant to run onboard the drone is written in C++. Nodes for visualization and ground station monitoring are written in Python.

A detailed diagram of the computational nodes and their connections can be seen in Fig. 6. The diagram is split into Perception and Autonomy and again subdivided into Pixhawk, Sensors, CPU, and FPGA to convey where system components are used and how they are processed. Some components are greyed out in the Autonomy section to signify that they are solely used to perform the autonomous flight demonstrations and do not have any functional roles in the perception system itself.

D. UAV AUTONOMY

Autonomous flight is mostly handled by the flight controller, while the onboard computer only publishes desired waypoints

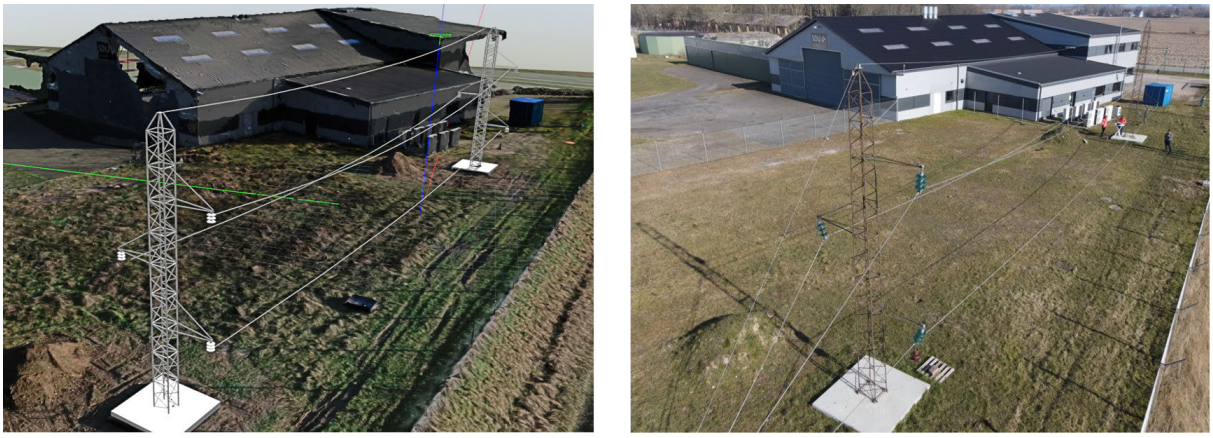


FIGURE 7. The powerline test environment at Hans Christian Andersen Airport in Odense shown in simulation (left) and real life (right). The two towers are approximately 10 meters tall and the span between them is about 35 meters. The bottom three powerlines have a diameter of 20mm while the top cable is 10mm.

to a specific topic on the ROS2 network. UAV position estimation is also handled internally in the flight controller based on Inertial Measurement Unit (IMU) and GPS readings. The flight controller is configured to limit its XY velocity to 2 m/s and Z velocity to 1 m/s, purely for safety reasons. This is done by setting the `MPC_XY_VEL_ALL` and `MPC_Z_VEL_ALL` PX4 parameters via the QGroundcontrol GUI. The baud rate of the serial communication between the flight controller and the onboard computer has been set to 460800 to ensure stable data transfer. In our testing, lower baud rates may interfere with the data transfer via RTPS. Prior to every test, the different sensors in the flight controller are calibrated by following instructions from QGroundControl. While the UAV flies autonomously in the tests presented in this work, a Secure Shell Protocol (SSH) connection between the ground station and the onboard computer is open at all times to monitor system health and provide input when needed, e.g. to choose which cable to align beneath.

E. TEST ENVIRONMENTS

Development and testing starts in our simulated powerline environment shown on the left in Fig. 7. The simulation environment is modeled after the real-world environment; the buildings, ground, and other structures are recreated via photogrammetry while the powerlines and pylons are modelled in CAD since their details are too fine to be accurately recreated via photogrammetry. When a functionality has been validated in simulation, testing is moved outdoors to the real setup, shown in Fig. 7 right. The two towers are approximately 10 meters tall and the span between them is about 35 meters. The bottom three cables, which are meant for 3-phase transmission, are each 20mm in diameter while the top cable, meant as ground, is just 10mm in diameter.

The powerlines and pylons are decommissioned parts obtained from an energy transmission provider, and the real-world powerline setup has the option to be powered with 120 amperes of current to simulate the magnetic field seen in actual transmission infrastructure. This is mainly used to test the effects of the magnetic field on the drone flight controller and other onboard equipment like sensors and computer as well as another source of information that can be used for powerline pose estimation. However, no current flows in the powerlines during the tests performed in this work, and the simulation environment does not support this feature yet.

1) SOFTWARE IN THE LOOP SIMULATION

New functionality is tested in a safe space in a Software In The Loop (SITL) environment during development. This environment runs entirely on a workstation, and testing is done by synthesizing inputs to and responses from the new functionality. SITL is implemented using ROS2 and PX4 with Gazebo as the virtual world simulator. Gazebo plugins have been developed to mimic the functionality and behavior of the sensors mounted on the physical drone. Figure 7 shows a screenshot of the SITL environment on the left.

The perception system running in SITL is mostly identical to the system running onboard the drone with the exception being Hough Lines Transform executed in software since no FPGA is available on the workstation. Besides that, every node and interface remains the same. Instead of real data, the system is instead fed synthetic sensor data and vehicle odometry, and the resulting autonomous flight is based on simulated drone dynamics.

2) HARDWARE IN THE LOOP SIMULATION

Once new functionality has been verified in SITL, Hardware In The Loop (HITL) is used to tune the system's behavior when running on the actual hardware of the UAV while still in a safe space. This includes making sure the new functionality runs on the different compute architecture of the embedded device as well as optimizing for the resource-constrained platform. HITL is implemented similarly to SITL, but instead of running system functionalities on the developer's workstation, the workload is now shifted to the real UAV hardware,

i.e. the Ultra96-V2. New functionalities are tested by running the developed software on the real drone while real or synthetic sensory data is streamed to it. The simplest way of testing and tuning the perception system in hardware with real data, without flying every time a test is needed, is to record the needed data (i.e. vehicle odometry and sensor data) during a flight with ROS2 bags. These bags can then be replayed as many times as needed and the data will publish to the relevant topics as if the system were in flight. The downside to this development methodology is that the autonomy part of the system cannot be tested. Thus, this method is used for fine-tuning the perception system only.

V. IMAGE PROCESSING

This section describes the role of the camera in the powerline perception system as well as how images are processed to extract required information.

While the mmWave radar measurements provide positional data of the powerline cables, the data is too sparse to be able to extract information about the powerline directions. The direction is needed in order to align the drone yaw with the powerline direction e.g. to allow the cable to enter a gripping mechanism in the case of a landing on the cable.

Therefore, in order to fully perceive the powerline environment, a camera is utilized for visual evaluation of the powerline direction which is assumed to be equal among the cables. The camera is mounted pointing upwards in the same direction as the mmWave radar. Images are captured at a rate of 10Hz and processed on the onboard computer. First, the images are passed through a Canny edge detector which binarizes the image based on whether a given pixel belongs to a detected edge or not. The resulting black-and-white image is then loaded into the Hough Line Transform accelerator on the FPGA. Typically, this algorithm will return the position and angle of any detected line in the input image above a threshold. The actual implementation only returns the angles of the 8 lines with the strongest responses. In the FPGA, the accelerator will produce a result within 4.9ms according to the High-level Synthesis (HLS) post-synthesis estimation given a 100 MHz clock frequency, and including data transfer to and from the accelerator with a custom data-loader the total time becomes 38.8ms. The output values of the hardware accelerated Hough Lines Transform given an image are identical to the outputs from the same algorithm running in software.

The resulting angle is then filtered to avoid 180° flips when the relative yaw between the drone and powerline cable crosses 90° or 270°. While the system does not functionally differentiate between e.g. 0° and 180°, this filtering is done to avoid sudden jumps in the final estimated value.

An acceleration module is developed for the FPGA to run the Hough Lines Transform algorithm in hardware. Running algorithms in hardware offloads heavy tasks from the CPU to the FPGA where they can be run on custom circuits to increase deterministic behavior without affecting execution of what is running on the CPU. The hardware acceleration module is developed using Xilinx's HLS tool which synthesises code written in C/C++ with special directives (`#pragma`) into a hardware definition language like VHDL. In the provided Vision Library, Xilinx offers many OpenCV-like algorithms ready to be modified and implemented into an HLS project, and the Hough Lines implementation is modified from that library. The specific implementation of Hough Lines has been modified to only output the angles of the 8 most distinct lines in the input image. When looking at powerlines from below, the clearest lines will always be the mostly straight and monochrome cables against the random patterns of the sky above.

TABLE 3. Hough lines accelerator FPGA resource utilization.

Resource	Utilization	Available	Utilization %
LUT	58676	70560	83.16
LUTRAM	1879	28800	6.52
FF	66420	141120	47.07
BRAM	169	216	78.24
DSP	9	360	2.50
BUFG	14	196	7.14

The Hough Lines accelerator takes up a significant portion of the resources on the FPGA. Tab. 3 shows the utilization of FGPA resources as presented by Vivado post-implementation analysis. While some resources such as DSP slices are underutilized, the hardware accelerated image processing meets the aforementioned 10 Hz output rate requirement and has therefore not been optimized further.

With Vivado, the custom Hough Lines accelerator is integrated into the overall MPSoC hardware architecture. Here, it is connected to the CPU via the AXI bus and memory addresses are assigned to it, which are used to read from and write to the accelerator from software, where it appears as a device file in the Ubuntu operating system. For more information, see our previous MPSoC4Drones work [54].

VI. SENSOR FUSION

This section describes the implemented fusion of mmWave radar, vision, and vehicle odometry data to obtain estimates of the positions of nearby powerlines.

As the powerline cables for the application are assumed to be parallel to each other and perpendicular to the gravity vector, full information about the powerline cables are given by their two dimensional positions and their direction. The plane in which the positions of the cables are described is fully given by the UAV position and the direction of the cables as obtained from the vision system described in Sec. V. Thus, the estimation of the cable poses involves keeping a description of this plane, termed the projection plane, along with the positions of the cables within this plane. The fusion of the three data sources into this description is facilitated using a Kalman filter for the powerline direction and individual Kalman filters for the position of each registered cable.

A tracking scheme is then defined to evaluate the number of powerline cables, to register new cables, and to remove

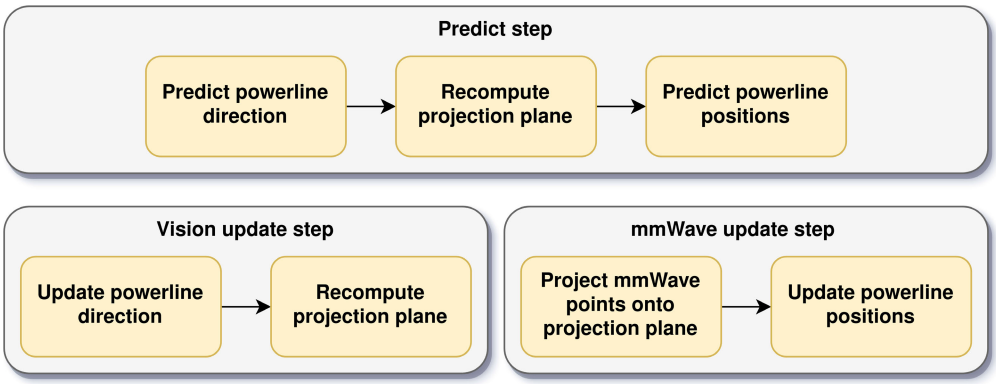


FIGURE 8. Functionality of the sensor fusion kalman filters.

cables that are not perceived any more. The tracking scheme is based on an alive counter, a model of the mmWave sensor's field of view (FOV), and simple Euclidean clustering.

Direction data from the image, depth data from the mmWave radar, and vehicle odometry data from the flight controller are fused together to create a coherent powerline pose estimate with the information from all available sensors. The final output of the perception system is a point cloud containing one point per nearby powerline cable, along with a vector representing the powerline direction. Each point in the point cloud corresponds to the closest point on that powerline relative to the current drone position.

A. PERCEPTION MODEL

Given the powerline direction $\boldsymbol{d}$ and the UAV position p, a projection plane $\mathcal{A}$ is defined as the plane parallel to the gravity vector $\boldsymbol{g}$ and perpendicular to $\boldsymbol{d}$ which intersects p. Thus, $\boldsymbol{d}$ is the normal to $\mathcal{A}$, and $\mathcal{A}$ is fully described by $\boldsymbol{d}$. Since $\mathcal{A}$ is perpendicular to $\boldsymbol{d}$ and intersects p, the closest point on any of the powerline cables will lie in $\mathcal{A}$, as will the shortest vector between p and the powerline cable. This is visualized in Fig. 9. Since the mmWave radar data points are somewhat unpredictable and do not always correspond to the nearest point on the powerline, a scheme is needed to calculate the closest point given a mmWave radar data point.

When new mmWave radar data (in blue in Fig. 9) is obtained from the sensor, the points are projected onto the plane $\mathcal{A}$ in the direction of $\boldsymbol{d}$. The unpredictability of the mmWave radar data is mostly in either direction of d, as is shown later in Fig. 10, so this projection removes the largest source of unpredictability.

B. KALMAN FILTER

To reduce the noise and increase the accuracy of the powerline cable direction and position estimates, the mmWave and vision data is fused with inertial odometry measurements from the flight controller on the drone in multiple linear Kalman filters. A single Kalman filter tracks the powerline direction estimate, while one Kalman filter per registered

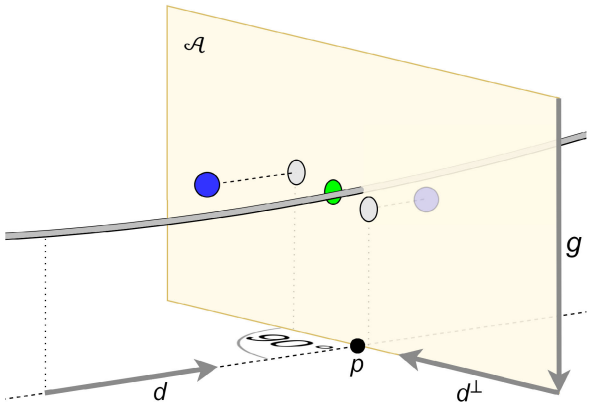


FIGURE 9. The powerline cable perception model with the UAV positioned in the projection plane $\mathcal{A}$.

powerline cable tracks its two dimensional position. However, the Kalman filters are not independent of each other since the direction estimate dictates the formulation of the projection plane which again influences the perceived powerline cable positions.

Thus, the combined filtering algorithm defines three different steps which are visualized in Fig. 8; the *predict step*, the *vision update step*, and the *mmWave update step*, triggered asynchronously by the reception of odometry data, vision data, and mmWave data, respectively. Generally, the reception of vision data, i.e. the powerline direction, will update the direction Kalman filter, and the reception of mmWave data will update the individual powerline cable position Kalman filters. Meanwhile, the reception of odometry data drives the predict step of both the direction and position Kalman filters.

Estimates ${}^{\mathcal{P}}\widehat{\psi}$, $\widehat{\boldsymbol{d}}$, and ${}^{i}\widehat{\boldsymbol{p}}$, $i = 1, \cdots, n$ are kept for the yaw angle ${}^{\mathcal{P}}\psi$ between the powerline direction $\boldsymbol{d}$ and the drone frame x-axis, the powerline direction $\boldsymbol{d}$, and the positions ${}^{i}\boldsymbol{p}$, $i = 1, \cdots, n$ of the closest points to the drone on the n registered powerlines, respectively, as well as variance estimates $\widehat{\psi P}$ and ${}^{i}\widehat{\boldsymbol{pP}} = \left[{}^{i}\widehat{pP}_{\mathrm{x}}\ {}^{i}\widehat{pP}_{\mathrm{y}}\ {}^{i}\widehat{pP}_{\mathrm{z}}\right]^{\mathrm{T}}$, $i = 1, \cdots, n$. The *predict step* of Fig. 8 is driven by the reception of new internal odometry from the flight controller which comes with a

steady period of $\Delta t = 0.01$ s. At time step t, the previous powerline yaw angle estimate ${}^{\mathcal{P}}\widehat{\psi}_{t-1}$ is projected forward to obtain an a priori powerline yaw angle estimate ${}^{\mathcal{P}}\widehat{\psi}_t^-$ along with the a priori variance estimate $\psi\widehat{P}_t^-$ as

$$\begin{aligned} {}^{\mathcal{P}}\widehat{\psi}_t^- &= {}^{\mathcal{P}}\widehat{\psi}_{t-1} - \Delta\psi_t \text{ and} \\ \psi\widehat{P}_t^- &= \psi\widehat{P}_{t-1} + \psi Q, \end{aligned} \tag{1}$$

where $\Delta\psi_t = \psi_t - \psi_{t-1}$ is the change in drone yaw angle ψ between time step $t-1$ and t as obtained from the flight controller and ψQ is the model noise variance. Here, ${}^{\mathcal{P}}\widehat{\psi}_{t-1}$ is either the previous a priori estimate ${}^{\mathcal{P}}\widehat{\psi}_{t-1}^-$, or the previous a posteriori estimate ${}^{\mathcal{P}}\widehat{\psi}_{t-1}^+$ if the *vision update step* has executed, as detailed below. The powerline direction estimate $\widehat{\boldsymbol{d}}_t$ at time step t, describing the projection plane $\mathcal{A}$ is then computed by applying the negative drone pitch angle to the drone x-axis unit vector followed by a yaw rotation corresponding to the estimated powerline yaw angle as

$$\widehat{\boldsymbol{d}}_t = \boldsymbol{R}_{\mathrm{z}}({}^{\mathcal{P}}\widehat{\psi}_t) \cdot \boldsymbol{R}_{\mathrm{y}}(-\phi_t) \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \tag{2}$$

where ϕ_t is the drone pitch angle at time step t as obtained from the flight controller internal odometry. Finally, the a priori position estimates ${}^i\widehat{\boldsymbol{p}}_t^-$ of currently tracked powerlines $i = 1, \cdots, n$ at time step t are predicted along with their a priori variance estimates ${}^i\boldsymbol{p}\widehat{\boldsymbol{P}}^-$ as

$$\begin{aligned} {}^i\widehat{\boldsymbol{p}}_t^- &= {}^i\widehat{\boldsymbol{p}}_{t-1} - \Delta\boldsymbol{p}_t \text{and} \\ {}^i\boldsymbol{p}\widehat{\boldsymbol{P}}^- &= \quad \forall i = 1, \cdots, n, \end{aligned} \tag{3}$$

where $\Delta\boldsymbol{p}_t = \boldsymbol{p}_t - \boldsymbol{p}_{t-1}$ is the change in drone position $\boldsymbol{p}$ between time step $t-1$ and t, obtained from the flight controller. Here, the ${}^i\widehat{\boldsymbol{p}}_{t-1}$ is either the previous a priori powerline position estimates ${}^i\widehat{\boldsymbol{p}}_{t-1}$ or the previous a posteriori powerline position estimates ${}^i\widehat{\boldsymbol{p}}_{t-1}^+$ if the *mmWave update step* has been executed, as explained below.

The *vision update step* will execute at time step t if the observed yaw angle ${}^{\mathcal{P}}\mathrm{v}\psi_t$ between the drone and the powerlines at time step t is received from the image processing pipeline, triggering an update of the powerline direction estimate, referring to Fig. 8. Here, the a posteriori angle estimate ${}^{\mathcal{P}}\widehat{\psi}_t^+$ and angle variance estimate $\psi\widehat{P}_t^+$ are obtained as

$$\begin{aligned} \psi\bar{y}_t &= {}^{\mathcal{P}}\mathrm{v}\psi_t - {}^{\mathcal{P}}\widehat{\psi}_t^-, \\ \psi S_t &= \psi\widehat{P}_t^- + \psi R, \\ \psi K_t &= \frac{\psi\widehat{P}_t^-}{\psi S_t}, \\ {}^{\mathcal{P}}\widehat{\psi}_t^+ &= \psi K_t \cdot \psi\bar{y}_t, \quad \text{and} \\ \psi\widehat{P}_t^+ &= (1 - \psi K_t) \cdot \psi\widehat{P}_t^-, \end{aligned} \tag{4}$$

where ${}^{\mathcal{P}}\widehat{\psi}_t^-$ is the a priori powerline angle estimate at time step t, $\psi\widehat{P}_t^-$ is the a priori estimate of the powerline angle estimate at time step t, $\psi\bar{y}_t$ is the powerline direction measurement residual, ψS_t is the powerline direction measurement residual variance, ψR is the powerline direction measurement noise variance, and ψK_t is the powerline direction Kalman gain. From the a posteriori powerline yaw angle estimate, the projection plane $\mathcal{A}$ description is updated by again computing the powerline direction estimate $\widehat{\boldsymbol{d}}_t$ using Eq. 2.

The *mmWave update step* is triggered at time step t if a mmWave pointcloud is received containing m measured points ${}^j\mathrm{m}\boldsymbol{p}_t, j = 1, \cdots, m$. Based on the powerline direction estimate $\widehat{\boldsymbol{d}}_t$, the received points are projected onto $\mathcal{A}$ to obtain the projected points ${}^j\mathrm{m}\boldsymbol{p}_t^{\mathcal{A}}, j = 1, \cdots, m$:

$${}^j\mathrm{m}\boldsymbol{p}_t^{\mathcal{A}} = {}^j\mathrm{m}\boldsymbol{p}_t - \left(\frac{\left[\widehat{\boldsymbol{d}}_t\right]^{\mathrm{T}} {}^j\mathrm{m}\boldsymbol{p}_t}{\left[\widehat{\boldsymbol{d}}_t\right]^{\mathrm{T}} \cdot \widehat{\boldsymbol{d}}_t} \right) \cdot \widehat{\boldsymbol{d}}_t, \quad j = 1, \cdots, m, \tag{5}$$

following standard procedure for projecting a point on a plane given the plane normal. Then, for each currently tracked powerline $i = 1, \cdots, n$ given the existence of a point ${}^k\mathrm{m}\boldsymbol{p}_t^{\mathcal{A}}$ in the set of projected measured points ${}^j\mathrm{m}\boldsymbol{p}_t^{\mathcal{A}}, j = 1, \cdots, m$, satisfying

$$\begin{aligned} |{}^k\mathrm{m}\boldsymbol{p}_t^{\mathcal{A}} - {}^i\widehat{\boldsymbol{p}}_t| &\leq \mathrm{dist}_{\max} \quad \text{and} \\ k &= \underset{j}{\operatorname{argmin}} \left(|{}^j\mathrm{m}\boldsymbol{p}_t^{\mathcal{A}} - {}^i\widehat{\boldsymbol{p}}_t| \right), \end{aligned} \tag{6}$$

the position update step is executed, obtaining the a posteriori powerline position estimate ${}^i\widehat{\boldsymbol{p}}_t^+$ and variance estimate ${}^i\boldsymbol{p}\widehat{P}_t^+$ at time step t. The following formulation describes the x-axis computation but is similarly computed for the y- and z-axis, as

$$\begin{aligned} {}^i\boldsymbol{p}\bar{y}_{\mathrm{x},t} &= {}^k\mathrm{m}p_{\mathrm{x},t}^{\mathcal{A}} - {}^i\widehat{p}_{\mathrm{x},t}^- \\ {}^i\boldsymbol{p}S_{\mathrm{x},t} &= {}^i\boldsymbol{p}\widehat{P}_{\mathrm{x},t}^- + \boldsymbol{p}R \\ {}^i\boldsymbol{p}K_{\mathrm{x},t} &= \frac{{}^i\boldsymbol{p}\widehat{P}_{\mathrm{x},t}^-}{{}^i\boldsymbol{p}S_{\mathrm{x},t}} \\ {}^i\widehat{p}_{\mathrm{x},t}^+ &= {}^i\boldsymbol{p}K_{\mathrm{x},t} \cdot {}^i\boldsymbol{p}\bar{y}_{\mathrm{x},t} \\ {}^i\boldsymbol{p}\widehat{P}_{\mathrm{x},t}^+ &= (1 - {}^i\boldsymbol{p}K_{\mathrm{x},t}) \cdot {}^i\boldsymbol{p}\widehat{P}_{\mathrm{x},t}^-. \end{aligned} \tag{7}$$

Here, k is the index of the point in the set of points ${}^j\mathrm{m}\boldsymbol{p}_t^{\mathcal{A}}, j = 1, \cdots, m$ closest to the current estimate ${}^i\widehat{\boldsymbol{p}}_t$ of the position of powerline i, $\mathrm{dist}_{\max}$ is the maximum distance threshold for considering a measured point as belonging to a currently tracked powerline, ${}^i\widehat{\boldsymbol{p}}_t^-$ is the a priori powerline i position estimate at time step t, ${}^i\boldsymbol{p}\widehat{P}_{\mathrm{x},t}^-$ is the estimate of the variance of the a priori powerline i position x-axis estimate at time step t, ${}^i\boldsymbol{p}\bar{y}_{\mathrm{x},t}$ is the powerline i position measurement x-axis residual, ${}^i\boldsymbol{p}S_{\mathrm{x},t}$ is the powerline i position measurement x-axis residual variance, $\boldsymbol{p}R$ is the powerline position measurement noise variance assumed to be equal across all axes, and ${}^i\boldsymbol{p}K_{\mathrm{x},t}$ is the powerline i position x-axis Kalman gain.

C. MATCHING AND TRACKING

In order to match received mmWave points to existing powerline cable position estimates, the projected points (in light grey in Fig. 9) onto $\mathcal{A}$ are clustered based on a threshold for Euclidean distance. If the point is within the distance threshold from an already registered point, it will be considered as a measurement of the position of the corresponding cable.

Once a set of estimated closest powerline points has been produced, they must be merged with the previous estimates to

update the current best estimate. If the current best estimate does not have a point in the same approximate location as a point in the latest set of projection estimates, a new point in the best estimate is spawned at that location. Each new point in the best estimate is assigned a unique ID such that they can later be selected when choosing a cable to align beneath.

If the current best estimate already has a point in the same approximate location as a point in the latest set of projection estimates, the Kalman filter update step is triggered for that particular best estimate point with the projection estimate point as corrective input.

If no new projection estimate point matches an existing current best estimate point, a new position for the best estimate point is instead predicted based on the observed vehicle movement derived from the flight controller odometry, as resulting from the Kalman filter functionality.

Each new point in the current best estimate is assigned an `alive`-counter. If a best estimate point is updated with new mmWave radar data, the counter is incremented. If the counter is above a threshold, the point becomes a valid powerline estimate which can for example be selected as a point to align beneath during an autonomous mission. If no new mmWave data matches a best estimate, and a new position is instead predicted based on odometry data, the counter is decremented if its point's new position is within the sensor's field of view. Once the counter of a best estimate point reaches 0, it is deleted. If the point is outside the sensor's field of view, it will continuously have new positions predicted via odometry data until it comes back into field of view. Using a counter like this means that points that get less frequent but regular data have a chance of becoming powerline estimates. Similar for points that for some reason receive no new data for a few cycles. And points that move outside the sensor's field of view are still tracked, though only with odometry predictions which make them prone to drift over time.

Algorithm 1 outlines the powerline tracking implementation where plane $\mathcal{A}$, vector $\widehat{\boldsymbol{d}}$, and point p were explained with the perception model, $\boldsymbol{D}$ is a matrix holding new mmWave points, and $\boldsymbol{C}$ is a matrix holding the current powerline position estimates.

Algorithm 1 Powerline Tracking Algorithm

Require: $\widehat{\boldsymbol{d}}, p, \boldsymbol{D}, \boldsymbol{C}$

while tracking powerlines **do**
  if received $\boldsymbol{D}$ **then**
    $\mathcal{A} \leftarrow$ `createPlane`$(p, \widehat{\boldsymbol{d}})$
    $\boldsymbol{D}_{\text{proj}} \leftarrow$ `projectOnPlane`$(\boldsymbol{D}, \mathcal{A})$
    for $i = 0$ **to** `sizeOf`$(\boldsymbol{D}_{\text{proj}})$ **do**
      `index` $\leftarrow$ `findMatch`$(\boldsymbol{C}[:], \boldsymbol{D}_{\text{proj}}[i])$
      if match found **then**
        `update`$(\boldsymbol{C}[\text{index}], \boldsymbol{D}_{\text{proj}}[i])$
        `incrementAliveCounter`(index)
      else
        `pl` $\leftarrow$ `createPowerline`$(\boldsymbol{D}_{\text{proj}}[i])$
        `append`$(\boldsymbol{C}, \text{pl})$
      end if
    end for
  else if received p **then**
    for $j = 0$ **to** `sizeOf`$(\boldsymbol{C})$ **do**
      `predict`$(\boldsymbol{C}[j], \Delta p)$
      `decrementAliveCount`(j)
      if `getAliveCount`(j) is 0 **then**
        `deleteElement`$(\boldsymbol{C}, j)$
      end if
    end for
  end if
end while

VII. SYSTEM AUTONOMY

As seen in Fig. 6 the section responsible for autonomous flight is only a small part of the overall system and has no impact on the perception system itself. This section briefly outlines how the implemented flight autonomy works.

The main node for the autonomy part of the system is the Offboard Controller. This node issues desired trajectory setpoints to the flight controller based on two inputs. The first input is the current powerline estimates which holds both the closest points of each detected powerline as well as the direction of the powerlines. The second input is used to determine which powerline the UAV should align beneath. As mentioned in Sec. VI, each powerline estimate has a unique ID. If a valid ID (i.e. a powerline estimate with this ID exists) is published, the Offboard Controller accepts it as the goal powerline. For the sake of demonstration, this ID can be entered from the ground station, but other sensors like magnetometers could be used to identify and determine which cable to go to in a dynamic setting.

Once the drone is switched to offboard mode and a valid ID has been issued, the drone will arm itself, take-off to a set altitude, and then begin alignment beneath the selected powerline point. Originally, the Offboard Controller would send a setpoint 1 meter below the selected powerline with a desired final yaw as well. This implementation resulted in significant overshooting of both the position in XY and yaw. To correct this, intermediary position and yaw setpoints are calculated and issued instead. These intermediary setpoints are simply calculated as being a fraction towards the desired position and yaw with a configuration parameter to determine the size of the fraction.

VIII. EXPERIMENTAL RESULTS

This section describes the tests performed to validate the system and the results of the tests are presented. This includes static and autonomous flight tests in a real powerline environment.

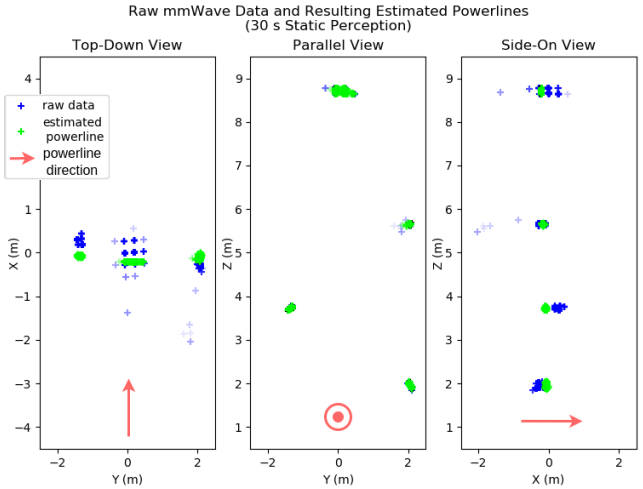


FIGURE 10. Raw mmWave data and estimated powerline positions after sensor fusion over a 30 second period. Drone positioned at (0,0,0).

A. PERCEPTION SYSTEM

Fig. 10 shows the raw mmWave data points (blue) and resulting powerline estimates (green) over a period of around 30 seconds. The data is collected with the system standing on the ground and remaining in the same position throughout the test duration. The red arrow on the bottom of each plot shows the direction of the powerline relative to the data.

As previously mentioned, the mmWave data exhibits a certain unpredictability as to where a point is produced along the direction of an observed powerline. This effect can also be observed here, where raw data points are spread out more along the powerline direction (Fig. 14 in Sec. IX shows a very pronounced example of this behavior as well).

The job of the perception system is to estimate the closest point on each powerline and to reduce noise in the data. The output of the system shown as green points have less spread in all directions compared to the raw data. Additionally, they are all in the plane that is perpendicular to the powerline direction. This is especially clear in the XZ and XY plots where all the green points appear to all share the same X value. The larger variance in the Y-direction of the powerline estimate of the highest cable compared to the three lower ones may be explained by a more pronounced sway induced by the wind since this cable is significantly lighter than the three lower ones.

The depth accuracy of the perception system is mostly dependent on the accuracy of the incoming mmWave radar data. We have compared the accuracy of mmWave radar data with other sensors such as LiDAR and stereo cameras in our previous work on sensors for powerline detection [16]. A simple indoor ground truth measurement was conducted by placing the perception system equipped drone beneath a section of suspended powerline cable. The distance between the mmWave radar and the 40mm diameter cable was measured to be 100cm, and the mmWave radar sensor was placed directly beneath the cable. In this situation, the output of the perception system should be a single point at XYZ(0m, 0m, 1m).

TABLE 4. Result of static perception 1m below cable.

Axis	Mean (m)	Variance (m)
X	-0.026	$2.425 \cdot 10^{-5}$
Y	-0.023	$1.098 \cdot 10^{-5}$
Z	0.982	$8.015 \cdot 10^{-6}$

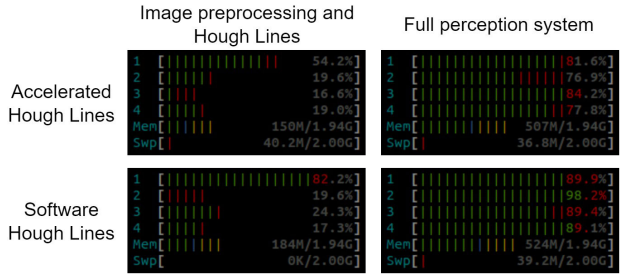


FIGURE 11. System load differences between the software and accelerated Hough Lines algorithm implementations.

The actual outputs of the 10 second static measurement can be seen in Tab. 4. The error in any axis is below 3% of the distance to the target.

Fig. 11 compares typical CPU and memory loads when running the Hough Lines algorithm in software and in the hardware accelerator.

The left column compares the loads when running only image pre-processing and Hough Lines, and the right columns compares the loads when the full perception system is running. In both situations the loads are lower when running the accelerated algorithm which makes sense since the accelerator removes part of the workload off the CPU. Comparing the throughput of the two implementations also shows that the accelerator has an advantage; processing 1000 images, the software implementation takes on average 50.5ms to compute a result while the accelerator takes just 38.8ms with a clock frequency of 100 MHz - the software implementation takes 29% more time to produce the same output. Additionally, the variance in the execution times is much greater in the software implementation, and additional system load would likely impact the software implementation to a greater degree than the accelerator, since only data loading is CPU reliant with the accelerator. The HLS tool used to build the accelerator reports a total estimated latency of just 4.9ms with a 100 MHz clock frequency. These estimates are quite accurate, and the remaining 33.7ms are caused by inefficiencies in the way data is written to the acceleration kernel in the custom data-loader implementation. The data-loader utilizes AXI-Lite which is meant for simple memory-mapped communication with low expected throughput. Given the relatively high throughput of the application, the protocol causes large delays. For future work we are moving to AXI, AXI-Stream, or AXI-DMA. However, despite the delays, the image processing meets the aforementioned 100 millisecond execution time requirement and is therefore not negatively impacted by the significant data loading wait times.

B. FLIGHT TESTS

After verifying the system's functionality in simulation, the testing is moved to the real powerline environment at the Hans Christian Andersen Airport in Odense as seen in Fig. 7 on the right. Prior to testing, the drone is placed on the ground approximately halfway between the pylons. Here it is powered on and the ground station initiates an SSH connection with the onboard computer over WiFi. From the ground station, the microRTPS agent for PX4 communications as well as the whole perception system is launched on the drone. Once all nodes are confirmed healthy, the drone is ready for autonomous flight demonstrations.

The test procedure is as follows:

- From the ground station, the drone is put in the Offboard mode which means it is now in control of its own flight.
- From the ground station, a detected powerline ID is chosen for the drone to align beneath.
- The drone will arm itself, takeoff to a specified height, and then align at 1 meter below the selected powerline.
- New powerline IDs can be chosen to have the drone fly between the powerlines.
- From the ground station, a landing can be commanded which will make the drone land at the current XY position and disarm itself.

A video of such a test flight demonstration can be found on our YouTube channel [9], and Fig. 1 shows a time-lapse of the flight.

An autonomous flight demonstration is analyzed in the following paragraphs. The test procedure is as outlined previously, and the UAV is asked to align beneath a sequence of cables: cable 1, cable 2, cable 1, cable 2, and finally cable 3 (refer to Fig. 2 for cable numbering). Going from a lower number to a higher number cable means the drone likely has the cable in its field of view, but when going from a higher number to a lower number this is usually not the case. Therefore, going from cable 2 to cable 1 in the above sequence showcases the system's ability to successfully track, re-acquire, and finally align beneath powerlines that have moved outside field of view.

Figure 12 shows the path of the drone when completing the mentioned sequence of powerline alignments. The first part of the path in yellow shows the takeoff and alignment at cable 1. When looking at the YZ plot, there is a large deviation from the closest path. This happens because the drone is unbalanced and the thrust at takeoff is too low, which results in the drone dragging along the ground until more height is gained. This is also evident when watching the video of the flight. The next part of the path in green is the drone's alignment below cable 2. Then the drone is commanded to again align with cable 1, and this path is shown in blue and looks as smooth as the previous alignment despite the fact that this powerline is outside the field of view of the perception system for most of the maneuver. The next section in red is the second alignment at cable 2, followed by alignment at the top cable, number 3.

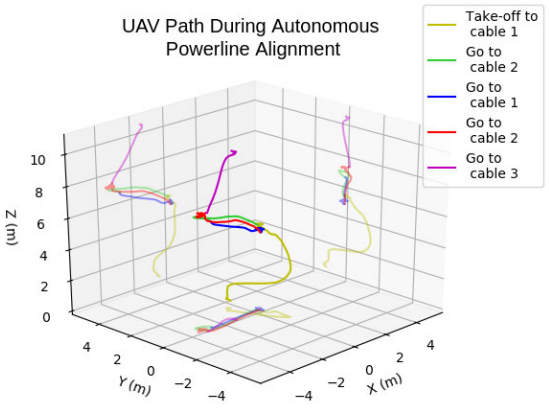


FIGURE 12. Path of drone during powerline alignment derived from inertial odometry.

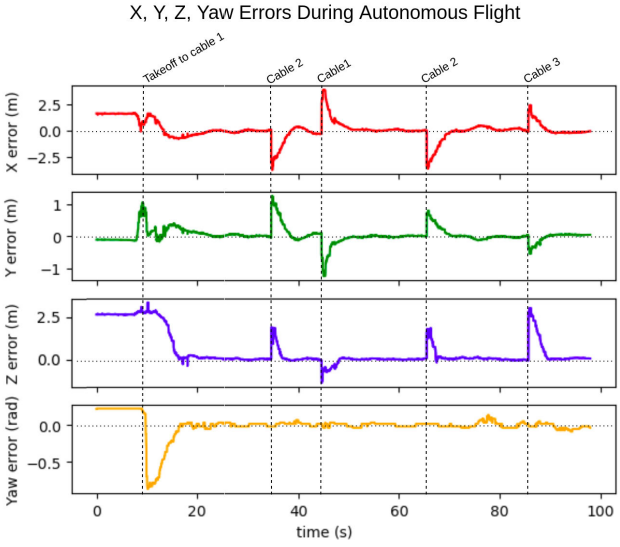


FIGURE 13. X, Y, Z, Yaw errors during autonomous cable alignment mission.

Fig. 13 shows the X, Y, Z, and yaw difference between the desired and actual drone pose over the course of the same flight as covered in Fig. 12. A difference, or error, of 0 means that the drone has aligned perfectly under the cable with respect to that parameter. From the figure it can be seen that alignment takes about 6 seconds in yaw as well as the XY plane while being completed in around just 3 seconds in the Z direction - despite the fact that Z velocity is limited to 1m/s while 2m/s is allowed in the XY plane. Besides the faster completion, the Z direction alignment also shows less overshoot and deviation over time.

The test flight shows that the perception system can enable safe and simple navigation within the powerline environment. On the day of testing, winds of around 20km/h were blowing at the test facility, which is at the upper limit of what the system is designed to withstand. While the tests show no significant performance degradation in 20km/h winds, the system is not meant to fly in precipitation and stronger winds.

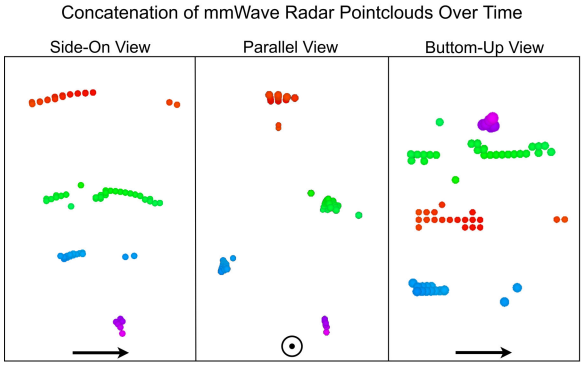


FIGURE 14. Result when concatenating mmWave radar point clouds of the powerline environment overtime. Notice that there seems to be significantly more spread along the powerline direction (indicated by black arrow). Colors indicate distance in Z direction.

IX. DISCUSSION AND FUTURE WORK

As seen in the autonomous flight test demonstration, the powerline perception system works as intended. However, several points of interest are being investigated for the next iteration of the system.

The first point on the future work agenda is the removal of the camera dependency. While the powerline direction estimation is currently robust to different weather and lighting, it will not work in darkness, in environments with fog or smoke, or if rain or snow obstructs the lens. The mmWave radar is mostly free of these shortcomings and therefore it would be ideal to be able to rely only on that sensor. Additionally, this would lower the weight and cost of the overall system.

Fig. 14 shows the result of concatenating mmWave radar point clouds of the powerline environment over a period of several seconds while not moving the sensor. The plots suggest that the points are significantly more spread out along the powerline direction (indicated by the black arrow) compared to any other direction. This behavior may be exploited to extract the powerline direction purely from mmWave data, but a method must be developed for doing it dynamically without having to concatenate several seconds' worth of point clouds without moving.

The second point of interest for future investigations is the potential for combining the perception system with more advanced trajectory planning. The current method works but is simply a crude demonstration. For missions where more complex paths are required, e.g. for landing the drone on a powerline, techniques like model predictive control may be used to generate more fitting trajectories.

A third opportunity for future investigations is to reduce the resource utilization of the Hough Lines accelerator on the FPGA as well as optimize the AXI interface. Currently, most of the chip area is used for just the one algorithm. With FPGAs, high throughput means higher resource utilization. Since the images are fetched from the camera at just 10 Hz, the higher throughput capability of 25.8 FPS (or even 204 FPS if considering the HLS tool's throughput estimate), and consequently higher resource utilization, of the current accelerator is wasted. Reducing the throughput to match the image feed rate would likely allow for additional hardware accelerators to run simultaneously.

The fourth point of interest for future work is to improve the tracking of powerlines when outside the sensors' field of view. One approach is to save the transforms between all the estimated powerlines. If a powerline leaves the field of view, its position can be calculated relatively accurately based on previously saved transforms between the powerline and other powerlines that are still in field of view. As long as one powerline remains in field of view, this method will be able to calculate the position of any previously seen powerline if the respective transform has been saved. In a worst case scenario where all powerlines are outside the field of view, the currently implemented method would still perform the tracking, although only with the accuracy provided by the flight controller's vehicle odometry.

Finally, research from our group suggests that the magnetic field may be another source of data with which to perform even more accurate and robust powerline pose estimations. The same magnetometers could also be used to select a cable with the optimal current when performing a landing and recharging mission. Additionally, the magnetic field in a powerline environment can induce unexpected behavior in the flight controller, and this particular issue is another topic worth investigating.

X. CONCLUSION

The autonomous flight demonstrates that the perception system works as intended. Given a desired goal powerline, the drone is able to align below it within seconds. A system demonstration can be viewed on YouTube [9].

Several points of interest for future investigations have been suggested, most notably ability to extract the powerline direction from mmWave radar data, thus removing the camera dependency of the system. The perception system is applicable in many use cases that require operations in close proximity to powerlines. It could serve as a navigational aid or semi-autonomous flight assistance during manual inspection flights. Currently, much research is devoted to aerial manipulation and inspection of infrastructure, and a powerline perception system would greatly ease autonomous navigation in powerline clusters. Another application is powerline landings where a drone latches onto the powered cable and recharges itself.

Finally, the code developed for the perception system as well as the tools used to configure the onboard computer with required software and drivers are fully available on the Drone Infrastructure Inspection and Interaction (DIII) Group GitHub page [55].

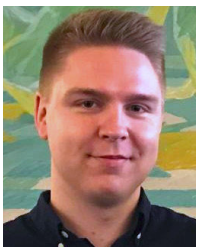
REFERENCES

[1] S. Liu, Z. Wan, B. Yu, and Y. Wang, *Robotic Computing on FPGAs* (Synthesis Lectures on Computer Architecture), vol. 16. Springer, Jun. 2021, pp. 1–218, doi: 10.2200/S01101ED1V01Y202105CAC056.

[2] D. G. Vutetakis and J. Xiao, "An autonomous loop-closure approach for simultaneous exploration and coverage of unknown infrastructure using MAVs," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 2988–2994.

[3] Q. Kuang, J. Wu, J. Pan, and B. Zhou, "Real-time UAV path planning for autonomous urban scene reconstruction," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 1156–1162.
[4] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "A survey of FPGA-based neural network inference accelerators," Assoc. Comput. Machinery, New York, NY, USA, Mar. 2019, pp. 1–26, Art. no. 2, vol. 12, no. 1, doi: 10.1145/3289185.
[5] *Xilinx*. Accessed: Sep. 2, 2021. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html
[6] N. Iversen, O. Schofield, L. Cousin, N. Ayoub, G. Vom Bögel, and E. Ebeid, "Design, integration and implementation of an intelligent and self-recharging drone system for autonomous power line inspection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep./Oct. 2021.
[7] N. Iversen, A. Kramberger, O. B. Schofield, and E. Ebeid, "Pneumatic-mechanical systems in UAVs: Autonomous power line sensor unit deployment," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 548–554.
[8] N. H. Malle, F. F. Nyboe, and E. Ebeid. (2022). *ROS2 Package for the III-Drone System*. [Online]. Available: https://github.com/DIII-SDU-Group/III-Drone-ROS2-pkg
[9] DIII. *Drone Infrastructure Inspection & Interaction Group Youtube Channel*. Accessed: Oct. 10, 2022. [Online]. Available: https://youtu.be/hNfyQMsd-AQ
[10] F. Miralles, P. Hamelin, G. Lambert, S. Lavoie, N. Pouliot, M. Montfrond, and S. Montambault, "LineDrone technology: Landing an unmanned aerial vehicle on a power line," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 6545–6552.
[11] J. L. Paneque, J. Ramiro Martínez-de Dios, A. Ollero, D. Hanover, S. Sun, A. Romero, and D. Scaramuzza, "Perception-aware perching on powerlines with multirotors," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 3077–3084, Apr. 2022, doi: 10.1109/LRA.2022.3145514.
[12] N. Iversen, O. B. Schofield, and E. Ebeid, "LOCATOR–lightweight and low-cost autonomous drone system for overhead cable detection and soft grasping," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot. (SSRR)*, Nov. 2020.
[13] F. von Frankenberg and S. Nokleby, "Detection of long narrow landing features for autonomous UAV perching," in *Proc. 11th IEEE Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Nov. 2020, pp. 0565–0570.
[14] P. Ramon-Soria, A. E. Gomez-Tamm, F. J. Garcia-Rubiales, B. C. Arrue, and A. Ollero, "Autonomous landing on pipes using soft gripper for inspection and maintenance in outdoor environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 5832–5839.
[15] J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar, "Visual servoing of quadrotors for perching by hanging from cylindrical objects," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 57–64, Jan. 2016.
[16] N. H. Malle, F. F. Nyboe, and E. Ebeid, "Survey and evaluation of sensors for overhead cable detection using UAVs," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2021, pp. 361–370.
[17] I. Ünal and S. Eker, "Investigations on millimeter wave detection of power lines from a safe distance," in *Proc. 10th Int. Conf. Elect. Electron. Eng. (ELECO)*, Nov./Dec. 2017, pp. 964–967.
[18] S. Futatsumori, C. Amielh, N. Miyazaki, K. Kobayashi, and N. Katsura, "Helicopter flight evaluations of high-voltage power lines detection based on 76 GHz circular polarized millimeter-wave radar system," in *Proc. 15th Eur. Radar Conf. (EuRAD)*, Sep. 2018, pp. 218–221.
[19] D. Barrett, D. Wang, A. Ahmad, and V. Mahimkar, "Using mmWave sensors to enhance drone safety and productivity," Texas Instrum., Dallas, TX, USA, white Paper SPYY001, 2017.
[20] C. Lim, B. Li, E. M. Ng, X. Liu, and K. H. Low, "Three-dimensional (3D) dynamic obstacle perception in a detect-and-avoid framework for unmanned aerial vehicles," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2019, pp. 996–1004.
[21] W. Zhang, J. Heredia-Juesas, M. Diddi, L. Tirado, H. Singh, and J. A. Martinez-Lorenzo, "Experimental imaging results of a UAV-mounted downward-looking mm-wave radar," in *Proc. IEEE Int. Symp. Antennas Propag. USNC-URSI Radio Sci. Meeting*, Jul. 2019, pp. 1639–1640.
[22] S. Dogru and L. Marques, "Pursuing drones with drones using millimeter wave radar," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4156–4163, Jul. 2020.
[23] B. B. Tierney and C. T. Rodenbeck, "3D-sensing MIMO radar for UAV formation flight and obstacle avoidance," in *Proc. IEEE Radio Wireless Symp. (RWS)*, Jan. 2019, pp. 1–3.
[24] D. Barnes, M. Gadd, P. Murcutt, P. Newman, and I. Posner, "The Oxford radar RobotCar dataset: A radar extension to the Oxford RobotCar dataset," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020.
[25] T. Zhou, M. Yang, K. Jiang, H. Wong, and D. Yang, "MMW radar-based technologies in autonomous driving: A review," *Sensors*, vol. 20, no. 24, p. 7283, Dec. 2020.
[26] X. Gao, S. Roy, and G. Xing, "MIMO-SAR: A hierarchical high-resolution imaging algorithm for mmWave FMCW radar in autonomous driving," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 7322–7334, Aug. 2021.
[27] H. Yu, F. Zhang, P. Huang, C. Wang, and L. Yuanhao, "Autonomous obstacle avoidance for UAV based on fusion of radar and monocular camera," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5954–5961.
[28] H.-H. Chang, Z.-X. Wang, and J.-J. Lin, "Radar and image fusion for power line detection in UAV applications," in *Proc. IEEE 4th Int. Conf. Knowl. Innov. Invention (ICKII)*, Jul. 2021, pp. 199–202.
[29] T. Santos, M. Moreira, J. Almeida, A. Dias, A. Martins, J. Dinis, J. Formiga, and E. Silva, "PLineD: Vision-based power lines detection for unmanned aerial vehicles," in *Proc. IEEE Int. Conf. Auto. Robot Syst. Competitions (ICARSC)*, Apr. 2017, pp. 253–259.
[30] G. Zhou, J. Yuan, I.-L. Yen, and F. Bastani, "Robust real-time UAV based power line detection and tracking," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 744–748.
[31] H.-S. Son, D.-K. Kim, S.-H. Yang, and Y.-K. Choi, "Real-time power line detection for safe flight of agricultural spraying drones using embedded systems and deep learning," *IEEE Access*, vol. 10, pp. 54947–54956, 2022.
[32] S. Vemula and M. Frye, "Real-time powerline detection system for an unmanned aircraft system," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2020, pp. 4493–4497.
[33] F. Azevedo, A. Dias, J. Almeida, A. Oliveira, A. Ferreira, T. Santos, A. Martins, and E. Silva, "Real- time LiDAR-based power lines detection for unmanned aerial vehicles," in *Proc. IEEE Int. Conf. Auto. Robot Syst. Competitions (ICARSC)*, Apr. 2019, pp. 1–8.
[34] R. Sabatini, A. Gardi, and M. Richardson, "LIDAR obstacle warning and avoidance system for unmanned aircraft," *Int. J. Mech., Ind. Mechtron. Eng.*, vol. 8, no. 4, pp. 718–729, 2014.
[35] A. Dietsche, G. Cioffi, J. Hidalgo-Carrio, and D. Scaramuzza, "Powerline tracking with event cameras," 2021, *arXiv:2108.00515*.
[36] J. L. Konopka, "Dynamic vision sensor and magnetometer data fusion for autonomous power line detection using unmanned aerial vehicles," M.S. thesis, Univ. Southern Denmark, Odense, Denmark, 2019.
[37] G. Vasiljevic, D. Martinovic, M. Orsag, and S. Bogdan, "Grabbing power line conductors based on the measurements of the magnetic field strength," in *Proc. Aerial Robotic Syst. Physically Interacting Environ. (AIRPHARO)*, Oct. 2021, pp. 1–7.
[38] D. Martinović, S. Bogdan, and Z. Kovačić, "Mathematical considerations for unmanned aerial vehicle navigation in the magnetic field of two parallel transmission lines," *Appl. Sci.*, vol. 11, no. 8, p. 3323, Apr. 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/8/3323
[39] Y. Wu, G. Zhao, and J. Hu, "Overhead transmission line parameter reconstruction for UAV inspection based on tunneling magnetoresistive sensors and inverse models," *IEEE Trans. Power Del.*, vol. 34, no. 3, pp. 819–827, Mar. 2019.
[40] B. Cain, Z. Merchant, I. Avendano, D. Richmond, and R. Kastner, "PynqCopter—An open-source FPGA overlay for UAVs," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 2491–2498.
[41] F. F. Nyboe, N. H. Malle, and E. Ebeid. (2022). *MPSOC4Drones: An Open Framework for ROS2, PX4, and FPGA Integration*. [Online]. Available: https://portal.findresearcher.sdu.dk/en/publications/mpsoc4drones-an-open-framework-for-ros2-px4-and-fpga-integration
[42] B. B. Kövari and E. Ebeid. (2021). *MPDrone: FPGA-based Platform for Intelligent Real-time Autonomous Drone Operations*. [Online]. Available: https://portal.findresearcher.sdu.dk/en/publications/mpdrone-fpga-based-platform-for-intelligent-real-time-autonomous-
[43] S. V. Ragavan, V. Ganapathy, and E. C. M. Xian, "A reconfigurable FPGA framework for data fusion in UAV's," in *Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC)*, 2009, pp. 1626–1631.
[44] E. Moreac, E. M. Abdali, F. Berry, D. Heller, and J.-P. Diguet, "Hardware-in-the-loop simulation with dynamic partial FPGA reconfiguration applied to computer vision in ROS-based UAV," in *Proc. Int. Workshop Rapid Syst. Prototyping (RSP)*, Sep. 2020, pp. 1–7.
[45] B. Balaram, T. Canham, C. Duncan, H. F. Grip, W. Johnson, J. Maki, A. Quon, R. Stern, and D. Zhu, "Mars helicopter technology demonstrator," in *Proc. AIAA Atmos. Flight Mech. Conf.*, Jan. 2018, pp. 1–18.
[46] K. M. Abughalieh and S. G. Alawneh, "A survey of parallel implementations for model predictive control," *IEEE Access*, vol. 7, pp. 34348–34360, 2019.

[47] S. P. Diwan and S. S. Deshpande, "Nonlinear model predictive controller for the real-time control of fast dynamic system," in *Proc. Int. Conf. Commun. Electron. Syst. (ICCES)*, Jul. 2019.
[48] V. Patne, D. Ingole, and D. Sonawane, "FPGA implementation framework for low latency nonlinear model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 7020–7025, 2020, doi: 10.1016/j.ifacol.2020.12.443.
[49] D. Honegger, H. Oleynikova, and M. Pollefeys, "Real-time and low latency embedded computer vision hardware based on a combination of FPGA and mobile CPU," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 4930–4935.
[50] R. Ladig, S. Leewiwatwong, and K. Shimonomura, "FPGA-based fast response image analysis for orientational control in aerial manipulation tasks," *J. Signal Process. Syst.*, vol. 90, no. 6, pp. 901–911, Jun. 2018.
[51] *IWR6843AOPEVM Evaluation Board | TI.com*. Accessed: Oct. 10, 2022. [Online]. Available: https://www.ti.com/tool/IWR6843AOPEVM
[52] *Texas Instruments*. Accessed: Aug. 3, 2022. [Online]. Available: https://dev.ti.com/gallery/view/mmwave/mmWave_Demo_Visualizer/ver/4.2.0/
[53] *Avnet*. Accessed: Sep. 2, 2021. [Online]. Available: https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/ultra96-v2/
[54] F. F. Nyboe, N. H. Malle, and E. Ebeid, "MPSoC4Drones: An open framework for ROS2, PX4, and FPGA integration," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2022, pp. 1246–1255.
[55] *DIII Group*. Accessed: Sep. 8, 2021. [Online]. Available: https://github.com/DIII-SDU-Group



NICOLAJ HAARHØJ MALLE (Member, IEEE) was born in Denmark. He received the B.S. degree in robotic systems and the M.S. degree in advanced robotics technology from the University of Southern Denmark, in 2018 and 2020, respectively. He is employed as a Ph.D. Fellow at the UAS Center, University of Southern Denmark, under the EU Financed H2020 Research Project Drones4Safety. His research interests include embedded systems, perception for UAVs, and AI on the edge, which he brings together in his research in drone perception systems to enable accurate landing on power lines for recharging of the UAV.

FREDERIK FALK NYBOE (Member, IEEE) was born in Denmark. He received the B.S. degree in robotic systems and the M.S. degree in drones and autonomous systems from the University of Southern Denmark, in 2019 and 2022, respectively. He is employed as a Ph.D. Fellow at the UAS Center, University of Southern Denmark, under the EU Financed H2020 Research Project Drones4Safety. His research interests include embedded systems, control, and sensing for UAVs, which he brings together in his research in techniques for achieving highly accurate landing on power lines for recharging of the UAV.

EMAD SAMUEL MALKI EBEID (Senior Member, IEEE) received the Ph.D. degree in distributed embedded systems from the University of Verona, Italy, with a European Doctorate Label, in 2014. He is currently an Associate Professor and the Leader of the DIII Research Group, University of Southern Denmark. He is also the Coordinator of the EU H2020 Drones4Safety and Innovation Fund Grand Solutions Drones4Energy Projects. He has published more than 70 articles in flagship journals and conferences in robotics and embedded systems. His research interests include autonomous drone system design for infrastructure inspection and interaction utilizes cutting-edge technologies (MPSoCs and FPGAs) to build an advanced reconfigurable onboard unit for controling the drone in real-time to detect and grasp the powerlines for inspection and recharging applications.