Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Last Mile Urban Drone Delivery to Mobile Outdoor Clients

Semester Thesis

Raphael Schnider

sraphael@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Gino Brunner, Simon Tanner
Prof. Dr. Roger Wattenhofer

February 8, 2019

Acknowledgements

I thank the Distributed Computing Group of ETH Zurich for the possibility of conducting this thesis and providing the necessary resources. I also thank to Anna Kiener who helped to adapt the 3d printable mounting mechanism and to the D-ITET workshop which helped to print it.

Abstract

Drone delivery is currently a hot topic in the industry, however existing approaches do not provide the ability to make deliveries to individual customers in an urban environment. This thesis shows a proof of concept of a drone capable of delivering payloads to mobile clients in an urban environment. We build a prototype using an off-the-shelf drone to which we add an additional camera. The drone starts close to the approximate drop-off location and detects and approaches the client using computer vision. The target is indicated by a visual marker which is displayed on a smartphone. We test the system in a simulated environment and in a real-world scenario.

Contents

Chapter 1

Introduction

Altough drone delivery is a hot topic, companies like Amazon, Google, UPS mostly focus on rural areas with almost no buildings in sight. But according to this statistic [1] 55% of the world population live in an urban environment with a growing tendency. Therefore, it seems important to tackle the last mile drone delivery problem in an urban environment.
There are two reasons why a different approach than simple GPS navigation is needed:

- The GPS signal reflects off of buildings and therefore the received GPS position can include up to approximately 10 meters of inaccuracy. This is unacceptably big if the goal is to deliver payloads to individual clients,e.g.,to their balcony.
- To avoid collisions with obstacles while approaching the client sensors for mapping the environment of the drone are needed

Brunner, Szebedy, Tanner and Wattenhofer [2] developed a prototype drone capable of accurately finding and approaching a visual marker attached to a wall in an urban environment using visual navigation and collision avoidance.
Our contribution is a new target tracker. It uses a smartphone as target marker instead of a printed symbol attached to a wall. According to this statistic [3] 90% of the swiss population own a smartphone. Therefore, the developed approach is very mobile as potential clients do not need any special equipment except their smartphone.
Our approach is based on the work of Nakazawa et al. [4] who use LEDs with two different brightness levels and a PPM modulation to track and identify individual LEDs for indoor positioning of a ground robot.
There are some other approaches for tracking LEDs, e.g., Nagura et al. [5] who develop a visual light communication mechanism which relies on an LED array to detect the correct signal. But they are not applicable to our scenario because they either rely on:

- LED arrays, which we cannot use because the target location tracker is

supposed to work with a single smartphone and dividing the already small smartphone screen into an array of markers would significantly reduce the detection range.

- Special unique color of the LED which is used in a predefined environment where it is known that no other object has this particular color. We cannot use this approach because the detection has to work outdoors in a generic environment where we do not know which colors are present.

Other research papers in the area of autonomous drone delivery include the work of Krakowczyk et al. [6] who focus on the distributed aspect of an autonomous drone delivery fleet and Mardiansyah et al. [7] who focus on an emergency payload delivery system including a mechanism to release the payload. They also use visual navigation and target detection. Unfortunately they do not share the implementation details but it seems that they use a very big marker with a color unique in the environment. Further, Amazon [8], Google [9] and UPS [10] all have current research projects on autonomous drone delivery in rural areas.

Chapter 2

Background

This thesis is based on and extends the work of Brunner, Szebedy, Tanner and Wattenhofer [2] who show a proof of concept for the urban last mile delivery using the Intel Aero RTF drone [11].
Their work provides the following software components which are used in this thesis:

- **Visual Odometry:** vision based localization using Semi-Direct Visual Odometry (SVO) [12] and calibrated for the Intel Aero RTF drone
- **Trajectory Planner:** calculates a collision free trajectory depending on the current position of the drone, the desired trajectory endpoint and the image stream of the depth cameras. It uses a three-dimensional circular buffer based on the proposition of Usenko et al. [13] to create an occupancy grid of the drone environment.
- **Control Logic:** interface between the system components and responsible for autonomous mission execution. It uses ROS topics to communicate with the other nodes and the MAVROS package to communicate with the autopilot.
- **Simulation Environment:** Gazebo simulation environment with a pre-configured scenario including a model of the Intel Aero RTF drone and simulated camera streams.

Figure 2.1 shows an overview of the hardware components of the unmodified Intel Aero RF drone and the following list includes the most important specifications of the used Intel Aero RTF drone:

- Intel Aero Flight Controller
  - PX4 Autopilot
  - STM32 Microcontroller
  - IMU, magnetometer, altitude sensors

Figure 2.1: Intel Aero RTF drone [11]

- Intel Aero Compute Board
  - Ubuntu 16.04
  - Intel Atom x7-Z8750 Processor
- Realsense R200 camera, front facing
- GPS

Chapter 3

Approach

Our scenario is the following: A client uses a smartphone app to start the delivery. The smartphone transmits its approximate GPS location and displays a marker. The delivery drone then autonomously navigates to the received GPS coordinates flying above the rooftops, because the GPS signal is accurate in this height since it is not disrupted by buildings. Then it switches to vision based navigation and descends to an altitude above ground from which it can reliably detect the target. If the target cannot be directly localized the drone starts to scan the area until the target is found. The drone approaches the target, drops the payload next to it and then leaves. The target marker needs to face the sky and no obstacles should be located directly above the target, otherwise the drone is not able to see the target.
The target marker displayed on the smartphone are plain red and blue screens which toggle with a certain frequency. Therefore the target cannot be detected from a single frame and the color of objects needs to be tracked over time. All potential targets, meaning they have either the color blue or red, are tracked using optical flow to determine if one of them changes to the other color and is therefore the target marker.
However, this work does not treat the transmission of the GPS coordinates from the client, the payload release, and the return back to the launch location. These topics need to be approached in future work.

Chapter 4

System Architecture

An overview of the system architecture of the autonomous drone can be seen in Figure 4.1. The application is executed on two different processors on the drone: the autopilot runs on an embedded microcontroller with real-time guarantees and all other software components, including the visual marker tracker, are executed on a quad-core processor located on the compute board. All these software tasks except the autopilot use the Robot Operating System (ROS) [14] as a middleware. ROS processes are called nodes and they can communicate through message queues called topics.
The following list briefly explains the task of each software component:

- **Autopilot:** Take position setpoint as input and based on the estimate of the current position calculate the required motor power as output. PX4 is used like in [2] without changes.
- **Visual Odometry:** Take visual data from a down-facing monochrome camera as input. The output is the estimate of the local position. This block is especially important in urban environments where the accuracy of GPS localization is limited. The component of [2] is used without changes.
- **Trajectory Planner:** Take depth images from front- and bottom-facing RealSense camera and compute an occupancy grid. Based on desired endpoint coordinates, the trajectory planner computes the setpoints for the drone which result in a collision free trajectory to the endpoint and publishes them on the appropriate ROS topic. Our implementation is based on the corresponding software component of [2] and extended with depth information from the additional depth camera that we attach to the drone.
- **Visual Marker Tracker:** Take RGB and depth frames as input and publish the position of the target, if detected, in camera coordinates. If the distance is unknown, a normalized vector pointing in the direction of the target is published. We develop this component from scratch with a new approach.

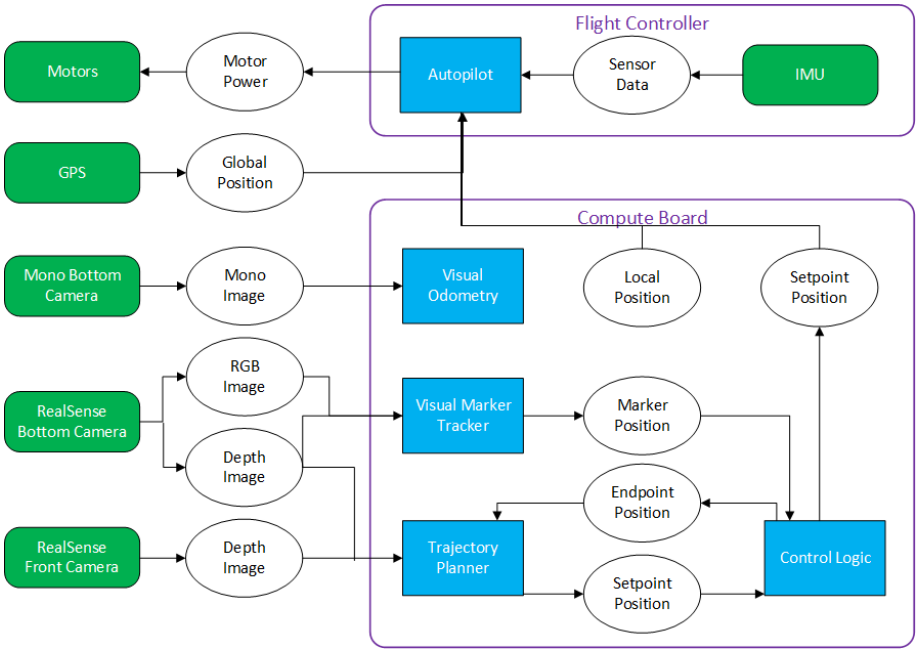


Figure 4.1: System architecture of the drone. The rounded green rectangles represent hardware components, software components are represented by the blue rectangles.

- **Control Logic:** Is responsible for the autonomous execution of the mission. It manages the current system state and is the interface between the two processors. Based on the current system state it calculates the next endpoint position and sends setpoint positions to the autopilot based on the generated trajectory for the given endpoint. The implementation we use is based on the previous work of [2] and adapted to the new target marker scenario.

Chapter 5

Implementation

The quad-core processor of the Intel Aero RTF drone allows us to run Ubuntu 16.04 and the Robot Operating System(ROS), as well as computer vision algorithms, on-board. Compared to [2] an additional bottom camera is mounted to the drone and a new method for the visual target tracker is implemented. The depth information of the added camera is added to the occupancy grid and enhances collision avoidance. The control logic is only slightly adapted to work with our visual target tracker.

5.1 Robot Operating System (ROS)

The Robot Operating System is a middle-ware or framework for building robot applications. The processes of such an application are called nodes and they communicate through message queues which are called topics. ROS is compatible with various data types as topics and it is also possible to define new data types. Only the visual target tracker node is written from scratch. To publish the images from the camera added at the bottom of the drone in ROS topics an open source node from the community is used. All other nodes are based on the previous work of [2] with some small adaptions.

5.2 Bottom Camera

To detect the target marker and improve collision avoidance with additional depth information, a RealSense D435 [15] camera is attached to the bottom of the drone as seen in Figure 5.1. It is a wide-angle camera with global shutter and high resolution. The camera can measure distances from 0.2 meters to approximately 10 meters.
The camera is screwed onto a 3D printed holder based on this model [16] which is attached to the drone.
To publish the RGB and depth image as ROS topics, the realsense2_camera [17] ROS wrapper is used.

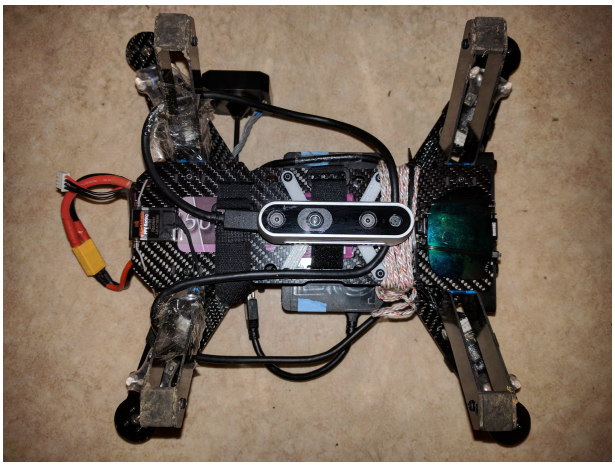
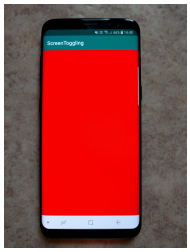
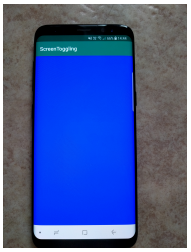


Figure 5.1: Bottom view of the Intel Aero RTF drone with the Intel RealSense D435 camera attached attached to the 3D printable mounting part in white.



(a) Red screen of the target marker



(b) Blue screen of the target marker

Figure 5.2: Red and blue screen of the target marker.

5.3 Visual Marker

We develop a simple Android application capable of switching the display between red and blue color indicating the target location. Figure 5.2 shows the two different screens. For testing a frequency of 1 Hz is used. A possible adaption in the future could be to use an appropriate modulation scheme, e.g. PPM as used by Nakazawa et al. [4] for identification of the delivery recipient.

5.4 Visual Marker Tracker

The visual marker tracker is implemented as a ROS node. It receives depth and color images from the bottom camera and publishes the target position in camera coordinates if the target is found. Additionally it also needs to publish the transformation from camera to drone frame coordinates using the tf2 ROS transform package.

Computer vision algorithms from the OpenCV [18] library are used to process the color frames. In each cycle, the following steps are executed:
Receive a new frame from the appropriate ROS topic published by the bottom camera. This frame is processed by applying an averaging filter and a series of erosions and dilations and then converted to the HSV color model. Thresholding operations on the HSV image are executed to detect the pixels in the desired red or blue color. The resulting contours are filtered according to their size: only contours with at least 4 pixels are accepted. The contours are also filtered on their shape, accepting only contours which can be approximated by 4 points meaning it has rectangular shape.
The tracking over time uses optical flow. A grayscale frame from the current and last iteration is used together with the pixel coordinates of the centers of all the contours found in the previous iteration. Optical flow calculates the pixel coordinates of these old centers in the current frame. If the calculated new position of an old center falls within one of the current contours, the respective contours can be associated to be the same object. If no old center falls within a current contour, a new object is created for that contour. If an old center can not be associated with a new contour for a few iterations, the respective object is deleted. If an object is seen in multiple frames and it can be associated with contours of both red and blue color it is the target marker. The pixel coordinates of the center of the target is used to calculate the direction of the target and read the distance information from the depth frame. If there is a valid distance value for the target, the target marker coordinates are published on the appropriate ROS topic in camera coordinates. Otherwise, if there is no valid distance value a normalized vector pointing towards the target is published on the appropriate ROS topic in camera coordinates.

The biggest challenge proved to be finding the right camera settings and threshold values applied to the HSV image. The color of the smartphone display as perceived by the camera is very dependent on the environment. In bright environments the image tends to be over exposed and therefore the color of the smartphone on the image changes.
To mitigate this effect the camera exposure time and gain are manually set to the minimun possible value. It is still possible to see the screen on the image even in a completely dark room. Therefore, we can be sure that the image is never underexposed to detect the screen. We can do this because we only want to be able to see the screen of the smartphone and do not care about the rest of the image, since the frames from the bottom camera are only used to detect and track the visual marker.

The visual target tracker node outputs the current frame with circles drawn for debugging and visualization purposes. An object which can only be associated with one of the two colors is surrounded by a purple circle as seen in Figure 5.3.

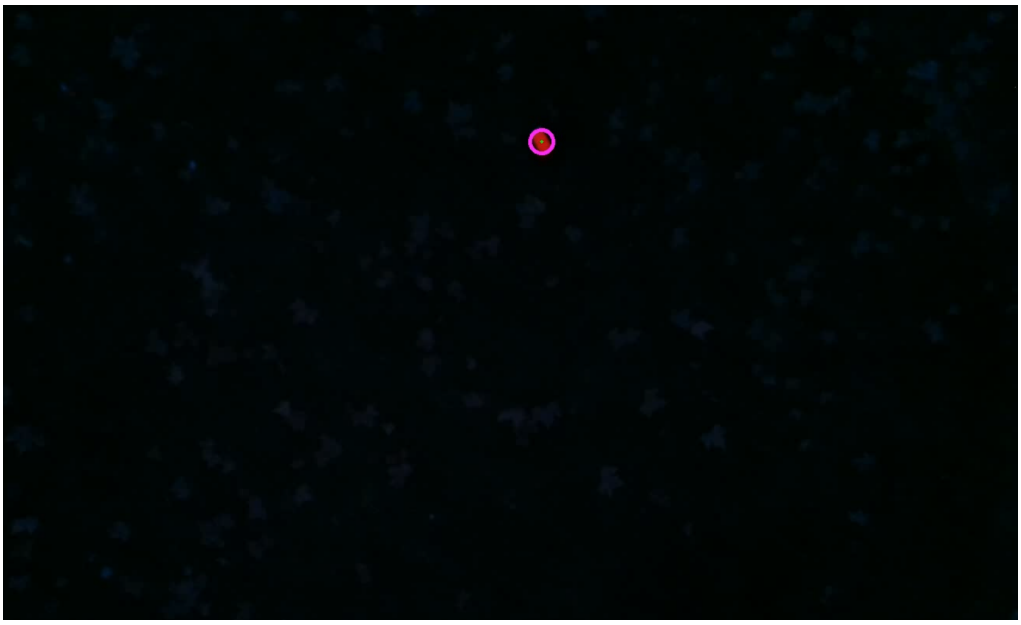


Figure 5.3: Object with one of the two target colors detected.

As soon as the second color is detected on the object it is identified as the target and a yellow circle is drawn around the object as seen in Figure 5.4.

5.5 Trajectory Planner

The trajectory planner node from the work of [2] is used. It is extended to also read depth images from the bottom camera. The transformation from bottom camera to drone frame is applied to the data which is then inserted to the occupancy grid.

5.6 Control Logic

The implementation of [2] is used as a base. It is adapted for our new visual marker tracker. When the location of the target marker is received, it is transformed to the world coordinate frame.

The autonomous drone delivery application is executed in the following steps: The drone takes off and flies to a predefined starting point from where to start scanning for the target. When the drone arrives at this location, it starts scanning the area on a given height above ground until the target marker is detected. The scanning scheme is visualized in Figure 5.5. When the marker is detected the drone centers the marker at the bottom and approaches it from above. When

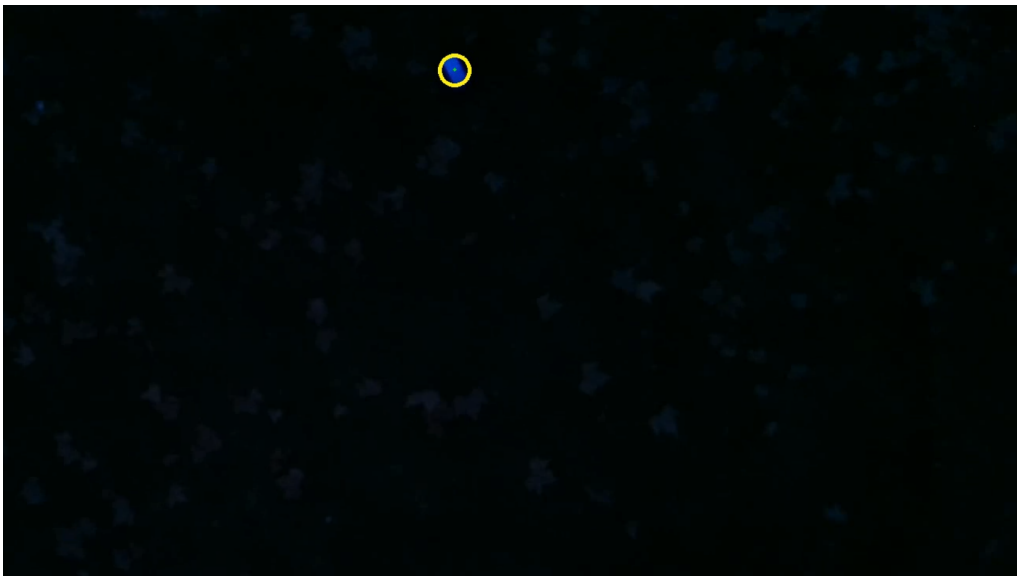


Figure 5.4: Second target color detected on the object. The drone now knows that it is the target marker.

located about 1 meter above the target, the drone signals that it successfully approached the target marker and lands about 1 meter to the side.

5.7 Simulation

The simulation is executed in the Gazebo environment. Figure 5.6 shows an overview of the simulation architecture.
A generic quadcopter model is used and cameras with the appropriate position, orientation and technical specifications are attached to the quadcopter. A model of the drone can be seen in Figure 5.7. The output of these cameras is simulated by Gazebo. Also a model of a smartphone which toggles its color between red and blue, the target marker, is inserted to the simulation based on the dimensions of a Samsung Galaxy S8.
PX4 offers software-in-the-loop (SITL) simulation. All the other ROS nodes are executed as normal and receive their inputs from the simulated components. Therefore it is possible to conveniently simulate all system components on a single Ubuntu computer.

5.8 Test Flights

Test flights in the field are done on a big open field. The field is partially covered by leaves and on some days with snow. Two different kind of flight tests are

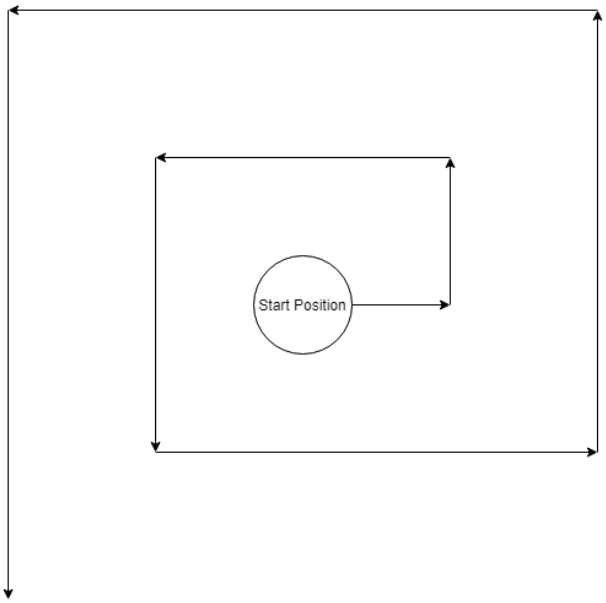


Figure 5.5: The trajectory of the autonomous delivery drone in the x-y plane when scanning the area. The height above ground is kept constant.

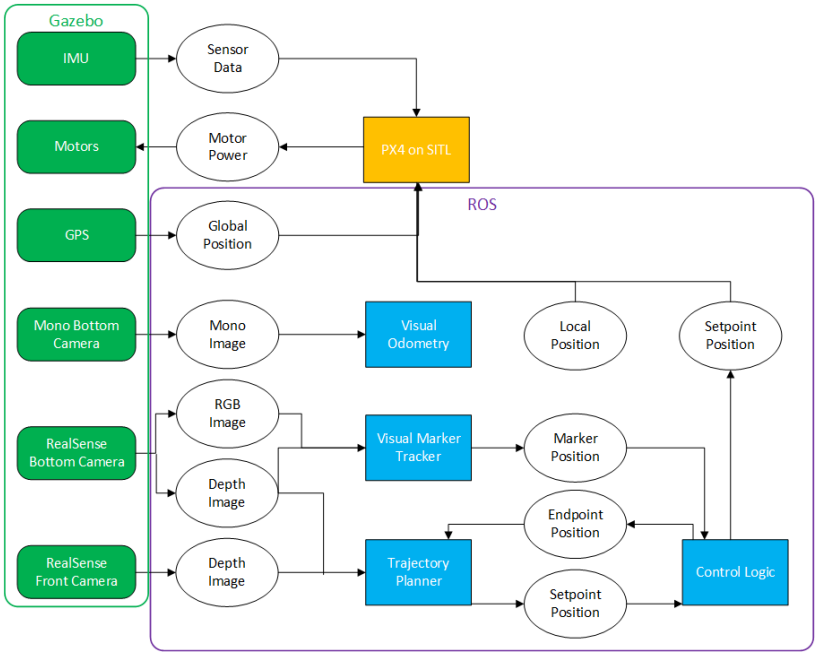


Figure 5.6: System architecture of the autonomous delivery drone in the simulation. The green rounded rectangles represent the simulated hardware components and the blue rectangles represent the ROS software nodes. The components communicate with each other through the data structures described by the ellipsoids.

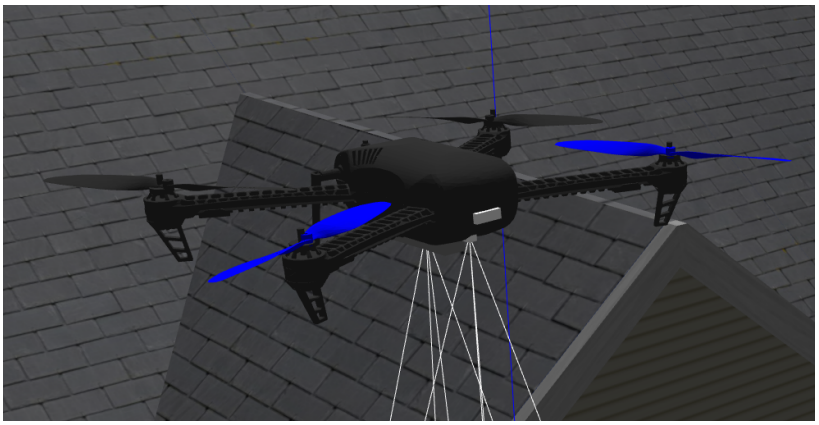


Figure 5.7: Model of the autonomous delivery drone in the simulation with simulated cameras attached.

executed:

- Test the maximum detection distance in different conditions (e.g., snow or no snow, different lighting conditions).
- Demonstrate that the system is working for a given scanning altitude which is appropriate for the given conditions, as determined by the previous tests.

Figure 5.8 shows a test flight. The drone detected the target marker and is currently approaching it.

Figure 5.8: Autonomous delivery drone approaching the target marker in a field test.

Chapter 6

Evaluation

Since there are no standardized evaluation methods for drone delivery applications used in the community, empirical results are the best option. A simulation environment is essential for development and testing of an autonomous delivery drone since field tests are costly in terms of time and also money in case of a crash. But test flights are also executed to demonstrate that the system works.

6.1 Simulation

We implemented a simulation scenario with a few obstacles placed in the world and a smartphone displaying the target marker located on the floor. The autonomous drone is capable of localizing and approaching the marker from a starting position 20 meters above ground.
We can also verify that the trajectory planner includes the distance information into the occupancy grid by looking at the RVIZ visualization.

6.2 Test Flights

Figure 6.1 shows an overview of the field test setup. A laptop is used to start code execution using an ssh connection. The transmitter is used to control the drone in manual mode in case something goes wrong.
The maximum detection distance of the target marker depends on the environment. If the weather is very sunny or the scene is covered in a lot of reflective surfaces (e.g.,snow) the target marker usually can not be detected because in this case the scene is too bright, chaning the perceived color of the target marker. The problem is that the gain is already at the minimum and the RealSense API does not allow to choose an even shorter exposure time. However, in the realsense-viewer, a tool from Intel, it is possible to go below that level. We believe that the application can also work in sunny conditions if the API is updated and a shorter exposure time can be set, or alternatively by using a different camera which allows to set the exposure time adequately short.

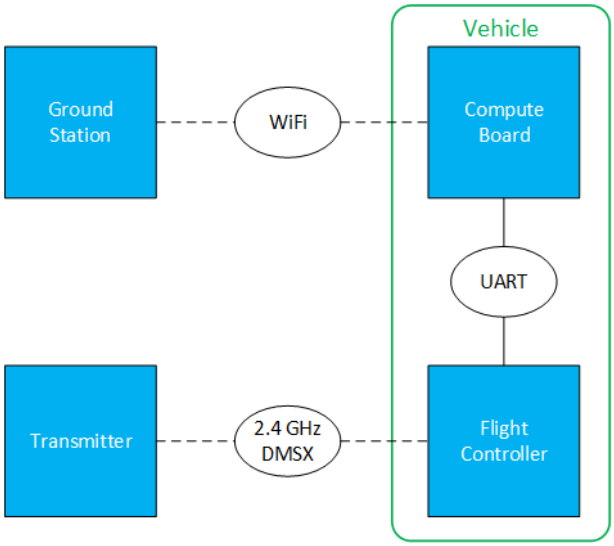


Figure 6.1: The field test setup. The transmitter can be used to control the drone in manual flight mode or the compute board can control the drone through the MAVLink protocol in offboard mode. A groundstation can establish an ssh connection to the compute board to start code execution.

However, without direct sunlight and snow we were able to execute a successful test flight. The drone starts scanning the floor at 10m above ground level and successfully detects and approaches the target marker as seen in Figure 5.8.

6.3 Image Resolution

We evaluated how the maximum detection range and average processing time per frame compare using two different image resolutions of the bottom camera.
The values are only meaningful in relation to each other and do not have an absolute interpretation since they vary depending ond the environment and the processing power of the laptop that this test was executed on.
From Table 6.1 we can see that for an increased detection range by a factor of approximately 1.2 the average processing time is increased by a factor of approximately 2.8. Still it is unclear which resolution is the better, they both have advantages: An increased detection range allows the drone to fly higher and potentially detecting the target marker faster. However, a lower processing time makes the detection more stable as the performance of optical flow improves if the time between two consecutive frames is decreased.
It is not trivial to say what resolution is optimal for the application. Additional extensive testing is needed.

Image Resolution [pixels × pixels]	Maximum Detection Range [m]	Average Processing Time [ns]
1920 × 1080	18 ± 1	50 ± 2
1280 × 720	15 ± 1	18 ± 2

Table 6.1: Comparison of two different bottom camera resolutions considering maximum detection range of the target tracker and average processing time of the target tracker for analyzing a single frame.

Chapter 7

Conclusion And Future Work

Altough today autonomous drone delivery is not really commercially used, this may become reality in a couple of years. Our work provides a further step in the usability of such an autonomous delivery system as it shows a proof of concept for an autonomous drone capable of detecting and approaching a target marker displayed on a smartphone screen. The only equipment a potential client needs is a smartphone which is convenient for the client and very mobile. But before this approach can be commercially used it needs to be improved to work in every environment.
Additional technical challenges that need to be solved include: designing a custom application-specific drone that is efficient and safe, a mechanism to transport and release the payload once the target is reached, as well as appropriate security measurements against,e.g., jamming or malicious clients trying to hijack a delivery. Also the trajectory planner needs to be improved not only to avoid obstacles which it recognizes from the available depth data, but also to always orient the drone in a way that the available depth cameras take measurements in the direction of movement.
Further challenges for managing a delivery fleet include: distribution of base stations, optimal assignment of drones to deliveries.

Bibliography

[1] "Urban population," accessed: 2019-02-07. [Online]. Available: https://data.worldbank.org/indicator/SP.URB.TOTL.IN.ZS

[2] G. Brunner, B. Szebedy, S. Tanner, and R. Wattenhofer, "The urban last mile problem: Autonomous drone delivery to your balcony," vol. abs/1809.08022, 2018.

[3] "Ownership of tablets and smartphones in switzerland." statista, accessed: 2019-02-07. [Online]. Available: https://de.statista.com/statistik/daten/studie/537944/umfrage/besitz-von-smartphone-bzw-tablet-in-der-schweiz/

[4] Y. Nakazawa, H. Makino, K. Nishimori, D. Wakatsuki, and H. Komagata, "Led-tracking and id-estimation for indoor positioning using visible light communication," in *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Oct 2014, pp. 87–94.

[5] T. Nagura, T. Yamazato, M. Katayama, T. Yendo, T. Fujii, and H. Okada, "Tracking an led array transmitter for visible light communications in the driving situation," in *2010 7th International Symposium on Wireless Communication Systems*, Sep. 2010, pp. 765–769.

[6] D. Krakowczyk, J. Wolff, A. Ciobanu, D. Julian Meyer, and C.-E. Hrabia, *Developing a Distributed Drone Delivery System with a Hybrid Behavior Planning System: 41st German Conference on AI, Berlin, Germany, September 24–28, 2018, Proceedings*, 01 2018, pp. 107–114.

[7] D. Mardiansyah and A. H. S. Budi, "UAV vision system for rescue payload delivery," *IOP Conference Series: Materials Science and Engineering*, vol. 384, p. 012005, jul 2018. [Online]. Available: https://doi.org/10.1088%2F1757-899x%2F384%2F1%2F012005

[8] "Amazon prime air." Amazon, accessed: 2019-02-07. [Online]. Available: https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011

[9] "Our approach wing." X, accessed: 2019-02-07. [Online]. Available: https://x.company/projects/wing/

[10] "Ups tests residential delivery via drone launched from atop package car." UPS, accessed: 2019-02-07. [Online]. Available: https://pressroom.ups.com/pressroom/ContentDetailsViewer.page?ConceptType=PressReleases&id=1487687844847-162

[11] "Intel aero ready to fly drone." Intel, accessed: 2019-02-06. [Online]. Available: https://www.intel.de/content/www/de/de/products/drones/aero-ready-to-fly.html

[12] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "Svo: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, April 2017.

[13] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "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)*, Sep. 2017, pp. 215–222.

[14] "Robot operating system," accessed: 2019-02-07. [Online]. Available: http://www.ros.org/

[15] "Realsense d435." Intel, accessed: 2019-02-06. [Online]. Available: https://click.intel.com/intelr-realsensetm-depth-camera-d435.html

[16] "Cgo3+ gimbal mount," accessed: 2019-02-06. [Online]. Available: https://github.com/intel-aero/Documents/blob/master/mechanical_cad/cgo3%2B-gimbal/cgo3%2B-gimbal-mount.STL

[17] "Realsense2 camera ros wrapper," accessed: 2019-02-06. [Online]. Available: http://wiki.ros.org/realsense2_camera

[18] "Open source computer vision library," accessed: 2019-02-07. [Online]. Available: https://opencv.org/