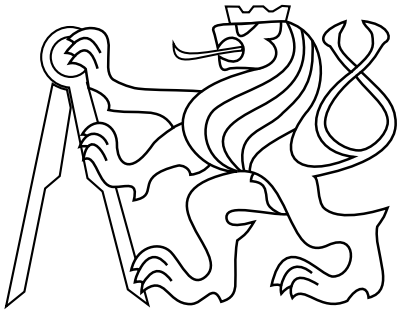
Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics
Multi-robot Systems



Automated Detection and Closing of Holes in Point Clouds Using Unmanned Aerial Vehicles

Bachelor's Thesis

Jan Ernée

Prague, May 2023

Study programme: Open Informatics
Specialisation: Artificial Intelligence and Computer Science

Supervisor: Ing. Vít Krátký

Acknowledgments

I would like to express my gratitude to my supervisor for his advice and guidance the whole time of my work.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Ernée Jan** Personal ID number: **486485**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Automated Detection and Closing of Holes in Point Clouds Using Unmanned Aerial Vehicles

Bachelor’s thesis title in Czech:

Automatická detekce a doplňování děr v mračnech bodů pomocí autonomních bezpilotních helikoptér

Guidelines:

The aim of the thesis is to implement an approach for identifying and completing occluded areas in unstructured 3D point clouds using Unmanned Aerial Vehicle (UAV). The thesis focuses on the implementation of an algorithm for finding boundaries of holes in unstructured 3D point clouds, the determination of areas suitable for mapping of detected holes, and the application of multi-goal path planning to find paths efficiently connecting the areas.
The following tasks will be solved:
• Implement an algorithm for finding boundaries of holes in unstructured point clouds and verify it on data obtained by terrestrial laser scanners in historical buildings.
• Propose and implement an approach for UAV path planning for mapping of the detected holes using onboard sensors with a limited field of view.
• Familiarize yourself with the system of the Multi-Robot Systems group for stabilization and control of UAVs [3].
• Verify the proposed approach in the Gazebo simulator under ROS using the MRS system and models of historical buildings.
• Prepare a real-world experiment which will be conducted based on the availability of a real multi-rotor helicopter and permission to access a building suitable for the realization of the experiment.

Bibliography / sources:

[1] G. Bendels, R. Schnabel and R. Klein, „Detecting Holes in Point Set Surfaces,“ Journal of WSCG, 14(1-3):89-96, 2006.
[2] A. Kazi, A. Sausthanmath, S. M. Meena, S. V. Gurlahosur, and U. Kulkarni, „Detection of holes in 3D architectural models using shape classification based Bubblegum algorithm,“ Procedia Computer Science, 167:1684-1695, 2020.
[3] T. Báča, M. Petrlík, M. Vrba, V. Spurný, R. Pěnička, D. Heřt and M. Saska, “The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles,” Journal of Intelligent & Robotic Systems, 102(26):1–28, 2021.

Name and workplace of bachelor’s thesis supervisor:

Ing. Vít Krátký Multi-robot Systems FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **31.01.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

Ing. Vít Krátký	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____ Date of assignment receipt

_____ Student's signature

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in th epreparation of university theses.

Prague, May 26, 2023 Jan Ernée

Abstract

Using a terrestrial laser scanner is one of the approaches to get a model of an inner space of historical monuments giving a model in the form of scanned points (point cloud) as a result. However, terrestrial laser scanners are usually not able to cover all areas of the scanned building. Thus areas, called holes, where no data was taken can appear in the model. The goal of this work is to find these holes in the unstructured point cloud and fill them using data captured by onboard sensors of unmanned aerial vehicles (UAV).

The first part of this work focuses on hole finding in unstructured point clouds. Holes are found by a combination of several criteria and methods, which are further discussed in more detail. To filling holes, areas from where the data can be obtained has to be found. This is a problem discussed in the second part of this work. The developed and implemented approach was verified in the realistic simulation using models of existing buildings.

Keywords Pointcloud Hole Detection, Angle Criterion, Shape Criterion, Halfdisc Criterion, RANSAC, GTSP, Unmanned Aerial Vehicles

Abstrakt

Jedním ze způsobů, jak získat model vnitřních prostor historických budov, je použití pozemního skeneru. Výsledkem může být model tvořený naskenovanými body (mračnem bodů). Pozemní skener často nezvládne zmapovat všechna místa skenované budovy. V modelu pak vzniknou tzv. díry, tedy oblasti, ve kterých nebyla data nasnímána. Cílem této práce je nalezení děr v nestrukturovaném mračnu bodů a jejich následné doplnění pomocí senzorů z autonomních bezpilotních prostředků (UAV).

První část práce se zabývá hledáním děr v nestrukturovaném mračnu bodů. Díry jsou nalezeny pomocí kombinace několika kritérií a metod, které jsou dále detailněji rozebrány. Pro doplnění děr daty je potřeba vybrat oblasti, ze kterých bude snímání pomocí bezpilotního prostředku provedeno. Této problematice se věnuje druhá část. Implementovaný přístup byl ověřen realistickou simulací za použití modelů skutečných budov.

Klíčová slova Detekce děr v mračnech bodů, úhlové kritérium, křivkové kritérium, polodiskové kritérium, RANSAC, GTSP, Bezpilotní prostředky

Contents

Chapter 1

Introduction

Visiting historical monuments is an activity many people like. Institutions and private owners often let people in, at least to some parts of its monuments. But there could be some rooms, cellars, halls, corridors, turrets, or whole buildings where visitors are not allowed for some reason even if there is a lot to see; private owner's areas, bad conditions for daily visits, or hardly accessible places. One way to resolve this situation could be a virtual tour of these places. But how to create a proper virtual model?

One solution could be done by using a terrestrial laser scanner to scan the whole place and create a virtual model based on obtained data. However, not even a laser scanner is perfect. Laser rays can reach only directly accessible places, therefore no data from behind obstacles can be obtained. Although the laser scanner is mobile and most of the surfaces are suitable for placing a scanner there, some areas could be inaccessible by the light ray from any angle from the ground. This leads to missing pieces of information about some parts of the scanned object and some uncertainty in the whole model result. In this work, we are focusing on finding these unscanned areas, called holes, and finding areas from which the missing data can be obtained by unmanned aerial vehicle (UAV) using onboard sensors.

In the first part of this work, we are resolving the problem of hole finding. Hole finding consists of several steps, together giving the required boundary points of the holes [1]. A combination of three criteria is used to find the probability of some point lying on the boundary or not. After that, extraction of the hole boundary is done. The second part focuses on finding suitable areas for obtaining required data by the UAV. Based on UAV's scanner properties like the minimum or maximum distance to take a scan or an angle where the obtained data are still accurate enough, this area is gained and sampled. Every area has to be visited by the UAV to collect required data. The suggested approach is validated in simulations using unstructured point clouds of some Czech historical monuments, containing hundreds of thousands or lesser millions of points.

1.1 Related works

Detection of holes in point cloud surfaces is not an easy task in general, mostly since point clouds could appear in an unstructured form. Points in the unstructured point cloud do not have any information about relationships with other points [2], thus some neighborhood of points has to be found. Some of the holes finding methods follow simple ideas and intuitiveness of how the point cloud surface behaves or looks on the hole boundary. There are methods of finding boundaries based on the size of the angle constricted by points on the boundary [1] or comparing the local neighborhood with the half-discs [1]. Other methods, on the other hand, compare the shape of boundary points neighborhood with the referenced shapes of correlation elipsoid [1], [3]. In [4], the method of finding boundaries is disengaged of any threshold values

and focuses on finding as many boundaries as possible, passing unique circles through three points in 3D space, and extracting the boundary. There are also methods assuming the point cloud is structured; for example [5] builds on the fact that the point cloud is organized and reduces the dimension of data by a projection of 3D data to the 2D grid. Another approach is to use convex hulls and clustering the points neighborhoods [6], applied in bioinformatics to detect skin cancer. In [7], the method uses clustering points in the point cloud to specific structures and finding holes in specific structure shapes. Various methods for various purposes either remove or provide more information. In this work, we follow the approach from [1], mainly because of a number of criteria and their combination, which allow us to modify the search methods based on the input data.

The missing point cloud data can be obtained in various ways, e.g by movement of the terrestrial scanner to different places (if the hole is reachable from the ground). The approach we are focusing on in this work is the usage of UAV. The interesting approach is presented in [8], where the UAS changes the flight direction depending on the founded point cloud holes. The impropriety of this approach for our proposes is its limitation to ground scanning, which makes it unusable for finding holes in the wall or ceiling. The inspiration to use the UAV for scanning is taken from [9] or [10]. In our work, we assume the usage of only one UAV during the scanning.

Chapter 2

System architecture

The model implemented in this task has several different parts. The goal is to obtain missing data of a point cloud of a historic building by a UAV. To reach this goal, the missing areas in the point cloud have to be found, the places to take appropriate scan shots have to be defined, the path through the defined areas and the trajectory have to be found and the data has to be obtained by the UAV following this trajectory.

The method we use to find the missing places, called holes, in the unstructured point cloud is based on the computation of the boundary probability of each point cloud's point and applying some constraints on them to receive the final points belonging to the point cloud's boundaries. The boundary probability is computed by a combination of three criteria, each focusing on a different description of what the boundary looks like. These three criteria compute a probability of how likely a point is a boundary point or not. Their combination is weighted, thus if one of the criteria is much better on the specific data set than the others, it could have a bigger weight. The number of detected boundary points, based on the boundary probability, is reduced depending on their coherence. Finally, the cycle of boundary points is extracted.

The founded hole is approximated by the plane and divided into smaller rectangle holes of specific proportions to satisfy the constraints of a scan. The areas to take scan shots are mainly limited by the distances from the hole and the angle constricted with the hole's approximation plane. The problem of visiting areas could be described as TSP, but typically the found areas are bigger than the UAV and thus the areas are sampled to allow UAV to pass through some point in the area. This sampling leads to a generalization of TSP to GTSP. Finally, the path is founded through GTSP solution points and a trajectory is generated. The whole approach is tested in realistic simulation and models of Czech historical buildings.

The realistic simulation with the models of real historical buildings is a good preparation for real experiments in real historical buildings inspired by [11]. The generated trajectories and the MRS systems [12], [13] allow usage of this approach to obtain missing data from real buildings by real UAV.

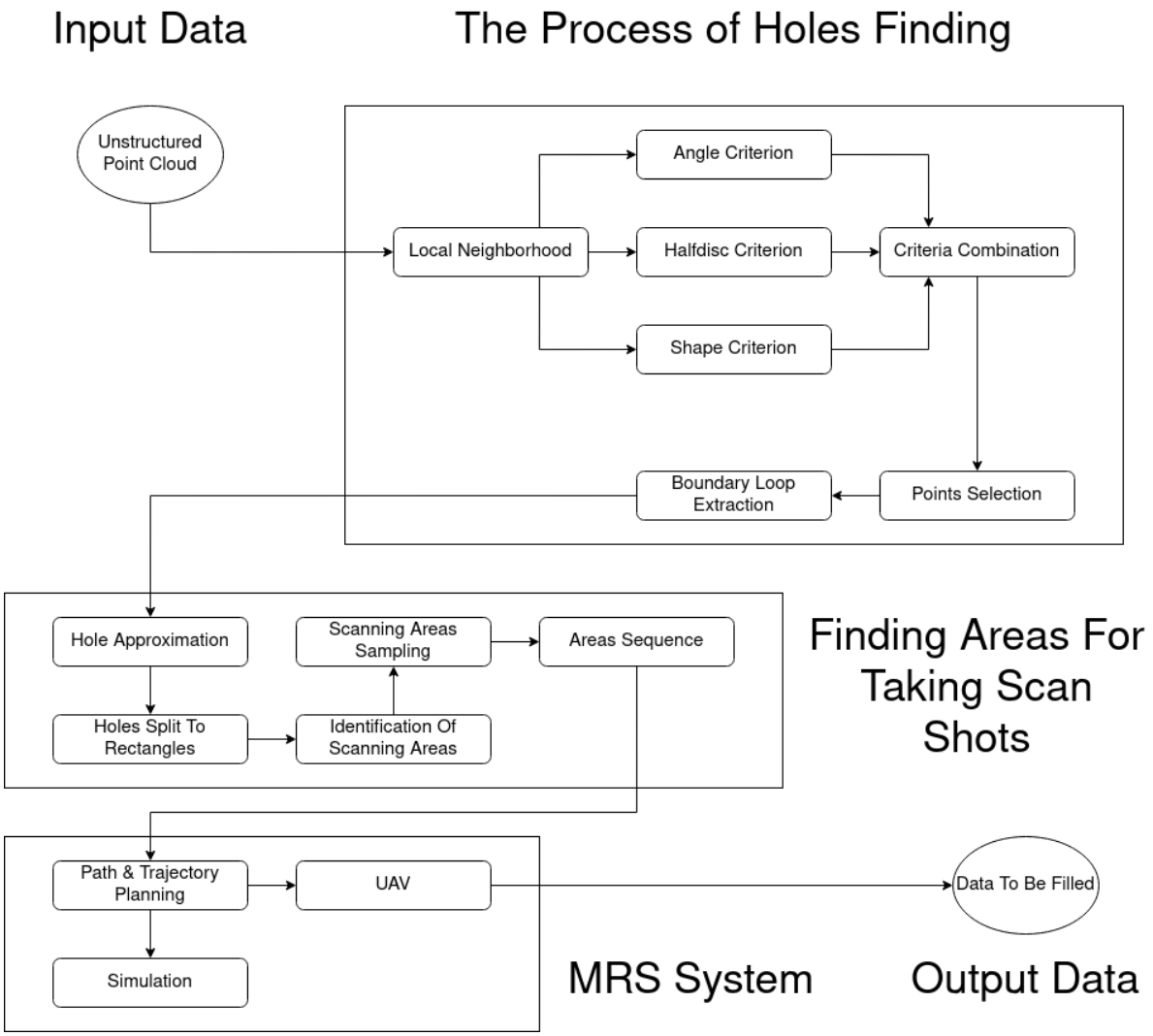


Figure 2.1: Scheme of the whole approach. The input data is an unstructured point cloud of a building (or another object where the scanning by UAV makes sense). The hole finding task (the first rectangle) has several parts, together providing the point cloud holes' boundary points. The second part focuses on finding areas suitable for taking scan shots. This process shows the second rectangle (following the direction of the arrows) and provides the points in space which have to be visited by the UAV as a result. The final step is the path planning and the realistic simulation (done by using the [12] system) to verify if the desired data will be obtained during the real drone flight. The output data from the system are the missing data scanned by the UAV.

Chapter 3

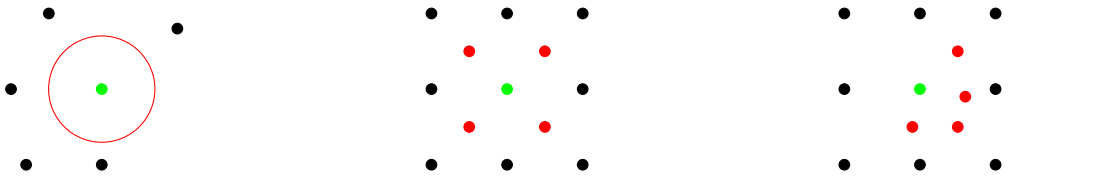
Pointcloud hole detection

This chapter covers the whole process of detecting holes in the unstructured point cloud. Unstructured point cloud, unlike its structured version, misses any information about the connectivity between neighboring points [2]. In our case, the only information we have about each point individual is its position and color. The approach described in this chapter is based on method [1] with minor modifications required by intended application.

3.1 Local neighborhood

Basic source of information about the relation between points is their mutual position. Being a boundary point depends more on the distribution of the nearest points instead of the boundary point itself [1], i.e. it is a property dependent on the points in certain neighborhood.

For some point, the neighborhood could be simply defined as a sphere of specific diameter ϵ around that point. This neighborhood definition has one significant deficiency. For small diameter and small sampling density, there could be no point in the neighborhood, i.e. there is no information about the relationship with other points, and the problem about missing relations remains, as shown in Figure 3.1a. Taking K-nearest points, determined by the Euclidean distance, could be a better approach than a sphere of predefined size. K-nearest neighbours ensures the neighbourhood will contains $min(K, \textit{total data points})$ points.



(a) ϵ-neighborhood

(b) K-nearest neighbors on uniform sampled data

(c) K-nearest neighbors on nonuniformly sampled data

Figure 3.1: ϵ-neighborhood of the green point, missing any other point in it (a), whereas K-nearest neighbors for $K = 4$ contains the required number of points (b). The difference between K-nearest neighbors applied on a uniform and nonuniform sampling density is shown in (b) and (c)

K-nearest neighbors works well for cases where the sampling density is uniform for most of the points, see Figure 3.1b. Neighboring points occur in all directions, the point seems to lay in the middle of its neighbors. Unfortunately, uniform sampling density is not guaranteed data property. K-nearest neighbors used on unevenly sampled data gives information biased

to the high-density direction [1] and the information locally differs from the global point of view. The information about the point based on its neighborhood leads to the conclusion that it lays on the margin of the data, but this does not have to be true, as shown in Figure 3.1c.

Disadvantages of both approaches could be eliminated by the combination of both approaches. For the point $\mathbf{p}$, let us set the neighborhood as $N_{\mathbf{p}}^{k\epsilon}$, where N is a neighborhood of $\mathbf{p}$ containing k-nearest neighbors likewise all points in the sphere of a small radius ϵ. For well-chosen ϵ, the problem with uneven sampling density is partially removed. While areas with lower sampling density will work without any affection (for appropriate ϵ), highly sampled places will contain much more points; a high amount of points slows down the process of finding boundaries [1], moreover if many points have no bonus information, just lays in the high sampled region. This behavior is shown in Figure 3.2.

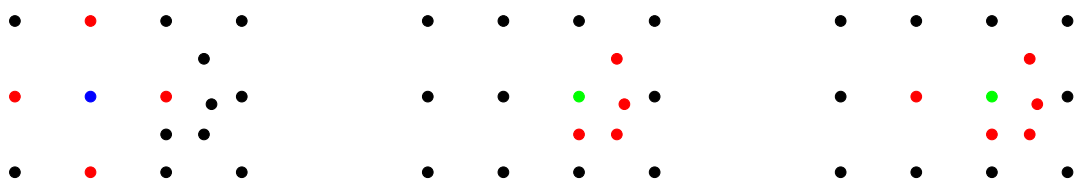
(a) (b)

Figure 3.2: $N_{\mathbf{p}}^{k\epsilon}$, $\mathbf{p}$ is the green point and red points correspond to neighbors in the ϵ radius. For $K = 4$, the number of points in the high sampled region is doubled (b) compared to the lower one (a).

For points on the boundary between densely and sparse sampled regions, moreover if the crossing is sharp, points in the sparse area will contain members from the densely sampled region in their neighborhood [1]. On the other hand, points on the dense boundary will most probably contain only points from the densely sampled region in their neighborhood. To avoid this difference, points are considered to lie in some point's neighborhood also if their neighborhood contains that point, i.e.

$$N_{\mathbf{p}} = \{\mathbf{q} \in \mathcal{P} \,|\, \mathbf{q} \in N_{\mathbf{p}}^{k\epsilon} \vee \mathbf{p} \in N_{\mathbf{q}}^{k\epsilon}\}, \tag{3.1}$$

where $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\} \subset \mathbb{R}^3$ is a set of points. The final definition of neighborhood encapsulates Figure 3.3.



(a) (b) (c)

Figure 3.3: (a) and (b) shows the neighborhood of two near points without our new definition. Because a green point is in the blue's neighborhood, it should also contain it in its neighborhood, as shown (c).

Searching for the neighborhood is done using library PCL [14]. This library provides the implementation of the kd-tree structure, which makes it easy and efficient to use for finding KNN, as well as ϵ-neighbourhood. Kd-tree is a multidimensional binary tree, where K stands for the data dimension, as described in detail in [15]. Finally, the graph $\mathcal{G}(\mathcal{P}, \mathcal{E})$ is created, where $\mathcal{P}$ stands for nodes and set of

$$\mathcal{E} = \{(i,j) \,|\, \mathbf{p}_j \in N_{\mathbf{p}_i}\} \tag{3.2}$$

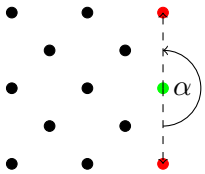
for edges. In connection with our definition of $N_{\mathbf{p}}$, for every edge (i,j) there is another edge (j,i), i.e. the graph is symmetric.

3.2 Boundary probability

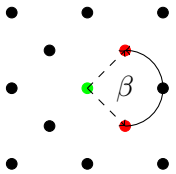
As was mentioned before, the property of being a boundary point mostly depends on the point's neighborhood. Therefore, the boundary probability $\Pi(\mathbf{p})$ is computed for every point $\mathbf{p} \in \mathcal{P}$. This probability reflects if the point lies exactly on the hole boundary, or near the boundary [1]. A point is classified as suspicious of being a boundary point if its boundary probability is above some threshold, typically different for various data. This section covers finding this probability for each point $\mathbf{p} \in \mathcal{P}$ using three different criteria; angle criterion, half-disc criterion and shape criterion.

3.2.1 Angle criterion

Angle criterion points to the fact, that there is a much larger gap between points laying on the boundary (Figure 3.4a) than the interiors ones [1] (Figure 3.4b), i.e. angle around origin $\mathbf{p}$ of two boundary points from $N_{\mathbf{p}}$ is larger than the angle of the interior points in the same meaning.



(a) Angle on boundary



(b) Angle inside surface

Figure 3.4: For green point, $\mathbf{p}$ and red points $\mathbf{q}_1, \mathbf{q}_2 \in N_{\mathbf{p}}$, angle α between boundary points is much bigger (a) than angle β constricted by two interior points (b).

There is a problem how to choose points which constrict the biggest angle. It is hard to sort angles in 3D based on the same comparisson rule, while in 2D problem it is easy to compare consecutive angles between points sorted in some way, e.g points around some origin. To avoid this problem with angle comparison, one dimension of the data has to be reduced so all data points from $N_{\mathbf{p}}$ will be situated on the same plane. There are various methods for dimension reduction and in the case of finding the biggest gap between two vectors the projection to the tangent plane is used. For every $\mathbf{q} \in N_{\mathbf{p}}$, projection to the tangent plane of $\mathbf{p}$ is made. The reduced problem now consists of finding the largest angle between vectors

connecting the origin point $\mathbf{p}$ with consecutive neighbors from $N_\mathbf{p}$. Boundary probability is then given as

$$\Pi_\angle(\mathbf{p}) = \min\left(\frac{g - \frac{2\pi}{|N_\mathbf{p}|}}{\pi - \frac{2\pi}{|N_\mathbf{p}|}}, 1\right), \tag{3.3}$$

where $\Pi_\angle(\mathbf{p})$ stands for boundary probability of $\mathbf{p}$ using angle criterion, g stands for the largest angle/gap (in radians), and $|N_\mathbf{p}|$ stands for the number of points belonging to $N_\mathbf{p}$. The whole process and the difference between interior and boundary points is shown in Figure 3.5.

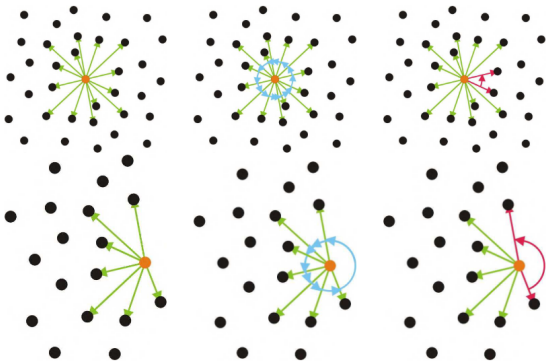


Figure 3.5: The process of finding the maximum gap. The top row shows the interior point whereas the boundary point is located in the bottom row. Vectors on the tangent plane (left), sorting by angles (middle) and found largest angle between two consecutive points (right)(image taken from [1]).

3.2.2 Halfdisc criterion

A simple idea to identify a hole in a surface is based on the fact that the boundary has some points on one side and no points on the other side compared to the interior point where points appear in all directions (related to the 2D surface). This idea is exploited by the half-disc criterion. For an interior point, its neighborhood is similar to a disk (Figure 3.6a), thus average of all neighboring points $\mu_\mathbf{p}$ does not differ from the interior point much. Opposite to this, neighboring points of a boundary point lies only in some directions, i.e there is an area without any point and the neighborhood resembles a half-disc (Figure 3.6b). In this case, the average $\mu_\mathbf{p}$ differs much more from the boundary point because it is biased towards the neighborhood majority [1]. Therefore, the difference between point $\mathbf{p}$ and the average of its neighborhood $\mu_\mathbf{p}$ projected to the tangent plane is used to compute the boundary probability by comparing $\mu_\mathbf{p}$ with the center of ideal half-disc. Boundary probability of $\mathbf{p}$ using a half-disc criterion is thus

$$\Pi_\mu(\mathbf{p}) = \min\left(\frac{\|\mathbf{p} - \bar{\mu}_\mathbf{p}\|}{\frac{4}{3\pi} r_\mathbf{p}}, 1\right), \tag{3.4}$$

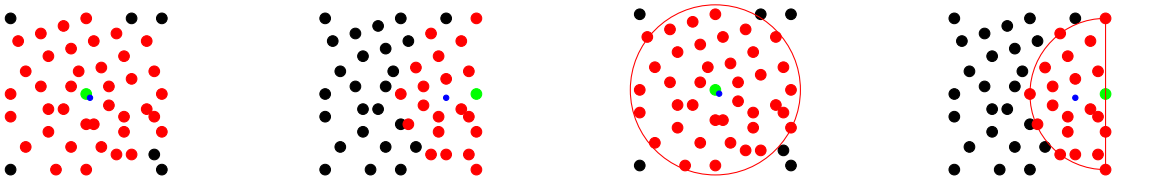
where $\bar{\mu}_{\mathbf{p}}$ stands for tangent projection of $\mu_{\mathbf{p}}$ to avoid surface properties influencing half-disc criterion and include only the sample ones [1], $r_{\mathbf{p}}$ is the average distance between $\mathbf{p}$ and points from $N_{\mathbf{p}}$. $\mu_{\mathbf{p}}$ itself is computed as weighted average

$$\mu_{\mathbf{p}} = \frac{\sum_{\mathbf{q}\in N_{\mathbf{p}}} g_\sigma(\|\mathbf{p}-\mathbf{q}\|)\mathbf{q}}{\sum_{\mathbf{q}\in N_{\mathbf{p}}} g_\sigma(\|\mathbf{p}-\mathbf{q}\|)} \tag{3.5}$$

to better deal with the undesirable effect of various sampling densities or some point's deviation, where a Gauss kernel is used as a weight in the form of

$$g_\sigma(d) = \exp\left(\frac{-d^2}{\sigma^2}\right). \tag{3.6}$$

Here, σ factor depends on the average distance as $\sigma = \frac{r_{\mathbf{p}}}{3}$ and neglect points outside of the neighbourhood $N_{\mathbf{p}}$.



(a) Interior neighborhood

(b) Boundary neighborhood

(c) The approximate shape of an interior neighborhood

(d) The approximate shape of the boundary neighborhood

Figure 3.6: Let $\mathbf{p} \in \mathcal{P}$ and $\mathbf{q} \in N_{\mathbf{p}}$. An average $\mu_{\mathbf{p}}$ of the neighborhood of an interior point $\mathbf{p}$ differs not so much from $\mathbf{p}$ (a), whereas the difference between $\mu_{\mathbf{p}}$ and $\mathbf{p}$ is more significant for $\mathbf{p}$ lying on boundary (b). Appropriate disk of neighboring points shows (c), while the halfdisc shape of a neighborhood is on (d).

3.2.3 Shape criterion

Shape criterion is based on the shape of correlation ellipsoid of $N_{\mathbf{p}}$ approximating the shape of the point's neighborhood in general [3]. For shape criterion, covariance matrix $C_{\mathbf{p}} \in \mathbb{R}^{3\times3}$ is essential. The covariance matrix has a form

$$C_{\mathbf{p}} = \sum_{\mathbf{q}\in N_{\mathbf{p}}} (\mu_{\mathbf{p}} - \mathbf{q})(\mu_{\mathbf{p}} - \mathbf{q})^T, \tag{3.7}$$

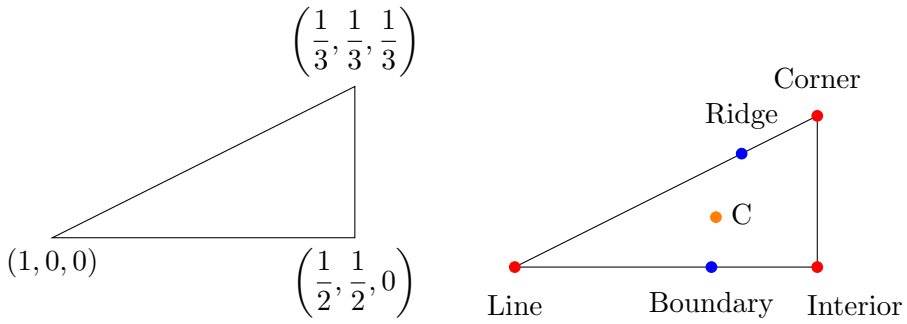
where $\mu_{\mathbf{p}}$ has the same meaning as for the Halfdisc criterion [1]. As mentioned in [3], eigenvectors $\{e_0, e_1, e_2\}$ and eigenvalues $\{\lambda_0, \lambda_1, \lambda_2\}$ contain the information about the shape of correlation ellipsoid, i.e the shape of the surface around some point. For different surface areas, the correlation ellipsoid approximates the appropriate shape and the eigenvalues have specific relation for some kind of shapes, e.g $\lambda_0 \approx \lambda_1$ and $\lambda_2 \approx 0$. For decision vector $\Lambda_{\mathbf{p}} =$

$\left(\frac{\lambda_0}{\alpha}, \frac{\lambda_1}{\alpha}, \frac{\lambda_2}{\alpha}\right)$, $\alpha = \lambda_0 + \lambda_1 + \lambda_2$, $\lambda_0 \geq \lambda_1 \geq \lambda_2$ [1], the relations between eigen values are normalized and shown in Table 3.1.

$\phi =$ Boundary	$\Lambda_\phi = \left(\frac{2}{3}, \frac{1}{3}, 0\right)$
$\phi =$ Interior	$\Lambda_\phi = \left(\frac{1}{2}, \frac{1}{2}, 0\right)$
$\phi =$ Ridge	$\Lambda_\phi = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right)$
$\phi =$ Corner	$\Lambda_\phi = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$

Table 3.1

The table shows situations $\phi \in \Phi$, $\Phi = (\text{Boundary}, \text{Interior}, \text{Corner}, \text{Ridge})$ and its corresponding normalized decision vector values. Decision vector $\Lambda_\mathbf{p}$ for each point $\mathbf{p}$ lies in the triangle [1] 3.7a with appropriate meaning shown on 3.7b.



(a) Triangle defining points (b) Points on triangle

Figure 3.7: Points defining the triangle (a) and points representing some shape lying on the triangle (b).

Points shown on 3.7b are used as reference points for finding the boundary probability of point $\mathbf{p}$ using the shape criterion. For each state the tentative probability $\tilde{\Pi}_\phi$ is computed as

$$\tilde{\Pi}_\phi(\mathbf{p}) = g_{\sigma_\phi}\left(\|\Lambda_\mathbf{p} - \Lambda_\phi\|\right) \tag{3.8}$$

for each $\phi \in \Phi$, where $\Lambda_\mathbf{p}$ is a decision vector of point $\mathbf{p}$, Λ_ϕ is a reference vector of specific shape and g_{σ_ϕ} is a Gauss kernel with $\sigma_\phi = \frac{1}{3}\|\Lambda_\phi - C\|$, where C is a centroid of triangle (Figure 3.7b). To keep final probability in interval $[0, 1]$, the normalisation is necessary

$$\Pi_\phi(\mathbf{p}) = \frac{\tilde{\Pi}_\phi(\mathbf{p})}{\sum_{\psi \in \Phi} \tilde{\Pi}_\psi(\mathbf{p})}. \tag{3.9}$$

3.2.4 Criteria combination

We have discussed three different approaches to hole finding so far. To make hole finding much more robust and get better results, a combination of all three criteria is used [1]. There is one important requirement. Given data has to be meaningful and clear for humans, at least in the case of finding holes in point-cloud by visual inspection. Point-cloud data with a property of clear hole recognition by humans allows users to choose proper values of used parameters, e.g. value of K for K nearest neighbors, ϵ neighborhood, or weights of each criterion. For different data, criteria can behave differently; different sampling densities or different distances between points, as well as the required size of detected holes, will require different values of K or ϵ. The final boundary probability is computed as a weighted sum of boundary probabilities given by all three criteria; angle criterion, half-disc criterion and shape criterion. The final formula has form

$$\Pi(\mathbf{p}) = w_{\angle}\Pi_{\angle}(\mathbf{p}) + w_{\mu}\Pi_{\mu}(\mathbf{p}) + w_{\phi}\Pi_{\text{Boundary}}(\mathbf{p}), \tag{3.10}$$

where $w_{\angle}$, w_{μ} and w_{ϕ} are weights and $w_{\angle} + w_{\mu} + w_{\phi} = 1$ to keep the properties of probability.

3.3 Boundary points extraction

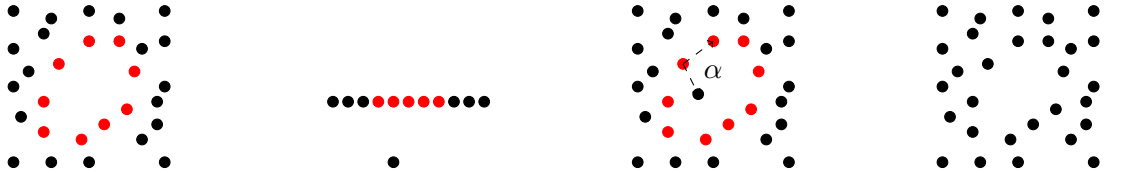
Methods for finding a point's boundary probability have been discussed so far. Depending on criteria weights and the minimum acceptable value of boundary probability (threshold set by user), the point is marked as *boundary point* or an *interior point*. This approach marks a point based only on the probability of this point, without any affection by the probability of other nearest points. In other words, there is no binding between points marked as *boundary point*s. To improve the method of finding holes, points will be marked as *boundary point*s only if these points will belong to boundary loops [1] with some *boundary point*s in their neighborhood. This improvement will be discussed in this section.

3.3.1 Points selection

Not every point marked as a boundary point by the criteria mentioned above really belongs to the boundary. There could be some noise points or wrongly marked points for example. One property of being a boundary point is that the point can not lay on the boundary alone. To avoid some noise points being evaluated as boundary points after the whole process of hole finding, points that do have not at least two boundary points in their neighborhood (allow them to be a part of the boundary loop) will be marked as interior points. This approach is done by an iterative algorithm taken from [1].

The angle criterion described above is used again to get the biggest angle between points in the neighborhood of the point marked as *boundary point*. The point remains *boundary point* if and only if points from its neighborhood taking the biggest angle are also marked as *boundary point*s. While there is a change of state at any point, this approach is repeated. Because of the sensitiveness of hole finding in this part, it works well for us to let the point's neighborhood of flexible size, i.e neighborhood size of point during angle criterion algorithm computing a boundary probability and this *selecting* algorithm can differ. The main reason is that computing boundary probability depends on the majority of the neighborhood and some

deviations can be neglected with smaller affection, but for the boundary loop preparation during this algorithm, any deviation can lead to marking all boundary points as interiors (Figure 3.8).



(a) Boundary points from the front view

(b) Top view in the middle of the hole with some noise

(c) Front view with the projected noise

(d) Front view after iteration end

Figure 3.8: Black points stand for interior points, while the red points stand for boundary points. Front view on the hole with marked boundary points before iterative algorithm (a). Top view (from the middle of the hole) with some noise at the side of the hole (b). Front view during the iterations, where the maximum angle is made by another boundary point and the projected noise (c). Points marked as *boundary points* after the iteration (none of the points remains) (d).

3.3.2 Minimum spanning graph

In our case, the minimum spanning graph (MSG) is a graph obtained by modified Kruskal's algorithm [1] for finding a minimum spanning tree. In Kruskal's algorithm, every edge has a price (or length, weight). The search begins by lining their edges up in ascending order by their weights. Starting with the cheapest edge, the tree is constructed by adding every edge (taken in the sorted order) which does not create a loop in the tree [16].

As mentioned at [1], the edge weight $w(i,j)$ is computed as combination of two parts

$$w_{\text{total}}(i,j) = w_{\text{probability}}(i,j) + w_{\text{density}}(i,j). \tag{3.11}$$

The first component of the final edge weight focuses on boundary probability. Boundary probability is a value that belongs to points, so the weight $w_{\text{probability}}(i,j)$ of edge $e_{ij} = (i,j)$ depends on boundary probability of both graph nodes (both points) as

$$w_{\text{probability}}(i,j) = 2 - \Pi(\mathbf{p}_i) - \Pi(\mathbf{p}_j). \tag{3.12}$$

Second member of the edge total weight $w_{\text{total}}(i,j)$ is $w_{\text{density}}(i,j)$. This weight part has form

$$w_{\text{density}}(i,j) = \frac{2\|\mathbf{p}_i - \mathbf{p}_j\|}{r_{\mathbf{p}_i} + r_{\mathbf{p}_j}} \tag{3.13}$$

and focuses on points distance and sampling density. The numerator contains the distance between both points, i.e. the further these points are, the bigger value of $w_{\text{density}}(i,j)$, and the later in the modified Kruskal's algorithm corresponding edge will be processed. This property is good to have more points in the boundary loop so that the shape of the boundary

will be more precise. Opposite to this, the denominator contains information about sampling density around the point, more specifically in its neighborhood. $r_{\mathbf{p}}$ stands for average distance from point $\mathbf{p}$ to each point in its neighborhood. The smaller $r_{\mathbf{p}}$ is, the small the denominator is and the bigger the $w_{\text{density}}(i,j)$ so the algorithm will not stack in the highly sampled regions.

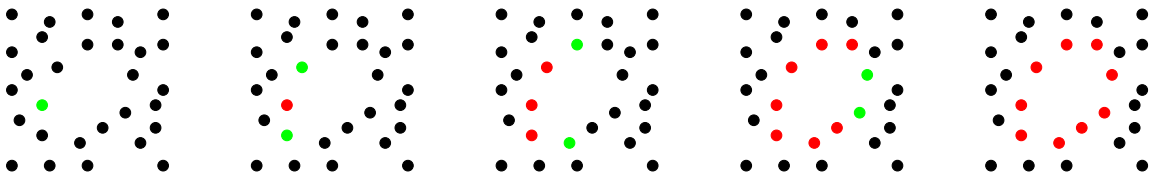
The procedure of finding the minimum spanning graph starts by sorting all edges $e_{ij} \in \mathcal{E}$ of graph $\mathcal{G}$, where $\mathbf{p}_i$ and $\mathbf{p}_j$ are boundary points, by weight $w_{\text{total}}(i,j)$ where $w_{\text{total}}(i,j)$ is below some threshold (the threshold can be different for different data sets). After that, every edge is put into MSG as a single component. Starting with the lowest cost edge if the edge connects two distinct components in MSG, these components are joined and the edge is put in this component. In another case, if an edge connects points in one component, i.e. after insertion there will be a loop, the edge is inserted only if the newly arising loop is longer (in the meaning of a number of edges) than the predefined minimum loop length [1] and also if the new loop will be longer than any existing loop in this component. If there is a loop yet, the longer one remains and the shorter one is removed. The minimum loop length e is interconnected with the ϵ defining size of the neighborhood by the formula

$$e = \frac{2\pi\epsilon}{d}, \tag{3.14}$$

where d stands for average edge weight. The minimum length of the cycle also defines the minimum size of the hole.

3.3.3 Boundary loop extraction

The minimum spanning graph is a graph containing one cycle whose nodes define the desired boundary, i.e points in the cycle are the searched boundary points. The last step is to extract this cycle and its points from the graph to make the boundary points accessible directly without searching for them every time they are needed. To find the cycle in the MSG the breadth first search (BFS) [16] algorithm is used.



(a) BFS start (b) 1st iteration of BFS (c) 2nd iteration of BFS (d) *Queued* point adjoins the processing one (e) The final loop is found and marked

Figure 3.9: BFS algorithm used for loop finding. Black points are *untouched*, greens are *queued* and reds are *processed*. The beginning of BFS is shown on (a), next iterations are shown on (b) and (c). Cycle location is shown on (d); one of the adjacent points is *queued* yet. The final loop is shown (e).

As mentioned in [1], a point can be in one of the three states; *untouched*, *queued*, *processed*. In the beginning, every point is marked as *untouched*, except the starting point which is marked as *queued* and is put into the queue. In every algorithm step, a front point in

the queue is removed and marked as *processed*. All *processed* neighboring points are ignored (since typically in every step the parent point has already been processed) and all of its *untouched* neighboring points are marked as *queued* and put at the end of the queue. If there is a *queued* adjacent point, the cycle is found because this *queued* point connects a path from its *processed* neighboring points to a cycle [1]. These cycle points are extracted by backtracing the BFS steps and are stored for later use as points of specific boundary, i.e the searched boundary points. The whole process of boundary loop extraction is shown in Figure 3.9.

Chapter 4

UAV path planning

The second part of this work covers the process of finding the UAV path between previously detected holes. Primarily, we are focusing on finding suitable areas from where the missing data can be obtained by UAV. For finding the path between these areas the planner [17] is used.

4.1 Hole's approximation plane

The point cloud hole is defined by its boundary points and the process of finding these holes boundaries is described above. In some cases, typically when the hole is lying on the planar surface, the boundary points resemble some planar shape. Unfortunately, not every hole approximates a planar shape, thus there can appear holes in which the boundary points create complicated shapes in 3D space. To make it easier to work with sets of boundary points that resemble these complicated shapes, we have decided to approximate each hole with a plane. With the knowledge of boundary points, their positions and their coherence, the algorithm called *random sample consensus* [18] (RANSAC) has been used for finding the best plane that approximates the boundary points and containing at least three points of them. This approach is used for boundary points of every discovered hole in the point cloud.

(a) Boundary points from the front view (b) Side view: 1st iteration (c) Side view: 2nd iteration (d) Side view: 3rd iteration

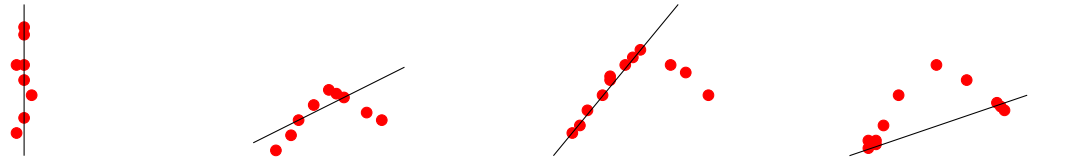
Figure 4.1: The process of finding the best approximating plane of the hole (a). The RANSAC algorithm makes three iterations in this situation (b), (c), (d). The second iteration (c) shows the plane which approximates the hole (a) the best.

RANSAC algorithm iterates k-times over some data set $\mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_1, ..., \mathbf{b}_j\}$ and at every iteration i, it *randomly* chooses appropriate number n of data points $\mathbf{b} \in \mathcal{B}$ that defines the desired shape, e.g two points for the line, three points for the plane.

Since the plane is defined by three points, which do not lie on the same line, in every iteration different three points $\mathbf{b} \in \mathcal{B}$ are randomly chosen from the set of boundary points $\mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_1, ..., \mathbf{b}_j\}$ corresponding to the currently approximated hole. How well some plane fits the set of boundary points is measured by the number of boundary points $\mathbf{b}$ (corresponding

to the same hole) which have distance d from the plane below some threshold t. The plane which fits the best all boundary points after all iterations, i.e. has the most boundary points in the distance below the threshold, is chosen as the plane approximating the corresponding point cloud hole. Because of different data models, as well as different model scales or different sizes of holes (even within one point cloud), parameters like the number of iterations k or the threshold distance t are set individually for every model. Figure 4.1 shows the process of finding the best plane approximating some hole.

Discovered planes approximating the point cloud holes are really good to work with. However, the approximation of a hole could differ from the original hole shape significantly. Figure 4.2 shows the difference between the plane approximating some planar hole (e.g glass table in the door) and the plane which approximates the more complex shape of boundary points (missing corner of some room). There is also affection by the density of boundary points in some areas of the boundary, as shown in Figure 4.2 as well.



(a) Top view on the boundary points of a hole in a planar surface

(b) Top view on the boundary points of a corner hole

(c) Top view on the boundary points of a corner hole with higher point density on one side

(d) Top view on the boundary points of a corner hole with higher point density far from the corner in one direction

Figure 4.2: Different types and accuracy of approximation plane found by RANSAC algorithm for different types of holes. The hole in a planar surface (a) and the hole in a corner (b), both from the top view. (c) and (d) show how approximation planes of the same hole with different boundary point density can differ.

4.2 Finding areas for taking scan shots

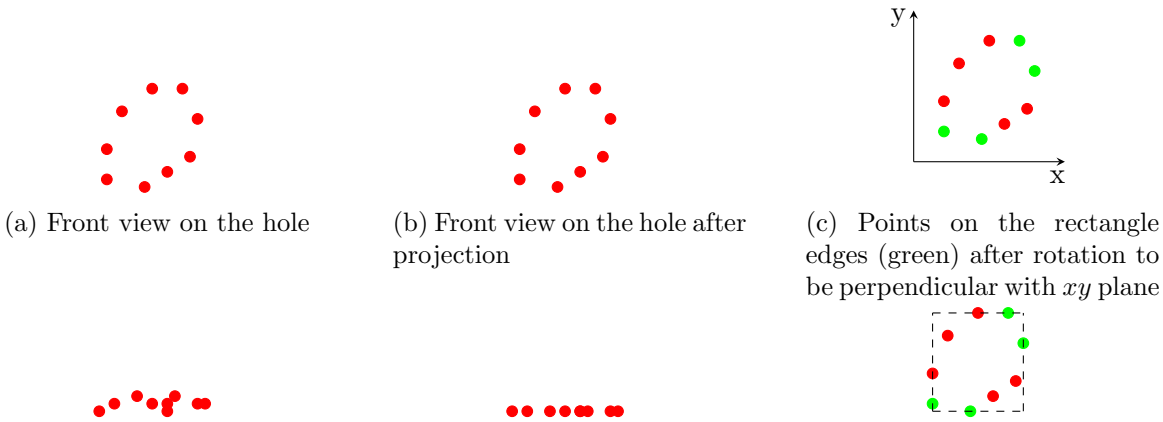
Point cloud hole's approximation plane allows us to measure the distance from the hole and make the work with the hole easier. Because of the rectangular shape of the field of view of the used UAV sensor, the hole can be also approximated by a rectangle. Since there are other limitations, e.g. required scanning distance or the hole could be too large to be scanned by one scan shot. Thus the large holes have to be divided into smaller ones. Finally, areas from where the approximating holes rectangles could be scanned have to be found. The following sections describe the finding of the areas from where the required scan shots can be obtained.

4.2.1 Rectangle surrounding boundary points

One limitation of onboard sensors on UAV is their limited field of view. The shape of this view is typically a rectangle and to take a scan of the hole there has to be a rectangle surrounding this hole. For small holes, a rectangle of appropriate size is enough for the hole to be scanned, but for large holes, it has to be divided into multiple smaller rectangles.

The hole's boundary points can form a complex shape in 3D space and it could be hard to identify the rectangle surrounding this boundary. The approximate plane now represents the place where we assume that the boundary is. For better work with the boundary points, the projection of all hole's boundary points to the approximation plane is made and the boundary still has the same shape from the point of view in the direction perpendicular to the approximate plane. Perpendicular direction to the approximate plane is a direction we assume the scan shots will be taken from.

The rectangle is defined by points lying the most up, down, left and right on the plane. To extract them, all projected points are rotated to be parallel with the xy plane (there is no problem because rotation is a linear function [19]) and the points with the smallest x, the smallest y, the biggest x and the biggest y coordinate are marked as projected points on the edges of the projected rectangle. Original points corresponding to the marked projected ones are marked as points defining the corresponding rectangle of the boundary points plane projections, e.g these points lie on the edge of the rectangle surrounding boundary points projected to the approximate plane (see Figure 4.3).



(a) Front view on the hole

(b) Front view on the hole after projection

(c) Points on the rectangle edges (green) after rotation to be perpendicular with xy plane

(d) Top view on the hole

(e) Top view on the hole after projection

(f) Marked points and corresponding rectangle after the whole process on the original boundary points

Figure 4.3: (a) and (d) shows the initial hole from two points of view. (b) and (e) shows the hole after projection to the approximate plane. Points lying on the rectangle edges in the xy plane are on (c). The final rectangle with its boundary points is on (f).

4.2.2 Hole splitting

UAV sensors which are used to scan missing point cloud data have some limitations. Especially for our work, besides the limited field of view, we are limited by the distance from where the scan shot can be taken (both, minimum and maximum distance) or by an angle, which defines if the obtained data are still accurate enough, i.e there is no new hole between scanned data points because of big distance between scanned points caused by too small scanning angle. For small holes, we are done with the boundary points of the surrounding rectangle. But some holes can be really big, at least big enough for one scan shot from the UAV and thus the hole has to be divided into some smaller segments.

The smaller scannable areas are gained utilizing the surrounding rectangle of a boundary. The rectangle can represent the boundary according to the shape of the scanner's field of view. Thus, if the surrounding rectangle is larger than the field of view, it is divided into several smaller rectangles of suitable proportions of the field of view (Figure 4.4). These smaller rectangles are counted as separate holes of rectangle shape and each of them has to be scanned.

(a) (b)

Figure 4.4: Boundary points with surrounding rectangle too big for accurate scanning (a) and the same rectangle after division to appropriate smaller areas (b).

4.2.3 Identification of scanning areas

Rectangles, surrounding parts of the hole, defines areas which should be scanned. With these rectangles in hand, areas from where the scans can be taken by the UAV have to be found. There are three limitations on the suitable scanning area in our case; the minimum distance to the hole d_{min}, the maximum distance to the hole d_{max} and the minimum angle constricted with the hole ϕ. We assume that the condition $d_{max} > d_{min}$ is satisfied. All these three limitations are measured according to the rectangle on the approximate plane of the hole for simplifying the task.

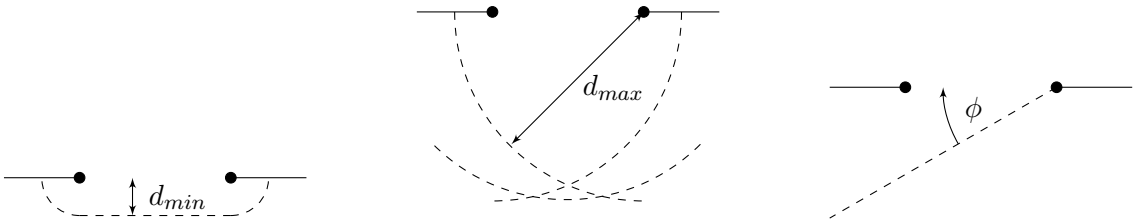
The size of the rectangle which can be captured by an onboard sensor with limited field of view depends on the distance to the hole. Thus one more simplification for our case is made. The rectangles into which the hole is split corresponds to the size of projection of field of view into the plane in distance d_{min}. This is our reference rectangle size, e.g the size of the area that can be captured by a single scan. The appropriate area to take scans from is then found to satisfy the limitations according to the size of this scannable area rectangle.

The first limitation is the minimum distance d_{min}, which means that the UAV has to be at least as far from the hole as the minimum distance is. This leads to constraint given by a plane which is parallel with the hole's plane and the curved shape around the hole's rectangle boundary with the distance between equals to the minimum distance d_{min} (Figure 4.5a).

The second limitation is the maximum distance d_{max}. The shape of the outer boundary of the area for the UAV is more complicated than the plane. Not every point distant from the hole's plane at the distance d_{max} has the same or lesser distance from the different place on the same hole's plane. The maximum distance d_{max} defines for each point on the hole's plane a sphere of diameter d_{max} where the condition is satisfied. Intersection of all these spheres gives the searched area (Figure 4.5b).

Not only the distance from the hole's plane is the limitation. If the UAV scans the area under too acute angle, the obtained data is not accurate enough. Even if the obtained data looks good from the point of view of the UAV, there could be a low points density and big gaps

between the points from the point of view perpendicular to the obtained data. To avoid this, one of the limitations is the minimum angle ϕ constricted with the hole's plane. The plane which constricts this angle ϕ with the hole's plane stands for another "minimum distance" boundary (Figure 4.5c).



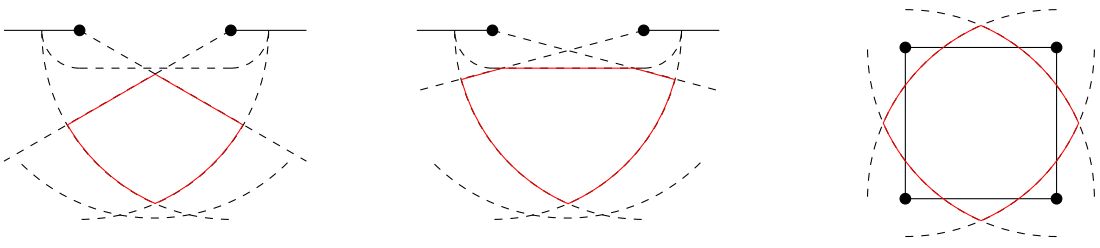
(a) Minimum distance to the boundary plane

(b) Maximum distance to the boundary plane

(c) Minimum constricted angle with the boundary plane

Figure 4.5: The boundary defined by the minimum distance d_{min} to the boundary plane. UAV has to be at least as far as the d_{min} from the boundary plane (a). The shape of area boundary defined by the maximum distance d_{max}. The crucial is the distance from the boundary of the scaned rectangle (b). The plane defined by the minimum angle ϕ defines the minimum distance boundary in another way (c). All images are from the top view, but the side views are the same. The dashed line shows area boundaries.

The final area has to satisfy all the three limitations, thus the area is made by the intersection of all the shapes. The final shape of the area depends on the values of the distances d_{min} and d_{max} and also on the size of the angle ϕ. The minimum distance to the hole is given by the maximum of the distance d_{min} and the distance to the plane defined by the angle ϕ (Figure 4.6b). In some extreme cases, where the angle ϕ is too big or too small, the d_{min} is the only minimal boundary or do not contribute on the minimal boundary. Different shapes of the area shows Figure 4.6.



(a) Top view

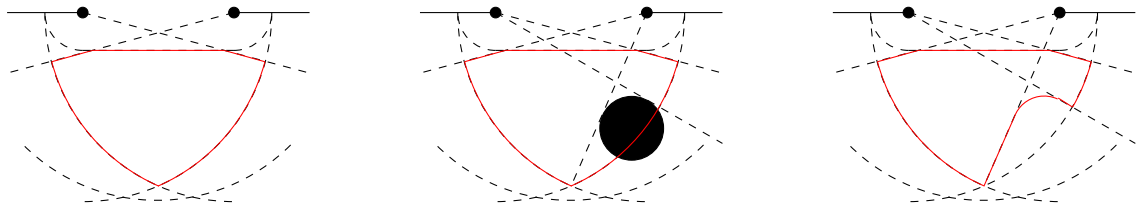
(b) Top view

(c) Front view

Figure 4.6: The area from where the missing data can be scanned with the constriction of $\phi = 30°$ (a). The area of the same hole but with the constriction $\phi = 15°$ (b). The intersection gives different shapes for different constrictions. The same hole is shown from the front view on (c). The dashed line shows the limitation shapes and the final areas are highlighted by the red line.

The data can be scanned from every point that belongs to the area gained by the intersection of the limitation shapes. The important assumption in this approach is that

there has to be no obstacle in the area. Figure 4.7 shows the area affection by the obstacle interfering it. The complication consists of the requirement that every point on the hole's rectangle boundary has to be reachable by the laser ray from the scanning point. This work does not deal with the obstacle complication and assume that there are not obstacles in the founded area.



(a) Top view (b) Top view (c) Top view

Figure 4.7: The original area (a), the area with the obstacle (b), where the new dash lines defines the boundary for the laser ray to reach the whole hole. The final area without the visualization of the obstacle is on (c).

4.3 Path planning for visiting scanning locations

The process of finding suitable area for the hole scanning is applied on every founded hole in the data set; each of hole's rectangles are considered as a hole. To collect a required scans all of the identified scanning areas have to be visited. The areas could have various sizes and they will be typically larger than the UAV. To choose where in the area the UAV has to move, the areas has to be sampled. The final task has to solve the problem of finding the best places in the areas to be visited to visit all the areas and fly the shortest way.

4.3.1 Identifying optimal sequence of areas

The problem of visiting founded areas to take some scan shots could be described as a *traveling salesman problem* (TSP) [20]. The main idea of the TSP lay in the visitation of all towns in the set exactly once. There is a set of towns, with the defined distances between them and the salesman's task is to visit every town exactly once, travel the shortest distance and return to the initial point [20]–[22]. TSP can be described as a graph problem, where the set of towns $\mathcal{T} = \{t_1, t_2, ...\}$ stands for the nodes of a graph $\mathcal{G} = (\mathcal{T}, \mathcal{R})$ and the set of routes $\mathcal{R}$ stands for edges with positive cost values. We assume that the distance from node n_1 to node n_2 has same value in both directions.

The solution of the TSP in our case will be the shortest path between the areas. But the TSP works with the areas as it would be a *single point* in the space. However, the area volume is typically bigger that only one point in the space and has some shape which defines where the UAV can pass-through, so the better approach is to find no only the path between the areas, but also the best place in the area, where the UAV should fly. This improvement could be solved as a *generalized traveling salesman problem* (GTSP).

The GTSP formulation extends the TSP by nodesets or clusters. GTSP is defined as a directed graph $\mathcal{G}$ with nodes $\mathcal{N}$, edges $\mathcal{E}$ with some cost c_{ij} for edge $e_{ij} \in \mathcal{E}$. Each node $n \in \mathcal{N}$

is part of some cluster/nodeset $\mathcal{S}_i, i = 1, 2, ..., m$. The solution of GTSP is then a cycle, which enters each cluster exactly ones [23]. The TSP can be perceived as a GTSP where each cluster $\mathcal{S}_i$ contains exactly one graph node $n \in \mathcal{N}$. The difference between the TSP and the GTSP shows figure 4.8.

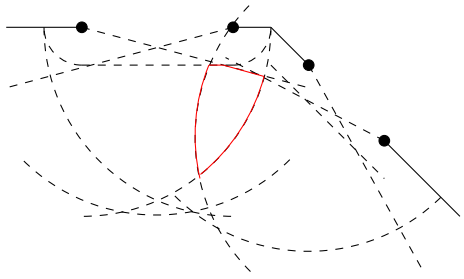
(a) (b)

Figure 4.8: The illustration of the solution of TSP task (a) and the GTSP task (b). The points with the same color belongs to the same cluster.

The task of finding the optimal places in the previously identified areas which should be visited is now considered as a GTSP problem. The areas are the GTSP clusters and the graph nodes will be represented by the points sampling the areas. The process of finding the areas sampling points is described in the next section.

4.3.2 Area sampling

The areas for collecting desired scans of a hole defines places where the UAV can collect desired data. We assume that there are no obstacles in this areas. In the case of two neighboring holes, moreover if planes of these holes constrict an acute angle, there could appear an overlaping of these areas. It means that there is an intersection area, where the conditions for scanning both holes are satisfied, i.e. the UAV can scan both holes from any point in that area and there is no need to visit both areas separately. The areas overlap is shown in Figure 4.9.

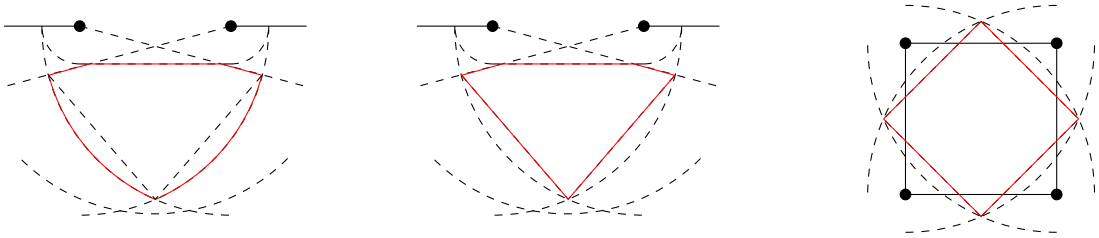


(a) Top view

Figure 4.9: The red line shows the area where two areas from where the missing data can be obtained overlap so the required data of both holes can be obtained from a single point instead of visiting two points, one in each area.

There are various methods of sampling some area in the 3D space, one of the simplest are random sampling or uniform sampling. In this work, the boundary sampling is used to avoid problems with the overlaping of the areas [24]. This approach is based on the fact that any path visiting any point inside the area has to go through or touch the area boundary. If there is no overlaping between some areas, the task remains the same; visit every area exactly once. Even if the optimal place to visit lay in the middle of the area, there is no way without crossing the area's boundary. If there is an overlap between some areas, the way still goes through the areas boundaries.

As shows Figure 4.6, the area's boundary contains arc shapes. One more simplification is used in this work, and it is the approximation of these arcs by another type of plane. Depending on d_{min}, d_{max} and ϕ the arcs will be more acute or more blunt and thus the approximation will be more precise or less. Figure 4.10 shows the approximated area and Figure 4.11 shows the final area which is sampled. The sampling density is different for different data sets and different sizes of the areas, based on the visual surveillance.



(a) Top view: original area

(b) Top view: approximated area

(c) Front view: approximated area

Figure 4.10

(a) Top view: approximated and sampled area

(b) Front view: approximated and sampled area

Figure 4.11

The points sampling the area are the graph nodes used as an input to the GTSP problem, where the areas correspoinding to the sampling points stands for the clusters. The GTSP problem is solved by the GLKH solver [25], which results in exactly one of the sampling points from each boundary to be visited.

4.4 Path planning

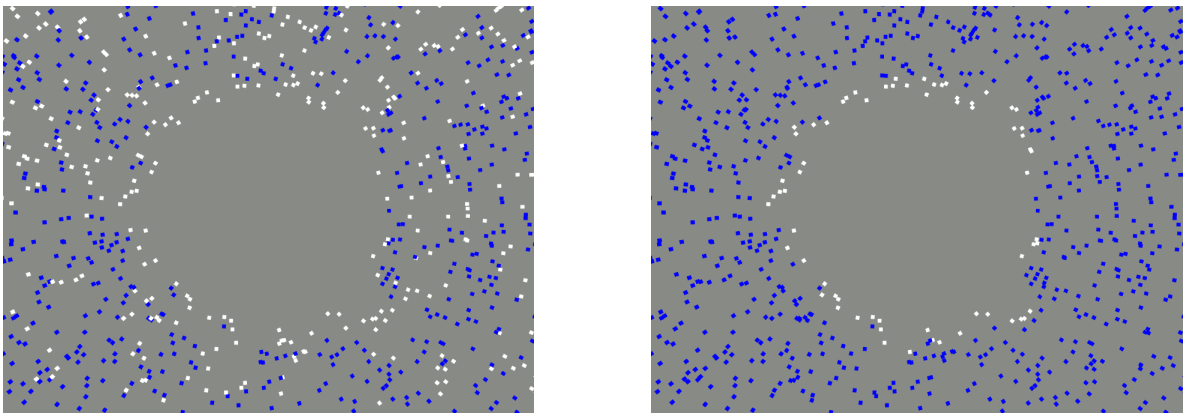
The solution of the GTSP with the input data in the form of suitable areas to take scan shots is a set $\mathcal{S}$ of points which has to be visited by the UAV. One part of the system of Multi-

Robot Systems (MRS) research group [17] is used to end our task by finding the appropriate path, which passes through all points $s \in \mathcal{S}$. The path is used to find the corresponding trajectory and the system [12] is used for the realistic simulation of the whole scanning process.

Chapter 5

Results

The three holes finding criteria were applied on various types of holes and data sets. The quality of the given result of each criterion is done by the shape of the hole, the noise participating on or near the boundary, the data set sampling density, and the size of the point's neighborhood $N_{\mathbf{p}}$. The values of the neighborhood's parameters K and ϵ define the foundable hole size [1]. The bigger values of these parameters, the larger holes could be founded. The impact of the neighborhood size is shown for each criterion on the simple circle hole in the plane with the nonuniform sampling density, i.e the points are sampled randomly in the plane. Figure 5.1 shows the angle criterion affection by the neighborhood size. The affection of the halfdisc criterion shows Figure 5.2 and Figure 5.3 shows the affection of the shape criterion.

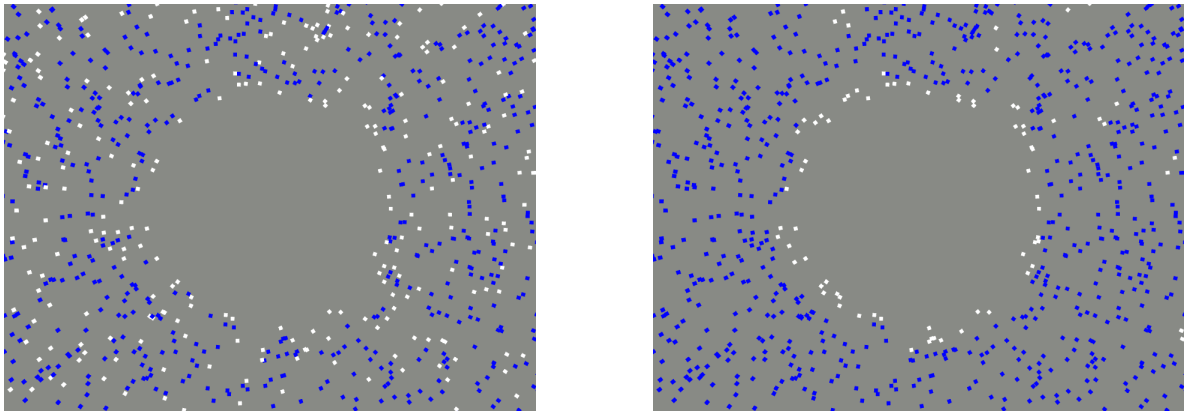


(a) (b)

Figure 5.1: The plane with a hole of a circle shape. The neighborhood $N_{\mathbf{p}}$, where the $K = 11$, $\epsilon = 0.05$, and the threshold $t = 0.4$, giving after applying the angle criterion the boundary points on (a). The boundary shown on (b) has the neighborhood parameters as follows; $K = 51$, $\epsilon = 0.05$, and $t = 0.4$. The interior points have the blue color while the white points stand for the boundary points, e.g. points, whose have the probability of being boundary points greater than the threshold t ($\Pi_{\angle}(\mathbf{p}) \geq t$).

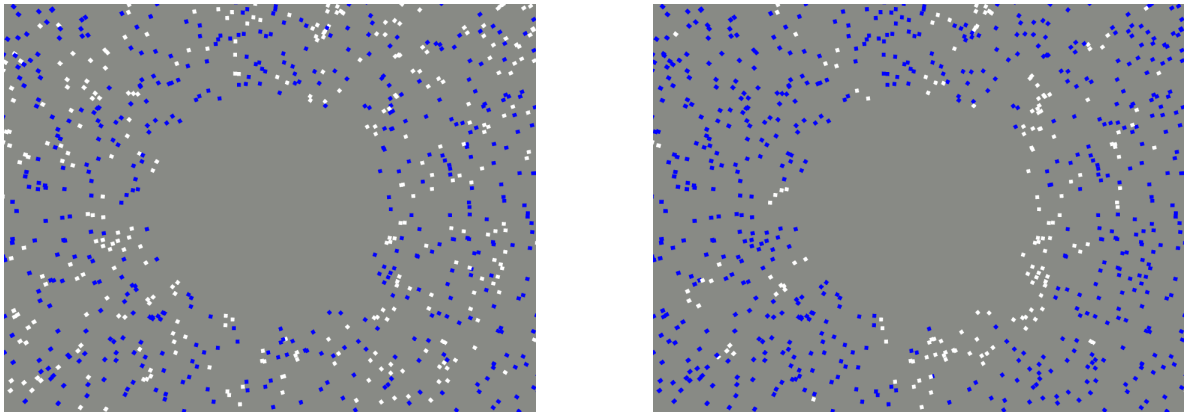
One interesting thing is that there is not only a difference between the number of points in the neighborhood of the same criterion but also between the number of points of the criteria themselves. To achieve similar results, the angle criterion needs a much smaller neighborhood than the halfdisc criterion needs, thus a good balance of the neighborhood size is essential for the criteria combination (as shown below). Figure 5.4 shows the usage of all three criteria and their weighted combination on the hole in the wall. This hole results from an occlusion caused by an obstacle during wall scanning. Compared to the randomly sampled plane in Figure 5.1,

the density of the points is much more uniform, which makes each criterion more stable. The threshold $t \in [0, 1]$ is often set to lower values, especially for the criteria combination. The combination needs the compromise between the neighborhood sizes and thus the quality and accuracy of each criterion is smaller, so the threshold t is better to be set lower.



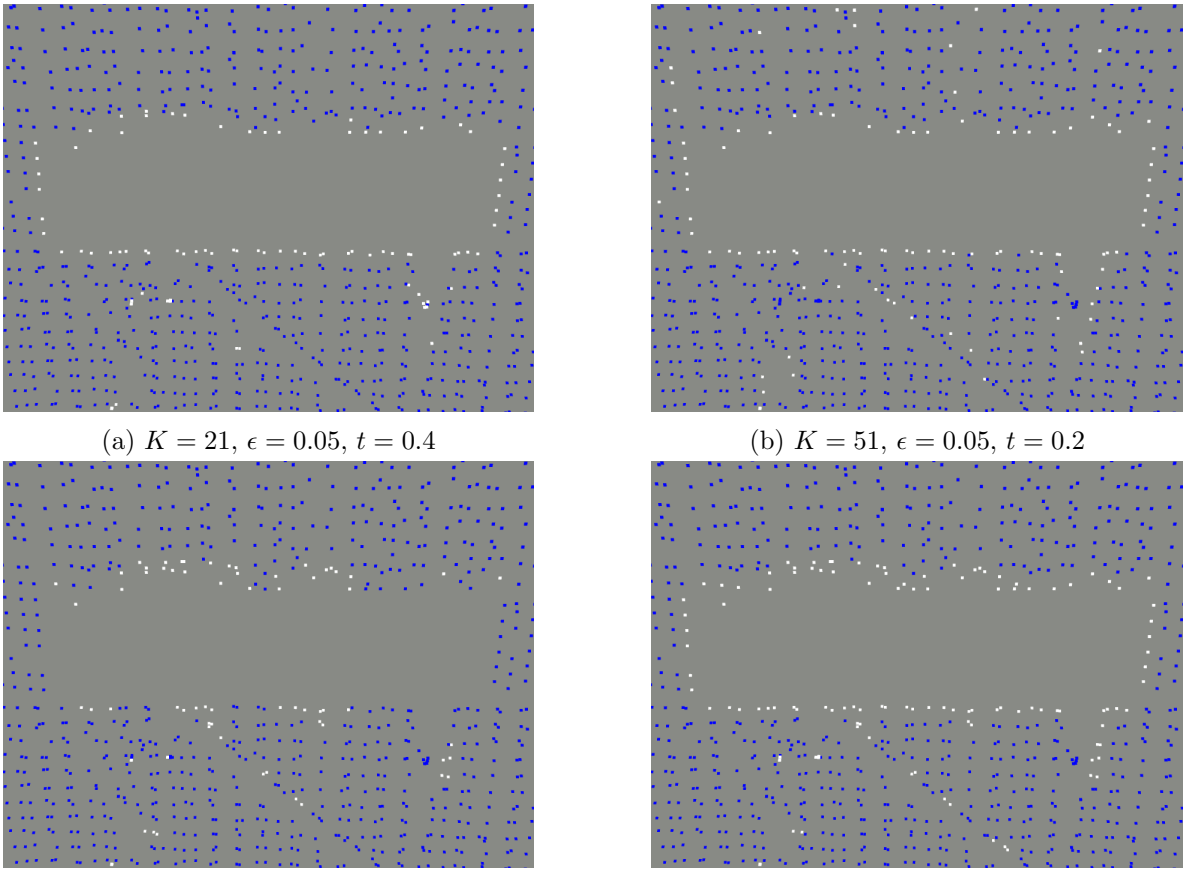
(a) (b)

Figure 5.2: The halfdisc criterion has different results based on the different neighborhood $N_{\mathbf{p}}$. The parameters of the neighborhood $N_{\mathbf{p}}$ are $K = 51$, $\epsilon = 0.05$ and the threshold $t = 0.2$ (a). (b) shows the result for $K = 201$, $\epsilon = 0.05$ and $t = 0.2$. The interior points are blue, the boundary points are white ($\Pi_{\mu}(\mathbf{p}) \geq t$).



(a) (b)

Figure 5.3: The shape criterion using the neighborhood $N_{\mathbf{p}}$, where $K = 21$, $\epsilon = 0.05$ and $t = 0.5$ (a). The neighborhood's parameters $K = 201$, $\epsilon = 0.05$, and $t = 0.5$ of the shape criterion result in (b). The interior points are blue, the boundary points are white ($\Pi_{\phi}(\mathbf{p}) \geq t$).



(a) $K = 21$, $\epsilon = 0.05$, $t = 0.4$

(b) $K = 51$, $\epsilon = 0.05$, $t = 0.2$

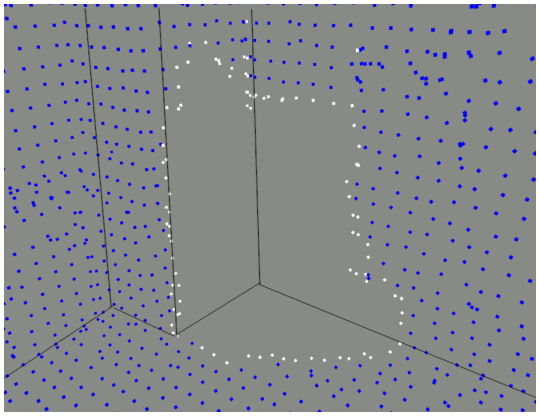
(c) $K = 101$, $\epsilon = 0.05$, $t = 0.2$

(d) $K = 91$, $\epsilon = 0.05$, $t = 0.2$

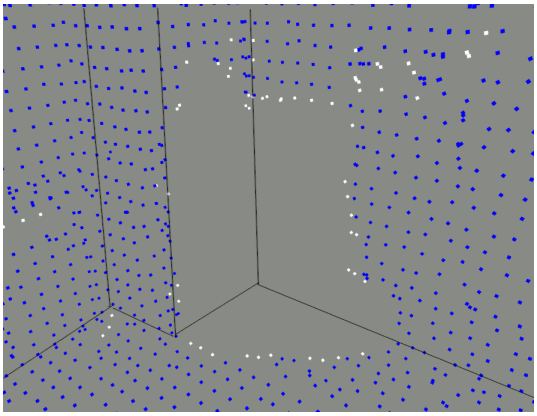
Figure 5.4: Boundary points after angle criterion (a), halfdisc criterion (b), shape criterion (c), and the combination of all three criteria (d), where the uniform weights $w_\angle = w_\mu = w_\phi = 0.333$ are used. The resulting boundary points are only after the criteria usage, the other improvements are not applied.

Figure 5.5 shows again the comparison between all three criteria and their combination. The hole is not in the planar point cloud now, but there is a missing corner in the point cloud data set. For better visual recognition of the 3D shape of the data on the image, black lines are added to the sharp edges. Because of many sharp edges and the need for the bigger values of K for halfdisc and shape criteria, the resulting boundary points better approximate the hole after the usage of angle criteria instead of the halfdisc or shape. These results of each criterion for this type of hole lead to the conclusion, that for the criteria combination the bigger value for $w_\angle$ compared to w_μ and w_ϕ works well.

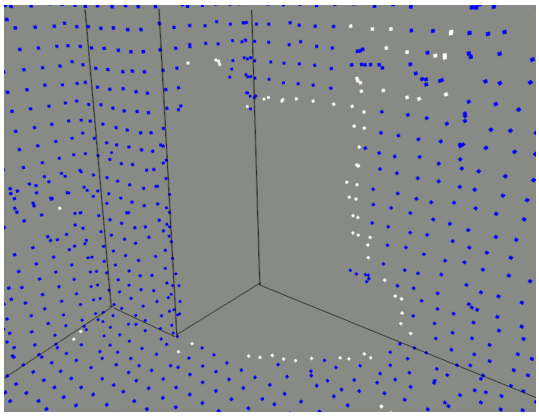
The previous figures show the points marked as boundary points after the usage of the three criteria but without the rest parts of the hole finding process. Figure 5.6 shows the boundary points after the whole process of hole finding, i.e. the improvement of the points' coherence and the loop extraction in the weighted graph is applied. The process is shown on the clear holes where the three criteria work well and where is no noise to disrupt the hole finding process.



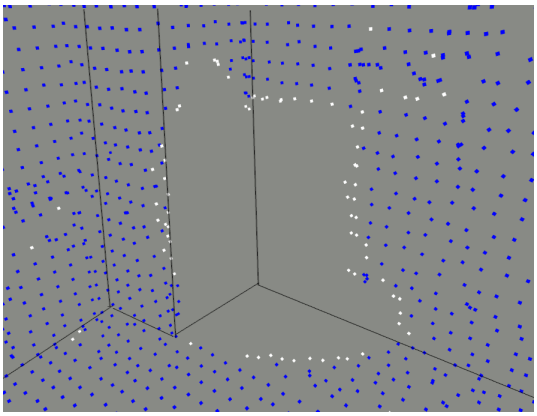
(a) $K = 21$, $\epsilon = 0.05$, $t = 0.4$



(b) $K = 91$, $\epsilon = 0.05$, $t = 0.2$

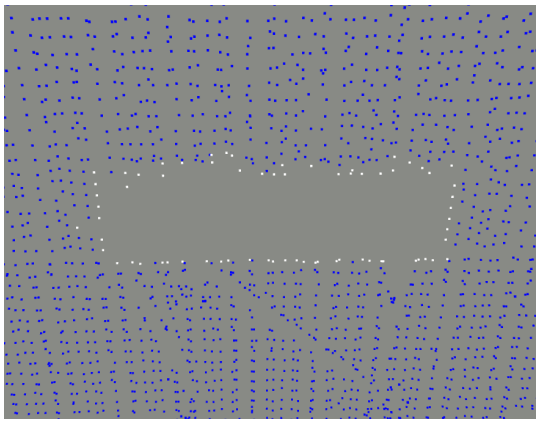


(c) $K = 91$, $\epsilon = 0.05$, $t = 0.2$

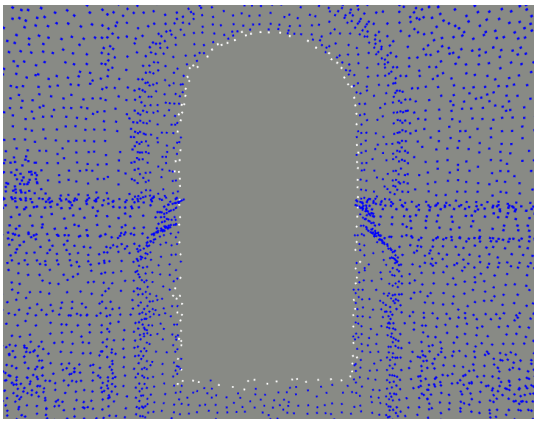


(d) $K = 71$, $\epsilon = 0.05$, $t = 0.3$

Figure 5.5: The usage of each criterion and their combination on the corner hole. The angle criterion's result (a), the halfdisc criterion's result (b), the shape criterion's result (c), and the combination of all three criteria results to (d). The weights $w_\angle = 0.5, w_\mu = w_\phi = 0.25$ were used. The black line shows the wall's sharp edges.



(a) $K = 71$, $\epsilon = 0.1$, $t = 0.2$

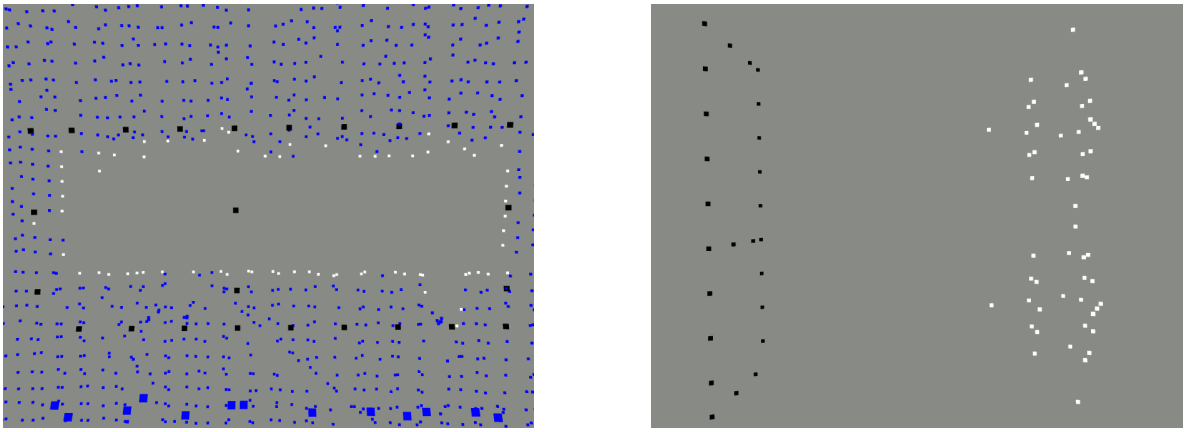


(b) $K = 31$, $\epsilon = 0.1$, $t = 0.2$

Figure 5.6: The weights $w_\angle = w_\mu = 0.5, w_\phi = 0$ are used to find the hole on (a), while the weights $w_\angle = 0.6, w_\mu = 0.4, w_\phi = 0$ are used on hole (b).

In Figure 5.7a is shown how the surrounding rectangle looks after the splitting to appropriate proportions. If the hole's approximation rectangle is not directly dividable by the smaller rectangles of the suitable proportions, the small rectangles around the margin which are smaller than the proportions are left smaller, thus the suitable proportions are the top restriction on the size. The sampled area from where the data can be scanned is shown in Figure 5.7b.

The holes finding approach used in this work works well for small amounts of similar holes, i.e the holes have similar neighborhoods, sizes, or shapes. It is hard to set the weights $w_\angle, w_\mu, w_\phi$ to proper values to find all holes in a point cloud. One weight combination works well for one hole but eliminates two other holes which are more susceptible to some criteria or neighborhood changes. The application on the point cloud data sets of the buildings' interiors showed that it is not as good as we have expected. The buildings' interiors often contain some noise from objects which affects the finding criteria significantly.



(a) Front view on the hole

(b) Top view, the same hole as on (a)

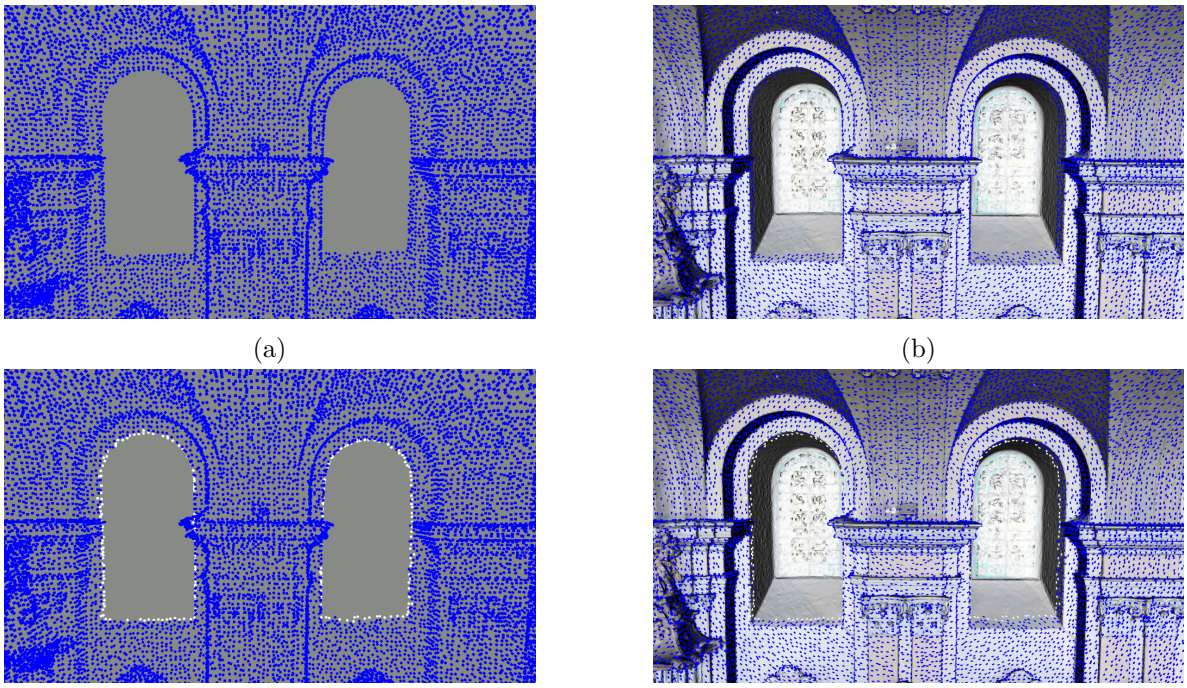
Figure 5.7: The rectangles splitting the hole's surrounding rectangle are marked as black points (a), while the sampled areas (right) for each small rectangle of the same hole (left) are shown in (b).

The verification of the approach was finally done by the simulation. The path for the UAV passing through the points found as a solution of the GTSP to take scan shots was found by the path planner [17] from the Multi-Robot Systems (MRS) research group at CTU. The simulation run in the MRS UAV system [12]. The simulation space was the model of the church's interior of Nový Malín municipality. The provided data was the unstructured point cloud of the Nový Malín church interior, with two holes shown in Figure 5.8. There was also a model of the same church, but not the point cloud, which was used during the simulation as a real environment.

The two point cloud holes were successfully detected and marked by the hole finding process described in the chapter 3. The size of the points neighborhood has been defined by the parameters' values as follows; $K = 31$, $\epsilon = 0.1$m, and $t = 0.2$m. The three criteria combination was used with the weights $w_\angle = 0.6, w_\mu = 0.4, w_\phi = 0$. The planes approximating the holes' boundary points were found by the RANSAC algorithm with the number of iterations $k = 1000$. According to the methods from chapter 4, the points in the inner space of the building which have to be visited by the UAV were successfully found. The final path was found

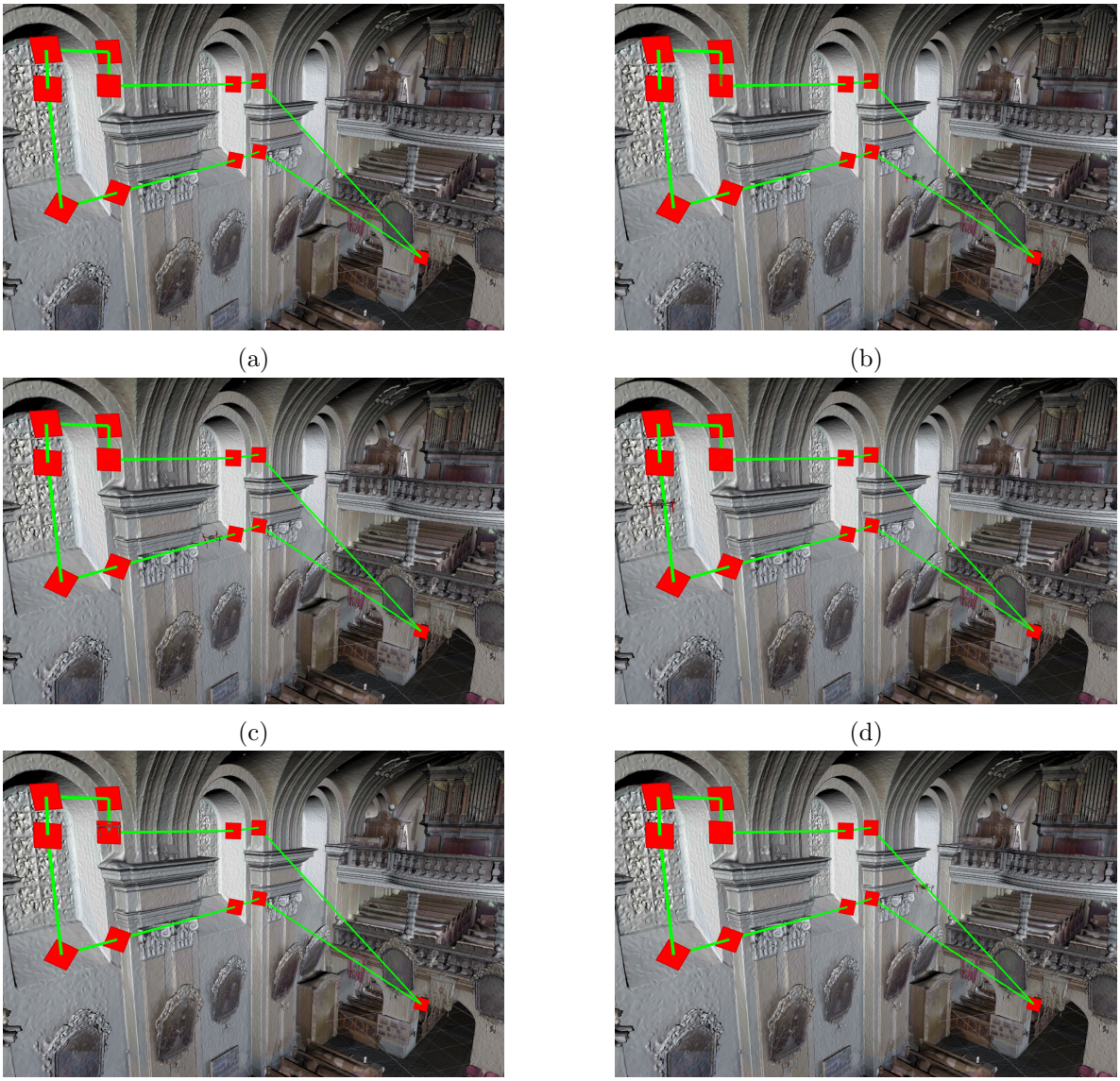
using the MRS planner [17] and the simulation has run with the help of the MRS system [12]. The simulation is illustrated by the images in Figure 5.9. The red points represent the previously founded points for taking scans, and the green line stands for the UAV trajectory. The interesting thing is that even if the holes look very similar, each of them has a different number of found scanning points. Because of the complexity and number of steps in the whole process of finding these points, starting with the hole detection and ending with the GTSP solving, there could be various reasons why the number of points differs. One reason could be the different approximating planes of each hole, which stands for hole representation during the final steps of finding these points. The constraints defining the areas from chapter 4 could be satisfied with the lesser number of points with the different approximation plane. Another reason may be caused by the small difference between the sizes of both holes.

The obtained missing data from the UAV sensor is shown in Figure 5.10. During the simulation, the ouster OS0-64 was used as a sensor. Even if the ouster has a scanning range bigger than the distance from the starting point to each waypoint, to scan all parts of the hole the waypoints had to be visited, and for purposes of the simulation and the provided approach testing, the maximum ouster range was reduced to 2m.



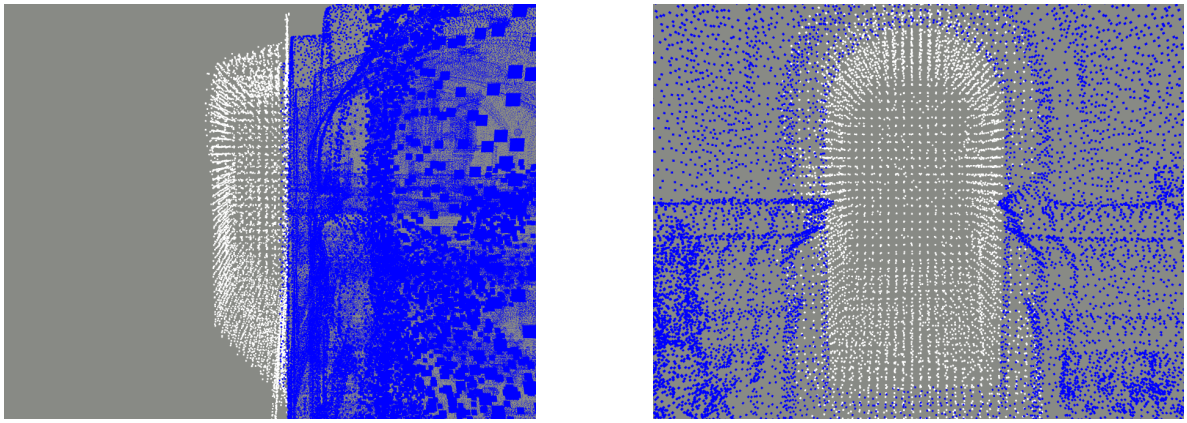
(a) (b) (c) (d)

Figure 5.8: The two holes in the unstructured point cloud of a Nový Malín church (a). The detected boundary points of the holes are shown in (c). The right column shows the same point cloud images but with the 3D model used during the simulation. The missing parts are visible in (b) and (d).



(a) (b) (c) (d) (e) (f)

Figure 5.9: The snapshots of the simulation. The image (a) shows the whole trajectory (green line) and the visiting points (red points). The two missing parts are the niches and the drone starting point is the red point in the bottom right corner. The rest of the snapshots shows the UAV flight following the green trajectory in order (b) to (f). When the UAV passes the red points, the scans are taken.



(a) Side view (b) Front view

Figure 5.10: The original point cloud with the holes is shown by the blue points. The white points show the scanned missing data during the simulation with the ouster sensor.

Chapter 6

Conclusion

The approach of finding holes in the unstructured point cloud and their filling afterward was implemented and verified in the realistic simulation. The hole finding process follows the approach from [1]. The approach was applied on provided point cloud data sets of Czech historical monuments. The scans of buildings' interiors are often complex and contain a lot of objects, which make hole detection much harder. The approach works well on planar objects with well recognizable holes, but the buildings' scans often contain more complex objects and holes.

The areas from where the missing data can be obtained were found by application of the UAV's onboard sensors limitations like the distance from the hole to the areas around the holes. A discrete sampling of these areas was done and the best area places to visit was solved as a GTSP. The approach was finally verified by a realistic simulation with the models of the Czech historical monuments. Thanks to the realistic simulation which verified the applied approach and the usage of the models of real buildings during the simulation, the generated trajectories are prepared for usage in real-world experiments.

Chapter 7

References

[1] G. H. Bendels, R. Schnabel, and R. Klein, "Detecting holes in point set surfaces," vol. 14, no. 1-3, 2006, ISSN: 1213-6964.

[2] C. C. You, S. P. Lim, S. C. Lim, J. S. Tan, C. K. Lee, and Y. M. J. Khaw, "A survey on surface reconstruction techniques for structured and unstructured data," in *2020 IEEE Conference on Open Systems (ICOS)*, 2020, pp. 37–42.

[3] S. Gumhold, X. Wang, and R. MacLeod, "Feature extraction from point clouds," *Proceedings of 10th international meshing roundtable*, vol. 2001, Nov. 2001.

[4] C. Mineo, S. G. Pierce, and R. Summan, "Novel algorithms for 3d surface point cloud boundary detection and edge reconstruction," *Journal of Computational Design and Engineering*, vol. 6, no. 1, pp. 81–91, Feb. 2018, ISSN: 2288-5048.

[5] V. S. Nguyen, T. H. Trinh, and M. H. Tran, "Hole boundary detection of a surface of 3d point clouds," in *2015 International Conference on Advanced Computing and Applications (ACOMP)*, 2015, pp. 124–129.

[6] S. Suer, S. Kockara, and M. Mete, "An improved border detection in dermoscopy images for density based clustering," in *BMC Bioinformatics*, 2011.

[7] A. Kazi, A. Sausthanmath, M. S M, S. V. Gurlahosur, and U. Kulkarni, "Detection of holes in 3d architectural models using shape classification based bubblegum algorithm," *Procedia Computer Science*, vol. 167, pp. 1684–1695, 2020, International Conference on Computational Intelligence and Data Science, ISSN: 1877-0509.

[8] T. Fiolka, F. Rouatbi, and D. Bender, "Automated detection and closing of holes in aerial point clouds using an uas," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, p. 101, 2017.

[9] P. Petracek, V. Kratky, T. Baca, M. Petrlik, and M. Saska, "New era in cultural heritage preservation: Cooperative aerial autonomy: Supervised autonomy for fast digitalization of difficult-to-access interiors of historical monuments," *IEEE Robotics & Automation Magazine*, pp. 2–19, 2023.

[10] V. Krátký, P. Petráček, V. Spurný, and M. Saska, "Autonomous reflectance transformation imaging by a team of unmanned aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2302–2309, 2020.

[11] V. Krátký, P. Petráček, T. Nascimento, M. Čadilová, M. Škobrtal, P. Stoudek, and M. Saska, "Safe documentation of historical monuments by an autonomous unmanned aerial vehicle," *ISPRS International Journal of Geo-Information*, vol. 10, no. 11, 2021, ISSN: 2220-9964.

[12] T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, and M. Saska, "The mrs uav system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, 2021.

[13] D. Hert, T. Baca, P. Petracek, V. Kratky, V. Spurny, M. Petrlik, M. Vrba, D. Zaitlik, P. Stoudek, V. Walter, P. Stepan, J. Horyna, V. Pritzl, G. Silano, D. Bonilla Licea, P. Stibinger, R. Penicka, T. Nascimento, and M. Saska, "Mrs modular uav hardware platforms for supporting research in real-world outdoor and indoor environments," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2022, pp. 1264–1273.

[14] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[15] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, 509–517, 1975, ISSN: 0001-0782.

[16] J. Demel, *Grafy a jejich aplikace*. Academia, 2002, 2015, 2019.

[17] V. Krátký, P. Petráček, T. Báča, and M. Saska, "An autonomous unmanned aerial vehicle system for fast exploration of large complex indoor environments," *Journal of Field Robotics*, vol. 38, no. 8, pp. 1036–1058, 2021.

[18] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, 381–395, 1981, ISSN: 0001-0782.

[19] L. Balková, *Lineární algebra 1*, 2013. [Online]. Available: `https://kmlinux.fjfi.cvut.cz/~balkolub/Vyuka/skripta/main.pdf`.

[20] Y. Xu and C. Che, "A brief review of the intelligent algorithm for traveling salesman problem in uav route planning," in *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 2019, pp. 1–7.

[21] M. Jünger, G. Reinelt, and G. Rinaldi, "Chapter 4 the traveling salesman problem," in *Network Models*, ser. Handbooks in Operations Research and Management Science, vol. 7, Elsevier, 1995, pp. 225–330.

[22] A. Talska, "Problém obchodního cestujícího, aplikace v dopravních a logistických systémech," Bachelor's Thesis, 2019.

[23] C. E. Noon and J. C. Bean, "An efficient transformation of the generalized traveling salesman problem," The University of Michigan, Tech. Rep., 1989.

[24] P. Maini, P. Tokekar, and P. B. Sujit, "Visual monitoring of points of interest on a 2.5d terrain using a uav with limited field-of-view constraint," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 6, pp. 3661–3672, 2021.

[25] K. Helsgaun, "Solving the equality generalized traveling salesman problem using the lin–kernighan–helsgaun algorithm," *Mathematical Programming Computation 7*, 269–287, 2015.