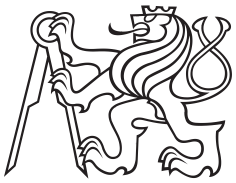
Bachelor's Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Door Detection on Monocular Camera Data

Nikita Kisel

Supervisor: RNDr. Petr Štěpán, Ph.D
Study Program: Open Informatics
Specialisation: Artificial Intelligence and Computer Science
May 2024

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Kisel Nikita** Personal ID number: **507403**

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:

Door Detection on Monocular Camera Data

Bachelor’s thesis title in Czech:

Detekce dveří na datech z monokulární kamery

Guidelines:

1) Learn about the ZoeDepth neural network that estimates the distance of objects in an image.
2) Test the results of this neural network on data containing doors.
3) Create a door detection algorithm for drone passage based on data from the ZoeDepth neural network and original camera data.
4) Test the algorithm on the dataset and evaluate the detection success and accuracy.

Bibliography / sources:

[1] A. Spournias, C. Antonopoulos, G. Keramidas, N. Voros and R. Stojanović, "Enhancing Visual Recognition for Door Status Identification in AAL Robots via Machine Learning," 2020 9th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 2020, pp. 1-6, doi: 10.1109/MECO49872.2020.9134108.
[2] A. Llopart, O. Ravn and N. A. Andersen, "Door and cabinet recognition using Convolutional Neural Nets and real-time method fusion for handle detection and grasping," 2017 3rd International Conference on Control, Automation and Robotics (ICCAR), Nagoya, Japan, 2017, pp. 144-149, doi: 10.1109/ICCAR.
[3] J. G. Ramôa, L. A. Alexandre and S. Mogo, "Real-Time 3D Door Detection and Classification on a Low-Power Device," 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Ponta Delgada, Portugal, 2020, pp. 96-101
[4] M. Sonka, V. Hlavac, and R. Boyle, „Image processing, analysis and machine vision“, Springer, 2013.

Name and workplace of bachelor’s thesis supervisor:

RNDr. Petr Štěpán, Ph.D. Multi-robot Systems FEE

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

Date of bachelor’s thesis assignment: **31.01.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

RNDr. Petr Štěpán, Ph.D.	prof. Dr. Ing. Jan Kybic	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

Acknowledgements

I would like to express my sincere gratitude to my work leader, RNDr. Petr Štěpán, Ph.D, for his invaluable guidance and support throughout this research.

I have used ChatGPT for text rephrasing and Grammarly for ensuring grammatical accuracy.

Lastly, I am grateful to CTU for giving me such a good *alma mater*.

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 the preparation of university theses.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

In Prague, 24. May 2024

..... Nikita Kisel

Abstract

Doors serve as important landmarks within spaces, facilitating the connection of separate areas and providing access through navigational manoeuvres. Consequently, detecting doors is becoming a key tool for solving various navigation and wayfinding problems. Such applications range from helping visually impaired people navigate various rooms to deployment in autonomous robotic navigation systems. This research focuses on using door detection for the navigation of autonomous drones. The drone is expected to be very small, so a Micro Aerial Vehicle (MAV) is considered. By specifying this domain, places inaccessible to humans can be reached. The question of this work is whether it is possible to accurately detect open doors in unfamiliar indoor environments, having a MAV with only a monocular camera and relying on off-board computational resources (a constraint imposed due to MAV's cargo capacity). This study presents an original approach for detecting open doors and their doorways in unknown indoor environments for autonomous robot navigation or any other appropriate use. The proposed algorithm is unique due to its integration of modern techniques with the highest precision, which involves combining depth and colour data with the state-of-the-art neural network of single-image depth estimation. To test the robustness and generalisability of this approach, the algorithm is thoroughly evaluated on the collected dataset.

Keywords: door detection, monocular depth estimation, autonomous robot navigation

Supervisor: RNDr. Petr Štěpán, Ph.D
KN:E-116,
Resslova 307/9,
Praha 2

Abstrakt

Dveře slouží jako důležité orientační body v prostorech, usnadňují spojení oddělených oblastí a poskytují přístup prostřednictvím navigačních manévrů. V důsledku toho se detekce dveří stává klíčovým nástrojem pro řešení různých problémů s navigací a hledáním cesty. Tyto aplikace se pohybují od pomoci lidem se zrakovým postižením orientovat se v různých místnostech až po nasazení v autonomních robotických navigačních systémech. Tento výzkum se zaměřuje na využití detekce dveří pro navigaci autonomních dronů. Očekává se, že dron bude velmi malý, takže se uvažuje o Micro Aerial Vehicle (MAV). Zadáním této domény se lze dostat na místa nepřístupná lidem. Otázkou této práce je, zda je možné přesně detekovat otevřené dveře v neznámých vnitřních prostředích, mít MAV pouze s monokulární kamerou a spoléhat se na externí výpočetní zdroje (omezení způsobené kapacitou nákladu MAV). Tato studie představuje originální přístup k detekci otevřených dveří a jejich vchodů v neznámých vnitřních prostředích pro autonomní navigaci robotů nebo jakékoli jiné vhodné použití. Navržený algoritmus je jedinečný díky integraci moderních technik s nejvyšší přesností, která zahrnuje kombinaci hloubkových a barevných dat s nejmodernější neuronovou sítí odhadu hloubky jednoho snímku. Pro testování robustnosti a zobecnitelnosti tohoto přístupu je algoritmus důkladně vyhodnocen na shromážděných datech.

Klíčová slova: detekce dveří, monokulární odhad hloubky, autonomní navigace robota

Contents

List of Abbreviations 1

1 Introduction 3

1.1 Statement of Contributions 4

2 Related Work 5

2.1 Detailed Method Comparison . . . 5

3 Depth Estimation Method 11

3.1 Method Choice 11

3.1.1 Models Configuration 11

3.1.2 Model Choice 12

4 Door Detection 13

4.1 Definition of the Door Geometry Model 13

4.2 Line Detection 14

4.2.1 Depth Estimation 14

4.2.2 Gradient Computation 15

4.2.3 Segment Extraction 16

4.2.4 Clustering 18

4.2.5 Prototyping 21

4.3 Door Model Assembly 22

4.3.1 Edge Grouping 23

4.3.2 Model Construction 24

4.4 Model Validation 28

4.4.1 Geometric Validation 28

4.4.2 Edge Validation 29

4.4.3 Door Model Rectangle Validation 29

4.4.4 Doorway Integration 30

4.4.5 Doorway Assembling 30

4.4.6 Door Filtering 31

5 Results 33

5.1 Dataset Description 33

5.2 Performance Evaluation Metrics 34

5.3 Results Analysis 35

6 Conclusion 39

6.1 Summary of Key Findings 39

6.2 Limitations of the Study 40

6.3 Future Work 40

6.4 Contributions to the Field 41

6.5 Final Remarks 41

A Attachments **43**

B Bibliography **57**

Figures

3.1 Images processed using different ZoeDepth models. 12

4.1 Outcome of the ZoeDepth applied to images. 15

4.2 Estimated colour gradient image. 15

4.3 Estimated depth gradient image. 16

4.4 Output of the Hough Line Transform. 17

4.5 Output of the Probabilistic Hough Line Transform. 17

4.6 Output of the LSD. 18

4.7 Output of the ELSED. 18

4.8 Result of filtering based on line angles. 19

4.9 Distance measurement methods. 20

4.10 Result of the adapted DBSCAN algorithm. 21

4.11 Lines fitted by the RANSAC algorithm. 22

4.12 Top edges identified. 23

4.13 Left and right edges identified. 24

4.14 Upper edge search boundaries . 25

4.15 Possible upper edge positions. . 26

4.16 Determining corners through intersection line calculations. 28

4.17 Doorway assembly 31

5.1 Algorithm's high-performance output. 36

5.2 Showcase of the algorithm's incapability to recognize doors. . . . 36

5.3 Algorithm's prediction of doors different from their doorways. . . . 37

5.4 Algorithm's output for partially overlapped doors. 37

5.5 Algorithm's outcome for True Negative data. 38

A.1 DBSCAN results for 1st test. . . 44

A.2 DBSCAN results for 2nd test. . . 45

A.3 DBSCAN results for 3rd test. . . 46

A.4 DBSCAN results for 4th test. . . 47

A.5 DBSCAN results for 5th test. . . 48

A.6 Algorithm's prediction for the 2nd image from the dataset 49

A.7 Algorithm's prediction for the 3rd image from the dataset 49

A.8 Algorithm's prediction for the 5th image from the dataset 50

A.9 Algorithm's prediction for the 6th image from the dataset 50

A.10 Algorithm's prediction for the 8th image from the dataset 51

A.11 Algorithm's prediction for the 9th image from the dataset 51

A.12 Algorithm's prediction for the 12th image from the dataset 52

A.13 Algorithm's prediction for the 13th image from the dataset 52

A.14 Algorithm's prediction for the 14th image from the dataset 53

A.15 Algorithm's prediction for the 16th image from the dataset 53

A.16 Algorithm's prediction for the 17th image from the dataset 54

A.17 Algorithm's prediction for the 18th image from the dataset 54

A.18 Algorithm's prediction for the 19th image from the dataset 55

Tables

2.1 Door detection methods comparison 6

4.1 DBSCAN modification comparison. 21

5.1 Detection evaluation results. . . . 35

5.2 Results of the proposed algorithm. 38

List of Abbreviations

AI	Artificial Intelligence
CNN	Convolutional Neural Network
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
ELSED	Enhanced Line SEgment Drawing
IoU	Intersection over Union
LSD	Line Segment Detection
MAV	Micro Aerial Vehicle
MDE	Metric Depth Estimation
ML	Machine Learning
NN	Neural Network
RANSAC	RANdom SAmple Consensus
RDE	Relative Depth Estimation
ROI	Region Of Interest
SIDE	Single Image Depth Estimation
SLAM	Simultaneous Localisation And Mapping

Chapter 1

Introduction

Correct interaction with an environment leads to safe and efficient autonomous robot navigation. It can be employed in different modern high-end tasks (e.g. exploration, inspection and mapping). Navigating within an unfamiliar indoor environment is a complex issue that includes various tasks such as localisation, mapping, simultaneous localisation and mapping (SLAM), path planning, and also object recognition. This research primarily concentrates on the detection of open doors, a critical aspect in facilitating autonomous robot navigation. Doors serve as key transition points demarcating distinct sub-areas within a given spatial environment. Accurate door detection facilitates robots' adaptive navigation in unfamiliar and potentially dynamic environments in real-time.

Selection of a Micro Aerial Vehicle (MAV) as a camera platform offers many benefits, including increased manoeuvrability, reduced visibility, and comparatively lower costs. However, these drones also have certain disadvantages due to their miniature size (typically <100g), which limits their payload to just grams. Consequently, only conventional monocular cameras can be placed on board, and more advanced technologies such as lidar or 3D depth cameras are not compatible with devices of this class due to their large size. This constraint places significant limitations on the methods applicable to door detection. Additionally, such a drone cannot have a lot of computing power, so all operations must be carried out in real-time on some powerful system. It could be some kind of desktop, cloud or something else. Using external computing power allows resource-intensive methods to be used and even combined to achieve better results. Thereby the utilisation of a resource-intensive neural network for single-image depth estimation can be incorporated as a component within the algorithm.

The rest of the work is organised as follows. Chapter 2 provides an overview of state-of-the-art door detection methods. The selected single image depth estimation method is presented in Chapter 3. Chapter 4 presents the door detection algorithm. The estimate obtained by the method applied to the collected dataset is presented in Chapter 5. Chapter 6 presents the conclusion and possible future improvements of the method. Appendix A contains attachments.

1.1 Statement of Contributions

Although the subject of door detection has been considered in previous studies, this paper proposes a unique approach that:

1. Combines RGB image with state-of-the-art depth estimation technique.
2. Provides a comprehensive solution for open door detection, including doorway detection mechanisms.
3. Does not require pre-scanning of the area.

In the subsequent chapters, it will be demonstrated how the proposed algorithm combines the various aspects delineated in points 1-3.

Chapter 2

Related Work

The problem of door detection within indoor settings is not new today. There are two main areas of door recognition: firstly, within the realm of wayfinding systems designed to aid individuals with visual impairments, and secondly, in the extensive domain of robotic navigation with social or assistive applications. Methodologies employed for door recognition can typically be classified based on the dimensional scope of data utilised as input. Predominantly, such data comprises either 2D RGB images or 3D data, supplemented by depth information corresponding to each pixel or region of the image (e.g. Lidar, depth camera or laser may be employed). Methods can also be categorised based on their classification capabilities, as not all methods are designed to distinguish between closed, open and semi-open doors.

2.1 Detailed Method Comparison

Table 2.1 provides a summary of cutting-edge research in computer vision-based systems dedicated to door detection.

Starting a detailed analysis with methods utilising both RGB data and depth data, several fundamental techniques can be identified. Foremost among these is Machine Learning (ML), representing a state-of-the-art approach deployed in [1], [4], [5], [6] works.

Method	Input data		Applicability			Result
	RGB	RGB + depth info	Closed	Open	Semi-Open	
[1]		X	X	X		F_1 - 96.53%
[2]	Depth only		X	X	X	-
[3]		X	DND			acc. - 91%
[4]		X	X	X		-
[5]		X	X	X	X	acc. - 90.9%
[6]		X	X	X	X	F_1 - 99.05% prec. - 98.6%
[7]		X	DND			rec. - 90%
[8]		X		X		-
[9]	X		DND			F_1 - 86.9% prec. - 97.2%
[10]	X			X		prec. - 97.46%
[11]	X		DND			prec. - 89.4%
[12]	X		DND			rec. - 83%
[13]	X		X			prec. - 97.18%

Table 2.1: Door detection methods comparison.
DND - Does not distinguish, acc. - accuracy, prec. - precision, rec. - recall.

In the most recent work [1], authors based their technique on the deep machine learning system YOLO [14], with an adopted specific model YOLOv8. Two approaches have been tried. Firstly, the model pre-trained on the COCO dataset [15] was taken and then fine-tuned on the Roboflow doors dataset [16]. It showed satisfactory performance working with real data, but its effectiveness with the Habitat simulator [17] was found to be insufficient. Therefore, the second model was trained on the dataset made from Habitat Simulator scenes. Consequently, an impressive F_1-score 96.53% is achieved when evaluating the algorithm on Habitat Simulator scenes. This method demonstrates good performance, but the training and testing data in the simulator are shown to differ from the real data.

In the paper [4] authors used Faster R-CNN Inception [18] the v2 model, which underwent training using a dataset compiled from web images, each manually labelled by hand. The Softmax function is utilised to map the output probability distribution to predicted output classes. A similar approach was employed in the work [5]. The algorithm uses PointNet classifier [19] for 3D door detection. Despite studies [4], [5] show good results, they both rely on a large RGB-D camera.

Additionally, modern techniques such as Artificial Intelligence (AI) have been employed in door detection problems. In the work [7] authors used Convolutional Neural Networks (CNN). The algorithm begins its operation by

focusing on the retrieval of a region of interest (ROI) through the employment of CNN. Following this initial step, it proceeds to visual segmentation and planar model extraction. Finally, the algorithm integrates fusion techniques, derives key features, and extracts handle grasping, as the objective is to identify the handle on the door. The algorithm under consideration exhibits interest, although its applicability to the autonomous navigation of drones appears limited. This constraint arises from its primary focus on handle detection, rather than the precise identification of doors and doorways, which are essential for effective navigation in such contexts.

Proceeding with techniques that incorporate geometric features of the doors, such as their quadrangular shape. These attributes may be derived from the image data through a combination of vertical and horizontal lines or corners, augmented by depth information obtained from the sensor. This technique is used in studies [2], [6] and [8]. The majority of these algorithms necessitate a pre-scanning of the environment, as they rely on point cloud maps.

The methodology outlined in the referenced study [2] starts with the plane extraction using depth data segmentation, based on RANdom SAmple Consensus (RANSAC) [20] reconstruction. Subsequently, the corner points of the planes are calculated using the edges associated with each plane. This information is then merged with feature constraints to categorise both door and wall components. Then straight lines are fitted based on the projection on the disordered and discontinuous boundary points. Finally, the door state is determined by analysing the angle formed between the straight lines of the wall and the door. As a result, the algorithm is solely dependent on depth data and shows commendable performance. However, the potential benefits of integrating RGB data with a similar approach can be considered.

The work [6] combines RGB data with depth information obtained from the scanned environment. This methodology comprises two primary stages: initial identification of door openings followed by the subsequent detection of doors in general. In the process of detecting door openings, a trinary orthoimage is generated and utilised for voxel clustering. This involves constructing all possible rectangles from two horizontal and two vertical lines, subsequently identifying the optimal candidate by assessing the largest number of voxel overlaps within each cluster. Following this step, the door is categorised as either open or semi-open based on an angle, which is determined by taking a horizontal half-elevation section of the door data and identifying the line that best fits the points on the door leaf using RANSAC. For door detection, gradient images derived from both the RGB image and depth map are combined using the bitwise OR operator. Subsequently, vertical and horizontal lines are detected using a lateral histogram algorithm [21]. Further, all possible

rectangles are generated as previously described. Subsequently, the algorithm evaluates each rectangle against various criteria, including colour and depth consistency, door frame occlusion, wall overlap, and the determination of the smallest rectangle satisfying the above conditions. Therefore, using this methodology, the algorithm achieves one of the best results observed in recent years. The limitation arises in that it still requires pre-scanning the area to create the point cloud.

In the study [8] authors used a RANSAC-based algorithm for the plane detection, akin to the approach outlined in [2]. Following plane detection, all empty regions are checked as potential candidates for door openings. To ensure that the found region is a real door opening, a pre-trained detector based on Aggregate Channel Features [22] is used. This detector employs a sliding window technique on regions within the RGB image that correspond to the identified candidate openings. If a door is found within this area, it is annotated as such in both the depth and RGB images. This algorithm is trivial, its limitation being its ability to solely identify open doors.

In the work [3], authors have integrated two methodologies. Initially, the algorithm employs multi-layer thresholding to convert a point map into a roaster depth map. Subsequently, door candidates are selected from segments with local minimum elevations. Following this, the section of the point cloud corresponding to the potential door is isolated and examined by applying component histograms to its coordinates, determining the presence of a door. Although this method is highly accurate, it does not address the detection of semi-open doors.

The second part of the table presents a comparison among methods exclusively reliant on 2D image processing. Modern approaches in the field such [9], [10] and [13] are also based on machine learning.

The latest study [9] adopts MobileNet-SSD v2 model [23] in a TensorFlow environment. Training and evaluation were carried out on the dataset created from proprietary and web images, which were manually labelled. The primary objective was to optimise the method for lightweight execution, suitable for deployment on resource-constrained wearable devices. This approach achieves a high F_1-score 86.9%. Nevertheless, this approach prioritises the detection of doors in a broad sense, rather than pinpointing their precise locations.

In the work [10], authors trained a CNN model on the collected SRIN dataset [24]. The CNN, in this context, distinguishes the presence or absence of a doorway behind the robot. A notable aspect of this study is the use of door detection to facilitate autonomous robot navigation. This is achieved by

estimating the depth of an image using the Depth Dense network [25]. The algorithm implemented in this study incorporates a module for selecting a pixel within the door region, identified as the area with the greatest depth in the image, to compute the subsequent rotation angle of the robot. After angle determination, fundamental action primitives are executed. It is noteworthy that this algorithm operates exclusively with open doors, predicated on the assumption that the door is situated within the region of maximum depth. Enhancements to this algorithm could be added by integrating predicted depth within the CNN processing pipeline.

The researchers cited in the study [13] additionally undertook training CNN employing a substantial dataset comprising 20,500 images. Taking into account the time context of the publication of the study in 2014, it is worth noting that no machine learning model has been adopted as the basis for CNN. The method has a favourable outcome, achieving a precision rate of 97.18% in door detection. However, it is noteworthy that the model is restricted to detecting closed doors exclusively.

Studies [11], [12] are based on segmentation of door geometry features.

In the work [11], a standardised door model is defined. The methodology begins by employing the Line Segment Detection (LSD) [26] algorithm for line detection. Subsequently, the algorithm initiates the search for two vertical line segments and a head line segment. Finally, all potential door candidates undergo scoring and filtering processes. The algorithm proposed in the study [12] is based on edge and corner detection. Initially, all edges are identified using the Canny edge detector [27], and a search space for corners is constructed. Subsequent to corner detection, lines are generated by connecting pairs of corners with the edge map. Only lines surpassing a predetermined threshold are considered for polygon construction. Finally, the search space for quadrilateral polygons, formed from the identified lines, is matched with the standard geometric model of a door. It should be noted that algorithms [11], [12] might prove ineffective, if certain parts of the door's edges or corners are obstructed, making it challenging to acquire the accurate geometric representation of the door.

Summarising all the algorithms discussed above, it becomes evident that methods reliant on the processing of RGB-D data exhibit superior performance in door detection. However, such methods typically necessitate pre-scanning of the environment. To date, no algorithm using 2D input data demonstrates the capability to classify all possible door states. The proposed approach aims to use efficient techniques using 3D data as input. Yet, it operates exclusively with conventional 2D images, emulating the depth dimension through state-of-the-art depth estimation methodology. This approach aims

to detect open doors and their doorways.

Chapter 3

Depth Estimation Method

As shown in the previous chapter, the most effective door detection methodologies include additional depth data to increase the accuracy of door detection. As originally envisioned, the proposed method also uses depth data to achieve better results, this data is estimated from a 2D RGB image by applying a modern Neural Network (NN).

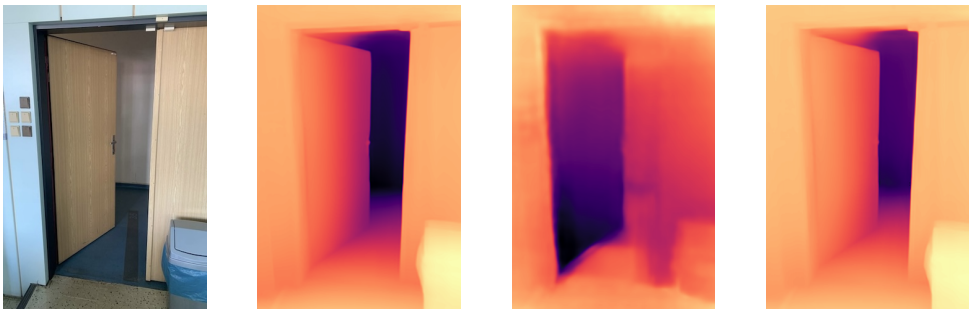
3.1 Method Choice

ZoeDepth Network [28] has been selected as the Single Image Depth Estimation (SIDE) method. The reason for this choice is that it is the most modern and currently one of the most effective SIDE methods available.

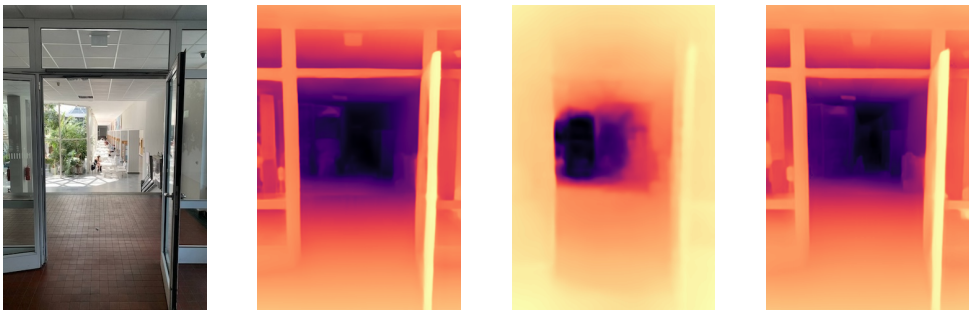
3.1.1 Models Configuration

ZoeDepth Network has three main model configurations for Relative Depth Estimation (RDE) & Metric Depth Estimation (MDE). The first configuration is ZoeD-M12-N, which is pre-trained on 12 datasets for relative depth and then metric fine-tuned on the NYU Depth v2 dataset. The second one is ZoeD-M12-K, which is pre-trained on 12 datasets for relative depth as the previous one and then metric fine-tuned on the KITTI dataset. The third

and flagship architecture is ZoeD-M12-NK. It is relatively pre-trained on 12 datasets combined with metric fine-tuning on indoor and outdoor datasets (NYU Depth v2 and KITTI). Figure 3.1 illustrates the results of applying a neural network to images containing doors.



(a) : RGB **(b) :** Zoe-M12-N **(c) :** Zoe-M12-K **(d) :** Zoe-M12-NK



(e) : RGB **(f) :** Zoe-M12-N **(g) :** Zoe-M12-K **(h) :** Zoe-M12-NK

Figure 3.1: Images processed using different ZoeDepth models.

3.1.2 Model Choice

As shown above, the ZoeD-M12-K tuned on the outdoor dataset performs relatively poorly on the indoor samples. Given the objective of identifying doors within indoor settings, the selection process is limited to two models: ZoeD-M12-N and ZoeD-M12-NK. Notably, the flagship model, ZoeD-M12-NK, distinguished by its superior MDE, has been selected for depth estimation within the proposed methodology. This decision is based on the necessary reliance of the proposed algorithm on both RDE and MDE accuracy.

Chapter 4

Door Detection

The proposed algorithm for door detection aims to identify open doors across various configurations. It relies on geometric features inherent to the door, particularly focusing on the door shape and depth consistency within the door opening region. In contrast to algorithms solely reliant on detecting a geometric door model within RGB images, the proposed approach acknowledges the potential overlap of certain edges. Respectively, to enhance robustness, it concurrently operates on both RGB and computed depth data. The door detection algorithm includes three primary stages: line detection, assembly of the door model, and model validation.

4.1 Definition of the Door Geometry Model

Since the proposed approach is based on the geometric features of the door, the door model should be determined by taking into account several assumptions:

1. At least 1 segment of the vertical edge of the door outline is visible in the depth gradient image. This is valid for at least 2 vertical edges of the door.
2. At least 1 segment of the horizontal upper edge of the door is visible in the depth gradient image.
3. Vertical edges of doorframes are almost perpendicular to the horizontal axis of the image (an angle of 80-90° is allowed).

4. Horizontal upper edge of doorframes is almost perpendicular to the vertical axis of the image (an angle of 60-90° is allowed).

5. The doors in the image have at least a certain width and a certain length.

In the subsequent discussion, the term "rectangle" will refer to a convex quadrilateral polygon that closely approximates a true rectangle. This usage is adopted for its intuitive appeal, as it more readily evokes the familiar shape of a door, rather than an abstract quadrilateral.

4.2 Line Detection

Given that the door is typically delineated by a rectangular shape that can be formed by straight lines, the algorithm is concerned with locating it. Within this domain, several gradient-based algorithms have been selected and evaluated as the most popular approach in the field. The detection process itself consists of several parts.

4.2.1 Depth Estimation

Image depth estimation is done by applying the ZoeDepth network to the input image. The results obtained using the flagship ZoeD-M12-NK model demonstrate satisfactory accuracy in all common scenarios. Figure 4.1 illustrates the network output.

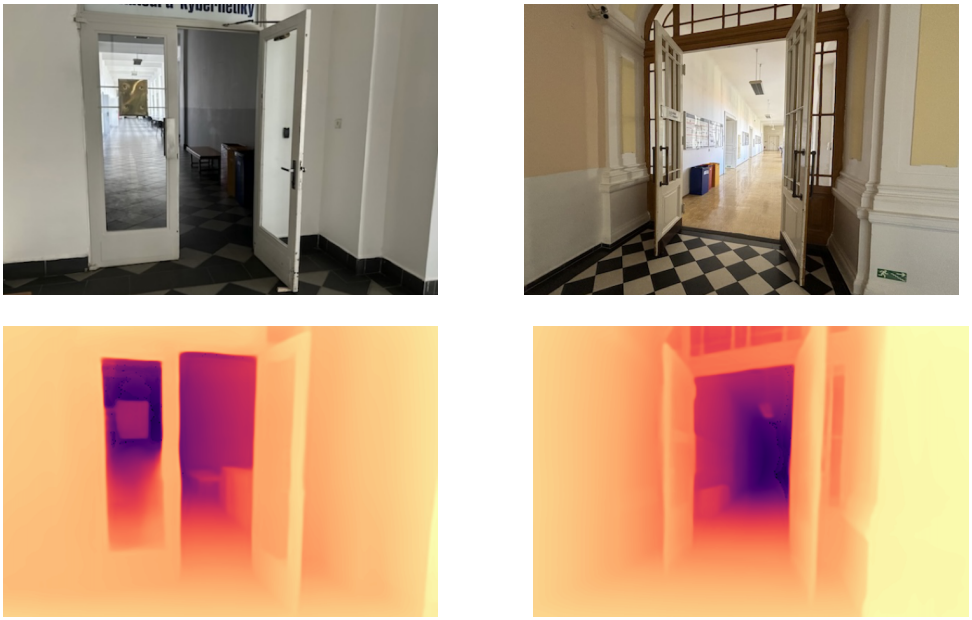


Figure 4.1: Outcome of the ZoeDepth applied to images.

4.2.2 Gradient Computation

Since the proposed algorithm simultaneously works on both RGB and computed depth images, it requires the computation of two gradient images. This process involves the utilisation of Canny Edge Detection, which is implemented in the OpenCV library [29]. For the RGB image, the initial step involves deriving a grayscale representation, whereas for the depth image, normalisation of depth data is performed to fit the applied Canny filter. The result of Canny Edge Detection is shown in Figure 4.2 and Figure 4.3.

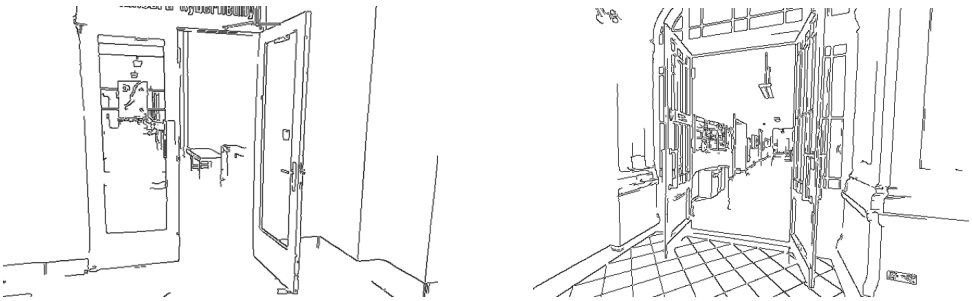


Figure 4.2: Estimated colour gradient image.

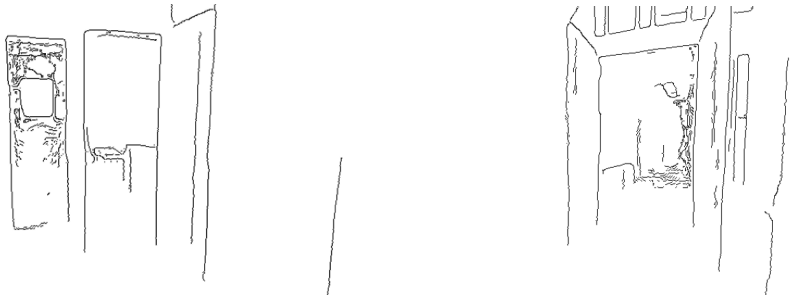


Figure 4.3: Estimated depth gradient image.

4.2.3 Segment Extraction

With the computed gradient images at hand, the line detection algorithm may be employed. Initially, four algorithms were evaluated: the Hough Line Transform [30], Probabilistic Hough Line Transform [31], and Line Segment Detection, all of which are implemented within the OpenCV library, along with the open-source Enhanced Line SEgment Drawing (ELSED) [32] algorithm. These algorithms can be categorized into two distinct groups: global and local methods. Global methods possess the capability to identify complete lines within the image, whereas local methods selectively exclude pixels with strong gradients and incrementally include neighbouring pixels based on gradient information. Consequently, local methods tend to return line segments rather than complete lines.

Method Choice

Global methods: The Hough Line Transform algorithm presents several drawbacks. First and foremost, it often returns straight lines that are too long, it tends to combine several lines following each other into a singular, disregarding their discontinuity. Additionally, the algorithm is very sensitive to parameter settings, resulting in variable efficacy across different scenes when employing a single set of predefined parameters. Furthermore, due to its requirement to traverse a high-dimensional parameter space and use a voting process for each pixel within the image, this algorithm is characterised by a high computational complexity. Figure 4.4 displays the results of applying the Hough Line Transform to images. The outcome is calculated using the depth gradient image, which is employed throughout all subsequent figures in this section, given its significant role within the algorithm.

Figure 4.4: Output of the Hough Line Transform.

The Probabilistic Hough Transform for line detection represents an optimisation of the conventional Hough Transform method. It notably mitigates computational demands in contrast to its traditional counterpart. However, it is accompanied by several drawbacks. Initially, the probabilistic approach aims to strike a balance between speed and accuracy. By sampling a subset of edge points, it gains efficiency but may lose some accuracy. Furthermore, it also requires careful tuning of parameters such as the threshold for line detection, the minimum line length and the number of iterations. This tuning process can be non-trivial and vary depending on the specific image. Following Figure 4.5 shows the Probabilistic Hough Line Transform applied to images.

Figure 4.5: Output of the Probabilistic Hough Line Transform.

Additionally, it is common for such methods to generate false positives in regions of high edge density.

Local methods: The Line Segment Detection algorithm excels due to its robustness and efficiency in detecting line segments within images. This algorithm employs a combination of gradient information and probabilistic techniques to efficiently extract line segments making it a preferred option for real-time applications. However, a notable drawback of this method is its necessitation of additional post-processing to derive complete lines from the

identified line segments. Figure 4.6 presents the results of applying the LSD to images.

Figure 4.6: Output of the LSD.

ELSED represents a modern approach for identifying line segments within an image. Although it is still based on gradient pixel change it uses an enhanced routing algorithm for neighbour region growth. This method is characterised by its notable speed and efficiency. However, there is a notable limitation in its design since it is designed to operate directly on image data, thereby computing gradients internally. Figure 4.7 below illustrates the ELSED applied to images.

Figure 4.7: Output of the ELSED.

Following an evaluation process, the LSD algorithm has been selected for the identification of line segments within both depth and RGB images.

4.2.4 Clustering

Given the selection of LSD, the proposed algorithm necessitates getting complete lines from the detected segments. This process involves the use of a

clustering algorithm applied to the identified line segments, followed by line prototyping for each cluster.

Pre-Clustering Filtering

To optimize cluster formation, a pre-clustering process is executed, comprising the filtration of line segments based on vertical, horizontal, and outlier criteria. The procedure is performed according to the angles specified in section 4.1. This allows clusters to be found independently for vertical and horizontal lines. Figure 4.8 shows the outcome of angle filtering.

Figure 4.8: Result of filtering based on line angles.

Method Choice

In the context of segment clustering, various methods were considered. Dynamic K-means [33] was abandoned due to its inherent disadvantages. The dynamic adjustment of cluster count can significantly increase the computational complexity of the algorithm, thereby leading to longer processing time and increased resource requirements. Furthermore, this algorithm is known to overfit and it lacks efficacy in handling noisy datasets.

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [34] algorithm has been chosen for several reasons, Firstly it can identify noise points and does not force every point into a cluster, making it robust to outliers. Secondly, unlike certain other clustering algorithms, DBSCAN does not require specifying the number of clusters beforehand. Lastly, It is relatively efficient for large datasets because it only needs to calculate distances between points within ϵ-distance. However, DBSCAN has a few drawbacks. Similar to numerous clustering algorithms, DBSCAN's performance can degrade in high-dimensional spaces due to the curse of

dimensionality. Nonetheless, this limitation is minor in the case of lines within two-dimensional space. Additionally, it may struggle with datasets where clusters have significantly varying densities or where the density of points within clusters changes substantially, but it is supposed that the LSD algorithm will yield a sufficient number of line segments for each line, thereby ensuring nearly uniform distances between them.

Algorithm Modification

The standard DBSCAN implementation served as the foundation for evaluating the algorithm's performance. However, given its original design for point data in a 2D space, several adjustments were necessary. Initially, the calculation of Euclidean distances between points was adapted to the Euclidean norm of the perpendicular vector extending from one line to the endpoint of another line. This modification offered a clear, geometrically intuitive, and mathematically sound approach for measuring the separation between lines. While initially effective, this approach encountered challenges in accurately handling wider discontinuities between segments of the same line that lay on a shared straight trajectory.

Consequently, a modification to the algorithm was added. Line segments may now be clustered together only if the variance in their angles does not exceed 10 degrees. This criterion is established based on the assumption that line segments belonging to the same line tend to have a nearly parallel orientation. Additionally, the distance between lines is now determined as the minimum value between the perpendicular distance and the Manhattan distance. The utilisation of the Manhattan distance serves to measure the relaxed distance between the endpoints of lines, taking into account potential changes along the axis of the intended line (horizontal axis for horizontal lines and vertical axis for vertical lines). The following Figure 4.9 shows the change in distance measurement as described.

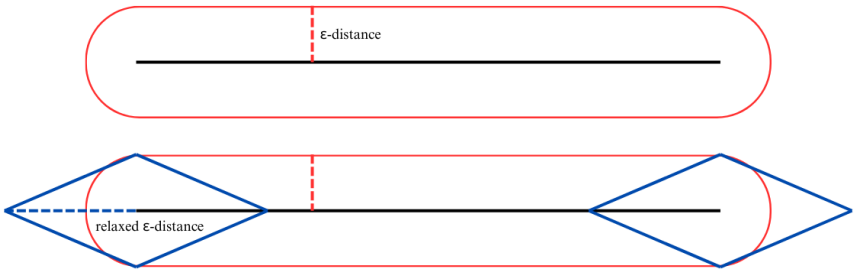


Figure 4.9: Distance measurement methods.

Subsequently, the modified algorithm was evaluated on the dataset, containing five tests with synthetic data. Each evaluation test is located within the chapter A. The following Table 4.1 presents the aggregated results of this assessment averaged over the entire dataset.

Performance Measures	DBSCAN	Modified DBSCAN
Precision	97.9%	98.3%
Recall	52.3%	60.9%
F_1-score	61.7%	69.0%

Table 4.1: DBSCAN modification comparison.

As shown in the table, the modified DBSCAN algorithm provides better results, making it the preferred choice for integration within the proposed algorithm. The following Figure 4.10 illustrates the real-world data output generated by the modified method.

Figure 4.10: Result of the adapted DBSCAN algorithm.

4.2.5 Prototyping

Once the clusters have been identified, the subsequent stage for the algorithm involves obtaining the original line prototypes. This task can be likened to the construction of a linear regression model, with the primary distinction being that line segments, rather than individual points, define the original line. Conventionally, solving this problem using real data involves using the RANSAC algorithm. Known for its efficacy, robustness, and flexibility, RANSAC operates by iteratively sampling subsets of the data, enabling the identification of inliers and the exclusion of outliers during the model fitting

process. This characteristic renders RANSAC more robust than classical methods for linear regression construction (e.g. Least Squares Method). Additionally, due to its iterative nature, RANSAC often converges to a solution quickly, making it computationally efficient.

Method Application

Each line segment within each cluster is interpolated by uniformly spanning points along it. The quantity of points generated is directly proportional to the segment length. This approach allows the use of an existing RANSAC implementation: RANSACRegressor implemented in the sklearn library [35] is used to compute the original line prototypes. After receiving the lines, they go through a filtering process, where lines shorter than a set threshold are excluded from further consideration. The outcome of this line prototyping process combined with length filtering is shown in Figure 4.11.

Figure 4.11: Lines fitted by the RANSAC algorithm.

In this phase, lines derived from both colour and depth images are obtained. These prototypes will henceforth be referred to as colour and depth lines in subsequent investigations.

4.3 Door Model Assembly

Upon identifying line prototypes, the next step of the algorithm can be employed. This section describes the process of constructing potential door models from the identified lines. It is based on a series of controls to omit obvious non-door candidates, thereby easing the workload for the subsequent model validation module.

4.3.1 Edge Grouping

The search algorithm groups all identified lines into separate categories. In the following section, the algorithm will work exclusively with lines derived from the depth gradient image.

First, the algorithm identifies the potential door upper edge by iterating over all horizontal lines and applying the `is_horizontal_edge` method implemented within the Line class. This method determines whether a line is a potential edge based on the depth difference above and below it. The method begins by interpolating the given line with points at specified intervals. For each point, the depth difference above and below it, within a defined distance, is calculated and compared to a threshold value. If this value exceeds the threshold, the point is flagged as a top edge point, if it falls below the negative threshold, it is flagged as a bottom edge point, otherwise, it is flagged as not an edge point. These flags are recorded in a control array. The algorithm then evaluates whether the number of top edge flags exceeds a given proportion of the control array length. If it does, the line is confirmed as a top edge, otherwise, it is flagged as not an edge. Each horizontal line flagged as a top edge is added to the top edges group. The following Figure 4.12 shows the top edges highlighted with red along with all other horizontal lines.

Figure 4.12: Top edges identified.

Second, the algorithm identifies potential door left and right edges by iterating over all vertical lines and applying the `is_vertical_edge` method, which operates similarly to the `is_horizontal_edge` method. The primary difference is that it compares depth on the left and right sides of the given line. This method returns flags for left edge, right edge, or no edge. The algorithm then creates groups for left edges and right edges accordingly. Figure 4.13 below shows the left and right edges highlighted with red and blue accordingly along with all other vertical lines.

Figure 4.13: Left and right edges identified.

4.3.2 Model Construction

The algorithm initiates the assembly of each door from the left edge. For each left edge, it iterates over all right edges. During this process, the appropriate right edge is selected based on its position relative to the left edge, ensuring it is located to the right.

Construction of the Door Upper Edge Search Boundaries

The coordinate boundaries for the door upper edge search are established through the formation of specific points. Initially, from the left edge's minimum x-axis coordinate, a distance of α is subtracted to determine the $x_{\alpha-\text{left}}$ coordinate. Similarly, a distance of β is subtracted to obtain the $x_{\beta-\text{left}}$ coordinate. The same y-axis value for these two points is calculated by adding the α value to the top y-axis coordinate of the left edge, resulting in the creation of $\alpha-\text{left}$ and $\beta-\text{left}$ points, both having the $y_{\alpha-\text{left}}$ coordinate. The process is mirrored on the right edge, where α and β values are added to the right edge's maximum x-axis coordinate, generating the $\alpha-\text{right}$ and $\beta-\text{right}$ points, which share the $y_{\alpha-\text{right}}$ coordinate. The following Figure 4.14 shows boundaries for the upper edge search.

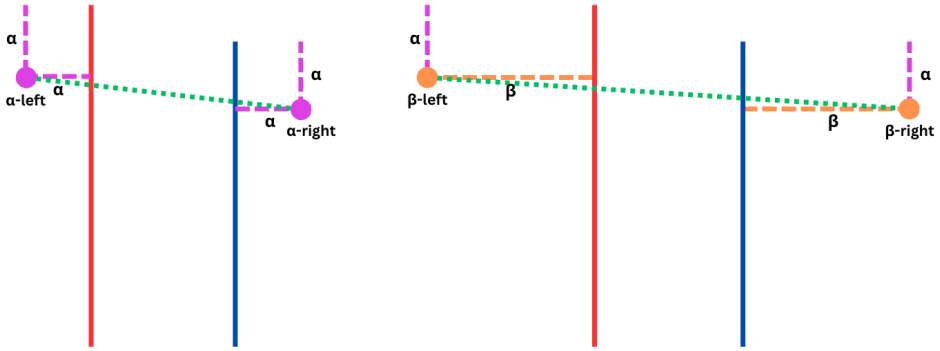


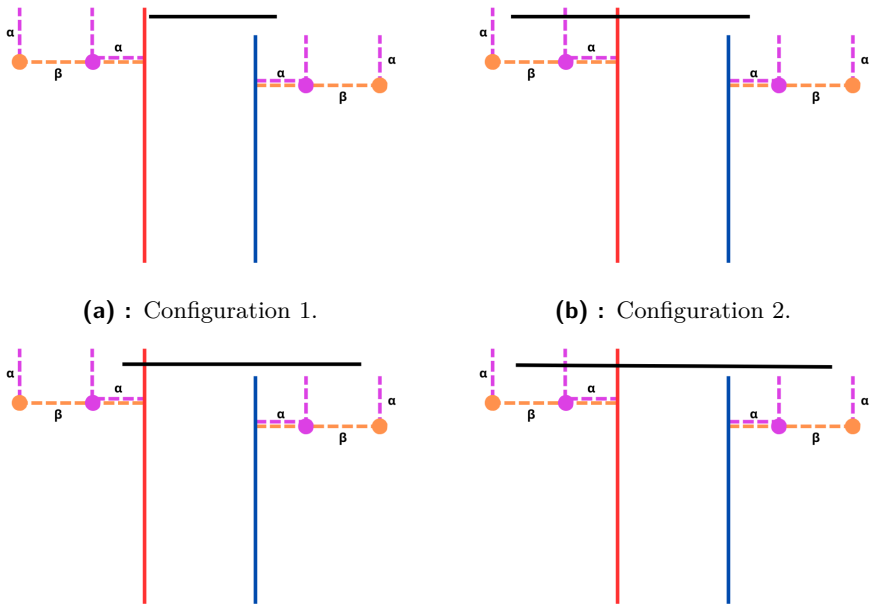
Figure 4.14: Upper edge search boundaries.
The green dotted line indicates the region above which the door's upper edge is examined.

Search for the Door Upper Edge

The search for a suitable door upper edge is conducted by iterating over the group of top edges and evaluating their positional conditions. First, the y-axis coordinates of the left and right endpoints are checked to ensure they are less than the corresponding $y_{\alpha-\text{left}}$ and $y_{\alpha-\text{right}}$ values, respectively. This verification essentially guarantees that the upper edge is located within the upper part of the door model.

Door State Identification

The algorithm subsequently continues by evaluating whether the upper edge is suitable with parallel potential door state extraction. There exist multiple outcomes of interest that are mutually exclusive regarding the specified upper edge location to the defined search boundaries Figure 4.15.



(a) : Configuration 1. **(b) :** Configuration 2.

(c) : Configuration 3. **(d) :** Configuration 4.

Figure 4.15: Possible upper edge positions.

1. **Upper edge within the $x_{\alpha-\text{left}}$ and $x_{\alpha-\text{right}}$ interval:**
   - If the edge does not protrude from either side, it is flagged as suitable, indicating that the door coincides with its doorway.
2. **Upper edge within the $x_{\beta-\text{left}}$ and $x_{\alpha,\text{right}}$ interval, covering $x_{\alpha-\text{left}}$:**
   - The edge is flagged as suitable, suggesting that the door is located to the right of the doorway.
3. **Upper edge within the $x_{\alpha-\text{left}}$ and $x_{\beta-\text{right}}$ interval, covering $x_{\alpha-\text{right}}$:**
   - The edge is flagged as suitable, suggesting that the door is located to the left of the doorway.
4. **Upper edge within the $x_{\beta-\text{left}}$ and $x_{\beta-\text{right}}$ interval, covering both $x_{\alpha-\text{left}}$ and $x_{\alpha-\text{right}}$:**
   - The edge is flagged as unsuitable. This indicates a potential central door configuration or the absence of a door, as the upper edge is disproportionately wide relative to the door's vertical edges. Thus, the algorithm disregards this upper edge.

If none of the conditions above are met, the upper edge is marked as inappropriate. If configurations 2 or 3 occur, the lines derived from the colour gradient image are utilised.

For configuration 2: The algorithm progresses by first identifying the upper edge of the doorway from the provided horizontal lines, which should be in close proximity to the previously detected upper edge of the door. If such an edge is found, it is assigned with the upper edge of the doorway, otherwise, the upper edge of the door is assigned with the upper edge of the doorway. Subsequently, it iterates through vertical lines to locate the nearest line below either the left endpoint of the previously detected upper edge of the door or the determined upper edge of the doorway. This process involves defining search boundaries for the line. The identified line has the potential to establish the left edge of the doorway. Upon finding such a line, its uniqueness is assessed. If no comparable line exists among the depth lines, the identified line is assigned as the left edge of the doorway for utilisation in the doorway assembly phase. Conversely, if a similar line is present, it is disregarded.

For configuration 3: The algorithm performs a similar procedure as described above, but searches for the right doorway edge instead of the left. If a unique line is found (with no similar line in the depth lines), it is assigned as the doorway right edge and used in the doorway assembling phase.

Door Corners Determination

In this step, the algorithm verifies whether the difference between the lower y-axis coordinates of the left and right door edges is less than a predefined threshold. If the difference is within this threshold, it is interpreted as a minor error in line prototyping or a result of perspective distortion. In such cases, a single door will be assembled using the detected left, right, and upper edges, along with a newly constructed bottom edge.

The bottom edge is determined by taking the maximum of the two endpoint coordinates (the lowest point in the image) and mirroring the upper edge line along the x-axis from this point. This method is based on the intuition that perspective distortion is typically mirrored along the horizontal axis of the image. Consequently, the four door corners are identified by calculating the intersections of each pair of lines. Once these steps are completed, the door model is constructed.

If the difference between the lower y-axis coordinates exceeds the predefined threshold, two door models will be constructed in the same manner. In this case, the bottom edge is determined separately for both the left and right edge bottom endpoints.

After completing these steps, one or two door models are assembled in each iteration of the algorithm. The validation module is then applied to check whether the constructed door model(s) represent actual doors. Figure 4.16 shows corner determination. The bottom line is always drawn to the left side, since in practice only its direction vector is needed to calculate the intersection.

Figure 4.16: Determining corners through intersection line calculations.

4.4 Model Validation

The validation module comprises several checks, each designed to filter out inappropriate door models based on different criteria.

4.4.1 Geometric Validation

The validation process begins with assessing the door model's geometry. It ensures that the upper-left corner is positioned above the bottom left corner, and similarly for the right corners, thereby guaranteeing the door rectangle's convexity. Subsequently, it verifies that the door rectangle's width and height exceed minimum length requirements.

4.4.2 Edge Validation

Following geometric validation, the module proceeds to verify the left and right edges of the constructed door model. Since the corners of the rectangle model are determined by the intersections of the original edges, the lines may have changed their lengths. Therefore, it is necessary to revalidate edge consistency.

Firstly, the left line is examined to ensure it remains an edge line by applying the `is_vertical_edge` method with a reduced majority parameter, which yields a left edge flag. Next, the midpoint of the left line is calculated and used to create a new left half-line, with the original bottom endpoint and the newly calculated midpoint. The `is_vertical_edge` method is then applied to this new half-line with the default majority parameter but a lowered depth difference threshold. These two checks ascertain whether the new door line remains aligned with the real edge in the image and whether the edge is not excessively long, as the checks assess line protrusions on both sides. The same validation procedures are applied to the right line, using the `is_vertical_edge` method to obtain the right edge flag.

Subsequently, the rectangle's bottom line is checked to ensure it is an edge. The `is_horizontal_edge` method is initially used with a small depth difference threshold and the default majority value to confirm it is a line at the bottom part of the door, as minor depth variations are expected due to floor perspective changes in an open door. The method is then applied again with the default threshold and a smaller majority value, where it is anticipated to yield a non-edge flag, as a significant difference would indicate the bottom edge is likely the window edge, not the door.

As a result, all door models failing to meet these conditions are excluded.

4.4.3 Door Model Rectangle Validation

In the subsequent step, the algorithm verifies the absence of edges of the left, right, and top edge groups in the door model that are comparable in length to the door rectangle's width or height (for the upper, left, and right edges, respectively). It also ensures that there are no edges from these groups that originate inside the rectangle and intersects it to the outside. Edges that are too close to the rectangle lines are disregarded, as multiple line prototypes

may exist for a single line forming the rectangle, which are in close proximity to each other.

4.4.4 Doorway Integration

In the subsequent step, the algorithm verifies whether the door has a status 2 or 3 indicated in section 4.3, and whether the outer doorway edge is identified. If these conditions are met, the doorway, which is distinct from the open door, is constructed in the following section and subsequently integrated into the door model, which is then appended to the array of doors. If both of these conditions are not met, the door is directly appended to the array.

4.4.5 Doorway Assembling

This process is described for the doorway configuration from point 3 of section 4.3. Initially, the top left and right corners of the doorway are set by calculating the intersections of the right doorway edge, the upper doorway edge, and the left door line from the existing door model. Then, the bottom doorway edge, which mirrors the upper doorway edge in the x-axis, is laid from the lowest point chosen from the right doorway edge bottom point, the bottom left corner of the door model, and the bottom endpoint of the door model's original left edge. The intersections for the bottom left and right corners are then calculated. The resulting doorway rectangle, defined by four points, is added to the door model instance. The same process applies to the doorway configuration from point 2 of section 4.3, with the difference of utilising the left doorway edge instead of the right, and the door model's right edge. Figure 4.17 illustrates the doorway assembly process.

Figure 4.17: Doorway assembly for the 3rd configuration.
Green line: door model's left line.
Red line: original door's left edge found in the image.
Lilac vertical line: doorway's right edge.
Lilac horizontal line: doorway's top edge.
Blue line: doorway's bottom edge, mirrored from the upper edge.

4.4.6 Door Filtering

In the final step, the proposed algorithm examines the array of doors to ensure their uniqueness in terms of their positions. If one door overlaps another by more than 90%, the largest door is chosen. This method is applied to all doors in the array, ensuring that the array contains unique doors without multiple doors occupying the same position in the image.

Chapter 5

Results

This chapter presents the results of the evaluation of the proposed door detection algorithm on the collected dataset. Extensive experiments have been conducted to vary the algorithm's performance, robustness, and real-world applicability. First, the steps taken to create the dataset are explained to ensure data quality. In addition, metrics for evaluating the effectiveness of the chosen algorithm are provided. Subsequently, the evaluation results and a comparative analysis are presented, highlighting progress and contextualizing challenges.

5.1 Dataset Description

The dataset used in the evaluation process was manually created. It consists of various doors in different states to encompass a wide range of possible scenarios. The dataset includes both horizontal and vertical images. Initially, 21+1 images underwent preprocessing, which involved resizing to the intended dimensions of 600×400 pixels for horizontal orientations and 400×600 pixels for vertical orientations. Subsequently, the precise locations of the door ground truths were manually extracted from each photo, resulting in the creation of an array of ground truth door corners.

5.2 Performance Evaluation Metrics

Precision, recall, F_1-score, and accuracy are essential metrics for evaluating algorithm performance. The standard formulas for precision 5.1, recall 5.2, and F_1-score 5.3 are used. In visual detection problems, the conventional accuracy formula is modified to exclude True Negatives 5.4, resulting in the Intersection over Union (IoU) metric.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{5.1}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{5.2}$$

$$F_1\text{-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.3}$$

$$\text{Accuracy} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives} + \text{False Negatives}} \tag{5.4}$$

Each metric has a distinct significance. Precision ensures the algorithm does not mistakenly identify a wall as a door, thereby preventing erroneous actions. Recall measures the proportion of correctly recognized open doors. The F_1-score, a statistical measure, provides a comprehensive assessment of the algorithm's overall efficiency. Accuracy, in the context of visual recognition tasks, is a spatial measure that directly evaluates the overlap between predicted and ground truth regions.

These metrics are utilized to assess the dataset images through the implementation of door overlap calculation for the ground truth and predicted door models. The shapely.intersection method within the Shapely library [36] is employed, as it provides a suitable approach for computing polygonal intersections.

5.3 Results Analysis

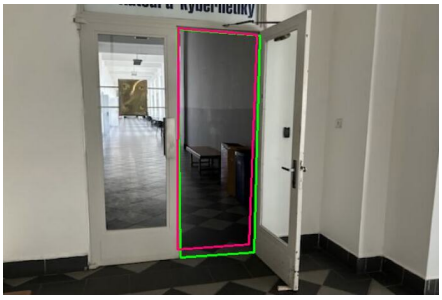
The visual results for each image, including both the ground truth and predicted door annotations, are presented in the Appendix A of this work. The ground truth door is represented with a green outline, while the predicted open door is illustrated with a pink outline. In cases where the doorway differs from the open door, the doorway is indicated with a violet outline. To avoid confusion, the ground truth doorway is not shown in these instances. Table 5.1 below provides the performance evaluation for each image in the dataset.

Image	**Accuracy**	**Precision**	**Recall**	F_1**-score**
5.1a	89.9%	98.2%	91.4%	94.7%
A.6	91.7%	96.9%	94.5%	95.7%
A.7	93.4%	99.7%	93.6%	96.6%
5.2a	0%	100%	0%	0%
A.8	89.3%	97.7%	91.3%	94.4%
A.9	89.2%	98.2%	90.7%	94.3%
5.4b	0%	100%	0%	0%
A.10	88.7%	99.1%	89.4%	94%
A.11	92.7%	99.8%	93%	96.2%
5.2b	0%	100%	0%	0%
5.1b	85.8%	100%	85.8%	92.4%
A.12	86.9%	98.5%	88%	93%
A.13	72.9%	72.9%	100%	84.3%
A.14	93.9%	99.9%	94%	96.9%
5.3b	96.5%	99%	97.5%	98.2%
A.15	89.8%	98.5%	91.1%	94.6%
A.16	91.5%	93.1%	98.1%	95.5%
A.17	94.3%	99.5%	94.7%	97.1%
A.18	89.3%	96.8%	92%	94.3%
5.3a	92.4%	98.9%	93.3%	96%
5.4a	83.5%	90.1%	91.9%	91%

Table 5.1: Detection evaluation results.

As can be seen from the presented table, the algorithm shows noticeable changes in performance depending on the input image. Therefore, to provide

a comprehensive overview of its results, it is necessary to highlight several key points from the assessment.



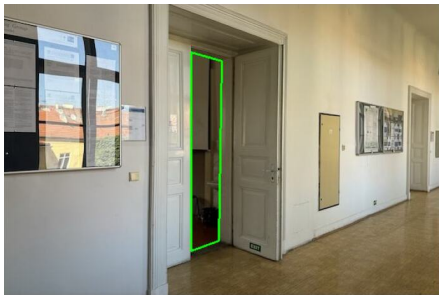
(a) : Image 1 outcome.

(b) : Image 11 outcome.

Figure 5.1: Algorithm's high-performance output.

Figure 5.1 shown above is an example of high-performance cases. The open doors exactly match their doorways, indicating a minimal difference between the predicted geometric model and the ground truth configuration. Notably, the primary variation observed between the models in most images is in the location of the lower edge of the doorway, leading to lower recall rates and consequently lower F_1 and accuracy metrics.

(a) : Image 4 outcome.



(b) : Image 10 outcome.

Figure 5.2: Showcase of the algorithm's incapability to recognize doors.

The following Figure 5.2, shown above, illustrates cases where the algorithm fails to detect doors. This deficiency primarily results from the constraints related to the lower edge of the door that are enforced during the execution of the validation module described in Section 4.4. Because the algorithm must distinguish between doors, windows, and glass components of the door, the threshold value set for the depth disparity between the bottom of the door and the surrounding environment does not allow the increased values observed in these examples. As a result, the algorithm incorrectly interprets these doors as windows.

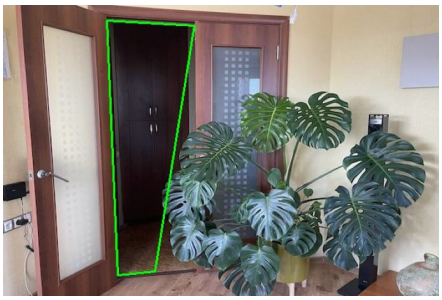
(a) : Image 20 outcome.

(b) : Image 15 outcome.

Figure 5.3: Algorithm's prediction of doors different from their doorways.

Figure 5.3 shown above presents an example of open doors, which differ from their corresponding doorways. Figure 5.3a shows near-flawless door recognition with precise open door identification and precise doorway recognition, resulting in high performance. Meanwhile, Figure 5.3b shows precise open door recognition along with low recall in doorway identification, which in turn leads to reduced F_1 and accuracy values. This deficiency is due to the inaccurate location of the lower edge of the doorway.

(a) : Image 21 outcome.



(b) : Image 7 outcome.

Figure 5.4: Algorithm's output for partially overlapped doors.

Figure 5.4 shown above, shows cases where the door is partially overlapped by objects. As can be seen in these images, the vertical edge of the door remains distinguishable and is not completely obscured by overlapping elements in both images, as mentioned in point 1 of section 4.1. In Figure 5.4a,

the overlap is minimal, which allows the algorithm to detect open door. Conversely, in Figure 5.4b, the extent of overlap along the vertical edge is considerable, resulting in the inability of the algorithm to predict the presence of a door.

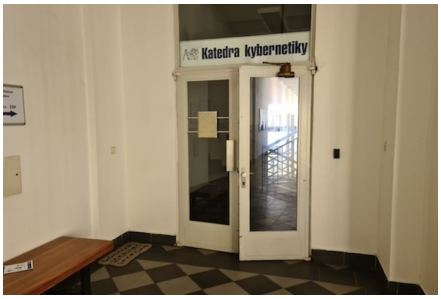


Figure 5.5: Algorithm's outcome for True Negative data.

Figure 5.5 shown above is for supplementary purposes and is not included in the dataset used for performance evaluation. Due to the absence of doors in this figure, its sole purpose is to make sure that the algorithm refrains from predicting doors in True Negative cases. As a result, the algorithm concludes its evaluation of this image without any door predictions, in accordance with the expected result.

The following Table 5.2 presents the aggregated performance results averaged over the entire dataset.

Accuracy	**Precision**	**Recall**	F_1**-score**
76.7%	97%	75.8%	80.9%

Table 5.2: Results of the proposed algorithm.

The proposed algorithm demonstrates high precision, thereby enhancing safety in guiding drone navigation through door detection. However, the recall rate is relatively low because the lower edge of the door is often detected at a significant distance from its actual position in most evaluated scenarios and the inability of the method to predict doors in certain scenarios. This discrepancy of the first arises from inaccuracies in the coordinates of vertical edges along the y-axis. While these limitations may not critically impact autonomous drone navigation, as drones typically do not operate close to the ground and might skip certain doors, the limitations make the algorithm less suitable for other robotic platforms or alternative applications. Consequently, this inconsistency results in a lower F_1-score and reduced accuracy.

Chapter 6

Conclusion

This research aims to enhance autonomous robot navigation by focusing on the detection of open doors, which are crucial for navigating unfamiliar indoor environments. Using a MAV as the camera platform provides some benefits like increased manoeuvrability and lower costs, despite its limited payload capacity restricting advanced sensors.

To address these limitations, conventional monocular cameras and off-board intensive processing to powerful external systems are employed. By integrating a neural network for Single Image Depth Estimation with an original RGB image, precise and efficient real-time door detection based on geometric features is enabled.

6.1 Summary of Key Findings

Based on the results discussed in Chapter 5, the suggested algorithm performs well only in situations where the door has a clear depth separation from the background environment. In such cases, the algorithm demonstrates excellent precision and high recall, resulting in high overall metrics such as F_1-score and accuracy. When comparing the proposed algorithm with existing ones, it performs worse than one of the best algorithms in the field [6]. However, this comparison is not entirely valid, as the referenced algorithm requires pre-scanning of the environment, which is not feasible for autonomous drone navigation. Therefore, the algorithm suggested in this research should be

compared with algorithms that do not require any pre-processing.

When compared to RGB-D data-based methods [1], [4], and [5], which does not require pre-scanning, the suggested algorithm performs worse, as expected, due to the superior performance of modern ML techniques in visual detection problems. However, when compared to 2D input data ML-based methods [9], [10], and [13], the suggested approach shows similar precision, with its F_1-score not being far off. Additionally, when compared to methods [11], [12] based on geometric features using only RGB images, the proposed method demonstrates higher precision and similar recall.

In summary, the algorithm is not outstanding, particularly in non-standard door configurations. Nonetheless, it should be noted that the approach of combining SIDE techniques for door recognition has the potential to yield better results, even when used in methods based solely on the geometric features of doors.

6.2 Limitations of the Study

As discussed in the previous section 6.1, the proposed algorithm encounters limitations when the door configuration deviates from a simple structure. The algorithm performs poorly in unconventional door situations, particularly when parts of the door are obscured by objects that cover the vertical edge of the door or when the door does not have a clear depth separation from the background environment.

6.3 Future Work

Considering the potential for improvement of the suggested method to achieve better results, the proposed algorithm can be enhanced by incorporating advanced model validation techniques. This enhancement can lead to higher recall in certain door configurations, as the current algorithm tends to struggle in difficult scenarios, often opting to predict no door to maintain higher precision.

A potential research direction to achieve superior performance in door detection from monocular camera data is to explore a Machine Learning-

based method. This method would combine the predicted depth map and the given 2D image as input. Such an approach holds significant promise for achieving exceptional performance.

6.4 Contributions to the Field

The proposed algorithm has practically broadened the research field by providing a solution that integrates a 2D image with a predicted depth map to address the problem of door recognition without requiring any additional pre-processing. This approach is also extended to detect doorways in special door configurations. Although the solution is not perfect, it is innovative due to the combination of these elements.

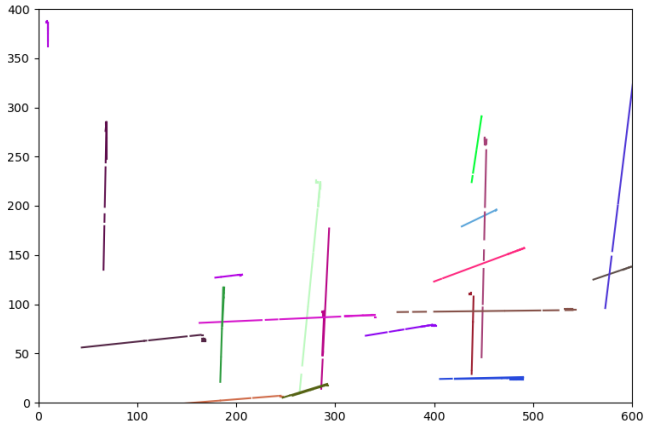
6.5 Final Remarks

This research journey has been both challenging and enlightening. Navigating through limitations, the integration of Neural Network for SIDE required extensive experimentation and optimisation in the context of door recognition. Various approaches were tested at each step of the algorithm, ultimately leading to the proposed method.

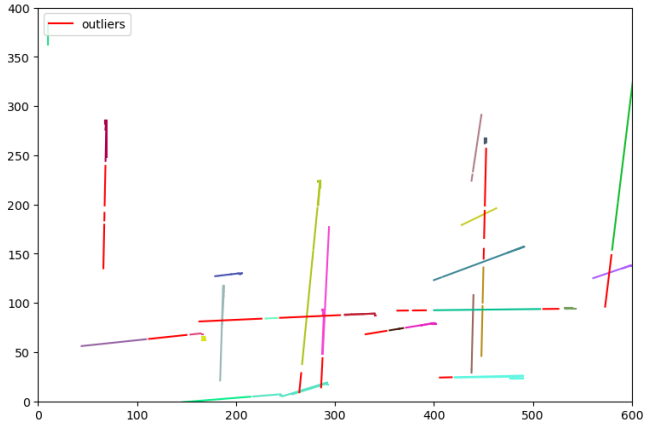
In conclusion, this research demonstrates the potential of using MAVs and advanced computational techniques to enhance autonomous navigation. By focusing on open door detection and leveraging off-board processing, the approach of combining predicted depth data with the original RGB image has been shown to improve the performance of recognition algorithms. This work advances the field of single-image door recognition and paves the way for further innovations.

Appendix A

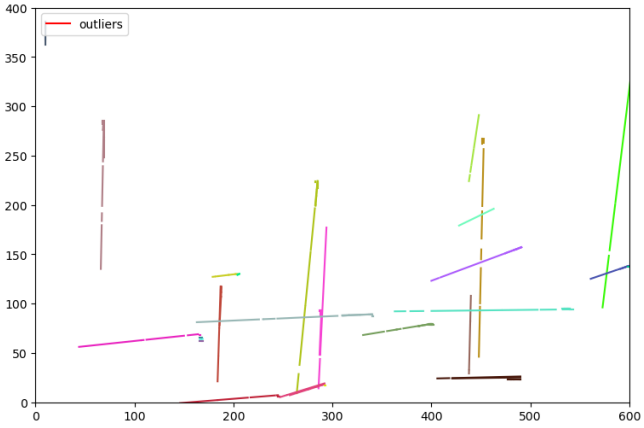
Attachments



(a) : Ground Truth: 20 clusters.

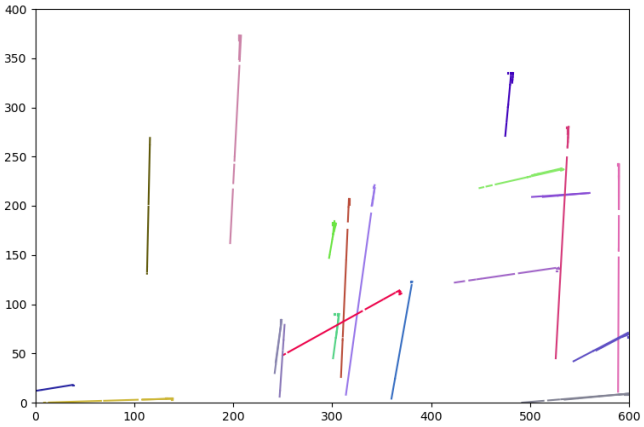


(b) : Prediction: 29 clusters.
Precision = 98.4%, Recall = 53.8%, F_1-score = 62%.

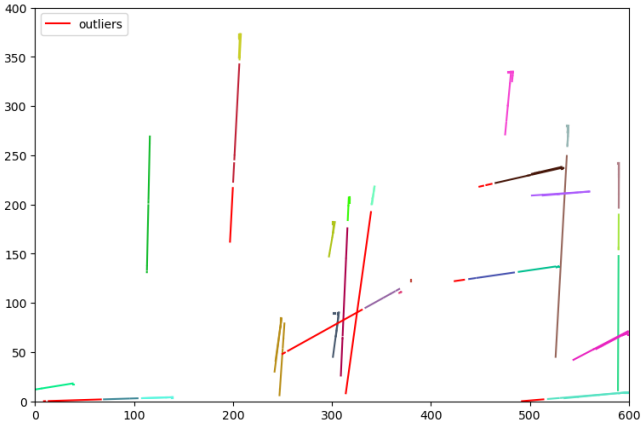


(c) : Modified algorithm's prediction: 29 clusters.
Precision = 100%, Recall = 60.2%, F_1-score = 69.5%.

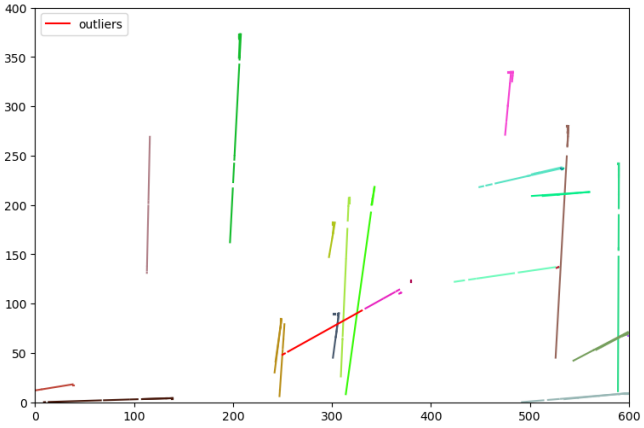
Figure A.1: DBSCAN results for 1st test.



(a) : Ground Truth: 20 clusters.

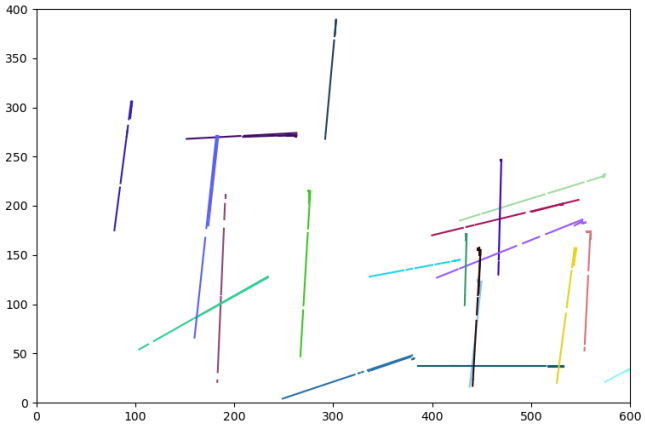


(b) : Prediction: 28 clusters.
Precision = 98.2%, Recall = 59.8%, F_1-score = 68.4%.

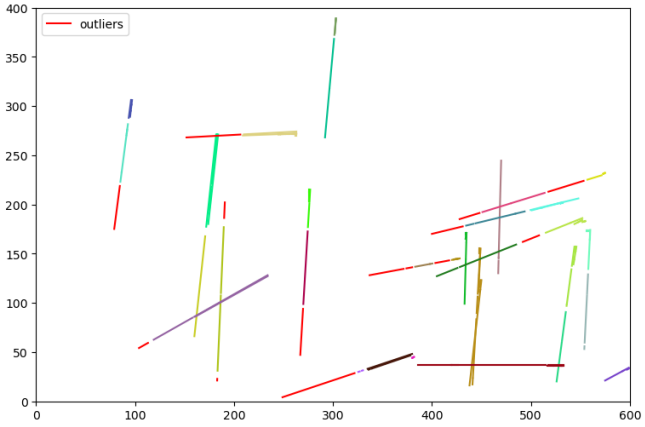


(c) : Modified algorithm's prediction: 24 clusters.
Precision = 98.5%, Recall = 70.9%, F_1-score = 77.1%.

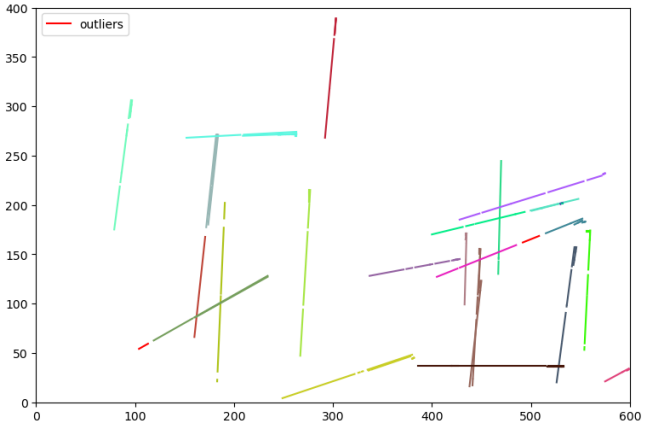
Figure A.2: DBSCAN results for 2nd test.



(a) : Ground Truth: 20 clusters.

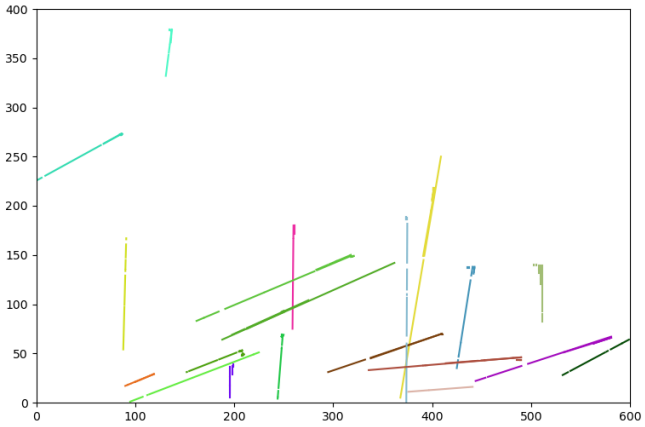


(b) : Prediction: 36 clusters.
Precision = 99.2%, Recall = 44.8%, F_1-score = 56.2%.

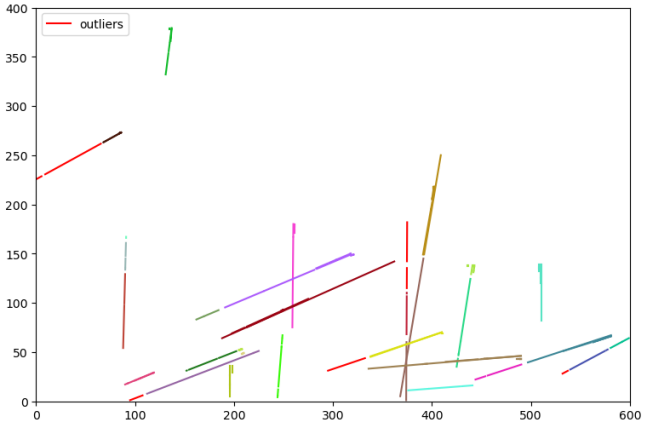


(c) : Modified algorithm's prediction: 28 clusters.
Precision = 99%, Recall = 60.5%, F_1-score = 69.4%.

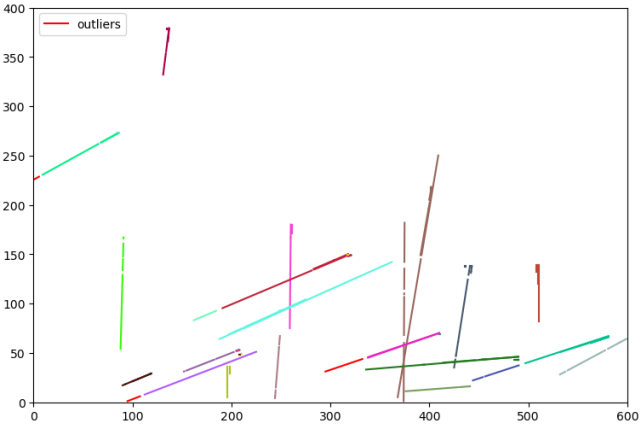
Figure A.3: DBSCAN results for 3rd test.



(a) : Ground Truth: 20 clusters.

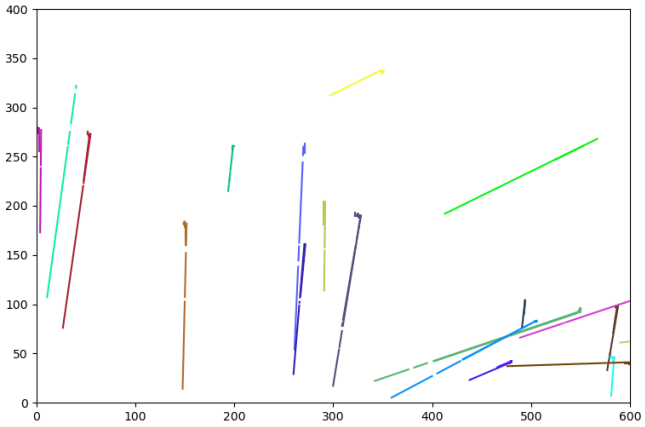


(b) : Prediction: 34 clusters.
Precision = 99%, Recall = 47.6%, F_1-score = 58.8%.

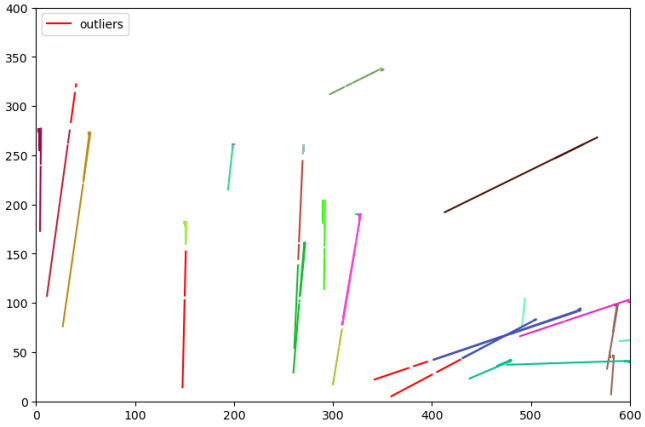


(c) : Modified algorithm's prediction: 31 clusters.
Precision = 98.8%, Recall = 51.7%, F_1-score = 61.1%.

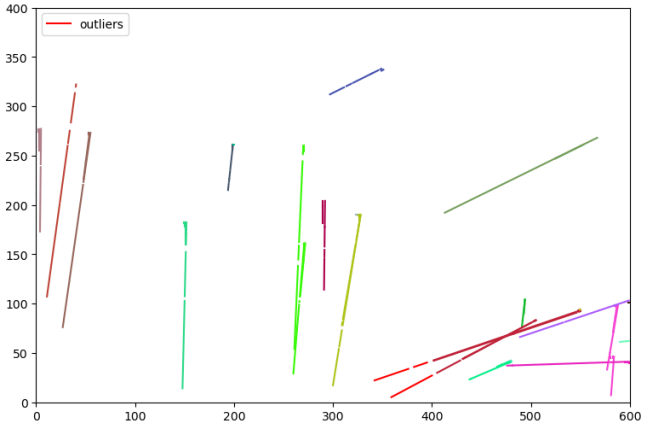
Figure A.4: DBSCAN results for 4th test.



(a) : Ground Truth: 20 clusters.



(b) : Prediction: 24 clusters.
Precision = 94.6%, Recall = 55.4%, F_1-score = 62.9%.



(c) : Modified algorithm's prediction: 24 clusters.
Precision = 95.2%, Recall = 61.3%, F_1-score = 68%.

Figure A.5: DBSCAN results for 5th test.

Figure A.6: Image 2 outcome.
Accuracy = 91.7%, Precision = 96.9%, Recall = 94.5%, F_1-score = 95.7%.

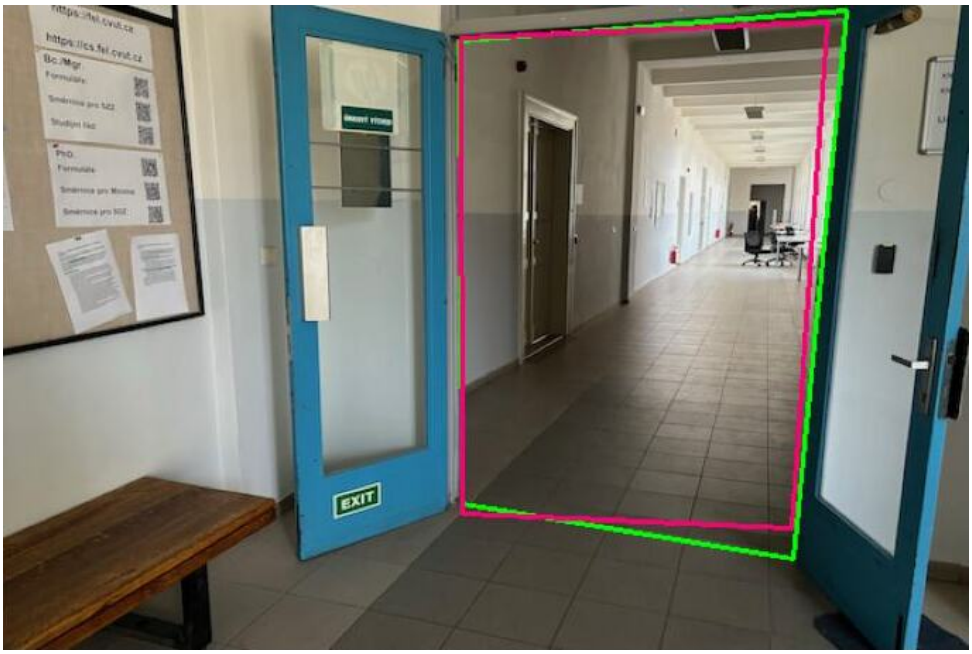


Figure A.7: Image 3 outcome.
Accuracy = 93.4%, Precision = 99.7%, Recall = 93.6%, F_1-score = 96.6%.

Figure A.8: Image 5 outcome.
Accuracy = 89.3%, Precision = 97.7%, Recall = 91.3%, F_1-score = 94.4%.

Figure A.9: Image 6 outcome.
Accuracy = 89.2%, Precision = 98.2%, Recall = 90.7%, F_1-score = 94.3%.

Figure A.10: Image 8 outcome.
Accuracy = 88.7%, Precision = 99.1%, Recall = 89.4%, F_1-score = 94%.

Figure A.11: Image 9 outcome.
Accuracy = 92.7%, Precision = 99.8%, Recall = 93%, F_1-score = 96.2%.

Figure A.12: Image 12 outcome.
Accuracy = 86.9%, Precision = 98.5%, Recall = 88%, F_1-score = 93%.

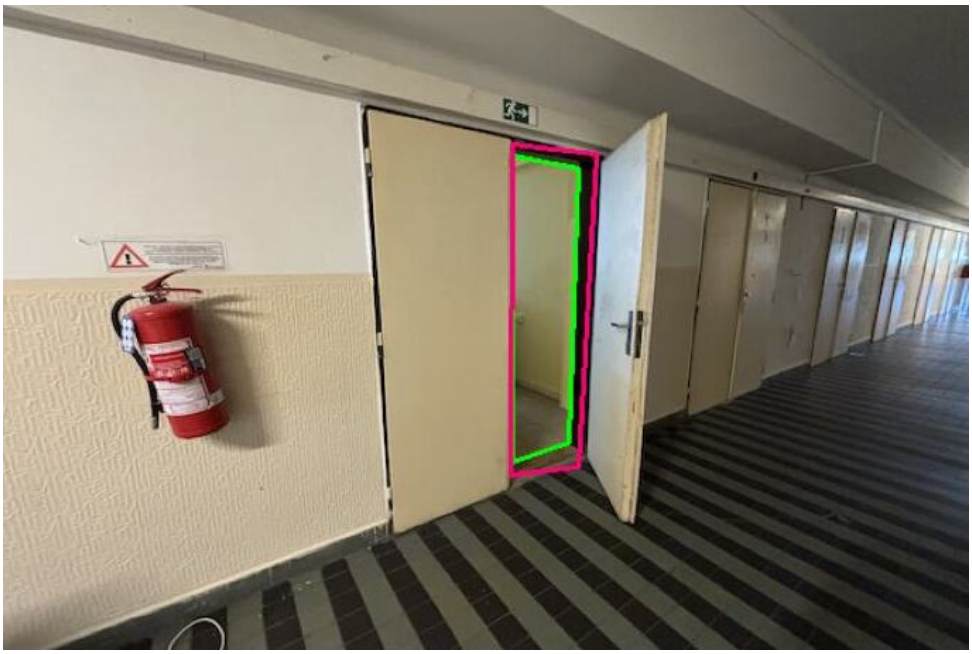


Figure A.13: Image 13 outcome.
Accuracy = 72.9%, Precision = 72.9%, Recall = 100%, F_1-score = 84.3%.

Figure A.14: Image 14 outcome.
Accuracy = 93.9%, Precision = 99.9%, Recall = 94%, F_1-score = 96.9%.

Figure A.15: Image 16 outcome.
Accuracy = 89.8%, Precision = 98.5%, Recall = 91.1%, F_1-score = 94.6%.

Figure A.16: Image 17 outcome.
Accuracy = 91.5%, Precision = 93.1%, Recall = 98.1%, F_1-score = 95.5%.

Figure A.17: Image 18 outcome.
Accuracy = 94.3%, Precision = 99.5%, Recall = 94.7%, F_1-score = 97.1%.

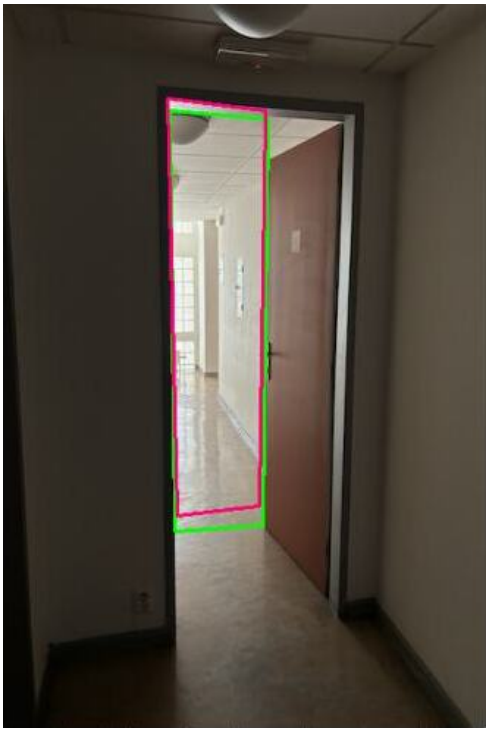


Figure A.18: Image 19 outcome.
Accuracy = 89.3%, Precision = 96.8%, Recall = 92%, F_1-score = 94.3%.

Appendix B

Bibliography

[1] A. H. Bhatti, A. U. Zafar, and M. Ahmed, "Door detection and distance estimation in habitat simulator for 3d environments," in *2023 3rd International Conference on Digital Futures and Transformative Technologies (ICoDT2)*, 2023, pp. 1–8.

[2] X. Ning, Z. Sun, L. Wang, M. Wang, Z. Lv, J. Zhang, and Y. Wang, "Door state recognition method for wall reconstruction from scanned scene in point clouds," *Mathematics*, vol. 11, no. 5, 2023.

[3] M. Akhoundi Khezrabad, M. J. Valadan Zoej, and A. Safdarinezhad, "A method for detection of doors in building indoor point cloud through multi-layer thresholding and histogram analysis," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. X-4/W1-2022, pp. 43–48, 2023.

[4] A. Spournias, C. Antonopoulos, G. Keramidas, N. Voros, and R. Stojanović, "Enhancing visual recognition for door status identification in aal robots via machine learning," pp. 1–6, 2020.

[5] J. G. Ramôa, L. A. Alexandre, and S. Mogo, "Real-time 3d door detection and classification on a low-power device," pp. 96–101, 2020.

[6] B. Quintana Galera, S. Prieto, A. Adan, and F. Bosché, "Door detection in 3d coloured point clouds of indoor environments," *Automation in Construction*, vol. 85, p. 146–166, 01 2018.

[7] A. Llopart, O. Ravn, and N. A. Andersen, "Door and cabinet recognition using convolutional neural nets and real-time method fusion for handle detection and grasping," pp. 144–149, 2017.

[8] B. Kakillioglu, K. Ozcan, and S. Velipasalar, "Doorway detection for autonomous indoor navigation of unmanned vehicles," pp. 3837–3841, 2016.

[9] H. R. Gonçalves and C. P. Santos, "Deep learning model for doors detection: A contribution for context-awareness recognition of patients with parkinson's disease," *Expert Systems with Applications*, vol. 212, p. 118712, 2023.

[10] K. M. Othman and A. B. Rad, "A doorway detection and direction (3ds) system for social robots via a monocular camera," *Sensors*, vol. 20, no. 9, 2020.

[11] Z. He and M. Zhu, "Real-time door detection for indoor autonomous vehicle," vol. 10420, pp. 177–184, 2017.

[12] M. M. Shalaby, M. A.-M. Salem, A. Khamis, and F. Melgani, "Geometric model for vision-based door detection," pp. 41–46, 2014.

[13] W. Chen, T. Qu, Y. Zhou, K. Weng, G. Wang, and G. Fu, "Door recognition and deep learning algorithm for visual based robot navigation," pp. 1793–1798, 2014.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016, visited on 2024-05-10.

[15] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Doll'a r, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[16] M. Naji, "doors dataset," https://universe.roboflow.com/mohammed-naji/doors-6g8eb, feb 2022, visited on 2024-01-14. [Online]. Available: https://universe.roboflow.com/mohammed-naji/doors-6g8eb

[17] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets, A. Gokaslan, V. Vondruš, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra, "Habitat 2.0: Training home assistants to rearrange their habitat," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 251–266. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/021bbc7ee20b71134d53e20206bd6feb-Paper.pdf

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016, visited on 2024-05-11.

[19] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," 2017, visited on 2024-05-11.

[20] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, 1981.

[21] E. Davies, *Machine Vision: Theory, Algorithms, Practicalities*, ser. Signal Processing and its Applications. Elsevier Science, 2004. [Online]. Available: https://books.google.cz/books?id=uY-Z3vORugwC

[22] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1532–1545, 2014.

[23] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2019, visited on 2024-05-12.

[24] K. Othman and A. Rad, "Srin: A new dataset for social robot indoor navigation," *Glob. J. Eng. Sci*, vol. 4, 2020.

[25] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," *arXiv e-prints*, vol. abs/1812.11941, 2018. [Online]. Available: https://arxiv.org/abs/1812.11941

[26] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: a Line Segment Detector," *Image Processing On Line*, vol. 2, pp. 35–55, 2012, https://doi.org/10.5201/ipol.2012.gjmr-lsd.

[27] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[28] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller, "Zoedepth: Zero-shot transfer by combining relative and metric depth," 2023, visited on 2024-01-05. [Online]. Available: https://arxiv.org/abs/2302.12288

[29] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[30] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[31] R. Stephens, "Probabilistic approach to the hough transform," *Image and Vision Computing*, vol. 9, no. 1, pp. 66–71, 1991.

[32] I. Suárez, J. M. Buenaposada, and L. Baumela, "Elsed: Enhanced line segment drawing," *Pattern Recognition*, vol. 127, p. 108619, 2022.

[Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320322001005

[33] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[34] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[36] S. Gillies *et al.*, "Shapely: manipulation and analysis of geometric objects," toblerity.org, 2007–, visited on 2024-05-15. [Online]. Available: https://github.com/Toblerity/Shapely