FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Power line vegetation management using UAV images
Student:	Bc. Radek Ježek
Supervisor:	Ing. Lukáš Brchl
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of winter semester 2022/2023

Instructions

This thesis aims to create an algorithm for the automatic detection of vegetation encroachment in the power line corridors. The source images for the algorithm will be captured from the aerial perspective using an Unmanned Aerial Vehicle (UAV) and then further processed with methods of photogrammetry and computer vision.

1. Research existing solutions for automated power line corridor inspection.
2. Create a representative photogrammetric dataset with UAV.
3. From the captured images create a detailed 3D representation of the power line corridor.
4. Design and implement a robust power line detection algorithm and measurement of vegetation encroachment within the power line corridor.
5. Visualize the detected power lines and locations of problematic vegetation.
6. Evaluate the results and suggest future improvements.

Electronically approved by Ing. Karel Klouda, Ph.D. on 27 August 2021 in Prague.

Master's thesis

POWER LINE VEGETATION MANAGEMENT USING UAV IMAGES

Bc. Radek Ježek

Faculty of Information Technology
Department of Applied Mathematics
Supervisor: Ing. Lukáš Brchl
December 27, 2021

Contents

List of Figures

List of Tables

List of Listings

I would like to thank my supervisor Ing. Lukáš Brchl for all his help and dedicated support, especially on maintaining and piloting the UAV platform. I am also thankful to Ing. Petr Kasalický for helping with connecting and setting up processing in the available cloud infrastructure. Finally, I would like to thank my family and friends for their immense support and patience throughout writing this thesis.

Declaration

In Prague on December 27, 2021 . . . .

Abstrakt

Provozovatelé elektrických distribučních sítí vynakládají každoročně velké množství peněz a úsilí, aby zajistili plynulou a bezpečnou dodávku elektřiny. Nejčastějším zdrojem výpadků proudu je poškození drátů vysokého napětí zásahem vegetace, například spadaných stromů. Z toho důvodu provozovatelé provádějí údržbu a pravidelné inspekce koridorů s elektrickým vedením, především v lesích a hustě zarostlých oblastech. Tím vytváří poptávku po nenákladných a vysoce automatizovaných metodách pro průzkum ochranných pásem elektrického vedení. Cílem této práce je vytvořit robustní algoritmus pro automatickou detekci zásahů vegetace do ochranného pásma elektrického vedení pomocí bezpilotních letadel (dronů), s využitím metod z fotogrammetrie a počítačového vidění. Studie pokrývá celý pracovní postup pro inspekci ochranného pásma drátů vysokého napětí, od obsáhlých pokynů pro sběr dat, přes 3D rekonstrukci elektrického vedení, až po detekci zásahů vegetace a vizualizaci výsledků.

Klíčová slova dráty vysokého napětí, správa vegetace, UAV, fotogrammetrie, 3D rekonstrukce

Abstract

The electric utility companies spend large amounts of money and effort every year to ensure the safe and uninterrupted operation of the electric power infrastructure. The most common source of outages is vegetation damaging power lines, for example, fallen trees. For this reason, companies perform regular inspections and maintenance of power line corridors, especially in forests and densely vegetated areas, creating a high demand for inexpensive and highly automated methods of power line corridor surveys. This work aims to create a robust algorithm for automatic detection of vegetation encroachment in the power line corridor using an Unmanned Aerial Vehicle (UAV), the techniques of photogrammetry, and computer vision. The study will cover the workflow for power line corridor inspection from comprehensive guidelines for data acquisition through power line 3D reconstruction to vegetation encroachment detection and visualization of the results.

Keywords power lines, vegetation management, UAV, photogrammetry, 3D reconstruction

List Of Abbreviations

API	Application Programming Interface
ATV	All-Terrain Vehicle
DEM	Digital Elevation Model
DSM	Digital Surface Model
EXIF	Exchangeable Image File Format
FN	False Negative
FP	False Positive
GCP	Ground Control Points
GPS	Global Positioning System
GPU	Graphics Processing Unit
GSD	Ground Sample Distance
GUI	Graphical User Unterface
IoU	Intersection over Union
LiDAR	Light Detection and Ranging
MLS	Mobile Laser Scanning
MVS	Multi View Stereo
PLS	Personal Laser Scanning
PPK	Post-Processed Kinematic
RANSAC	Random Sample Consensus
RGB	(Red, Green, Blue) color representation
RTK	Real-Time Kinematic
SAR	Synthetic Aperture Radar
SIFT	Scale-Invariant Feature Transform
TLS	Terrestrial Laser Scanning
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
UAV	Unmanned Aerial Vehicle
UTM	Universal Transverse Mercator coordinate system
WGS84	World Geodetic System 1984

Chapter 1
Introduction

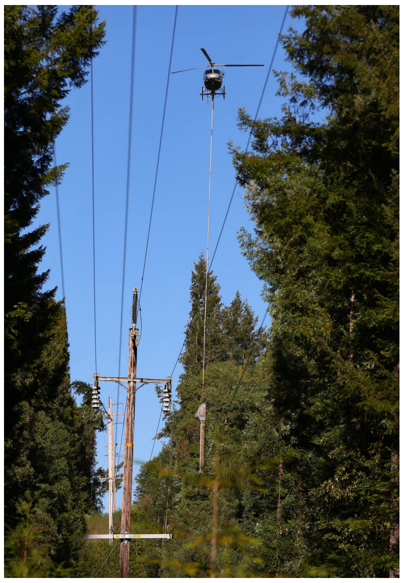
Power lines are omnipresent in our modern society and supply us with one of the essential utilities we rely on heavily – electricity. The importance of electricity does not need to be emphasized, as anyone can imagine, just how much of today's critical infrastructure is built on it. Banks, hospitals, even nuclear power plants have strategies for coping with power outages, usually diesel generators or emergency batteries, that help them survive several hours with no electricity. For example, if the Thames Barrier (a movable flood barrier on the river Thames) failed at the wrong time, 125 square kilometers of central London would be flooded, causing a major disaster [1].

The uninterrupted supply of electricity is thus critical for modern civilization, which is why the electric power industry makes great efforts to ensure it. One of the most common causes for outages is vegetation coming into contact with exposed wires of the power lines. It can be trees that fell during a thunderstorm or vegetation growing too close to the wires in an unmaintained power line corridor. In the best-case scenario, this will only cause a power supply failure. However, this may be much more dangerous during hot summer days, as it can cause a grave wildfire, such as the Camp Fire in California. This wildfire was a major disaster in 2018 that cost 85 lives and burned more than 62 thousand hectares, making it the most destructive fire in California history [2]. The company, Pacific Gas and Electricity (PG&E) that was operating the power lines, later filed for bankruptcy and made a settlement of $13.5 billion going to the victims, as it faced potential liabilities of $30 billion [3]. This is just one of many examples where vegetation growing too close to the power lines caused severe damage and even casualties.

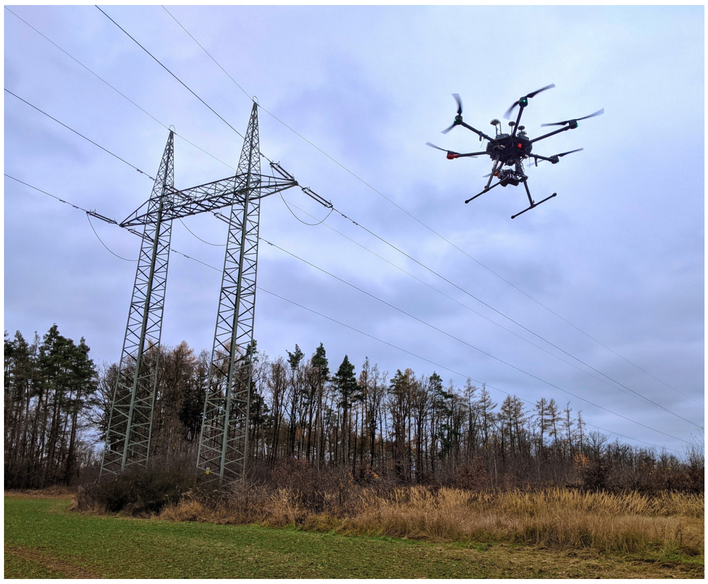
As a result, the electric power industry must actively prevent the danger by regularly monitoring, inspecting, and maintaining the power lines, especially in densely vegetated areas and forests. It is not easy, as the power lines often span hundreds of kilometers, sometimes in difficult or inaccessible terrain. Trimming the vegetation is time-consuming and expensive work, involving human crews cutting trees or even a helicopter with enormous chainsaws hanging down below (see Figure 1.1a). The high demands for resources created the need for precise and regular assessment of the state of the vegetation in power line corridors.

As of today, various techniques are being applied for identifying vegetation risks around power lines. These comprise human field inspections and remote sensing methods, including satellite imagery or laser and optical scanning from helicopters. Recently, the improvements in Unmanned Aerial Vehicles (UAV), especially multi-rotor and fixed-wing drones, allow for more efficient and less expensive workflows. The UAVs are controlled remotely and can fly autonomous missions, requiring a far lower level of expertise than flying a helicopter. Moreover, the equipment is much cheaper than other methods, and its ability to fly closer to the power lines allows for a higher level of detail, especially in difficult or inaccessible terrain.

The purpose of this work is to create a robust, highly automated algorithm for assessing vegetation risks near power lines by providing a detailed 3D reconstruction of the power lines



(a) Trimming vegetation using a big chainsaw hanging from a helicopter.



(b) A multi-rotor UAV taking images of the power lines during our experiments.

Figure 1.1 Power line vegetation management and corridor inspection.

and the surrounding terrain. We use a multi-rotor UAV with a high-end consumer camera (Figure 1.1b) and the science of photogrammetry, computer vision, and machine learning to achieve this goal.

The work is organized as follows: a chapter summarizing all the theoretical background and related knowledge required for this work, followed by a research of existing vegetation inspection methods and power line detection algorithms. Afterward, we propose an integrated and highly automated workflow covering the entire process from data acquisition to visualization of the results. We conclude with experiments and evaluation of our solution on three representative datasets created for this work.

Chapter 2

Theoretical Background

This chapter first introduces vegetation management near power lines, followed by a basic introduction to computer vision and the tasks relevant for power line 3D reconstruction. After that, we outline the science of photogrammetry, focusing on the mathematical model of a camera and the basic theory of image projection. The chapter does not aim to be a comprehensive explanation of all the topics, rather a brief overview comprising the knowledge necessary to understand the rest of this work. We provide a mathematical basis only for a small subset of these topics, mainly where mathematics is essential for implementation.

2.1 Vegetation management near power lines

Vegetation management has been a long-standing topic in the electric power industry. Its purpose is to ensure the safe and uninterrupted operation of the outdoor electric power infrastructure. The main focus of vegetation management is on forests or other densely vegetated areas containing power lines, where nature could come into contact with an exposed wire and cause an outage or even fire.

2.1.1 Power Line Corridor

For ensuring a safe distance between the wires and vegetation, the law establishes a power line right of way. According to the regulations in the Czech Republic, it is defined as a continuous space between two vertical planes on each side of the wires. The distance of the planes is measured from the outermost wire and varies based on the voltage (Figure 2.1) [4]. Within this area, plants are allowed to reach only a certain height – 1 to 4.5 m far from the closest wire, depending on the voltage and type of vegetation. It is typically defined by technical standards, such as [5]. We will use the term power line corridor to refer to the definition above throughout this work.

2.1.2 Physical Model of a Power Line

Cables, chains, ropes, and power lines form curves with a unique shape determined by the weight and tension of the material. This curve is called catenary, and in 2D, the point $[x, y]^T \in \mathbb{R}^2$ lies on the curve if it satisfies Equation [6]:

$$y = c \cosh \frac{x}{c}, \tag{2.1}$$

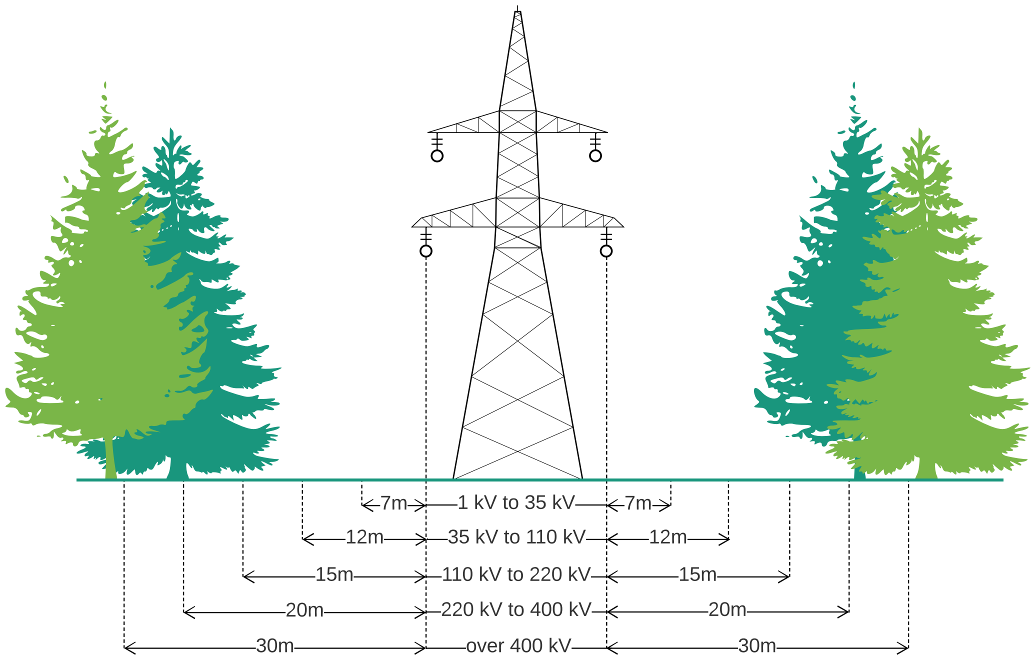


Figure 2.1 Power line right of way based on voltage (upper bounds are inclusive) [4].

where $c \in \mathbb{R}$ is a scaling factor encompassing the weight and tension of the cable. This can be further extended by shifting the catenary in space using two translation parameters $a, b \in \mathbb{R}$ [6]:

$$y = a + c \cosh \frac{(x-b)}{c}. \tag{2.2}$$

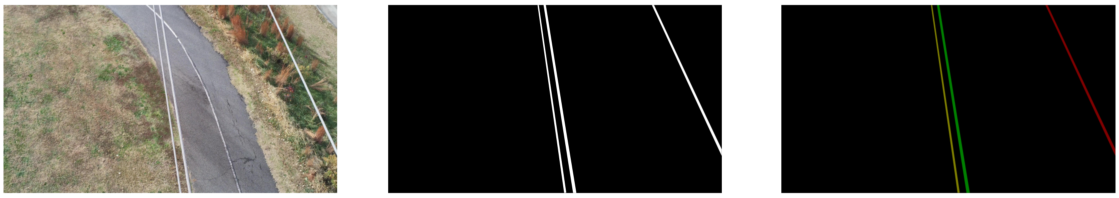
2.2 Computer Vision

Computer vision emerged from the growing desire to mimic human vision. Since the beginning of computers, humans strove to create a machine comparable to them in challenging tasks, such as understanding the world around us. Computer vision aims to solve this problem by understanding the images taken by the best approximation of human eyes yet – cameras [7].

This section outlines the essential tasks of computer vision applicable for power line 3D reconstruction. The mathematical model of a camera and image projection theory, although part of computer vision, is described in a Section 2.3. dedicated to photogrammetry.

2.2.1 Image

An *image* in computer vision is often defined as a continuous or discrete function, mapping the (x, y) coordinate in the image plane to a brightness value, inherently modeling the image as a signal [7]. We simplify the matter and define the image as *pixel data* in the spatial domain, consistent with [8, 9]:



(a) Input image. **(b)** Semantic segmentation. **(c)** Instance segmentation.

Figure 2.2 Input image and its semantic and instance segmentation masks [11].

▶ **Definition 2.1.** *Image is a sensor data matrix indexed from zero, $I \in \{0, \ldots, 255\}^{h,w,c}$, where h and w is the image height and width in pixels respectively, where $I_{ijk} \, \forall i \in \{0, \ldots, h-1\}, \forall j \in \{0, \ldots, w-1\}, \forall k \in \{0, \ldots, c-1\}$ represents the brightness value of the particular channel c.*

- For $c = 3$, the definition mirrors the output from a typical consumer digital camera – a three-dimensional matrix comprising three 2D matrices, one for each color in the RGB (red, green, blue) color representation.
- For $c = 1$, we use the term grayscale image, which can be obtained either by a grayscale camera or by averaging the color values of an RGB camera.

Without loss of generality, we limit the range of the brightness values to $\{0, 1, \ldots, 255\}$, which is the standard for most cameras. By pixel, we mean c brightness values (one for each channel) $I_{ij:}$ with (sensor) coordinates $[i, j]^T$.

2.2.2 Image Segmentation

Image segmentation is one of the key tasks in computer vision. The objective is to identify regions in the image containing real-world objects of our interest. For this work, we only assume a single class of objects – the power lines, although, in literature, authors often work with multiple classes [10].

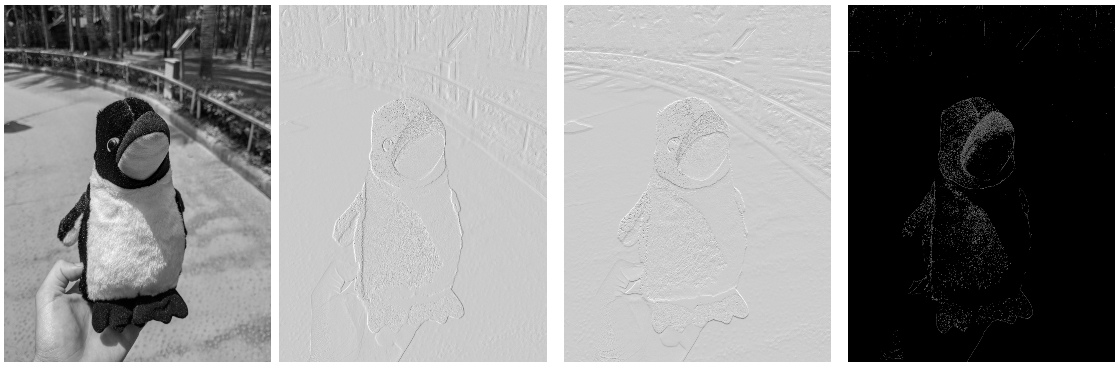
For single class problems, we distinguish between two types – semantic and instance segmentation:

Semantic segmentation – given the input image $I \in \{0, \ldots, 255\}^{h,w,c}$, the semantic segmentation task is to find a binary segmentation mask $M_{semantic} \in \{0, 1\}^{h,w,1}$, where the value 1 corresponds to pixels containing the objects of interest and 0 is the background. An example semantic segmentation mask is illustrated in Figure 2.2b.

Instance segmentation – compared to semantic segmentation, instance segmentation distinguishes between individual instances of the objects of interest – power lines in our case. Given the input image $I \in \{0, \ldots, 255\}^{h,w,c}$, the instance segmentation task is to find a segmentation mask $M_{instance} \in \mathbb{N}_0^{h,w,1}$, where the value 0 is the background and $i \in \mathbb{N}_+$ are unique identifiers for each individual instance of the objects of interest. An example instance segmentation mask is illustrated in Figure 2.2c.

2.2.3 Linear Filtering

Linear image filtering is one of the fundamental preprocessing techniques used in many computer vision methods, and it is especially important for line detection in images. The cornerstone of linear filtering is mathematical convolution with a special matrix called filter kernel.



(a) Input image. **(b)** Vertical Prewitt. **(c)** Horizontal Prewitt. **(d)** Canny detector.

Figure 2.3 Input image, the result of convolution with Prewitt filters (normalized), and the result of Canny edge detector.

Convolution of the grayscale image $I \in \{0,\dots,255\}^{h,w,c}$ with a filter kernel $F \in \mathbb{R}^{m,n,c}$, as defined in [8, p. 328], generalized for c channels:

$$O_{ij0} = \sum_{p=-m}^{m} \sum_{q=-n}^{n} \sum_{k=0}^{c-1} I_{i-p,j-q,k} \cdot F_{p,q,k} \tag{2.3}$$

A careful reader will notice that we did not specify the domain for i and j in the definition above. This is because for $i \in \{0,\dots,h-1\}$, and $j \in \{0,\dots,w-1\}$, the formula would be accessing elements outside the image I. This is usually solved by defining padding, e.g., zero padding: $I_{i,j,k} := 0,\ \forall i \in \mathbb{Z} \setminus \{0,\dots,h-1\},\ \forall j \in \mathbb{Z} \setminus \{0,\dots,w-1\},\ \forall k \in \{0,\dots,c-1\}$ [8, p. 328].

The output O has only one channel and is not technically an image by our Definition 2.1 because it contains real numbers. For that reason, O is usually normalized or further processed, such that it fits into $\{0,\dots,255\}^{h,w,1}$.

Filtering is often used to extract features, such as edges, or perform other operations, e.g., blurring or sharpening. The kernel is usually square ($m = n$) with m equal to a small odd number, i.e., $F \in \mathbb{R}^{3,3,c}$ or $F \in \mathbb{R}^{5,5,c}$. Equation 2.4 shows example kernels for edge feature extraction with the resulting images O after normalization in Figure 2.3b and 2.3c.

2.2.4 Edge Detection

An edge in an image is a pixel, where the brightness changes abruptly [7, p. 133]. Finding edges is crucial for many computer vision algorithms, and can be utilized for an edge-based segmentation and line detection.

One of the most common methods for detecting edges is linear filtering described above. An example of an edge filter is the horizontal and vertical Prewitt filter kernel [7, p. 137]:

$$F_{vertical} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad F_{horizontal} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.4}$$

Figures 2.3c and 2.3b show the results of convolving the input image 2.3a with the Prewitt kernels above.

Filter convolution is often immediately followed by image thresholding, which transforms the output grayscale image into a binary segmentation mask keeping only the strongest edges. The most basic binary thresholding methods simply replace all values of the input image I above some specified threshold t with 1 and all the others with 0:

$$M_{ij0} = \begin{cases} 1 \text{ if } I_{ij0} > t \\ 0 \text{ otherwise} \end{cases} \tag{2.5}$$

Other filters, e.g., Sobel or Laplace and more sophisticated preprocessing pipelines, combining multiple steps and filters, are often utilized to achieve different results. One of the most popular advanced methods is the Canny edge detector (output in Figure 2.3d). A detailed description of these methods is out of scope of this work and can be found in [7, p. 116–173].

2.2.5 Neural Network Segmentation

An alternative to edge-based segmentation might be the use of the ever-so-popular neural networks. Just as computer vision was created to mirror human vision, the first neural networks were inspired by their biological counterparts. The artificial neuron takes k weighted input signals, accumulates them, and passes them through a nonlinear activation function. The weight of each signal determines its strength and therefore controls the contribution of the input. The neurons are stacked in connected layers, the first layer takes the input, and the last layer outputs the prediction. The weights can be trained using gradient descent with training pairs of input and output values. This way, neural networks are able to solve various regression and classification tasks.

As this field progressed, neural networks became more complicated, and specialized neural network architecture for solving computer vision tasks emerged, in particular the *convolutional* neural network. This type of network was designed to work with images and utilizes convolution with a filter kernel, as described in Section 2.2.3. The crucial difference is that the kernel does not contain fixed values, like the Prewitt kernel (Equation 2.4), but trainable weights.

This work focuses on solving the semantic segmentation task using a neural network. This task requires a complicated neural network architecture heavily based on convolution. The training pairs consist of the input image and the output binary segmentation mask. The description of this type of network is out of scope of this work. Instead, we refer to a particular implementation [12] that describes the algorithm in detail.

Evaluation Metrics

If we take each pixel as a separate sample, we can reformulate the binary segmentation as a binary classification task, assigning either positive (1) or negative (0) labels to data samples. For binary classification, there are numerous metrics that measure different aspects of the performance. Most of them use the following division of data samples by comparing the predicted label ($\hat{y}$) vs. ground-truth (y):

True Positive (TP) – number of samples, where $\hat{y} = 1$, and $y = 1$.

True Negative (TN) – number of samples, where $\hat{y} = 0$, and $y = 0$.

False Positive (FP) – number of samples, where $\hat{y} = 1$, and $y = 0$.

False Negative (FN) – number of samples, where $\hat{y} = 0$, and $y = 1$.

This is often summarized to a so-called confusion matrix:

	$y = 1$	$y = 0$
$\hat{y} = 1$	TP	FP
$\hat{y} = 0$	FN	TN

For measuring the trained neural network performance, we use four metrics common for image segmentation:

True Positive Rate (TPR) measures the ratio of correctly classified positive samples to all positive samples: $\frac{TP}{TP+FN}$.

True Negative Rate (TNR) measures the ratio of correctly classified negative samples to all negative samples: $\frac{TN}{TN+FP}$.

Intersection over Union (IoU) calculates the ratio of the intersection of the ground truth and predicted segmentation masks ($\hat{y} = 1$ and $y = 1$) over a union of them ($\hat{y} = 1$ or $y = 1$) – $\frac{TP}{TP+FN+FP}$. For image segmentation it can be visually interpreted as the overlapping areas divided by the combined areas of the ground-truth and predicted segmentation masks.

Overall accuracy measures the ratio of correctly classified to all samples $\frac{TP+TN}{TP+TN+FP+FN}$. This is however misleading for unbalanced datasets. In our case, the power lines form only a small percentage of the image. If the model predicted $\hat{y} = 0$ for all pixels, it would achieve great overall accuracy, but the model would be useless.

2.2.6 Line Detection

The need for line detection naturally appears in many problems, such as lane detection in the automotive industry [13] and vanishing point detection for image rectification [14], to name a few.

The input for line detection is typically a binary segmentation mask containing raw edges from edge detection (described in Section 2.2.4) or objects of interest obtained by some additional processing, e.g., using a neural network described above.

Hough Line Transform

The prevailing method for line detection is the Hough line transform. This algorithm works with lines parametrized as $\rho = x \cos\theta\, y \sin\theta$, where ρ is the distance from the origin of the image, and θ is the angle with the x-axis, as illustrated in Figure 2.4b. The pixel $[x_0, y_0]^T$ corresponds to a pencil of lines (outlined by lines 1–4 in Figure 2.4b). Figure 2.4c further shows that the pencil of lines corresponding to a single pixel forms a curve in the ρ, θ parametric space. If we take many pixels in the image corresponding to a real line, their curves in the parametric space will intersect. The algorithm records a small contribution for each curve, and then simply returns the local maxima in the ρ, θ plane [15].

There are multiple versions of this algorithm. In this work, we use a *probabilistic* Hough line transform, which is more efficient and also retrieves the coordinates of the beginning and end of each line.

2.3 Photogrammetry

The term photogrammetry comes from the three Greek words *phot* (light), *gramma* (something drawn), and *metrein* (measure), which constitute the science of taking measurements from photographs. It is deeply rooted in computer vision, sharing most of the same concepts.

This section is a brief overview of the photogrammetric models and methods required for this work, abstracting many details. We mostly follow [17], which provides a comprehensive explanation of the theory.

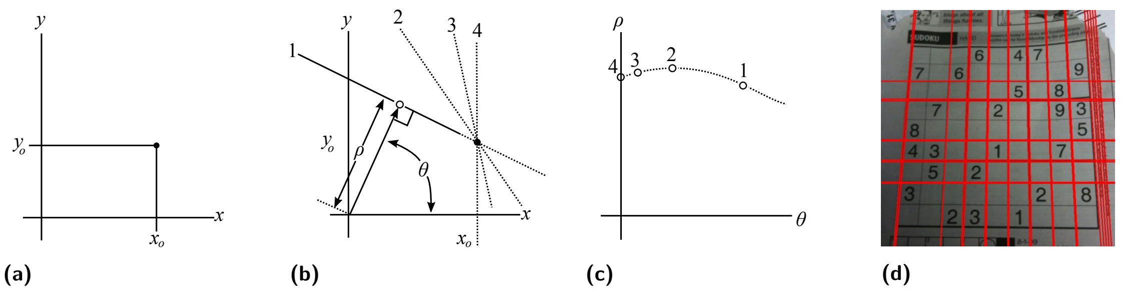


Figure 2.4 Hough line transform. A point $[x_0, y_0]^T$ in **(a)** corresponds to a pencil of lines (for example 1–4) in **(b)**. These lines form a sinus-like curve in the parametric ρ, θ space **(c)**. If we sample many pixels from the actual line 1, their curves in the parametric space will intersect [15]. **(d)** shows Hough line transform applied on a sudoku image (after Canny edge detection) [16].

2.3.1 Homogeneous Coordinates

In photogrammetry, many entities, such as points, lines, and transformations, are often expressed in so-called homogeneous coordinates. These particular coordinates make certain geometric operations and transformations simpler and expressible by matrix-vector multiplication. These include rotations, translations, projective transformations, finding the line intersections, etc. The transformations, represented by matrices, can easily be chained to form compact representations of a complex reality. More examples can be found in [17, p. 247–324] and the following sections.

▶ **Definition 2.2** (J. Plücker 1829). *Homogeneous coordinates* $\mathbf{x}$ *of a geometric entity* χ *are invariant with respect to multiplication by a scalar* $\lambda \neq 0$*: thus* $\mathbf{x}$*, and* $\lambda\mathbf{x}$ *represent the same entity* χ.

We can utilize this property to illustrate the relation between the homogeneous and commonly used Euclidean coordinates. In [17, p. 199], the authors demonstrate this on a simple case of a 2D point. The homogeneous coordinates of a 2D point $\mathbf{x}$ with inhomogeneous coordinates $\boldsymbol{x} = [x, y]^T \in \mathbb{R}^2$ are defined as

$$\chi(\mathbf{x}) = \begin{bmatrix} \boldsymbol{x}_0 \\ x_h \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix}$$

where the factor $w \neq 0$ can be chosen arbitrarily. The $\boldsymbol{x}_0$ is called the Euclidean part, whereas x_h is the homogeneous part. Consequently, given the homogeneous coordinates of a 2D point, we can obtain the Euclidean coordinates simply by dividing by the third element and then taking only the first two elements, i.e., following Euclidean normalization [17, p. 206]:

$$\mathbf{x}^e = \frac{1}{w} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \frac{1}{w} \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

This relation is further visualized in Figure 2.5. Homogeneous coordinates are not limited to 2D points. In fact, many other entities such as 2D lines and 3D points and even transformations (e.g., the essential or fundamental matrix discussed in the sections below) are homogeneous entities as long as they conform to the invariance to multiplication by a scalar.

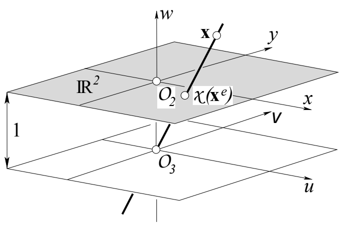


Figure 2.5 Relation between homogeneous and Euclidean coordinates. If we embed the real plane $\mathbb{R}^2$ into a 3D space $(u, v, w) \in \mathbb{R}^3$, with origin O_3, a point with Euclidean coordinates $\chi(\mathbf{x}^e)$ lying in the plane $w = 1$ is represented in homogeneous coordinates by any point $\mathbf{x}$ on the line joining O_3 and the point χ [17, p. 200].

As opposed to Euclidean space, which contains elements in inhomogeneous coordinates, the homogeneous coordinates can be used to define a projective space $\mathbb{P}^n(\mathbb{R})$, as in [17, p. 215]:

▶ **Definition 2.3.** *The projective space $\mathbb{P}^n(\mathbb{R})$ contains all $(n+1)$-dimensional points χ with homogeneous real-valued coordinates $\mathbf{x} \in \mathbb{R}^{n+1} \setminus \mathbf{0}$,*

$$\chi(\mathbf{x}) \in \mathbb{P}^n(\mathbb{R}) : \mathbf{x} \in \mathbb{R}^{n+1} \setminus \mathbf{0}$$

with

$$\chi(\mathbf{x}) \equiv y(\mathbf{y}) \iff \mathbf{x} = \lambda\mathbf{y}, \textit{ for some } \lambda \neq 0.$$

The projective space forms the basis of projective geometry, which is extensively used in photogrammetry. As in [17], we will denote points in homogeneous coordinates by upright letters and Euclidean by inclined, i.e., $\mathbf{X}$ vs. $\boldsymbol{X}$. We will also use lowercase letters for 2D points and uppercase for 3D points. In this summary, we only scratched the surface of this area of mathematics. For more details on homogeneous coordinates and their use in various geometric operations and transformations, we refer the reader to [17, p. 195–324].

2.3.2 Camera Model

The simplest model of a camera is a so-called pinhole camera. The term dates deep into history, long before the invention of the digital camera. The principle is straightforward, take a dark room with blinds over the windows with a tiny hole – the size of a pin. The light passes through the hole and projects an upside-down image of the outside world on the wall. Such a setting is called camera obscura and was first described in China in the 5th century BC. Figure 2.6a shows how the camera obscura was used to observe the sun. Figure 2.6b describes the mathematical model of the pinhole camera. It also highlights a few important terms:

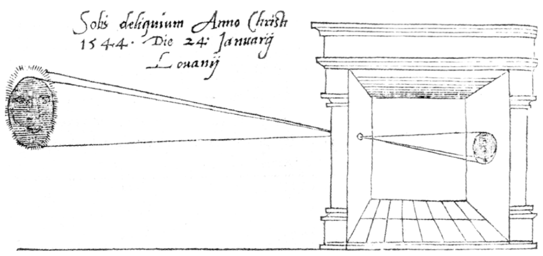
Image plane $\mathcal{I}$ is the plane where the image is projected. This is where we would typically place a sensor or a film to capture the image.

Principal point $\mathcal{H}$ is the center of the image plane.

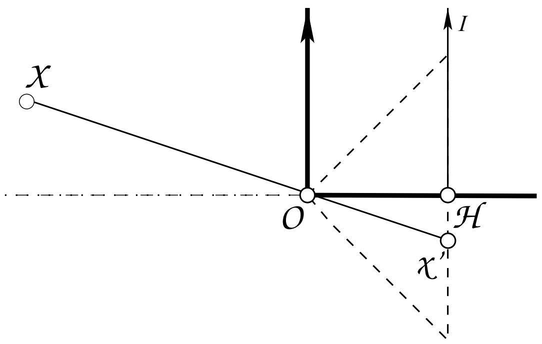
Projection center $\mathcal{O}$ is the single point through which all the light rays pass.

Optical axis is the line passing through $\mathcal{O}$ and $\mathcal{H}$,

The problem with a pinhole camera is that the pinhole must be very small to get a sharp image, limiting the amount of light passing through. For overcoming this issue, lenses were invented. The model of a camera with a lens involves other parameters, such as focal length, and introduces several optical defects like diffraction, vignetting, chromatic aberration, and nonlinear distortion. For brevity, we omit the details and instead refer the reader to [17, p. 256, 461] and [18], where the authors explain the geometry of thin and thick lenses and aberrations, respectively.



(a) The first published picture of camera obscura [20].



(b) Pinhole camera model projecting a 3D point $\mathcal{X}$ into the 2D image point $\mathcal{X}'$ [17, p. 469, simplified by author].

Figure 2.6 Camera obscura and the simplest model of a camera – the pinhole camera.

For power line 3D reconstruction, we use a perspective camera model with distortion, following the authors of [17, p. 462] and [19], which is sufficient for this work and most photogrammetry applications in general. The model assumes a thick lens geometry and accounts for lens distortion. The process of capturing an image with this model is described in more detail in the next section.

2.3.3 Projection of a 3D point into 2D pixel coordinate

For 3D reconstruction from images, we need to describe the image capture as a mapping from a real-world point coordinate to the pixel coordinate in the resulting image. In mathematical terms, we are looking for the transformation:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.6}$$

where $[x, y]^T \in \mathbb{R}^{2,1}$ are the Euclidean pixel coordinates, $[X, Y, Z]^T \in \mathbb{R}^{3,1}$ are Euclidean 3D world (object) coordinates and P is the projection matrix.

Using the perspective camera model with distortion, we follow the authors of [17, p. 462] and [19], who break down the process into several steps, using four coordinate systems:

Object coordinate system S_o is written as $[X, Y, Z]^T$ and is often also called world or scene coordinate system. This usually represents the real-world 3D coordinates, either GPS position or some arbitrary reference frame.

Camera coordinate system S_c is written as $[{}^cX, {}^cY, {}^cZ]^T$ and represents the view of the world from the position of the camera, with the camera projection center being at the origin.

Image coordinate system S_i is written as $[{}^ix, {}^iy]^T$ and describes the image plane. Its origin is the principal point of the camera, and it is parallel to the camera coordinate system, only shifted by the camera constant in the direction ${}^cZ$.

Sensor coordinate system S_s is written as $[{}^sx, {}^sy]^T$, and characterizes the pixel coordinates as in the common image matrix produced by digital cameras with the origin at the top left corner of the image ($I_{0,0,:}$ according to Definition 2.1).

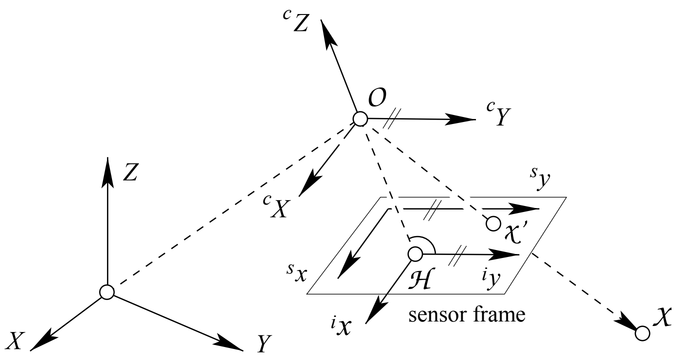
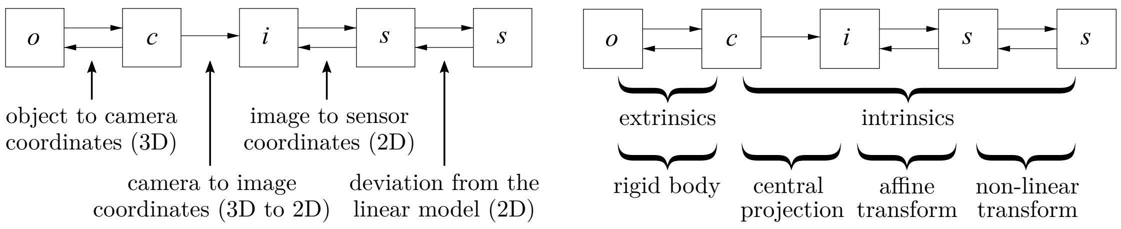


Figure 2.7 Perspective projection of a 3D point $\mathcal{X}$ into the 2D image point χ'. The four coordinate system in Figure: $[X, Y, Z]^T$ – object coordinate system, $[{}^cX, {}^cY, {}^cZ]^T$ – camera coordinate system, $[{}^ix, {}^iy]^T$ – image coordinate system, $[{}^sx, {}^sy]^T$ – sensor coordinate system. $\mathcal{O}$ and $\mathcal{H}$ are the camera projection center and principal point, respectively [17, p. 462, simplified by author].



o
c
i
s
s
object to camera coordinates (3D)
image to sensor coordinates (2D)
camera to image coordinates (3D to 2D)
deviation from the linear model (2D)
o
c
i
s
s
extrinsics
intrinsics
rigid body
central projection
affine transform
non-linear transform

(a) Point projection focusing on coordinate systems. **(b)** Point projection focusing on transformations.

Figure 2.8 Different views on perspective projection [19, modified by author].

As a result, there are four steps in the mapping process (Figure 2.8a) [17, p. 459–475], [19]:

1. From object coordinates to camera coordinates – a rigid body transformation involving the camera rotation and translation with respect to the origin of the object coordinate system. In Euclidean coordinates, this reads ${}^c\boldsymbol{X} = R\boldsymbol{X} + \boldsymbol{T}$, where $\boldsymbol{X}$ is the original point in object coordinates, ${}^c\boldsymbol{X}$ is the point in camera coordinates, $\boldsymbol{T}$ is the translation vector $[T_0, T_1, T_2]^T$ and R is the matrix rotating the camera coordinate system. R can be obtained from the three rotation parameters (yaw, pitch, roll) using the Rodrigues formula [21]. In homogeneous coordinates, this is represented as multiplying $\mathbf{X}$ (homogeneous representation of $\boldsymbol{X}$) by two matrices:

   $$\text{a rotation matrix } \begin{bmatrix} R & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}, \text{ followed by translation matrix } \begin{bmatrix} I_3 & \boldsymbol{T} \\ \mathbf{0}^T & 1 \end{bmatrix},$$

   where I_3 is the 3×3 identity matrix. Combined, we obtain:

   $${}^c\mathbf{X} = \begin{bmatrix} I_3 & \boldsymbol{T} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} R & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{X} = [R \mid \boldsymbol{T}]\,\mathbf{X}$$

   ▶ Note 2.4. Here we diverge from [17, p. 466] and instead follow the convention of [22, p. 24] to describe the transformation. The difference is in the order of the operations, where [17, p. 466] first translates the point and then rotates it to obtain the correct coordinates, whereas we first apply rotation and then translation. This is consistent with the 3D reconstruction algorithm we chose in Section 5.2.

2. From camera coordinates to image coordinates – as Figure 2.7 shows, the two coordinate systems are parallel and the origins $\mathcal{O}$ and $\mathcal{H}$ are shifted by some distance in the ${}^cZ$ direction.

This distance is called the camera constant and is typically denoted by f – as it represents the camera's focal length. This mapping is not invertible and in homogeneous coordinates is expressed as:

$$^{i}\mathbf{x} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} {}^{c}\mathbf{X}$$

After this step, we have arrived at a pinhole or ideal camera model.

3. From image coordinates to sensor coordinates – the image coordinates have their origin at the center of the image – the principal point, whereas the sensor of a camera typically stores the image as a matrix of pixels starting with pixel $[0,0]^T$ at the top left corner of the image. Therefore we shift the coordinates by x_H and y_H. We also take into account the scale difference between the width and height m and often negligible sheer s. Put together, we get a mapping:

$$^{s}\mathbf{x} = \begin{bmatrix} 1 & s & x_H \\ 0 & 1+m & y_H \\ 0 & 0 & 1 \end{bmatrix} {}^{i}\mathbf{x}$$

4. Correcting for nonlinear errors – this stage compensates for errors caused by lens distortion and imperfections, for example, cushion or barrel distortion. As we cannot express this step as a linear transformation, it is typically done separately as a so-called image undistortion.

Combining all the steps above, we arrive at the desired projection matrix P from Equation 2.6:

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} 1 & s & x_H \\ 0 & 1+m & y_H \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \left[R \mid \boldsymbol{T}\right] \mathbf{X} \\ \mathbf{x} &= \begin{bmatrix} f & fs & x_H \\ 0 & f(1+m) & y_H \\ 0 & 0 & 1 \end{bmatrix} \left[R \mid \boldsymbol{T}\right] \mathbf{X} \\ \mathbf{x} &= K \left[R \mid \boldsymbol{T}\right] \mathbf{X} \qquad (2.7) \\ \mathbf{x} &= P\mathbf{X} \end{aligned}$$

The projection matrix $P = K\left[R \mid \boldsymbol{T}\right]$ is often also called *camera matrix*.

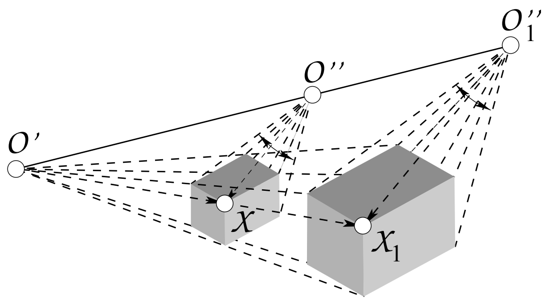
The matrix K in Equation 2.7 gathers all the linear intrinsic parameters. Together with the distortion coefficients from step 4 of the process above, it determines the camera *intrinsics*, a set of constant parameters that do not change from shot to shot. If these parameters are known (i.e., obtained by camera calibration), we talk about a *calibrated* camera. Otherwise, we refer to the camera as *uncalibrated*.

The rest of the parameters – the rotation and translation of the camera are called *extrinsics*, and they vary between shots. The intrinsics and extrinsics in the context of perspective projection are shown in Figure 2.8b.

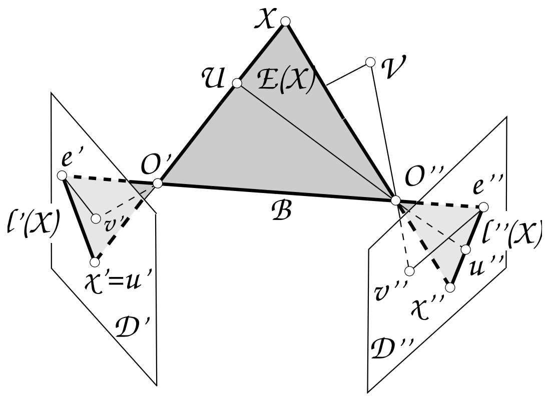
2.3.4 Epipolar geometry

Now that we described the geometry of a single camera, we can define the geometric properties of an image pair. In this section, we only state the essential characteristics without their derivation. A detailed explanation can be found in [17, p. 547–620] and [23].

In this overview, we assume two cameras with the projection centers $\mathcal{O}'$ and $\mathcal{O}''$, image planes $\mathcal{D}'$, $\mathcal{D}''$ and projection matrices P' and P''. We further define a relative rotation matrix R_{12} and translation vector $\boldsymbol{T}_{12}$ of the second camera with respect to the first camera (using the camera coordinate system $S_{c'}$ of the first camera as defined in Section 2.3.3). Next, we describe important geometric entities as in [17, p. 563]:



(a) The absence of scale information in epipolar geometry. The same images can be generated from several different viewpoints if the scene is scaled appropriately. [17, p. 551].



(b) Geometric description of two cameras observing the same scene – $\mathcal{B}$ is the epipolar axis, $\mathcal{X}\mathcal{O}'\mathcal{O}''$ is the epipolar plane, e' and e'' are the epipoles and $\ell'(\mathcal{X})$ and $\ell''(\mathcal{X})$ are the epipolar lines. $\mathcal{X}$, $\mathcal{U}$, $\mathcal{V}$ are 3D points in the scene and χ', u', v' and χ'', u'', v'' are their projection to image planes $\mathcal{D}'$ and $\mathcal{D}''$ respectively. [17, p. 563].

Figure 2.9 Geometry of an image pair.

Epipolar axis is the line connecting the two projection centers $\mathcal{O}'$ and $\mathcal{O}''$.

Epipolar plane is a plane defined by the two projection centers $\mathcal{O}'$ and $\mathcal{O}''$ and a 3D point X we chose to observe. That means that all the possible epipolar planes „rotate“ around the epipolar axis.

Epipoles are the camera projection centers projected to the image plane of the other camera. The epipole e' is the projection of $P'\mathcal{O}''$, and similarly $e'' = P''\mathcal{O}'$.

Epipolar lines $\ell'(\mathcal{X})$ and $\ell''(\mathcal{X})$ are the projections of the lines connecting $\mathcal{X}$ and $\mathcal{O}''$, and $\mathcal{X}$ and $\mathcal{O}'$ to the image planes $\mathcal{D}'$ and $\mathcal{D}''$, respectively. Also, they are the intersections of the epipolar plane with the image planes $\mathcal{D}'$ and $\mathcal{D}''$.

Figure 2.9a illustrates that it is impossible to estimate scale given only a pair of images. Indeed, cameras with projection centers $\mathcal{O}'$ and $\mathcal{O}''$ from Figure 2.9a generate the exact same image as cameras with projection centers $\mathcal{O}'$ and $\mathcal{O}''_1$ if the scene is scaled appropriately. Consequently, given a 2D point χ' in the image plane, we cannot determine the location of the original 3D point $\mathcal{X}$ because of the loss of information during step 2 of the perspective projection described in Section 2.3.3. This is demonstrated in Figure 2.9b, where two different 3D points $\mathcal{U}$ and $\mathcal{X}$ map to the same point $\chi' = u'$ in the $\mathcal{D}'$ image plane.

Nevertheless, there are some important implications from the image pair geometry, which we greatly exploit for the power line 3D reconstruction. Namely, from the coplanarity constraint [17, p. 552–553]:

▶ **Corollary 2.5.** *Given vectors $\mathcal{O}'\chi'$ and $\mathcal{O}''\chi''$ and using the fact that that vectors $\mathcal{O}'\chi'$, $\mathcal{O}'\mathcal{O}''$ and $\mathcal{O}''\chi''$ all lie in the epipolar plane, we can express the coplanarity constraint as*

$$\det\left[\mathcal{O}'\chi' \ \ \mathcal{O}'\mathcal{O}'' \ \ \mathcal{O}''\chi''\right] = 0$$

After a few derivations (see [17, p. 552–553]), one can arrive at the form

$$\mathbf{x}'^T F \mathbf{x}'' = 0, \tag{2.8}$$

where $\mathbf{x}'$ *and* $\mathbf{x}''$ *are the homogeneous image coordinates of the projections* $\mathcal{X}'$ *and* $\mathcal{X}''$ *(see Figure 2.9b), and* F *is called the* fundamental matrix. *Furthermore, following [23], we define an* essential matrix *as:*

$$E = [\mathbf{T}_{12}]_\times R_{12}, \tag{2.9}$$

where $\mathbf{T}_{12}$ *is a skew-symmetric matrix realizing the cross product operation with* $\mathbf{T}_{12}$. *Finally, assuming both images are taken with the intrinsic parameters* K*, the relationship between* F *and* E *is given as*

$$E = K^T F K. \tag{2.10}$$

▶ Note 2.6. Later, we use Equations 2.9 and 2.10 to reconstruct the fundamental matrix as:

$$F = K^{-T}[\mathbf{T}_{12}]_\times R_{12} K^{-1} \tag{2.11}$$

Given the two points $\mathcal{X}'$ and $\mathcal{X}''$, we can use the fundamental matrix to obtain the corresponding epipolar lines in the other image plane ($\ell''(\mathcal{X})$, and $\ell''(\mathcal{X})$, respectively). This can be achieved by simply multiplying the points by F or F^T. In homogeneous coordinates [17, p. 564]:

$$\mathbf{l}'(\mathcal{X}) = F\mathbf{x}'', \text{ and } \mathbf{l}''(\mathcal{X}) = F^T\mathbf{x}'.$$

2.3.5 Typical 3D Reconstruction Process

Equipped with this simplified mathematical description of the basic photogrammetric concepts, one can get a little insight into the typical process of 3D reconstruction from images. Let $\{I_i | I_i \in \{0, \dots, 255\}^{h,w,3}\}_{i=1}^n$ be the n input images. In this section, we provide a basic overview of the steps necessary to produce the following outputs required for this work:

- Intrinsic matrix K with the assumption that all the n images were taken by the same camera and therefore share the same intrinsics. K can be obtained using camera calibration or estimated during the 3D reconstruction.
- Camera extrinsics for every image $\{(R_i, \mathbf{T}_i)\}_{i=1}^n$ as defined in Section 2.3.3.
- A dense 3D point cloud, which is in its most basic form simply a set of p colored points $\{\mathcal{X}_i \,|\, \mathcal{X}_i := (X_i, Y_i, Z_i, r_i, g_i, b_i)\}_{i=1}^p$, where $(X_i, Y_i, Z_i) \in \mathbb{R}^3$ are the Euclidean coordinates of point $\mathcal{X}_i$ in the object coordinate system and $(r_i, g_i, b_i) \in \{0, \dots, 255\}^3$ are the red, green, and blue color components of the point.

2.3.5.1 (Incremental) Structure from Motion

Structure from Motion (SfM) is the initial stage of the reconstruction process with the goal to estimate the camera extrinsics $\{(R_i, \mathbf{T}_i)\}_{i=1}^n$ and intrinsics K and provide an initial sparse 3D point cloud. There are two main approaches to structure from motion, global SfM, which works with the set of all images at once, and incremental, which adds the images to the reconstruction one by one. In this work, we focus only on the most common incremental version. The algorithm consists of multiple phases, most importantly [24]:

Feature extraction phase attempts to identify a set of distinctive key points in every image, using a feature detector, most often SIFT (Scale-Invariant Feature Transform) [25]. For each key point, it then computes its unique descriptor.

Feature Matching phase matches the key points extracted in the previous steps between images using their descriptors. It generates pairs of images that observe the same 3D points.

Pose estimation and incremental reconstruction consist of several steps using the information about overlapping images from before. During this phase, the images are incrementally added to the reconstruction. Once a new image is registered, its rotation and translation (extrinsics) in an object coordinate system are estimated using the corresponding 2D key points and already reconstructed 3D points. Then new 3D points are reconstructed using triangulation, and a following bundle adjustment step minimizes the accumulation of errors.

At the end of this stage, we get a reconstruction set of camera shots with estimated extrinsics and a sparse 3D point cloud. The intrinsic parameters can also be estimated during the reconstruction or provided by the user if the camera has been calibrated.

2.3.5.2 Dense Point Cloud Reconstruction

After SfM, we can use the estimated camera extrinsics, intrinsics, and the sparse 3D point cloud to build a dense 3D reconstruction. This stage typically uses a method called Multi-View Stereo (MVS) [26]. The detailed description of this process is out of scope of this work. This stage is usually computationally expensive, and the output is a dense 3D point cloud.

2.3.5.3 Other stages

Throughout this work, we aim to create a 3D reconstruction of the power lines and the surrounding terrain to evaluate the distance between an exposed wire and vegetation. For this purpose, a dense 3D point cloud of the terrain is sufficient. However, in other applications of photogrammetry, more sophisticated outputs are required. We briefly summarize the stages typically following dense reconstruction for completeness:

Meshing computes a 3D mesh consisting of polygons.

Texturing provides a color texture for the 3D mesh.

Orthophoto generation leverages the information about camera poses and provides a geometrically rectified photograph composed from the input images, such that it can be used in digital mapping to measure distances.

Generation of various elevation models computes a Digital Surface Model (DSM) or a Digital Elevation Model (DEM), which are later used in many applications ranging from landscape modeling to planetary science [27].

2.4 Machine learning methods

Machine learning is closely related to computer vision and photogrammetry, with significant overlap. In this section, we collect a few methods and tools that are not specific to computer vision or photogrammetry but are more general and applicable to many other tasks that obtain knowledge from data.

2.4.1 Robust Estimation using Random Sample Consensus

Random sample consensus, or RANSAC in short, is a robust model estimation algorithm in the context of robust statistics. It is a generic method and works with any model that we would normally use with the input data, for example, a 2D line for 2D data points. The technique works iteratively and, in each round, selects a subset of the input data points and fits the model of choice. Then it classifies all the input points to *inliers* and *outliers* based on their distance to the model. This is repeated for a fixed number of iterations. Finally, the model with the largest amount of inliers is retrieved [8, p. 410–414].

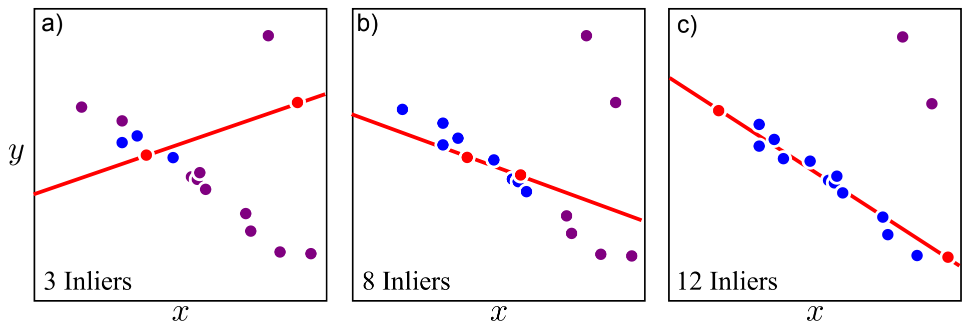


Figure 2.10 An example of random sample consensus (RANSAC) algorithm used for 2D line estimation. The technique works by a) selecting a subset of input points and fitting the model (red points). Then, distance to the model is calculated, and points are labeled as *inliers* (blue points) or *outliers* (purple points). This is repeated with different subsets of points b,c). The final model is then re-estimated from the best fit, using only the inliers [8, p. 411].

Compared to regular model estimation, RANSAC was designed to be much less sensitive to outliers, thus providing better results in noisy datasets. The algorithm is outlined in Figure 2.10.

RANSAC is used extensively throughout photogrammetry and computer vision because the image data are inherently filled with outliers due to electronic noise and various lighting conditions. In this work, we use RANSAC for merging Hough lines and catenary curve fitting.

2.4.2 Hierarchical Clustering

Clustering is an unsupervised machine learning task that aims to split input data into meaningful clusters of related data points.

Hierarchical clustering algorithm solves this task by recursively merging or splitting the input data, depending, whether the approach is „bottom-up“ or „top-down“. The agglomerative hierarchical clustering represents the „bottom-up“ way, starting with individual data points and merging them using one of the available linkage criteria. Common linkage criteria for two clusters include [28]:

Minimum or Single linkage computes the distance of two closest points for a pair of clusters.

Maximum or Complete linkage computes the distance of two farthest points for a pair of clusters.

Average linkage computes the average distance of all pairs of points for a pair of clusters.

Ward linkage is a special criterion that minimizes the sum of squared differences within all clusters.

Hierarchical clustering is often visualized in a dendrogram, as illustrated in Figure 2.11.

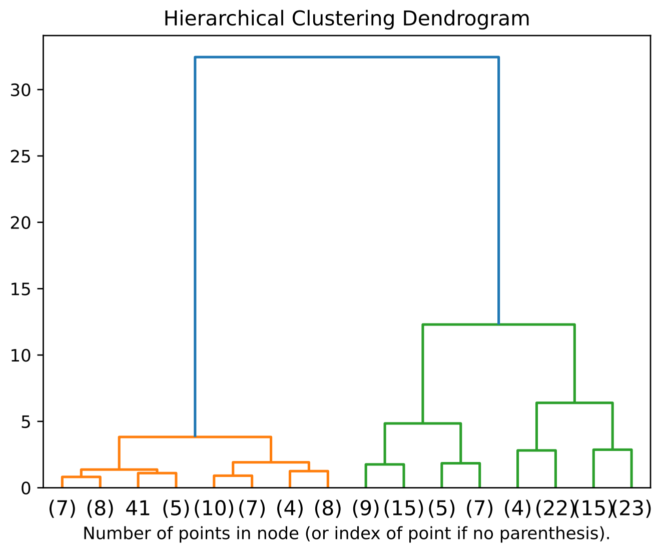


Figure 2.11 Dendrogram after applying agglomerative hierarchical clustering on the iris flower dataset, where each point comprises septal and petal length and width in centimeters [28, 29].

Chapter 3

Related Work

This chapter aims to provide thorough research of power line vegetation management approaches. It begins by exploring the traditional surveying methods, including land-based mapping and satellite imagery, and then focuses in depth on aerial mapping, especially with UAVs.

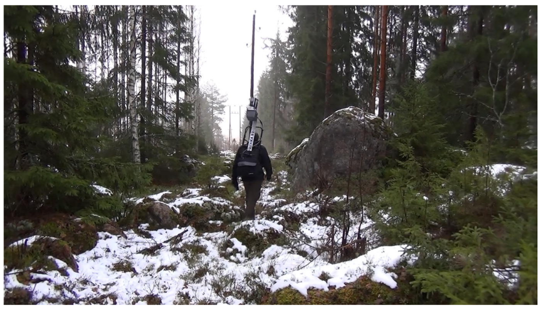
3.1 Traditional Approaches to Vegetation Management

For safe operation, the power line corridors must be inspected and cleared on a regular basis. The traditional surveying methods include land-based mapping, where a person walks through the corridor and scans the area or manually checks for issues. This is very laborious and time-consuming, especially in difficult terrain. For this reason, numerous remote sensing methods are often employed, ranging from satellite imagery to helicopter flights. The satellite images are usually not suitable for detailed measurements due to their low resolution while using a helicopter is very costly [30].

3.1.1 Land-based Mapping

In some cases, it is possible to perform a detailed, high-resolution mapping of the power line corridor using land-based laser scanning. These devices can be attached to either an all-terrain vehicle (ATV) or a backpack (Figure 3.1a). Hence, the methods are named Mobile Laser Scanning (MLS) and Personal Laser Scanning (PLS), respectively. A static approach with a non-moving laser scanner is called Terrestrial Laser Scanning (TLS). The accuracy of these methods highly depends on the precision of the positioning system, so it is affected by satellite visibility. They can achieve even sub-centimeter accuracy in the right conditions and with high-quality equipment [30].

The authors of [31] explore both the use of an ATV and PLS (Figure 3.1b). While providing great results, these methods have some clear disadvantages, namely the difficult inspection of rough or inaccessible terrain, the cost of high-quality equipment, or the dependence on precise positioning and satellite visibility. Another limitation might be that laser scanning directly produces a 3D point cloud, meaning the power lines have to be extracted from the 3D points. The point cloud is an irregular, unstructured set of points with a varying density and no particular order, making the reconstruction more difficult. In photogrammetry, on the other hand, we have the original images, so it is possible to apply a direct power line detection through computer vision and a subsequent geometric approach for reconstruction.


(a) A person scanning a power line corridor with a PLS backpack [30].

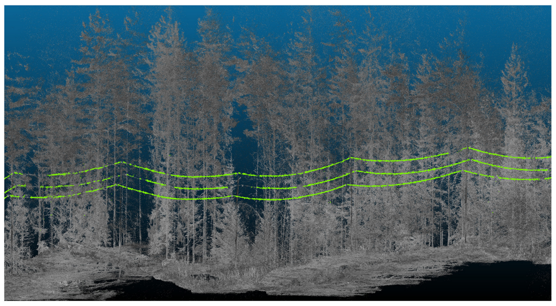

(b) Reconstructed power lines from land-based laser scanning [31]

Figure 3.1 Land-based approach to power line inspection.

While cameras can also be attached to an ATV or backpack [30], we argue that the UAV is a better, more versatile platform for power line corridor inspections using photogrammetry, especially in difficult terrain.

3.1.2 Satellite Imagery

Based on [30], the images obtained by satellites can be divided into two categories:

Synthetic Aperture Radar (SAR) images are taken by reflecting microwave radiation of the earth's surface. The advantage is that this sensor can penetrate clouds and is insensitive to light, which means images can be taken under any weather conditions at any time of the day. The maximum possible resolution is around 1 m per pixel, but the results are distorted by noise inherent to the sensor. Another disadvantage is the high cost of acquiring data frequently. This method was used in [32] for disaster monitoring around power lines.

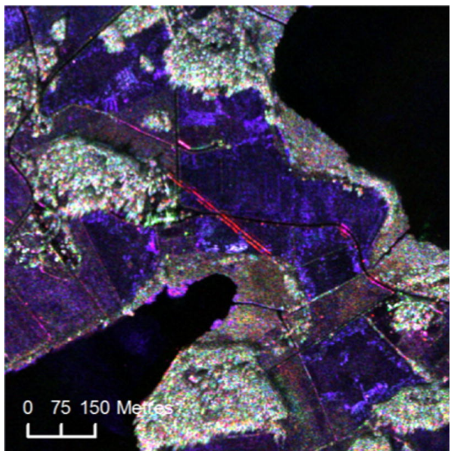
Optical satellite images are taken with a regular or near-infrared camera and therefore are sensitive to weather and time of day. The highest possible resolution is around 0.46 m (GeoEye-1 satellite).

Overall, using satellite is cost-inefficient and, due to limited resources, also less available. Furthermore, the practical resolution is too low for measuring distances between vegetation and power lines, which are only a few centimeters thick. This method is useful mostly for a basic overview of the desired area and rough estimates rather than detailed measurements. Figure 3.2 shows examples of SAR and optical satellite images [30].

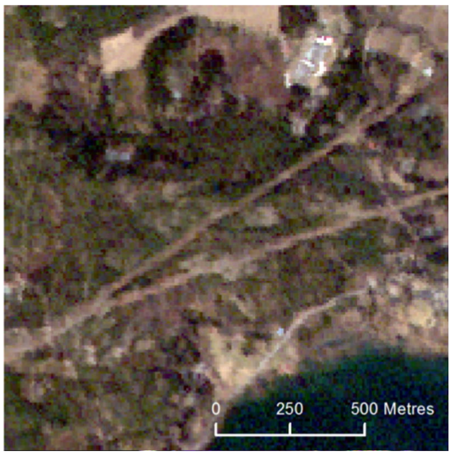
3.1.3 Aerial Mapping

So far, we have covered two approaches on the opposite sides of the spectrum if viewed by distance to the ground. The furthest possible, satellite imagery, has an insufficient resolution, while the land-based laser scanning is limited to accessible terrain. Naturally, one may ask whether the middle of the spectrum – the airspace, would not be more suitable for power line corridor inspection.

As expected, the use of fixed-wing airplanes and helicopters has been thoroughly explored by the industry and is being routinely utilized. For a traditional manned airplane, the resolution is limited up to 5 cm due to the restricted range of achievable altitudes. As we explain in Section 4.2.1, this is not sufficient for a detailed power line corridor inspection using images. For the same reason, the point density in laser-based approaches is also low in these high altitudes (dozens of points per m^2) [30].



(a) SAR satellite image.



(b) Optical satellite image.

Figure 3.2 Example SAR and optical satellite images [30].

Helicopters, on the other hand, can fly much closer and even hover in place for detailed inspection. They can achieve a sufficient sub-centimeter accuracy from altitudes as high as 100m, but they can fly even lower [33]. Helicopters often perform multiple types of scanning, optical – using a camera (possibly in multiple specters, such as near-infrared), and laser scanning – using a LiDAR (Light Detection and Ranging) laser-based sensor. The major disadvantage of using manned aircraft is the cost associated with the equipment and the technical skills required for such surveys [30].

The high cost and time demands of traditional methods are pushing the industry towards new, innovative approaches, such as using Unmanned Aerial Vehicles (UAVs). These include fixed-wing and multi-rotor drones with two prevailing types of payloads, a camera, and LiDAR. UAVs are also a step towards highly automated workflows since most data acquisition and processing can be fully completed with no human intervention. This approach is the focus of our work, so we discuss the research on this topic in a separate section below.

3.2 Vegetation management using UAV

Due to the increasing availability of relatively inexpensive UAV platforms in recent years, many researchers and companies are discovering their potential and deploying them for numerous different tasks with great success. UAVs have been immensely helpful in agriculture, geography, cultural heritage, archaeology, mining, gaming industry, and (sometimes infamously) military. In power line vegetation management, a lot of research on the use of UAVs has been done, too. This is especially due to the UAV's unique ability to fly close to the power lines, the improvements in the battery endurance, and the emergence of lightweight sensors [30].

3.2.1 UAV with LiDAR

The prevailing sensor in literature used for collecting data has been an ordinary RGB camera. While some studies [34, 35] and commercial solutions [36] use UAVs with the LiDAR sensor and provide high-quality results, the equipment is relatively costly, and the data acquisition is highly sensitive to errors, often requiring repeated flights [37]. Therefore, the research community mostly opted for the methods of computer vision and photogrammetry to measure and reconstruct the 3D structure of the wires from multiple photographs.

As a result, we focus on photogrammetric methods, and we will not further discuss UAVs with the LiDAR sensor in this work.

3.2.2 UAV with Camera

Photogrammetry has been a standard tool for reconstructing 3D objects from photographs for many years now, and in many applications, it works flawlessly. The critical part of the photogrammetric 3D reconstruction is the Structure from Motion (SfM) algorithm, which detects significant points in images (such as corners) and matches them across several images to calculate the 3D geometry. However, in our case, the power lines are too thin and uniform to find distinct points on them accurately. As a result, the standard methods are good at reconstructing the vegetation and surroundings, but not the power lines themselves.

3.2.2.1 Reconstructing Power Lines from Point Cloud

Nevertheless, in 2015, the authors of [38] tried to model the power lines directly from the dense point cloud obtained by the standard methods, using the catenary curve model. Their paper concludes that it indeed is feasible if the appropriate conditions on data capture are met. Still, the point cloud they are using for power line modeling is missing significant portions of the power line, sometimes even more than half of the wire in each span. Consequently, their algorithm is filling many gaps in the data, which might yield high errors and uncertainty.

3.2.2.2 Reconstructing Power Lines from Images

This is why nowadays, state-of-the-art methods are using specialized algorithms for first detecting and then reconstructing the power line geometry. In fact, many studies focus exclusively on the former – segmenting the power lines in images [39, 40, 41, 42]. These use advanced computer vision and deep learning techniques to accurately detect the pixel location of wires in the images. Other studies then build on this knowledge and use geometric models to reconstruct the 3D location of those pixels while exploiting some specifics of this task, such as the catenary curve.

Early works attempting to solve the entire problem of vegetation management in the power line corridor include [43], where the authors detect the occluding objects in near real-time and also provide an offline report containing 3D information. However, the study fails to provide sufficient experimental data and evaluation, so it is unclear how the algorithm performs in varying conditions and background complexity.

The more recent article [44] proposes a relatively straightforward method based on first detecting the 2D power line vector in images, selecting images corresponding to the same patch of the corridor, and using epipolar geometry to reconstruct the 3D geometry. Their method yields good results, yet there are several possible improvements. Firstly, they strictly require two flight strips along the corridor to form a pair of left and right images. However, one could argue that capturing more strips, possibly from an oblique perspective, could result in better reconstruction. Further, during the 3D reconstruction, they require manual intervention to correctly assign the corresponding power lines in the two images making the process only semi-automatic. Moreover, they are not exploiting the catenary nature of the wires. Additionally, in the 2D power line detection phase, they use operations based on the grey value profiles, which may make it vulnerable to varying lighting conditions and background.

The work [45] again uses line-based 2D and 3D reconstruction methods. Their main contribution is using sophisticated convolutional and recurrent neural networks to classify image pixels into eight classes. They use this information to filter the dense point cloud as well as the reconstructed power lines. Then they fuse all the information to obtain a clean dense reconstruction and even semantic labels of the detected objects on top of that. To obtain the 3D points of the power lines, they use an existing 3D line reconstruction algorithm [46]. However, they do not capture the individual instances of the power line, only reconstruct all of them as a group. This makes it impossible to filter out the noise and improve the quality by fitting a catenary curve.

Another study [47] proposes an interesting approach that goes against the aforementioned practices, that is, going from the 3D space back into the 2D image space. They generate a discrete dense 3D grid along the corridor and back-project each point into the images. If the point aligns with the segmented power lines in multiple images, it is kept. Otherwise, it is filtered out. Clearly, this is limited by the resolution of the generated 3D grid, which they improve by interpolating the points with a parabolic curve. This image-voting-based approach cleverly avoids the need for matching the corresponding wires in the images, where [44] requires manual intervention. The authors do not mention the processing time, which might be a disadvantage of this approach since projecting every point of a dense grid onto many images seems computationally intensive.

Lastly, the most recent work [48] improves on the previous by proposing a new technique for segmenting power lines in images using modified traditional computer vision operations requiring a minor manual intervention. Although they are proud of not using neural networks, it might be worth comparing with deep learning methods, especially when they report exceptional accuracy [42]. Moreover, they rely on the precise ordering of the images by acquisition time and assigning them to flight strips, which seems unwieldy and lays unnecessary constraints on the acquisition process. This can be automated by using the camera orientation and position obtained during the photogrammetric processing.

Chapter 4

Data Acquisition

Because the power lines are very thin structures, the accuracy of the 3D reconstruction highly depends on the input data quality. We followed a set of photogrammetry best practices together with insights from related research articles [18, 48] to form precise guidelines for image capture specific to power line detection. The guidelines are separated into two sections – camera settings and flight planning.

4.1 Camera Settings

Since we are using images as a primary source of data, the basic principles of photography apply. The detailed explanations of those are out of scope of this work. Nonetheless, a few essential parameters and settings related to photogrammetry from [18] are summarized below.

4.1.1 Exposure

The overall exposure is determined by three main settings, which together form a so-called *exposure triangle* – ISO, aperture size, and shutter speed. Figure 4.1 indicates that all these settings must be configured together to capture a well-exposed photograph.

Aperture describes the physical size of the lens opening. It is indicated as an *f-number*, where the higher number means a smaller opening. The smaller the aperture, the closer we are to an ideal *pinhole camera*, where the aperture is a single point. A small aperture results in a sharper image, but setting the aperture too small in the real world will cause diffraction – a kind of optical defect that causes losing the sharpness again. For missions with flight heights over 20 m, setting the aperture to medium settings ($f/5.6$ to $f/11$) is recommended [18].

Shutter speed determines the time during which the sensor is exposed. Larger values allow more light to be captured at the cost of motion blur. For photogrammetry, the authors of [18] recommend shutter speeds from 1/250 to 1/1000 s. Later we will describe how this affects the flight velocity.

ISO represents the camera sensor sensitivity or signal gain. Increasing the ISO leads to brighter images at the cost of more noise. The ISO should be set to the minimum possible value, usually around 100-800 in daylight conditions.

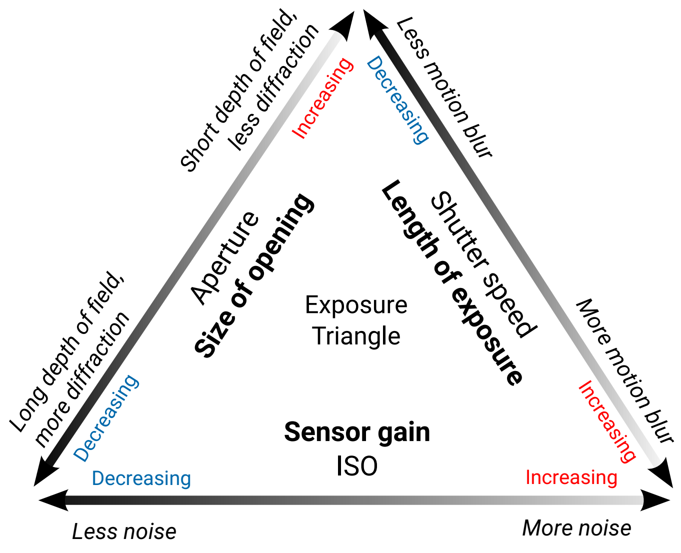


Figure 4.1 The exposure triangle, indicating that the three crucial camera settings all depend on each other and must be configured together [18].

4.1.2 Focal Length

The lenses in consumer cameras usually come in two types: fixed (prime lenses) or adjustable focal length (zoom lenses). In photogrammetry and computer vision in general, the fixed focal length is typically advised as there are no moving parts impairing the optics precision.

By combining the focal length and the physical size of the camera sensor, one can calculate the *effective* focal length, which determines the camera's field of view. That means that the lens should be chosen based on the type of UAV and optimal flight height. For example, if a fixed-wing UAV is used, the flight height must usually be higher, needing a larger effective focal length to capture detailed information.

4.1.3 Camera Calibration

Lastly, for taking reliable measurements from images, it is necessary to know the camera intrinsics, i.e., the matrix K and the distortion coefficients (see Section 2.3.3). Traditionally, these are obtained in a process called camera calibration. The most popular technique for camera calibration is Zhang's method [49], which entails taking multiple pictures of a checkerboard pattern from different angles and using a clever algorithm to estimate the intrinsics. However, in photogrammetry, it is possible to skip this step entirely and perform a so-called *self-calibration* during the bundle adjustment step [17, p. 674–686]. This approach estimates the camera intrinsics as part of the optimization process. Although camera calibration using the dedicated Zhang's method should be more precise in theory, we observed no improvement in the reconstruction quality in our experiments. Hence, we do not require a separate camera calibration step in our proposed pipeline.

4.2 Flight Planning

The critical parameters for a successful mission comprise ground sample distance (GSD), flight height, velocity, image overlap, and acquisition rate.

4.2.1 Ground Sample Distance (GSD)

The ground sample distance represents the level of detail captured in the image. It is defined as a ground distance in millimeters visible in one pixel of the image. The optimal value depends on the smallest feature we need to detect, which in our case is the power line wire. As a consequence, the GSD should be set independently for each survey depending on the actual wire diameter. To simplify the matter, we will only consider power lines over 45 kV, where the smallest possible diameter according to [50] is 15.5 mm. The authors of [48] recommend the GSD to be no higher than half the diameter of the wire leading to a GSD of 7.8 mm.

It might be tempting to choose a smaller GSD to achieve an even greater level of detail, but that is not without its trade-offs. A smaller GSD will result in a lower flight height, which in turn leads to more flight strips and a higher acquisition rate to ensure a sufficient overlap. Thus, it increases the overall flight time, data size, and processing time.

4.2.2 Flight Height

Once the GSD is set, we can compute the maximum flight height. The authors of [18] formulate the relationship between GSD and flight height as:

$$GSD = \frac{2H \cdot \arctan(S_{det})}{2f} \approx \frac{H \cdot S_{det}}{f}, \tag{4.1}$$

where H is flight height, S_{det} is the pixel pitch (the physical width of a pixel on the sensor) and f is the effective focal length.

One can notice that the measured wires are usually several meters high. Therefore, we must only ensure the GSD from the previous step at the height of the lowest wire. The power line sag and uneven terrain should be taken into account, so let H_0 be the height of the lowest wire at the lowest point of the survey and GSD_{H_0} the required sample distance at H_0. The final height can then be calculated as

$$H = H_0 + \frac{GSD_{H_0} \cdot f}{\arctan(S_{det})} \approx H_0 + \frac{GSD_{H_0} \cdot f}{S_{det}}.$$

This is then used to calculate the *true* ground sample distance using Equation 4.1.

It is recommended to use automatic flight planning tools that can follow terrain elevation using a Digital Elevation Model (DEM) to make sure that the correct flight height is maintained throughout the survey when the terrain is not perfectly flat.

4.2.3 Flight Velocity

The flight velocity mostly affects the overall flying time and thus the battery consumption of the planned mission, meaning one could fly longer missions on a single charge. In the case of power line inspection, this is crucial because the goal is to survey hundreds of kilometers of power lines fast and regularly.

To optimize the velocity, we will again follow [18] and consider the major limiting factor – motion blur b (in pixels), which is estimated by:

$$b = \frac{v \cdot t}{GSD_{H_{max}}},$$

where v is the velocity and t is the shutter speed. To be absolutely precise and acquire sharp images for all wires, $GSD_{H_{max}}$ should be the sample distance at the height of the highest wire (H_{max}) calculated by substituting H_{max} into Equation 4.1. The paper [18] recommends a maximum motion blur of 1.5 pixels.

4.2.4 Image Overlap

During the processing phase, the photogrammetry software requires a significant overlap between images to identify common points and calculate their 3D position based on multiple views. In the existing literature [38, 44, 48], the authors are using a side overlap anywhere from 70 to 80 % and a front overlap between 50 and 87 %.

Various software tools provide certain recommendations. For example, Agisoft Metashape recommends a side overlap of 60 % and a front overlap of 80 %. OpenDroneMap's documentation advises similar values in a practical example [51], where they additionally recommend increasing the overlap when flying higher.

To keep things simple and independent of flight height, we generally recommend both overlaps to be at least 80 %. Although it may require more storage and processing time, the higher data redundancy will improve the quality and precision of the reconstruction. In our experiments, we used an 85 % front overlap and a side overlap of 90 %, although the latter is mainly controlled by other requirements, such as sufficient GSD or the need for three flight strips with all wires visible in the image.

4.2.5 Acquisition Rate

Usually, there are two options for automatic image capturing during the flight – by time or by distance. Both options are equally valid, but we recommend using the latter because it might be difficult to maintain a constant speed during the flight. The acquisition rate can be easily calculated using the desired overlap and the physical area captured on the image.

For example, if we have a camera with a resolution 7952×5304 px and a GSD of 1 cm, the area captured on the image is 79.52×53.04 m. Assuming the shorter side of the image is aligned with the UAV direction, and the desired front overlap is 80 %, the UAV should move no more than 10.6 m between images. For the trigger based on time, one can simply divide this number by the flight velocity.

It should be noted that every camera has a maximum acquisition rate, e.g., the Sony Alpha 7R II can capture at most five images per second. If the acquisition rate of the camera is not sufficient for the desired overlap, we need to reduce the velocity, although modern cameras usually have a high enough acquisition rate, and the velocity is rather limited by the motion blur.

4.2.6 Requirements Specific to Our Solution

For fully automatic reconstruction, we additionally require three other constraints:

Three parallel flight strips provide more data redundancy and increase the robustness of the automatic power line matching, we will later propose, compared to two flight strips required by the authors of [44, 48]. As a result, three parallel images will cover every place along the power lines, which will allow our solution to sustain overlapping wires in one of them. We should take this into consideration during flight planning and make sure that the power lines are clearly separated in at least two out of the three images at all times.

Visibility of all power lines in each flight strip is also necessary for our automatic power line matching. This means that the distance between flight strips controlled by side overlap should be adjusted to fit all wires in the image. In our experiments, we used a side overlap of 90 % to satisfy this requirement.

Power lines should appear vertically in the images. We can leverage this to filter any horizontal lines, such as trees or transmission towers.

These requirements are easy to fulfill in practice and do not impede the quality of the photogrammetric reconstruction, as we will later prove in our experiments. Additionally, all images should be orthogonal to the ground, i.e., „nadir“, with no oblique images necessary.

4.2.7 Georeferencing

Georeferencing means associating images with their position in the real world. The object coordinate system is then characterized in some geographic coordinates, for example, GPS (Global Positioning System) or WGS84 (World Geodetic System 1984). There are two main approaches to georeferencing [52]:

Ground Control Points (GCPs) are points for which we know their precise geographic locations. Usually, they are measured over several hours and labeled with a checkerboard pattern, so they are clearly visible in the images. Obtaining GCPs is an arduous process that is especially hard in difficult terrain.

Direct positioning uses satellites to measure the position of each image. There are two prevailing methods, RTK (Real-Time Kinematic) and PPK (Post-Processed Kinematic), both need an additional base station. This approach does not require any prior measurements and is ideal for rough or inaccessible terrain.

Our proposed method supports both approaches. In our experiments, we use the direct RTK for its convenience, as it requires no additional measurements or processing. The positions are written directly into the image metadata during the survey. The direct positioning is also more suitable for surveys in difficult terrain, which is often the case around power lines. Generally, PPK might be more accurate in situations with occlusions between the UAV and the base station. However, we have not observed any loss of RTK signal during our experiments. A detailed comparison of the different approaches can be found in [52].

Chapter 5

Analysis and Design

This chapter provides an in-depth explanation of all the steps we propose to transform the input UAV images to a 3D visualization of the site with highlighted vegetation encroachment. We start by giving an overview of the architecture of our solution, followed by a detailed analysis and design of the individual components.

5.1 Reconstruction Pipeline Architecture

Section 3.2.2 outlined two prevailing approaches for 3D power line reconstruction in the existing literature:

From 2D images to 3D points using geometric triangulation – by applying epipolar geometry to overlapping images, we can precisely triangulate 3D points. This approach requires knowing the camera extrinsics and intrinsics and pairs of corresponding power line pixels for a given pair of overlapping images. The works [44, 48] implement this approach and use image segmentation and power line detection, followed by geometric triangulation.

However, the authors of both papers claim that obtaining pairs of corresponding wires is difficult and apply manual work to solve this problem.

From 3D points to 2D images using reprojection and voting – the authors of [45] and [47] circumvent the issue by going in the opposite direction. They first obtain candidate 3D points, either by an existing straight-line extractor, in case of [45] or by simply dividing the 3D volume into a cubic grid and treating the center of each cube as a candidate point [47]. Then they reproject the candidate 3D points into the images using the camera extrinsics and intrinsics. For deciding whether a particular point belongs to a power line or not, multiple images vote based on their segmentation masks.

This method loses the information about individual instances of the power lines, which prevents fitting the catenary curve without complicated 3D point clustering. Also, we argue that the time complexity is higher compared to the first approach. Extracting straight lines using the algorithm from [46] is computationally comparable to extracting power lines geometrically but is merely a subtask of [45]. In [47], the computational time directly depends on the resolution of the 3D cubic grid. Obtaining results with comparable accuracy to the first method would mean projecting every cm^3 of the scene to multiple images.

We conclude that the first method is more efficient and straightforward, allowing for better precision, distinguishing the individual power line instances, and subsequent catenary curve fitting.

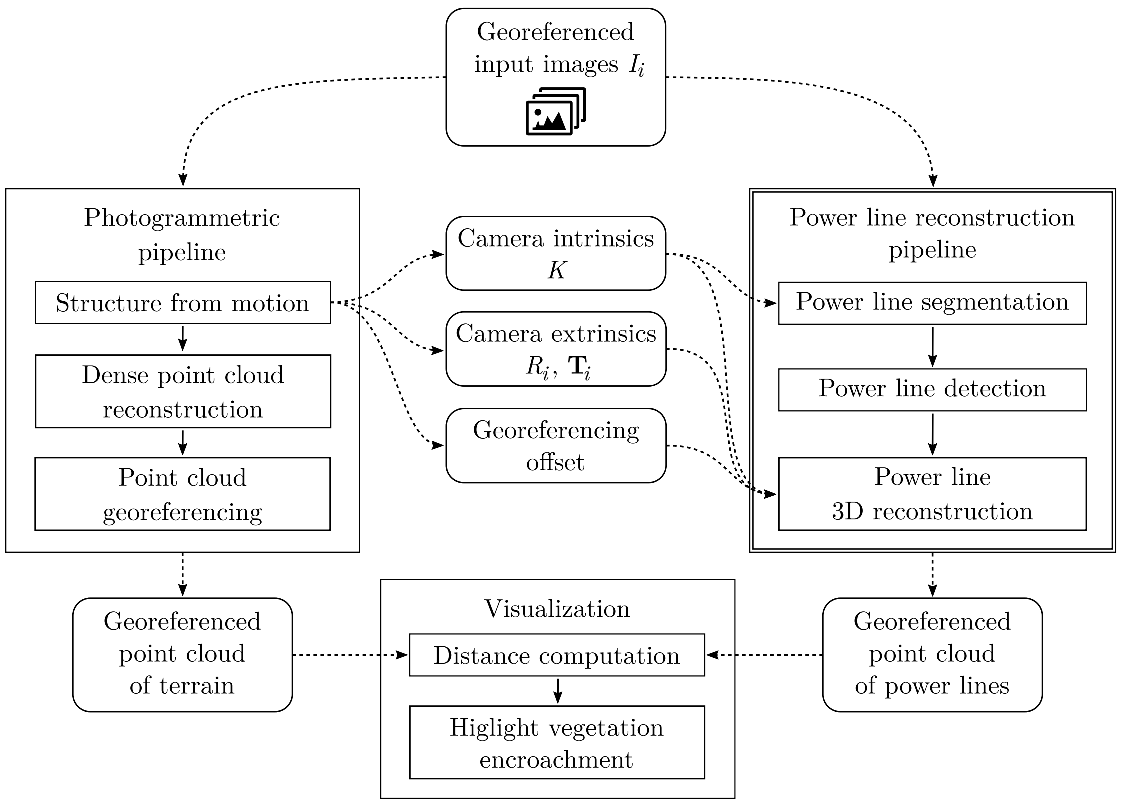

Georeferenced input images I_i
Photogrammetric pipeline
Structure from motion
Dense point cloud reconstruction
Point cloud georeferencing
Camera intrinsics K
Camera extrinsics R_i, $\mathbf{T}_i$
Georeferencing offset
Power line reconstruction pipeline
Power line segmentation
Power line detection
Power line 3D reconstruction
Georeferenced point cloud of terrain
Visualization
Distance computation
Higlight vegetation encroachment
Georeferenced point cloud of power lines

Figure 5.1 Architecture of our solution. The algorithm consists of three main components, a standard photogrammetric pipeline, a power line reconstruction pipeline, and a visualization module. In this work, we focus primarily on the power line reconstruction pipeline and use third-party software for the other components.

As a result, we use it as the basis for our solution. We mitigate the need for manual intervention by introducing constraints on the data acquisition process (Section 4.2.6) and matching the corresponding power lines automatically. The proposed constraints do not limit the applications or feasibility, as we later prove by performing experiments in three different locations.

Main Components

The proposed solution consists of three main components. We compute the point clouds of the terrain and power lines separately in the photogrammetric and power line reconstruction pipelines. Afterward, we fuse them and calculate the distances between the power lines and terrain in a visualization tool. Figure 5.1 gives a high-level overview of the reconstruction pipeline architecture:

Georeferenced input images are n RGB UAV images $\{I_i\}_{i=1}^{n}$, as in Definition 2.1. For simplicity, we assume that the images are accompanied by positioning information (latitude, longitude, elevation) stored in the EXIF metadata of the image. Although, georeferencing using Ground Control Points (GCP) is also possible.

Photogrammetric pipeline takes the input images and produces several outputs, mainly the camera intrinsics K and extrinsics $\{(R_i, \mathbf{T}_i)\}_{i=1}^{n}$, required for the power line 3D reconstruction and a point cloud representation of the terrain. This is a part of the typical photogrammetric workflow described in Section 2.3.5. Therefore, we use one of the many photogrammetry software tools available. The options are carefully assessed in Section 5.2.

Power line reconstruction pipeline is responsible for detecting the power lines in the images and obtaining their 3D representation through epipolar geometry. The result is then enhanced by catenary curve fitting. This component depends on the camera extrinsics, intrinsics, and georeferencing information, obtained by the photogrammetric pipeline. The component is further split into three separate steps: power line detection, power line segmentation, and power line 3D reconstruction, which will be explained in detail as part of this chapter.

Visualization is performed by a third-party tool. It is an interactive application with a Graphical User Interface (GUI), where we import the two point clouds and calculate the distance between them. Then we are able to define a color scale in meters and highlight the vegetation encroachment. This is the only part of the solution that is not automated since the interactive approach is much more configurable and enables additional postprocessing of the point clouds.

It is important to say that even with high-quality equipment, it is impossible to reconstruct the power lines using only the traditional photogrammetric pipeline since it was not designed to reconstruct thin homogeneous objects. In our experiments, we were able to recreate a few points at best (we estimate less than 10 % in some cases) using the photogrammetric pipeline only. Moreover, the point cloud would have to be accurately classified into power-line and non–power-line points to effectively measure distances. This is why we reconstruct the power lines separately in the power line reconstruction pipeline we propose.

The architecture described above allows for a fully automated workflow up to the visualization part. The rest of this chapter presents the analysis and design of the aforementioned components in more detail.

5.2 Photogrammetric Pipeline

Building photogrammetric models is a common use case in many industries, which created a demand for high-quality automated software. Consequently, there are many tools on the market, including both free and paid options. This section aims to analyze and select a third-party photogrammetric software for estimating the camera intrinsics and extrinsics and reconstructing a georeferenced 3D point cloud of the terrain.

5.2.1 Photogrammetric software selection

First, we select a few candidates, which we then score based on several criteria and requirements specific to this work. The choice is mainly limited by the cost, as well as automation options and the ability to run in a remote Linux environment.

We consider only software tools that allow for remote processing (using either a Python API or a command-line interface) and provide all required exports. The selected candidates are the three most popular paid options:

- RealityCapture [53],
- Agisoft Metashape [54],
- PIX4Dmapper [55],

accompanied by free, open-source tools:

- Meshroom [56],
- OpenDroneMap [57],
- MicMac [58].

We do not have the resources to evaluate every option in this work experimentally, so we rely on the existing literature and official documentation.

5.2.2 Methodology

We assign a subjective weight (importance) to each category, according to our use case, and give every candidate a negative (−1), neutral (0), or positive (1) score. The final score is determined as a weighted average of the individual scores.

The list of evaluation criteria (with the assigned weight in brackets):

Quality of reconstruction (2) – the density and accuracy of the 3D point cloud, subjectively rated based on the available literature [59, 60, 61, 62, 63].

Computational time (1) – the overall processing time of structure from motion and dense point cloud reconstruction, subjectively rated based on the available literature [59, 60, 61, 62, 63].

Price (2) – for-profit licensing, excluding educational and trial offers. If there are multiple tiers available, we select the most sensible one for our work that allows for remote processing:

- more than \$3000 perpetual license or more than \$1500 for a one year subscription (−1),
- less expensive (0),
- free (+1).

Ease of use (2) – a subjective evaluation of the usability of the software based on the output formats and the amount of extra work required for setting up remote processing on a private server. We are not interested in paid proprietary cloud solutions that some tools offer, as the Faculty of Information Technology, Czech Technical University in Prague, provided powerful Linux-based computational resources for this work. The criteria are:

- difficult to use for our purposes or unnecessarily complicated (−1),
- remote processing requires manual work or additional scripting (0),
- easy to set up remote processing, and download the required exports (+1).

5.2.3 Evaluation

In the literature, RealityCapture and Agisoft Metashape have been observed to produce the highest quality point clouds [59, 60]. However, they both provide only a command-line interface for automation. The same is true for MicMac. OpenDroneMap and Pix4Dmapper, on the other hand, enable easy deployment with remote processing API and automation tools integrated into Python.

The full results of the evaluation are shown in Table 5.1. All things considered, we selected the free, open-source OpenDroneMap software, which comes with a palette of easy-to-use remote processing options while still producing good results. The advantages for our use case include a simple local and remote deployment using docker, a web-based graphical user interface and administration, and a Python library for easy integration into automated pipelines [57]. We then conducted a UAV test flight and 3D terrain reconstruction to validate our choice.

▶ Note 5.1. Agisoft Metashape professional has a built-in proprietary image-based algorithm for power line detection. As far as we know, there is no evaluation of their methods by the research community. They only mention that the algorithm can struggle with other linear objects in the scene, and manual correction may be required [64].

Software	Quality (2)	Time (1)	Price (2)	Ease of use (2)	Overall score
RealityCapture	$+1$	$+1$	0	0	3
Agisoft Metashape	$+1$	-1	-1	0	-1
PIX4Dmapper	0	0	-1	$+1$	0
Meshroom	0	0	$+1$	0	2
OpenDroneMap	0	0	$+1$	$+1$	4
MicMac	0	-1	$+1$	-1	-1

Table 5.1 Summary of photogrammetry software evaluation.

5.3 Power Line Reconstruction Pipeline

This part of the architecture is the core component of our work. The process is split into three subsequent steps: power line segmentation, power line detection, and power line 3D reconstruction. In this section, we present a detailed analysis and design of those steps.

5.3.1 Power Line Segmentation

The first step in the reconstruction process is the power line segmentation. This module aims to create a binary segmentation mask $M_{semantic} \in \{0,1\}^{h,w,1}$, with the value 1 at the pixel locations of power lines and 0 everywhere else. The quality of this mask is crucial since it directly affects the subsequent steps. In this section, we implement two common methods for power line segmentation and discuss the differences. First, we attempt to replicate the algorithm based on a modified Prewitt filter from [48]. Then we address its weaknesses by proposing a new solution using an „off-the-shelf“ pretrained neural network.

5.3.1.1 Input and Output

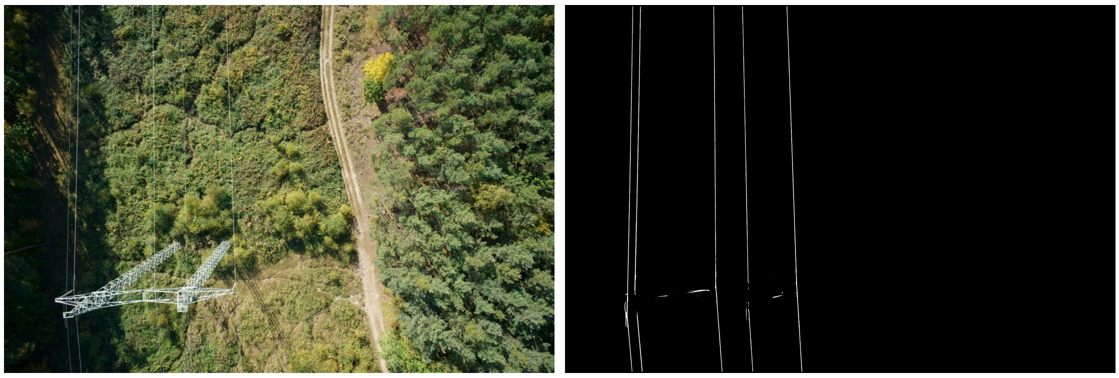
The input for this step consists of the original images without lens distortion. We can get rid of some of the imperfections of the lens if we know the camera intrinsics – the matrix K and the distortion coefficients, as explained in Section 2.3.3. This is called image undistortion. In this work, we leverage the fact that the undistorted images are one of the photogrammetric pipeline outputs, so there is no need for a separate undistortion step.

The output is a binary semantic segmentation mask of the power lines. Figure 5.2 shows an example input and output of this step.

5.3.1.2 Filter-based Segmentation

The traditional methods for semantic segmentation of lines utilize linear filtering (Section 2.2.3). The process typically consists of convolution with a filter kernel for edge detection, followed by a thresholding operation to select the most prominent edges. Optionally, the input can also be preprocessed by various techniques to increase the contrast and highlight the edges. We closely follow [48], which has successfully applied the linear filtering methods on power lines. We obtain the output segmentation masks in three stages:

Decorrelation stretch is a preprocessing technique to enhance the input image colors by „stretching“ the saturation. Here we directly follow [48], which uses a method based on a principal component analysis from [65]. The details of this algorithm are out of scope of this work. The output is illustrated in Figure 5.3b.



(a) Input – undistorted original image. **(b)** Output – binary segmentation mask

Figure 5.2 Input and output of the power line segmentation step.

Edge detection using a modified Prewitt kernel exploits the fact that the power lines are linear structures and can be detected as edges in the image. Here, the authors of [48] modify the traditional Prewitt kernel from Equation 2.4 by scaling and rotating it according to the power line direction. This is where the authors apply manual work since finding the precise angle of the power lines at this stage is difficult. We avoid this issue by assuming that the power lines will always be vertical in the input image. It can be easily achieved during the data acquisition process. Our experiments found that the vertical Prewitt kernel works well even when the power lines are not strictly vertical (with up to ±20 degrees deviation). Therefore, we detect the vertical edges by convolving with the following 31×31 filter kernel:

$$F_{vertical} = \begin{bmatrix} 0 & \dots & 0 & -1 & 0 & 1 & 0 & \dots & 0 \\ 0 & \dots & 0 & -1 & 0 & 1 & 0 & \dots & 0 \\ \vdots & & & & \vdots & & & & \vdots \\ 0 & \dots & 0 & -1 & 0 & 1 & 0 & \dots & 0 \\ 0 & \dots & 0 & -1 & 0 & 1 & 0 & \dots & 0 \end{bmatrix} \in \mathbb{R}^{31,31} \tag{5.1}$$

Figure 5.3c shows the real-valued output image of this operation.

Thresholding selects only the most prominent edges from the previous output. We use the binary thresholding operation from Equation 2.5 and set the threshold as the 99-quantile of the pixels. This means that the pixels are sorted by their value, and only the highest 1 % are kept. The result is a binary segmentation mask illustrated in Figure 5.3d.

5.3.1.3 Neural Network segmentation

Nowadays, neural networks hold a state-of-the-art performance in many computer vision problems. While we remain skeptical and think that many tasks can be solved efficiently with the traditional methods, it is difficult to ignore them, as they were successfully used for power line segmentation in the past (as discussed in Section 3.2.2).

Neural Network Architecture

Instead of inventing a custom neural network architecture or replicating one of the power line segmentation networks used in the literature, we decided to take a different approach. We looked at the state-of-the-art methods for the segmentation task in general and based our solution on one of the best models – DeepLabv3+ from [66].

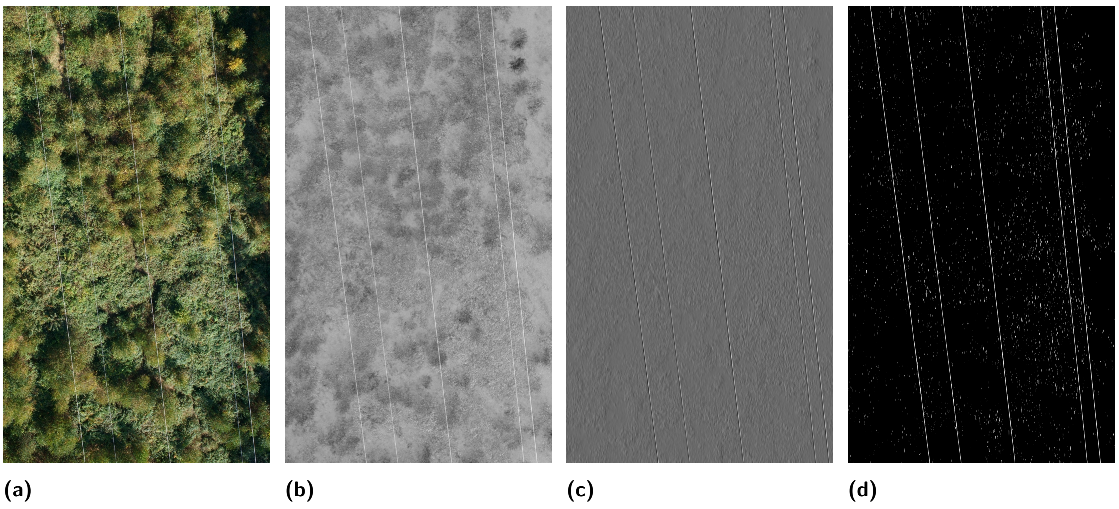


Figure 5.3 Filter-based power line segmentation. The undistorted input image **(a)** is first decorrelated **(b)** and then convolved with a modified Prewitt filter **(c)**. The most prominent edges are obtained by quantile-based thresholding **(d)**.

This has some advantages. Firstly, we found an open-source implementation in a general-purpose semantic segmentation framework *MMSegmentation* [67] that provides a simple configuration for an otherwise complex model. Secondly, there are pretrained weights available, which we can reuse and dramatically reduce the required training time. And thirdly, since the model achieved exceptional results on several complex datasets, we expect it to perform well for power line segmentation too. Figure 5.4 illustrates the model architecture, although a detailed description is out of scope of this work. Instead, we refer the reader to [66].

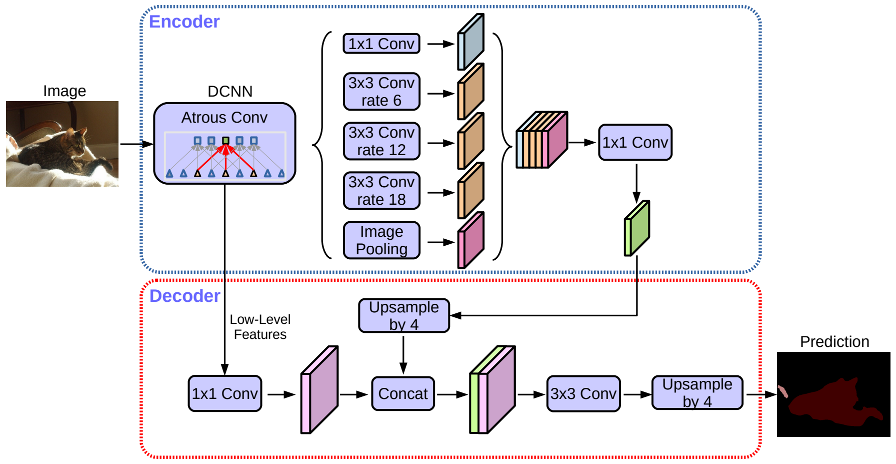
Training Data

Similar to other machine learning models, neural networks have parameters (weights) that must be trained using a training dataset. For segmentation, the dataset usually consists of pairs of images – the RGB input image and the target segmentation mask.

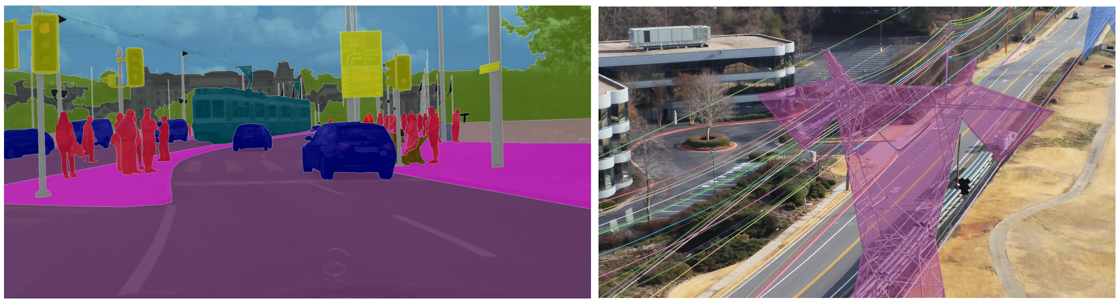
The complex state-of-the-art models, such as the one we selected above, require training over a long period of time on a sufficiently large dataset. As a result, researchers often publish the trained weights, so other people can use them to initialize their models and start training from there. Such a process is called fine-tuning and often results in much faster training with less data required because the model learned to generalize enough to be able to transfer the knowledge between different tasks. As an example, computer vision models typically learn a set of abstract features, such as lines, edges, corners, and other patterns, that are useful in multiple tasks. A model that has been trained to segment street lights with thin grey poles might be easily fine-tuned to segment power lines.

The DeepLabv3+ model described above has been trained on a popular dataset of urban scenes – the Cityscapes dataset [68], consisting of 5000 annotated images with pixel-level segmentation masks. The pretrained weights are publicly available, so we can use them to initialize our model. A sample image from the Cityscapes dataset is shown in Figure 5.5a.

Finally, we still need a power line segmentation dataset to fine-tune the neural network. Luckily, there is a public dataset TTPLA, consisting consists of 1231 images. The segmentation masks contain pixel-level annotations of four classes, including power lines and three types of transmission towers [11]. We are interested only in the power lines, so we removed the other classes when transforming the dataset to the required format for training. A sample image from the TTPLA dataset is shown in Figure 5.5b.



■ **Figure 5.4** DeepLabv3+ model architecture. One can notice the heavy utilization of the convolutional layers (*Conv*) described in 2.2.5 common in neural networks that work with images. A detailed description is out of scope of this work [66].



(a) A sample image from the Cityscapes dataset [68]. **(b)** A sample image from the TTPLA dataset [11].

■ **Figure 5.5** Training data for the neural network segmentation.

Methodology

The standard methodology for training and evaluating models in machine learning is to shuffle the dataset and split it into three parts – training, validation, and testing data.

Having multiple datasets is necessary to avoid a so-called over-fitting, which means that the model adapts too well to the training data, including their noise, and does not generalize to other data. This is why there is a validation dataset that allows observing the generalization performance of a model.

Often, we are not happy with the first result we get from the model, so we try to change it to get better performance. For example, we might alter the learning rate or regulate the model expressive power by changing the number of trainable parameters. We then apply the same framework again – fitting the model to the training data and observing the generalization on the validation data. However, altering the model to get better validation results effectively biases the model towards the validation data. This is the reason why there is a third part – the testing data. These are samples from the dataset the model has never seen. We use the testing data at the end to evaluate and report the final model performance.

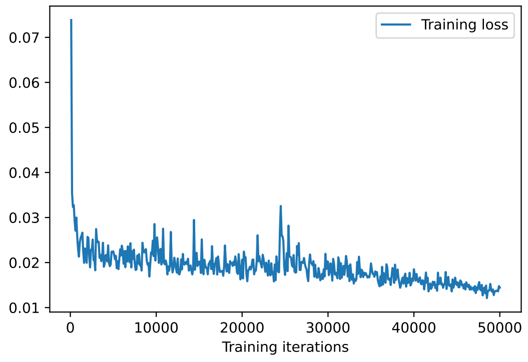


Figure 5.6 Training loss.

Training and Evaluation

We initialized the DeepLabv3+ neural network with the pretrained weights from the Cityscapes dataset and then fine-tuned the model on the TTPLA dataset for 50 000 training iterations. In conformity with the above, we randomly shuffled the dataset and split it into three parts: training data (1025 images), validation data (102 images), and testing data (104 images).

During training, we observed several metrics, beginning with the model loss, which is the objective function the model is minimizing. In the case of DeepLabv3+, the loss function is a categorical cross-entropy, which captures the difference between the target segmentation masks and the model's output. In a healthy training process, the loss should have a decreasing trend, suggesting that the model is learning from the training data. We measured the model performance by three metrics common for image segmentation: True Positive Rate (TPR), True Negative Rate (TNR), and Intersection over Union (IoU), described in Section 2.2.5. Figures 5.6 and 5.7 show the training loss and the validation performance observed throughout training, respectively. The final performance evaluated on the test dataset is captured in Table 5.2.

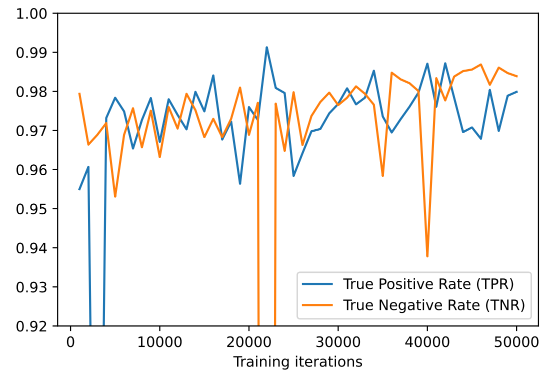
5.3.1.4 Discussion

Using Figure 5.8 as an example, we can identify some advantages of neural network segmentation:

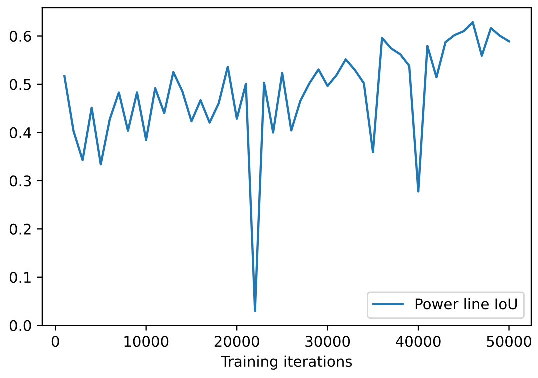
- The model is better at filtering linear objects in the background. It learned a deeper understanding of the scene, and it ignores objects where the filter-based method failed, e.g., transmission towers, or a road in the same direction as the power lines, as illustrated in Figure 5.8.
- No direction of the power lines is assumed, contrary to the filter-based approach, which assumes that the power lines appear vertical in the image.
- Obtaining a cleaner mask overall with less noise coming from short edges in the background.

Metric	value
TPR	97.86 %
TNR	97.88 %
IoU	50.35 %

Table 5.2 Performance of neural network power line segmentation measured on the test dataset.

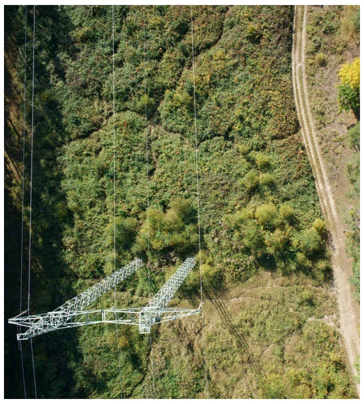


(a) True positive rate and true negative rate.

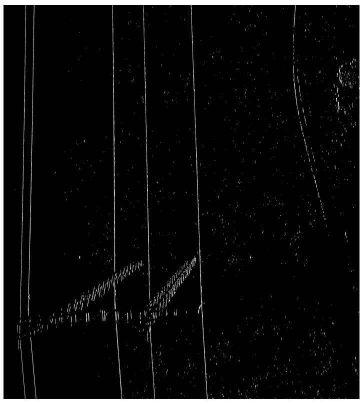


(b) Intersection over union.

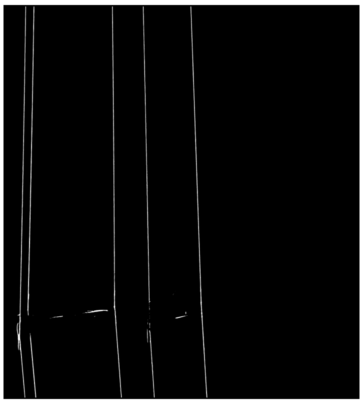
Figure 5.7 Neural network performance on three different metrics throughout training. The performance was evaluated on the validation dataset every 1000 training iterations.



(a) Input image.



(b) Filter-based segmentation.



(c) Neural network segmentation.

Figure 5.8 Comparison of binary segmentation masks obtained by two different methods.

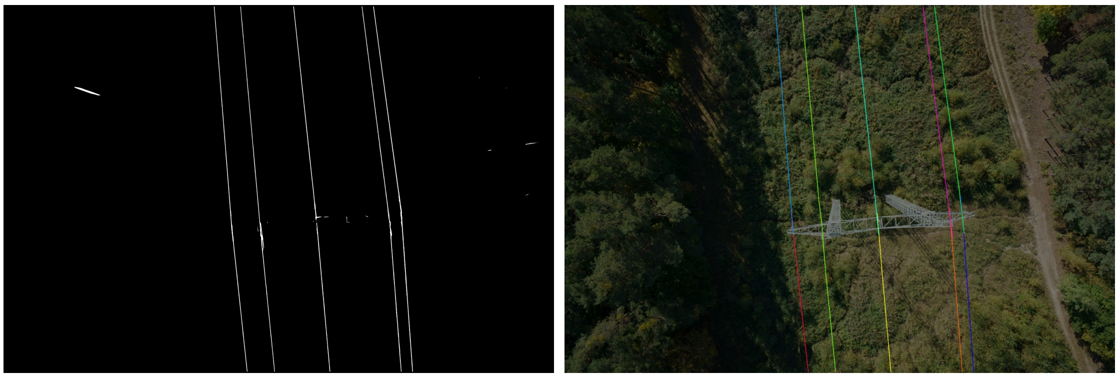
The neural network approach has some disadvantages as well:

- A high-quality manually annotated dataset is required. In our case, we were lucky there is a public TTPLA dataset [11] that sufficiently aligns with our domain. For other similar tasks, it might be necessary to create a new dataset, which is very time-consuming.
- A GPU is necessary for both training and inference.

We provided two different methods for power line segmentation. In ideal conditions, both are suitable for power line detection, but the neural network segmentation produces clearly better results on average because of its more profound understanding of the scene and the resulting ability to filter out other linear structures in the background.

5.3.2 Power Line Detection

The next step is to extract the power lines from the binary segmentation mask. The task is to identify distinct power lines in the image and represent them as geometric entities rather than a set of pixels. As simple as it may seem, we faced many challenges, i.e., missing portions of the wire, noise and outliers, linear structures in the background, wire curvature, and closely recorded or even overlapping wires.



(a) Input – binary segmentation mask. **(b)** Output – detected power lines.

Figure 5.9 Input and output of the power line detection step. Each detected power line is represented by a polyline.

In this section, we propose a domain-specific algorithm for power line detection based on our knowledge and experiments. We give a brief overview of the process followed by a detailed description of each step. When we refer to points, we mean their representation by Euclidean sensor coordinates, as in Section 2.3.3, i.e., pixel coordinates in accordance with Definition 2.1.

5.3.2.1 Input and Output

The input is a binary segmentation mask obtained in the previous step, as described in Section 5.3.1. As an output, we chose to represent each power line as a polyline consisting of several connected line segments. This is simpler than working with curves or polynomials yet sufficient for capturing the line curvature. In addition, we think of the segments of the power line wire before and after the transmission tower as two separate power lines. This allows us to fit individual catenary curves later. The input mask and the detected power lines are visualized in Figure 5.9.

5.3.2.2 Overview of the Algorithm

We start by finding Hough lines in the binary segmentation map and filtering them. Then we split the image into horizontal segments and apply two-level hierarchical clustering of the lines, first by angle and then by distance. Next, we carefully merge the horizontal segments again while matching corresponding clusters of lines to compose distinct power lines. We also count how many segments each power line spans during this step and use this information to filter the power lines. Finally, we merge all Hough lines corresponding to a power line into a simple geometric shape, a polyline, by fitting RANSAC lines. The high-level overview of the algorithm is illustrated in Listing 1.

5.3.2.3 Hough Line Detection (FindHoughLines)

We use the probabilistic version of the Hough line transform explained in Section 2.2.6 to extract a set of h line segments from the input segmentation mask's pixels. As illustrated in Figure 5.10a, many parallel lines were found for each wire, as well as some noisy lines that do not belong to the power lines.

▶ Note 5.2. Technically, the detected entities by the probabilistic Hough line transform are line segments, not infinite lines. For simplicity, we do not distinguish between these two and refer to both as Hough lines unless a clarification is necessary. In mathematical expressions, we denote

```
function PowerLineDetection($segmentationMask$)
    $lines \leftarrow$ FindHoughLines($segmentationMask$)
    $lines \leftarrow$ FilterHoughLines($lines$)
    $lineSegments \leftarrow$ SplitLinesToSegments($lines$)
    for $i \leftarrow 1, |lineSegments|$ do
        $clusteredLines[i] \leftarrow$ ClusterLines($lineSegments[i]$)
    end for
    $powerLinesHough \leftarrow$ MergeClusters($clusteredLines$)
    $clusteredLines \leftarrow$ FilterLineSegments($clusteredLines, powerLinesHough$)
    $powerLinesHough \leftarrow$ MergeClusters($clusteredLines$)
    $powerLines \leftarrow$ MergeLines($powerLinesHough$)
    return $powerLines$
end function
```

Listing 1 Power line detection algorithm.

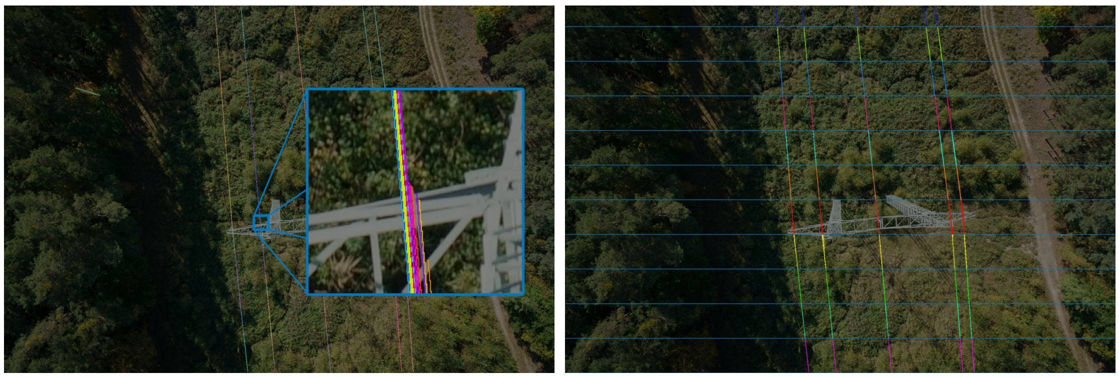

(a) Detected Hough lines. Because of the thickness of the wire, there are tens of parallel lines for every power line.
(b) Splitting the detected Hough lines into horizontal segments. The height of the segment determines the model's tolerance to power line curvature.

Figure 5.10 Detected Hough lines and their subsequent split into horizontal segments.

the detected Hough line segments as $\{l_i\}_{i=1}^{h}$ and the lines that arise by extending each line segment to infinity as $\{L_i\}_{i=1}^{h}$.

When we talk about the distance between a point $\boldsymbol{x}$ and a Hough line L, we mean the shortest orthogonal distance between $\boldsymbol{x}$ and the infinite line L. We denote this distance as $d(\boldsymbol{x}, L)$.

5.3.2.4 Hough Line Filtering (FilterHoughLines)

As demonstrated in Section 5.3.1, sometimes, there can be linear objects, other than power lines, incorrectly highlighted in the binary segmentation mask. These include, for example, roads, trees, or transmission towers in the background (Figure 5.8b). As expected, they are picked up by the Hough line transform as well and need to be filtered out.

At this stage, we have no understanding of the scene yet, but we can leverage one of the constraints we set on the data capture process (Section 4.2.6). According to the guidelines, the power lines should appear vertically in the images. Given a set of h Hough lines $\{L_i\}_{i=1}^{h}$, we calculate the positive angle $\theta_i \in [0°, 180°)$ between each line L_i and the horizontal axis (x) of the image. Then we filter out all the „horizontal“ lines that do not satisfy: $45° \leq \theta_i \leq 135°$.

5.3.2.5 Splitting into Horizontal Segments (SplitLinesToSegments)

The power lines do not always appear entirely straight in the images, and sometimes exhibit a significant curvature. Because we want to apply a precise hierarchical clustering by line angle and distance later, it is necessary to eliminate the influence of curvatures. We do this by splitting the Hough lines into k horizontal segments bounded by $k+1$ horizontal lines $\{s_i | y = i \cdot H_s\}_{i=0}^{k}$, where H_s is the height of each segment. We define a new set of Hough lines for each segment by cropping the existing lines to the segment area: $S_i = \{l_j \cap \mathbb{R} \times [s_i, s_{i+1})\}_{j=1}^{h}, \forall i \in \{1, \dots, k\}$. The segments are visualized in Figure 5.10b.

As a consequence, we also obtain a simple measure for quantifying the power line presence. Later, in the FilterLineSegments step, we can expose a configuration to filter out power lines that appear in less than l segments or have a gap of more than m segments.

5.3.2.6 Line Clustering (ClusterLines)

Line clustering is the core step of the proposed algorithm. We focus on a precise division of Hough lines into groups corresponding to the actual power lines. We iterate over all segments from the previous step and perform hierarchical clustering (as described in Section 2.4.2) on two levels for each one:

Clustering by angle – we state that two Hough lines that differ in angle by more than $\theta_{cluster}$ belong to a different power line. With this assumption, we are able to divide the lines into groups by computing the angle of each line as in FilterHoughLines and applying agglomerative clustering with a threshold $\theta_{cluster}$. We expose the threshold as a configurable parameter but suggest a value of $\theta_{cluster} = 1°$, as shown in the dendrogram in Figure 5.11a. We obtain a_i distinct angle clusters $\{C_{ij}^{angle} \subset S_i\}_{j=1}^{a_i}$, such that $\bigcup_{j=1}^{a_i} C_{ij}^{angle} = S_i$ for the current segment i.

Clustering by distance – it is apparent from Figure 5.11b that clustering by angle is not sufficient to distinguish between all power lines. Hence, for each cluster of lines C_{ij}^{angle} obtained so far, we apply a second level of clustering – by distance. We define the origin $\boldsymbol{o}_i$ of the current segment i as $\boldsymbol{o}_i = [0, \frac{(s_i, s_{i+1})}{2}]^T$ and calculate the distance $D_{ik} = d(\boldsymbol{o}_i, L_k)$ from $\boldsymbol{o}_i$ to each Hough line L_k in the current angle cluster C_{ij}^{angle}. We then perform agglomerative clustering with the average linkage over those distances with a threshold $d_{cluster}$. The result is visualized in Figure 5.11c.

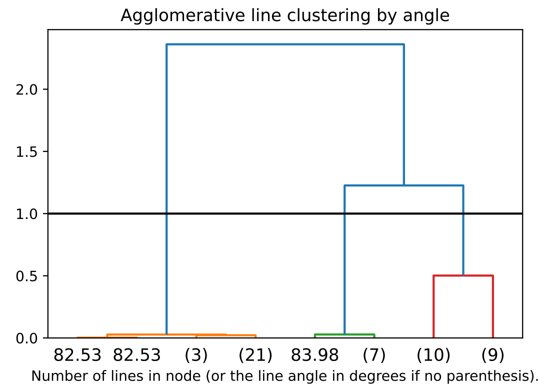
After aggregating all groups of lines for each segment, we obtain the clusters $\{C_{ij} \subset S_i\}_{j=1}^{p_i}$, such that $\bigcup_{j=1}^{p_i} C_{ij} = S_i$ for each segment i. The number of clusters p_i represents the number of power lines identified in segment i.

5.3.2.7 Merge Line Clusters across Segments (MergeClusters)

Once we have clustered the lines in all segments, we have local information about the power lines in each segment. Now we need to merge the segments again while matching the corresponding clusters that belong to the same power line. In other words, we want to find a mapping that assigns each cluster C_{ij} a unique identifier of the actual power line.

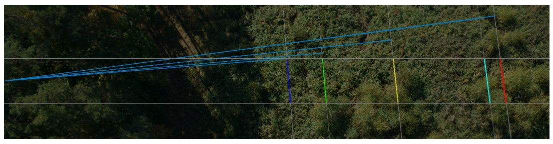
For that, we iterate over the segments and keep track of the power lines detected so far. We declare a set of power lines P, which is initially empty $P := \emptyset$. For each power line, we store a list of segment indices where the power line was detected: $I_{segments} \subset \{1, \dots, k\}$ (where k is the number of horizontal segments). We also store the cluster of Hough lines, which was assigned in each segment: C_{sj_s}, $\forall s \in I_{segments}$.

When stepping into a new segment i, we attempt to match each cluster C_{ij}, $\forall j \in \{1, \dots, p_i\}$ with the closest power line based on the average distance to the segment origin $\boldsymbol{o}_i$. Technically, we compare the average of the distances $d(\boldsymbol{o}_i, L_i)$ for $L_i \in C_{ij}$ of the current cluster and the average



(a) Dendrogram of Hough line clustering by angle. We identified three clusters of lines with the threshold $\theta_{thr} = 1$ (black line). These are visualized in **(b)**.

(b) Hough lines clustered by angle. For the current segment i, we can see three clusters $\{C_{ij}^{angle} \subset S_i\}_{j=1}^{3}$ (red, green, orange), as in the dendrogram in **(a)**.



(c) Hough lines clustered by distance (blue horizontal lines highlight the distance to the segment origin).

Figure 5.11 Hough line agglomerative clustering, first by angle **(a)**, **(b)** and then by distance **(c)**.

of the distances $d(\boldsymbol{o}_i, L_m)$ for $L_m \in C_{kl}$, where C_{kl} is the last detected cluster of some power line in P. The closest power line is the candidate match. We accept the match based on the following conditions:

- The gap between the current segment and the last detected segment of the power line is less than g_{merge} segments.
- The average angle of the last detected power line cluster C_{kl} and the current cluster C_{ij} is not different by more than θ_{merge}.
- The average distance to the segment origin from the last detected cluster C_{kl} and the current cluster C_{ij} is not different by more than d_{merge}.

If all of these conditions are met, the match is accepted, and the current cluster is assigned to the power line. Otherwise, we create a new power line, initiate it with the current cluster, and add it to P.

5.3.2.8 Filtering Power Lines(FilterLineSegments)

After the previous step, we have a much deeper understanding of the scene. We obtained a set of power lines P, and for each one detailed information, in which horizontal segments it appears. Now we can remove all „power lines“ that appear in less than s_{filter} segments as noise.

We rely on the fact that the actual power lines always cover large portions of the image (many horizontal segments), and other linear structures, for example, tree trunks or transmission towers, are typically much shorter. If s_{filter} is too large, this may remove real power lines, especially in the images containing a transmission tower. However, this might be an acceptable trade-off if we have especially noisy segmentation masks. If the data acquisition process is followed correctly, the redundancy of images will still allow us to reconstruct the power lines in 3D successfully.

▶ Note 5.3. Technically, rather than removing the power lines themselves, we remove all clusters of Hough lines that belong to the power lines and perform MergeClusters again, as we observed better performance with this approach.

5.3.2.9 Merging Hough Lines (MergeLines)

Finally, we represent each power line by a single polyline instead of clusters of Hough lines. We make this transition in four steps:

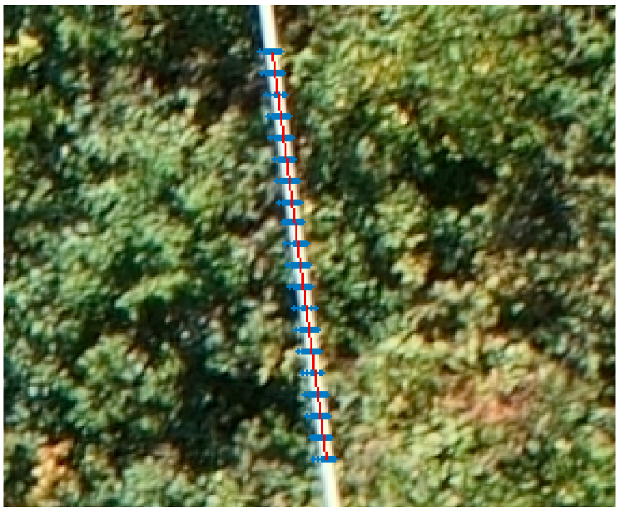


Figure 5.12 Transforming a set of Hough lines into a polyline by sampling points (blue) and fitting a RANSAC line (red).

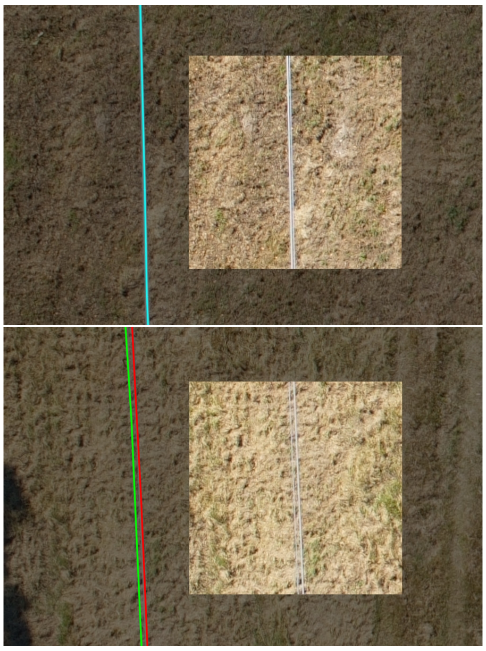


Figure 5.13 Example of unsuccessful (top) and successful (bottom) detection of two closely recorded wires.

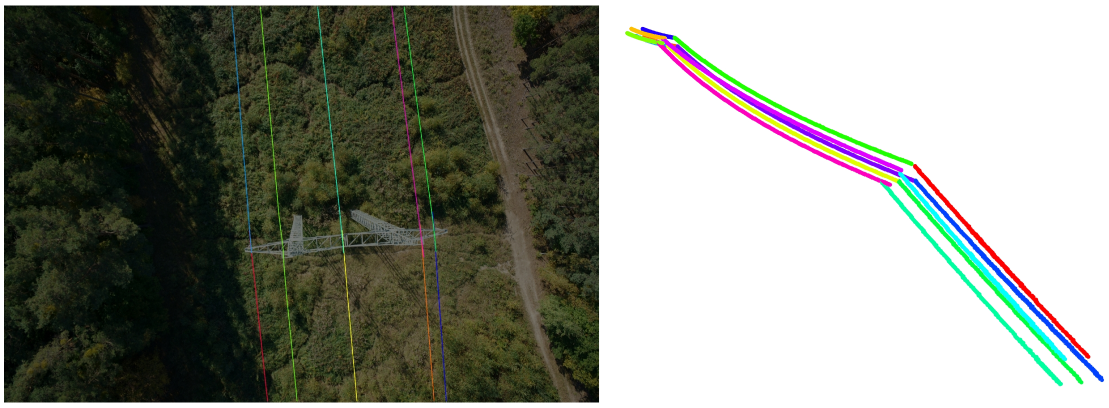
1. Sample all the lines in all the clusters that belong to the power line. We collect points from each Hough line by sampling it every 10 pixels.
2. Fit a single line through all the points using the RANSAC algorithm as explained in Section 2.4.1.
3. Slice the points along the line by projecting them onto the line and splitting them into groups every p pixels.
4. Fit a RANSAC line segment to the points in each group and connect them into a polyline.

The procedure is illustrated in Figure 5.12.

5.3.2.10 Discussion

This section proposed a rather complex domain-specific algorithm for automatic power line detection. We mentioned several parameters, i.e., $\theta_{cluster}, d_{cluster}, g_{merge}, s_{filter}$, which assure a powerful configurability. For instance, they influence the model's sensitivity to wire curvatures, gaps or noise in the segmentation masks, and the minimum distance between two wires. The parameters should be set based on the choice of the segmentation method. We can afford to set stricter thresholds for neural network segmentation because the output has less noise than filter-based segmentation, leading to a better detection quality.

Figure 5.13 shows the limitations as well as strengths of this approach. While the algorithm cannot deal with overlapping wires, in our experiments, it successfully detected power lines that are as close as 10 pixels (with $d_{cluster} = 10$).



(a) Input – detected power lines. **(b)** Output – 3D point cloud of distinct power lines.

Figure 5.14 Input and output of the power line 3D reconstruction step. Each power line in the input is represented by a polyline.

5.3.3 Power Line 3D Reconstruction

This is a critical step of our power line reconstruction pipeline and requires all the previous outputs – the detected power lines, as well as the required results of the photogrammetric pipeline. In this section, we will use photogrammetry and epipolar geometry to reconstruct the 3D power lines. We will therefore use the entities and mathematical notation from 2.3.

5.3.3.1 Input and Output

As input, this step requires the detected power lines visualized in Figure 5.14a and the camera intrinsics and extrinsics from the photogrammetric pipeline, as illustrated in Listing 2. The output is a georeferenced 3D point cloud of distinct power lines, as visualized in Figure 5.14b.

5.3.3.2 Understanding the Output of the Photogrammetric Pipeline

Our photogrammetric pipeline of choice – OpenDroneMap, produces an output JSON (JavaScript Object Notation) file with all necessary camera intrinsics and extrinsics. However, how the parameters are represented may not be immediately clear from the example in Listing 2. It is important to correctly interpret the output to create a high-quality reconstruction. We assume that all the n input images were taken with the same resolution – height h and width w in pixels: $\{I_i | I_i \in \{0, \dots, 255\}^{h,w,3}\}_{i=1}^{n}$.

Camera Intrinsics

At this point, the photogrammetric pipeline has already undistorted the images and the intrinsic parameters for us. Since we used the undistorted images for segmentation and subsequent detection, we must use the corresponding undistorted camera intrinsics. These are a simplified set of parameters from K after performing all nonlinear image corrections.

As a result, the distortion coefficients are now zero, as well as some parameters of the intrinsic matrix, i.e., s and m from 2.3.3. Moreover, the image has been centered, and the principal point $[x_H, y_H]^T$ is assumed to be exactly at the center of the image. In the end, we are left only with *focal* in the output (Listing 2), which is the focal length f from 2.3 normalized by $\max(h, w)$.

```
{
  "cameras":{
    "v2 sony zeiss batis 2/25 7952 5304 brown 0.6944 rgb":{
      "projection_type":"perspective",
      "width":7952,
      "height":5304,
      "focal":0.6742461763542501,
      "k1":0.0,
      "k2":0.0
    }
  },
  "shots":{
    "_DSC4353.TIF.tif":{
      "rotation":[-0.943152446085517,-2.98453285092054, -0.0463902852019688],
      "translation":[-9.21832147611485, -63.4186131306435, 507.880054984430],
      "camera":"v2 sony zeiss batis 2/25 7952 5304 brown 0.6944 rgb",
      "gps_dop":10.0,
      "gps_position":[18.69186778482931, 45.22068640713041, 509.798712238669],
      ...
    },
    "_DSC4379.TIF.tif":{ ... },
    ...
  }
}
```

Listing 2 Example output from the structure from motion step of the photogrammetric pipeline. Under *cameras* are the intrinsic parameters of the *undistorted* camera. Under *shots*, we obtain the extrinsic parameters (rotation and translation).

Finally, we set: $f := focal \cdot \max(h, w)$, $x_H := \frac{w-1}{2}$, $y_H := \frac{h-1}{2}$ and build the undistorted camera intrinsic matrix K:

$$K = \begin{bmatrix} f & 0 & x_H \\ 0 & f & y_H \\ 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

Camera Extrinsics

In Listing 2, we can also find the extrinsic parameters *rotation* and *translation* for each image. Consistently with Section 2.3.3, we create the rotation matrix R from the three parameters (yaw, pitch, roll) in *rotation* using the Rodrigues formula [21]. The vector $\boldsymbol{T}$ is directly listed in the *translation* field of the output. In summary, we obtain the extrinsic parameters for each image: $\{(R_i, \mathbf{T}_i)\}_{i=1}^{n}$, as explained in Section 2.3.3.

5.3.3.3 Overview of the Algorithm

Listing 3 gives a high-level overview of how we divided the power line reconstruction into smaller steps. The algorithm begins by automatically assigning images to flight strips and forming groups of parallel images (left, middle, right). Then we perform a simple power line matching, followed by transfer matching in images where the simple matching failed. Next, we reconstruct the 3D points from the 2D power lines by triangulation, and finally, we fit a catenary curve to reduce

```
function PowerLineReconstruction($powerLines, K, \{(R_i, \mathbf{T}_i)\}_{i=1}^n$)
    $imageGroups \leftarrow$ AssignImagesToGroups($\{(R_i, \mathbf{T}_i)\}_{i=1}^n$)
    $simpleMatches \leftarrow$ SimpleMatching($imageGroups, powerLines$)
    $stereoPairs \leftarrow$ TransferMatching($simpleMatches, imageGroups, powerLines,$
                                            $K, \{(R_i, \mathbf{T}_i)\}_{i=1}^n$)
    $powerLine3DPoints \leftarrow$ Reconstruct3DPoints($stereoPairs, K, \{(R_i, \mathbf{T}_i)\}_{i=1}^n$)
    $powerLine3DPoints \leftarrow$ FitCatenaryCurve($powerLine3DPoints$)
    return $powerLine3DPoints$
end function
```

Listing 3 Power line 3D reconstruction algorithm.

noise and improve quality. The rest of this section will explain the individual steps in more detail.

5.3.3.4 Assigning Images to Parallel Groups (AssignImagesToGroups)

In our data acquisition process, we strictly require three flight strips along the power lines, left, middle, and right, with a side overlap of at least 80 %. Compared to [48], we robustly sort the images based on the location of their camera projection centers instead of sorting by acquisition time. We automatically assign the images to their flight strips and form an ordered list of image triplets (left, middle, right). We then choose two out of the three images to form a stereo pair and perform the 3D reconstruction using epipolar geometry. The third image is crucial for extra robustness, and we use it to disentangle the overlapping wires and compensate for errors in the detection process.

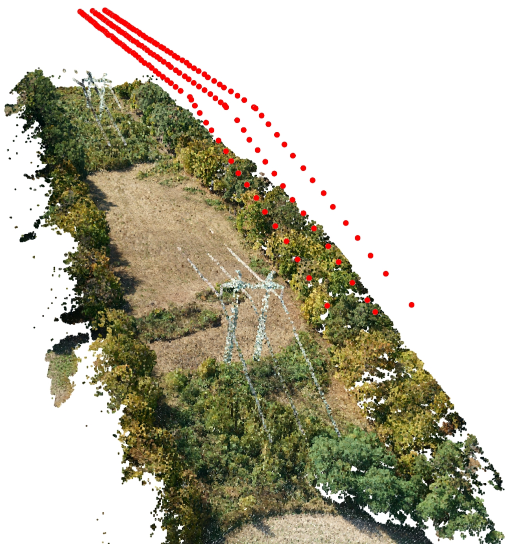
First, we compute the object coordinates of the camera projection center O_i for each image. Since the projection center is at the origin of the camera coordinate system ${}^c\boldsymbol{O}_i = [0, 0, 0]^T$, and a transformation between the world and camera coordinate systems is given by: ${}^c\boldsymbol{X} = R_i\boldsymbol{X} + \boldsymbol{T}_i$, where ${}^c\boldsymbol{X}$ are the Euclidean camera coordinates, and $\boldsymbol{X}$ are the Euclidean object coordinates, we can simply perform the inverse transformation: $\boldsymbol{X} = R_i^T({}^c\boldsymbol{X} - \boldsymbol{T}_i)$. Applied to point ${}^c\boldsymbol{O_i}$, this yields the projection centers $\{\boldsymbol{O_i} | \boldsymbol{O_i} = -R_i^T\boldsymbol{T}_i\}_{i=1}^n$, which are visualized in Figure 5.15a.

Once we have the 3D object coordinates of the camera projection centers, we lose the Z coordinate and fit a 2D polyline using the RANSAC algorithm similarly to Section 5.3.2.9. Afterward, we calculate the signed distance from each projection center point to the polyline (negative distance left of the polyline and positive right of the polyline) and perform hierarchical clustering into three clusters using the average linkage. This classifies the points into the three flight strips. Afterward, we sort all points along the direction of the polyline. Finally, we iterate over the ordered images in the left strip and find the closest middle and right image to form groups of three images along the polyline direction. The resulting groups are visualized in Figure 5.15b.

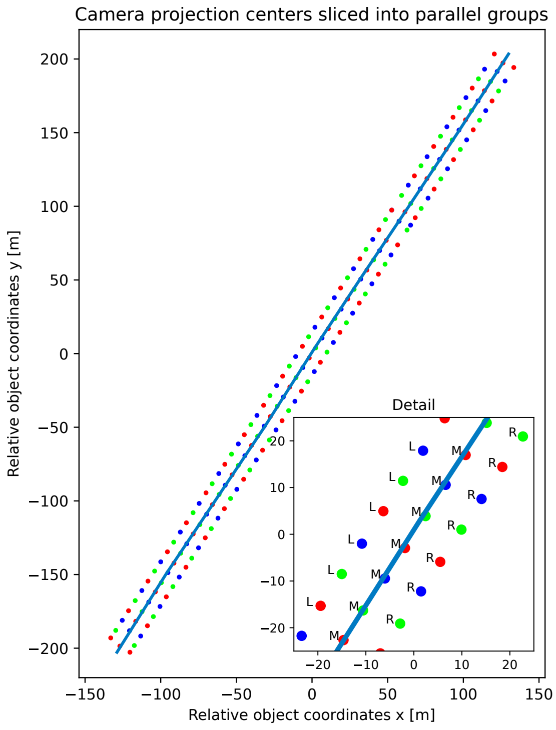
5.3.3.5 Simple Matching of the Power Lines (SimpleMatching)

To achieve a successful 3D reconstruction from two images, it is necessary to have pairs of sensor coordinates in both images that correspond to the same 3D point in the scene. Traditionally, these are obtained by running a feature detector followed by a point matcher, just like in the photogrammetric pipeline described in Section 5.2. This is, however, inefficient for thin homogeneous structures. Hence, we leverage our understanding of the scene and detected power line locations and derive the pairs of sensor coordinates purely geometrically, just like [44, 48].

For that, we must first match the corresponding power lines across images. As opposed to [44] and [48], where the authors apply manual work, we are able to match the power lines automatically by exploiting the constraints we set on the image acquisition. Namely, we require all power lines to be visible in each image and three parallel flight strips for redundancy. Additionally, we



(a) Camera projection centers $\{\boldsymbol{O_i}\}_{i=1}^n$ in the object coordinates, visualized by red points.



(b) Groups of left (L), middle (M), and right (R) parallel images, formed by fitting a polyline through the camera projection centers, hierarchical clustering, and computing signed distances to the polyline.

Figure 5.15 Assigning camera projection centers to groups of parallel images.

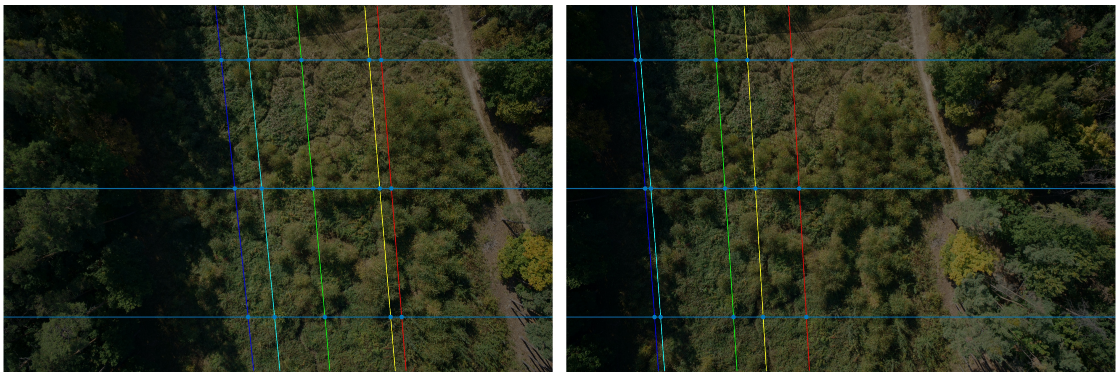


Figure 5.16 Simple matching of the power lines in a valid pair of left and right parallel images. We draw three horizontal lines (blue), record the intersection with the power lines, validate that they are complete and consistent, and then match the power lines by sorting them according to the x coordinates of their intersection points.

need a user input – the number of power lines p in the scene. These are all easily achievable in practice and give us the ability to perform what we call a *simple matching* of the power lines.

This straightforward process begins with a parallel group of left, middle, and right images. We draw three horizontal lines across each image (one at the top, middle, and bottom) and record the intersections with the detected power lines. If the image contains exactly p power lines and all three horizontal lines register p intersections, we pronounce the image as *valid* for simple matching. Otherwise, we skip the image and mark it as *invalid*.

For each parallel group, we select two valid images as a stereo pair for 3D reconstruction. Since we know that there are exactly p power lines in both images, we can simply match them according to their horizontal position left to right.

If all three images in the group are valid, we select the left and right images. Otherwise, thanks to the third redundant strip, there is a high chance that at least the left and middle, or right and middle images will be valid and selected. It is now apparent how the third flight strip provides extra robustness to overlapping or undetected power lines. If there are no two valid images, we skip this image group and mark it for transfer matching. Figure 5.16 demonstrates a simple match of the power lines in a valid pair of left and right images.

5.3.3.6 Transfer Matching of the Power Lines (TransferMatching)

So far, we matched the power lines „horizontally“ across images in the same group, where the simple matching succeeded. However, for catenary curve fitting, we actually need to assign a unique global identifier to each power line, in other words, match them also „vertically“ across parallel groups. When iterating over the groups, there are two possibilities:

- An uninterrupted series of simple matches. This is normally the case for all the groups in between transmission towers if the data acquisition constraints are satisfied. In this case, we can simply unify all the power lines across the series and assign the same global identifiers in each parallel group since we know that the images contain all p power lines ordered horizontally from left to right.
- We run into an unsuccessful match. This usually happens around transmission towers, where a handful of images typically do not meet the conditions for a simple match. In this case, we apply *transfer matching* to carry the global identifiers from the previous simple match and continue carrying them for all consecutive unsuccessful matches (normally, these are three to five groups of images around the transmission tower, depending on the front overlap). Eventually, we encounter a simple match again and start assigning new global identifiers.

From 2D to 3D back to 2D

Transfer match works with two consecutive groups of parallel images: $G^1 = (I_L^1, I_M^1, I_R^1)$ and $G^2 = (I_L^2, I_M^2, I_R^2)$. It further assumes that G^1 was successfully matched, and a stereo pair S^1 for 3D reconstruction was selected. Without loss of generality, let us assume that the power lines in G^1 were matched by simple matching, and the stereo pair was created from the valid left and right images: $S^1 = (I_L^1, I_R^1)$. This means that for every power line in I_L^1, we know its location in I_R^1. Furthermore, G^2 did not satisfy the conditions for a simple match and was marked for transfer matching. We select a stereo pair containing the left and right images $S^2 = (I_L^2, I_R^2)$ for G^2. The transfer match happens in three steps:

1. Reconstruction of 3D points for each power line in S^1. This will be explained in more detail in Section 5.3.3.7.
2. Reprojecting the 3D points to S^2. We create the projection matrices P_L^2 and P_R^2 for the images I_L^2 and I_R^2, respectively. According to Section 2.3.3: $P_L^2 = K \left[R_L^2 \mid \boldsymbol{T}_L^2 \right]$, where K is the intrinsic matrix, and $R_L^2, \boldsymbol{T}_L^2$ are the rotation matrix and translation vector for I_L^2. The matrix P_R^2 for I_R^2 is created analogously.

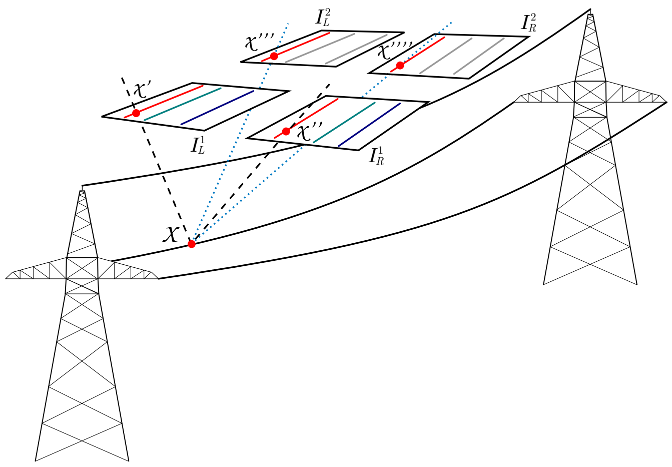


Figure 5.17 Transfer match of the power line global identifiers between two stereo pairs of images $S^1 = (I_L^1, I_R^1)$ and $S^2 = (I_L^2, I_R^2)$. The power lines in S^1 were matched by simple matching (red, green, blue). The point $\mathcal{X}$ is reconstructed from two matching points χ' and χ'' obtained using epipolar geometry from the „red“ power line in S^1. The point $\mathcal{X}$ is then reprojected to I_L^2 and I_R^2, yielding the 2D projections χ''' and χ''''. As it happened, the 2D projections landed on one of the three power lines in I_L^2, and I_R^2, and we can therefore assign it the „red“ label. We call this a transfer match.

We then apply the matrices on the 3D points. As an example, we can apply it on the 3D point $\mathcal{X}$ from Figure 5.17 expressed in homogeneous object coordinates ($\mathbf{X}$) and obtain the corresponding projections $\mathbf{x}''' = P_L^2\mathbf{X}$ and $\mathbf{x}'''' = P_R^2\mathbf{X}$ in I_L^2 and I_R^2, respectively.

3. Matching the power lines in S^2 based on their distance to the 2D projections.

Figure 5.17 shows an example of reconstructing the 3D point $\mathcal{X}$ from two corresponding 2D points χ' and χ'' in I_L^1 and I_R^1, respectively, and reprojecting it to I_L^2 and I_R^2 to find the power line match in S_2.

We perform two passes of the transfer matching algorithm – forward and backward pass through all the groups to further increase the number of matches. Thanks to the data acquisition constraints and the third flight strip, we are able to match the power lines fully automatically, without any intervention from the user. Most power lines are matched during the simple matching phase, and the ones in cluttered images around transmission towers are matched during the transfer matching phase.

5.3.3.7 3D Reconstruction (Reconstruct3DPoints)

Finally, after the power lines are matched, we can reconstruct the 3D points from the stereo pairs we selected (note that reconstructing the 3D points was also part of transfer matching). In this section, we explain how we use epipolar geometry to find corresponding points that belong to power lines in a pair of images and triangulate the location of the 3D point. We demonstrate the process on a stereo pair $S = (I_1, I_2)$.

Preparation

As outlined in Section 2.3.3, we can now build a fundamental matrix F_{12} that captures the relationships between the two images I_1 and I_2, as in Equation 2.11:

$$F_{12} = K^{-T}[\mathbf{T}_{12}]_\times R_{12}K^{-1}.$$

From Equation 5.2, it is clear that our matrix K is invertible. Hence, K^{-1} and K^{-T} always exist. The relative rotation matrix R_{12} and translation vector $\mathbf{T}_{12}$ of the second camera with

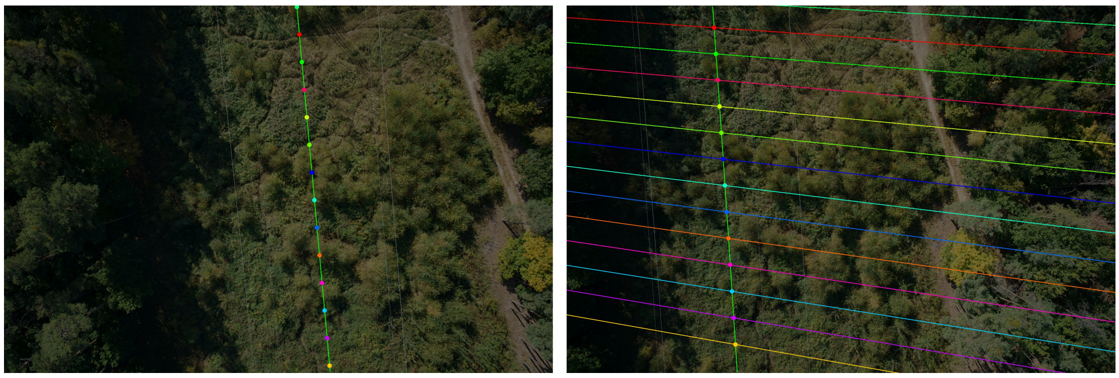


Figure 5.18 Given a stereo pair $S = (I_1, I_2)$, we draw the epipolar lines $\{\ell''(\chi_i)\}_{i=1}^{k}$ corresponding to k points $\{\chi_i'\}_{i=1}^{k}$ sampled from the „green“ power line in I_1. For each χ_i', we obtain the matching point χ_i'' by taking the intersection of the corresponding epipolar line $\ell''(\chi_i)$ with the „green“ power line in the image I_2. Each pair of matching points χ_i' and χ_i'' is visualized by a unique color. These are projections of the exact same 3D points in the scene and can be used for triangulation.

respect to the first camera must be explicitly derived from the extrinsics $R_1, R_2, \mathbf{T}_1, \mathbf{T}_2$. We do so by going from the first camera through the object coordinates, to the second camera. In other words, we transform the point in camera coordinates of the first camera ${}^{c_1}\boldsymbol{X}$ to its coordinates in the object coordinate system $\boldsymbol{X}$:

$$\boldsymbol{X} = R_1^T({}^{c_1}\boldsymbol{X} - \boldsymbol{T}_1),$$

and project it to the second camera ${}^{c_2}\boldsymbol{X}$:

$$\begin{aligned}
{}^{c_2}\boldsymbol{X} &= R_2\boldsymbol{X} + \boldsymbol{T}_2 \\
{}^{c_2}\boldsymbol{X} &= R_2(R_1^T({}^{c_1}\boldsymbol{X} - \boldsymbol{T}_1)) + \boldsymbol{T}_2 \\
{}^{c_2}\boldsymbol{X} &= R_2R_1^T{}^{c_1}\boldsymbol{X} - R_2R_1^T\boldsymbol{T}_1 + \boldsymbol{T}_2 \\
{}^{c_2}\boldsymbol{X} &= R_{12}{}^{c_1}\boldsymbol{X} - \boldsymbol{T}_{12}.
\end{aligned}$$

In the last step, we arrived at the relative rotation $R_{12} = R_2R_1^T$ and the relative translation $\boldsymbol{T}_{12} = -R_2R_1^T\boldsymbol{T}_1 + \boldsymbol{T}_2 = -R_{12}\boldsymbol{T}_1 + \boldsymbol{T}_2$. Finally, we can build the fundamental matrix F_{12} using these parameters and proceed with the reconstruction.

Finding Matching Points

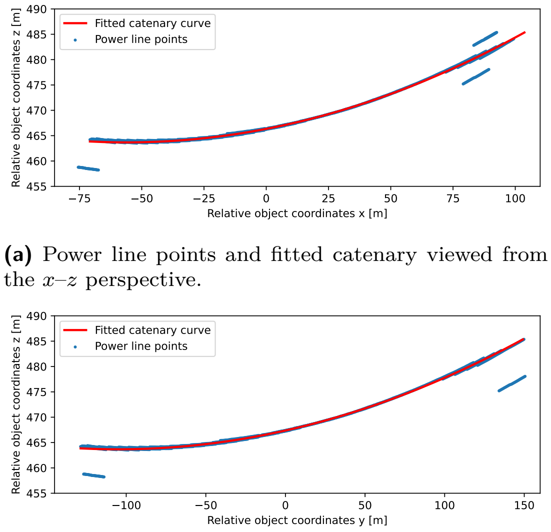
As Figure 5.17 shows, we need two corresponding points, χ' and χ'', for every reconstructed 3D point $\mathcal{X}$. Obtaining these points is now simple, given all the previous results. We choose one image from a stereo pair $S = (I_1, I_2)$, e.g., I_1, sample a point χ' on some power line and use the fundamental matrix to calculate the corresponding epipolar line in the other image. As outlined in Section 2.3.3, the line is given by

$$\mathbf{l}''(\chi) = F_{12}^T\mathbf{x}'.$$

To obtain the point χ'', we calculate the intersection of $\mathbf{l}''(\chi)$ and the corresponding matched power line in the other image. Figure 5.18 visualizes the process of 3D reconstruction of the power lines from a stereo pair.

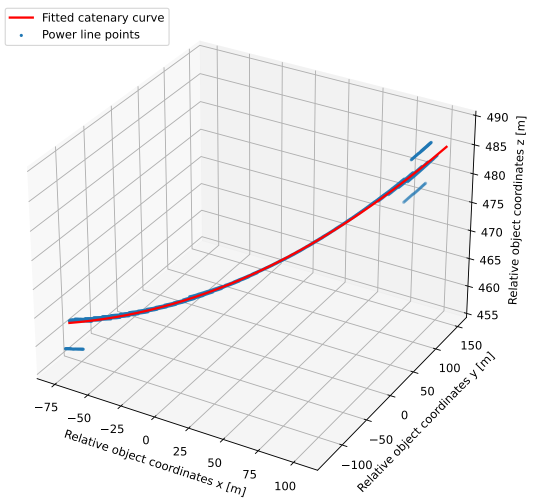
3D Point Triangulation

Finally, to find a 3D point $\mathcal{X}$ corresponding to a pair of matching 2D points χ', and χ'' in the sensor coordinates, we use the least-squares triangulation method from [23, p. 312].



(a) Power line points and fitted catenary viewed from the x–z perspective.

(b) Power line points and fitted catenary viewed from the y–z perspective.



(c) Power line points and fitted catenary in 3D.

Figure 5.19 An example of 3D power line points (blue) and a catenary curve (red) fitted using the RANSAC algorithm. The 3D power line points keep the catenary shape when viewed in the x–z or y–z planes.

5.3.3.8 Catenary Curve Fitting (FitCatenaryCurve)

As the last step of the power line reconstruction pipeline, we fit the catenary curve to the individual 3D reconstructed power lines. This improves the reconstruction quality significantly since we fit the curve robustly using the RANSAC algorithm. So far, each reconstructed power line contains a set of p 3D points $\{\boldsymbol{X}_i = [X_i, Y_i, Z_i]^T\}_{i=1}^p$. The catenary is fitted in three steps:

1. We fit a regular RANSAC line to the 2D points $\{\boldsymbol{x}_i^{XY} = [X_i, Y_i]^T\}_{i=1}^p$ (dropping the Z coordinate).

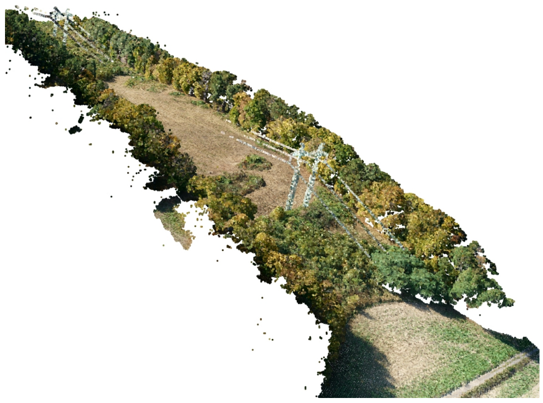
2. We leverage the fact that the 3D points keep the catenary shape even if we view them from any side, e.g., the x–z or y–z plane [6]. This allows us to fit the 2D catenary curve from Equation 2.2 to the 2D points $\{\boldsymbol{x}_i^{XZ} = [X_i, Z_i]^T\}_{i=1}^p$ or $\{\boldsymbol{x}_i^{YZ} = [Y_i, Z_i]^T\}_{i=1}^p$. We decide which „side“ to use based on the length of the domain that the points occupy. In other words, we use the points $\{\boldsymbol{x}_i^{XZ}\}_{i=1}^p$ if $(\max_{i=1}^p X_i - \min_{i=1}^p X_i) > (\max_{i=1}^p Y_i - \min_{i=1}^p Y_i)$, and $\{\boldsymbol{x}_i^{YZ}\}_{i=1}^p$ otherwise.

3. Lastly, we sample XY points from the line we got in step 1 and obtain their Z coordinates using the 2D catenary model.

A real example of the fitted catenary curve is illustrated in Figure 5.19, which also highlights the robustness to outliers as a result of using the RANSAC algorithm.

5.4 Visualization

This component is the last step of the whole pipeline, and it is the only interactive part that involves a human operator. The goal is to calculate the distance between the terrain and power lines and highlight areas of vegetation encroachment.


(a) Input – georeferenced point cloud of the terrain.

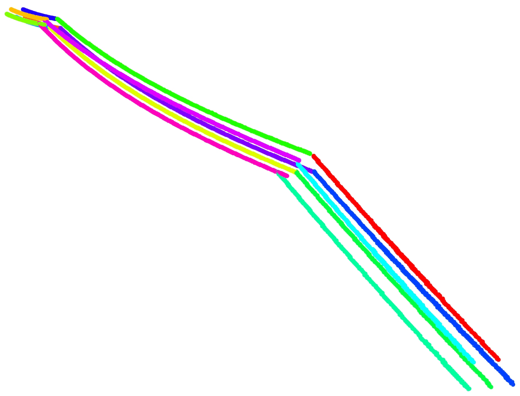

(b) Input – georeferenced point cloud of the power lines.

Figure 5.20 Input of the visualization component.

5.4.1 Input and Output

The inputs of this component are two georeferenced point clouds:

1. Point cloud of the terrain from the photogrammetric pipeline (Figure 5.20a).
2. Point cloud of the power lines from the power line reconstruction pipeline (Figure 5.20b).

The output is an interactive visualization of vegetation encroachment in the power line corridor, including its geographic coordinates (Figure 5.21).

5.4.2 Realization

We chose the open-source tool CloudCompare [69], because of its efficient manipulation with large point clouds. We import the two point clouds illustrated in Figures 5.20a and 5.20b and use the *cloud-to-cloud distance* feature of CloudCompare to calculate their distances. Then we define a color scale based on the power line corridor right of way (see Figure 2.1) and examine the results. An example output is shown in Figure 5.21.

These operations are done manually in a graphical user interface, which allows for a more interactive approach. The user can, for example, adjust the color scale or define multiple color scales, one for evaluating the vegetation encroachment from the side and another one from underneath the power lines (as there are often different regulations for the minimum allowed distance). It is also possible to manually edit the point cloud and combine the two different color scales into a single point cloud 5.22. Other operations, such as manually removing points, subsampling, or creating a mesh for the terrain point cloud, can further improve the visualization. These are out of scope of this work.

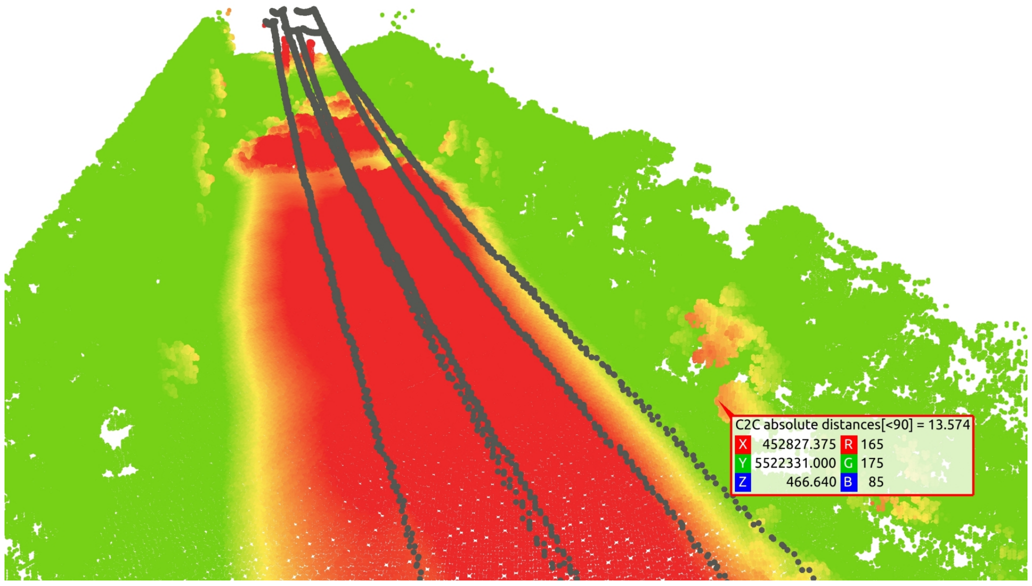


Figure 5.21 Output of the visualization component – the power line corridor with visualized distances in meters between **(a)** and **(b)**. The color scale is customizable, and in this example, the points closer than 12 m are displayed in red, 15 m yellow, and 17 m green, with continuous transitions between the colors. There is, in fact, some vegetation closer than the allowed distance (15 m) for this particular type of power lines (220 kV), as illustrated by the highlighted point on the right. The actual distance from the clutter to the power lines is directly available in the tooltip (13.574 m), as well as the real object coordinates (X, Y, Z), written in the *WGS84* format (UTM zone 33N in this example), which can be easily converted to GPS coordinates.

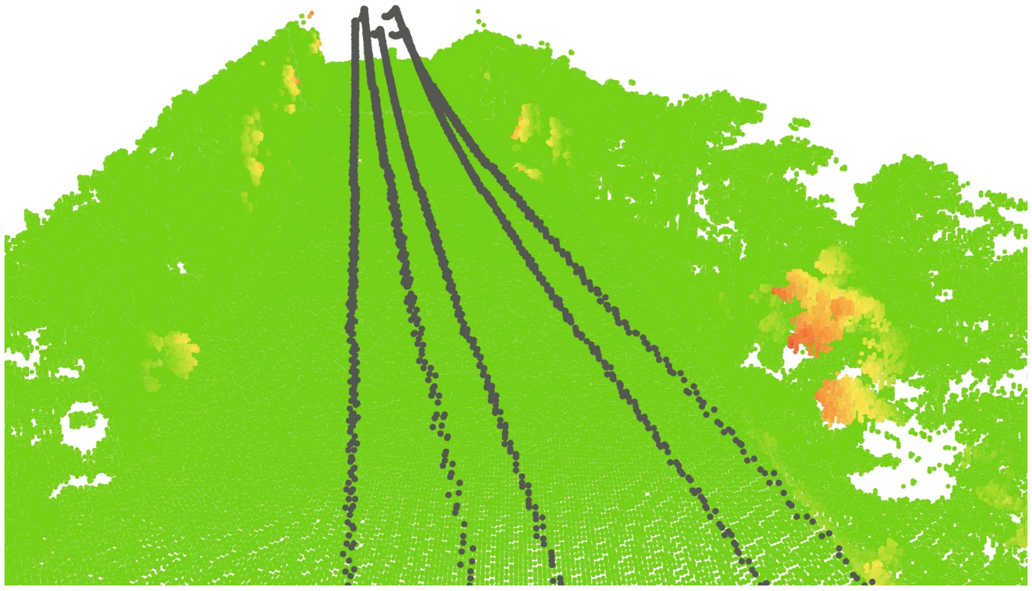


Figure 5.22 Advanced visualization of the power line corridor, created by manually segmenting the point cloud, and combining two different color scales: one for vegetation hazard from the side, and a second one for vegetation hazard underneath the power lines. Automation of this step is out of scope of this work.

Chapter 6

Implementation

This chapter focuses on the implementation of the proposed power line reconstruction pipeline and the support tools for full automation encompassing the entire process of power line vegetation management using UAV images. We chose the Python programming language for its abundance of high-quality open-source libraries, not only for computer vision and machine learning.

The chapter begins with the list of essential libraries we used throughout the implementation, followed by a basic structure of the application. Lastly, we give a brief example of the workflow, including the setup and processing of a dataset.

6.1 Python Libraries

We used several popular Python libraries for various computer vision and machine learning algorithms mentioned throughout this work. All software used for implementation is open-source, with licensing allowing for-profit use. The list of essential libraries:

Poetry [70] is a modern package and dependency manager for Python. It automatically creates a virtual environment for each project and exposes a simple configuration that can be versioned.

Poe the Poet [71] is a task runner for Poetry that allows us to write commands that always execute code in the Poetry virtual environment with the correct versions of all packages. We wrote Poe commands for installing dependencies, downloading the required files, and executing all parts of the pipeline. These commands are our main interface with the application.

OpenCV [72] is a popular highly-optimized computer vision library. It is written in the C++ programming language with Python bindings and offers implementation of the most known computer vision algorithms, including some for photogrammetry. We used it mainly for implementing filter-based segmentation (Section 5.3.1.2), finding Hough lines (Section 5.3.2), and 3D point triangulation using the least-squares algorithm (Section 5.3.3.7).

MMSegmentation [67] is a deep learning segmentation framework based on PyTorch, which implements the DeepLabv3+ model we used in the neural network segmentation (Section 5.3.1.3).

PyODM [57] is a Python API from the OpenDroneMap ecosystem. We used it to write a small client for remote execution of the photogrammetric pipeline and automated download of the outputs.

scikit-learn [73], scikit-image [74] contain implementations of the RANSAC algorithm (Section 2.4.1).

Shapely [75] is a planar geometry library that we used for various 2D geometric operations.

6.2 Application Structure

The application is structured into four main parts:

config collects all configuration files for both photogrammetric and power line reconstruction pipelines. The main configuration file is called `power_lines_cfg.yaml` and holds all user configuration, including parameters from the power line reconstruction pipeline described in Section 5.2.

data contains the inputs and outputs of the application, and some other large files, such as the trained segmentation weights.

powerline3d contains the source code of the power line reconstruction pipeline. It consists of three modules – `segmentation`, `detection`, `reconstruction`, and shared utilities.

tools hold the scripts for executing individual steps from `powerline3d`, training and testing the segmentation neural network, running the photogrammetric pipeline, and a few other tools, e.g., for transforming the *ttpla-voc* dataset, flight planning, or camera calibration. We execute the scripts in this folder through Poe.

The overview of the most important parts of the application:

```
config ........................................ configuration files
 └─ power_lines_cfg.yaml ...................... main configuration file
├─ powerline3d ........ source code of the power line reconstruction pipeline
│  ├─ segmentation ........................ power line segmentation module
│  ├─ detection ............................... power line detection module
│  ├─ reconstruction .................... power line 3d reconstruction module
│  ├─ geometry ................................. common geometric utilities
│  └─ util ....................................... other common utilities
├─ tools ....... various tools and Python scripts for executing the pipeline
└─ data ............................... input and output of the application
   ├─ images ..................................... place for the input images
   ├─ odm ........................................... OpenDroneMap output
   ├─ segmentation .... power line segmentation output (and trained neural network weights)
   ├─ detection ................................... power line detection output
   └─ power_lines.ply ....................... power line reconstruction output
```

6.3 Power Line Inspection Processing Workflow

In this section, we zoom in on the actual processing workflow. After obtaining all application files and external system dependencies (e.g., Python, the complete list is part of the documentation), we can start with the initialization.

Initialization

First, we must install all required Python packages using `poe install` and start the OpenDroneMap server. We recommend running the photogrammetric pipeline remotely on a powerful Linux server or cluster since OpenDroneMap is computationally heavy and can consume more than 100 GB of RAM (depending on the number of images and CPU cores).

The next step is to check the main configuration file `config/power_lines_cfg.yaml`. If the OpenDroneMap server is running remotely, we must configure its network address. We may optionally change any of the other settings.

Processing

Once everything is ready, we can place the georeferenced UAV images in `data/images` and run the pipeline by `poe runall <number-of-power-lines>`, which automatically executes all processing steps and stores the output. We must supply the number of power lines in the scene. After the process is completed, the georeferenced point cloud of the terrain can be found in `data/odm/odm_georeferencing/odm_georeferenced_model.laz` and the point cloud of the power lines in `data/power_lines.ply`.

For maximum efficiency, we can run the steps separately. Since the power line reconstruction pipeline depends only on the output from the structure from motion (see Figure 5.1), it can be executed in parallel with the dense point cloud reconstruction. We provide the individual commands for more advanced usage:

- `poe runodm-opensfm` – run the structure from motion part of OpenDroneMap pipeline,
- `poe runodm-pc` – run the rest of the photogrammetric pipeline,
- `poe segmentation` – run power line segmentation,
- `poe detection <number-of-power-lines>` – run power line detection,
- `poe reconstruction <number-of-power-lines>` – run power line reconstruction.

Each step can be run individually if all its input dependencies are satisfied (see Figure 5.1). This enables a more interactive approach. For example, we can examine the output of any step, change the configuration based on what we learned, and rerun the step only without executing the entire pipeline (which typically takes hours).

Visualization

Lastly, we import the two aforementioned point clouds into CloudCompare and compute the distances using its graphical user interface. The vegetation encroachment can be highlighted using an arbitrary user-defined color scale.

Chapter 7

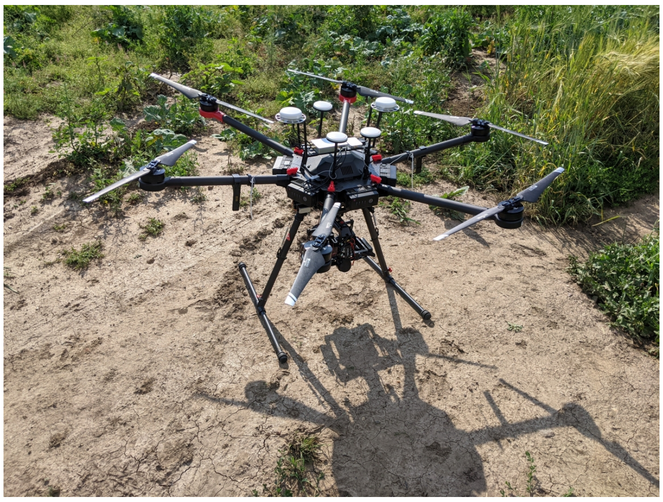
Experiments

In this chapter, we report the experiments we conducted to verify the robustness of the proposed solution. We created three representative datasets of the 220 kV power lines at various lighting conditions and vegetation life cycle phases. The chapter begins with an overview of the hardware and a brief section about flight planning and autonomous missions. Then, it continues to describe each of the three datasets in detail. Lastly, we report the results of our proposed solution.

7.1 Hardware

The UAV platform of choice was DJI Matrice 600 (Figure 7.1a), a hexa-rotor drone capable of lifting 6 kg of additional weight. Its stability, as well as georeferencing precision, was improved by DJI D-RTK GNSS mobile station (Figure 7.1b).

The payload consisted of SONY Alpha 7R II full-frame camera with a ZEISS Batis 25 mm lens and Air Commander ENTIRE r3, which allows for the remote camera trigger as well as writing accurate GPS coordinates in the image EXIF metadata. The camera was stabilized with the GREMSY H3 gimbal. Additionally, we used Air Commander LINK v2 to control the camera exposure settings remotely (Figure 7.2).


(a) DJI Matrice 600 hexa-rotor UAV with payload.

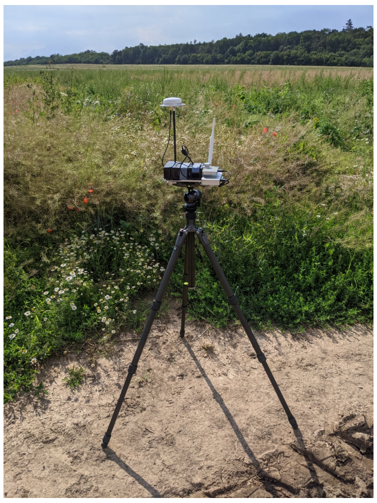

(b) DJI D-RTK GNSS mobile station.

Figure 7.1 Hardware equipment.

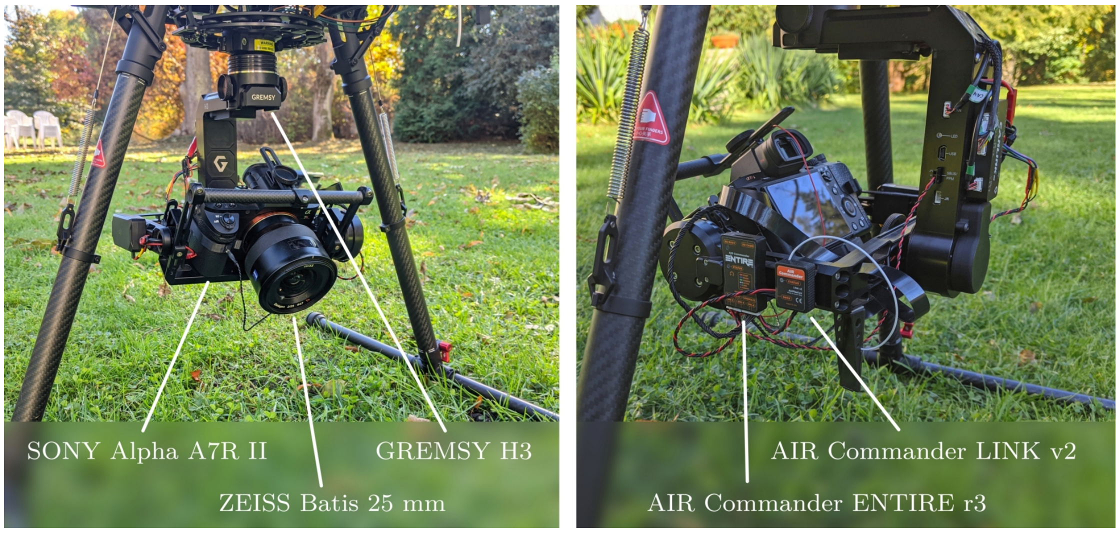


Figure 7.2 UAV Payload. The images were captured with a SONY Alpha A7R II camera combined with a ZEISS Batis 25 mm lens mounted on a GREMSY H3 gimbal. We used two additional devices, AIR Commander ENTIRE r3 for triggering the camera and writing GPS coordinates to the EXIF metadata and AIR Commander LINK v2 for controlling camera exposure settings during flight.

7.2 Flight Planning

For planning autonomous missions, we used the UgCS PRO planner [76], which includes support for digital elevation models to follow the terrain height. This feature is essential to keep the flight height constant throughout the whole mission, especially in difficult terrain with hills. This allowed us to plan and execute a fully autonomous flight exactly according to the plan in Section 4.2. UgCS PRO is the only paid software we used in this work. Nonetheless, we found it was the best option for our purpose during our research. Its main benefits are the already mentioned support for terrain elevation, importing the location of power lines, and a highly configurable corridor mapping mode (Figure 7.3).

As discussed in Section 4.2, many of the mission parameters are interconnected, for example, the maximum speed of the UAV depends on the exposure settings, flight height on GSD, and everything depends on the camera and lens. As a result, we created a simple utility for calculating the optimal flight height, speed, GSD, and the resulting diffraction. Moreover, it provides a useful summary of the relevant parameters of the equipment, according to Section 4.2. The output of the utility is illustrated in Listing 4, where one can also find the relevant parameters of the camera hardware we used in our experiments.

7.3 Datasets

To verify the robustness of the proposed algorithm, we created three datasets under different lighting conditions and vegetation life cycle phases. The datasets were captured at three different locations along a single line of 220 kV wires. This line has 5 power lines (three transferring electricity and two communication wires). All three datasets contain a single span of wires between two transmission towers, with a small overlap on each side.

Dataset 1 was created on October 10, before the leaf fall, and datasets 2 and 3 were both created on November 20, with significantly fewer leaves and under cloudy weather. We have already shown examples from dataset 1 throughout Chapter 5, in Figures 5.20, 5.21, and 5.22.

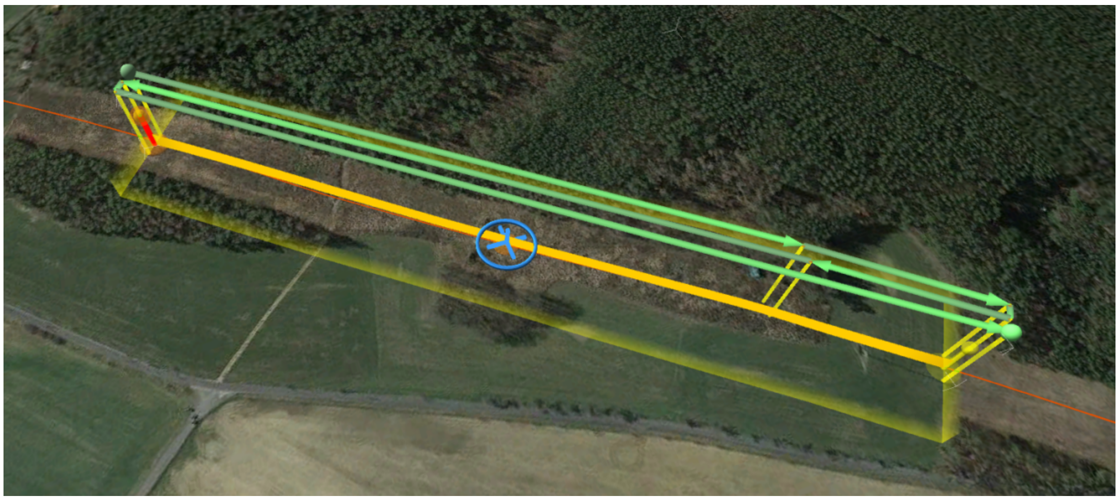


Figure 7.3 Example of an autonomous mission planned in UGCS. This mission, in particular, was used to capture dataset 3 (see Table 7.2). Notice especially the three parallel flight strips and the orange line, which is the power line center location imported from the OpenStreetMap [77] data.

Command:

```
poe flightplanner sony_a7r2 --lens batis_25 \
                            --gsd 1.0 \
                            --aperture 5.6 \
                            --shutter 0.01
```

Output:

```
Camera:                  Sony Alpha A7R II
Image size:              7952 x 5304
Sensor size:             36mm x 24mm
Crop factor:             1.0
Pixel pitch:             0.0045249mm
------------------------------------------
Lens:                    Zeiss Batis 25mm
Focal length:            25mm
------------------------------------------
Equiv. focal length:     25.00mm
Shutter speed:           0.01s
Aperture size:           f/5.6
------------------------------------------
GSD:                     1.00cm
Flight height:           55.2m
Max speed:               2m/s ~ 5km/h
Diffraction:             0.0075152mm
```

Listing 4 Example output of the flight planner utility. Given the hardware specifications, target GSD, and exposure settings, the utility calculates the optimal flight height, maximum speed, and diffraction.

Photogrammetric Pipeline	Dataset 1	Dataset 2	Dataset 3
Structure from Motion	14 hrs. 25 min.	1 hr. 49 min.	3 hrs. 53 min.
Dense Point Cloud Reconstruction	7 hrs. 37 min.	5 hrs. 12 min.	6 hrs. 48 min.
Power Line Reconstruction			
Power Line Segmentation	50 min.	49 min.	55 min.
Power Line Detection	27 min.	18 min.	30 min.
Power Line Reconstruction	1 min.	1 min	1 min
Total Processing time	23 hrs. 20 min.	8 hrs. 9 min.	12 hrs. 7 min

Table 7.1 Processing time of each stage.

Figures 7.4 and 7.5 show datasets 2 and 3, respectively. The mission details for each dataset are summarized in Table 7.2.

The datasets were processed on a powerful Linux compute cluster with 64 cores of Intel(R) Xeon(R) Gold 6254 CPU @ 3.10GHz, 314 GB of RAM, and a Tesla V100 32GB GPU. Table 7.1 shows the processing times of each dataset. We can see that the structure from motion step of dataset 1 took very long compared to the other two datasets. Since all parameters between flights were held mostly constant (see Table 7.2), we suspect it is because dataset 1 was created during a sunny day, resulting in sharp shadows moving throughout the data capture, causing noise during the incremental reconstruction. Naturally, the processing time depends on the server load and other running processes. However, each dataset was processed several times with similar results. The time of neural network and filter-based segmentation was almost exactly the same (± 2 min.). Thus, we report them both as the time of segmentation. Training of the segmentation neural network took 8 hrs. 10 min.

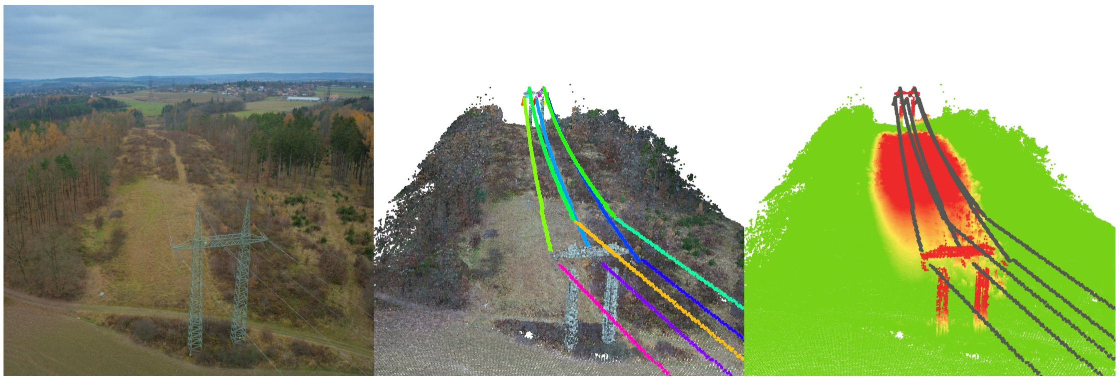
7.4 Evaluation

Unfortunately, we found that evaluating the power line 3D reconstruction accuracy by measuring the ground truth data is immensely difficult. For example, the authors of [48] tried to capture the ground truth data using a stationary Terrestrial Laser Scanner (TLS). However, even then, the results are only indicative due to wind and temperature changes resulting in different sag and location of the wires. Moreover, this option is out of our reach because of the high cost of the TLS scanner and the lack of expertise for operating it. As a result, we mostly rely on a visual evaluation to estimate the power line completeness and relative comparison of the terrain and power line point clouds to measure the reconstruction accuracy.

7.4.1 Evaluating Completeness

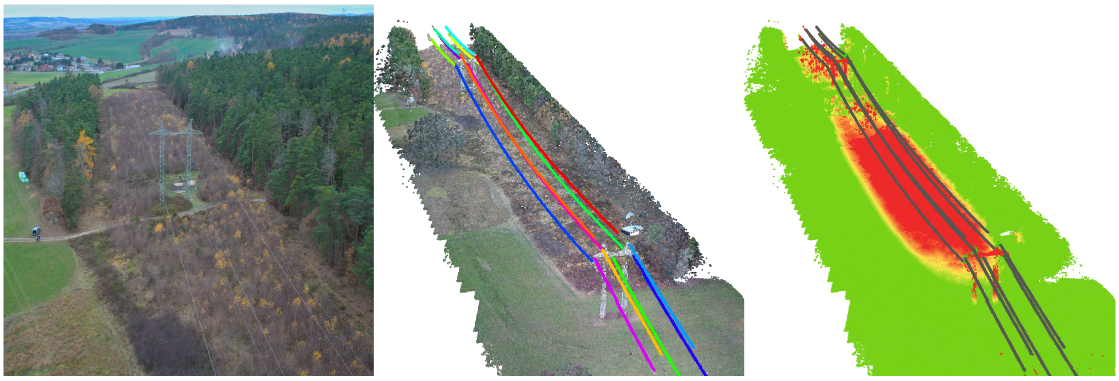
For the reasons mentioned above, we have no precise measure for evaluating what portion of the power line was actually reconstructed. However, we can at least estimate it using the visualization. Figure 7.6 shows the top view of the reconstructed power lines in each dataset for both our proposed segmentation methods.

As expected, the neural network segmentation better handles the difficult background and lighting conditions, especially in dataset 3, where the filter-based segmentation missed a significant portion of the power lines (Figure 7.6c). Overall, the power line reconstruction pipeline with neural network segmentation reconstructed all power lines in all three datasets. From Figure 7.6, we estimate that the overall reconstructed portion of the wires was at least 95 %.



(a) Photo of the location. **(b)** Terrain and power lines. **(c)** Basic visualization of distances.

Figure 7.4 Dataset 2.



(a) Photo of the location. **(b)** Terrain and power lines. **(c)** Basic visualization of distances.

Figure 7.5 Dataset 3.

Mission Details	Dataset 1	Dataset 2	Dataset 3
Number of Images	187	135	197
Ground Sample Distance	1 cm	1.12 cm	1 cm
Flight Height	55 m	62 m	55 m
Flight Velocity	3 m/s	2 m/s	2 m/s
Front Image Overlap	85 %	85 %	85 %
Side Image Overlap	90 %	90 %	90 %
Scanned Area	4.29 ha	3.84 ha	4.61 ha
Scanned Area Length	476 m	384 m	511 m
Scanned Area Width	90 m	100 m	90 m
Capture Date	October 10	November 20	November 20
Capture Time	1:00 p.m.	12:00 p.m.	1:30 p.m.
Duration	8 min.	10 min.	13 min.
Exposure Settings			
Lighting Conditions	Sunny, lot of light	Cloudy, lot of light	Cloudy, less light
ISO	200	640	1000
Shutter Speed	1/250 s	1/100 s	1/100 s
F Number	$f/6.3$	$f/8$	$f/5.6$

Table 7.2 Mission details for each dataset.

Metric [cm]	Dataset 1	Dataset 2	Dataset 3
Neural Network Segmentation			
Average Distance	55.6	7.7	18.4
Median Distance	19.6	4.7	9.7
95-quantile	162	18	50.1
Standard Deviation	58.1	12.8	17.4
Filter-based Segmentation			
Average Distance	90.5	12.8	176
Median Distance	54.7	7.7	50.4
95-quantile	349	55.2	699
Standard Deviation	106	16.1	226

Table 7.3 Evaluation results based on the distances between the reconstructed power line points from the photogrammetric pipeline and the power line reconstruction pipeline (in centimeters).

7.4.2 Evaluating Accuracy

To evaluate the accuracy of the power line reconstruction, we manually extracted the power line points (Figure 7.7a) from the terrain point cloud and compared them with the result of the power line reconstruction pipeline. We can see that OpenDroneMap reconstructed only a small portion of the wires. Some power lines were missing entirely, emphasizing the need for a specialized algorithm for power line reconstruction, such as our proposed method. Nonetheless, it is a sufficient sample for evaluating the accuracy relative to the points created by verified, industry-standard algorithms in the photogrammetric pipeline.

In addition to manually extracting the power lines from the terrain point cloud, we removed a small fraction of the points from areas around the transmission towers, begging and end of the corridor. We reasoned that the power lines were sometimes incomplete in those challenging areas (as shown in Figure 7.6), and the goal of this evaluation is to measure the relative accuracy, not completeness, which we already evaluated in Section 7.4.1.

It is important to say that the terrain point cloud contains numerous sources of errors and cannot be taken as the actual ground truth data, which would be best approximated by precise laser scanning. For the reasons mentioned above, this is difficult and out of scope of this work. The most obvious issue is the lack of enough points along all power lines, with some wires completely missing. Moreover, the power line point cloud is a result of sampling 1000 points from each catenary curve fitted during the reconstruction phase (Section 5.3.3.8), which means that in the longest span, there is one point roughly every 10–20 cm. Consequently, the calculated distances are not orthogonal projections and could be lower if we chose a different sampling rate.

Nevertheless, our proposed method with neural network segmentation achieved outstanding results, with an average distance of 7.7 cm in dataset 2, 18.4 cm in dataset 3, and acceptable results with an average distance of 55 cm in dataset 1. We investigated the higher average error in dataset 1 and found that it was mostly caused by the imprecise fitting of two catenary curves due to many outliers around the transmission tower (Figure 7.9a). The error was mostly in the altitude, so the accuracy of vegetation management from the side was not affected by much. The filter-based segmentation achieved worse results in all cases, but especially in dataset 3 (Figure 7.9b), where it failed in areas with a difficult background (Figure 7.9c).

The detailed distribution of the distances for each dataset and segmentation method is visualized in the histograms in Figure 7.8. Table 7.3 further supplements the histograms with some statistical metrics.

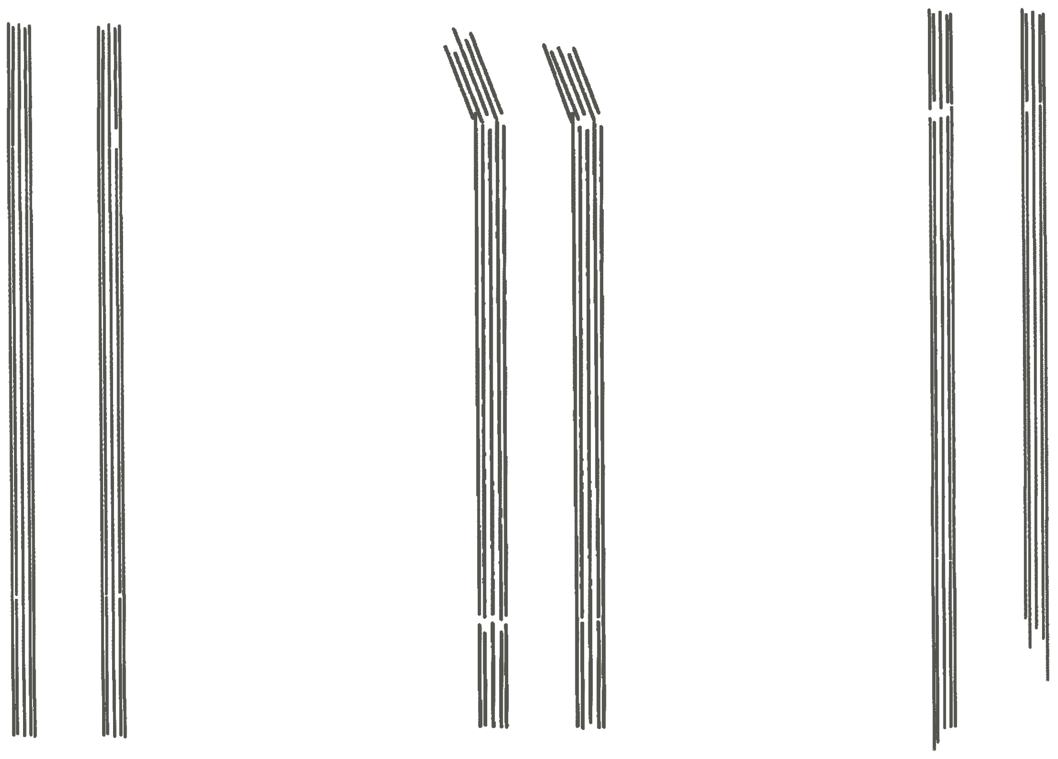
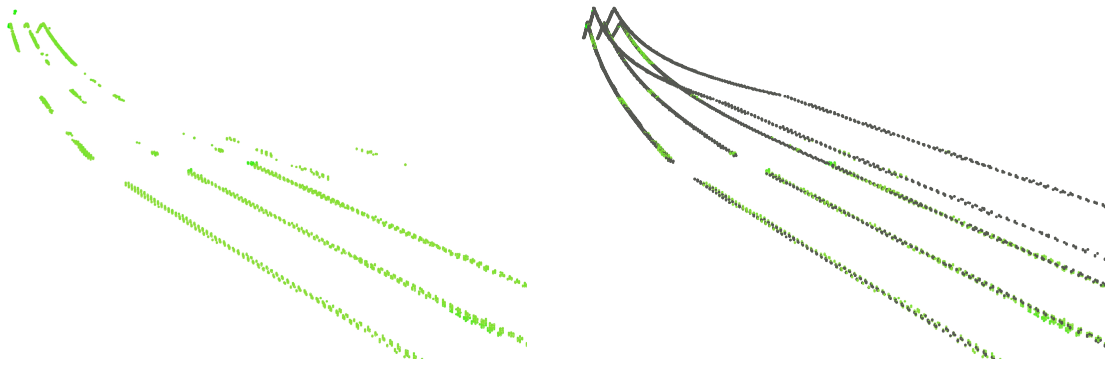
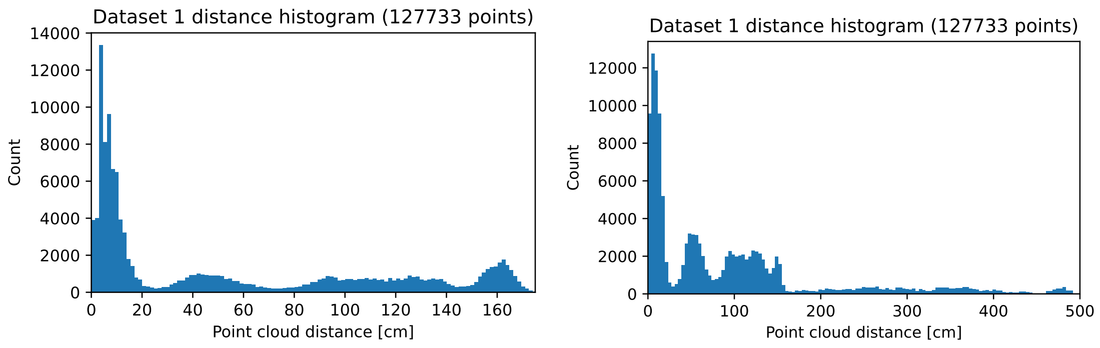

(a) Dataset 1. **(b)** Dataset 2. **(c)** Dataset 3.

Figure 7.6 Top view of the reconstructed power lines. For each dataset, the result of reconstruction with neural network segmentation is on the left, and filter-based segmentation is on the right. We can see that a significant part of the power lines in dataset 3 is missing when using filter-based segmentation.

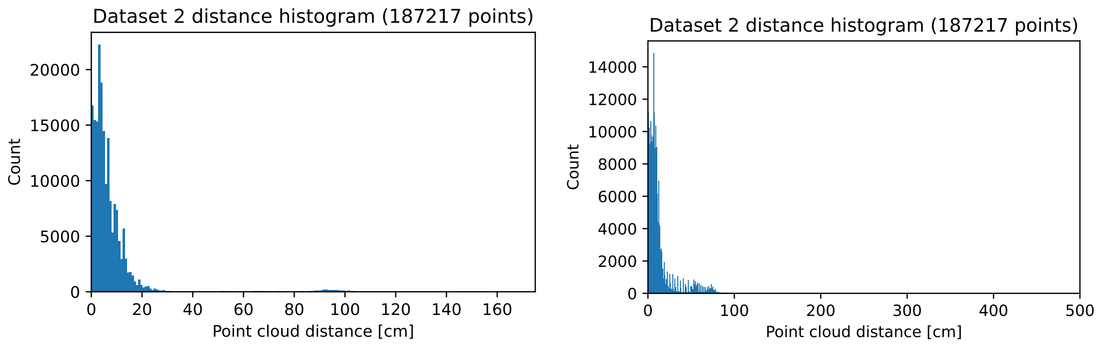

(a) Power line points reconstructed by photogrammetric pipeline.

(b) Comparison of **(a)** and points reconstructed by power line reconstruction pipeline.

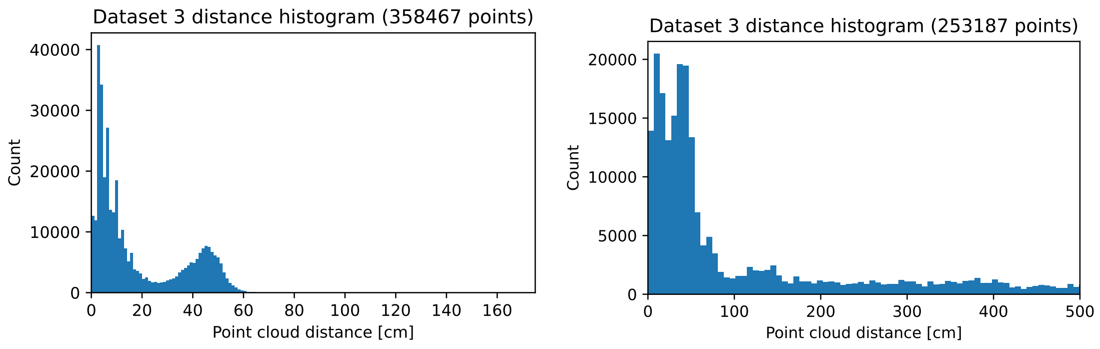
Figure 7.7 Comparison of photogrammetric and power line reconstruction pipeline points. The photogrammetric pipeline points **(a)** were manually segmented from the terrain point cloud.



(a) Dataset 1. Neural net segmentation (left), and filter-based segmentation (right).

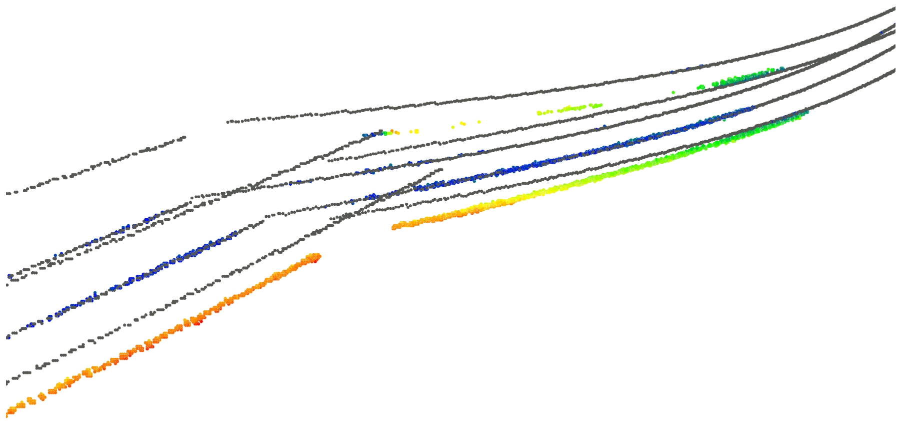


(b) Dataset 2. Neural net segmentation (left), and filter-based segmentation (right).

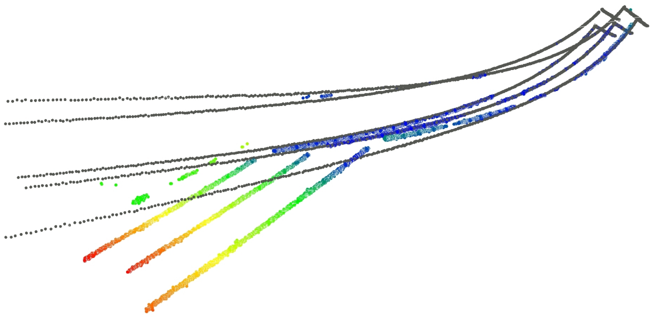


(c) Dataset 3. Neural net segmentation (left), and filter-based segmentation (right).

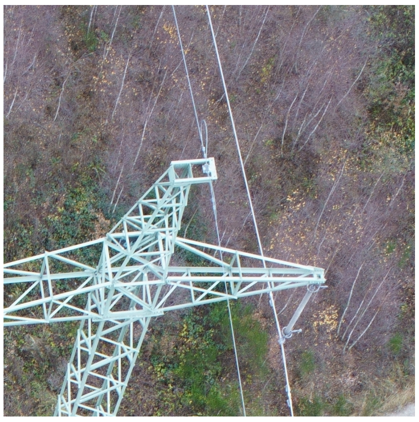
■ **Figure 7.8** Histograms of the distances between the reconstructed power line points from the photogrammetric pipeline and the power line reconstruction pipeline (in centimeters).



(a) In dataset 1, some errors were caused by an imprecise fitting of the catenary curve due to many outliers around the transmission towers. As a result, these errors are mainly in the altitude, so it should not impede the accuracy of vegetation management from the side too much. Moreover, the catenary curve fitting can be disabled, which would give more accurate results, albeit with some noise and outliers that would have to be filtered manually.



(b) The filter-based segmentation was unable to cope with the difficult background and around a transmission tower, causing significant errors in the reconstruction.



(c) Difficult background containing thin grey tree stems in dataset 3.

Figure 7.9 Errors in reconstruction.

7.5 Summary

Overall, through this work, we conducted many flights and experiments. Over the period of five months, we executed around 20 autonomous missions at 5 different locations, collecting more than 2500 images. We created three representative datasets of 220 kV power lines, which prove that our algorithm is robust even under difficult lighting conditions and with a complicated background. The proposed method with the neural network segmentation successfully reconstructed all power lines in all datasets with great coverage of the wires. The filter-based segmentation method resulted in good reconstruction in two out of the three datasets and failed in areas with a difficult background in the third dataset.

7.6 Future Improvements

We are confident from our experiments that the proposed method provides good results for vegetation management around similar power lines to those in our datasets. We expect it will achieve good results even with more than five power lines, as the algorithm has a tolerance for wire overlaps in one of the three images in each parallel group. However, it remains to be seen how the method performs on different layouts or types of wires. Before the method becomes routinely used in practice, it might be necessary to investigate the issue with outliers during catenary curve-fitting that manifested in dataset 1 to improve the vertical accuracy of the algorithm. One possibility would be to enhance the 3D reconstruction step with smart outlier filtering methods other than RANSAC. Additionally, a comparative evaluation with data obtained by laser scanning would help to increase the confidence in the results further.

Another interesting study would be to adapt the algorithm for inspection with a fixed-wing UAV, which can cover larger distances on a single charge. The power lines usually extend over tens or hundreds of kilometers, but a multi-rotor UAV can cover at most one kilometer before it is necessary to swap batteries.

Chapter 8
Conclusion

The goal of this work was to create a robust, fully automated algorithm for power line vegetation management using UAV images. As part of this work, we conducted thorough research of the related literature and state of the art and proposed a new method with several improvements. We built the architecture of our solution on two pipelines, combining the industry standard established software for terrain reconstruction and our proposed pipeline for 3D power line reconstruction. The cornerstones of our method are three steps – power line segmentation, detection, and 3D reconstruction. All three steps were equally difficult and involved many challenges we had to overcome. Lastly, we visualize the distance between the power lines and vegetation to allow for a simple assessment of the risks by a human operator.

In our analysis, we compared the filter-based and neural network segmentation methods and concluded that the latter is more suitable for the purpose of power line 3D reconstruction.

In the end, we were able to fully automate the process of power line vegetation management up to the visualization part. During 3D reconstruction, we attained automated power line matching by imposing constraints on the data acquisition process. Mainly, we require an extra third flight strip for more data redundancy and robustness. Ultimately, the implementation allows for fine control and configurability of the process, as well as fully automatic reconstruction using a single command.

Moreover, we used autonomous missions during data acquisition to capture consistent data with minimal human intervention. We created three datasets of 220 kV power lines going through densely vegetated areas. The proposed algorithm was able to successfully reconstruct all power lines in all three datasets with great coverage of the wires.

Bibliography

1. *The Thames Barrier* [GOV.UK] [online]. 2021 [visited on 2021-12-18]. Available from: `https://www.gov.uk/guidance/the-thames-barrier`.
2. ROBERTSON, Adi. *Investigators confirm that PG&E power lines started the deadly Camp Fire* [The Verge] [online]. 2019-05-15 [visited on 2021-12-18]. Available from: `https://www.theverge.com/2019/5/15/18626819/cal-fire-pacific-gas-and-electric-camp-fire-power-lines-cause`.
3. GONZALES, Richard. PG&E Announces $13.5 Billion Settlement Of Claims Linked To California Wildfires. *NPR* [online]. 2019 [visited on 2021-12-18]. Available from: `https://www.npr.org/2019/12/07/785775074/pg-e-announces-13-5-billion-settlement-of-claims-linked-to-california-wildfires`.
4. AION CS, s.r.o. *458/2000 Sb. Energetický zákon* [Zákony pro lidi] [online] [visited on 2021-11-11]. Available from: `https://www.zakonyprolidi.cz/cs/2000-458`.
5. PNE 33 3300. *Navrhování a stavba venkovních vedení nad AC 45 kV*. 2016. ČEPS, ČEZ Distribuce, PRE Distribuce, E.ON Czech.
6. CHAN, Ting; LICHTI, Derek. 3D catenary curve fitting for geometric calibration. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. 2012, vol. XXXVIII-5/W12, pp. 259–264. Available from DOI: `10.5194/isprsarchives-XXXVIII-5-W12-259-2011`.
7. SONKA, Milan; HLAVAC, Vaclav; BOYLE, Roger. *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 2014. ISBN 9781285981444. Google-Books-ID: QePKAgAAQBAJ.
8. PRINCE, Simon J. D. *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, 2012. ISBN 9781107011793. Google-Books-ID: PmrICLzHutgC.
9. KLETTE, Reinhard. *Concise Computer Vision: An Introduction into Theory and Algorithms*. Springer London, 2014. ISBN 9781447163190. Google-Books-ID: EgSRngEACAAJ.
10. CHENG, Bowen; COLLINS, Maxwell D.; ZHU, Yukun; LIU, Ting; HUANG, Thomas S.; ADAM, Hartwig; CHEN, Liang-Chieh. Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation. *arXiv:1911.10194 [cs]* [online]. 2020 [visited on 2021-11-17]. Available from arXiv: `1911.10194`.
11. ABDELFATTAH, Rabab; WANG, Xiaofeng; WANG, Song. TTPLA: An Aerial-Image Dataset for Detection and Segmentation of Transmission Towers and Power Lines. In: *Proceedings of the Asian Conference on Computer Vision (ACCV)* [online]. 2020 [visited on 2021-11-17]. Available from: `https://openaccess.thecvf.com/content/ACCV2020/html/Abdelfattah_TTPLA_An_Aerial-Image_Dataset_for_Detection_and_Segmentation_of_Transmission_ACCV_2020_paper.html`.

12. CHEN, Liang-Chieh; PAPANDREOU, George; KOKKINOS, Iasonas; MURPHY, Kevin; YUILLE, Alan L. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2018, vol. 40, no. 4, pp. 834–848. ISSN 1939-3539. Available from DOI: `10.1109/TPAMI.2017.2699184`.

13. KIM, ZuWhan. Robust Lane Detection and Tracking in Challenging Scenarios. *IEEE Transactions on Intelligent Transportation Systems*. 2008, vol. 9, no. 1, pp. 16–26. ISSN 1558-0016. Available from DOI: `10.1109/TITS.2007.908582`.

14. CHAUDHURY, Krishnendu; DIVERDI, Stephen; IOFFE, Sergey. Auto-rectification of user photos. In: *2014 IEEE International Conference on Image Processing (ICIP)*. 2014, pp. 3479–3483. Available from DOI: `10.1109/ICIP.2014.7025706`. ISSN: 2381-8549.

15. KAEHLER, Adrian; BRADSKI, Gary R. *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, 2017. ISBN 9781491937990. Google-Books-ID: c7UXswEACAAJ.

16. *Hough Line Transform — OpenCV-Python Tutorials 1 documentation* [online] [visited on 2021-11-21]. Available from: `https://opencv24-python-tutorials.readthedocs.io/en/stable/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html`.

17. FÖRSTNER, Wolfgang; WROBEL, Bernhard P. *Photogrammetric Computer Vision*. Springer International Publishing Switzerland, 2016. Geometry and Computing. ISBN 978-3-319-11550-4. Available also from: `https://doi.org/10.1007/978-3-319-11550-4`.

18. O'CONNOR, James; SMITH, Mike J; JAMES, Mike R. Cameras and settings for aerial surveys in the geosciences: Optimising image data. *Progress in Physical Geography: Earth and Environment* [online]. 2017, vol. 41, no. 3, pp. 325–344 [visited on 2021-11-11]. ISSN 0309-1333. Available from DOI: `10.1177/0309133317703092`.

19. STACHNISS, Cyrill. *Lecture: Camera Parameters - Extrinsics and Intrinsics* [Lecture]. 2020 [visited on 2021-11-13]. Available from: `https://www.youtube.com/watch?v=uHApDqH-8UE&t=592s&ab_channel=CyrillStachniss`.

20. FRISIUS, Rainer Gemma. *De radio astronomico et geometrico liber* [online]. Ap. Gul Cavellat, 1545 [visited on 2021-11-14]. Available from: `https://sbc.org.pl/dlibra/publication/12435/edition/34981/content`.

21. WEISSTEIN, Eric W. *Rodrigues' Rotation Formula* [online] [visited on 2021-11-14]. Available from: `https://mathworld.wolfram.com/RodriguesRotationFormula.html`.

22. ADORJAN, Matthias. *OpenSfM : a collaborative Structure-from-Motion System* [online]. 2016 [visited on 2021-11-14]. Available from: `https://repositum.tuwien.at/handle/20.500.12708/5906`. Thesis. Wien.

23. HARTLEY, Richard I.; ZISSERMAN, Andrew. *Multiple View Geometry in Computer Vision* [online]. Second. Cambridge University Press, 2004 [visited on 2021-11-15]. ISBN 0521540518. Available from: `https://doi.org/10.1108/k.2001.30.9_10.1333.2`.

24. BIANCO, Simone; CIOCCA, Gianluigi; MARELLI, Davide. Evaluating the Performance of Structure from Motion Pipelines. *Journal of Imaging* [online]. 2018, vol. 4, no. 8, p. 98 [visited on 2021-11-14]. Available from DOI: `10.3390/jimaging4080098`.

25. LOWE, David G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* [online]. 2004, vol. 60, no. 2, pp. 91–110 [visited on 2021-11-14]. ISSN 1573-1405. Available from DOI: `10.1023/B:VISI.0000029664.99615.94`.

26. SEITZ, S.M.; CURLESS, B.; DIEBEL, J.; SCHARSTEIN, D.; SZELISKI, R. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. 2006, vol. 1, pp. 519–528. Available from DOI: `10.1109/CVPR.2006.19`. ISSN: 1063-6919.

27. KRAETZIG, Nikita Marwaha. *Everything you need to know about Digital Elevation Models (DEMs), Digital Surface Models (DSMs), and Digital Terrain Models (DTMs)* [UP42 Official Website] [online] [visited on 2021-11-15]. Available from: `https://up42.com/blog/tech/everything-you-need-to-know-about-digital-elevation-models-dem-digital`.

28. *Clustering* [scikit-learn documentation] [online] [visited on 2021-11-22]. Available from: `https://scikit-learn.org/stable/modules/clustering.html`.

29. *The Iris Dataset* [scikit-learn] [online] [visited on 2021-11-22]. Available from: `https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html`.

30. MATIKAINEN, Leena; LEHTOMÄKI, Matti; AHOKAS, Eero; HYYPPÄ, Juha; KARJALAINEN, Mika; JAAKKOLA, Anttoni; KUKKO, Antero; HEINONEN, Tero. Remote sensing methods for power line corridor surveys. *ISPRS Journal of Photogrammetry and Remote Sensing* [online]. 2016, vol. 119, pp. 10–31 [visited on 2021-11-11]. ISSN 0924-2716. Available from DOI: `10.1016/j.isprsjprs.2016.04.011`.

31. LEHTOMÄKI, Matti; KUKKO, Antero; MATIKAINEN, Leena; HYYPPÄ, Juha; KAARTINEN, Harri; JAAKKOLA, Anttoni. Power line mapping technique using all-terrain mobile laser scanning. *Automation in Construction* [online]. 2019, vol. 105, p. 102802 [visited on 2021-11-25]. ISSN 0926-5805. Available from DOI: `10.1016/j.autcon.2019.03.023`.

32. ZHANG, Jixian; HUANG, Guoman; LIU, Jiping. SAR remote sensing monitoring of the Yushu Earthquake disaster situation and the information service system [online]. 2010 [visited on 2021-11-23]. Available from: `http://en.cnki.com.cn/Article_en/CJFDTOTAL-YGXB201005017.htm`.

33. WHITWORTH, C. C.; DULLER, A. W. G.; JONES, D. I.; EARP, G. K. Aerial video inspection of overhead power lines. *Power Engineering Journal* [online]. 2001, vol. 15, no. 1, pp. 25–32 [visited on 2021-11-25]. ISSN 0950-3366. Available from DOI: `10.1049/pe:20010103`.

34. CHEN, Chi; YANG, Bisheng; SONG, Shuang; PENG, Xiangyang; HUANG, Ronggang. Automatic Clearance Anomaly Detection for Transmission Line Corridors Utilizing UAV-Borne LIDAR Data. *Remote Sensing* [online]. 2018, vol. 10, no. 4, p. 613 [visited on 2021-11-11]. Available from DOI: `10.3390/rs10040613`.

35. AZEVEDO, Fábio; DIAS, André; ALMEIDA, José; OLIVEIRA, Alexandre; FERREIRA, André; SANTOS, Tiago; MARTINS, Alfredo; SILVA, Eduardo. LiDAR-Based Real-Time Detection and Modeling of Power Lines for Unmanned Aerial Vehicles. *Sensors (Basel, Switzerland)* [online]. 2019, vol. 19, no. 8, p. 1812 [visited on 2021-11-11]. ISSN 1424-8220. Available from DOI: `10.3390/s19081812`.

36. ROUTESCENE. *Vegetation management - ensure quick and detailed survey across sectors* [Routescene] [online] [visited on 2021-11-11]. Available from: `https://www.routescene.com/applications/vegetation-management/`.

37. TORRES, Gabriel. *Drone photogrammetry vs. LIDAR: what sensor to choose for a given application* [Wingtra] [online]. 2021-03-16 [visited on 2021-11-11]. Available from: `https://wingtra.com/drone-photogrammetry-vs-lidar/`.

38. JÓŹKÓW, Grzegorz; JAGT, B.; TOTH, Charles. Experiments with UAS Imagery for Automatic Modeling of Power Line 3D Geometry. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. 2015, vol. XL-1/W4, pp. 403–409. Available from DOI: `10.5194/isprsarchives-XL-1-W4-403-2015`.

39. RAJEEV M BHUJADE, Adithya V.; RAJEEV M BHUJADE, Adithya V. Detection of Power-Lines in Complex Natural Surroundings. *CS & IT Conference Proceedings* [online]. 2013, vol. 3, no. 9 [visited on 2021-11-11]. Available from: `http://csitcp.com/abstract/39csit10`.

40. YETGIN, Ömer Emre; BENLIGIRAY, Burak; GEREK, Ömer Nezih. Power Line Recognition From Aerial Images With Deep Learning. *IEEE Transactions on Aerospace and Electronic Systems*. 2019, vol. 55, no. 5, pp. 2241–2252. ISSN 1557-9603. Available from DOI: `10.1109/TAES.2018.2883879`.

41. LI, Zhengrong; LIU, Yuee; WALKER, Rodney; HAYWARD, Ross; ZHANG, Jinglan. Towards automatic power line detection for a UAV surveillance system using pulse coupled neural filter and an improved Hough transform. *Machine Vision and Applications* [online]. 2010, vol. 21, no. 5, pp. 677–686 [visited on 2021-11-11]. ISSN 1432-1769. Available from DOI: `10.1007/s00138-009-0206-y`.

42. ZHANG, Heng; YANG, Wen; YU, Huai; ZHANG, Haijian; XIA, Gui-Song. Detecting Power Lines in UAV Images with Convolutional Features and Structured Constraints. *Remote Sensing* [online]. 2019, vol. 11, no. 11, p. 1342 [visited on 2021-11-11]. Available from DOI: `10.3390/rs11111342`.

43. LARRAURI, Juan I.; SORROSAL, Gorka; GONZÁLEZ, Mikel. Automatic system for overhead power line inspection using an Unmanned Aerial Vehicle — RELIFO project. In: *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2013, pp. 244–252. Available from DOI: `10.1109/ICUAS.2013.6564696`.

44. ZHANG, Yong; YUAN, Xiuxiao; LI, Wenzhuo; CHEN, Shiyu. Automatic Power Line Inspection Using UAV Images. *Remote Sensing* [online]. 2017, vol. 9, no. 8, p. 824 [visited on 2021-11-11]. Available from DOI: `10.3390/rs9080824`.

45. MAURER, Michael; HOFER, Manuel; FRAUNDORFER, Friedrich; BISCHOF, Horst. AUTOMATED INSPECTION OF POWER LINE CORRIDORS TO MEASURE VEGETATION UNDERCUT USING UAV-BASED IMAGES. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*. 2017, vol. IV-2/W3, pp. 33–40. Available from DOI: `10.5194/isprs-annals-IV-2-W3-33-2017`.

46. HOFER, Manuel; MAURER, Michael; BISCHOF, Horst. Efficient 3D scene abstraction using line segments. *Computer Vision and Image Understanding* [online]. 2017, vol. 157, pp. 167–178 [visited on 2021-11-11]. ISSN 1077-3142. Available from DOI: `10.1016/j.cviu.2016.03.017`.

47. OH, Jaehong; LEE, Changno. 3D Power Line Extraction from Multiple Aerial Images. *Sensors* [online]. 2017, vol. 17, no. 10, p. 2244 [visited on 2021-11-11]. Available from DOI: `10.3390/s17102244`.

48. PASTUCHA, Elzbieta; PUNIACH, Edyta; ŚCISŁOWICZ, Agnieszka; ĆWIĄKAŁA, Paweł; NIEWIEM, Witold; WIĄCEK, Paweł. 3D Reconstruction of Power Lines Using UAV Images to Monitor Corridor Clearance. *Remote Sensing*. 2020, vol. 12, p. 3698. Available from DOI: `10.3390/rs12223698`.

49. ZHANG, Z. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2000, vol. 22, no. 11, pp. 1330–1334. ISSN 1939-3539. Available from DOI: `10.1109/34.888718`.

50. PNE 34 7509. *Holé vodiče pro venkovní vedení ze soustředně slaněných kruhových drátů* [online]. 2009 [visited on 2021-12-18]. Available from: `http://www.mojeenergie.cz/cz/pne-34-7509-2009`.

51. *Tutorials – OpenDroneMap 2.6.7 documentation* [online]. 2020 [visited on 2021-11-25]. Available from: `https://docs.opendronemap.org/tutorials/`.

52. *Do RTK/PPK drones give you better results than GCPs?* [Pix4D] [online]. 2017-08-27 [visited on 2021-12-22]. Available from: `https://www.pix4d.com/blog/rtk-ppk-drones-gcp-comparison`.

53. *RealityCapture* [online]. Epic Games Slovakia s.r.o., [n.d.]. Version 1.2.0.17385 [visited on 2021-11-28]. Available from: `https://www.capturingreality.com/`.

54. *Agisoft Metashape* [online]. Agisoft, [n.d.]. Version 1.7.5 [visited on 2021-11-28]. Available from: `https://www.agisoft.com/`.

55. *PIX4Dmapper* [online]. Pix4D S.A., [n.d.]. Version 4.7.5 [visited on 2021-11-28]. Available from: `https://www.pix4d.com/product/pix4dmapper-photogrammetry-software`.

56. *Meshroom* [online]. AliceVision, [n.d.]. Version 2021.1.0 [visited on 2021-11-28]. Available from: `https://alicevision.org/#meshroom`.

57. *OpenDroneMap (NodeODM)* [online]. [N.d.]. Version 2.2.0 [visited on 2021-11-28]. Available from: `https://www.opendronemap.org/`.

58. *MicMac* [online]. IGN (French National Geographic Institute), ENSG (French national school for geographic sciences), [n.d.]. V1.0.beta14 [visited on 2021-11-28]. Available from: `https://micmac.ensg.eu/index.php/Accueil`.

59. AATI, Saif; RUPNIK, Ewelina; NEJIM, Samir. Comparative study of photogrammetry software in industrial field. *Revue Francaise de Photogrammetrie et de Teledetection.* 2020, vol. 1, pp. 37–48. Available from DOI: `10.52638/rfpt.2019.439`.

60. ALIDOOST, F.; AREFI, H. Comparison of UAS-based photogrammetry software for 3D point cloud generation: A survey over a historical site. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* [online]. 2017, vol. IV-4/W4, pp. 55–61 [visited on 2021-11-28]. ISSN 21949050. Available from DOI: `10.5194/isprs-annals-iv-4-w4-55-2017`.

61. KINGSLAND, Kaitlyn. Comparative analysis of digital photogrammetry software for cultural heritage. *Digital Applications in Archaeology and Cultural Heritage* [online]. 2020, vol. 18, e00157 [visited on 2021-11-28]. ISSN 2212-0548. Available from DOI: `10.1016/j.daach.2020.e00157`.

62. *Photogrammetry: Step-by-Step Guide and Software Comparison* [Formlabs] [online] [visited on 2021-11-28]. Available from: `https://formlabs.com/blog/photogrammetry-guide-and-software-comparison/`.

63. ĐURIC, Isidora. Comparative Analysis of Open-Source and Commercial Photogrammetry Software for Cultural Heritage. In: *Stojakovic, V and Tepavcevic, B (eds.), Towards a new, configurable architecture - Proceedings of the 39th eCAADe Conference - Volume 2, University of Novi Sad, Novi Sad, Serbia, 8-10 September 2021, pp. 243-252* [online]. CUMINCAD, 2021 [visited on 2021-11-28]. Available from: `http://papers.cumincad.org/cgi-bin/works/2015%20+dave=2:/Show?ecaade2021_117`.

64. *Powerlines detection : Helpdesk Portal* [online] [visited on 2021-11-29]. Available from: `https://agisoft.freshdesk.com/support/solutions/articles/31000161295-powerlines-detection`.

65. GILLESPIE, Alan R; KAHLE, Anne B; WALKER, Richard E. Color enhancement of highly correlated images. I. Decorrelation and HSI contrast stretches. *Remote Sensing of Environment* [online]. 1986, vol. 20, no. 3, pp. 209–235 [visited on 2021-12-01]. ISSN 0034-4257. Available from DOI: `10.1016/0034-4257(86)90044-1`.

66. CHEN, Liang-Chieh; ZHU, Yukun; PAPANDREOU, George; SCHROFF, Florian; ADAM, Hartwig. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. *arXiv:1802.02611 [cs]* [online]. 2018 [visited on 2021-12-01]. Available from arXiv: `1802.02611`.

67. *MMSegmenation* [online]. OpenMMLab, 2021. Version 0.19.0 [visited on 2021-12-01]. Available from: `https://mmsegmentation.readthedocs.io/en/latest/`.

68. CORDTS, Marius; OMRAN, Mohamed; RAMOS, Sebastian; REHFELD, Timo; ROTH, Stefan; ENZWEILER, Markus; BENENSON, Rodrigo; FRANKE, Uwe; SCHIELE, Bernt. The Cityscapes Dataset for Semantic Urban Scene Understanding. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* [online]. 2016, pp. 3213–3223 [visited on 2021-12-01]. Available from: `https://openaccess.thecvf.com/content_cvpr_2016/html/Cordts_The_Cityscapes_Dataset_CVPR_2016_paper.html`.

69. *CloudCompare* [online]. [N.d.]. 2.11.3-4 [visited on 2021-12-12]. Available from: `http://www.cloudcompare.org/`.

70. *Poetry* [online]. [N.d.]. Version 1.1.11 [visited on 2021-12-12]. Available from: `https://python-poetry.org/`.

71. *Poe the Poet* [online]. 2021. Version 0.10.0 [visited on 2021-12-12]. Available from: `https://github.com/nat-n/poethepoet`. original-date: 2020-05-28T21:56:22Z.

72. *OpenCV* [online]. OpenCV, 2021. Version 4.5.2.54 [visited on 2021-12-12]. Available from: `https://opencv.org/`. original-date: 2012-07-19T09:40:17Z.

73. *scikit-learn* [online]. 2021. Version 1.0.1 [visited on 2021-12-12]. Available from: `https://scikit-learn.org/`. original-date: 2010-08-17T09:43:38Z.

74. *scikit-image* [online]. 2021. Version 0.18.3 [visited on 2021-12-12]. Available from: `https://scikit-image.org/`. original-date: 2011-07-07T22:07:20Z.

75. *Shapely* [online]. 2021. Version 1.8.0 [visited on 2021-12-12]. Available from: `https://github.com/shapely/shapely`. original-date: 2011-12-31T19:43:11Z.

76. *UgCS* [online]. SPH Engineering, 2021 [visited on 2021-12-15]. Available from: `https://www.ugcs.com`.

77. *OpenStreetMap* [online]. 2021 [visited on 2021-12-15]. Available from: `https://osm.org/`.

SD Card Contents

`readme.txt`	a brief description of the SD card contents
`guide.pdf`	software installation and user guide
`src`	directory of source codes
`app`	application source code in Python
`thesis`	thesis source code in LaTeX
`datasets`	directory of the captured datasets
`dataset1…3`	images in raw format
`example-output`	results of processing datasets 1–3
`thesis.pdf`	the thesis text in PDF