

# Detecting Abnormalities of a Wall via Autonomous Inspection using Quadcopters

By

Youngsang Suh

BS (Seoul National University) 2021

A report submitted in partial satisfaction of the  
Requirements for the degree of

Masters of Science, Plan II

in

Mechanical Engineering

at the

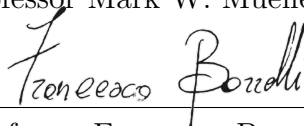
University of California at Berkeley

Committee in Charge:



---

Assistant Professor Mark W. Mueller, Chairman



---

Professor Francesco Borrelli

Fall 2022

Detecting Abnormalities of a Wall via Autonomous Inspection using Quadcopters

Copyright 2022  
by  
Youngsang Suh

## Abstract

Detecting Abnormalities of a Wall via Autonomous Inspection using Quadcopters

by

Youngsang Suh

Masters of Science, Plan II in Mechanical Engineering

University of California, Berkeley

Assistant Professor Mark W. Mueller, Chairman

Autonomous inspection can benefit the industry by detecting structural damages or abnormalities of a wall, building, or even a bridge without intense resources. However, limited flight time and collision from unknown objects prevent quadcopters from being widely used in building inspections. This report presents how current problems in autonomous inspection can be resolved by demonstrating the abnormality detection of a wall with quadcopters.

This report is composed of three parts. In the first part, we define the structural damage of a wall. Then, a problem statement that can be validated with quadcopters is constructed. As quadcopters have limited flight time and computation power, we introduce a realistic scenario. Instead of conducting autonomous inspection without a 3D model, we assume to have a given model, and detect differences between the model and collected depth data. Furthermore, for safe autonomy, we implement self-localization with a tracking camera, and set up a goal to achieve collision avoidance in case of unknown obstacles.

In the next part, we focus on methodologies to solve the problem statement. From our threshold (structural damage) to detect abnormalities, we justify some measures such as distance from the target and location of inspection points. Then, it is followed by details of how safe autonomy is achieved. The methodologies are based on the assumptions we have made and the technical specifications of our sensors. A lidar and a tracking camera is used to capture depth information and track its pose, respectively. As we extract abnormalities offline, we further elaborate on post-processing of collected data.

Finally, we demonstrate our approach via simulation and real flight experiments. The simulation is based on a **Unity** simulator with a customized environment. In the environment, we generate an abnormality that could be captured. Since autonomous inspection also has to avoid possible collisions, we test safe inspection by occluding one of the inspection points with an obstacle. Real flight experiments are also conducted to confirm our proposed method.

To my family.

# Contents

<b>Contents</b>	<b>ii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Problem statement</b>	<b>3</b>
2.1 Detect abnormalities . . . . .	4
2.2 Achieve safe autonomy . . . . .	4
<b>3 Methodologies</b>	<b>5</b>
3.1 Flowchart . . . . .	5
3.2 Inspection point computation . . . . .	7
3.3 Safe autonomy . . . . .	8
3.4 Post-processing depth data . . . . .	9
<b>4 Experimental results</b>	<b>11</b>
4.1 Simulation . . . . .	11
4.2 Real flight experiment . . . . .	13
<b>5 Conclusion and future work</b>	<b>17</b>
5.1 Conclusion . . . . .	17
5.2 Future work . . . . .	17
<b>Bibliography</b>	<b>19</b>

## Acknowledgments

I would like to first and foremost thank my parents for their love and support not only during my time at UC Berkeley, but throughout my entire academic career. I could not have undertaken this journey without your unconditional encouragement. For this I will be forever grateful.

I am particularly indebted to my M.S. advisor Professor Mark Mueller, who is a fantastic mentor with immense knowledge, brilliant ideas, and intense enthusiasm. His scholarly advice and scientific approach have helped me in all the time of research and writing of this report. I sincerely appreciate your careful and meticulous guidance in my study.

I genuinely thank Professor Mark Mueller and Professor Francesco Borrelli for serving on my M.S. degree committee.

To all of my labmates and friends at UC Berkeley, thank you for making my time here so memorable and fun. Your help and friendship has been a constant source of encouragement that I will not forget.

Finally, the experimental testbed at the HiPeRLab is the result of the contributions of many people, a full list of which can be found at [hiperlab.berkeley.edu/members/](http://hiperlab.berkeley.edu/members/).

# Chapter 1

## Introduction

A quadcopter is an aerial vehicle propelled by four motors, an example of which is shown in Fig. 1.1. As a result of its simple design and high agility, a quadcopter is useful for a wide range of applications such as aerial imaging [1], transportation [2], crowd monitoring [3], and inspection [4].



Figure 1.1: Quadcopter

Concurrent with the advances in other fields, the number of robotics applications for infrastructures is rapidly increasing [4–12]. Infrastructure inspection, 3D point cloud modeling, mapping, and remote demolition are the examples. Here, we particularly restrict our attention to autonomous inspection.

Inspection process is necessary for every building since failure to analyze the current state would lead to a catastrophic disaster, including building collapse. Not even human impacts but also unexpected events like earthquakes can damage buildings. Thus, professional engineers should inspect buildings often to check if they qualify such criteria for longer use.

Inspectors detect abnormalities such as cracks or structural deformations for building maintenance. However, current inspection methodologies involve use of climbing operators, who, by means of ropes, hang from the structure and perform the measurements required by the inspectors to evaluate its current state. Hence, current building inspection methodologies are time-consuming and resource intensive activities as they require heavy involvement of specifically trained personnel and other high cost safety machinery.

A method that automates the inspection process without climbing operators enables to reduce manpower and expenses, also creating safer environments for workers. Noticing this huge potential, there have been several studies using quadcopters to inspect/monitor infrastructures. Since GPS signal is unreliable in urban areas, localization with other sensors namely, tracking sensor and depth sensor are required. Despite the presence of a human operator, a vehicle with a Skybotix/ASL visual-inertial sensor has been used to collect depth data of a building [8, 13]. Sensor fusion with cameras and a robotic total station has been implemented to a vehicle for localization in [14]. However, such approaches involve human inputs and does not introduce real infrastructure inspection. Several studies demonstrating autonomous inspection have also been presented. Researchers have demonstrated autonomous inspection of a wind turbine by detecting and tracking objects [9, 10]. Similarly, autonomous inspection of a transmission line has been shown via a quadcopter [11]. Also, autonomous inspection of a viaduct has been achieved by maintaining a certain distance from the object in front of a quadcopter [12]. Yet, they usually assume no unknown obstacles and have hard constraints on a target, difficult to implement to other types of inspection subjects.

In this report, we propose an approach to monitor/inspect a wall via quadcopters. Since infrastructures consist of walls, the proposed method is scalable. We here develop an abnormality detection framework with quadcopters through autonomous flight assisted by a real-time collision detecting motion planner and offline data processing. A quadcopter needs extensive flight time to self-determine appropriate waypoints without a given model since inspection should address a full coverage of a target. Thus, it justifies the necessity of a given model. After safe autonomous inspection by avoiding possible collisions and collecting data of a target, we create a real scene based on the data and compare it with the given model. Abnormalities are detected during the post-processing. The proposed method is implemented in simulation and real flight experiments for validation.

The rest of the paper is constructed as follows. Chapter 2 introduces our problem statement that is handled throughout this report. Chapter 3 describes the overview and detailed methods of our approach. Chapter 4 illustrates the simulation and real flight experimental results. Finally, Chapter 5 presents our conclusions and future work.



## Chapter 2

### Problem statement

Without a given model, a quadcopter with lack of computation power necessitates extensive flight time to address full coverage during the flight. Hence, in this report, we assume a given model, and pursue to extract abnormalities of a wall based on the given model and collected depth data.

In this chapter, we propose two main goals to conduct autonomous inspection of a wall. Before that, let us first define the term **structural damage**. It is called a structural damage when the crack width is greater than 25 mm [15]. In [15], building assessors analyzed 130 properties and identified that a structural damage (crack width greater than 25 mm) requires a major repair job, involving partial or complete rebuilding. As it is a crucial flaw that inspectors have to mark, the proposed autonomous inspection framework is designed to detect a structural damage.

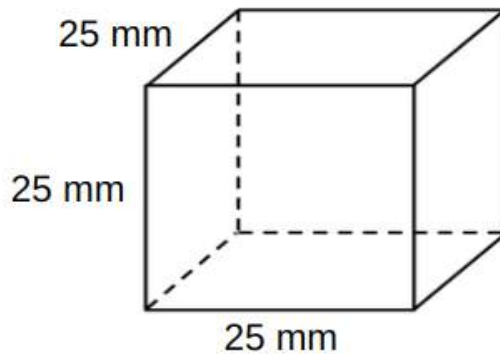


Figure 2.1: Our goal is to detect an abnormality that has a scale greater than a cube with each edge 25 mm

## 2.1 Detect abnormalities

First goal is to detect abnormalities that go beyond the threshold (25 mm). However, despite an anomaly on a wall, it is impossible to conclude the wall is problematic unless the scale of the anomaly is greater than model resolution. Furthermore, a sensor collecting depth data is not perfect. Due to the irreducible errors, the threshold to detect abnormalities is set to be at least greater than  $0.5 \cdot \text{model resolution} + \text{sensor accuracy}$ .

$$\text{threshold} \geq 0.5 \cdot \text{model resolution} + \text{sensor accuracy} \quad (2.1)$$

$$\text{abnormality} \geq \text{threshold} \quad (2.2)$$

Regarding the 25 mm upper limit of threshold, we create a minimum guideline. Sensor accuracy is a physical limitation whereas, model resolution is a parameter we can adjust. In this report, we select Intel Realsense Lidar Camera L515 as a depth sensor. L515 fits well with quadcopters as it is light and consumes low power. It also outputs a dense depth image ( $640 \times 480$  pixels) and has high accuracy ( $< 15$  mm through 9 m distance), outperforming the stereo depth cameras usually used in robotics area [16–18]. The sensor depth accuracy is precise enough for the problem statement and therefore, model resolution is set to be better than 20 mm.

## 2.2 Achieve safe autonomy

Second goal is to achieve safe autonomy. In this report, we consider an environment size of an office or a house. Regarding the limitation of flight time (less than 15 min), we don't tackle such environments that have scale similar to the Golden Gate Bridge. Not only we need accuracy better than of GPS localization, but also GPS signal is not reliable in urban areas. Thus, self-localization via additional sensor such as tracking camera is required.

In addition, avoiding possible collisions from unknown obstacles is essential. There is no guarantee that a given model remains the same as time pasts. For example, residents might leave a flower pot on their balcony or a window frame might be deformed due to an earthquake. As a result, capability to recognize collisions from obstacles is required. Also, when a quadcopter realizes a possible collision, it is necessary to replan the trajectories with inspection point candidates. In this report, we present a framework of trajectory planning by implementing a recent planner named RAPPIDS (rectangular pyramid partitioning using integrated depth sensors) [19]. Detailed description of RAPPIDS is given at Chapter 3.3.

# Chapter 3

## Methodologies

In this chapter, prior to fundamental methodologies, a flowchart is brought out for better comprehension. Then, we introduce the main context. It is separated into three items.

First, we describe how inspection points are determined offline. Assuming a given 3D model, we have enough computation power to compute inspection points based on some heuristics. Note that the inspection points should be close enough to a target to capture rich information. Details are discussed at Chapter 3.2.

Next, we explain how safe autonomy is achieved. A tracking camera attached to a quadcopter enables to localize itself without use of GPS signals. A lidar camera enables a quadcopter to determine collisions with RAPPIDS planner. When a vehicle detects a collision to arrive at the offline-computed inspection point, it samples inspection point candidates. Replanning occurs until the planner returns a collision-free trajectory.

Last, details of post-processing collected depth and estimation data are given. Combining depth and state estimation, we generate a real scene of a target in order to extract abnormalities based on a given model. We use a heuristic that overlapping areas should have matching depth values. This inference mitigates sensor drifts and estimation errors.

### 3.1 Flowchart

Fig. 3.1 shows the overview of the proposed method. After offline computation of inspection points, a quadcopter saves an ordered array of inspection points in its memory. After it takes off, a tracking camera T265 (discussed later at Chapter 3.3) outputs state estimate values. The quadcopter then tries to reach the inspection points in sequence. To arrive at the next inspection point, it turns yaw angle heading towards the inspection point. Using depth and state estimation data, RAPPIDS planner outputs a collision-free trajectory. If the planner fails to return a collision-free trajectory, the vehicle runs replanning process with inspection point candidates. As shown in Fig. 3.1, after computing inspection point candidates, the quadcopter turns yaw angle to each of them one by one until RAPPIDS planner determines a collision-free trajectory. Tracking collision-free trajectory enables the quadcopter to arrive

at the inspection point or inspection point candidate. At the destination, it turns to a predetermined yaw angle, normally perpendicular to a target wall to collect depth data. Aforementioned procedure repeats until the vehicle visits the last inspection point.

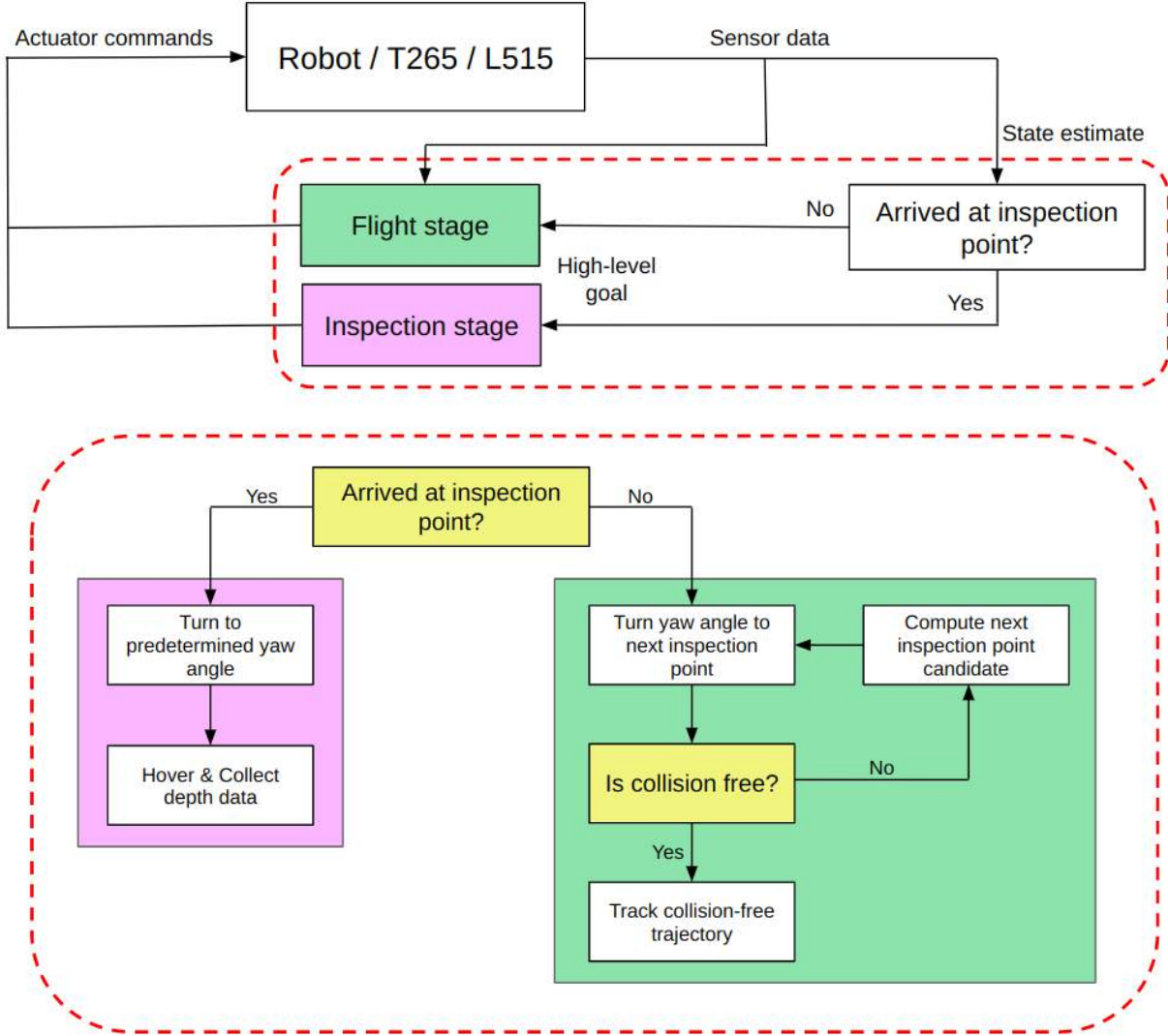


Figure 3.1: Flowchart of the proposed method. Top: Overall data and decision flow of our implementation. Bottom: Detailed view of **Flight stage** and **Inspection stage**. (Note that the red box region at the bottom represents the same thing as the red box region at the top.)

## 3.2 Inspection point computation

In this section, we explain how the inspection points are computed. The process is conducted offline based on a given model. From the abnormality threshold defined at Chapter 2, we choose a distance from the target. As mentioned earlier, in this report, we use L515 lidar camera. Its FOV (field of view) is  $70^\circ \times 55^\circ$ , meaning each width and height resolution is 2 mm at 1 m distance. Considering the abnormality we are to detect, it is sufficient to discretize a 3D space in 1 m grid. There are two benefits from discretizing the space in 1 m grid. L515 FOV overlaps at adjacent inspection points. Thus, we gather consecutive depth data without losing any information of the target. Furthermore, we can mitigate state estimation error by making overlapped region depth values to be the same.

From the reasoning above, a quadcopter collects depth data in 1 m distance. Fig. 3.2 describes the inspection points of a wall. The arrow shown at the figure illustrates a sequence that the vehicle visits, numbering from 1. In this report, we follow the presented path. Here, we haven't discussed the optimality of a path. Note that we further have possibility to improve the inspection process.

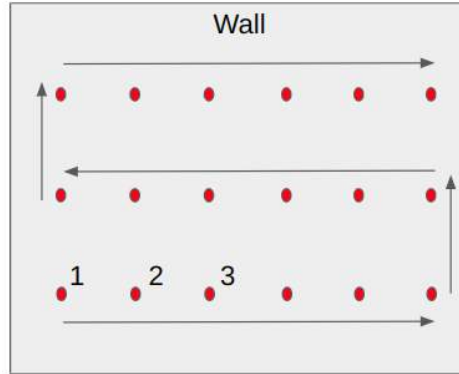


Figure 3.2: Inspection points of a wall. Inspection points are located at 1 m distance from the target and 1 m apart each other.

Due to presence of unknown obstacles, a collision might occur to arrive at precomputed inspection point. Thus, replanning is necessary for safe autonomous inspection. We compute inspection point candidates to successfully plan a collision-free trajectory. If a given point turns out to be compromised, a quadcopter checks out for inspection point candidates. They are sampled as follows.

Here, we use one heuristic. If the planner fails, we assume that a minimum collision ball centered at the inspection point is dangerous. The minimum collision distance is normally greater than the physical radius of a vehicle. To avoid passing the minimum collision ball, candidate points are selected outside the tangential line shown in Fig. 3.3, which is the area illustrated as the [reasonable region](#). Note that there is no strict guarantee that inspection point candidates are going to be safe. Therefore, we sample candidate points until a safe

trajectory is captured. For simplicity, candidate points are sampled on a wall’s perpendicular line that crosses the original inspection point.

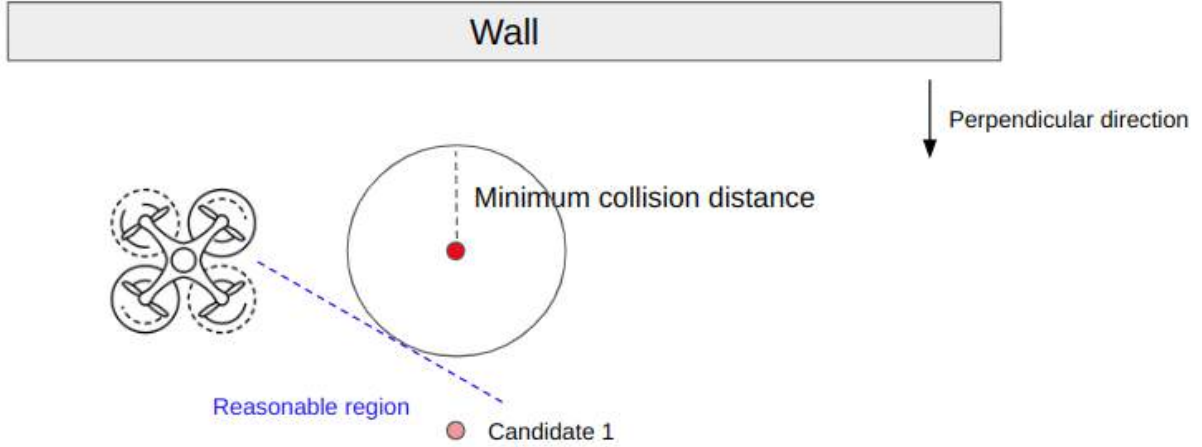


Figure 3.3: Inspection point candidate.

### 3.3 Safe autonomy

To achieve one of the goals in the problem statement, we introduce how state estimation is achieved. Considering the scale of our target environment, state estimation should be done without GPS signals. Hence, we use a tracking camera for localization. It is assisted by a visual-inertial odometry algorithm. And a vehicle hovers and completes missions via a lower-level controller receiving state estimation values.

Also, due to unknown obstacles, a vehicle should be able to detect collisions in real-time, which is essential for autonomous inspection. We attach a lidar camera to implement RAP-PIDS planner that has high computational efficiency even in high-speed collision avoidance flights. It receives a single depth image and generates collision-free flight trajectories. The planner uses motion primitives and then goes through a series of checks to find a trajectory that is minimum-cost, input-feasible, velocity-admissible, and collision-free, as shown in Algorithm 1 [20]. Note that we impose velocity constraints to prevent tracking loss in a tracking camera. The motion primitives are as below.

$$s(t) = \frac{\alpha}{120}t^5 + \frac{\beta}{24}t^4 + \frac{\gamma}{6}t^3 + \frac{\ddot{s}(0)}{2}t^2 + \dot{s}(0)t + s(0), t \in [0 \ T]$$

where  $T$  is the trajectory duration,  $s(0)$ ,  $\dot{s}(0)$ , and  $\ddot{s}(0)$  are the initial position, velocity, and acceleration of the vehicle at the time when the trajectory starts, and  $\alpha$ ,  $\beta$  and  $\gamma$  are

coefficients such that  $s(T) = s_T$ , and  $\dot{s}(T) = \ddot{s}(T) = 0$ . We fix an inspection point to be the goal point. Trajectory duration is uniformly sampled in a certain range, and the planner finds a trajectory that passes every check. Detailed information are further described at [19] and [20].

---

**Algorithm 1** Trajectory constraint checks

---

```

1: procedure CONSTRAINTSCHECK
2:   if lower cost than known then
3:     if input feasibility then
4:       if velocity admissibility then
5:         if collision_free then
6:           trajectory_status  $\leftarrow$  collision_free
7:         else
8:           trajectory_status  $\leftarrow$  in_collision
9:         end if
10:      else
11:        trajectory_status  $\leftarrow$  velocity_inadmissible
12:      end if
13:    else
14:      trajectory_status  $\leftarrow$  input_infeasible
15:    end if
16:  else
17:    trajectory_status  $\leftarrow$  higher_cost
18:  end if
19: end procedure

```

---

### 3.4 Post-processing depth data

After safe autonomous inspection has finished, we conduct post-processing of depth data. Detecting abnormalities is not necessarily required to be from onboard computation. With collected data, post-processing enables to take burden off of quadcopters since their onboard computers usually lack of computation power.

At post-processing, we construct a real scene from the data that had been collected, and extract abnormalities by comparing with a given model. To project depth images to a real scene, state estimation and depth data are recorded during the inspection stage. State estimation informs the position and rotation angle that a single depth image is originated from. Combining with depth values, we scale depth images to a real scene. Refer to the equation below for details.

$$x_{real} = \text{Rot}\left(\frac{x_{pixel} - c_x}{r_x} z_{pixel} r_z, att\right) + pos_x$$

$$\begin{aligned}
y_{real} &= \text{Rot}\left(\frac{y_{pixel} - c_y}{r_y} z_{pixel} r_z, att\right) + pos_y \\
z_{real} &= \text{Rot}(z_{pixel} r_z, att) + pos_z \\
r_x &= \frac{\text{image width}}{2} \frac{1}{\tan(\frac{\text{FoV}_x}{2})} \\
r_y &= \frac{\text{image height}}{2} \frac{1}{\tan(\frac{\text{FoV}_y}{2})} \\
r_z &= \text{depth resolution}
\end{aligned}$$

where  $x_{pixel}$ ,  $y_{pixel}$ , and  $z_{pixel}$  is a width, height, and depth pixel value (in a depth camera frame) respectively.  $x_{real}$ ,  $y_{real}$ , and  $z_{real}$  are real coordinate values of a pixel in a global map frame. Notice the  $\text{Rot}(-, att)$  function in the equations. It transforms the depth image from a depth camera frame to a global map frame.  $att$  and  $pos$  come from the collected state estimation data.  $r_x$ ,  $r_y$ , and  $r_z$  are the ratios to convert an image to a real scene. The image width and image height stand for width and height of an image in pixels.  $\text{FoV}_x$  and  $\text{FoV}_y$  is a field of view angle in image width and height direction, respectively. The depth resolution is the smallest depth difference that can be detected by a depth camera in meters.

The equations above enable to compare inspection results with a model. However, sensors are not perfect. From our experiments, it turns out that particularly perpendicular direction from a wall drifts in state estimation. The tracking camera is attached to the back of the vehicle thus, tracking features are relatively further. Thankfully, the drift can be mitigated by using a heuristic. There are overlapped regions from each adjacent inspection points pair. Notice that overlapped regions should have the same depth values. Thus, we correct the estimation drift. Although, the heuristic is strongly correlated to the drift at the first inspection point, it is relatively small at the beginning. Hence, we expect that the proposed heuristic generates a smooth scene with negligible errors.



# Chapter 4

## Experimental results

This chapter validates the proposed methodologies. By demonstrating in simulation and conducting real flight experiments, we check that it is possible to detect abnormalities that are bigger than the threshold, even avoiding unexpected obstacles.

### 4.1 Simulation

#### Environment setup

We have built an environment in a **Unity Simulator**, which is presented in Fig. 4.1. The black box region is the target wall. Its dimension is  $6.5 \text{ m} \times 4 \text{ m}$  (width  $\times$  height). We assume that the original target wall is a simple flat wall without any additional feature. To validate the proposed inspection process, we define an abnormality in the original model. The red box region represents the abnormality we are to detect. It is a size of  $100 \text{ mm} \times 100 \text{ mm} \times 400 \text{ mm}$ . Since its scale is greater than  $25 \text{ mm}$ , according to our methodologies, we should be able to detect it. In addition, we add an obstacle to check if our methodologies can avoid collisions. The blue box region is the obstacle with each edge  $1 \text{ m}$  to occlude one of the inspection points.

A quadcopter is spawned at  $(0, 0, 0)$  in a global map frame. The global map frame is presented in Fig 4.1. A tracking camera and a depth sensor are also simulated inside the environment. Tracking and depth cameras are both set to initialize at  $(0, 0, 0)$ , which is the same place where the quadcopter is spawned. We place the target wall to be at the distance of  $2.3 \text{ m}$  from the quadcopter origin in  $-z$  direction. Hence, depth images projected to a real scene should be at  $-2.3 \text{ m}$  ( $z$  axis) if there is no error.

We follow the inspection points given as Fig. 3.2. Bottom 6 figures in Fig. 4.1 illustrates a quadcopter located at the inspection point 1 to 6. We notice that the vehicle is close to the obstacle from inspection point 3 to 6 thus, we expect our proposed method to autonomously avoid collisions by replanning for these cases.

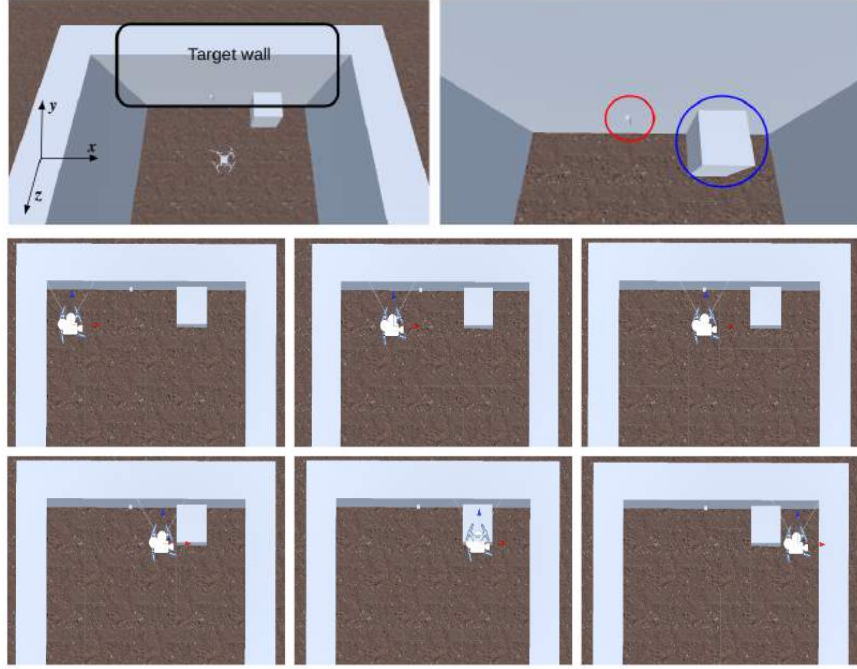


Figure 4.1: Top: Target wall with an abnormality (circled red) and an unknown obstacle (circled blue) that blocks one of the inspection points. Bottom: Quadcopter located at inspection point 1  $\sim$  6.

### Abnormality detection

Using the inspection points computed from the model, we ran a simulation to verify the proposed method. Consequently, the vehicle successfully avoids collisions and collects depth data from the target wall. Conducting post-processing of collected depth data combining with state estimation, we capture the abnormalities of the target. Following Fig. 4.2 shows the results. It displays a whole real scene view of depth images that contain abnormalities.

As the environment is a simulation setup, we have an accurate model. However, due to errors in the sensors, generated real scene has slight errors in each  $x$ ,  $y$ , and  $z$  axis. Thanks to the heuristic discussed at Chapter 3.4, we generated a smooth real scene. The real scene is projected at  $-2.302$  m in  $z$  axis. Despite 2 mm error in  $z$  axis, the sensor error was below the threshold. By running a script that extracts abnormalities, we notice that the depth data from inspection point 3  $\sim$  7 has the abnormalities. Fig. 4.3 describes the exploded view at each inspection point. Observing the colormap, we validate that an abnormality resembling a crack and one unknown obstacle are detected.

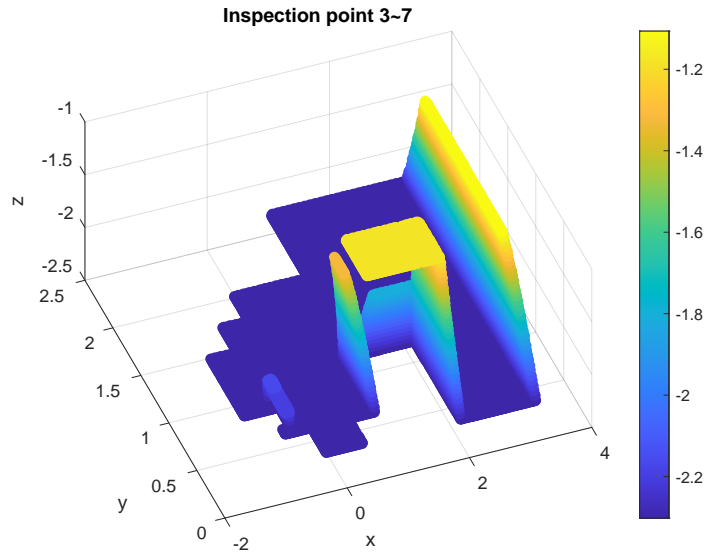


Figure 4.2: Simulation result (full view)

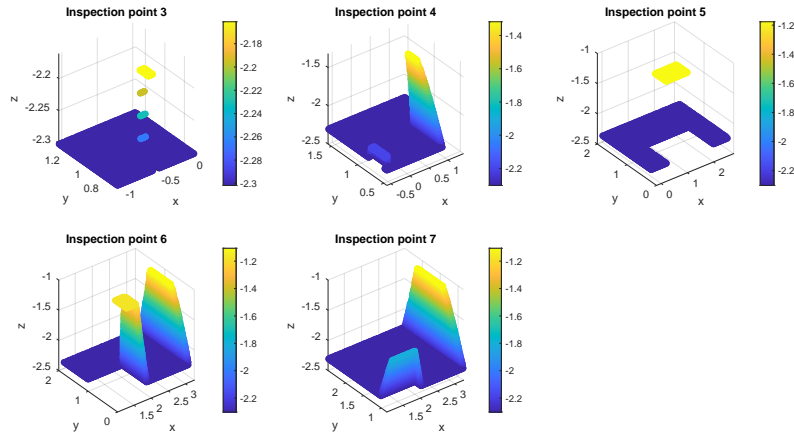


Figure 4.3: Simulation result (exploded view)

## 4.2 Real flight experiment

### Experimental setup

Here, we demonstrate the use of the proposed method on an experimental quadcopter. Basic information of the hardware setup is as follows. A quadcopter vehicle used in the experiments is shown in Fig. 4.4. An Intel Realsense Lidar Camera L515 (at front view) is used to acquire depth images, and an Intel Realsense Tracking Camera T265 (at back view) is used to obtain

state estimates of the vehicle. The inspection algorithm is run on a Qualcomm RB5 (at back view), which sends desired thrust and angular velocity commands to a Crazyflie 2.0 flight controller.

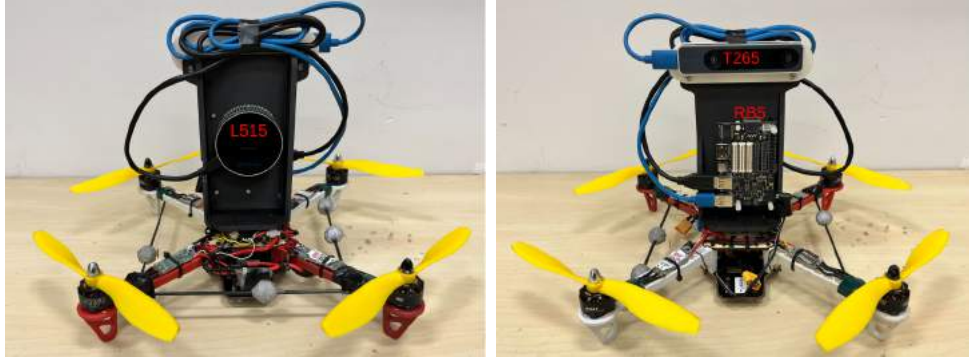


Figure 4.4: Vehicle used in experiments. Left: front view (L515). Right: back view (RB5 and T265).

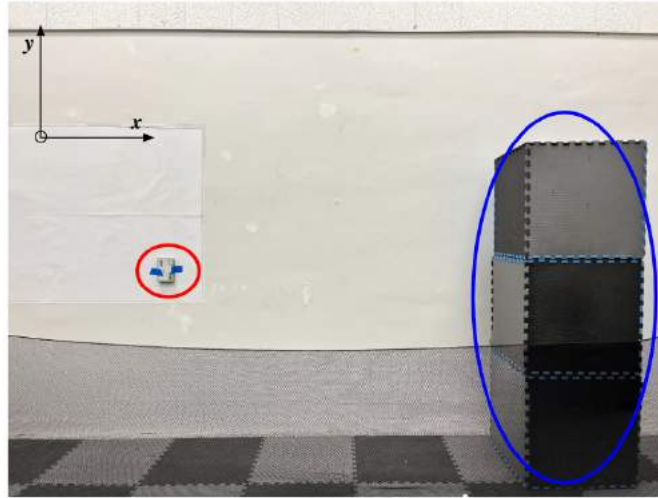


Figure 4.5: Target wall with one abnormality (red circle) and an unknown obstacle (blue circle) that blocks an inspection point.

We conduct autonomous inspection based on a real wall. The target wall with a global map frame is shown in Fig. 4.5. The width is 6.480 m, and the vehicle is initialized at 2.65 m distance from the wall. For simplicity, its initial  $x$  position is same as the center of the wall. The target wall dimension is measured by Bosch GLM 35, a laser distance meter that has an accuracy of 1.5 mm. Thus, we have a model resolution better than 20 mm, the minimum guideline discussed at Chapter 2. We attach an abnormality (a paper box) a size of 107.95 mm  $\times$  31.75 mm  $\times$  158.75 mm (width  $\times$  depth  $\times$  height). As the size is smaller than the

simulation setting, success in real flight experiments better validates the proposed method. To test if safe autonomous inspection is achieved, we also install an unknown obstacle similar to the simulation setting. A black net in the figure is to prevent damages in failures. The inspection points are computed to be above the safety net thus, the net does not affect the inspection process.

## Abnormality detection

We ran real flight experiments to check if it is possible to detect abnormalities that are greater than 25 mm cube. Following Fig. 4.6 shows the results. It displays a real scene view of depth images that contain abnormalities. Note that it has noisier depth values compared to the simulation results.

We find that there is a noticeable color difference (abnormality) on the left part of the wall. As we extract the abnormalities to have greater difference than 25 mm from the model, we detected the paper box that is attached to the wall. Despite its small depth (less than 50 mm), the proposed method is capable of detecting the abnormality defined in the problem statement.

Fig. 4.7 illustrates the exploded view of Fig. 4.6 for each inspection point that has abnormalities. Every precomputed inspection point has the paper box or the obstacle detected since they are bigger than the threshold. However, due to the noise and state estimation error, points other than the abnormalities are also extracted. Attitude estimation from the tracking camera had relatively greater error than position estimation and depth data. Unfortunately, the attitude error caused the points to diverge from the model at the areas far from the image center. Meaning that several areas that are supposed to be normal are also extracted. Therefore, it is safer to have a safety margin in the threshold regarding the state estimation error. However, despite the state estimation error, the abnormalities are successfully detected.

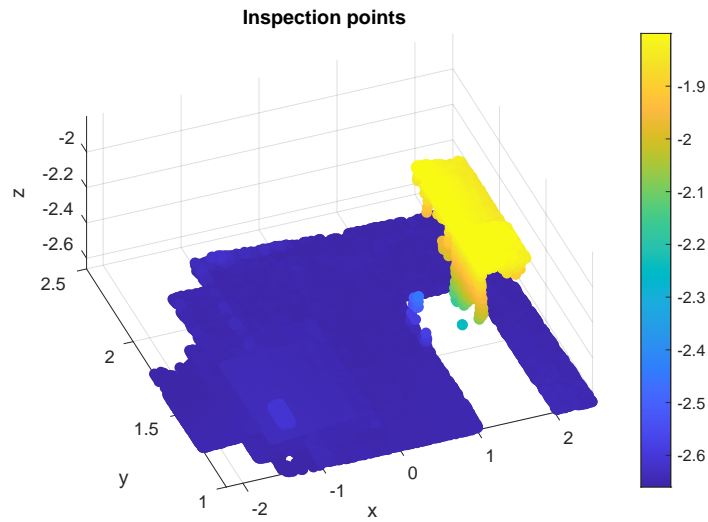


Figure 4.6: Real flight experiment result (full view)

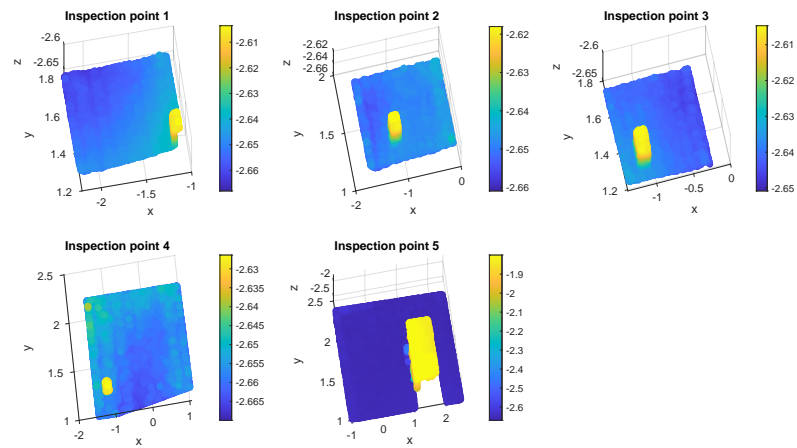


Figure 4.7: Real flight experiment result (exploded view)

# Chapter 5

## Conclusion and future work

### 5.1 Conclusion

This report presents a problem statement to conduct safe autonomous inspection of a wall to mitigate resource intensive works. In Chapter 2 and 3, we propose a solvable scenario and methodologies to achieve safe autonomous inspection with a quadcopter. Based on the designed parameters, we devised a realistic problem statement that can be implemented to a quadcopter. In Chapter 4, we validate the proposed method by demonstrating in simulation and conducting real flight experiments. We consider environments with an abnormality greater than the threshold and an unknown obstacle that is to avoid. Extensive simulation and experiments are conducted to validate the method. Both in simulation and real implementation, our method has shown to successfully inspect a target wall without any collision. Combining collected state estimation and depth data, we generate a real scene in order to compare with a given model. Despite errors in state estimation, we successfully detected abnormalities and obstacles. Also, without any assumption on a target, a quadcopter is able to autonomously inspect the target. A tracking camera and a depth sensor are the only sensors attached to the vehicle, making the implementation simple. This strategy reduces the mission complexity therefore, autonomous inspection is possible for a quadcopter that can carry the two sensors.

### 5.2 Future work

Currently, the proposed method utilizes depth images. While depth images only give us geometric information, RGB data has richer information that helps us better extract features of a target. As L515 depth sensor also outputs RGB images, our future plan is to utilize RGB data to detect abnormalities.

Furthermore, we notice that our experimental vehicle is weak to wind disturbances. As the inspection process requires close distance to a target, the vehicle should resist wind disturbances for better data quality. Also, high vibration leads to severe errors in state

estimation, making post-processing much difficult. We would like to implement the proposed method to a quadcopter that better endures wind disturbances.



# Bibliography

- [1] Youngjun Choi et al. “Multi-UAV trajectory optimization utilizing a NURBS-based terrain model for an aerial imaging mission”. In: *Journal of Intelligent & Robotic Systems* 97.1 (2020), pp. 141–154.
- [2] Andrea Tagliabue et al. “Robust collaborative object transportation using multiple MAVs”. In: *The International Journal of Robotics Research* 38.9 (2019), pp. 1020–1044.
- [3] Naser Hossein Motlagh, Miloud Bagaa, and Tarik Taleb. “UAV-based IoT platform: A crowd surveillance use case”. In: *IEEE Communications Magazine* 55.2 (2017), pp. 128–134.
- [4] Yunpeng Wu et al. “A UAV-based visual inspection method for rail surface defects”. In: *Applied sciences* 8.7 (2018), p. 1028.
- [5] Guglielmo Carra et al. “Robotics in the construction industry: State of the art and future opportunities”. In: *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*. Vol. 35. IAARC Publications. 2018, pp. 1–8.
- [6] Ronny Salim Lim, Hung Manh La, and Weihua Sheng. “A robotic crack inspection and mapping system for bridge deck maintenance”. In: *IEEE Transactions on Automation Science and Engineering* 11.2 (2014), pp. 367–378.
- [7] Inkyu Sa and Peter Corke. “Vertical infrastructure inspection using a quadcopter and shared autonomy control”. In: *Field and service robotics*. Springer. 2014, pp. 219–232.
- [8] Sammy Omari et al. “Visual industrial inspection using aerial robots”. In: *Proceedings of the 2014 3rd international conference on applied robotics for the power industry*. IEEE. 2014, pp. 1–5.
- [9] Marko Car et al. “Autonomous wind-turbine blade inspection using LiDAR-equipped unmanned aerial vehicle”. In: *IEEE Access* 8 (2020), pp. 131380–131387.
- [10] Martin Stokkeland, Kristian Klausen, and Tor A Johansen. “Autonomous visual navigation of unmanned aerial vehicle for wind turbine inspection”. In: *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2015, pp. 998–1007.

- [11] Tong He, Yihui Zeng, and Zhuangli Hu. “Research of multi-rotor UAVs detailed autonomous inspection technology of transmission lines based on route planning”. In: *IEEE Access* 7 (2019), pp. 114955–114965.
- [12] Rafael Caballero et al. “Aerial Robotic Solution for Detailed Inspection of Viaducts”. In: *Applied Sciences* 11.18 (2021), p. 8404.
- [13] Janosch Nikolic et al. “A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM”. In: *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2014, pp. 431–437.
- [14] Diego Benjumea et al. “Localization System for Lightweight Unmanned Aerial Vehicles in Inspection Tasks”. In: *Sensors* 21.17 (2021), p. 5937.
- [15] R Driscoll. *Assessment of damage in low-rise buildings, with particular reference to progressive foundation movement. Revised 1995*. BRE Electronic Publications, 1995.
- [16] *Intel RealSense LiDAR Camera L515 Datasheet*. Revision 003. Intel. Jan. 2021.
- [17] *Intel RealSense Product Family D400 Series Datasheet*. Revision 009. Intel. June 2020.
- [18] Salman Hussain, Akin Tatoglu, and Kiwon Sohn. “Mechanical Upgrade and Gait Development of Re-Sizable Quadruped, HARQ”. In: *ASME International Mechanical Engineering Congress and Exposition*. Vol. 85611. American Society of Mechanical Engineers. 2021, V07AT07A015.
- [19] Nathan Bucki, Junseok Lee, and Mark W Mueller. “Rectangular pyramid partitioning using integrated depth sensors (rappids): A fast planner for multicopter navigation”. In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4626–4633.
- [20] Junseok Lee et al. “Autonomous flight through cluttered outdoor environments using a memoryless planner”. In: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2021, pp. 1131–1138.