UNIVERSITY OF CAPE TOWN

Visual Localisation of Electricity Pylons for Power Line Inspection

by

Emmanuel Yahi Ali

Supervised by

Fred Nicolls

A thesis submitted in fulfillment for the
degree of Doctor of Philosophy

in

Electrical Engineering
Engineering and Built Environment

April 2023

Declaration of Authorship

I, Emmanuel Yahi Ali, declare that this thesis titled, 'Visual Localisation of Electricity Pylons for Power Line Inspection' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Abstract

Inspection of power infrastructure is a regular maintenance event. To date the inspection process has mostly been done manually, but there is growing interest in automating the process.

The automation of the inspection process will require an accurate means for the localisation of the power infrastructure components. In this research, we studied the visual localisation of a pylon. The pylon is the most prominent component of the power infrastructure, and can provide a context for the inspection of the other components.

Point-based descriptors tend to perform poorly on textureless objects such as pylons, therefore we explored the localisation using convolutional neural networks and geometric constraints.

The crossings of the pylon, or vertices, are salient points on the pylon. These vertices aid with recognition and pose estimation of the pylon. We were successfully able to use a convolutional neural network for the detection of the vertices.

A model-based technique, geometric hashing, was used to establish the correspondence between the stored pylon model and the scene object. We showed the effectiveness of the method as a voting technique to determine the pose estimation from a single image. In a localisation framework, the method serves as the initialization of the tracking process.

We were able to incorporate an extended Kalman filter for subsequent incremental tracking of the camera relative to the pylon. Also, we demonstrated an alternative tracking using heatmap details from the vertex detection.

We successfully demonstrated the proposed algorithms and evaluated their effectiveness using a model pylon we built in the laboratory. Furthermore, we revalidated the results on a real world outdoor electricity pylon. Our experiments illustrate that model-based techniques can be deployed as part of the navigation aspect of a robot.

Acknowledgements

I want to thank my supervisor Fred Nicolls for his guidance and support during the course of this research. He provided insight that has made this work possible.

Also, I want to thank the Postgraduate Funding Office of the University of Cape Town and the Petroleum Development Trust Fund, Nigeria for providing funding for the research.

Finally, I want to thank all my friends and family for supporting and encouraging me throughout the research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Electricity has become a major component of our society and we use it for virtually all aspects of our lives from manufacturing to household use. The constant supply of electricity comes at an expense. Like most things on earth, with constant usage there is fatigue, wear and tear, and break down. Therefore, routine checks on the facilities have become a necessity.

Power companies conduct routine maintenance and surveillance on their power infrastructure. Power infrastructure consists of electricity pylons, conductors, dampers, splices and insulators. The pylons serve as holding points for the passage of the conductor and are placed at regular intervals. The insulators are on the pylon to protect the conductors. While carrying out inspection, attention is paid to the components to check for faults, abnormalities, damage and missing parts.

The pylon is the most prominent component of the transmission line and localising it provides a context for locating other components that need to be inspected.

This research involves the study of visual perception techniques that can be used to guide a robot to function autonomously. Inspection is a two-way process involving localisation of an object and detection/fault analysis of the object. We focus on the object localisation in this research. We investigate a model representation of the pylon and a search-based strategy for localisation of the pylon. The localisation provides a six degree-of-freedom pose estimate of the camera relative to the pylon, which will guide component-based detection and fault analysis with pattern recognition methods.

Camera pose estimation of a pylon involves estimating the position and orientation of a pylon relative to the camera. Wherever the term pose estimation or localisation is used in this dissertation it refers to camera pose estimation unless clearly stated otherwise.

The pylon is a "wiry" object [2] and pose estimation for wiry objects is a difficult task. The difficulty arises due to lack of appearance information, the changing surroundings of any feature point on the structure, and the possibility for "false" image features caused by self-occlusion of the structure at different depths.

Therefore, we try to split the tasks involved in achieving pose estimation as follows:

- Feature extraction of the pylon as the feature representation.
- Image registration of the pylon to establish correspondence.
- Tracking of the pylon for incremental movement of the camera.

To achieve pose estimation for an object, we have to extract the image features and specify a way of representing it. This representation could use a combination of points, edges, corners and lines. In this work we observe that the structure of the pylon is made up of struts and bars which are straight lines. As much as the lines can represent the structure, most of the parts of the pylon intersect at points which can be called vertices. The vertices are distinct features on the pylon.

1.1 Robotic navigation

Robotic navigation is the act of monitoring and controlling the movement of the robot from one place to another. There are three components to robotic navigation namely localisation, mapping, and path planning and control. Localisation is the determination of the position of a robot, which tells the robot where it is located in the world. Mapping is the representation of the whole environment and storing it in a consistent way that enables localisation. Localisation and mapping are intertwined and mostly handled together as simultaneous localisation and mapping (SLAM). Path planning and control is the design of a path for the robot to follow to its target location.

Robots such as aerial robots or line inspection robots all have localisation, mapping, and path planning and control incorporated into them. There are various implementations of these components based on the type of sensors. Some examples of these sensors are laser range finders and cameras. Measurements obtained from the sensors are mostly noisy so Bayesian frameworks are needed to track the uncertainty. The preferred sensors for this work are cameras, which provide 2D information of the scene. However, the world is composed of 3D information, thus image information will have to be matched and triangulated between identical points in different scenes. The process described is called Structure from Motion (SfM) and visual SLAM when applied to image sequences.

In an outdoor environment, many structures and landmarks have stayed the same for years. These structures are fixed and their geometrical information is known. Such geometrical information is given in the form of 3D point clouds, 3D lines, dense volumetric reconstruction, mesh based, 2.5D digital surface models and 3D CAD measurements. The reconstruction of these structures have been done to provide rich 3D information of the structures. In other cases information about the structures is obtained from the owners and manufacturers. Publicly available data for such landmarks is available from NASA shuttle radar topography mission, Google street view or satellite imagery.

SLAM is designed to work in all environments. However, outdoor landmarks and environments generally have poor texture for successful working of SLAM algorithms. In model-based techniques, the knowledge obtained from existing structures provides rich information that can be used to build successful localisation techniques.

In this research we explore the use of geometrical information to design model-based localisation techniques. Structures such as the pylon are fixed in an environment and their geometrical information does not change. Thus they can serve as natural landmarks. In the vicinity of the pylon we can use a model-based localisation for the navigation of the robot instead of the visual SLAM techniques, which fail in such a scenario.

1.2 Problem definition

Current literature on the inspection of power infrastructure mostly deals with object detection as a bounding box. These approaches deal with the problem of identification and detection of components and faults but are insufficient in providing full autonomy for robots. To achieve accurate and autonomous inspection, we need to precisely locate the object with position and orientation using a sensor such as a camera.

Keypoint detection is about finding the salient points in an image or object which enables correspondence to be made. This is mostly a prerequisite for the overall camera pose estimation. We are concerned with finding the unique points on a pylon. These unique points, or vertices, will help represent a pylon and match the vertices as a way of identifying a pylon. Thus we are addressing the question of whether there are distinguishing features on the pylon that will enable us identify a pylon. Can a search-based algorithm and the distinguishing features be used to solve six degree-of-freedom pose estimation of the camera relative to the pylon? Finally we ask can we track the incremental movement of the camera while focused on the pylon?

When reconstructing the pylon Hofer et al. [3] attempted using the scale invariant feature transform (SIFT) but could only match based on the textured background instead of the features on the pylon. Morarjee [4] also tried using SIFT to establish a match but was not successful. Keypoints are points of grey-level discontinuities and are described by the appearance of patches of pixels around the keypoint. For the pylon, the surrounding patches around the vertex are the background. Therefore point-based descriptor techniques such as the SIFT, Speeded up robust features (SURF), and Oriented FAST and rotated brief (ORB) do not work for the overall goal of localisation of the pylon.

This research is an attempt to solve the localisation of the pylon using model-based techniques instead of point-based descriptor techniques.

1.3 Research Overview

The thesis investigates the use of computer vision to provide perception for a robot by obtaining the localisation of a pylon on power infrastructure to aid the

inspection process.

The experimental setup used in this work is shown in Figure 1.1. We build the setup in the laboratory to allow us test the ideas presented in this dissertation. Additionally, we revalidate our algorithms with a real world outdoor data. The

FIGURE 1.1: An image of the pylon used for the experimentation of ideas.

research setup provides us the opportunity to generate data, conduct experiments and build localisation algorithms.

The research involves obtaining the localisation of a camera relative to a pylon. The localisation deals with finding the position and orientation of the camera relative to the pylon, which is the six degree-of-freedom pose estimation problem. There are various methods to determine the solution to the problem. With point-based descriptors camera pose estimation is well understood, although it is difficult

on weakly textured objects such as pylons. Therefore, we decided to use model-based techniques for the solution to the localisation process. This requires us to have the model of the pylon stored. The model has the 3D coordinates measured for each vertex on the pylon. The vertices are the crossings on the pylon, and we identified them as salient points that can be detected and tracked. To solve for the localisation, we will need to obtain the correspondence between the vertices in the model and the image scene. The vertices will also need to be detected.

The summary of the work made during the course of this research includes:

- Vertex detection: We show how to do feature detection of vertices of a pylon using a convolutional neural network. The convolutional neural network learns to detect the vertices of the pylon, which are the salient points on the pylon.

  The vertices are features on the pylon that are used for search-based detection and as measurements for the trackers. To find the features, we use a convolutional neural network. Due to the nature of the structure of the pylon, the features are weakly textured and appearance-based feature methods tend not to work. We train a convolutional neural network to learn to detect the locations of vertices on the pylon. The neural network is designed to detect keypoints similar to the keypoints for human pose estimation.

  To detect the vertices, we attempted feature-based techniques, and a Hough voting technique to find the vertices, which failed when incorporated for the pose estimation of the camera relative to the pylon. Therefore, we decided to try learning techniques for the vertices using a convolutional neural network.

  The concept of transfer learning is used to build the network. Transfer learning is about taking a model pretrained on a large dataset like the ImageNet and fine-tuning it for a particular task. For this work we use a pretrained model that has been successfully used in other tasks.

  The architecture of the neural network is built for the purpose of vertex detection with an inference unit for prediction of the vertices. The network is made of different layers of filters, which involves downsampling and upsampling of the feature map. The upsampling unit has the number of output channels equivalent to the number of vertices, until the outputs are concatenated with the inference unit.

Inference is about the prediction from the network that achieves the goal of vertex detection. The inference takes the output from the trained model, and then applies maximisation of the feature map from a channel. Afterwards, all the heatmaps from each channel are concatenated into a single heatmap.

A dataset and a heatmap is created to train and evaluate the model. The data is generated from different views and background of the pylon. The vertices of the pylon are labelled uniquely. Two sides of the pylon are identical so we have to be careful to distinguish the faces. Ground truth heatmaps are made for each of the vertices.

The data is comprised of images split into training and testing datasets. After processing the data, it is applied to the network. Parameters such as the optimizer, the loss function, the learning rate, and the activation function are initialised and the model is trained.

We check the accuracy rate for the network by comparing the Euclidean distance between the ground truth vertices and the predicted vertices.

Figure 1.2 shows an example of a pylon with the vertices detected on it and the hand annotated label.

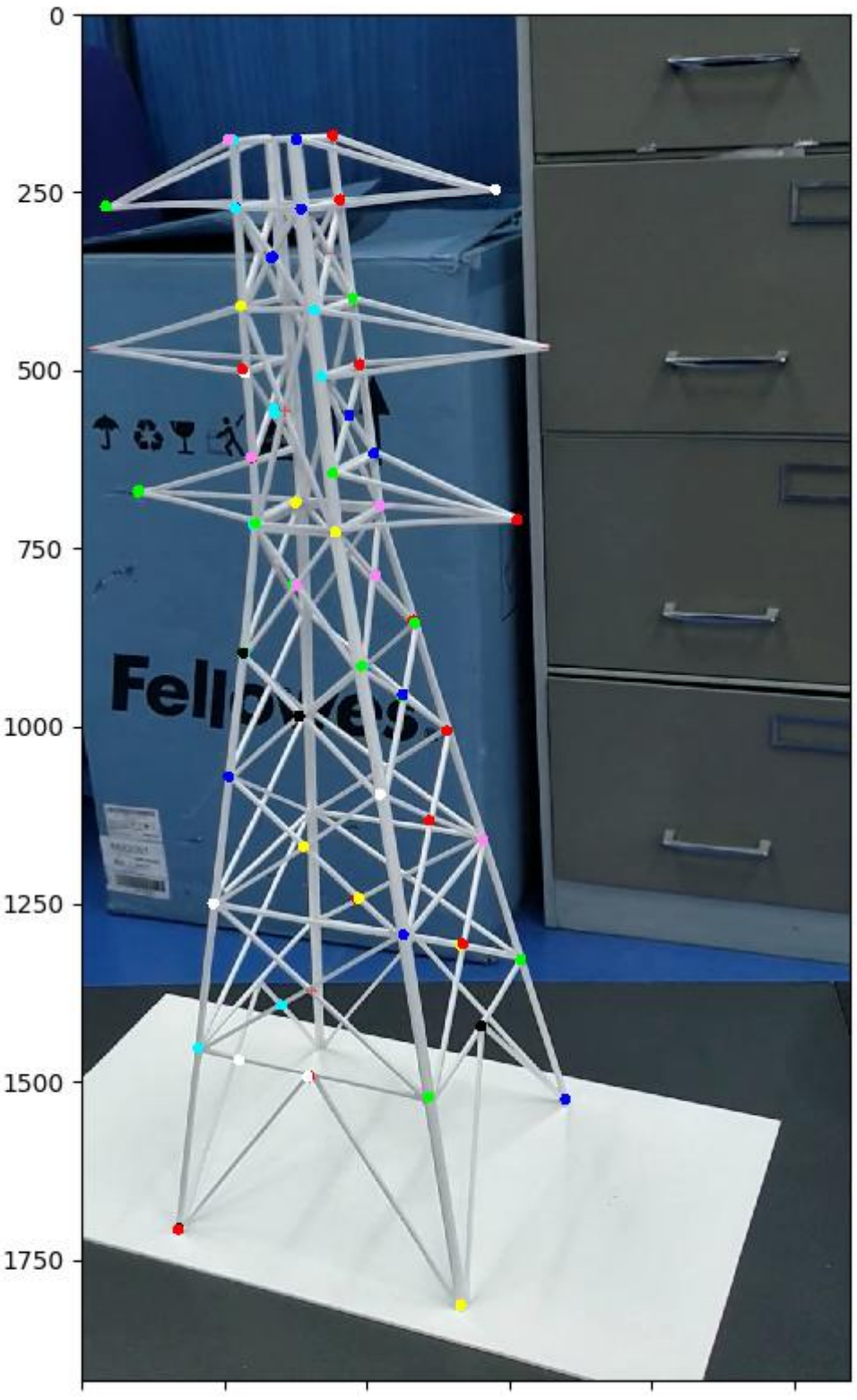


FIGURE 1.2: An image of the the CNN hand annotated label (+) and the predicted output of the network (•).

- Model-based image registration: We explore the use of a search-based strategy to register a pylon model onto an image. We obtain the model information of the pylon and store it in a hash table for the training. For the search, we use a voting technique to establish a match between the object in the scene and the stored model. It means that with available geometrical information of an object we can establish the pose estimation of the object in a single image.

  Image matching requires registering a model of the object onto a scene image and obtaining a match with the model. By successfully matching the image scene we establish correspondence with the model. Assuming we have a model and we want to match it to an object in the image scene, we can use a matching technique like geometric hashing, which is a model-based technique. The matching helps in detection of the object thereby obtaining the pose estimate between the model and the object.

  Features on the pylon are extracted using a method like vertex detection. Geometric hashing uses the features to establish a match between objects to obtain the detection and solve for the single image six degree-of-freedom pose estimation.

  Geometric hashing is used to match objects between the stored model and the image taken. Thus with the model of the object we can register the object onto an image. The model can be a CAD model, a point cloud, a mesh or an architectural drawing of the object.

  The algorithm works on ordered pair features called a basis set. The basis set is the minimum number of features for transformation of an object. For instance, three features are required for an affine transformation so the basis set for an affine transformation has three features. All possible basis sets of each model in the training stage are obtained and stored in a hash table. The hash table helps for a fast search. The model in this work is the face of a pylon.

  If a pylon exists in an image scene, we find all the possible basis sets in the image. From the collection of basis sets obtained we arbitrarily pick a basis set. We use this basis set to determine a transformation. Then the transformation is used to transform the other features. The features are used as voting elements for the model and basis set in the hash table. The model and basis set with the highest vote is equivalent to the scene basis set and a

match has been found. Therefore, the object has been recognised using the algorithm.

The information in the hash table experiences degradation when storing it during the training stage and at retrieval in the recognition stage. The degradation of the information has an effect on the accuracy rate. We see the degradation is caused by noise in the features, weighting by the hash bin size and the quantization of the values.

- Pose estimation and tracking: We use geometric constraints on a convolutional neural network for six degree-of-freedom pose estimation and tracking. We have the pose estimation from a single image, which we use for the initialisation. With the initialisation we are able to track the movement of the camera.

  The localisation framework uses single image pose estimation for initialisation and a tracker for the incremental movement of the camera relative to the pylon. The single image pose estimation uses the features from the vertex detection and the geometric hashing as the search-based strategy. The method is done with a model-based technique, where the world coordinates of the pylon are known. The tracking is done with an extended Kalman filter and vertex detection as measurements.

  We provide information about the movement of the camera, which gives the motion model as a representation of the movement. The motion model used is a random walk. The observation model provides the registration of the world model onto the image scene. The filter minimises the difference between registered world points and the measurements. For data association, we want to establish correspondence between the world points and the measurements by obtaining the inliers and finding closest neighbours.

  We show the extended Kalman filter is successful in tracking the camera relative to the pylon by comparing the ground truth pose and the estimated pose.

  Alternatively, we build a tracking algorithm that uses the heatmap information from the convolutional neural network to follow the movement of the camera. Instead of using data association with the inliers, we want to maximise the score obtained from all vertices while minimising the distance to maximum probability scores. A gradient descent approach is used to adjust the camera pose estimate in each frame.

Figure 1.3 shows the result of the tracking of the camera pose estimation from one of the frames in the sequence. The 3D pylon model is registered and displayed as a 3D bounding box in the image scene.

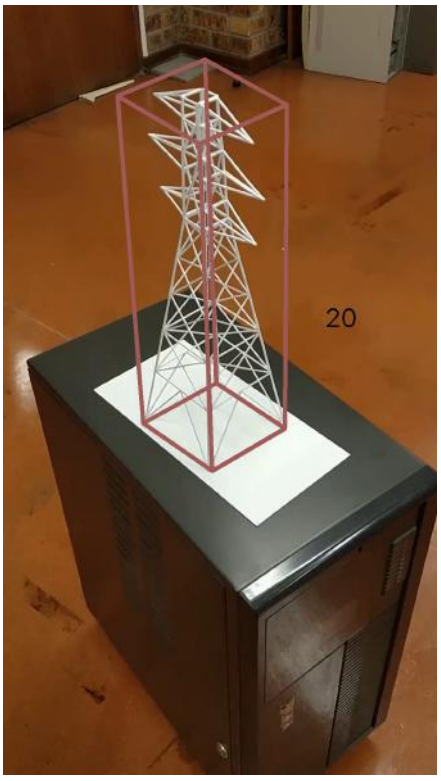


FIGURE 1.3: An example of a frame in the sequence with the 3D pylon model registered onto the image scene.

In Figure 1.4 we present a central diagram that shows the link between the various aspects of the research. Vertex detection is shown as an input to the initialisation and tracking process. Also, a fixed 3D model of an electricity pylon is fed into both the initialisation and the tracking process. The initialisation step is the stage where pose estimation of an object in a single image is obtained using a model-based registration technique like geometric hashing. Incremental tracking of the camera is done using an extended Kalman filter or a gradient descent tracker with an input of measurement from the detected vertices and 3D model of the electricity pylon.

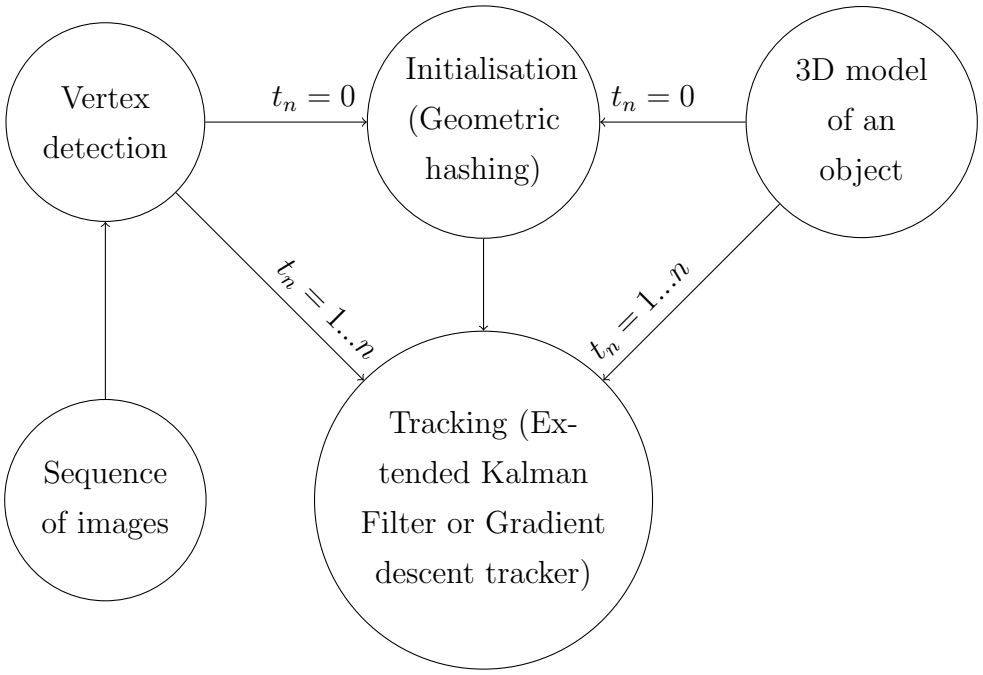


FIGURE 1.4: A block diagram that shows the various interconnecting parts of the algorithm.

During the course of the research we published some of the results in a paper as follows:

- Ali, Emmanuel Y., and Fred Nicolls. "3D Pose Estimation and Tracking of an Electricity Pylon." 2020 International SAUPEC/RobMech/PRASA Conference. IEEE, 2020.

1.4 Thesis Organisation

The thesis is organised in such a way where we discuss the concepts we used and provide chapters for the experiments.

In Chapter 2 we discuss the relevant background pertaining the inspection of power infrastructure. We cover the general details about inspection. In inspection you need a vehicle and data analysis. We discuss them and the gap identified that we are trying to solve in this research.

The Kalman filter is discussed in Chapter 3. We provide some theoretical background for the filter, discuss motion models, and details of the operation of the Kalman filter.

In Chapter 4 we discuss the convolutional neural network and some of its fundamental concepts. We show how we can achieve keypoint detection with the neural network. This keypoint detection serves as the feature of the pylon used as measurement for the geometric hashing and Kalman filter. The keypoints are the unique points on the pylon where the bars intersect, which we refer to as the vertices.

In Chapter 5 we discuss geometric hashing as an image registration technique to match models onto an image of a scene. We explore details about the geometric hashing, its performance under different factors and how well it works. In this chapter, we focus on the conceptual ideas behind geometric hashing. The implementation of the algorithm for the six degree-of-freedom registration is discussed in Chapter 6.

In Chapter 6 we provide the details and experiments we undertook to achieve the six degree-of-freedom pose estimate of the camera relative to the pylon. We show how we combine the concepts of extended Kalman filter in Chapter 3, convolutional neural network in Chapter 4, and geometric hashing Chapter 5 for the pose estimation and tracking experiments.

We conclude the thesis in Chapter 7, where we discuss our contributions and summarise the work. The work has been on the localisation of the pylon, which will allow for localised inspection of components on or around a power pylon, using a three-dimensional model of the structure.

Chapter 2

Background and Related Work

Inspection is the process of checking the functionality and the effectiveness of goods, structures and products. It is a way of ensuring quality control and making sure things are in good shape and functioning properly. This tends to be an expensive and time consuming process. Different means of automating the inspection for power infrastructure from line robots to aerial robots are being explored. Traditionally, inspection is done using foot patrols, vehicle patrols or helicopters.

We review work on inspection of power infrastructure from the vehicles used to collect data to various techniques utilised for the inspection process. The power line is made up of various components as pictured in Figure 2.1. The inspection of structures on the power line is mostly of two types:

- Component inspection: This inspection process concentrates on the transmission line and the pylon, hardware and attachments supported on or around the pylon. It is expected to check for broken, deteriorated or malfunctioning equipment, conductors, braces, insulators, pins, transformers, insulator ties and cross arms, and to check that the line is not obstructed by trees or vegetation.

- Structure inspection: According to the safety code in certain countries, the structure of the poles and pylon have to be within certain limits like the safety factor, withstand stress and have the required load capacity. The inspection looks for damaged poles, especially as poles degrade over time due to wear and tear.

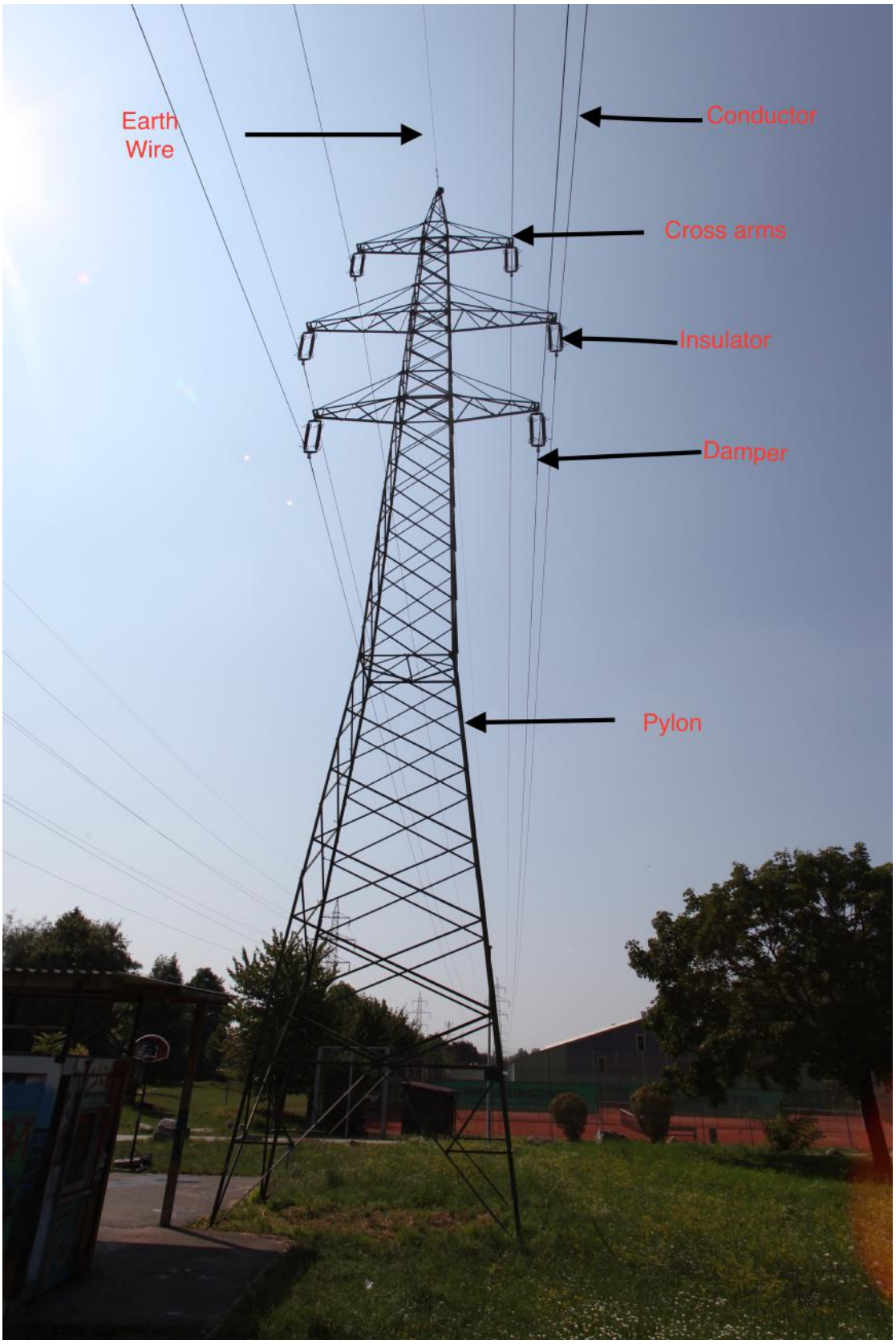


FIGURE 2.1: Components of a power line.

To undertake the inspection of either the components or the structure, we need to find a way to accurately locate their position. Conventionally, the localisation and inspection is performed physically by people, but increasingly robots are being used to automate the process. For robots to achieve full autonomy we need to find sensors that will help with accurate localisation.

The rest of this chapter discusses these ideas. Section 2.1 talks about the means of transportation used for the inspection process. We take a look at some studies that were done to aid with autonomous process of inspection. Section 2.2 discusses the main inspection processes that do not require human intervention. Finally, Section 2.3 discusses the localisation techniques and some methods used to solve the localisation problem.

2.1 The inspection vehicle

In order to perform inspection, vehicles are required to carry equipment and capture the details needed for the process, especially data collection and mapping of the faults. There are different vehicles used for this process and they are reviewed here, from conventional methods such as foot patrols and helicopters to methods trying to achieve autonomy for the entire process namely, line inspection robots and unmanned aerial vehicles (UAVs).

2.1.1 Foot Patrols

The power line is inspected regularly by people walking along the lines and inspecting it as in Figure 2.2. Inspection using foot patrols requires an individual or a group of people to walk or drive around while checking the power line using visual inspection or heat sensors. Detailed inspection is not possible unless aided with binoculars or other measuring instruments. Even with the aid of instruments, a person may miss some components or details especially for the parts at the top of the pylon or if some parts are occluded.

Sometimes the inspection requires a person climbing the pylon for the assessment of the structure; this requires power outage which is not favourable.

FIGURE 2.2: Foot patrols as means of inspection.

Foot patrols tend to be slow and time consuming. Access to some places is often difficult and becomes increasingly dangerous and impossible after a disaster. This constitutes a major limitation for this method of inspection. The key advantage of this method is that it is relatively cheap and does not require specialised skills.

2.1.2 Helicopters

There are various global helicopter utility companies providing services for the inspection of power lines. With helicopters it is possible to do aerial surveys and collect images of the facilities as shown in Figure 2.3.

The helicopter as a vehicle for inspection is costly. Also, a helicopter has to come close to the components for a detailed inspection. The danger of the helicopter for the inspection was highlighted when a helicopter was laser scanning a power line, hit a crossing power line, and cut three of the conductors [5]. Thus, a helicopter may be faster than a foot patrol but it is a very dangerous option.

FIGURE 2.3: Helicopter used for power line inspection.

2.1.3 Line Inspection Robots

The need for autonomous robots has led to the design of power line inspection robots. These are robots that move along the transmission line.

The research group in Hydro Quebec [6] have designed an inspection robot called LineScout (Figure 2.4) that has undergone different improvements through the years. The design paid attention to the electromagnetic interference (EMI) caused by the high voltages present. The robot comprises motorised wheels, arms and grippers. The wheels are used for the movement on the lines, while the grippers are used to cross between obstacles.

Another line inspection robot [7], the Expliner (Figure 2.5), was designed in Japan for the inspection of power lines. It is made up of pulleys driven by motors, pipes for connection, slide-in connectors and a manipulator that has a counter-weight which houses the electronics. The movement of the manipulator can change the center of mass to overcome any obstacle on its path. The mechanical design of the robot took into cognizance the friction between pulleys and the power cables. The motion unit comprises a brush-less motor and planetary gears. The robot is

FIGURE 2.4: LineScount robot.

remotely controlled with an operator instructing the robot on crossing obstacles such as the clamper and damper by using the manipulator.

FIGURE 2.5: Expliner robot.

The power line robot is a line inspection robot (Figure 2.6) made at the University of Kwazulu-Natal (UKZN) [8]. The power line robot was the inspiration for this

research and the question of how to provide localisation capabilities to it using visual components.

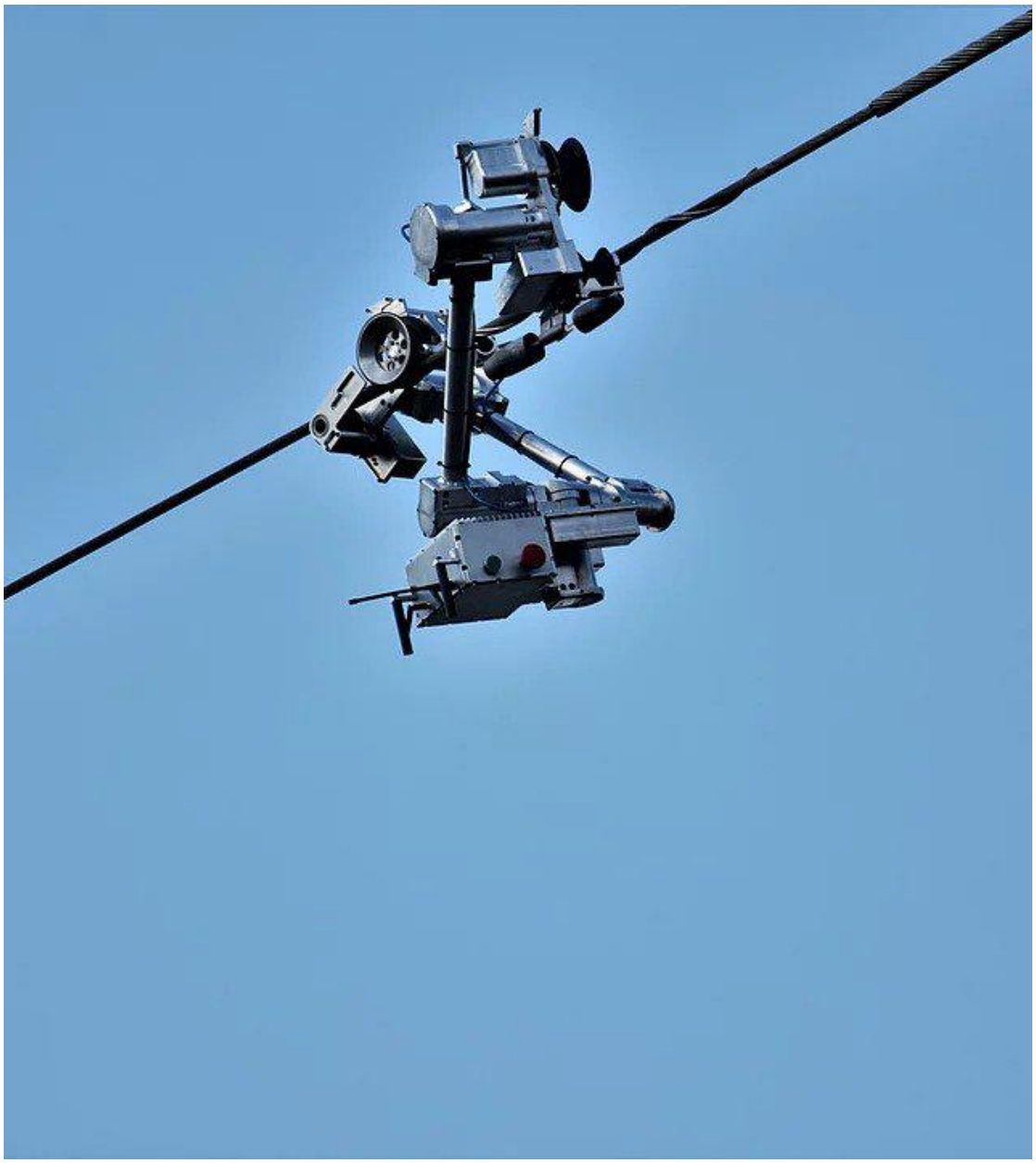


FIGURE 2.6: Power line inspection robot.

There are other inspection robots that have been designed with varied configurations such as flying-walking power line inspection robot [9], one-wheel-drive low-cost powerline inspection robot [10], electric power intelligent inspection robot [11] and a review of various power line robots is provided by Alhassan et al [12].

These robots are fitted with different sensors for the inspection of the power line but they are mostly remotely controlled. To achieve autonomy we will need an accurate localisation mechanism fitted to the robot.

2.1.4 Aerial Robots

Aerial vehicles (Figure 2.7) have gained popularity in recent years with advances in manufacturing, stabilization and control. These have enabled them to be cheap compared to other methods, with flexibility and manoeuvrability, fast speed, and high safety. Their usage is being explored for different purposes from military to civil. Uses cover pedestrian and traffic surveillance [13], autonomous surveillance [14], precision agriculture [15], search and rescue [16], civil infrastructure [17] and many other areas. We review their use in the inspection of power infrastructure.

Araar et al. [18] use a UAV for the tracking of a power line with visual servoing methods. They make use of two approaches for the control of the UAV. In the first approach they use a linear quadratic servo with image features and kinematics, while online partial pose estimation is used for the second approach.

FIGURE 2.7: Aerial robot from the UPenn Grasp Lab.

Golightly et al. [19] worked on visual control of an autonomous robotic vehicle for overhead power line inspection and the research was conducted in the UK for power infrastructure. The Hough transform was used to measure the pose of the

vehicle as well as for obstacle detection and path planning. The vehicle used a ducted fan design to achieve dynamic stability. A laboratory setup was made to test the results of the work.

Luque-Vega [20] proposed the use of a quadrotor to carry out inspection. The quadrotor has a manual flight mode handled by an individual, and can also use the Global Positioning System (GPS) for navigation. There is a control station on the ground monitoring the activity of the robot. The vision system is composed of a thermal infra-red camera and a colour camera. The camera is used to take the necessary images around a joint and a temperature difference is obtained to find if a fault exists. The issue with this method is it requires the use of two or more cameras. Moreover, the joint has to be manually localised for inspection to take place.

Schofield et al. [21] worked on a drone for the inspection of a power line. They use a combination cloud service providers and build some algorithms themselves. Their work focused on detection of the pylon and power line and tracking it with an extended Kalman filter.

2.2 Detection and Fault Analysis

Inspection involves identification of components and finding of faults or abnormalities. For the inspection process, each of the components of the power infrastructure will need to be detected and identified. The power infrastructure is made up of various components namely the electric tower (pylon), insulator, the transmission line (cable), damper, and splicer. For the inspection process, each of these components will need to be detected and identified. Montambault et al. [22] review the existing UAV application in the inspection of power utility assets. They provide a brief overview of the subsystem of the research and the various research centres involved in the field of inspection of power installations. The work discusses the vehicle for data collection, the control system and telecommunication, and the analysis of the data.

Jenssen et al. [23] reviews the vision-based systems used for inspection. They discussed concepts relating to the type of inspection, data sources for inspection, computer vision applications in inspection, constraints, and potential future directions for developing research on inspection procedures, particularly with regard to

navigation. Liu [24] also analyzed a number of publications that dealt with the identification and detection of power line components. The study provides details of publicly accessible datasets for various power line parts, including the tower, insulator, and conductor.

In this section we look at the work that deals with object detection relating to power infrastructure components. These components are the pylon, the insulator and the transmission line.

2.2.1 Pylon Detection

The pylon is a structure that supports a long stretch of the transmission line. Pylons are of different types from the truss-like pylon to poles and towers. There are several works that explore the classification and the detection of the pylon. These works do not attempt to solve the pose estimation problem.

We explore some of these works. The research lab in Bangor [25] explores the use of corner based methods for the inspection of poles. They used the Cooper, Venkatesh, Kitchen (CVK) approach for the corner detection, which claims to be fast, robust to image noise and is straightforward to program. The method can be summarised as i) find contours on the image with a steep intensity gradient, and ii) compute a dissimilarity measure for patches along the direction of the contour. The values for the four parameters are calculated as the gradient threshold, the dissimilarity threshold, the size of the image patch to calculate dissimilarity and the distance between successive image patches along a contour. The method found clusters of small-scale corner points in the vicinity of a physical corner.

The work of Cheng et al. [26] proposes the use of graph cuts for the detection of a power pole. The rough region of the power pole is found in the image using the Radon transform for a straight line. The region is divided into two, where one is enlarged and the other shrunk. With these regions found, a weighted graph is constructed. They made the assumption that the cross arm and the pole are straight lines and are at right angles, the pole is vertical, and the cross arm is shorter and narrower than the pole.

The use of the line segment detection (LSD) method together with colour detection of metals to find the pylon was done by Oaulid [27]. In the work, the colour attributes of the pylon were used to extract it from the image. They work on the

basis that the pylon is metallic and has linear features. The linear features can be found as a number of connected line segments. The work observed these features of the pylon: i) the colour of the pylon is fairly consistent, ii) the pylon is of low colourfulness, and iii) the pylon is made of a number of connected segments. The work used these features as the basis of their research. The connected segments are detected by the use of the line segment detection algorithm as they are also linear. The LSD was able to detect the line segments but it also detected objects in the background. Thus to remove the background, the pylon's low colourfulness was used as a filtering process. The approach used the intensity of the pixel and the order of the RGB channels of the pixel for the colour filtering.

Campos et al. [28] used a learning approach for the detection and classification of the pylon. The histogram of gradient (HOG) method is used for the extraction of the features and the multilayer perceptron (MLP) neural network is used as a classifier. The work was in two parts; the first part is the detection of the pylon in an image and the second part is the classification of the pylon according to different types. In the detection part, a two-class neural network is used to distinguish a pylon from a background image. The sliding window approach is used to locate the position of the pylon. In the classification part, a four-class neural network is used and the classifier tries to differentiate the type of pylon.

Rabab et al. [29] published datasets for detection and segmentation of electricity pylons.

Hui et al. [30] presents a navigation approach for an UAV to perform inspection. The approach uses a perspective image to obtain cues that aid in three-dimensional perception. To achieve, they use a detection and tracking strategy. The detection is based on faster region based convolutional neural network and the tracking is based on kernelized correlation filters. The vanishing point is thereafter calculated with segmented transmission lines. The segmentation is based on fully convolutional network.

To solve the navigation, Bian et al. [31] propose a detection of the pylon and simultaneously estimating an UAV positions. The detection of the pylon uses convolutional neural networks methods and to estimate the positions, they use point-line SLAM techniques.

2.2.2 Transmission Line Detection

The transmission line is the component that carries electric current between places. The rest of the components are there to support this component; even if there is no fault it needs to be given a clear right of way without any obstruction. Also, there is a need for the UAV to avoid colliding with it.

Golightly et al. [19] worked on visual control of an autonomous robotic vehicle for overhead power line inspection. The research was conducted for power infrastructure. The Hough transform was used to measure the pose of vehicle as well as for obstacle detection and path planning. A vehicle used the ducted fan design to achieve dynamic stability. A laboratory set-up was made to test the result of the work.

Zhang et al. [32] used the Hough transform and k-means clustering for the detection of the power line and a Kalman filter for tracking the line.

2.2.3 Insulator Detection

The insulator is made to support in holding the transmission line on the pylon because it does not allow the flow of electricity. The insulator is an example of a power component around the pylon and the localisation of the pylon facilitates the inspection of the insulator.

We introduced some of the research about finding the insulator and the detection of a faulty insulator. Wang and Zhang [33] focus their attention on the recognition of insulators. They use a support vector machine (SVM) as the classifier and Gabor filters for feature extraction. In their work the image background is first suppressed, then the features are extracted and finally trained with a classifier.

The work of Oberweger et al. [34] is on insulator detection and fault identification. The detection uses trained local gradient-based circular descriptors that vote for the localisation. The features were detected with Difference of Gaussian (DOG) and the circular GLOH-like (CGL) descriptor was calculated. They used principal component analysis (PCA) to reduce the dimensionality of the descriptor to make it faster. The k nearest neighbour (kNN) classifier is employed for the training of the features. The features are grouped according to scale and random sample consensus (RANSAC) is applied to fit the insulator model to the detected features.

After detecting the insulators, they analysed it for faults. An estimation of the orientation is made and each insulator cap separated with the Canny edge detector. The alignment of the partitioned insulator is calculated with the cross-correlation of signal. An elliptical descriptor is used to extract the features. The local outlier factor approach is used, which checks for dissimilarity with a voting threshold.

In a paper by Usiholo [35] they did detection of the insulator with the use of active contours. It detected edges and used morphological operations for the filtering. Active contours were able to segment the image and identify insulators.

Miao et al. [36] used a single shot detector (SSD) for object detection of insulators on a power line. They built two different datasets. The first dataset is for basic insulator detection that is to differentiate between an insulator and other components on the power line. The second dataset is for specific insulator detection to distinguish between various insulator types. They are a couples of papers that used other object detection with deep learning methods such as YOLO [37–39], Faster RCNN [40]

2.3 Pose estimation

In the previous section we discussed some of the work that has been done pertaining the power infrastructure inspection process. For the robots to function autonomously they will need to know where they are relative to the components, which is the pose estimation task. The pylon, being the most prominent component of the power infrastructure, provides a context for the other components. Pose estimation for the pylon has not received much attention. Thus we review the work that has been done on pose estimation. The work reviewed involves the pose estimation of textured and textureless objects.

In any inspection process, the first step is locating the most accurate position and orientation of the object. We can use different sensors to achieve this but each has its limitations. The camera gives the advantage of seeing the object and choosing the best position without collision. There are different visual localisation techniques in the field. Most of the methods for pose estimation discussed are for general types of object but there are others for specific objects such as cars [41] and humans [42].

Generally, pose estimation for well-textured objects has been done using feature based techniques such as SIFT, SURB [43] and ORB [44]. Feature keypoints are points of interest that are recognized to be unique. The points are distinct so they can be used to describe and to identify objects. The feature function has large gradients. The feature technique depends on the appearance of the image.

The scale invariant feature transform (SIFT) [45] is a common feature detection technique that has been used in object recognition, structure from motion and motion tracking. The method uses the difference of Gausssians to identify possible interest points that are invariant to scale and orientation. Keypoint localisation is done to check the stability of the keypoint. Each keypoint is assigned orientations to provide invariance. A keypoint descriptor is made by knowing the image gradient around a keypoint. To establish a match between keypoints, it finds keypoints with the same location, scale and orientation. The search for the match is aided by using a hash table implementation of a generalised Hough transform.

We provide some works that use feature techniques as the localisation framework. Savarese and Fei-Fei [46] worked on 3D object recognition. They made the assumption that reconstructing a 3D object is not necessary for recognition. Instead, they propose that using weak geometrical information is sufficient. They captured appearance in the canonical parts of the object in a class. The model representation also encodes the appearance variation observed in training in the form of distributions for the descriptors. They made linkages between the canonical parts. The implementation of the work starts by extracting saliency descriptors and SIFT descriptors to represent the local features. The local features are grouped into canonical parts and linked to provide a 3D model. The grouping is done for parts with consistent appearance and geometry. A global geometrical constraint is obtained by imposing feature match candidates. The matching was done using a global optimization technique with the candidate canonical parts.

The work of Ghordrati et al. [47] investigated the use of 2D and 3D representations for pose estimation. They use a 2D representation and show that it works for viewpoint estimation. The work takes an input from a detected bounding box using an off-the-shelf detector, extracts the features, and uses a Fisher Vector (FV) pyramid and neighbouring viewpoint suppression for pose estimation.

For weakly textured and textureless objects, the feature based techniques tend to fail. The pylon is weakly textured therefore the feature based techniques fail for

it.

The use of a RGB-D camera provides depth information and is being used to solve pose estimation for weakly textured objects. A template matching technique was used by [48] for the 3D pose estimation problem where they used the color and depth information to establish a match with templates from different views of the object.

Drost et al. [49] propose a model description with oriented point pair features. Similar features on the model are grouped and stored in a hash table during the training phase. In the on-line phase, the matching of the features is done with a voting scheme similar to the generalised Hough transform to recover the pose. They further improve their work [50] by refining the pose estimate with the iterative closest points algorithm (ICP).

The use of random forest was explored by [51]. The random forest was used to predict 3D object coordinates and instance probabilities of the object, and the output of the network is fed into a cost function which was solved using a RANSAC-based method for the solution of the pose estimate.

A Hough forest detector for the part-based detection and LINEMOD for the feature extraction was used by Tejani et al. [52] for the pose estimation problem.

Some of the pose estimation methods try to learn the six degree-of-freedom pose estimate directly. These methods mostly require that the objects be segmanted before the pose estimation task. DenseFusion [53] is an example of such a technique. In DenseFusion, the technique fuses two networks. The first network estimates the pose while the second network is used to refine the pose estimate. They combine the RGB values and the point cloud information generated from the depth for the training. The combination provides the model with local appearance and geometric information to handle occlusion.

PoseCNN [54] estimates the rotation and the translation separately. The rotation is obtained by regressing convolutional features of a bounding box of an object for the quaternion values and estimates the translation by obtaining the object center and predicting the distance. They used two types of losses where one measures the match between the objects, but their loss could not handle to symmetric objects; thus they had another loss to handle symmetric objects.

Crivellaro et al. worked on a pose estimation and tracking [55] algorithm suited to an augmented reality application. They use a convolutional neural network for the part based detection. Each part is a discriminative region of the object and represents a 2D reprojection of the 3D points. The 3D points they chose did not represent any special feature but are arbitrary points used to provide information for achieving 2D-3D projections. The network output was invariant to the image location on the parts and depends on the image appearance. The network detects several candidates for the parts and the most likely candidates given the prior on the pose represented as Mixture-of-Gaussians is selected. The prior is used to define the normal range of the camera. The pose computed for previous frames is incorporated into the system for temporal consistency. An optimization is done between the prior and the parts that is solved using the Gauss-Newton algorithm for the pose estimate. For the best pose estimate they evaluated the pose under different cues, and trained a linear regressor to predict the penalty for the different cues. An extended Kalman filter was used to smooth out the trajectory and reduce jitter. The technique does not learn a specific part but arbitrary chosen, which leads to inconsistency in choosing of parts.

Hofer et al. [3] attempted structure from motion for the reconstruction of a pylon and its environs. They detect line segments of the structure with a line segment detector (LSD), use epipolar geometry for the generation of 3D line segment hypotheses and apply spatial proximity-based grouping for clustering and removal of outliers and incorrectly triangulated line segments. When attempting the method they found localisation around the background area that was textured. This work was done to provide geometric information to be incorporated into a SLAM algorithm. Instead of reconstructing the pylon and combining with a SLAM we opted for a model-based approach for the localisation.

There are other techniques that have been explored for pose estimation. Su et al. [56] showed that a CNN can be used for viewpoint estimation with rendered images. They were able to use large-scale training data with fully annotated viewpoint information. Thus, a large amount of rendered data is required for this method.

Object detection has enjoyed relatively good success with an output being the bounding box. The work of Rubino et al. [57] uses the bounding box from multiple viewpoints as the first step for its localisation algorithm. They then formulated

the localisation as a conic optimisation problem with a closed-form solution in dual space.

Glasner et al. [58] explored the use of a non-parametric voting technique for the generation of hypothesis pose estimates and use a support vector machine for the verification of the result. They used a reconstructed object as the input of their algorithm. The method uses both learning and voting to achieve the pose estimation.

Most of the works we have reviewed are not concerned with the pylon as a textureless object. In our work we focus on the usage of a RGB camera and the application of geometric constraints for the pose estimation. We make use of learning and voting techniques for the work.

2.4 Summary

In this chapter we provide the background information pertaining to the research. We cover the need for inspection, how it is being undertaken and the attempts to automate the process. We also discuss some of the works that have been used for the detection of components.

Finally we introduce some of the work that has been done on pose estimation. The work ranges from pose estimation on textured objects using point descriptor techniques to the recent attempts at pose estimation on textureless objects.

Chapter 3

Mathematical framework for Kalman Filter

The Kalman filter has proven to be an effective tool when it comes to state estimation. It has been used since the 1950s [59, 60]. The filter can be used to extract information from noisy sources and be used for tracking. It uses knowledge of a model and a measurement to provide for accurate estimation of the environment. The level of uncertainty in the process and measurement is used to understand their contribution to the estimate.

There are different variants of the Kalman filter. The standard Kalman filter is a linear estimator that provides optimal results for a linear model with linear measurement and Gaussian noise. The issue is that most systems in the world are non-linear therefore the standard Kalman filter cannot handle them. For the standard Kalman filter to work, the system will need to be linearised around an operating point. This leads to other non-linear variants of the Kalman filter such as the extended Kalman filter, the unscented Kalman filter and the iterative extended Kalman filter.

During the course of this research we used the Kalman filter for tracking of the camera. In this chapter we discuss the theoretical background and in chapter 6 we show how the Kalman filter is used in the pose estimation and the tracking of pylon experiments.

3.1 The Kalman Filter

The Kalman filter holds true to the Markov condition that the state is the complete summary of the past [61]. It infers the state of a system from a set of noisy measurements.

Furthermore, there are certain assumptions that ensure the validity of the Kalman filter and these conditions are [62]:

1. The state transition function when applied to the Gaussian distribution state should produce a Gaussian distribution output.
2. The observation should be a linear function of the state with Gaussian observation errors.
3. The initialisation should be normally distributed and the future and past states must be independent given the current state.

The Kalman filter is meant to propagate the mean and the covariance of the system. The mean is the state of the system. We call the computation before the measurement the a priori state estimate $\hat{x}$. The variables after the measurement are the a posterior state estimate $\tilde{x}$.

The complete derivation of the Kalman filter is shown in [61] . It shows how the standard Kalman is an optimum filter. The Kalman filter is used as a tracking system and we provide explanation in the context of Bayesian inference.

The state transition model specifies the relationship between the various states. The model of a system and the state probability $Pr(x_n|x_{n-1}, u_n)$ have the following equation representation:

$$x_n = A_n x_{n-1} + B_n u_n + \epsilon_n. \tag{3.1}$$

The observation model provides the relationship between the measurements and the states at a certain time. The observation model of probability $Pr(z_n|x_n)$ of the system is given as

$$z_n = C_n x_n + \delta_n, \tag{3.2}$$

where, x_n is the state of the system l-dimension column vector, A_n is the transition matrix with $l \times l$ dimension, u_n is the input m-dimension column matrix, B_n is the input matrix $m \times m$ dimension, ϵ_n is the process noise, C_n is the observation matrix with $m \times l$ dimension, z_n is the output m-dimension column vector, and δ_n is the observation noise.

The a priori state estimate $\hat{x}_n$ and its covariance $\hat{\Sigma}_n$ can be obtained using Bayes' theorem. The mean is the optimal point and the covariance is the inverse of the curvature of the second derivative of the Bayesian inference given the state model and the last available belief.

The prediction state of the Kalman filter is the integral of the product of the state at a time given the state at a previous time and the associated probability of the previous state belief. The prediction involves finding the marginal probability density of the belief. The Bayesian inference for the prediction is given as

$$Pr(x_n|z_{1\ldots n-1}) = \int Pr(x_n|x_{n-1}, u_n)Pr(x_{n-1}|z_{1\ldots n})dx_{n-1}. \tag{3.3}$$

The equation can be expanded and placed in Gaussian form as

$$Pr(x_t|z_{1\ldots n-1}) = \kappa \int \exp\{\frac{1}{2}(x_n - A_n x_{n-1} - B_n u_n)^T \Omega_n^{-1}(x_n - A_n x_{n-1} - B_n u_n)\} \exp\{\frac{1}{2}(x_{n-1} - \tilde{x}_{n-1})^T \Sigma_{n-1}^{-1}(x_{n-1} - \tilde{x}_{n-1}))\}dx_{n-1} \tag{3.4}$$

with κ a constant and Ω_n as process noise covariance.

When the solution of the integral is obtained then the first derivative provides us with the equation of the a priori estimate $\hat{x}_t$:

$$\hat{x}_n = B_n u_n + A_n \tilde{x}_{n-1}. \tag{3.5}$$

Also, the covariance can be found as the inverse of the curvature of the system, and the second derivative is obtained as

$$\hat{\Sigma}_n = (\Omega_n + A_n \tilde{\Sigma}_{n-1} A_n^T). \tag{3.6}$$

We apply Bayes' rule for the incorporation of the measurements. The Bayesian inference for the update stage is given as

$$Pr(x_n|z_{1...n}) = \kappa Pr(z_n|x_n)Pr(x_n|z_{1...n-1}). \tag{3.7}$$

We can express the equation in Gaussian form with $Pr(x_n|z_{1...n-1})$ as the a priori state estimate $\hat{x}_n$,

$$Pr(x_n|z_{1...n}) = \kappa \exp(-\frac{1}{2}(z_n - C_n x_n)^T Q_n^{-1}(z_n - C_n x_n) - \frac{1}{2}(x_n - \hat{x}_n)^T \hat{\Sigma}_n^{-1}(x_n - \hat{x}_n)). \tag{3.8}$$

with Q_n as measurement noise covariance. The inverse of the second derivative is the covariance of the system,

$$\tilde{\Sigma}_n = (C_n Q_n C_n^T + \hat{\Sigma}_n^{-1})^{-1}. \tag{3.9}$$

Therefore, the a posterior state estimate is

$$\tilde{x}_n = \hat{x}_n + K_n(z_n - C_n \hat{x}_n) \tag{3.10}$$

with the Kalman gain $K_n = \tilde{\Sigma}_n C_n^T Q_n^{-1}$. The Kalman gain determines the amount of contribution to be made by the measurement to the a posterior estimate. Subsequently we can refine the Kalman gain as

$$K_n = \hat{\Sigma}_n C_n^T (Q_n + C_n \hat{\Sigma}_n C_n^T)^{-1}. \tag{3.11}$$

We can rewrite Equation 3.9 in terms of K_n:

$$\tilde{\Sigma}_n = (I - K_n C_n)\hat{\Sigma}_n. \tag{3.12}$$

Addition of the measurement improves the certainty of the estimate, and the Kalman gain modifies the certainty. The a posterior covariance is less than the a priori covariance by a factor of the Kalman gain.

3.1.1 Extended Kalman Filter

Most natural systems are not linear and therefore one cannot use the standard Kalman filter. A system that typifies this non-linearity is the extended Kalman

filter (EKF). The state and measurement equations can be non-linear as follows:

$$x_n = g(u_n, x_{n-1}) + \epsilon_n \tag{3.13}$$

$$z_n = h(x_n) + \delta_n. \tag{3.14}$$

The non-linear equations need to be approximated as linear systems. To do this we linearise the non-linear equations. Linearisation is obtained using the Taylor series to differentiate the equation with respect to the independent variables:

$$\begin{aligned} g(u_n, x_{n-1}) &\approx g(u_n, \tilde{x}_{n-1}) + g^{'}(u_n, \tilde{x}_{n-1})(x_{n-1} - \tilde{x}_{n-1}) \\ &\approx g(u_n, \tilde{x}_{n-1}) + G_n(x_{n-1} - \tilde{x}_{n-1}), \end{aligned} \tag{3.15}$$

where $g^{'}(u_n, x_{n-1}) = \frac{\partial g(u_n, x_{n-1})}{\partial x_{n-1}}$ is the Jacobian of $g(u_n, x_{n-1})$ and the higher order terms are considered negligible.

The state equation can be written as the Gaussian probability approximation

$$\begin{aligned} Pr(x_n|u_n, x_{n-1}) \approx \det(2\pi\Omega_n)^{-\frac{1}{2}} \exp -\frac{1}{2}[x_n - g(u_n, \tilde{x}_{n-1}) - G_n(x_{n-1} - \tilde{x}_{n-1})]^T \\ \Omega_n^{-1}[x_n - g(u_n, \tilde{x}_{n-1}) - G_n(x_{n-1} - \tilde{x}_{n-1})]. \end{aligned} \tag{3.16}$$

Similarly for the measurement equation, we implement the linearisation

$$\begin{aligned} h(x_n) &= h(\hat{x}_n) + h'(\hat{x}_n)(x_n - \hat{x}_n) \\ &= h(\hat{x}_n) + H_n(x_n - \hat{x}_n). \end{aligned} \tag{3.17}$$

The measurement equation is given as

$$\begin{aligned} Pr(z_n|x_n) = \det(2\pi Q_n)^{-\frac{1}{2}} \exp -\frac{1}{2}[z_n - h(\hat{x}_n) - H_n(x_{n-1} - \hat{x}_n)]^T \\ Q_n^{-1}[z_n - h(\hat{x}_n) - H_n(x_{n-1} - \hat{x}_n)]. \end{aligned} \tag{3.18}$$

The prediction equation is given as

$$Pr(x_n|u_n, z_{n-1}) = \int Pr(x_n|x_n, u_n), Pr(x_n|z_{1\ldots n})dx_n. \tag{3.19}$$

The solution of the integral and the first derivative provides us with the equation of the a priori estimate $\hat{x}_n$:

$$\hat{x}_n = g(u_n, \tilde{x}_{n-1}). \tag{3.20}$$

Also, the covariance is found as the inverse of the curvature of the system and the second derivative is obtained as

$$\frac{\partial^2 g(u_n, x_{n-1})}{\partial x^2} = (\Omega_n + G_n \Sigma_n G_n^T)^{-1}. \tag{3.21}$$

Therefore, the covariance of the a priori estimate $\hat{\Sigma}_n$ is

$$\hat{\Sigma}_n = (\Omega_n + G_n \tilde{\Sigma}_{n-1} G_n^T). \tag{3.22}$$

In the measurement update of the EKF, we incorporate the measurement value to find the a posterior estimate as

$$Pr(x_n|u_n, z_n) = \kappa Pr(z_n|x_n) Pr(x_n|z_{1\ldots n})). \tag{3.23}$$

We can express the equation in Gaussian form with $Pr(x_n|z_{1\ldots n-1})$ as the a priori state estimate $\hat{x}_n$:

$$\begin{aligned} Pr(x_n|z_{1\ldots n}) = \kappa \exp(-\frac{1}{2}(z_n - h(\hat{x_n}) - H_n(x_n - \hat{x_n}))^T Q_n^{-1}(z_n - h(\hat{x}_n) - H_n(x_n - \hat{x}_n)) \\ + \frac{1}{2}(x_n - \hat{x}_n)^T \hat{\Sigma}_n^{-1}(x_n - \hat{x}_n). \end{aligned} \tag{3.24}$$

The a posterior state mean and covariance is

$$\tilde{x}_n = \hat{x}_n + K_n(z_n - h(\hat{x}_n)) \tag{3.25}$$

$$\tilde{\Sigma}_n = (I - K_n H_n)\hat{\Sigma}_n. \tag{3.26}$$

The Kalman gain is

$$K_n = \hat{\Sigma}_n H_n^T (Q_n + H_n \hat{\Sigma}_n H_n^T)^{-1}. \tag{3.27}$$

3.2 Motion model

The motion model provides us with the information of the movement of the camera relative to the object. There are different motion models that are derived from classical physics. These are constant displacement, constant velocity, constant acceleration, constant snap and constant jerk, all depending on the order of the differential.

This research is based on the object pose estimation relative to the pylon. The pylon is considered stationary, while tracking is done with the camera in motion. The motion of the camera is represented as translation t_n which is the linear movement at various times $n = 0, 1, 2, ..., k$. Also, the rotation s_n represents the rotational movement at various times $n = 0, 1, 2, ..., k$.

The translation and rotation provides all the information needed to know the location of the object relative to the camera. The translation is the x, y, z coordinates of the Cartesian plane (Figure 3.1).

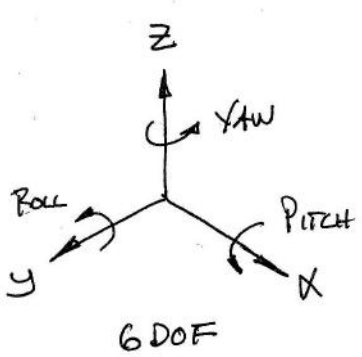


FIGURE 3.1: The six degree of freedom pose plane.

The rotation is more complicated than the translation. It has different representations. The Euler angle representation uses roll, yaw, and pitch as the rotation and is easy to understand and work with, but it has the problem of singularity due to gimbal lock (Figure 3.2). The axis-angle representation works but also has the singularity problem. To work around the problem we decided to use a quaternion representation.

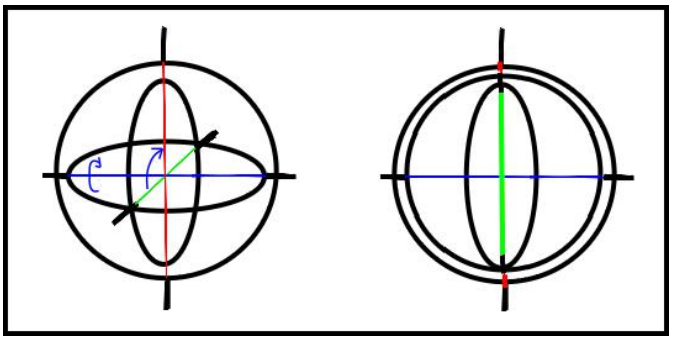


FIGURE 3.2: Gimbal lock [1].

3.2.1 Random walk model

The random walk model is based on the assumption that at each time the variable is a random step away from the previous value. The steps are independently and identically distributed: $X_{n+1} = X_n + W_n$. Therefore, the future variables depend on the previous observed variables with added Gaussian noise.

We applied the random walk model to track the movement of the camera. We should note that for the random walk model the covariance grows linearly with time, which produces large dispersion for many problems. Also, there is no well-defined velocity for the object.

The pose is given as the components of the translation and rotation of the object relative to the camera. The pose has six degrees-of-freedom. The translation can be represented as $t = [t_x, t_y, t_z]$ with a state transition of

$$t_n = t_{n-1} + \delta_n \tag{3.28}$$

where δ_t is a normal random variable with density $\mathcal{N}(0, \sigma_t^2 I)$.

We use a quaternion representation for the rotation. The quaternion needs to meet the nonlinear constraint $q_0^2 + q_x^2 + q_y^2 + q_z^2 = 1$. At each step, there is normalisation to ensure the constraint holds. We have a base unit quaternion q_0 representing the normal orientation, and a perturbation representing the deviation. At each incremental step we update the base quaternion with the axis-angle and reset the axis-angle. Then the base orientation and the perturbation quaternion is combined to provide the true quaternion $q = q_0 q_p$. The transition model of the rotation is represented as

$$s_n = s_{n-1} + \epsilon_n \tag{3.29}$$

with s the perturbation and $\epsilon_n \sim \mathcal{N}(0, \sigma_s^2 I)$. The combined state for the complete six degree of freedom pose is

$$x_n = \begin{pmatrix} t_n \\ s_n \end{pmatrix}. \tag{3.30}$$

The camera is moving and the object is fixed. Also, the 3D coordinates of the object are known and fixed, represented as Π. To obtain the camera projection matrix P, we will combine the calibrated intrinsic camera parameters K and the extrinsic camera parameters (R and t):

$$P = K[R|t].$$

Hence, the observation model is given as

$$z_n = P\Pi + e_n = h(x_n) + e_n, \tag{3.31}$$

with $e_n \sim \mathcal{N}(0, \sigma_z^2 I)$.

The function $h(x_n)$ is nonlinear and combines the 3D coordinates and the action of the camera. As the observation process is non-linear, an extended Kalman filter will be needed for the state estimation.

3.3 Error analysis

Propagation of covariances involves analysis of the transient behaviour of the output and the linear time varying properties of a random process and system such as is done in a Kalman filter. It is all about how the input variables uncertainty have an effect on the output variables uncertainty. For instance, in the prediction stage how do the process and noise uncertainties affect the a priori state estimate uncertainty?

Finding the right covariances is a difficult task and takes some troubleshooting, especially the process covariance. The covariance specifies the level of confidence we have in a particular system model. A high value indicates that we do not have confidence in the estimate for that variable while a low value shows a high level of confidence.

The Kalman filter provides an estimate that depends on the degree of confidence we have in the system.

When determining the uncertainty value we consider the number of points used for the computation, the accuracy of the given points and the configuration of the points.

The covariance of variables is taken into account if their uncertainties are correlated. The correlation happens if there is correlation in the measurement or if they are correlated across a group. The covariance is determined either analytically or with the use of Monte Carlo estimation technique.

The analysis for the performance of the Kalman filter is tricky. In the derivation some basic assumptions are made. In reality not all of those assumptions hold true. We may not know the initialisation values for the covariances and the state. Therefore, we need to tune these values to find the right value or at least something that works.

For the standard Kalman filter, when a system is observable and controllable the error tends to zero as $n \to \infty$ even with poor initialisation. The issue is that it takes longer for the system to settle if the initialisation is bad.

3.4 Implementing the Kalman filter

In this research we use the Kalman filter for tracking of the camera relative to the pylon. We have provided the theoretical background of the Kalman filter but we have to discuss how the method is implemented and used for our research. Section 3.2 shows the motion model we use for the work, which is the random walk model and the state transition model is the rate of change of the pose.

The observation model is given as registering the world model of the pylon onto the image scene and the Kalman filter is focused on minimizing the distance between the registered world model of the pylon and the measurements. The measurements for the Kalman filter are the vertices from the convolutional neural network in chapter 4. We have to filter the outliers from the measurements and establish correspondence between world points and the vertices.

The initialisation of the Kalman filter is set iteratively for the covariance. For the initialization of the state an iterative means can be used to set the initial state but it means the tracker will take some time before it settles. Alternatively, we can show how obtaining the single image pose estimate using geometric hashing (chapter 5) will aid in the initialization of the system.

Chapter 6 shows two experiments we implement using the extended Kalman filter and vertex detection as the measurement.

3.5 Summary

In this chapter, we discussed the Kalman filter and its use in tracking the pose of a camera. We provided details about the Bayesian derivation of the Kalman filter. Subsequently we discussed using a random walk for the motion model of a camera to see the relationship between the various states. The extended Kalman filter is used for the tracking of the camera in the pylon. We implement the filter to track the six degree-of-freedom pose estimate of the camera relative to the pylon as it moves around.

Chapter 4

Convolutional Neural Network method for vertex detection

The perceptron is the earliest neural network. Neural networks have made tremendous progress since the perceptron. There are different neural networks, ranging from the convolutional neural network (CNN) to the recurrent neural network (RNN). The success of the convolutional neural network [63] on the ImageNet dataset in the 2012 image classification challenge has generated a renewed interest in the field. The achievement from the event has made many researchers use neural networks for different applications.

Convolutional neural networks (CNN) have been successfully used in most aspects of computer vision. The usage has included such tasks as image classification, semantic segmentation, object detection and human pose estimation. A typical CNN has a convolutional filter, a pooling unit, a normalisation unit, an activation function and a classifier. A loss function is formulated and solved through an iterative process using an optimizer. These result in a CNN with the best weights.

In this work, we are concerned with obtaining the vertices of a pylon. To detect the vertices, feature-based techniques (SURF) and a Hough voting technique were used. However, these methods proved ineffective in detecting the vertices. Therefore, we decided to try learning techniques for vertex detection using a CNN. The vertices are keypoints on a pylon and the problem closely resembles keypoint detection for human pose estimation. We borrow from a technique used for human pose estimation to solve the vertex detection challenge. There are various CNN based human pose estimation techniques such as deepercut [64], openpose [65] and

deeppose [66]. This work uses the deepercut method, which has been successful for detecting keypoints in human pose estimation and has been used in other similar tasks such as animal pose estimation [67].

We use the concept of transfer learning for building the network. Transfer learning helps us to use an existing configuration without training a network from scratch. Transfer learning is about taking a model pretrained on a large dataset like the ImageNet and fine-tuning it for a task. For this work we use the ResNet pretrained model. The model has been successfully used in many tasks.

We need to adjust the architecture of the model for the purpose of our task, and we have an inference unit for the whole network. The network is made of different layers of convolutional filters, fully connected layers and a classifier. Vertex detection requires that the output of the network has feature map shape, therefore the deepest part of the network is removed to allow for higher resolution. The deepest part in this case is the average pooling unit, the fully connected layers and the classifier.

Even with the shape of the feature map at the resolution where the deepest part of the network is removed, the input has undergone downsampling and a higher resolution is needed, therefore we have to upsample the feature map. To upsample we use the hole algorithm that increases the shape of the feature map of the fifth layer of the ResNet by putting holes between the kernel of the convolutional filters. Then to further increase the shape of the feature map we add a transposed convolutional filter, which is a filter that helps recover the shape of the feature map of a convolutional filter. The depth of a transposed convolutional filter determines the number of output channels for the network as it is the last component before the loss function. The output from the channels of the network are heatmaps.

The inference is about the prediction from the network that achieves the goal of vertex detection. The inference takes the output from the trained model then applies maximisation of the feature map from a channel. Maximisation finds the pixel with the highest probability score in the heatmap as the probable vertex. Thereafter, all the heatmaps from each channel are concatenated into a single heatmap.

Two datasets are generated to test the idea and produce a ground truth heatmap for the training of the network. The data are generated from different viewpoints of the pylon and with different image backgrounds. The pylon has N vertices

on it, and each vertex is assigned a unique label. Opposite sides of the pylon look identical, therefore a marker is placed as a means of distinguishing the sides. There are N ground truth heatmaps, one for each of the vertices. This means the transposed convolutional filters are N in number and the output of the network will be N until concatenated into a single heatmap.

The data is pre-processed and applied to the network. The network then consists of the loss function, a pretrained model (the backbone network), the activation function and the optimizer.

The data is comprised of images split into a training and a testing dataset. We initialise the learning rate and adjust it according to the loss.

The performance of the network is evaluated by comparing the Euclidean distance between the ground truth vertices and the predicted vertices for the accuracy rate. A channel is expected to be the same for the ground truth and the predicted vertices.

The successful implementation of the vertex detection will serve as the feature representation to be used alongside other techniques as part of a localisation framework for pose estimation and tracking.

In subsequent sections we discuss some of the fundamentals of the CNN, the problem description, network architecture, data preparation, training details and analysis of the work.

4.1 Problem description

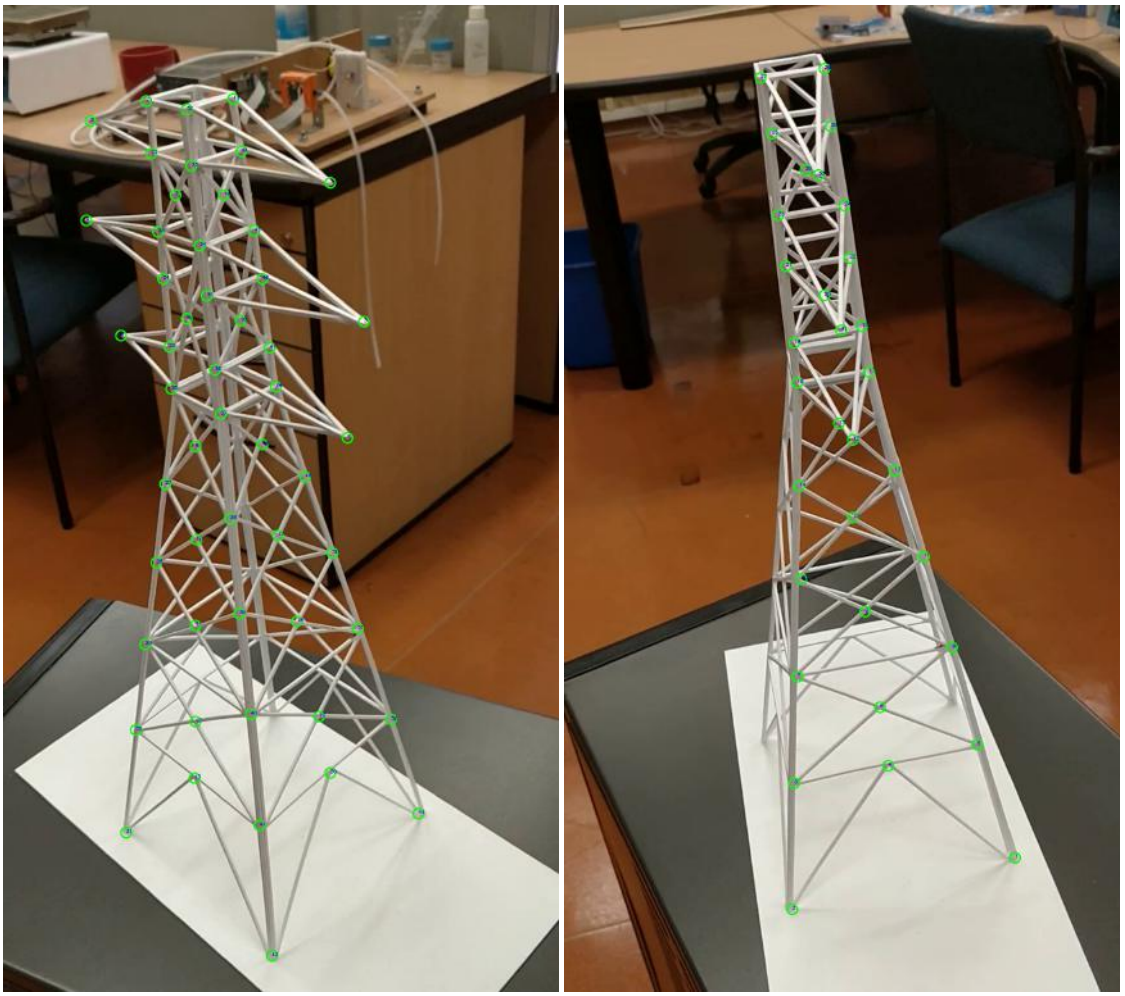


FIGURE 4.1: Illustration of the vertices labelled on different views of the pylon.

Keypoint detection is about finding the salient points on an object to enable a match between a model and an image scene. These salient points on a pylon are the vertices. Vertices will aid in the recognition and pose estimation of a pylon. We hypothesise that a neural network can learn to detect the location of vertices on a pylon.

In Figure 4.1 we show a pylon and the vertices on it from two different views. Pylons are symmetrical objects with many identical vertices on different faces.

During the course of this work, we attempted feature-based techniques on the pylon. An example was using the SURF method and the result is shown in Figure 4.2. The SURF method and other methods such as the SIFT proved to be insufficient when combined with our localisation framework.

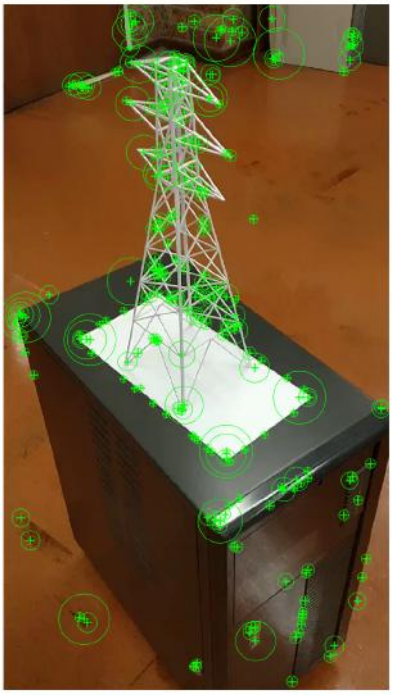


FIGURE 4.2: Vertex detection using SURF method.

Furthermore, we tried a Hough technique based on the work of Olga et al. [68]. We generated line segments using LSD. The endpoints of the line segments were our voting elements and the intersection of the line segments served as our hypothesis. Then we used a probabilistic voting technique to vote for the vertex. The vertex detection from the work also proved insufficient in our localisation framework.

This leads us to attempt to learn the vertices with the use of convolutional neural networks.

4.2 Background

In this section we discuss some components of the convolutional neural network. These are ideas that relate to the implementation of the network in our research especially as it relates to the adjustments we made to the network to make it better suit the task.

The network we use for the work is a pretrained network. Pretrained networks are models that have been trained on large datasets and have performed very well. Training a model is computationally expensive and requires a lot of hardware.

These models are trained on datasets such as ImageNet and PASCAL and can be fine-tuned for any particular application. Therefore, we revisit the ResNet as we use it build our model.

4.2.1 Convolutional Filter

Convolution is a type of mathematical operation that helps us to combine signals. In the frequency domain, it is the multiplication of two signals.

In the spatial domain, the convolution operation for discrete systems such as images is shown as

$$O(i,j) = \sum_a \sum_b I(a,b)W(i-a, j-b). \tag{4.1}$$

The variable $I(a,b)$ is the input of the operation in our case an image. The variable $W(i,j)$ is the kernel which is the convolutional filter and $O(i,j)$ is the output of the operation called the feature map.

The use of convolutional filter means fewer weights to learn for the model. Therefore, the model will require smaller memory space.

In real terms, the convolutional operation is specified as the size of the image, the channels and the batch size. The depth of the output is equivalent to the depth of the filter.

When we combine the convolutional filter and the image, we set out to extract the features of the image. In the CNN framework we are going to learn the weights of the convolutional filter instead of using hand-engineered filters like the Sobel filter.

4.2.2 Pooling

Pooling is used to achieve translation invariance over small spatial shifts in the feature map. The application of pooling means that a small translation in the input does not change the output. Invariance does not care about location but about the presence of a property. It aims to summarise a neighbourhood with important statistical information such as the average, median or maximum. It

helps in reducing the feature map and allows only the important information to be transmitted to the next layer.

4.2.3 Transposed Convolutional Filter

The transposed convolutional filter is also referred to as the deconvolutional filter or fractionally-strided convolutional filter in popular CNN frameworks. This is due to the fact that the filter helps recover the shape of the convolutional filter. It really does not do the inverse of the convolutional filter, but instead upsamples the data to obtain the shape of the input.

The operation is called transpose convolution because it uses the transpose of a convolution matrix C_m gotten using toeplitz matrix [69]. Thus, the transpose convolution operation is the multiplication of the transpose of a convolution matrix and the input as $O = C_m^T \times I_m$.

The difference between the convolutional filter and the transpose convolutional filter has to do with the mapping of the input to the output. In the convolutional filter the mapping is many-to-one, while for the transpose convolutional filter it is one-to-many. The convolutional filter downsamples the feature map, while the transpose convolutional filter upsamples the feature map. Therefore, the transpose convolutional filter upsamples the feature map and aids to recover the shape of a convolutional filter to achieve a higher resolution.

4.2.4 Hole algorithm

In most CNN architectures there is a downsampling of the data with the convolutional filter. This is ideal for classification tasks but is not desirable for segmentation and pose estimation tasks.

To retain the shape of the data, or recover it, the hole algorithm can be used. The hole algorithm, sometimes called dilated convolution, is a technique borrowed from wavelet theory that is used for neural networks. For a convolutional filter a hole is added between the values of the filter at a specific rate. It increases the output units without increasing the kernel size. The rate at which the hole is added is called the dilation rate. The dilation rate tells us about the spacing between each pixel in the kernel of the filter.

Using the hole algorithm means increasing the field of view, which translates to increasing the receptive field.

4.2.5 Transfer learning

Transfer learning is a technique where the weights and network architecture of a model learnt on a certain problem can be applied to another problem. It helps us to use models that were trained on very large datasets with expensive computational cost, and fine-tune the application to other tasks. Therefore, we can fine-tune a model designed to do classification to do keypoint detection or segmentation.

Transfer learning is possible if the features used are general [70]. For instance, the early layers of most classification networks are general and the average pooling and fully connected layers are specific for classification tasks only.

Transfer learning ensures we can use less time, better starting points and less data for the training tasks. It should be noted that with too little data, it may overfit.

4.2.6 ResNet

The deep residual network (ResNet [71]) is a network pretrained on the ImageNet dataset. It is a deep model and has proven to be successful when used on other applications. There are different ResNets with different number of layers as shown in Table 4.1. In many instances the depth of a network improves its performance, but as the network starts to converge there is degradation in performance. The ResNet attempts to address the problem by re-introducing the input at each layer.

The ResNet has five different blocks with each block having the same output size as the image. After the blocks, the input is passed through an average pooling and fully connected layer made up of a classifier.

TABLE 4.1: ResNet network architecture.

Layer name	Output size	18-Layer	34-Layer	50-Layer	101-Layer	152-Layer
conv1	112 × 112	7 × 7,64, stride 2				
conv2	56 × 56	3 × 3 max pool, stride 2				
		$\begin{matrix}3\times3,64\\3\times3,64\end{matrix} \times 2$	$\begin{matrix}3\times3,64\\3\times3,64\end{matrix} \times 3$	$\begin{matrix}1\times1,64\\3\times3,64\\1\times1,256\end{matrix} \times 3$	$\begin{matrix}1\times1,64\\3\times3,64\\1\times1,256\end{matrix} \times 3$	$\begin{matrix}1\times1,64\\3\times3,64\\1\times1,256\end{matrix} \times 3$
conv3	28 × 28	$\begin{matrix}3\times3,128\\3\times3,128\end{matrix} \times 2$	$\begin{matrix}3\times3,128\\3\times3,128\end{matrix} \times 4$	$\begin{matrix}1\times1,128\\3\times3,128\\1\times1,512\end{matrix} \times 4$	$\begin{matrix}1\times1,128\\3\times3,128\\1\times1,512\end{matrix} \times 4$	$\begin{matrix}1\times1,128\\3\times3,128\\1\times1,512\end{matrix} \times 8$
conv4	14 × 14	$\begin{matrix}3\times3,256\\3\times3,256\end{matrix} \times 2$	$\begin{matrix}3\times3,256\\3\times3,256\end{matrix} \times 6$	$\begin{matrix}1\times1,256\\3\times3,256\\1\times1,1024\end{matrix} \times 6$	$\begin{matrix}1\times1,256\\3\times3,256\\1\times1,1024\end{matrix} \times 23$	$\begin{matrix}1\times1,256\\3\times3,256\\1\times1,1024\end{matrix} \times 36$
conv5	7 × 7	$\begin{matrix}3\times3,512\\3\times3,512\end{matrix} \times 2$	$\begin{matrix}3\times3,512\\3\times3,512\end{matrix} \times 3$	$\begin{matrix}1\times1,512\\3\times3,512\\1\times1,2048\end{matrix} \times 3$	$\begin{matrix}1\times1,512\\3\times3,512\\1\times1,2048\end{matrix} \times 3$	$\begin{matrix}1\times1,512\\3\times3,512\\1\times1,2048\end{matrix} \times 3$
class	1 × 1	average pool, 1000 fc softmax				

4.3 Method

The objective is given an input image that contains a pylon, we want to detect all the vertices on the pylon. We train the network using annotated vertices in images and obtain the best weights. An input RGB image is passed through the trained network and the feature map is passed through an inference unit for the prediction of vertices. We provide the procedure for performing the vertex detection. We discuss the means used to build the network architecture including its components and details. We also describe the inference for the detection process as the final output for the prediction.

4.3.1 Network Architecture

Network architecture is a way of combining the various components of a CNN in order to achieve the desired result. Our network is a combination of a modified ResNet pretrained on ImageNet and two upsampling techniques to provide for the shape of the feature map.

The ResNet has five different blocks with varying numbers of convolutional filters. In Table 4.1 we provide the ResNet configurations with various depths.

We discard the fully connected layers in favour of retaining higher resolution feature maps at the deepest output.

The average pooling and fully connected layers downsample the feature map to be able to be used as a classifier. For keypoint detection, we need to have the shape of the feature map therefore we remove the average pooling and the fully connected layers. A hole algorithm and a transposed convolutional filter is applied to upsample the shape of the feature map with the same number of channels as the number of keypoints.

The output stride is the ratio of the input size to the output size. For a fine-grained localisation task we want the output stride as small as possible, which means the output size is close to the input size thereby avoiding loss of details through resizing.

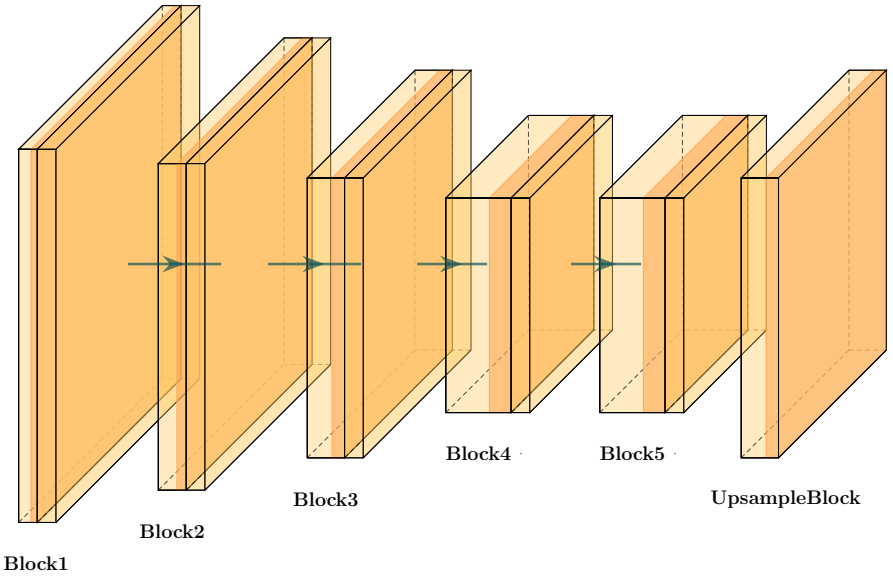


FIGURE 4.3: An illustration of the ResNet and the upsampling blocks as the network architecture of our vertex detection.

The output stride of a ResNet after the fifth block is 32 pixels, but for a fine-grained localisation task the output stride seems to be large as reported in [64]. Therefore, we set out to achieve an output stride of 8 pixels.

Using the hole algorithm, we reduce the stride to 16 pixels. The hole algorithm is applied to the fifth block of the ResNet where the 3×3 convolutional filter is added with holes in between the kernels. The stride of the initial convolutional filters in the fifth block are changed from 2 pixels to 1 pixel. This ensures there is no downsampling of the feature map at the fifth block.

Furthermore, we use a transposed convolutional filter to take the output stride to 8 pixels. A 3×3 transposed convolutional filter with a stride of 2 pixels is used to upsample the feature map to achieve the desired shape.

Figure 4.3 shows the shape of the feature maps of each of the layers of the network. The figure shows the ResNet and the upsampling unit.

4.3.2 Loss

The network produces two outputs to be fed to the loss when training the network. The first output regresses the probability of each of the vertices and is computed using a softmax. The loss of the softmax is given L_{key}.

The second output regresses the probability of the offsets. The offsets are used for location refinement to improve the precision of the location of the vertices. The offset loss is given as L_{off}

$$L = L_{\text{key}} + L_{\text{off}} \tag{4.2}$$

The general loss is given as the summation of the softmax loss and the offset loss [64]. The problem is formulated as a multi-label classification with each location having a set of probability distributions estimated for each vertex.

4.3.3 Inference

The desired goal is the detection of all vertices on a pylon. The output from the upsampling unit is a feature map, which is the heatmap of the probabilistic scores obtained from the training of the model. The location with the maximum probability score in a specific heatmap of a channel of an upsampling unit is considered as the vertex. Then all the heatmaps of the upsampling units are summed together to provide a single heatmap for the vertices.

Figure 4.4 shows the pipeline an input follows to produce the vertex output with the network and the inference shown. The network is the aspect used during the training stage and the details are described in section 4.3.1. At the prediction we take an input through the trained model and the inference unit to generate an output, which is the vertices.

4.4 Experimental details

In this section we provide details pertaining to the implementation of the algorithm, the preparation of the data, and a comparison of the algorithm to other sets of algorithms. We discuss the implementation details for the network and provide

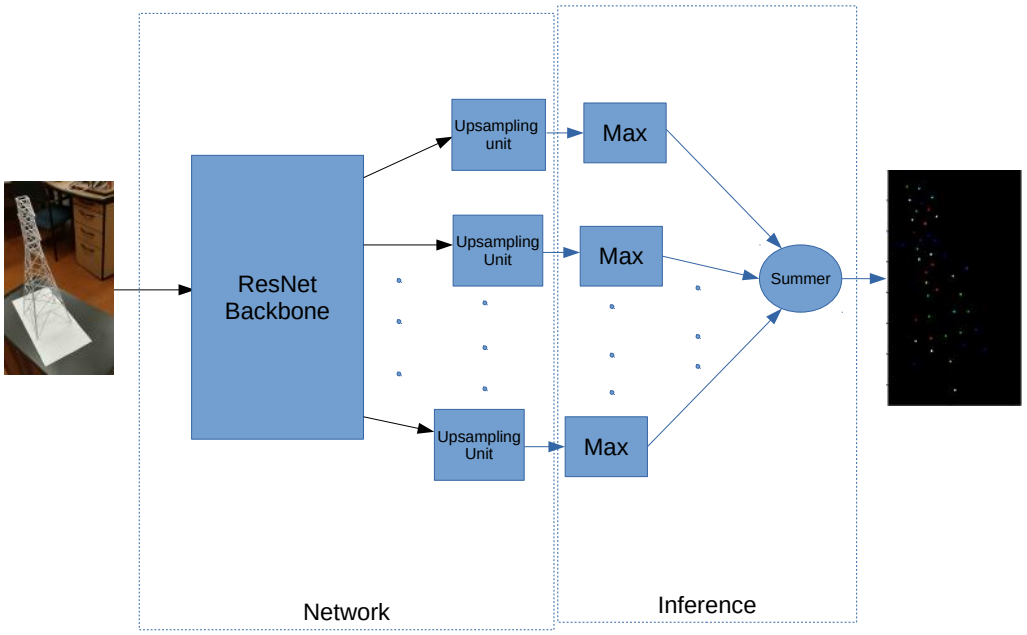


FIGURE 4.4: An illustration of the network architecture and the inference for the output of the vertex detection.

results. The details we provide are for the various parameters set and the outputs from the network. The output of the network is a heat map from each of the channels and the maximum probability score is selected as the probable vertex. When training the network, a channel is always given the same vertex label with the expectation that the channel will learn that vertex.

There are different ResNets with differing number of convolutional filters, and for these experiments we used the ResNet 50 similar to its usage in the deeplabcut paper [67]. In Table 4.1 we show the various ResNet configurations and in Section 4.3.1 we explain the adjustment made to the network for vertex detection. We are able to obtain satisfactory results with the ResNet 50.

Training is done with a sigmoid activation and a cross-entropy loss function. Stochastic gradient descent is used as the optimizer.

4.4.1 Evaluation with a prototype pylon

We made an attempt to obtain publicly available data, but at the time of writing this dissertation, we could not find well-labelled data that suits our tasks of keypoint detection and pose estimation. Therefore, we build a small pylon in the laboratory to test the algorithm and perform sets of experiments to validate the usage of the algorithm.

Data Preparation

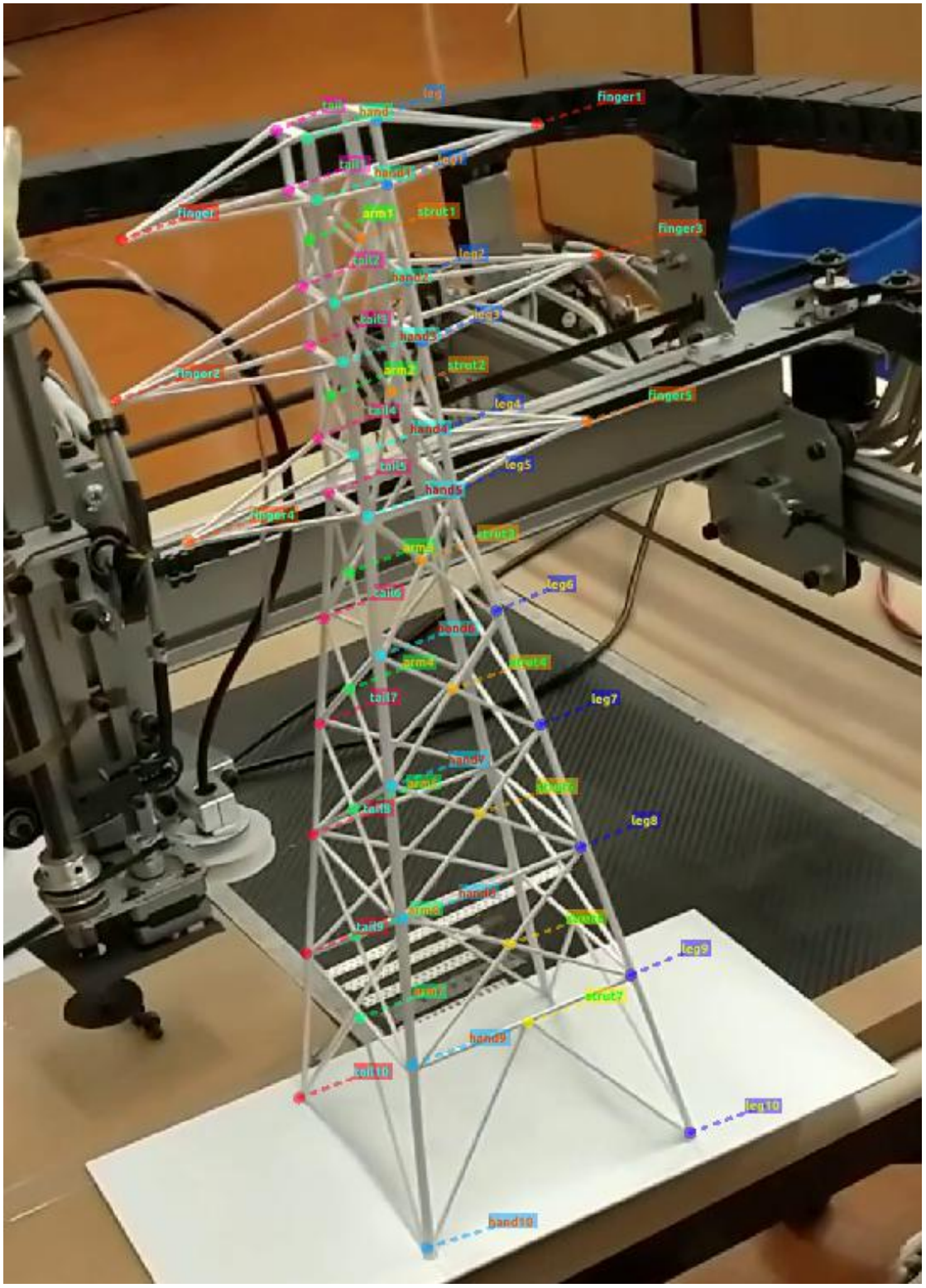


FIGURE 4.5: An example of a pylon image with labelling of the vertices. Each of the vertices is given a specific name and label.

The pylon is a wiry object with lines; the lines intersect at crossings which are the vertices of the pylon. The vertex points look identical with two or more lines meeting, and the faces of the pylon are symmetrical and similar. To label the images, we uniquely label each of the vertices of the pylon. We label the front side of the pylon fully. The back side was not labelled because it is considered as an occluded section of the pylon. Each vertex has a unique label. Figure 4.5 provides an example of the pylon with the labelling of vertices shown.

FIGURE 4.6: Ground truth heatmap for a channel in the network.

Using the labels on the pylon, a ground truth heatmap can be generated. All the vertices with a label are considered to be valid channels and a value is assigned within their heatmap. Vertices without a label are considered to have all pixels as zero. The ground truth heatmap for the keypoint detection has a number of channels equivalent to the number of vertices. The heatmap shape is the same as the shape of the pre-processed image. Each channel in the heatmap represents a vertex. For a particular channel, the location of the vertex is obtained and assigned an intensity value of 1. Every other location is assigned an intensity value of 0. This is shown in Figure 4.6. An output stride has to be chosen in such a way that the image does not lose detail and a fine-grained localisation can be obtained. Therefore, the heatmap has to be scaled using the output stride value.

In order to input an image to the network, the image has to be preprocessed. Image preprocesssing involves tasks such as scaling, random cropping and center cropping.

Memory is a concern when it comes to training a network, so we have to scale the images. Scaling the image to a smaller size helps reduce the training time required as the images are sent in batches.

Cropping is used as a way of reducing the contribution of the background to the network. It allows the network to see the neighbouring regions as background and facilitates better localisation.

The model is trained using a ground truth heatmap. The heatmap has the size of the input image resized with the output stride and a depth equivalent to the number of vertices. The input images were resized to 240 $\times$ 135 pixels.

We have 200 images with 78 unique vertices in each image. We understand the number of images is small for typical training of a CNN, but in this situation we have a single object and the variation is not high. Each vertex on the pylon is assigned a channel in the network. The dataset is divided into a ratio of 75% for the training and 25% for the testing stage. Additionally, in Section 4.4.2 we revalidate the results with more data.

We use transfer learning for training the network, which gives us a better starting point for the weights. The network used 1.03M iterations for the training, which is equivalent to 3433 epochs.

Discussion and Analysis

In Figure 4.7 we provide the loss performance and the learning rate used for the training. The learning rate starts at a high value and gets lower as the network starts converging.

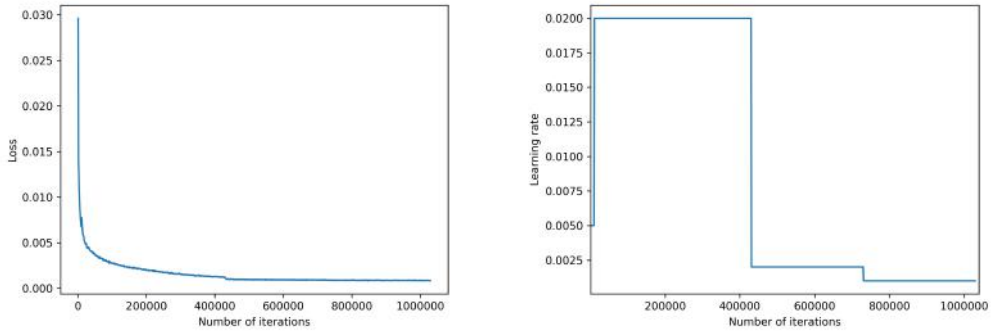


FIGURE 4.7: The loss and the learning rate for the training phase of the 78 vertices.

Each pixel in a heatmap with maximum probability score represents the vertex from the channel. The maximum probability scores obtained from each of the channels are concatenated into a single heatmap by summing their heatmaps, which provides the whole collection of predicted vertices. Figure 4.8 provides the predicted heatmap and shows the heatmap superimposed onto an image for some randomly selected images.

The network output has the number of channels equivalent to the number of vertices. For instance in Figure 4.9 we have an output from a single channel before the maximisation and the summation. We see the probability score on three different points in the heatmap. All the three points represent vertices but only a single vertex is meant to be detected in the channel. This is the reason for the maximisation at which point the score with the highest probability is considered the detected vertex. The vertices are very similar with four lines intersecting for each with the main distinguishing element as the location. This shows the difficulty associated with learning the vertices.

We check the accuracy of the detection technique in correctly identifying the vertices. To achieve this, we compare the Euclidean distance between the ground truth and the predicted vertices. We call this metric the channel-for-channel approach. The equation for the channel-for-channel metric is given as

$$\text{channel-for-channel metric} = \frac{1}{N}\sum_{X\in\mathcal{N}} ||\hat{X}_i - X_i)||, \tag{4.3}$$

with i as the channel, $\hat{X}_i$ as the predicted vertex, X_i as the ground truth vertex, N the total number of vertices and $\mathcal{N}$ is for the whole vertices. If the Euclidean

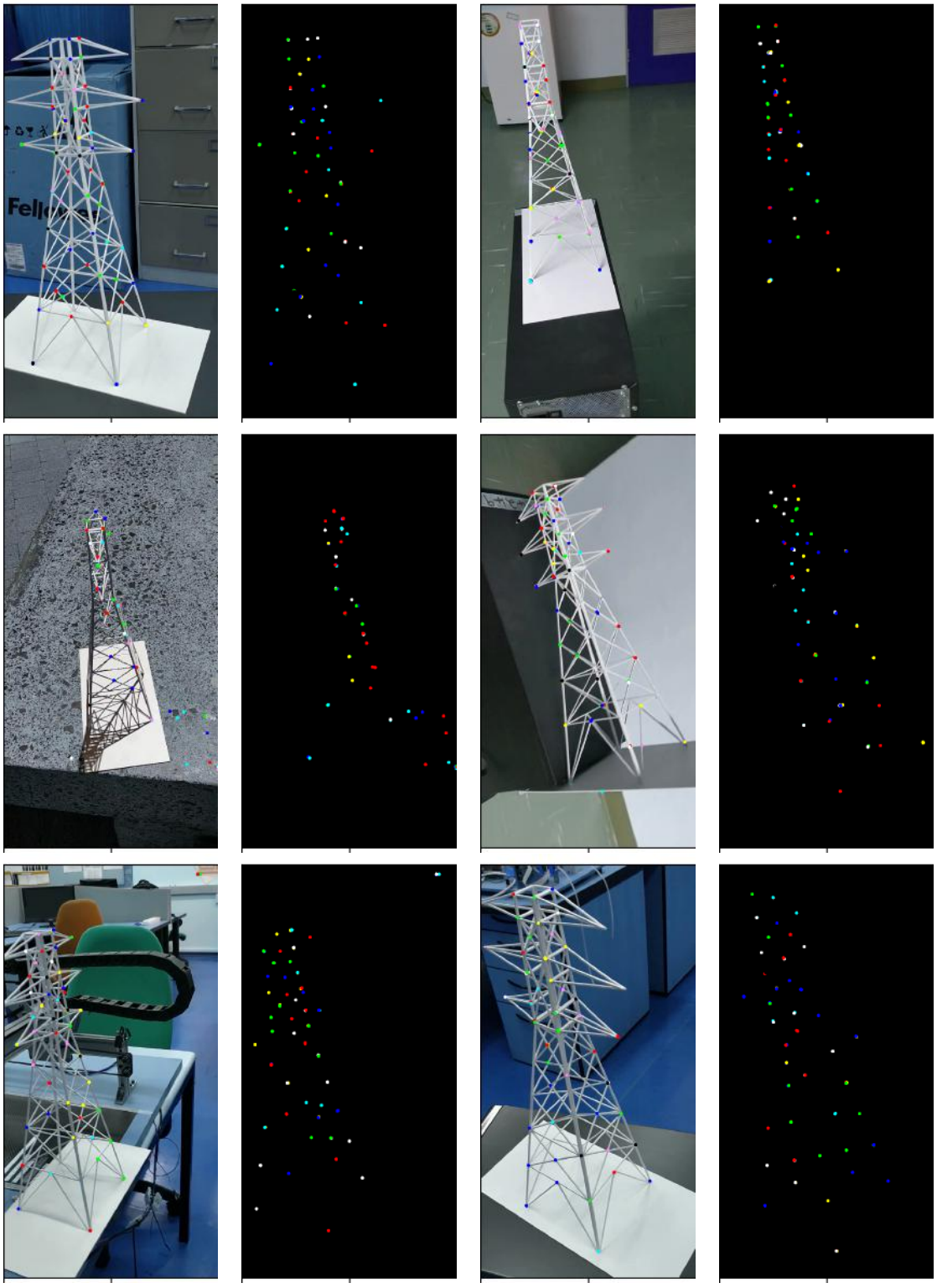


FIGURE 4.8: Output of the neural net shown as a heatmap and the heatmap with 78 vertices superimposed onto images.

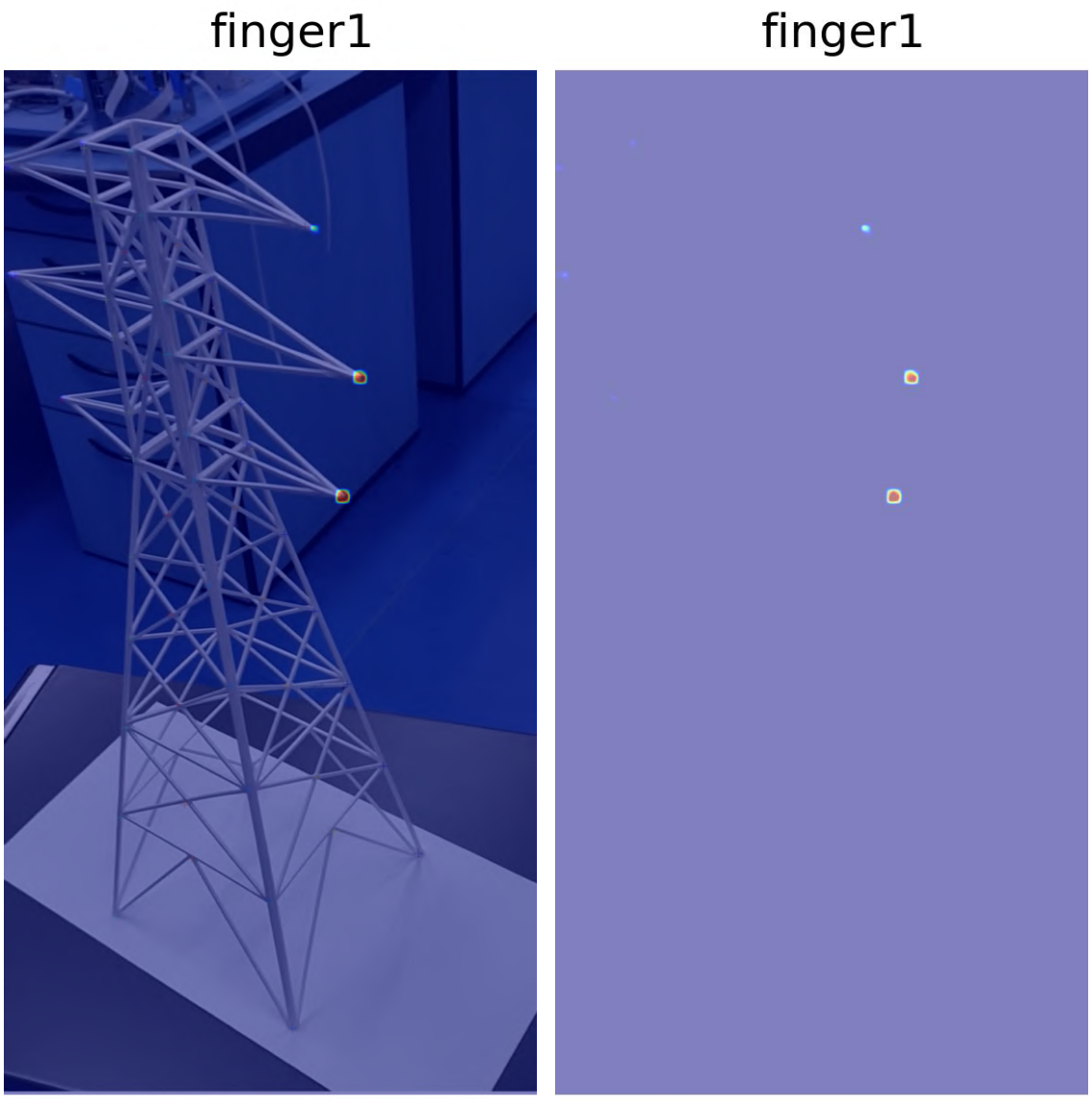


FIGURE 4.9: Heatmap from one of the channels.

distance is less than 10 pixels, we consider it an accurate match. The expectation is if a channel is given a ground truth vertex named finger, the predicted vertex from the channel should also detect a unique identical finger. These two channels should represent exactly the same location. Figure 4.10 provides the accuracy rate for the test images. The average accuracy rate at the 10 pixel tolerance is obtained as 67.55%.

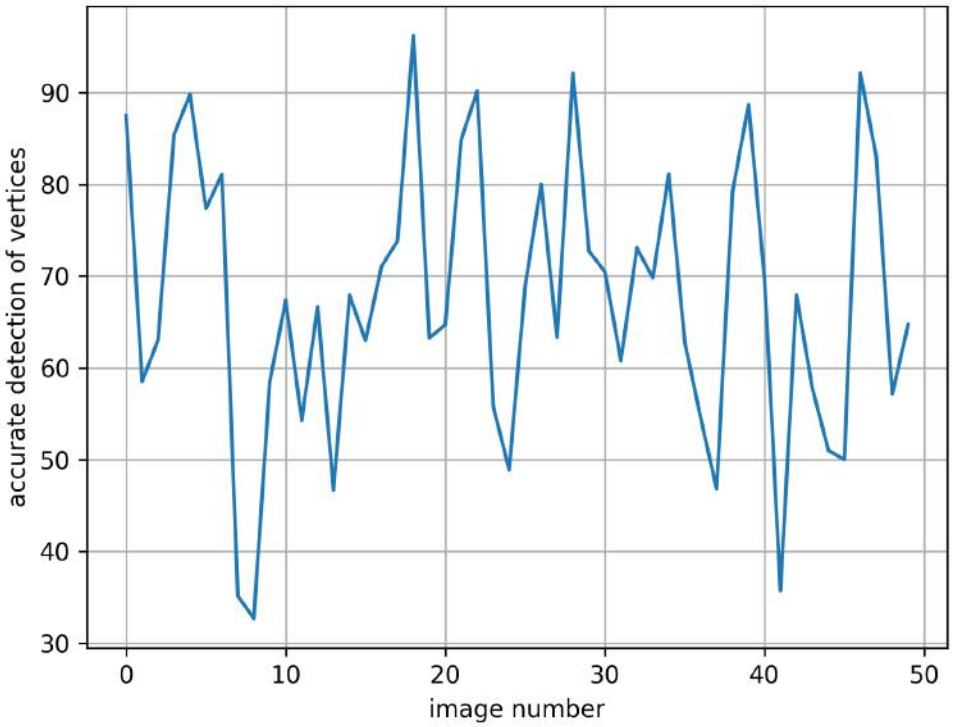


FIGURE 4.10: Accuracy rate for prediction vertices having exact channels as the ground truth labels.

The probability score provides the level of confidence in correctly detecting the vertices. Figure 4.11 shows that the probability score increases as the accuracy rate decreases. This decreases the usable number of vertices for the localisation framework.

The network output has 78 vertices in total and it outputs that exact same number. The labelling is done only for the front face of the pylon. Scrolling through the images, only a maximum of 53 vertices in any single image is seen with other images having the same number or fewer vertices on the front face of the pylon. In Figure 4.11 we see that if the probability score is 0.4 or less, then the number of vertices detected is more than the number of labels. This means the network wrongly predicts vertices at such a low score.

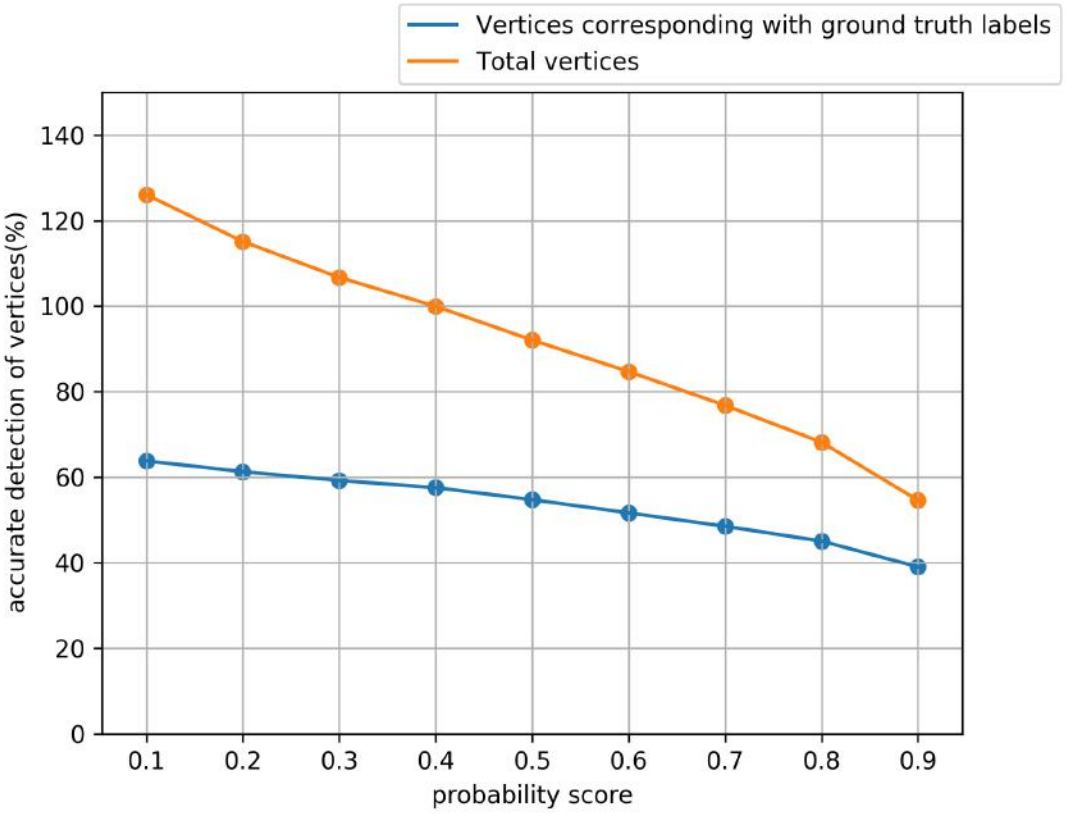


FIGURE 4.11: Accuracy rate for predicted vertices with various probability scores.

The similarity in the vertices causes the convolutional neural network to mislabel some vertices. The neural network predicted a vertex at a location which is recognized as a legitimate vertex but assigned it to the wrong output channel of the network. We compare the predicted vertices to their nearest ground truth vertices instead of comparing the ground truth and predicted vertices from similar channels. Therefore we use another metric to check for the accuracy rate, which we call the nearest neighbour approach. The equation for the nearest neighbour approach is given as

$$\text{nearest neighbour metric} = \frac{1}{N} \sum_{X_1 \in \mathcal{N}} \underset{X_2 \in \mathcal{N}}{min} ||X_1 - X_2||. \tag{4.4}$$

The metric finds the nearest neighbours to labelled vertices before obtaining the average Euclidean distance. Ground truth vertices are compared to the closest predicted vertices and if they are within 10 pixels in Euclidean distance they are considered an accurate match. Figure 4.12 shows the accuracy rate for the test images. An average accuracy rate of 84.84% was obtained at an Euclidean distance

threshold of 10 pixels. We see that the nearest neighbour approach has a higher accuracy rate.

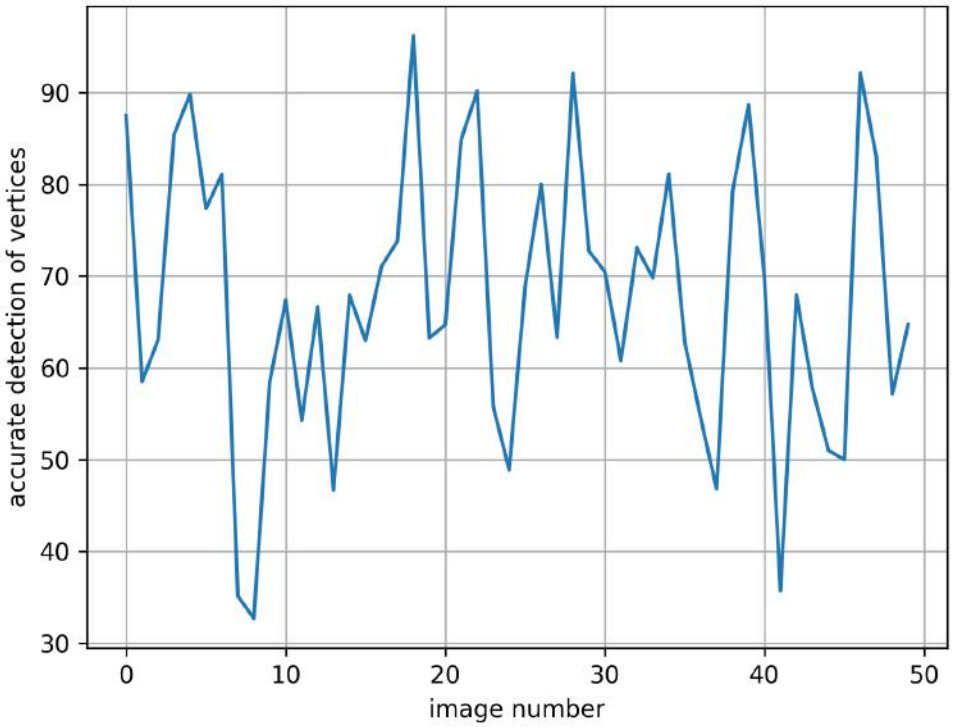


FIGURE 4.12: Accuracy rate for the prediction vertices nearest to the ground truth vertices.

The difference in accuracy rate for the results shows that the structure of the pylon confuses the neural network. The symmetry of the pylon and the similarity of the vertices means that a channel assigned to a specific vertex will detect a vertex but not for the exact same ground truth label. When results of neural networks with similar architecture were reported in the work for human pose estimation [64] and animal pose estimation [67], such observations were not made. We believe that this is due to the difference in the structure of the objects. Although in the six degree-of-freedom pose estimation works using neural networks [53, 54] such observations are made and it necessitated the use of loss functions with a nearest neighbour component for symmetrical objects.

4.4.2 Evaluation with outdoor pylons

We conducted further experiments on real world outdoor pylons to validate our proposed vertex detection technique. An electricity pylon is a lattice steel structure with crossings on the pylon. Our approach detects these crossings, and we focus on detecting the major vertices on the pylon. The pylon has four sides, with two sides of it identical by symmetry. Since two sides are the mirror of the other we label only the front faces of the pylon.

Data preparation

We need to pre-process the data to maximise memory usage and provide rich quality information to the network. We crop the images around the center of the pylon to bring it into focus and reduce the background clutter. Afterwards, we resize the image to be able to fit into the network. The image size after processing is 800×800 pixels. We use a large image size so that all the vertices can be within the view and there will be a distance separation between the vertices.

The dataset consists of 1020 images where we divided the data in a ratio of 70, 20 and 10 for training, validation and testing respectively. In Figure 4.13 we see some samples of the data. The data are captured with varying illumination, different intensities, and at different angles and views. For the training data, we performed data augmentation techniques to increase the amount of data and add to the diversity of the data. The data augmentation we did include random cropping around the keypoints area and rescaling of the data. The random cropping ensures the information around a keypoint is brought into full focus and allows the neural network to assess the rich details. Additionally, we use rescaling of the data to reduce the memory needed and ensure the input images are all of the same size.

Discussion and analysis

There are 99 keypoints on the pylons, so the output of the network has 99 channels. Each channel represents a vertex on the pylon. The number of the vertices selected are the larger vertices that can be identified on the front face of the pylon.

We present the results of the experiment that we evaluated on the test set. We compare different techniques of obtaining vertices on the pylon. For the accuracy

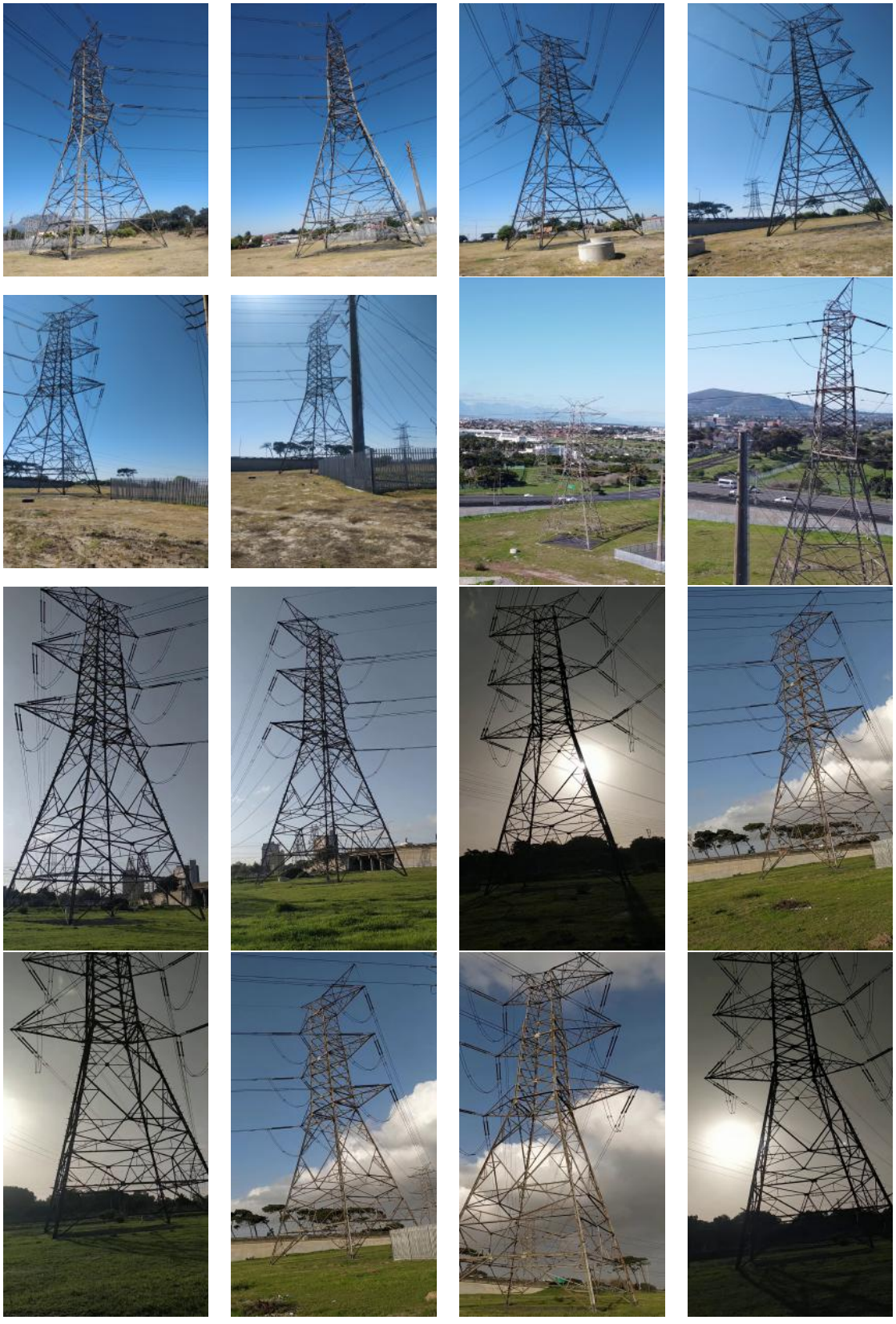


FIGURE 4.13: Some images from the outdoor pylon dataset.

rate we use the nearest neighbour metric, where the Euclidean distance between the detected vertex is compared to the ground truth vertex.

TABLE 4.2: Comparison of accuracy rate for keypoints.

Method	Percentage
Ours	94.94
Harris corner detector	62.18
ORB	48.48
Adapted Hough Transform	14.86

Our technique performs with better accuracy than the rest of the techniques outlined in Table 4.2. The techniques tested are the Harris corner detector, ORB and an adapted Hough transform. The Harris detector has a minimum distance of 10, pixel threshold of 0.02 and window size of 13. We use an ORB with 1000 features. As presented in Table 4.2, the handcrafted feature detection techniques, which are the Harris corner detector and ORB tend to perform poorly in finding the vertex. Additionally, the Hough Transform uses the endpoints of a line segment detector to find the vertices. The endpoint of a line segment is the voting element and the intersection points of the line segments as the hypothesis space. The detection of the vertex is done when a voting element is generated within the hypothesis space. Every voting element within the neighbourhood contributes to the total votes, and the peaks of the summation are the valid hypothesis. To find a most suitable threshold for the Hough technique is difficult and the method comes with a huge amount of outliers. To test for this experiment, we use a threshold of 10.

Figure 4.14 shows the images of some pylons and the detection of the vertices using different techniques. The images show that the neural network was able to best detect the vertices.

4.5 Vertex detection as part of the localisation framework

In Chapter 6 we use the vertex detection output to solve for pose estimation and tracking.

.

FIGURE 4.14: Detection of vertices using different techniques.

In the first experiment of Chapter 6 we use vertex detection including labels for the pose estimation and tracking. With the understanding gained from the channel-to-channel accuracy rate metric, each vertex is assigned a label, for instance the vertex from channel 44 is assigned to the point (0,0,0). The high probability score threshold is used to filter out unusable vertices. In most cases the higher the probability score the better the vertex detection, although we noticed a few mislabelled vertices even at a probability score of 0.9.

In the second experiment of Chapter 6 we use the vertices obtained from a detection process, but without the label estimates. As we have seen with the nearest neighbour accuracy rate metric, comparing prediction and the ground truth vertices with closest points produces a higher accuracy rate. Therefore, we ignore the unique labels assigned to the vertices. We use vertices as part of an initialization step, which is a search-based algorithm to obtain a single image pose estimate. For the tracking, we had to use model fitting and outlier rejection to filter out the outliers and to establish correspondence between the inlier vertices and the world pylon points.

In the third experiment of Chapter 6, to eliminate the need for model fitting and outlier rejection we track the camera using a heatmap without the maximisation of the inference unit. A gradual adjusting of the pose estimation was done using a gradient descent technique.

In the fourth experiment of Chapter 6 we use the vertex detection that we trained with real world pylon data to revalidate the effectiveness of pose estimation and tracking algorithms.

4.6 Summary

In this chapter, we discussed the convolutional neural network and its use for vertex detection. The vertex set is the feature representation of the pylon used as part of the framework for pose estimation and tracking. Data was successfully prepared and fed to the convolutional neural network. The vertices were successfully detected by the network, and the output of the network was measured and shown to provide useful results. The successful implementation and results obtained from the convolutional neural network serves as the keypoints to achieve the overall goal of pose estimation and tracking in Chapter 6.

Chapter 5

Geometric Hashing Based Image Matching

Geometric hashing is a voting technique formulated by Shwartz, Lamdan and Wolfson [72]. As a search-based technique, it uses the local features of an object for recognition. The method needs a way of representing salient characteristics of the object, namely features. It has been used with various modifications in many fields. These features have been tried for points [73], lines [74], and curves [73].

The technique has been used for 2D detection and also for 3D detection. Gavrilla et al. [75] tried the technique for 3D detection by taking images at various angles and storing the model for the training. The technique has been used in other fields such as protein matching [76].

Image matching requires registering a model of the object onto a scene image and obtaining a match with the model. By successfully matching the image scene we establish correspondence with the model. Assuming we have a computer aided drawing (CAD) model and we want to match it to an object in the image scene, we can use a matching technique like geometric hashing, which is a model-based technique. The matching helps in detection of the object and obtaining the pose estimate between the model and the object. In this chapter we discuss how geometric hashing as a matching technique is used to recognize the pylon and we provide details of the successful implementation of the method.

After extracting the features on the pylon using a method like vertex detection (Chapter 4), geometric hashing establishes a match between objects to obtain the detection and solve for the single image six degree-of-freedom pose estimate.

Geometric hashing will help us to match objects between the stored model and the image taken. Thus with the model of the object we can register the object onto an image. The model can be a CAD model, a mesh or an architectural drawing of the object.

The algorithm works on ordered pair features called a basis set. The basis set is the minimum number of features for transformation of an object. For instance, three features are required for an affine transformation so the basis set for an affine transformation has three features. All possible basis sets of each model in the training stage are obtained and stored in a hash table. The hash table helps for a fast search. The model in this work is the face of a pylon.

If a pylon exists in an image scene, we find all the possible basis sets in the image. From the collection of basis sets obtained we arbitrarily pick a basis set. We use the basis set to determine a transformation. Then the transformation is used to transform the other features. The features are used as voting elements for the model and basis set in the hash table. The model and basis set with the highest vote is equivalent to the scene basis set and a match has been found. Therefore, the object has been recognised using the algorithm.

Storing and retrieving the model and basis set in the hash table means there is degradation of values. We experiment to see the effect on the accuracy rate. The measurement of features is noisy and we want to understand how this affects the accuracy rate. The number of possible basis sets in the image scene is large and we experiment on the possible number of random basis sets needed before a match can be obtained. This chapter gives insight into the operation of the algorithm and serves as a guide for its usage in six degree-of-freedom pose estimation in Chapter 6.

There are different approaches to implementing the voting technique of geometric hashing. Some of the approaches are Bayesian [74, 77], and others are non-Bayesian [72, 75]. The values each element adds to the accumulator are different. Some methods make a contribution with the same values and others weight the contribution using Euclidean distance.

In the rest of the chapter we provide more details about the background theory of the algorithm. Section 5.1 discusses the structural ideas behind the method. We present pseudo-code and explain further some of the important concepts in the algorithm. The basis set is a central idea used both in the training and the recognition phase and is discussed in Section 5.2. Section 5.3 discusses voting as a way of identifying the match. Verification is the post-processing stage to ensure the accurate candidate is found and is discussed in Section 5.4.

The algorithm is affected by noise, and we give details in Section 5.5. In Section 5.6 we discuss the complexity of the algorithm. Section 5.7 we provide experimental results about factors such as the transformation type, performance under noise, the effect of quantization and hash bin size affecting the algorithm. Finally, in Section 5.8 we do an experiment to determine the single image pose estimation of an object when given the 3D model using geometric hashing.

5.1 The algorithm framework

Geometric hashing is a model-based technique that allows us to match and localise objects. It involves normalising the representation of an object to a canonical reference frame, and trying to align it with the search object. The algorithm is implemented in two stages, namely the training and the recognition stage. Training involves extracting features and storing them in a hash table, and it mostly occurs offline. The features are extracted in the recognition stage, similar to the training stage. The features are used to retrieve the stored information in the hash table, and a voting technique is used to find a match.

Assume we have a model with the feature points $p0, p1, p2, p3, p4$ represented in Figure 5.1. The model is assumed to undergo rotation, translation and scaling, which is the similarity transformation. Similarity requires two points for the transformation to happen and these two points we call the basis set.

Suppose we take points $p2, p3$ as the basis set and transform them to a new location (-0.5,0) and (0.5,0) in the canonical reference frame. We use the same transformation to map the rest of the points. The new locations for the points are quantized into bins.

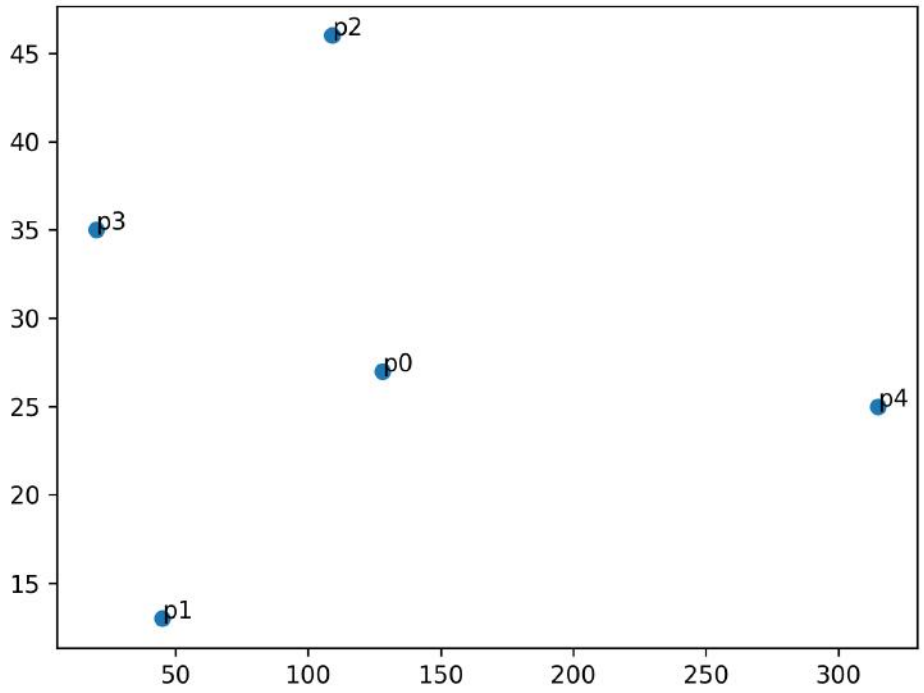


FIGURE 5.1: Five feature points representing a model.

Therefore, the quantized value (Figure 5.2) serves as the index of the hash table. There are three points so there will be three entries in the hash table with accompanying information as the model and the basis set.

TABLE 5.1: An example of the hash table with the points p2, p3 as the basis set.

Points	Index	Information (Model and Basis set)
p0	(-0.5, 0.5)	M0,E2,3
p1	(0.5, 0.5)	M0,E2,3
p4	(-2.5, 1.0)	M0,E2,3

The generation of the hash table will involve doing this for all possible combinations of the basis set that can be formed and the models involved. Figure 5.3 provides all the hash table entries produced from all possible combinations of the basis set.

In the recognition stage, we select a pair of points as the basis set. We transform them to the canonical reference frame points (-0.5,0) and (0.5,0). With this

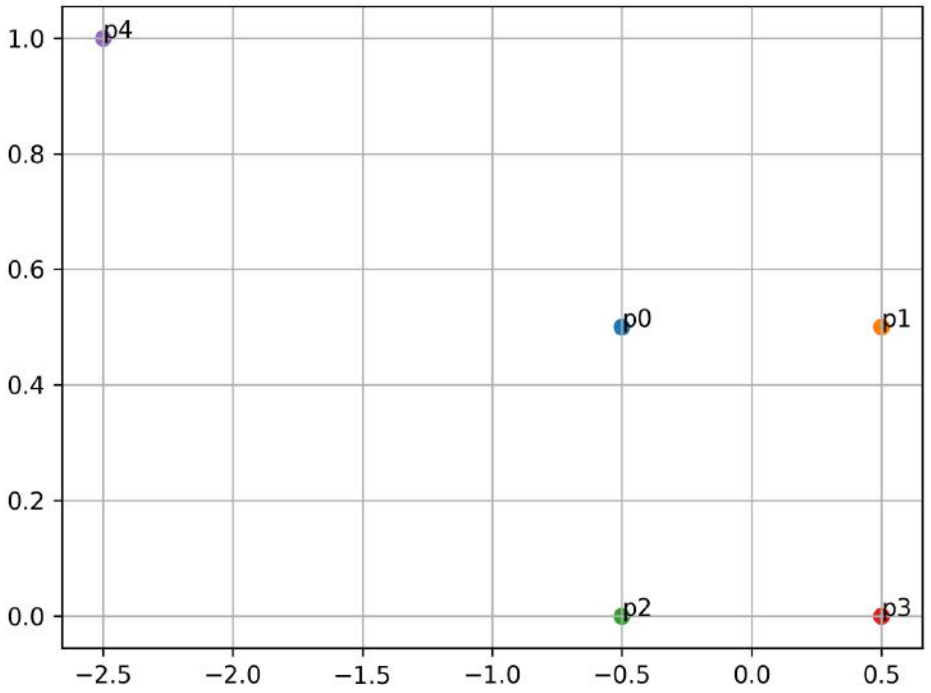


FIGURE 5.2: An example of the basis set $p2, p3$ at (-0.5,0) and (0.5,0) and the rest of the transformed points.

transformation the rest of the points are transformed to new locations. Each of the locations are weighted and quantized to get the index used to search the hash table. Any information in the hash table which exists within a bin is taken and a tally is added to it. Finally, we have a model and a basis set with their vote counts. The vote is sorted from the highest to the lowest providing for different candidates. In an ideal setting, the model and basis set with the highest vote matches the basis set for the recognition stage. This is not always the case, so we need to verify that the detection is accurate. If none of the candidate votes passes the verification stage we try another basis set.

In algorithms 1 and 2, we provide step-by-step implementations of the method.

Geometric hashing can be implemented in both a probabilistic and a non-probabilistic setting. Each setting has its advantages, especially when it comes to a trade-off between accuracy and speed. This work involves real time application so speed is a big concern.

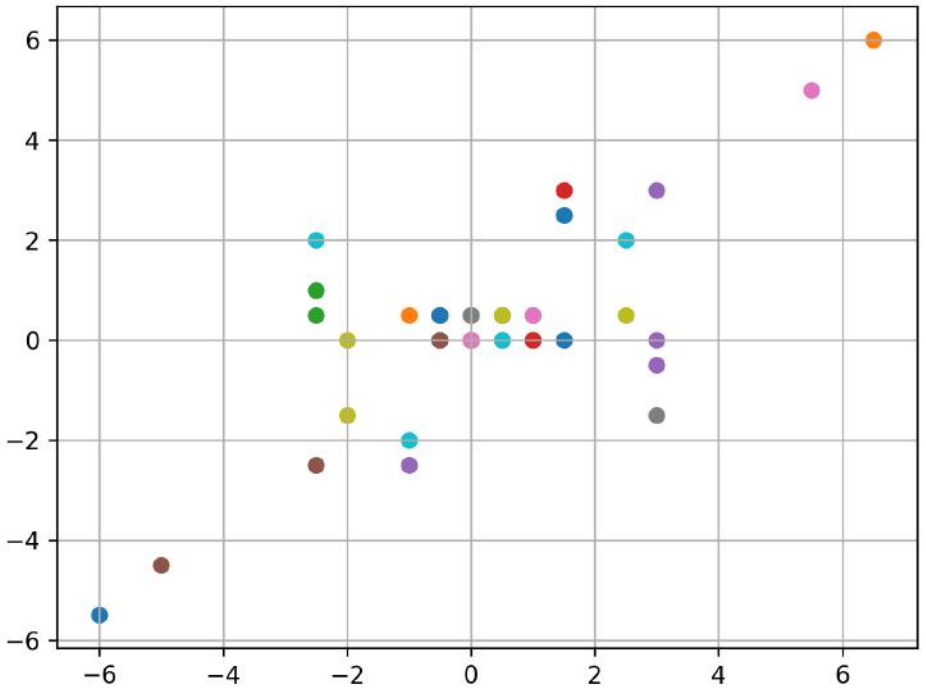


FIGURE 5.3: An example of a hash table for all the possible entries.

Algorithm 1 The training stage

```
Extract features
hash table = [ ]
for all models do
  for all non-collinear basis sets do
    New points = Transform the points excluding the basis set
    for all New points do
      index = quantized and weight a new point
      hash table (index) = [model,basis set]
    end for
  end for
end for
```

Algorithm 2 The recognition stage

```
1: Extract features
2: accumulator = []
3: for all non-collinear basis sets do
4:    New points = Transform the points excluding the basis set
5:    for all New points do
6:       index = quantized and weight a new point
7:       basis set and model = hash table[index]
8:       for all retrieved basis set and model do
9:          accumulator[basis set and model] += 1
10:      end for
11:   end for
12:   sort the votes
13:   for all vote do
14:      if error between basis sets < threshold1 then
15:         RANSAC for verification
16:         if error from RANSAC < threshold2 then
17:            return
18:         end if
19:      end if
20:   end for
21: end for
```

In subsequent sections we provide more details regarding the relationship between the index and the basis set. We discuss the voting approach for both probabilistic and non-probabilistic variants of the method. Furthermore, we discuss ways of verifying the hypothesis match.

5.2 The basis set and index

The basis set is the minimum number of ordered pairs required to transform a certain number of features to a new canonical reference frame. We can assign the basis set a particular location to calculate the transform matrix $\mathcal{T}$, and use the transform matrix to find the location of the rest of the features in the canonical reference frame as $p_i^r = \mathcal{T} p_i$.

We pick points in their reference frame and assign them a new fixed location, using the transform matrix $\mathcal{T}$ calculated. For instance, we can say the basis set for every similarity transformation (p_0, p_1) in 2D is (0.5,0) and (-0.5,0) since the similarity transformation requires two points to solve for it. In the case of an affine transformation we can pick the three locations (p_0, p_1, p_2) as (0,0), (0,1) and (1,0) in the basis set equivalent in the canonical reference frame. For the projective transform we can select the four points (p_0, p_1, p_2, p_3) as (0,0), (0,1), (1,0) and (1,1). We express the new locations of any arbitrary point in the canonical reference as the index.

In essence what we have done is place the points of the basis set as vertices of a shape. Using the location of the basis set as vertices of a shape along with the centre of the shape, we can calculate the new location of a point in the canonical reference frame.

We use the basis set to generate the index after quantization and weighting. The index is the key for the hash table we create to store the model and its accompanying basis set. It is also used for the retrieval of the information from the hash table. Equation 5.1 provides the relationship between the basis set columns p_x, p_y, the center of the basis set p_c, the index before weighting and quantization a, b and a random point p_i:

$$p_i - p_c = ap_x + bp_y. \tag{5.1}$$

We solve for the index by rearranging (5.1) to give the formulation as

$$Ax = y, \tag{5.2}$$

where the basis set matrix is $A = \begin{bmatrix} p_{x1} & p_{y1} \\ p_{x2} & p_{y2} \end{bmatrix}$, $y = \begin{bmatrix} (p_{i1} - p_{c1}) \\ (p_{i2} - p_{c2}) \end{bmatrix}$ and $x = \begin{bmatrix} a \\ b \end{bmatrix}$ is the index before weighting and quantization.

The transformation type determines the shape of the basis set, how to calculate its centre and the way to find its basis set column. The center point is considered as the origin of the canonical reference frame. We provide details for three different transformations below:

- Similarity: the basis set is the endpoints of a line (p_0, p_1) and the centre is the point midway point between them $p_c = (p_0 + p_1)/2$. The basis set column

is the difference between the two basis sets and their normal as can be seen in $p_x = p_1 - p_0$ and $p_y = p_x^T$.

- Affine: the basis set is the vertices of a triangle (p_0, p_1, p_2) and the center is $p_c = (p_0 + p_1 + p_2)/3$. The basis set column is the difference between the two points against one of the points: $p_x = p_1 - p_0$ and $p_y = p_2 - p_0$.
- Projective: the basis set corresponds to the vertices of a quadrilateral (p_0, p_1, p_2, p_3) and the center is $p_c = (p_0 + p_1 + p_2 + p_3)/4$. The basis set column is the difference between two diagonal points of the quadrilateral $p_x = p_3 - p_1$ and $p_y = p_2 - p_0$.

In a particular reference frame, features that represent the same object are likely to be aligned and identify the object. A configuration of all the basis sets possible is obtained during the training phase and is recovered during the recognition phase. At the training phase, the basis set and its accompanying object are stored in a hash table.

5.3 The voting

The scoring can be done using either a probabilistic or a non-probabilistic technique. At every iteration the non-probabilistic method ensures that each point contributes only itself to the voting process, while for the probabilistic method at a point each of the other scene points contributes a value. The closer points tend to contribute the most and the furthest points contribute the least. We provide a proof for the probabilistic method.

5.3.1 Probabilistic voting

The Bayesian formulation of the algorithm involves treating it as a weighted voting technique. By weighting we mean every point contributes to the voting process by adding a different amount at each iteration.

The derivation we provide for the voting is adapted from the work of Rigoutsos et al. [77], where the geometric hashing problem is treated using a maximum likelihood approach.

Let the hypothesis $\mathcal{H}$ be the stored information in our hash table for the model and basis set with the index produced from the basis set. The scene features are given by $S = s_1, s_2, s_3, ..., s_n$, and $S' = s'_1, s'_2, s'_3, ..., s'_{n-k}$ are the scene features excluding the basis set.

Using Bayes' theorem, we want a solution to $Pr(\mathcal{H}|\mathcal{S}')$, which scores the hypothesis given the scene features.

We use the maximum likelihood method to find the sets of the scene features that will make the hypothesis most likely. Therefore, we obtain all the possible hypotheses and the maximum hypothesis that is most probable.

If s_i is a point among the scene features S', then $Pr(s_i|\mathcal{H})$ is the probability of the point p_i given the hypothesis.

A complete solution can be given as

$$Pr(\mathcal{H}|S) \sim Pr(\mathcal{H}) \prod_{s_i \in S'} \frac{Pr(\mathcal{H}|s_i)}{Pr(\mathcal{H})}. \tag{5.3}$$

With the uses of Bayes' theorem we rewrite this as

$$Pr(\mathcal{H}|S) \sim Pr(\mathcal{H}) \prod_{s_i \in S'} \frac{Pr(s_i|\mathcal{H})}{Pr(s_i)}. \tag{5.4}$$

We take the logarithm of both sides of Equation 5.4, since we are interested in only the magnitude:

$$\log(Pr(\mathcal{H}|S)) \sim \log(Pr(\mathcal{H})) + \sum_{s_i \in S'} \log \frac{Pr(p_i|\mathcal{H})}{Pr(p_i)}. \tag{5.5}$$

Since $\log(Pr(\mathcal{H}))$ is the same for a particular point in the scene feature, $\sum \log \frac{Pr(p_i|\mathcal{H})}{Pr(p_i)}$ will be sufficient to calculate the contribution of the point for the voting.

The density function of the index in the hash space is $f(a, b)$, and the density function in the hash space is $g(a, b)$.

We solve the probability of the scene feature given the hypothesis by decomposing into the density function of the index and the density of the index of the point

$$Pr(s_i|\mathcal{H}) = f(a,b) + g(a,b), \tag{5.6}$$

and the probability of a scene feature is the density of the hash space it occupies,

$$Pr(s_i) = g(a,b). \tag{5.7}$$

Therefore, we substitute the various equations:

$$\log \frac{Pr(p_i|\mathcal{H})}{Pr(p_i)} = \log \frac{f(a,b) + g(a,b)}{f(a,b)}. \tag{5.8}$$

The density function of the index in the hash space is the distribution of all the indices in the hash space. The indices tend to have non-uniform distributions, but we assume a uniform Gaussian distribution of the indices with standard deviation $\mathcal{N}(0,\sigma^2)$. We can solve the joint distribution of the index (a,b) generated in Equation 5.1.

The density function is found as

$$g(a,b) = \sum_{j=1}^{j-k} \frac{1}{2\pi\sqrt{C}} \exp - \frac{(a-x_j, b-y_j)C^{-1}(a-x_j, b-y_j)^T}{2},$$

with C the covariance of the system taking into account the perturbations.

Similarity transformation

The density function has been solved for the similarity transformation as

$$f^s(a,b) = \frac{12}{\pi} \frac{1}{(4(a^2+b^2)+3)^2}. \tag{5.9}$$

The density function of a point in the hash space is the likely distribution of the other points around the point in the hash space predicted by the hypothesis $\mathcal{H}$.

The covariance for the similarity is found to be

$$C^s = \frac{(4(x^2+y^2)+3)\sigma^2}{2||p_x - p_y||^2} I.$$

Substituting all the values we obtain the vote contributed at a point as

$$v^s = \log(1 + \frac{4(a^2+b^2)+3)^2||p_x - p_y||^2}{12S(4(x^2+y^2+3))\sigma^2} \exp \frac{-||(a,b)-(x,y)||^2}{4(x^2+y^2+3)).\sigma^2/||p_x - p_y||^2}).$$

Affine transformation

The density function for the affine transform is given as

$$f^a(a,b) = \frac{2\sqrt{2}}{\pi} \frac{1}{(4a^2+4b^2+4ab+8/3)^{\frac{1}{2}}}. \tag{5.10}$$

The covariance of the affine transformation is found to be

$$C^a = \frac{(4(a^2+b^2+ab)+8/3)\sigma^2}{2}. \tag{5.11}$$

The vote contributed by a point for the affine transformation, when we substitute values, is

$$v^a = \log(1+\frac{4(a^2+b^2)+3)^2||p_x - p_y||^2}{12S(4(x^2+y^2+3))\sigma^2} \exp \frac{-||(a,b)-(x,y)||^2}{4(x^2+y^2+xy)+8/3)\sigma^2/||p_x - p_y||^2}). \tag{5.12}$$

5.3.2 Non-probabilistic voting

The non-probabilistic voting technique works under the assumption that each index represents the exact information needed. We pick the model and basis set and assign it a value of one. If the index exists, the accompanying information will be added. The model and basis set with the highest value is considered the right hypothesis.

5.4 Verification

The voting process will provide a set of hypothesis candidates that will need to pass through a verification stage. These candidates are the model and basis set with the highest score. The candidate with the highest vote is expected to be the correct match but this is not always the case due to factors such as noise, distance between the points in the basis set and quantization. Other incorrect correspondences may also have high vote counts. Further into the chapter, we provide the justification of why there is an occurrence of false matches.

The original paper [72] used the least square technique for verification, where we minimise the error distance between the scene points p_i^s and the transformed training points p_i^t that have been registered onto the scene image. Therefore, the loss requires minimising the error $\epsilon = p_i^s - p_i^t$.

Minimising the error at just a single iteration will not provide the best possible value. Hence iterative techniques such as RANSAC are used. Some points are outliers and RANSAC will handle the outlier issue.

The verification serves not only to confirm the accuracy of the search but also to refine the transformation.

There are two thresholds for the verification to ensure the right match has been found. The first threshold is obtained as root mean square error between the recognition and the training basis sets. The threshold is set as a first step verification to ensure the basis sets are equivalent. If the error is high then the basis sets are not equivalent and will avoid the iterative techniques which are computationally expensive.

The second threshold is set after the RANSAC method is used and the inlier points are obtained. The threshold is determined as the root mean square error between the training and recognition points. We ensure the inlier points are up to a certain number to ensure a good transformation. With the inlier points and using nearest neighbour for correspondence we know the training points that are associated with the recognition points.

The thresholds are found experimentally based on the image dimension and the noise in the measurement.

5.5 Noise analysis

Noise tends to affect the performance of the geometric hashing method. Noise is generated during the feature extraction and/or quantization of the values in the hash table. The noise due to the quantization we call quantization error, and that due to feature extraction we call measurement error. It is complicated to separate the effect of the quantization and the measurement errors.

We need to analyse the effect of noise on the basis set matrix A, the offset point y, and the index x. We add perturbations to Equation 5.2 and the equations become

$$\begin{aligned} (A + \delta A)x &= y + \delta y \\ A(I + A^{-1}\delta A)x &= y + \delta y. \end{aligned} \tag{5.13}$$

Expanding the equation using a Taylor series gives

$$\begin{aligned} x &= A^{-1}(I + A^{-1}\delta A)^{-1}(y + \delta y) \\ x &\approx A^{-1}(I - A^{-1}\delta A + h.o.t.)(y + \delta y) \\ x &\approx A^{-1}y + A^{-1}\delta y - A^{-1}(\delta A)A^{-1}y + h.o.t. \end{aligned} \tag{5.14}$$

Ignoring the higher order terms ($h.o.t.$) and having $\hat{x} = A^{-1}y$ as the true index, we get the perturbation of the index x as

$$\delta x = A^{-1}(\delta y) - A^{-1}(\delta A)\hat{x}. \tag{5.15}$$

Equation (5.15) relates the perturbation δx in the index x, given noise in the initial point and the point y, and the basis set matrix A.

The distance between the points in the basis set presents an important property from the equation: the smaller the error separation between the center of the points and the points δy, the smaller the error of the index δx.

The larger the matrix containing the basis set A, the smaller the error between the index x.

5.6 Complexity analysis

The complexity of the algorithm can be studied in terms of the training and recognition stages. In the training stage, it involves generating the basis set and storing in a hash table which is $O(Nn^{k+1})$ with N models, k transformation type and n training features.

The recognition stage has complexity $O(Mm^{k+1})$ with M models, k transformation type and m scene features for generating the basis set. The generation of the basis set is not the worst case for the recognition, rather the worst case is at the voting stage where the complexity is $O(Mm^{k+1}n^{k+1})$. The complexity relates to the generation of the basis set and also the search for a particular point in the stored information in the hash table.

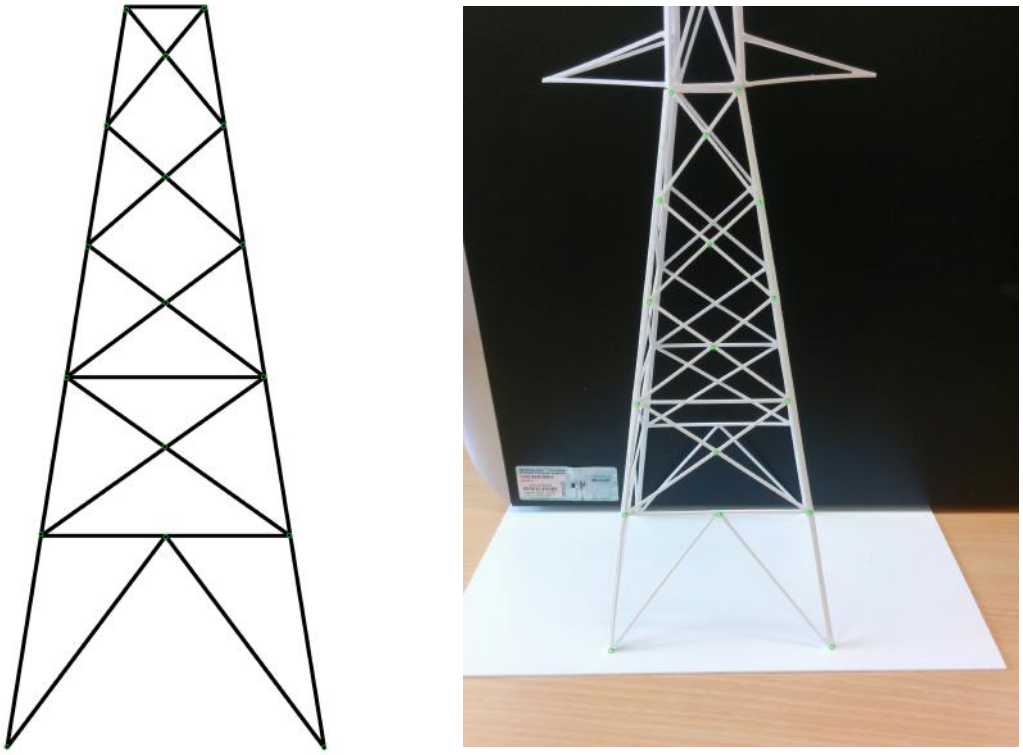
The complexity analysis of the algorithm informs us that the number of features greatly determines the time and space needed for the algorithm to function. The algorithm is mostly implemented such that the training stage happens offline and the recognition happens online. It is fine for the training stage to take time but if the worst case takes a lot more time than the recognition stage then we have to make some assumptions such as trying a limited number of basis sets.

The number of training features should be close to the number of features in the recognition stage. If not, then the number of possible basis sets tends to be very large. A transformation will have a number of possibilities to find the right corresponding basis set. For instance, an affine transform with 70 features will have 54,740 and 67,535 basis sets for 80 features, neglecting collinear points. As we are likely to select a random basis set for the search, this poses a problem if speed is considered a factor. It is best to have the number of features in the recognition stage less than or equal to the number of features in the training stage.

Also, in a system with a large number of features, the basis set can give an incomplete representation of the transformation and the estimation falls into a local minimum. Therefore we need to refine in the verification stage.

5.7 Implementation

Geometric hashing is a voting technique used to recognize an object in an image. We experimented with the method to establish the effectiveness of the method under different parameters such as bin size, quantization value, transformation type, voting approach and addition of Gaussian noise.



(A) Training Image.

(B) Recognition Image.

FIGURE 5.4: Example of images used in the training and recognition stage with their features.

The pylon has the same 'face' structure for all the four sides. Therefore, the CAD drawing (Figure 5.4a) as the image in the training stage represents the face of all the sides. In the recognition stage we used the image of the pylon model (Figure 5.4b) that we built for this research. With this setup we conduct the experiments in this section. We experiment with the voting type, noise, transformation type, bin size and quantization to see the effect on the accuracy rate of the algorithm.

Figures 5.5 and 5.6 are examples of successful application of the geometric hashing method using an affine transformation. The big blue circles represent the features

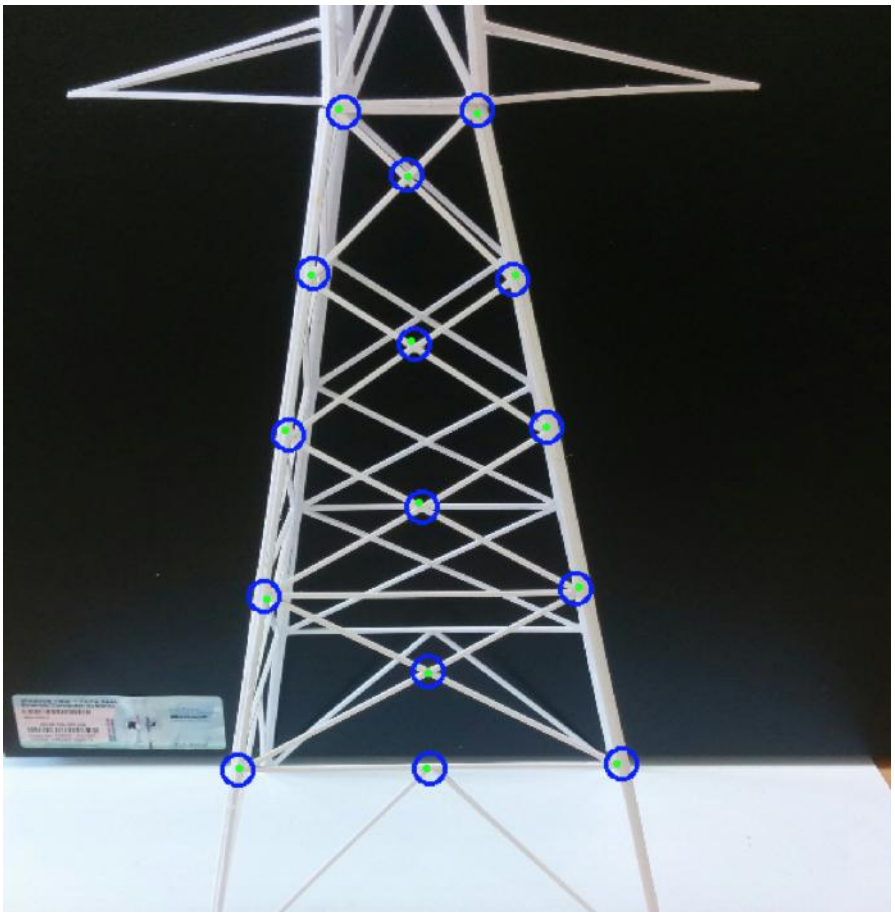


FIGURE 5.5: Matching with geometric hashing where the training and recognition images have the same number of points.

in the recognition stage and the small green circles are the features transformed from the training stage. Figure 5.5 is for the case where the training and recognition have the same number of features. Figure 5.6 shows a case where the number of features in the recognition stage exceeds the number of features in the training stage.

We further performed a thorough analysis of the algorithm. In the rest of the section we discuss the various factors and how they affect the performance of the algorithm. The recognition accuracy is the ability of the algorithm to have exactly the same model and basis set in the training stage as in the recognition stage. Therefore, we ensure that the features correctly correspond in both stages.

To store the model or vote for the model an element needs to be weighted and quantized. Therefore we are interested in knowing the effect of the hash bin size and the quantization on the accuracy rate.

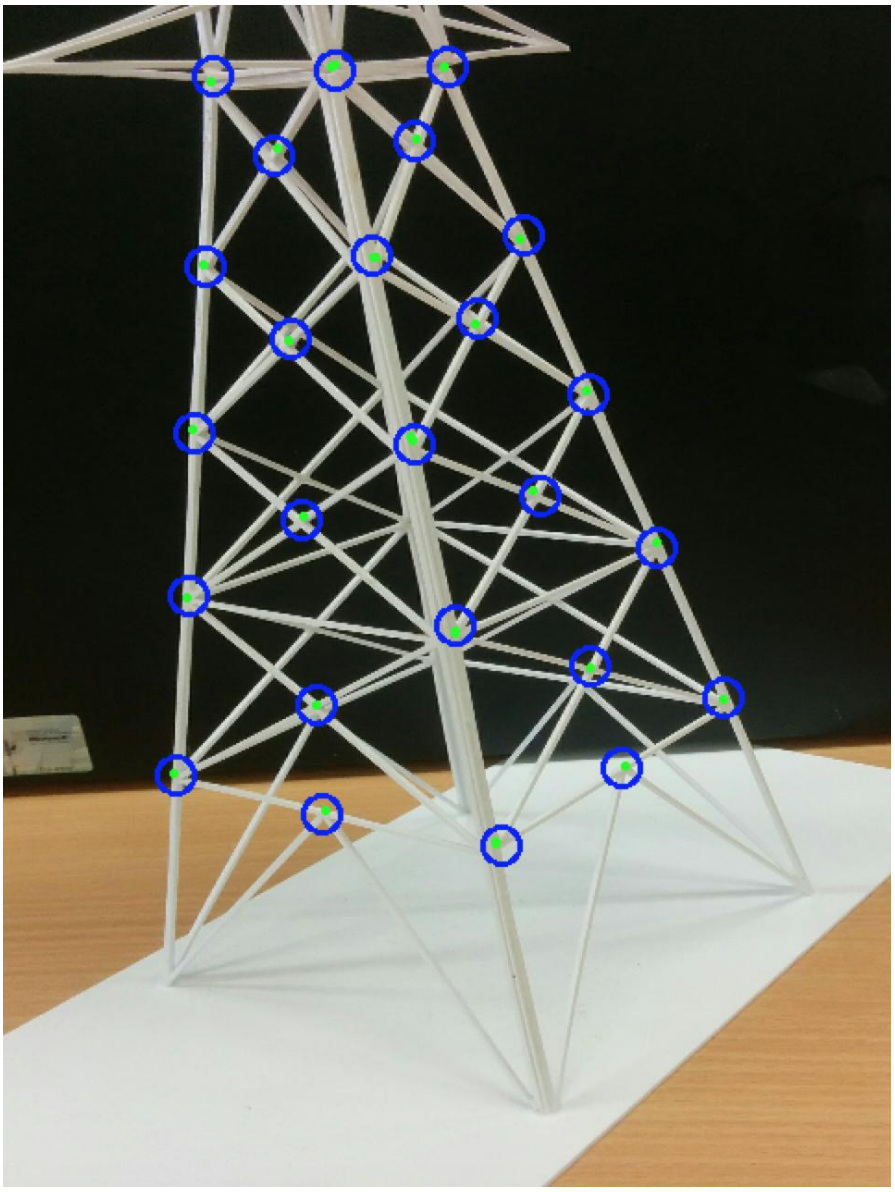


FIGURE 5.6: Matching with geometric hashing where the training image has less points than the recognition image.

The hash bin size allows us to partition a table into various regions. It helps with efficient retrieval of a query from the region. The region mostly contains relevant information, and in our case it is the model and basis set.

The hash bin size is determined by the weight. The weight is by default one, which means any point is rounded to the closest integer. Therefore, a bin is weighted as

$$id_i = \frac{p_i}{w}$$

with w as the weight.

The weight can range from 0 to ∞. A weight greater than one means the bin size is being decreased and the spread between points becomes smaller. For weights less than one the bin size is being increased and the spread becomes larger, which is like magnifying the bin to see the details.

The transformation of points to a new reference frame means we obtain floating point values, and we want points that have the same value or small differences between them to fall into the same bin.

The float value obtained from the transformation needs to be quantized. Quantization is a means of converting large sets of values or continuous sets to discrete sets. The discrete sets can be sets of integers, multiples of a number (2,3,5) or multiples of decimal points (0.25, 0.50, 0.001).

The application of quantization to the data introduces some errors. Therefore, we need to find a value of quantization such that the error has a minimal effect on our search technique.

Therefore, we quantize a value by multiplying the point p_i by the quantization value, obtain the ceiling of the multiplication, and then divide by the quantization value:

$$q_i = \frac{\lceil p_i \times qt \rceil}{qt}.$$

For instance, a quantization value of 4 means the values will be a discrete set ending with a multiple of 0.25. A multiple of 0.5 will be produced if the quantization value is 2.

The quantization happens after weighting a value and some points will fall into a neighbouring bin. The reason for taking a small weighting value is for the bin size spread to be large. Additionally a small quantization value will reduce the effect of the quantization error.

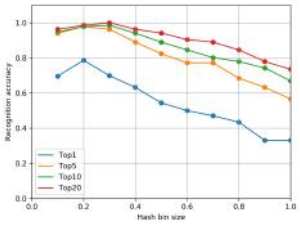
5.7.1 Effect of hash bin size and quantization

We perform experiments to check the effectiveness of the algorithm on various hash bin sizes and quantization settings for a similarity transformation type with Gaussian noise having a standard deviation of zero using the non-probabilistic voting method. We manually picked the features and ensured that features are

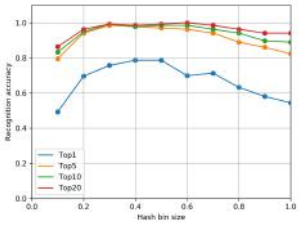
numbered the same in both the training and the recognition stages. The CAD diagram (Figure 5.4a) is used as the model for the training stage and Figure 5.4b is the image used to find the exact match in the recognition stage.

Figure 5.7 provides the result of the test of geometric hashing with several quantization values. The recognition accuracy is the rate when the basis set of the training stage finds its equivalent in the basis set of the recognition stage. We show the recognition accuracy when it falls into the top 20, top 10, top 5 and top 1 places of the hypothesis candidates obtained from the voting technique.

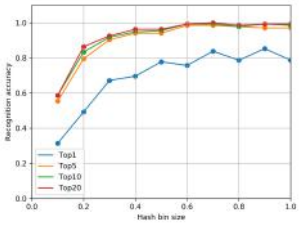
It can be seen that as we increase the quantization value, the accuracy rate decreases as voting elements do not fall into the same bin. The best quantization value was seen in Figure 5.7b where the accuracy was mostly 1 for various hash bin sizes. As the quantization value increases it makes the need for weighting the values unnecessary as the bin size tends closer to one as the best recognition rate.



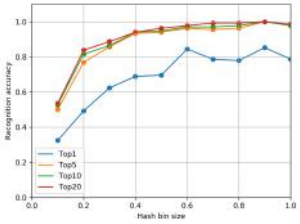
(A) Quantization of 1.



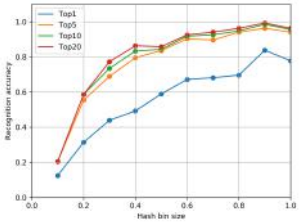
(B) Quantization of 2.



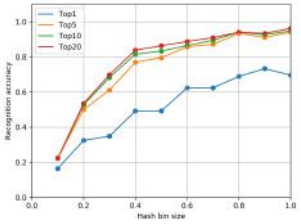
(C) Quantization of 4.



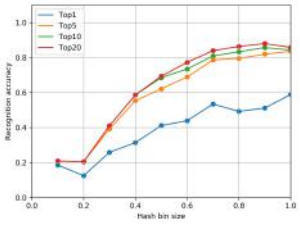
(D) Quantization of 5.



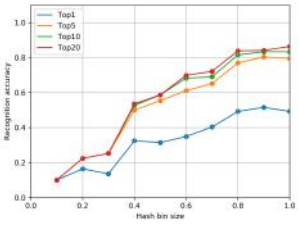
(E) Quantization of 8.



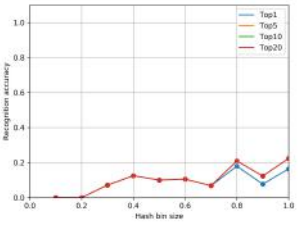
(F) Quantization of 10.



(G) Quantization of 16.



(H) Quantization of 20.



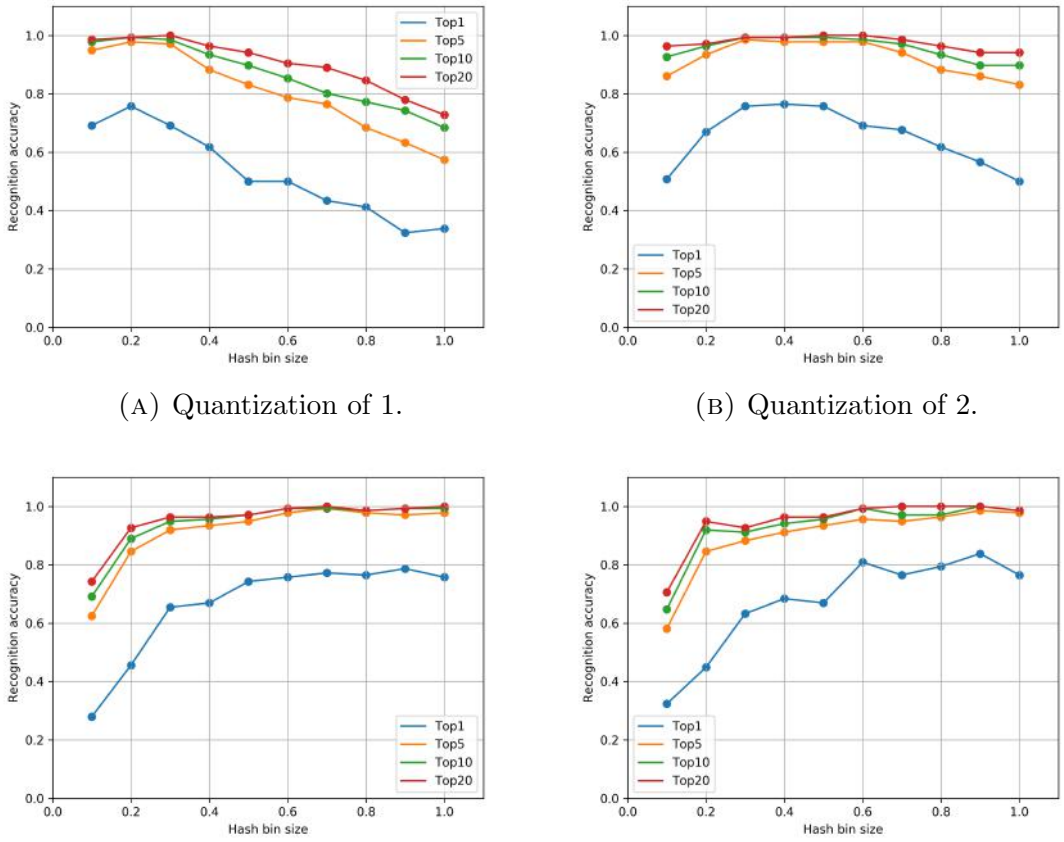
(I) Quantization of 100.

FIGURE 5.7: Results for different quantization values for similarity transformation and non-probabilistic voting technique.

5.7.2 Effect of voting type

The training image and the recognition image in Figure 5.4 with their features on the pylon is used to experiment for the different voting techniques. We test to check if there is a difference between the accuracy rate for the probabilistic and non-probabilistic voting techniques.

We next implemented the probabilistic voting technique (Figure 5.8) and it shows some improvement over the non-probabilistic technique. The improvement comes at the expense of speed, especially as the non-probabilistic implementation is a lot faster. As the quantization value becomes larger, the difference in the voting approach becomes more pronounced as the quantization and the hash bin size values have more effect on the non-probabilistic method.



(A) Quantization of 1.

(B) Quantization of 2.

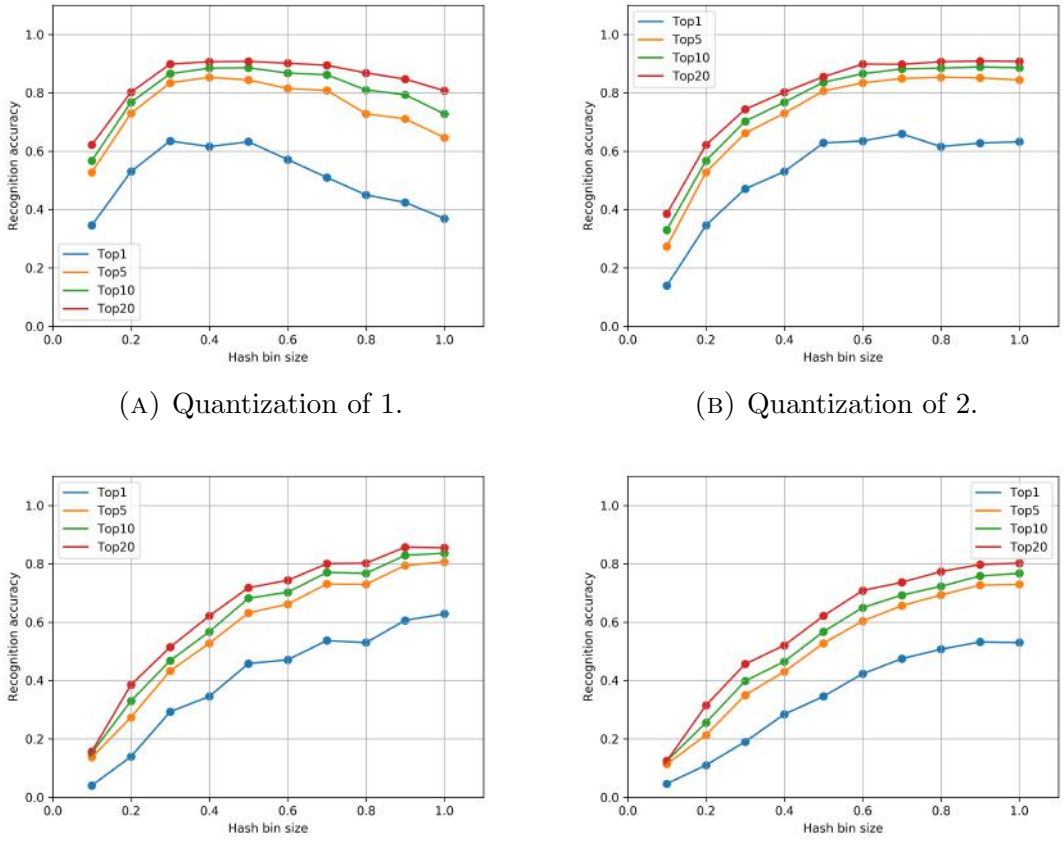
(C) Quantization of 4.

(D) Quantization of 5.

FIGURE 5.8: Results of different quantization values for similarity transformation and probabilistic voting technique.

5.7.3 Effect of transformation type

Objects undergo different transformation changes that alter their shape. We conducted experiments to check the transformation change that happened between the training and recognition in Figure 5.4 and the possibility of establishing a match. We present the result of using different transformations to test the effectiveness of the algorithm. Figure 5.9a provides results for an affine transformation and Figure 5.10 is for a projective transformation. We observe that the recognition accuracy decreases as we move from similarity to affine to projective transformations, this maybe because the fundamental transformation of the images is closest to the similarity transformation.



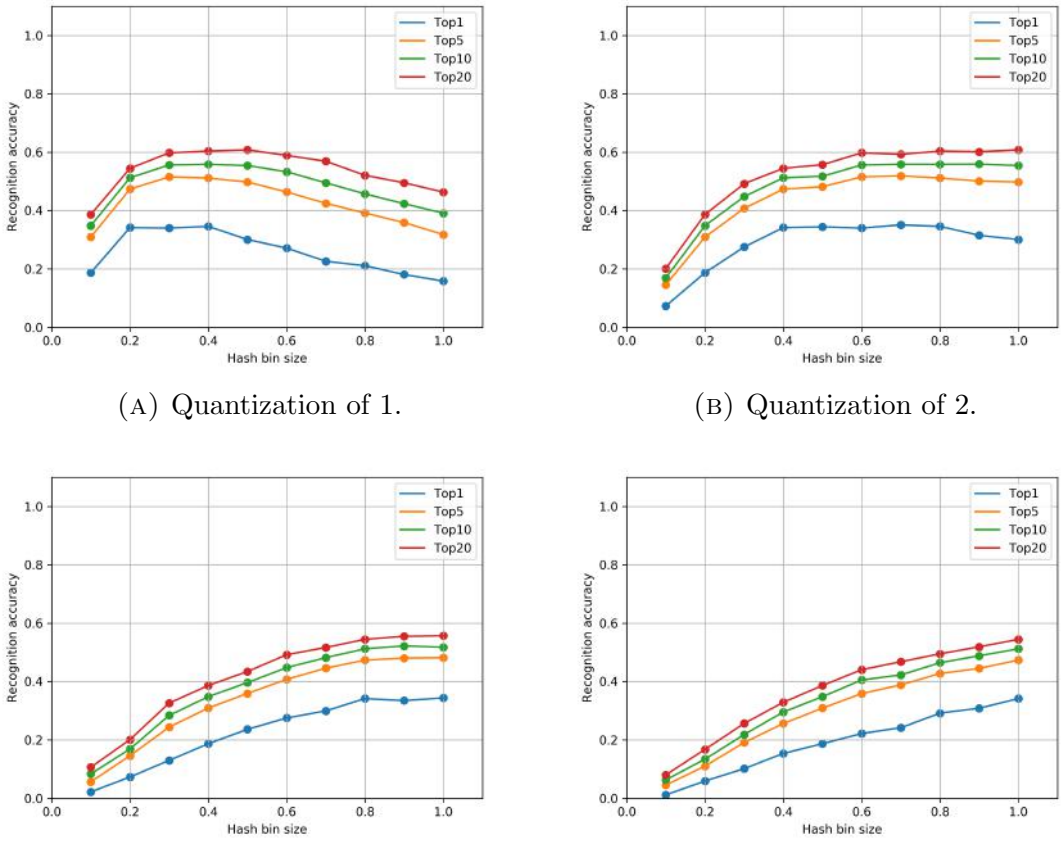
(A) Quantization of 1.

(B) Quantization of 2.

(C) Quantization of 4.

(D) Quantization of 5.

FIGURE 5.9: Results of different quantization values for affine transformation and non-probabilistic voting technique.



(A) Quantization of 1.

(B) Quantization of 2.

(C) Quantization of 4.

(D) Quantization of 5.

FIGURE 5.10: Results of different quantization values for projective transformation and non-probabilistic voting technique.

5.7.4 Effect of noise

To check how robustly the algorithm responds to noise, we conducted an experiment using the setup in Figure 5.4. We test the idea on the affine transformation, with a hash bin size of 0.5 and a quantization of 2.

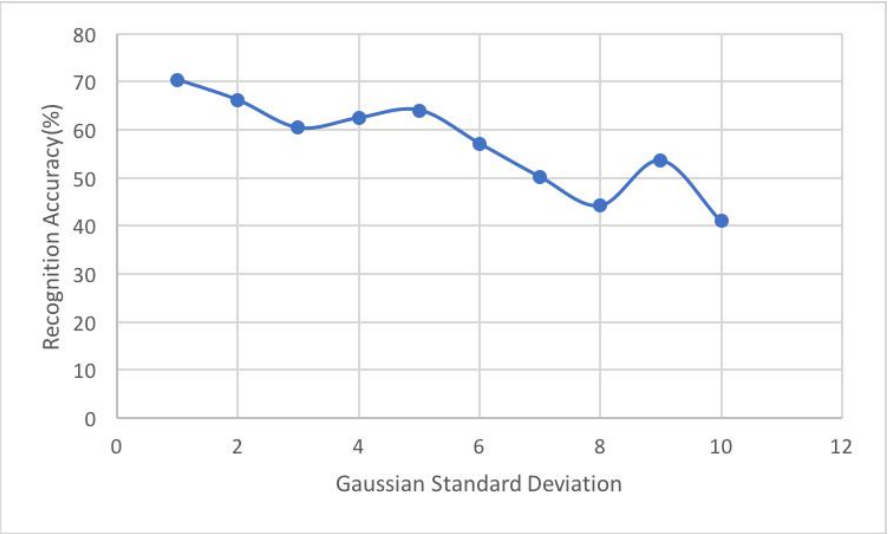


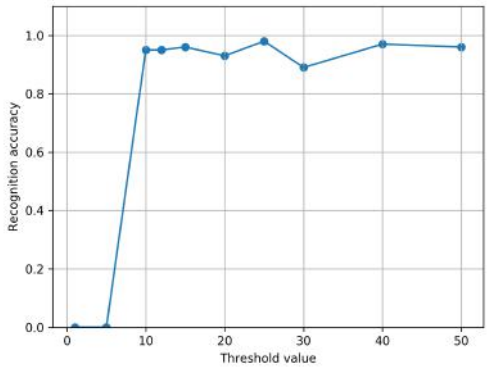
FIGURE 5.11: Effect of noise.

We add Gaussian noise with different standard deviations to the features. Figure 5.11 shows the effect of the Gaussian noise on the voting; the accuracy rate decreases as the noise increases.

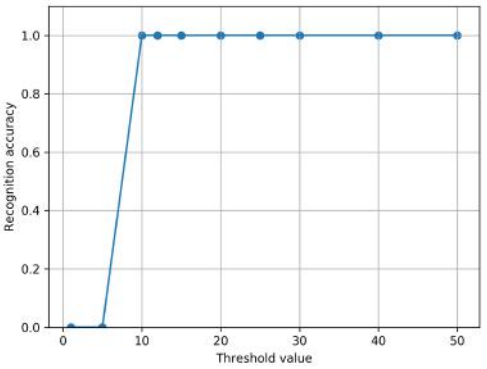
5.7.5 Effect of threshold

In trying to apply the algorithm we cannot attempt all the basis sets from the recognition stage. We randomly select a certain number of basis sets. We conduct an experiment to check the accuracy with a threshold for the root mean square error from the verification stage using RANSAC. Furthermore, the experiment tries a specific number of basis sets in the recognition stage until a match is found. The setup uses the training and the recognition images in Figure 5.4. The transformation used is affine, the hash bin size is 0.5 and the quantization value is 2.

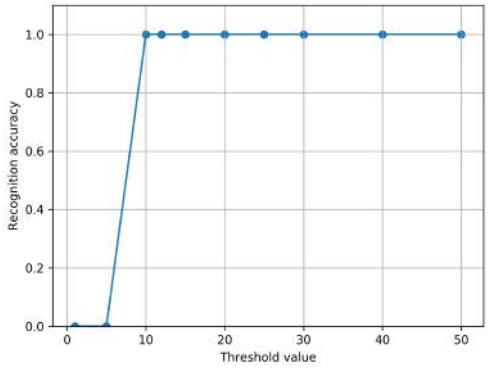
In Figure 5.12, we see that more trials means better results for the detection. A very low threshold value of less than 10 pixels shows there is no detection. We have to be careful when choosing large threshold values for the detection because even though increasing the threshold has a higher accuracy rate of detection, wrong candidates can be part of the group and lead to an inaccurate match. The results for a threshold set at 20 pixels and 50 pixels look identical as the accuracy rate is the same. It means any threshold set beyond 20 pixels offers no advantage when it comes to the accuracy rate.



(A) Threshold of 10.



(B) Threshold of 20.



(C) Threshold of 50.

FIGURE 5.12: Set of experiments with different threshold values.

5.8 Six degree-of-freedom object pose estimation with geometric hashing

We use geometric hashing to estimate the pose of an object from a single image. Geometric hashing is used to find a match between a stored template and a scene object, and the correspondence between the points of the template and the object are established. In the case of six degree-of-freedom object pose estimation, correspondence is established between the world points and the image points.

We use affine transformation in this work as the sides of the pylon are planar. As long as a camera is far enough from an object, affine transformation is an approximation of projective transformation. For this experiment, we use affine transformation for the generation of the basis sets. The basis sets are in a group of three points since the affine transformation requires three non-collinear points.

We obtain the world models by reconstructing the 3D model of an electricity pylon using a laser scanner (see Section 6.6.1). From the 3D model, we obtain the 3D coordinates of the vertices.

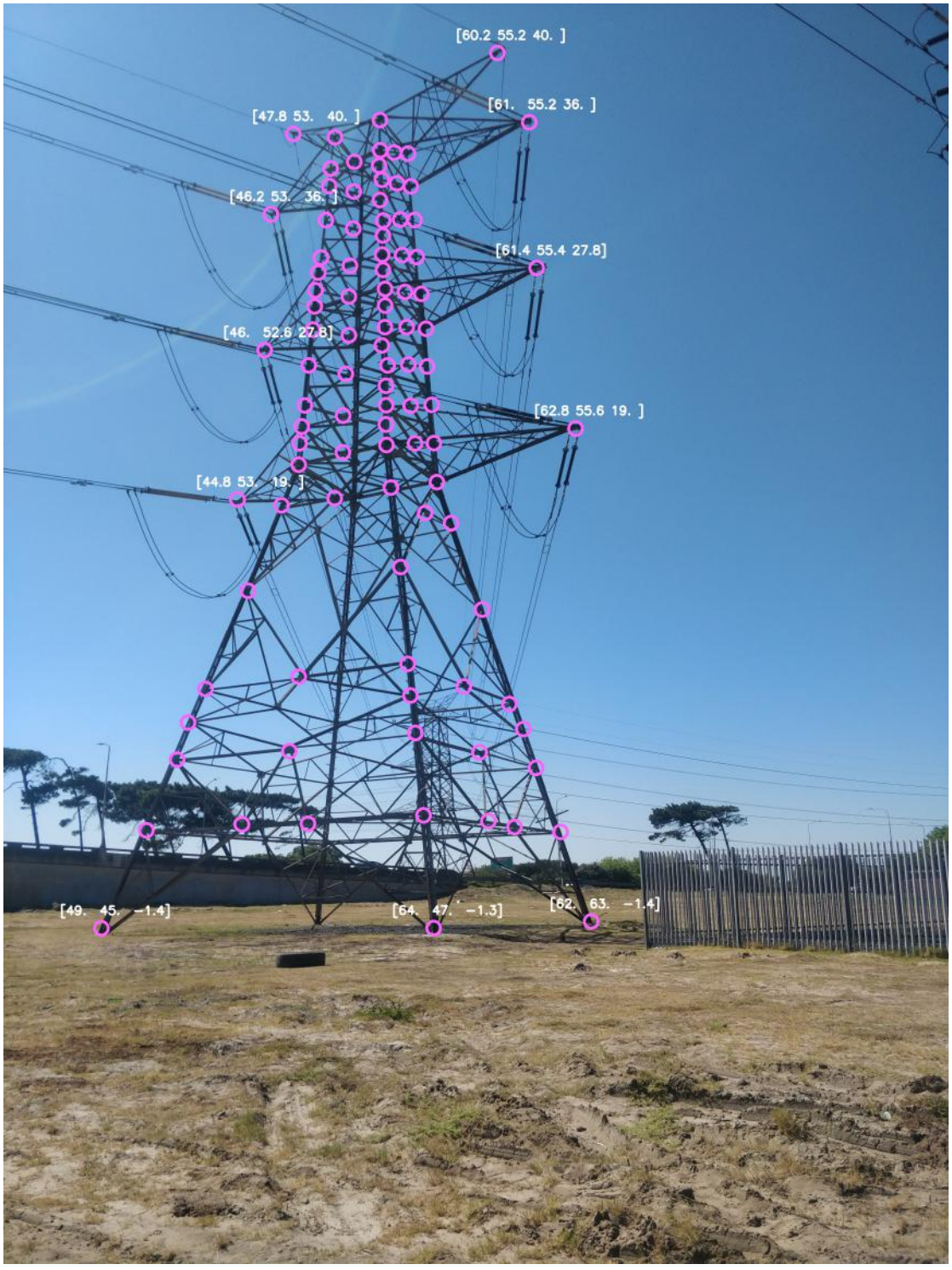


FIGURE 5.13: A template of an image with some of the 3D coordinates stored during the training phase.

Figure 5.13 shows the image of a pylon with the 3D coordinates of some of the vertices. During the training a template of the pylon with 99 vertices labelled and the image points of the corresponding world coordinates are stored. The 99 vertices show the two faces of the pylon. We store a single template for the experiment so we have one model with 99 features.

FIGURE 5.14: Some of the sample images in the electricity pylon data.

To test the performance of the algorithm, we generate a sequence. We extract images from a captured video to produce a sequence of 300 images. Figure 5.14 provides some sample images from the sequence. At the recognition stage, we

feed one of the images in the sequence and use the convolutional neural network to extract the features on the pylon. In addition, we mark twelve points on each pylon image and use the marked points to calculate the ground truth pose.

At the training stage, all the basis sets for a world model and its features are extracted and stored in a hash table. The training process is done offline, while the recognition process is done online. The recognition process involves the retrieval of the object from the hash table and finding a match. The features of the recognition stage are the vertices detected using the convolutional neural network presented in Section 4.4.2. Geometric hashing helps us retrieve the 3D coordinates in the training stage using the image scene 2D coordinates from the recognition stage. After successful matching, we have 3D to 2D correspondence between the object in the hash table and the object in the image scene. As there is correspondence between the training and recognition, we use the perspective-n-point (PNP) algorithm to solve for the pose estimation.

We use a hash bin size of 0.5, a quantization value of 4 and a non-probabilistic voting technique. Instead of trying all the possible basis sets, we select 100 random basis sets for the recognition stage. At each attempt for a basis set, we select the three highest scores to pass for a verification.

There are two thresholds: the first is set when there is a match between basis sets, and the second is set after the verification. The first threshold is set at a root mean squared error of 100 pixels, and the second threshold after verification is at a root mean squared error of 10 pixels.

We define an evaluation metric to test the accuracy of the algorithm. The evaluation metric is able to help us measure the accuracy of the algorithm. We compute the average mean distance between the transformed 3D model points with the ground truth and estimation values. With the rotation, R, and translation, t, of the ground truth pose estimate and the rotation, $\hat{R}$, and translation, $\hat{t}$, of the estimated pose, we can determine the average distance between the model points and the transformed points. In mathematical terms, the average distance is given as

$$= \frac{1}{N} \sum_{X \in \mathcal{N}} ||(RX + t) - (\hat{R}X + \hat{t})||, \tag{5.16}$$

where N is the number of points, and $\mathcal{N}$ denotes the set of 3D model points. The 3D points are transformed by the pose estimate and pose ground truth, then the average Euclidean distance between them is determined.

To determine the accuracy, we get the difference between the estimated pose and the ground truth pose. Then we set a threshold for the difference between the poses, and any value that meets threshold is an accurate recognition. Due to the differing sizes of objects, we cannot set a fixed threshold. Instead, we set a threshold that is 10 percent of the diameter of the object or of the longest side of the object. For the pylon, the longest side is 40m.

Method	Accuracy(%)
Geometric hashing + RANSAC	65.3
Geometric hashing + ICP	86

TABLE 5.2: Recognition accuracy rate.

Table 5.2 presents the accuracy rate for a test of the algorithm. The table shows the number of times within the sequence that the geometric hashing method successfully determines the average Euclidean distance to be within the threshold limit. The result shows the use of geometric hashing with two different verification methods namely RANSAC and ICP. The ICP has a better performance than the RANSAC for this experiment.

Figure 5.15 provides examples of a successful match of the image scenes with a stored model and template in the hash table. We show images of the results using the RANSAC and ICP verification methods. With the ICP, the algorithm tends to detect a greater number of features.

The geometric hashing technique is useful in obtaining a single-image pose estimate of a camera relative to an object, so the technique can serve as the initialisation step of a pose tracking system.

FIGURE 5.15: Images of matching scene images with the training template.

5.9 Summary

We discussed the geometric hashing algorithm, a method used in the matching of images. The method selects a certain number of features to generate the basis set and votes with the rest of the features. There is a verification stage at the end of the algorithm to ensure an accurate match has been found

We performed various experiments to compare the effects of different factors such as quantization, hash bin size, transformation type and the voting approach. The results show that the quantization and hash bin size have an effect on the accuracy rate of the algorithm. When the quantization value is too high, such as 10 or more, the accuracy rate greatly reduces. The transformation that is closely linking the model and the image scene has the best accuracy rate, and for the experiment we did in this chapter that is the similarity transformation. When the feature point is accurately determined the method works quite well, but noise greatly reduces the performance of the technique.

We provide the details of how to achieve the pose estimation of an object with geometric hashing. We show how we can establish correspondence between an object in the image scene to a stored template in the hash table.

Chapter 6

Camera Pose Estimation Tracking relative to an object

We have discussed various ideas and algorithms in the other chapters of the work. The focus of this chapter is on using the ideas from the earlier chapters to achieve our main goal of the research, which is obtaining the pose estimate of the camera relative to the pylon.

The theory of the Kalman filter has been presented in Chapter 3. The state transition and measurement model, filter tuning and other details relevant to this work have been thoroughly discussed accordingly. Implementation of an extended Kalman filter for tracking the camera movement in our experiment is hereby discussed in this chapter.

Chapter 4 discusses the use of a convolutional neural network for the vertex detection. We provided experimental details and showed the output of the convolutional neural network when tested on two datasets. Vertex detection serves to locate image keypoints on the pylon, which aids with the matching process. The first two experiments in this chapter use the output from the convolutional neural network as our feature while the third experiment utilises the heatmap of the channels.

Geometric hashing, a model-based technique used in finding the correspondence between a stored model and a scene image, was discussed in Chapter 5. We performed experiments to show the effectiveness of the method under certain parameters such as the transformation type, performance under noise, the effect of

quantization and hash bin size. The results obtained from the geometric hashing method provided us with input parameters for the experiments in this chapter.

We discuss the experiments carried out during the course of the work. We show how ideas that have been previously are harnessed to solve the six degree-of-freedom pose estimation and tracking problem. We set out to obtain the six degree-of-freedom pose estimate of the camera relative to the pylon in each frame of a sequence.

Four experiments are covered in this chapter. In the first experiment (Section 6.3), we test tracking with the use of an extended Kalman filter and vertex detection for the measurement.

The second experiment(Section 6.4) tests the use of geometric hashing in initialising the pose estimation, then tracking the pose with the extended Kalman filter while using the vertex detected with the neural network as the measurement.

For the third experiment (Section 6.5) we initialise the algorithm with geometric hashing while tracking the movement of a camera by optimising the pose estimation using gradient descent.

Finally, the fourth experiment (Section 6.6) makes use of real world outdoor data for tracking the movement of a camera. It uses the method introduced in the first and second experiment.

6.1 Data

The power companies have control of the pylon data needed for analysis in this work. Retrieving such data from these companies is not trivial. To circumvent this data challenge, however we generated our data by building a scale model which we captured using a OnePlus 5t phone camera in our laboratory. In order to create a world model of the pylon we manually annotated the points with 3D coordinates.

Additionally in Section 6.6, we provide a set of data for real world data. The real world data enable us to validate our algorithms and the data was captured using the DJI mini drone camera.

Figure 6.1 shows some of the 3D coordinate points in millimetres. The point $(0, 0, 0)$ is considered as the origin reference point, to the right of the origin as

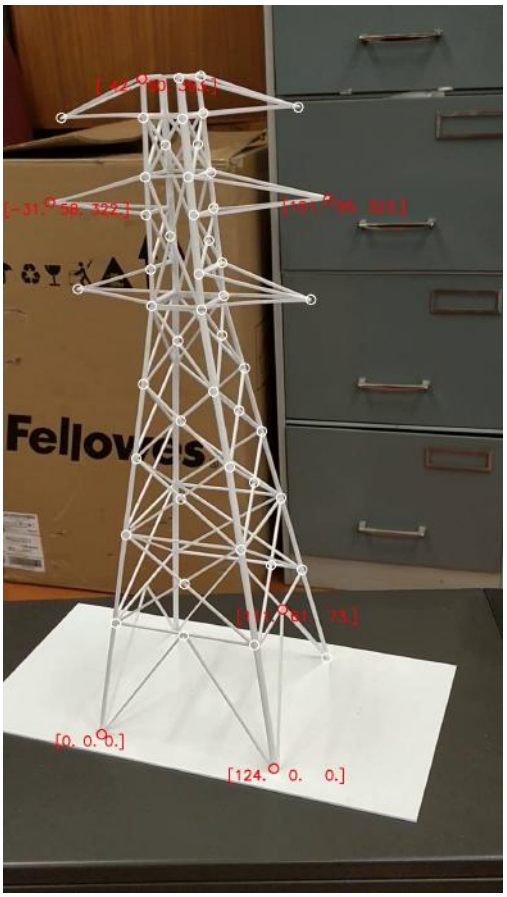


FIGURE 6.1: An image annotated with some of the 3D coordinates.

the positive x-axis, into the page as the positive y-axis and facing upwards as the positive z-axis.

Each hand-annotated vertex was provided with its corresponding 3D point as shown in Figure 6.1.

The feature representation of the pylon is the vertex and it is obtained using the convolutional neural network technique described in Chapter 4. Using a camera we captured images of the pylon from different positions, angles and background to serve as our data.

We generated different video sequences to mimic real world scenarios. Each of the images in the sequences have 100 images. The video sequences generated are used for the single image camera pose estimation and tracking task in this chapter. For the inspection of components on power infrastructure, a robot will need to find the most optimal position to carry out an inspection.

In sequence I the video is obtained by going around the pylon using the camera. Figure 6.2 shows some of the images in the sequence. We have a full view of the pylon at all times from different positions and orientations of the camera. This sequence encompasses a 360° view of the pylon.

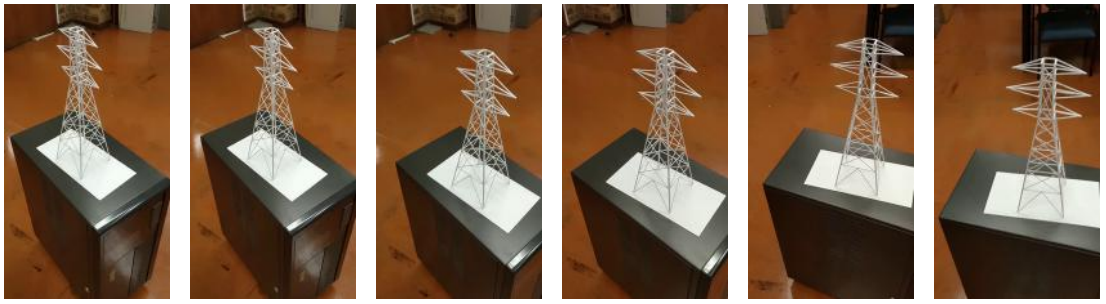


FIGURE 6.2: Images in sequence I.

In sequence II the video is generated by approaching the pylon from a distance. We show some examples of the images in Figure 6.3. The pylon is seen from the same view but the scale increases as the camera approaches. This mimics an aerial robot approaching from a distance or moving away from a pylon.

FIGURE 6.3: Images in sequence II.

Sequence III is obtained by focusing the camera on the view of the pylon while moving from right to left. This process is repeated in the reverse direction moving left to right. This sequence gives the same view of the pylon from different angles. Figure 6.4 provides some of the frames seen in the sequence.

FIGURE 6.4: Images in sequence III.

To test and validate the algorithms we needed to have ground truth of the pose of the camera relative to the pylon. We selected the four corner points on the base of the platform on which the pylon is mounted shown in Figure 6.5. Using the four corner points the pose is determined.

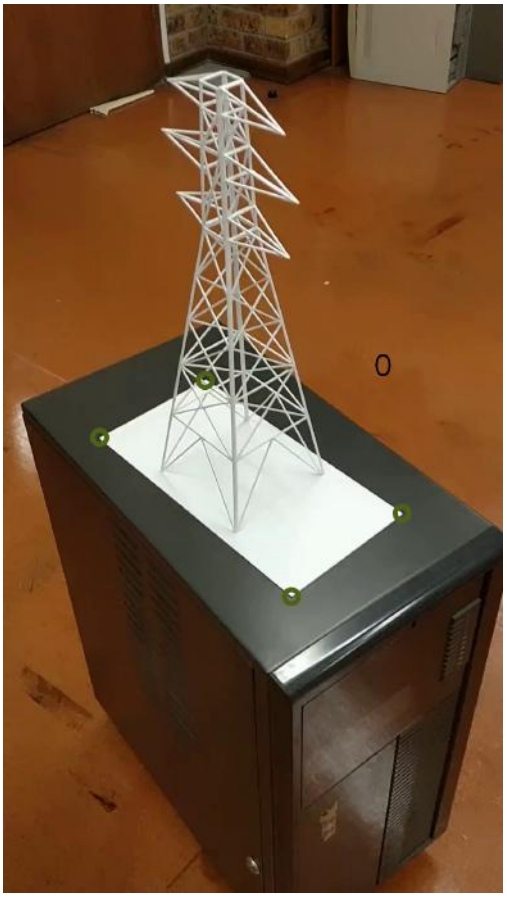


FIGURE 6.5: An image of a pylon mounted on a platform. The four corners of the platform marked as shown in the figure are used to derive the ground truth pose needed to test the algorithms .

The ground truth pose estimate will be used to test the effectiveness of the tracking algorithm.

6.2 Evaluation metrics

We introduce some concepts that will be used to evaluate performance of the pose estimation and tracking algorithms. Given the ground truth rotation R, the ground truth translation t, the estimated rotation $\hat{R}$ and the estimated translation $\hat{t}$, the Euclidean distance of the 3D coordinates after transformation with the ground truth pose and the estimated pose will be determined.

The average point distance (APD) is a pose error metric used to determine the Euclidean distance between the transformed 3D estimated points and transformed ground truth points. The metric is computed by using the pose to transform the 3D points for both the ground truth and the estimate. The difference between the 3D points is then found and finally the mean is determined. In mathematical terms this is shown as

$$\text{APD} = \frac{1}{N} \sum_{X \in \mathcal{N}} ||(RX + t) - (\hat{R}X + \hat{t})||, \tag{6.1}$$

where N is the number of points and $\mathcal{N}$ denotes the set of 3D model points. The 3D points are transformed by the pose estimate and pose ground truth, then the average Euclidean distance between them is determined.

The average closest point distance (ACPD) is a pose error metric used to handle the ambiguity caused by symmetrical objects. The ambiguity arises because the sides of the object look identical. For ACPD, the average distance is computed only for the closest points and is given as

$$\text{ACPD} = \frac{1}{N} \sum_{X_1 \in \mathcal{N}} \underset{X_2 \in \mathcal{N}}{min} ||(RX_1 + t) - (\hat{R}X_2 + \hat{t})||. \tag{6.2}$$

Therefore, the 3D points are transformed using the pose ground truth and the pose estimate. Thereafter the nearest neighbours are established between the transformed points before obtaining the average Euclidean distance. The APD and the ACPD measures are both in 3D coordinate space, as introduced by Hinterstoisser et al. [48].

The reprojection error involves transforming the 3D model onto the image space and determining the average distance between the ground truth transformed points and the estimated transformed points. As a pose error, the reprojection errors

determine if the 3D model fits onto the object in the image scene and is given as

$$\text{reprojection error} = \frac{1}{N}\sum_{X\in\mathcal{N}} ||(K[RX+t]) - (K[\hat{R}X+\hat{t}])||, \tag{6.3}$$

where K represents the calibrated intrinsic camera parameters.

6.3 Experiment one: Pose estimate tracking using extended Kalman filter

6.3.1 Aim

The aim of this experiment is to use vertex detection and the extended Kalman filter for pose estimation and tracking of the camera. We use the vertices detected by the convolutional neural network. The vertices have their location details in the image and the probability scores from the channels. We assume that the pylon is within the field of view of the camera. We want to answer the question "is the extended Kalman filter and vertex detection sufficient for the successful tracking of the camera relative to the pylon?" A successful implementation of camera tracking will serve to provide the robot with tracking capabilities while performing inspection.

6.3.2 Method

In this method, an extended Kalman filter and keypoint vertex detection from the convolutional neural network are used for tracking the position and orientation of the camera relative to the pylon.

We use the vertex detection from the convolutional neural network to detect the vertices on the pylon, which we described in Chapter 4. The convolutional neural network has 78 channels and each channel is assigned to a particular vertex. The vertex comes with a probability score and a threshold is set for accurate identification of a vertex. The vertex is the image keypoint which serves as the measurement for the extended Kalman filter and needs to be associated with an accurate correspondence of the world model.

We focus on minimising the distance between the projected 3D model coordinates and the image coordinates. The motion model used is the random walk model that was introduced in Chapter 3. The motion model contains the translation t_n and the rotation perturbation s_n as the variable we track while the quaternion always meets the non-linear constraint. The state transition model shows the rate of change of the pose estimate and is given as

$$g(u_n, x_{n-1}) = x_n = \begin{pmatrix} t_n \\ s_n \end{pmatrix}. \tag{6.4}$$

The observation model provides the relationship of registering the world coordinates to the image scene and is given as

$$z_n = h(x_n). \tag{6.5}$$

The 3D world coordinates are known and fixed. The camera is calibrated and we know the intrinsic camera parameters.

The extended Kalman filter is implemented in two stages, namely the prediction and the update stages. The mean and covariances of the states are propagated in these stages. The prediction state uses the state transition model, which for this work is the movement of the camera as in Equation 6.4, to predict an estimate of the a priori mean and a priori covariance. The update stage uses the observation model (Equation 6.5), which incorporates the measurement for residual error, to update the a priori mean and covariance pose estimate for the a posterior pose estimate.

The mean state is determined in the prediction and the update stage and in this work it provides the pose estimate of the camera relative to the pylon. The measurement is the vertex and the residual error in the update stage is the difference between the world points and the vertices.

The initialisation of the extended Kalman filter is the first step that has to be done. The initialisation can be done by knowing the state at the starting time or the states can be experimentally defined. With the measurements being accurate enough for the pose estimate to converge by the tracker. Therefore, the initialisation for the covariances and states is set experimentally through different trials.

After initialising the covariance and the state, with the equation

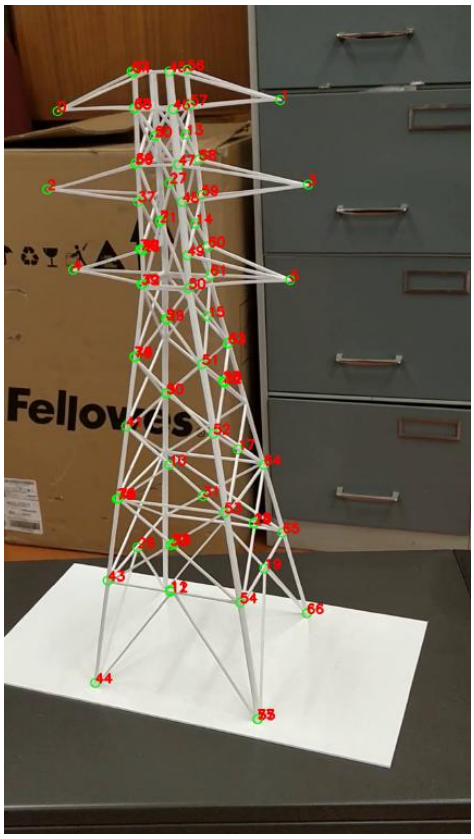
$$\hat{x}_n = g(u_n, \tilde{x}_{n-1}), \tag{6.6}$$

the mean is propagated to obtain the a priori pose estimate, $\hat{x}_n$, and with the equation

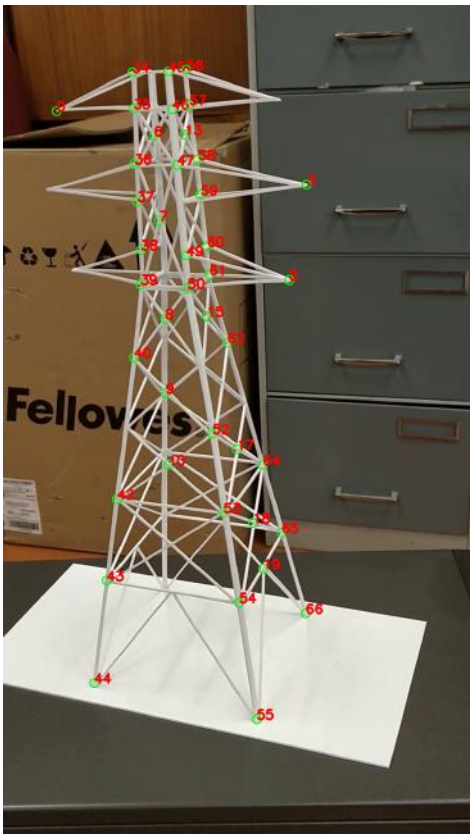
$$\hat{\Sigma}_n = (\Omega_n + G_n \tilde{\Sigma}_{n-1} G_n^T), \tag{6.7}$$

the covariance is propagated to obtain a priori pose estimate covariance $\hat{\Sigma}_n$, with Ω_n as process noise covariance and G_n as the Jacobian of the transition equation. This is the prediction stage of the extended Kalman filter

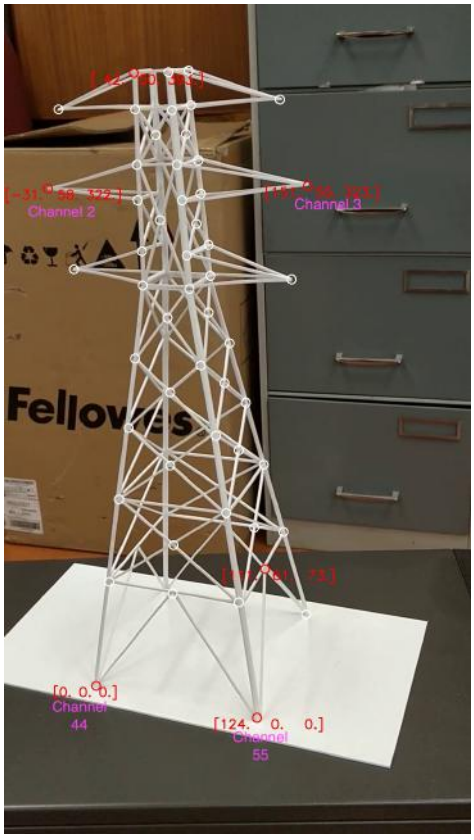
The update is the stage at which we propagate the mean and covariance by adding the measurement component. The measurement in this work is the feature from the vertex detection. Also, the model is the world coordinates of the vertices on the pylon. Therefore for the comparison to be possible, we register the 3D coordinates onto the 2D scene. The residual error component for the extended Kalman filter is obtained as the difference between the registered points and the vertex features. However, we cannot find the difference until we ensure that a registered point and the vertex features are exactly the same. This leads to the correspondence problem sometimes called data association. Data association is about establishing correspondence between points, which is ensuring an image keypoint has a match among the world points. Thus we are interested in finding the inlier measurement vertices that contain useful information and filter out the outliers. Each vertex detected has the location and its probability given, which provides the degree of confidence that the convolutional neural network rightly predicted the right vertex in the right channel as expected. A channel is uniquely assigned to a specific vertex and the vertex has a strict correspondence to a world point. With a high probability score, we filter out the unwanted vertices because they fall below the set threshold score.



(A) All the vertices detected by the CNN with channel information.



(B) Vertices with their channel information after filtering using confidence score of 0.9.



(C) Some vertices with associated channels.

FIGURE 6.6: Indicates the vertex specified to each channel as is assigned for data association.

Each world coordinate is known and needs to have a correspondence to a particular vertex. For instance, Figure 6.6 shows that the world coordinate point $(0, 0, 0)$ is always associated with channel 44 of the convolutional neural network output, and $(151, 55, 323)$ is associated with channel 3 of the convolutional neural network output. This is done for all the convolutional neural network outputs. Correspondence is established as each vertex is also assigned to a specific channel and a high confidence score is used to filter out the vertices.

The Kalman gain, K_n, is the weighting factor for the filter and is given as

$$K_n = \hat{\Sigma}_n H_n^T (Q_n + H_n \hat{\Sigma}_n H_n^T)^{-1}. \tag{6.8}$$

To obtain the Kalman gain we need the measurement noise covariance, Q_n, and to calculate the Jacobian of the measurement, H_n, based on the vertices that were associated with a world point. The Jacobian is numerically computed as the rate of change of the vertices with respect to the change in the pose estimate.

It should be noted that the number of the vertices that passes the threshold will vary in each iteration, therefore the size of the Jacobian will change.

With the Kalman gain and the residual error difference between the image keypoints and the model, we update the a priori pose estimate as the a posterior pose estimate $\tilde{x}_n$ using

$$\tilde{x}_n = \tilde{x}_n + K_n \underbrace{(z_n - h(\tilde{x}_n))}_{\text{residual error}}. \tag{6.9}$$

Then the a posterior covariance $\tilde{\Sigma}_n$ is also updated using the equation

$$\tilde{\Sigma}_n = (I - K_n H_n)\hat{\Sigma}_n. \tag{6.10}$$

We have a pose refinement to fine tune the a posterior pose estimate obtained from the Kalman filter. The refinement uses an iterative closest point technique. It is an iterative algorithm used to obtain the minimal solution between sets of points. This helps to match between the point sets. It involves alternating between determining the error and the pose estimate. During the error determination, we use the given translation and orientation to find the error. In the pose determination, we have to find the translation and orientation. The set of 3D coordinates of the pylon are known, while the set of the 2D coordinates are obtained from the vertex

detection. With the pose estimate obtained from the Kalman filter, we register the 3D coordinates into 2D space. The closest points between the sets are found. The pose estimate is updated. We keep iterating the steps until the error is below a certain threshold.

In Algorithm 3 we provide a step by step guide to the method.

Algorithm 3 Vertex detection and extended Kalman filter.

set Measurement covariance Q_0;
set Noise covariance Ω_0 ;
set Process covariance Σ_0;
set Initial pose estimate x_0;
Predict pose estimate $\hat{x}_n = g(u_n, \tilde{x}_{n-1})$;
Predict covariance $\hat{\Sigma}_n = (\Omega_n + G_n \tilde{\Sigma}_{n-1} G_n^T)$;
Project the 3D coordinates to 2D coordinates $h(\hat{x}_n)$);
Find the correspondence between vertex with high score z_n and $h(\hat{x}_n)$;
Calculate the measurement Jacobian $H_n = \frac{\partial z}{\partial x_n}$
Determine the Kalman gain $K_n = \hat{\Sigma}_n H_n^T (Q_n + H_n \hat{\Sigma}_n H_n^T)^{-1}$
Update the pose estimate $\tilde{x}_n = \hat{x}_n + K_n(z_n - h(\hat{x}_n))$
Update the state covariance $\tilde{\Sigma}_n = (I - K_n H_n)\hat{\Sigma}_n$
Refine with 3D-2D iterative closest point technique

6.3.3 Discussion and Analysis

The technique is done as a hard assignment of the vertex as each channel of the neural network is assigned to a specific vertex. The vertices are visually similar and the convolutional neural network learns from their location only, therefore it is difficult to differentiate between the vertices.

In this technique we assume initialisation by iterating the initial parameters until we get good results. We initialise the camera position and orientation to 1.00mm, 1.00mm and 1.00mm for the x, y and z-direction and 1°, 1° and 1° for the roll, yaw and pitch angles.

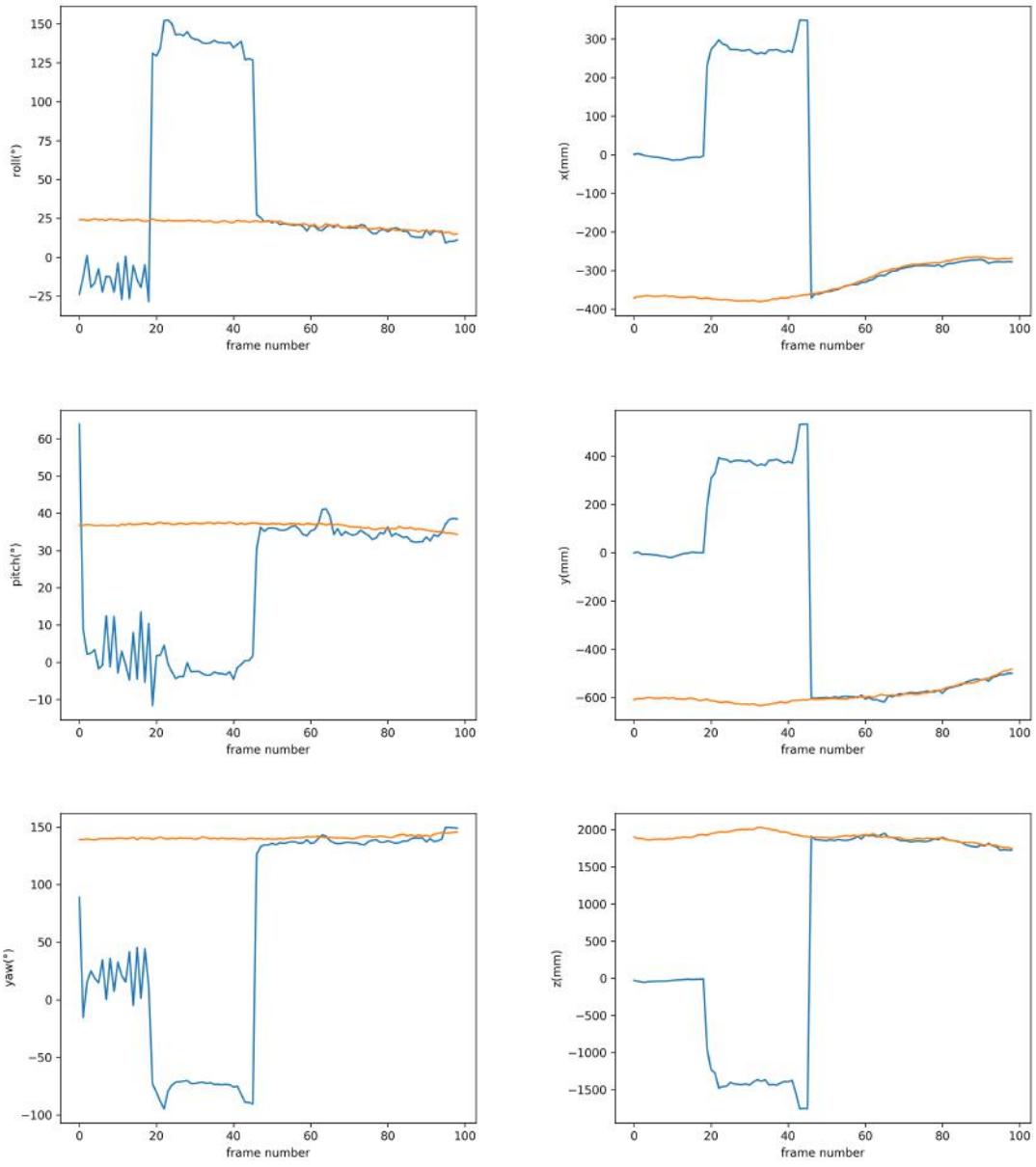


FIGURE 6.7: The position and orientation of the camera relative to the pylon. The x, y and z of the camera position and the roll, yaw, and pitch of the camera orientation. The ground truth pose in orange and the estimated pose in blue are shown.

The effectiveness of the technique is shown in Figure 6.7, where we use data from sequence I (Section 6.1) to compare the tracking of the filter. The pose estimate trajectory starts converging with the ground truth after frame 42. The speed of the tracker is determined to be 0.206s for each frame. When carefully observed in the figure, the tracker is going above and below the ground truth values as it jitters. The registration of the model does not perfectly fit onto the pylon in the

image, as the data association does not filter out the outlier vertices perfectly. As noticed in the discussion of Chapter 4, even with a high probability score outliers were still detected.

Figure 6.8 shows the representation in the tracking error of the pose estimation using the APD, ACPD and the reprojection error.

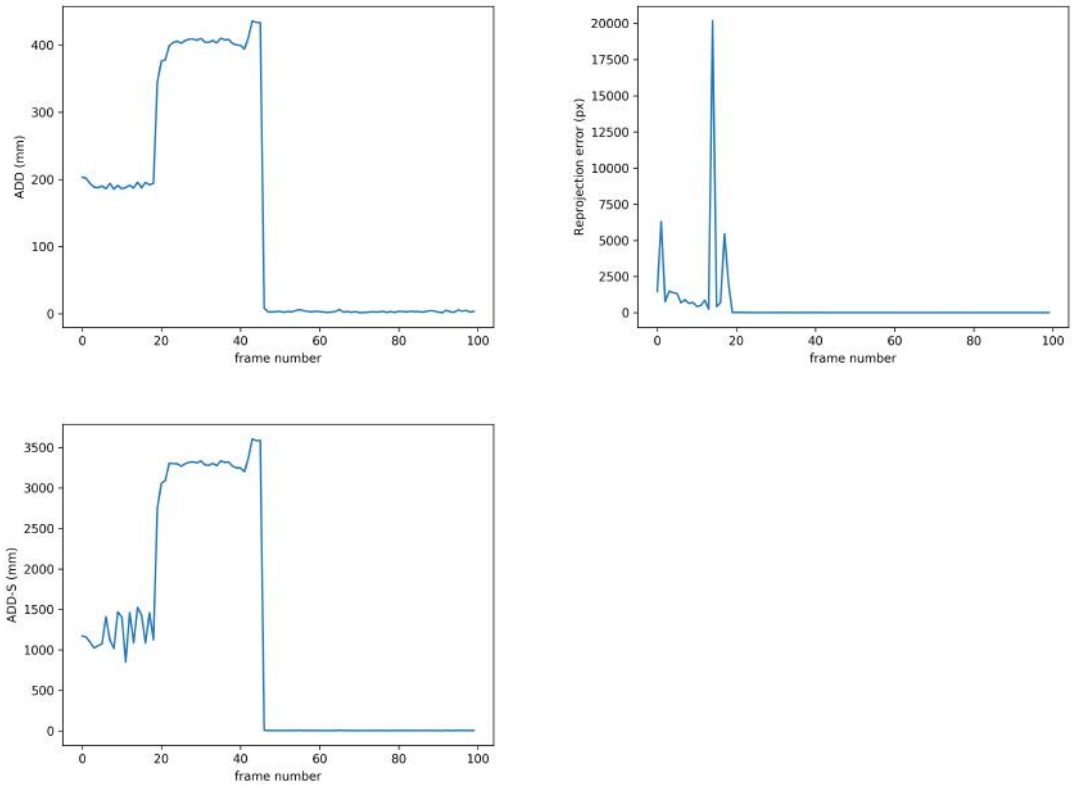


FIGURE 6.8: The pose error metric comparing the estimated and ground truth pose of the camera relative to the pylon.

The method requires that the vertex detector from the neural network be accurate. The threshold score has to be carefully selected, as choosing too low a value means a lot of wrong correspondences are input into the algorithm. If the threshold is too high, then there are not enough correspondence points for the tracking.

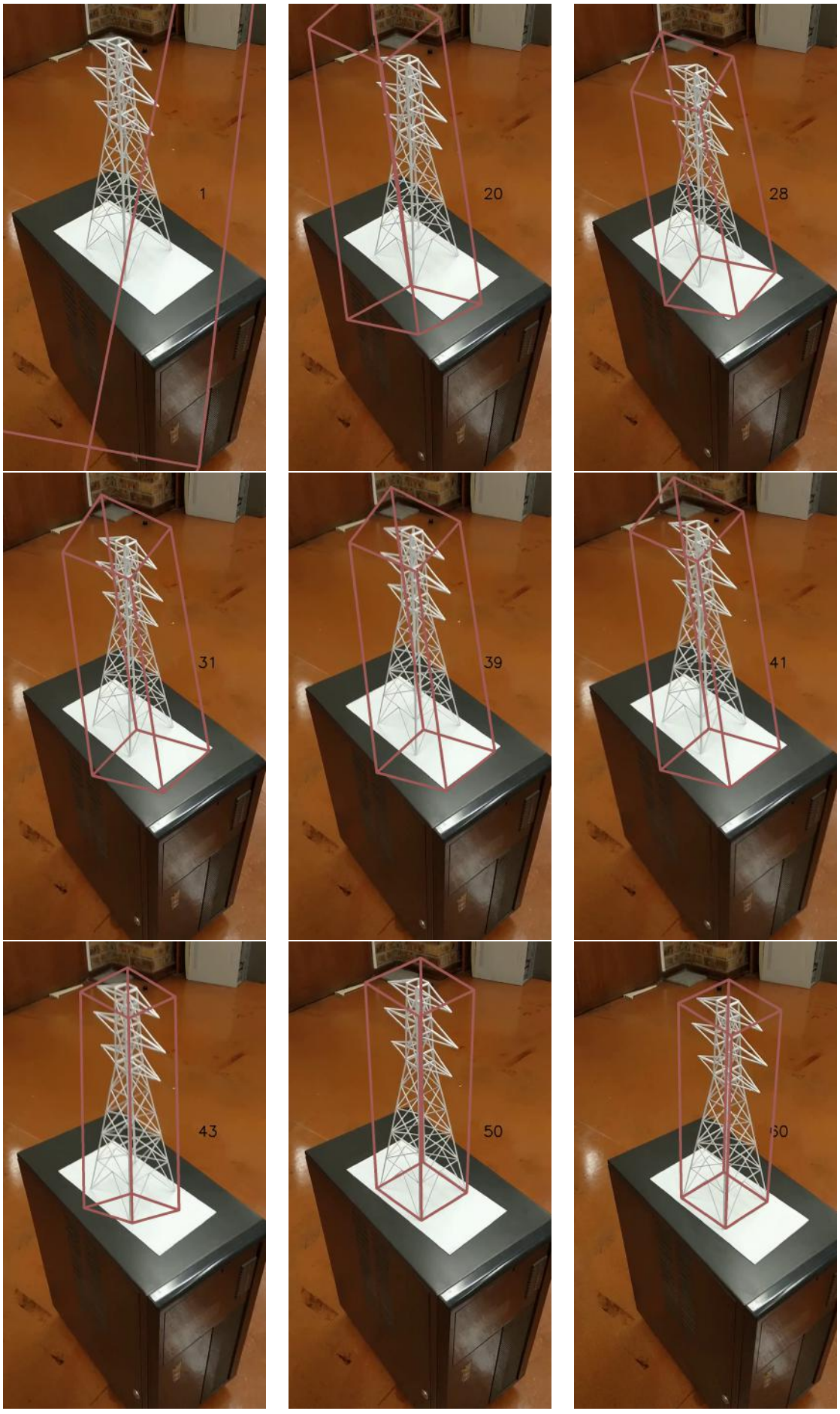


FIGURE 6.9: Results of the 3D model of the pylon registering onto the image of the pylon.

Figure 6.9 depicts the performance progression of our developed algorithm. As observed in the first frame in Figure 6.9, the estimated 3D model at initialization is completely out of the 2D image space of the registered pylon image due to maximal error. As this error is minimised in subsequent iterations, the algorithm gradually converges with the estimated 3D model being finally registered into the 2D image space at the optimal convergence.

We noticed the data association is not always accurate, as sometimes the neural network does not output the same required vertex. The technique itself proved to be insufficient for the pose estimation therefore we explored the addition of geometric constraints.

6.4 Experiment two: Geometric hashing and extended Kalman filter for pose estimation and tracking

6.4.1 Aim

The aim of this experiment is to apply geometric constraints to vertex detection and an extended Kalman filter to find the six degree-of-freedom pose estimate of the camera relative to the pylon. We will use the vertices and their location details obtained from the convolutional neural network. A geometric hashing method is available for object recognition at the start of the tracking. Can we solve pose estimation in a single image? Can the output from the solution of the pose estimation in a single image be used for initialisation of the tracker? Finally, can an extended Kalman filter track the movement of the camera relative to a pylon?

6.4.2 Method

To seek solutions to the questions raised above, we use a geometric hashing method to find the pose estimate from a single image. The extended Kalman filter is subsequently used to track the movement of the camera.

We initialise the tracking process by applying pose estimation to a single image using geometric hashing. To do this we need a world model of the pylon and its features extracted using vertex detection to be stored in a hash table for the training stage. The input is an image scene with a pylon in it. This image scene is captured with a RGB camera. The features obtained from this image scene are extracted to vote for pylon matching in the recognition stage of the geometric hashing method.

For pose estimation via tracking using an extended Kalman filter, we will require a motion model that provides information for the movement of the camera and an equation that relates the measurement of the image scene vertices to the world model vertices.

We use a random walk model as the motion model for this experiment. Similar to what we showed in Chapter 3, the state transition model is given by

$$g(u_n, x_{n-1}) = x_n = \begin{pmatrix} t_n \\ s_n \end{pmatrix}. \tag{6.11}$$

The 3D coordinates of the object are known and fixed. The observation model is given as

$$z_n = h(x_n) + e_n. \tag{6.12}$$

The extended Kalman filter is used for propagation of the mean and covariance and has two stages, namely the prediction and the update stages. The states used in this work are the translation t_n and the perturbation of the rotation s_n. The state transition model in Equation 6.4 is used in the prediction stage to estimate the a priori mean and a priori covariance. The observation model (Equation 6.5) and the measurement is used in the update stage to estimate the a posterior pose estimate and a posterior covariance.

The first step is the initialization of the extended Kalman filter. The initialization can be done if the state at the starting time is known. In this experiment a model-based technique, which finds the single image pose estimation method for the initialization, is used. The model-based technique is geometric hashing.

A detailed description of the geometric hashing method is provided in Chapter 5. The method has a training and a recognition stage. In the training stage the 3D

coordinates of the pylon, which is manually annotated as shown in Figure 6.1, are stored. Images are picked from different views to serve as the models. All possible basis sets from the models are generated, then both the model and its basis sets are stored in a hash table. Therefore, training involves storing the geometric information of the model pylon into a hash table.

At the recognition state, all the possible basis sets from the image scene are generated. Then we vote for the model and basis set that matches the image scene to the stored model. This helps us to retrieve the training model thereby establishing a correspondence between the pylon in the image scene and the model that was stored in the training stage.

During this experiment for initialisation, the geometric hashing method is invoked. Geometric hashing helps us to retrieve the 3D coordinates in the training stage using the image scene 2D coordinates from the recognition stage. Thus, we have a 3D to 2D correspondence issue, and it is solved as a perspective-n-point (PNP) problem.

We have initialised the covariance and the state, therefore in the prediction stage the a priori pose estimate $\hat{x}_n$ is obtained with the equation

$$\hat{x}_n = g(u_n, \tilde{x}_{n-1}), \tag{6.13}$$

and the a priori pose estimate covariance $\hat{\Sigma}_n$ is obtained with

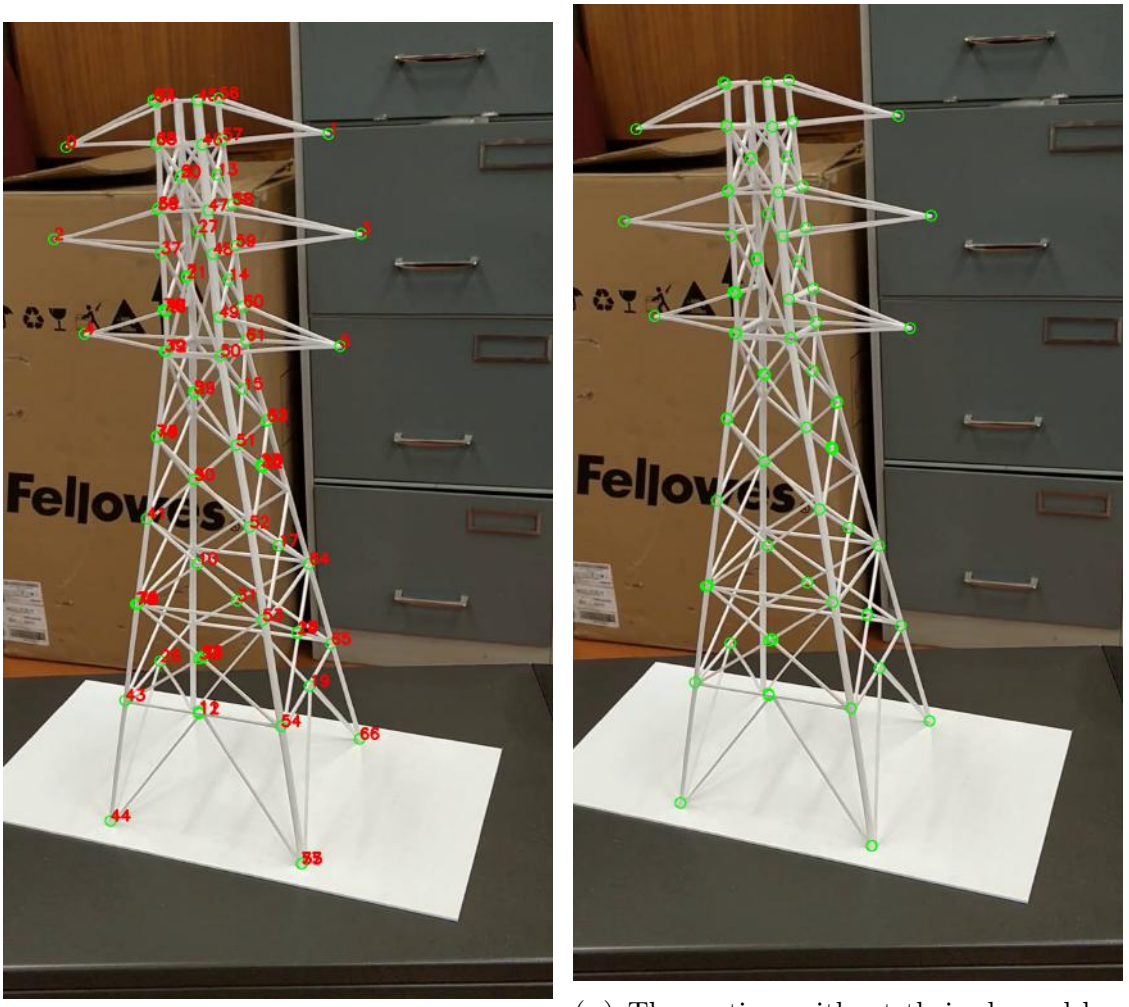
$$\hat{\Sigma}_n = (\Omega_n + G_n \tilde{\Sigma}_{n-1} G_n^T), \tag{6.14}$$

where Ω_n is the process noise covariance and G_n is the Jacobian of the state transition model.

The update is the stage where we propagate the mean and covariance by adding the measurement component. The measurement in this work is the vertices from the vertex detection. The world coordinates of the vertices on the pylon are the model. The 3D world coordinates are registered onto the 2D image scene using the a priori pose estimate. Then the residual error component for the extended Kalman filter is determined, which is the difference between the registered points and the vertex features. The difference can be determined only if the registered point and the vertex have correspondence. The process of establishing such correspondence is called data association.

Data association involves finding correspondence between the measurements and the world pylon model. Data association helps us identify which of the measurements belongs to the model. We have to ensure that each image keypoint has a corresponding world point and a relationship exists between the measurement and the model.

The output of the neural network for each vertex is associated with a unique channel. As we noted in Chapter 4 the convolutional neural network is able to detect vertices but sometimes assigns them to the wrong channels. Therefore we ignore the channel information and use the location information only (Figure 6.10). To find the correspondence we use the closest world point to a vertex.



(A) All the vertices with their channel details.

(B) The vertices without their channel because we are interested in only the location of the vertices.

FIGURE 6.10: Indicates vertices with their channel number and location.

An optimal solution for the correspondence problem is obtained via the use of the random sample consensus (RANSAC) algorithm. The algorithm helps to find inlier vertices and deals with the outliers. Using the nearest neighbour approach, we establish correspondence between the inlier vertices and the world points. For the data association, where we find measurements relating the model and image, we register the 3D coordinates with an a priori pose estimate onto the image scene and then use RANSAC to determine the inlier points and deal with the outliers.

The Kalman gain, K_n, is the weighting factor for the filter and is given as

$$K_n = \hat{\Sigma}_n H_n^T (Q_n + H_n \hat{\Sigma}_n H_n^T)^{-1}, \tag{6.15}$$

where H_n is the measurement Jacobian and Q_n is the measurement noise covariance.

We determine the measurement Jacobian for the vertices that are correctly identified as inlier points during correspondence with the world points. To obtain the Jacobian we need to calculate the Jacobian based on the vertices that are associated with a world point. The Jacobian is numerically computed as the rate of change of the vertices with respect to the change in the pose estimate.

The Kalman gain, the residual error difference between the vertices and the model, and the a priori pose estimate are used to determine the a posterior pose estimate using the equation

$$\tilde{x}_n = \hat{x}_n + K_n \underbrace{(z_n - h(\hat{x}_n))}_{\text{residual error}}. \tag{6.16}$$

Then the a posterior covariance is also updated using the equation

$$\tilde{\Sigma}_n = (I - K_n H_n)\hat{\Sigma}_n. \tag{6.17}$$

We use a pose refinement method to refine the a posterior pose estimate and obtain an optimum value. The pose refinement is done with an iterative closest point (ICP) algorithm. The a posterior state estimate sometimes exhibits drift due to the noise from measuring the 3D world coordinates and we refine the estimate to eliminate the drift.

Inlier points, obtained from data association, are good for establishing correspondence and can be used to calculate the a posterior pose estimate, but using them to

find the pose estimate at the refinement stage will result in local minima. Therefore, we use the whole vertices, which increases the representation and provides a better generalisation.

The refinement is done using ICP. The 3D points of the vertices, Π_i, serve as the global coordinates. The a posterior value obtained is used as a good estimate of the camera pose. The formulation of the objective function is given as

$$\min_{P} \sum_{i=1}^{n} ||P\Pi_i - \pi_i||, \tag{6.18}$$

with P the camera matrix and π_i the 2D coordinates.

The algorithm reaches convergence after several iterations of Equation 6.18. The initialization for the refinement stage is the a posterior estimate of the Kalman filter, and with a good initialization ICP converges to a better minimum.

In Algorithm 4 we provide the steps involved in the pose estimation and tracking.

Algorithm 4 Extended Kalman filter with geometric hashing

set Measurement covariance Q_0;
set Noise covariance Ω_0 ;
set Process covariance Σ_0;
if n == 0 **then**
  establish correspondence with geometric hashing
  solve PNP for the Initial pose estimation, x_0
end if
Predict pose estimate $\hat{x}_n = g(u_n, \tilde{x}_{n-1})$
Predict covariance $\hat{\Sigma}_n = (\Omega_n + G_n\tilde{\Sigma}_{n-1}G_n^T)$
convert the 3D coordinates into 2D coordinates $h(\hat{x}_n)$
Use RANSAC to find the inlier point $h(\hat{x}_n)$ and z_n
Calculate the measurement Jacobian $H_n = \frac{\partial z}{\partial x}$
Determine the Kalman gain $K_n = \hat{\Sigma}_n H_n^T (Q_n + H_n\hat{\Sigma}_n H_n^T)^{-1}$
Update the pose estimate $\tilde{x}_n = \hat{x}_n + K_n(z_n - h(\hat{x}_n))$
Update the state covariance $\tilde{\Sigma}_n = (I - K_n H_n)\hat{\Sigma}_n$
Refine with 3D-2D iterative closest point technique

6.4.3 Discussion and Analysis

We discuss the implementation of the details used for the initialization using geometric hashing and tracking with the extended Kalman filter.

Initialisation with geometric hashing

The sides of the pylon are planar, so any two views of a side are linked by a planar homography or projective transformation. In our work we use an affine transformation, which should be approximately valid as long as the camera is quite far from the pylon. Affine transformation can be a suitable approximation for projective transformations under certain viewing conditions [78–80] and can be used instead of a projective transformation. We require three non-collinear points to determine the affine transformation.

Our parameter choice for the geometric hashing is guided by the experiments in Chapter 5. The parameters hash bin size, quantization value and transformation are similar for both training and recognition stages, while the voting type is a parameter unique to the recognition stage. The weight of the hash bin is chosen as 0.5 with a quantization of 4. We use an affine transformation for the generation of the basis set for the geometric hashing method. Section 5.2 provides the mechanism for how to determine the basis sets and use it to transform the other features. All the possible basis sets are produced and stored in a hash table during an offline training process. The hash table represents a 2D array with the model and basis sets as the information.

The recognition stage is done online as part of the localisation framework. The number of basis sets produced is large and trying all of the basis sets is computationally expensive. Thus we select 100 random basis sets for trial to establish a match between a recognition basis set and a training basis set. During each trial, we generate 20 hypothesis candidates and pass them through the verification stage. In the verification stage, only candidates with a value less than the threshold and with a vote count of more than 25 in the accumulator are selected as accurate detections.

We set two thresholds. The first threshold is at 25 pixels for the root mean squared error between the basis set of the training stage and the basis set of

the recognition stage. There are many candidates that can pass this threshold and until the candidate passes through the verification it is considered as a false detection. Having two thresholds ensures that if a candidate fails the first threshold we move to the next basis set instead of wasting valuable time to do verification for a false detection. The second threshold is set after the verification is done with RANSAC and is set to 10 pixels.

We use the non-probabilistic voting technique for the voting process of geometric hashing because it is fast and relevant for an online system. The voting process of a non-probabilistic voting technique takes 2.12s for a search by a basis set during recognition while the search using the probabilistic voting technique takes 204s.

The effectiveness of the localisation framework depends on how well the geometric hashing is able to detect the object at the initialization of the detection. Therefore we attempted to determine the rate of successful detection for the geometric hashing over a certain number of frames in different video sequences.

We test the video sequences to see how geometric hashing will accurately detect the pylon and provide initialisation for the pose estimation tracking. An accurate detection is successful only if it passes the verification stage of the geometric hashing.

TABLE 6.1: Table for accurately detecting the pylon with geometric hashing.

	Sequence I	Sequence II	Sequence III
Successful detection	153	191	77
Total frames	200	200	200

We tabulate the results of the detection for the pose estimation using geometric hashing in Table 6.1. Data from sequence I is obtained by going around the pylon and it has the intermediate accuracy rate among the sequences. Data from sequence II are created by approaching the pylon from a distance and has the best accuracy rate as the frames are from the same view but with differences in scale. Data from sequence III has some occluded frames and it has the lowest accuracy rate, showing that geometric hashing does not work well with occluded images. We observed that the algorithm works best when the camera has a view of two faces of the pylon. Such a wide view provides the highest number of vertices for detection.

An issue observed with geometric hashing is that it fails when the neural network falsely predicts a vertex which then becomes part of the basis set. Therefore, there

is a need for doing multiple trials with different vertices forming the basis set, until a detection is passed by the verification stage.

The pylon used in this work has 78 vertices, but some pylons in an open environment have more than 300 vertices. This means an increase in the number of basis sets possible and thus an increase in the computational cost. Therefore, the vertices will need to be filtered out to reduce the rate of failure and computational cost so that only accurate vertices are used to form basis sets. In this experiment, we filter out the vertices using a probability score threshold from the vertex detection algorithm, and other possible ways of filtering can be done with only vertices that are connected with each other to form a basis set.

Tracking

We test the effectiveness of the tracking algorithm with manual annotation of the four corners of the rectangular stand of the model pylon used to estimate the ground truth pose. Data used for tracking is obtained from sequence I (Section 6.1) with the image frame size of 1920×1080 pixels.

Figures 6.11 and 6.12 provide information on the trajectory of the camera as it moves and the video data of the pylon is obtained from sequence I (Section 6.1). The figures show that the proposed methods succeed and are quite accurate, with mean camera position errors of 4.43mm, 2.17mm, 25.91mm for the x, y, and z directions, and mean errors of 0.58°, 1.79°, and 4.53° for the roll, pitch and yaw orientations.

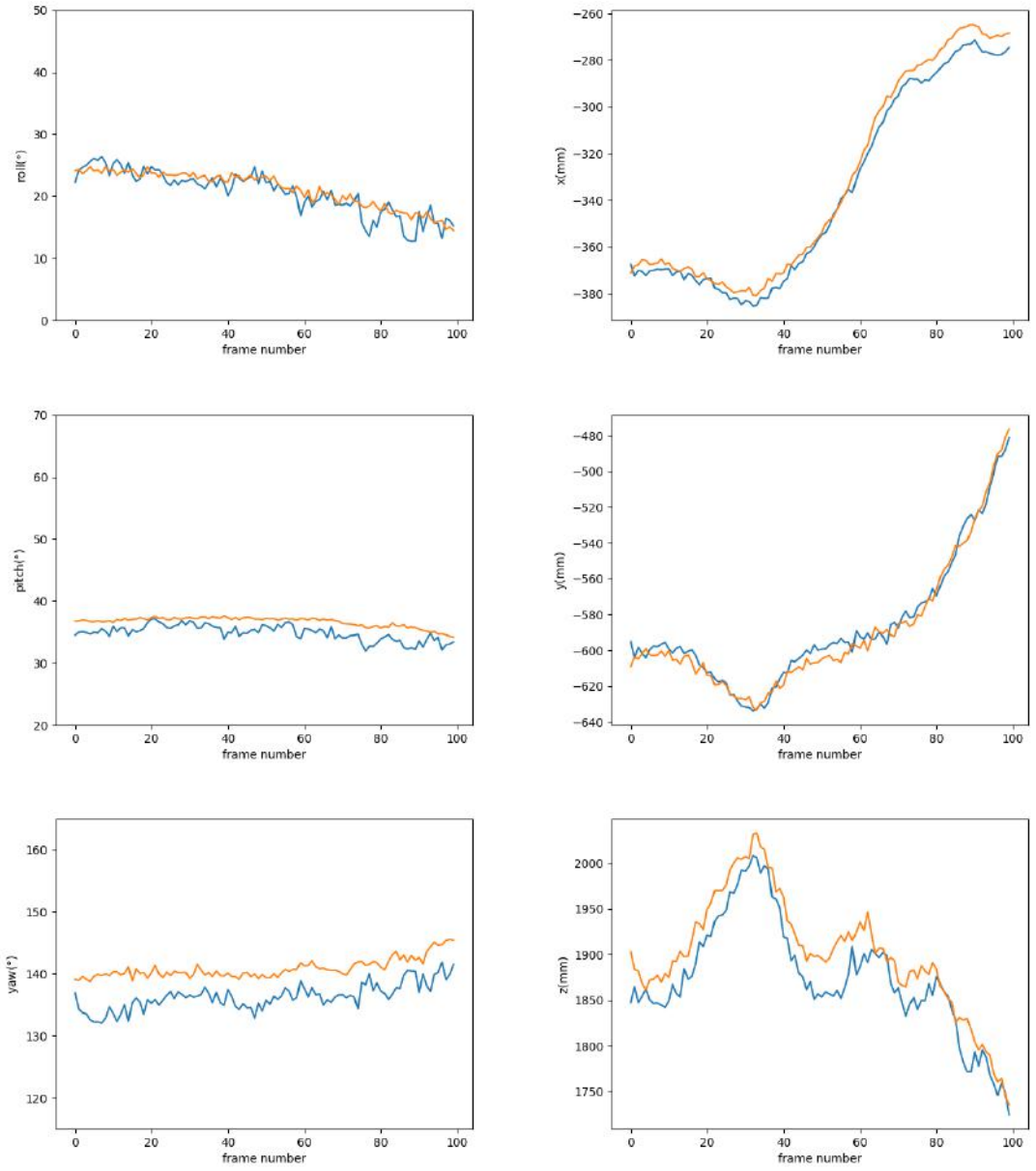


FIGURE 6.11: The position and orientation of the camera relative to the pylon during the tracking process. The orange represents the ground truth pose and the blue represents the estimated pose.

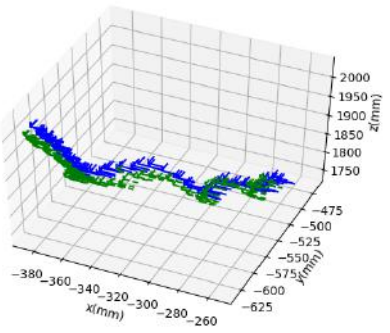


FIGURE 6.12: The translation component of the camera pose in the x, y and z axis fused together when tracking. The blue represents the ground truth and the green represents the estimate of the camera position.

The pose error for the average distance in the 3D space and the reprojection error are shown in Figure 6.13. The APD and ACPD metrics produced identical results. This shows that the tracker is able to handle the symmetrical nature of the pylon. The success of the tracker in handling symmetry is because of iterative algorithms such as RANSAC used during data association and ICP for refinement. Moreover, RANSAC deals with the outliers and smooths out jitter.

We see that the reprojection error is small and well within good accuracy levels, with an average error value of 5.03 pixels.

The extended Kalman filter is successful in tracking and smoothing of the pose estimates in each frame. The tracker takes 0.204s for each iteration.

Some snapshots of the results of the registration of the pylon are shown in Figure 6.15, with an image of the initialization with geometric hashing in Figure 6.14. We also provide a video of the registration of the 3D model to the pylon image using the estimated pose of the camera obtained in each iteration at https://youtu.be/vqVtfkYOi8A.

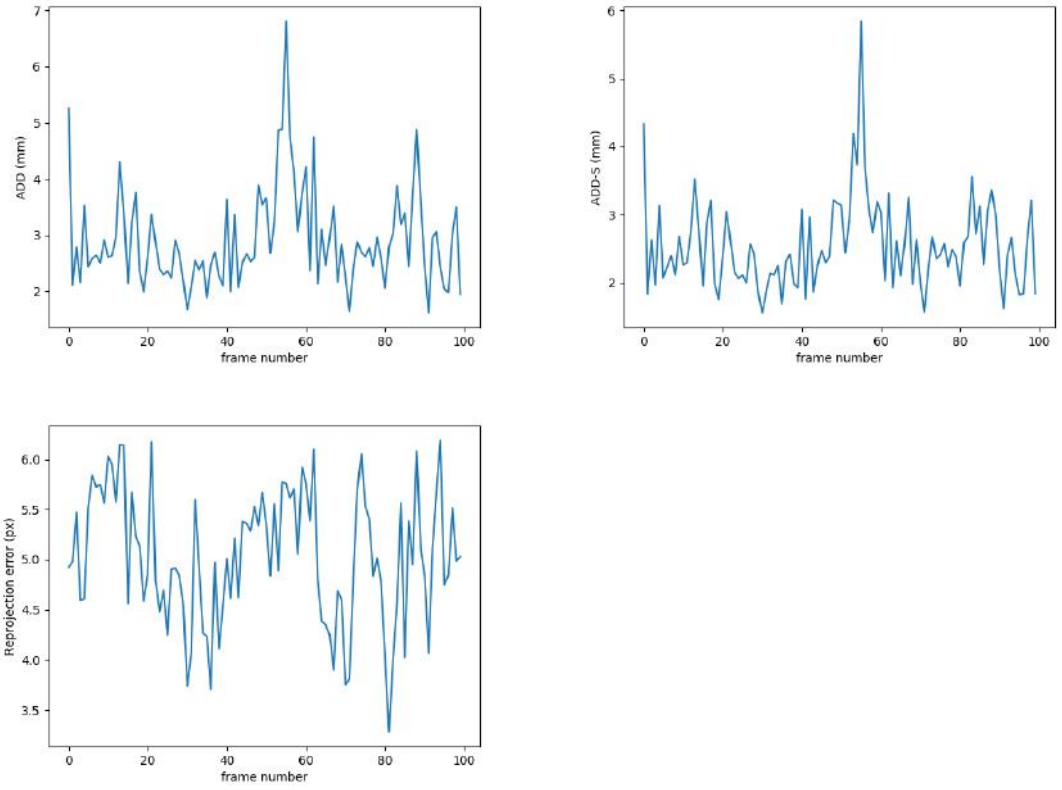


FIGURE 6.13: The pose error metric comparing the estimated and ground truth pose.

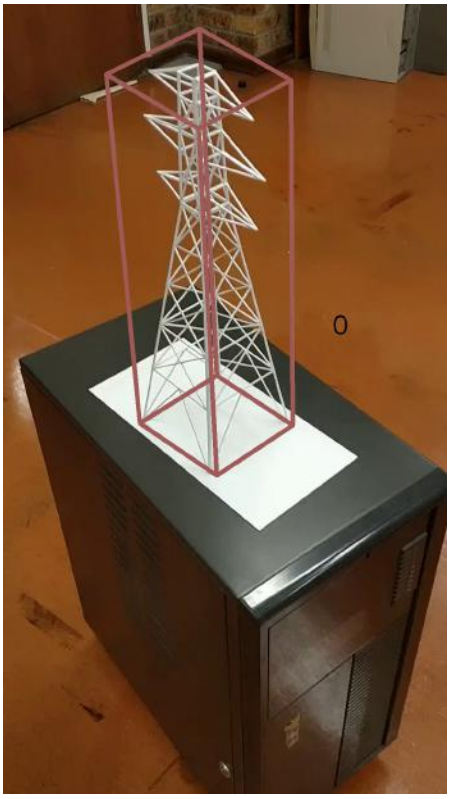


FIGURE 6.14: Initialization using geometric hashing.

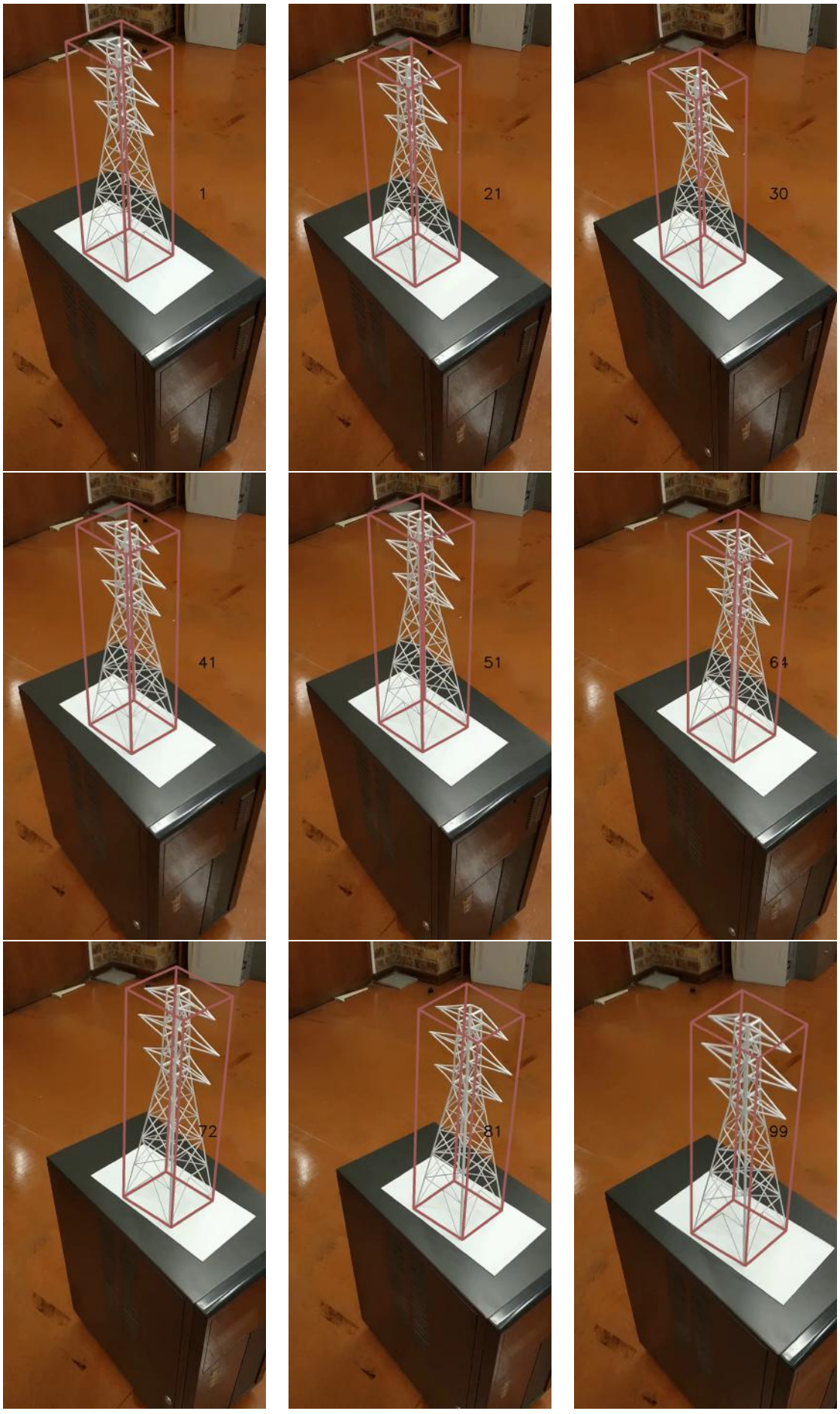


FIGURE 6.15: Results of the 3D model of the pylon registering onto the image of the pylon.

The algorithm breaks down when there are no usable measurements. As RANSAC cannot determine inlier points, the tracker fails as data association cannot establish correspondence. Thus there are no usable measurements to feed the tracker.

For effective usage of the algorithm, the pylon should be in full view of the camera. With a full view of the pylon, the vertex detection will determine the vertices and RANSAC can find inlier points to be used for correspondence as lack of the inlier points will make them fail.

At any given point while tracking with the pylon, the view shows different vertices. Some of these vertices are false and are due to the apparent crossings of the bars in a projected image. This makes the refinement and outlier rejection necessary. However, when an outlier rejection and refinement process is done the tracker is robust even with a fluctuating number of vertices detected.

The method presented in this experiment provides a localisation framework for the navigation of a robot while conducting an inspection on or around a pylon. When the robot arrives an inspection site, a pylon is detected to be within a view of the camera. The 3D model is already available to the robot then we switch to geometric hashing to lock the initial pose estimate and subsequent tracking of the camera with the extended Kalman filter. Using this method the robot will have six degree-of-freedom pose estimate of the camera relative to the pylon to aid with inspection.

6.5 Experiment three: Tracking with vertex detection heatmaps and gradient descent

6.5.1 Aim

We use heatmaps from the convolutional neural network and a gradient descent approach for tracking the six degree-of-freedom camera pose. We assume that there is an initialisation of the pose estimation using a method like geometric hashing. We ask the question can we track the movement of the camera by utilising a scoring method that checks for maximisation of the sum of probability scores of the vertices, thereby eliminating the need for model fitting and outlier rejection algorithm?

6.5.2 Method

This method is a tracking algorithm. The tracking has an initialization that is obtained using a method like geometric hashing. Geometric hashing is introduced in Chapter 5 and was used in experiment two (Section 6.4) for six degree-of-freedom pose estimation. The method requires a pylon model and its features obtained using vertex detection to be stored in a hash table during the training stage. At the recognition stage, an image scene and its features are obtained and used to establish a correspondence with the information from the hash table, using a voting technique.

The convolutional neural network in Chapter 4 outputs 78 single channel heatmaps before inference. Each single channel heatmap has heatmap information of the probability score and at times the maximum probability score corresponds to the wrong vertex. To obtain a vertex keypoint, an inference unit was used for maximisation and summation of the single channel heatmaps and then the heatmaps are concatenated into a single summed heatmap. We use the single channel heatmaps to track the pose.

After obtaining the initial pose estimate, we track the movement of the camera relative to the pylon. We use the heatmap from each channel of the convolutional neural network output.

To achieve tracking we register the 3D model vertices onto single channel heatmaps of the convolutional neural network. The 3D model is comprised of the 3D coordinates of the vertices. The registration of the 3D model vertex onto the single channel heatmap is done with the initialization at the first frame and subsequently by the succeeding pose estimate. Each world vertex is associated with a channel of the neural network and the vertex is registered onto a single channel heatmap. For instance, the world vertex $(0, 0, 0)$ is associated with channel 44, so vertex $(0, 0, 0)$ is registered onto the single channel heatmap of channel 44. This is done for all the world vertices and their associated heatmaps.

The single channel heatmap sometimes has multiple scores in it and the maximum score can represent the wrong vertex. To obtain the pose estimation from only the maximum score of a single channel heatmap may not produce an optimal pose estimate.

The optimal solution is obtained as the summation of all the probability scores from the single channel heatmaps as a vertex approaches the location of maximum score. We are maximising the probability score while minimising the distance between the registered vertex and the maximum score. To make the problem a minimisation, we take the inverse of the scores; this ensures we minimise the inverse of summed scores instead of maximizing the scores. We use only single heatmaps that have high maximum probability scores. This is because we are minimising the distance and use only heatmaps that are accurate.

Alternatively, rather than iterating through multiple single channel heatmaps, which is computationally expensive, we adjust the inference unit by eliminating the need for maximisation before concatenating the heatmaps into a single summed heatmap (Figure 6.16).

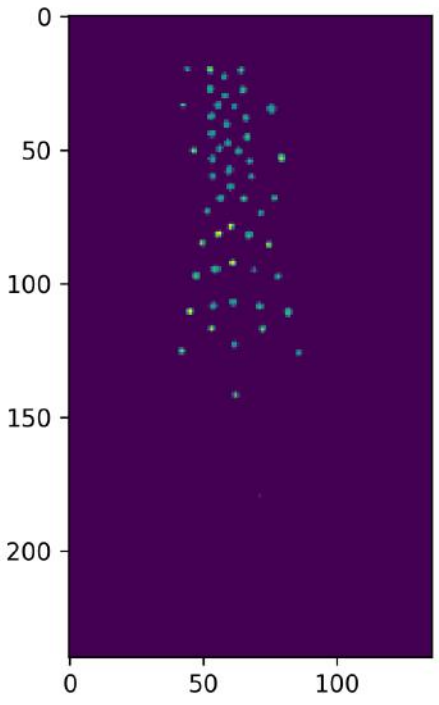


FIGURE 6.16: A summed heatmap with the whole vertices.

We register the 3D world vertices onto the summed heatmap. Each single channel heatmap has a maximum score and an associated world vertex. The associated world vertex is registered onto the summed heatmap and this is done for all the

vertices. The vertices are registered to a location on the summed heatmap and have probability scores.

To obtain the optimal pose estimate we determine the maximisation of the sum of all probability scores of each registered vertex while minimising the distance between the registered vertices as their associated maximum scores.

The difference between the above procedure and the one outlined at the beginning of the section is that instead of iterating through each heatmap separately, we iterate through a single summed heatmap.

Therefore, the objective function is defined as

$$e = \sum_{i}^{N} \alpha + ||\pi_i - \pi_i^p|| \tag{6.19}$$

where i is the channel of the heatmap, α is the inverse of the score c at the projected point, π_i is the projected point, and π_i^p is the peak point in a heatmap.

FIGURE 6.17: An example of single channel heatmap before and after blurring.

We use a gradient descent technique to obtain the minimum pose estimate at each frame. Due to the sparsity of the heatmap, obtaining the gradient becomes difficult. Thus we spread the intensity by blurring the heatmap using a Gaussian filter (Figure 6.17).

Finally, an iterative closest point algorithm is used to refine the pose estimate. The algorithm uses a set of 2D points of the keypoint vertex detection method and a set of 3D points from the pylon model. The algorithm iterates through the process by obtaining the error between the vertices and the model. Then it determines

the closest points using a nearest neighbour approach before updating the pose estimate. The iteration continues until the error is below a certain threshold.

6.5.3 Discussion and Analysis

We use a gradient descent technique to obtain the six degree-of-freedom pose estimate of the camera relative to the pylon at each frame. The learning rate is set at 10^{-5} for the gradient descent. Only single heatmaps that have a point with maximum probability score of 0.95 are used.

The effectiveness of the tracking algorithm is tested using the ground truth pose estimate obtained from the base corners of the stand of the pylon. The data is taken from sequence I (Section 6.1) as the camera moves around the pylon.

The trajectory of the camera movement is shown in Figure 6.18. The mean error of the camera pose translation is given as 4.64mm, 4.94mm, and 34.20mm for the x, y and z-directions and the mean error of the camera pose orientation is given as 1.86°, 2.32° and 3.51° for the roll, pitch and yaw angles.

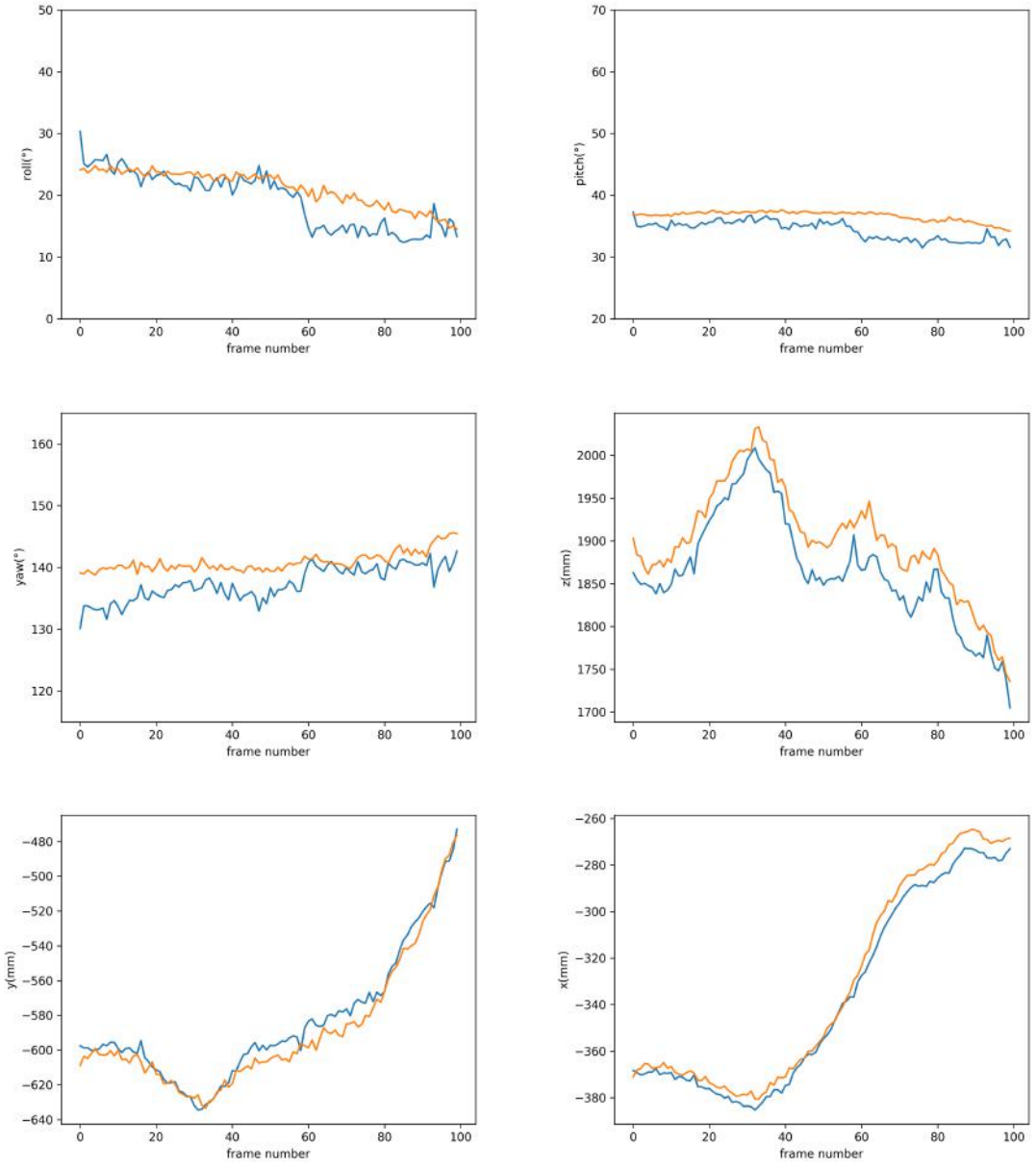


FIGURE 6.18: The position and orientation of the camera relative to the pylon. The orange is the ground truth pose and the blue is the estimated pose.

The algorithm is evaluated using the APD, ACPD and reprojection error metrics. These metrics show the error in the method and are provided in Figure 6.19. The similarity between the result from APD and ACPD shows the algorithm is effective in handling symmetrical objects like the pylon.

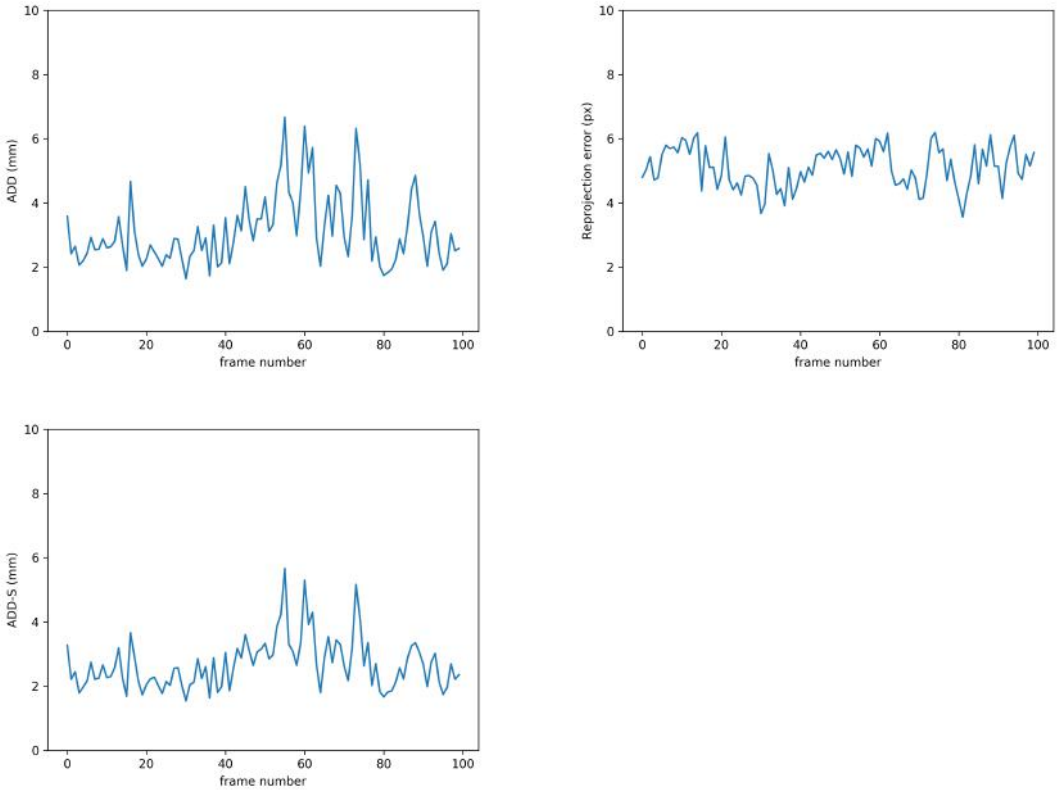


FIGURE 6.19: The pose error metric comparing the estimated and ground truth pose.

Instead of using a RANSAC for outlier rejection and nearest neighbour to establish correspondence as was done in experiment two, the technique uses the heatmap to adjust the pose estimation with its heatmap information. Figure 6.20. The figure shows a 3D bounding box on the pylon in the When the movement of a camera between frames is large the tracker tends to drift away. Refining the pose with ICP improves the performance and aids in eliminating the drift.

We provide some snapshots from the results of the tracking process in the image scene, showing the tracker is able to track the camera relative to the pylon by registering the 3D model of the pylon onto the image scene.

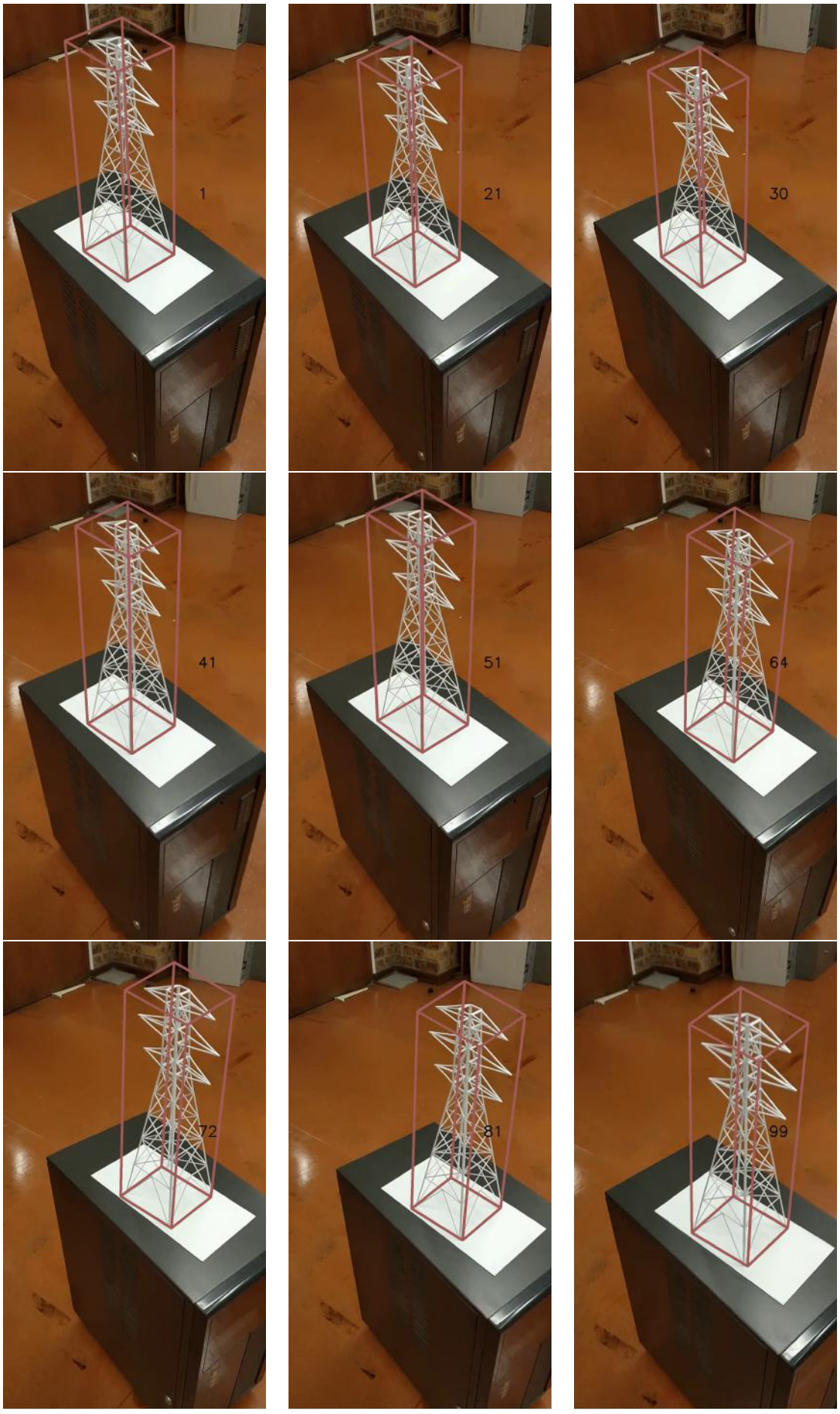


FIGURE 6.20: Results of the 3D model of the pylon registering onto the image of the pylon.

The speed of the tracker is 2.08s. Thus, the speed of the tracker is slower compared to the extended Kalman filter. The speed is affected by the iterations due to the gradient descent and trying to optimise so many vertices.

We present a tracking method for the movement of the camera relative to the pylon. The tracker is an alternative to the extended Kalman filter and can be used as part of the localisation framework for the navigation of the robot.

6.6 Experiment four: Tracking of outdoor electricity pylons

We experiment with testing the tracking on outdoor electricity pylons. To enable us to achieve tracking of the 3D coordinates of the vertices of an electricity pylon, the 3D coordinates have to be known. We use a laser scanner to obtain the 3D model that provides the coordinates of the vertices.

With 3D coordinates of the vertices, we use geometric hashing to obtain the initial six degree-of-freedom pose estimate, and subsequently we use an extended Kalman filter to track the movement of the camera.

6.6.1 Data

In this section, we discuss the data we generated in order to test the performance of our algorithms. We discuss the use of a laser scan to build a 3D model, extracting the 3D coordinates of the pylon vertices, and how we are able to generate a sequence of pylon images.

3D model of an electricity pylon with a laser scanner

Laser scanners use light rays to scan objects and reconstruct 3D models of an object. We use the laser scanner [81] for the reconstruction of an electricity pylon. To undertake the task, we take individual scans of the pylon from different viewpoints. By taking scans from various viewpoints, we are able to capture the complete structure of the pylon. We took 14 distinct scans with the laser scanner.

The intricacy of the surface is taken into account while selecting the appropriate resolution for the laser scanner. The Z+F IMAGER 5010C laser scanner, which is employed for the scans in this work, is the laser scanner that was used to obtain such an acceptable resolution.

We employ the iterative closest points approach to register the numerous scans into a single point cloud. All of the scans must be in the same coordinate frame in order to achieve accurate registration. When capturing the scan, overlap areas that have sufficient detail have to be taken to ensure a unique solution can be obtained.

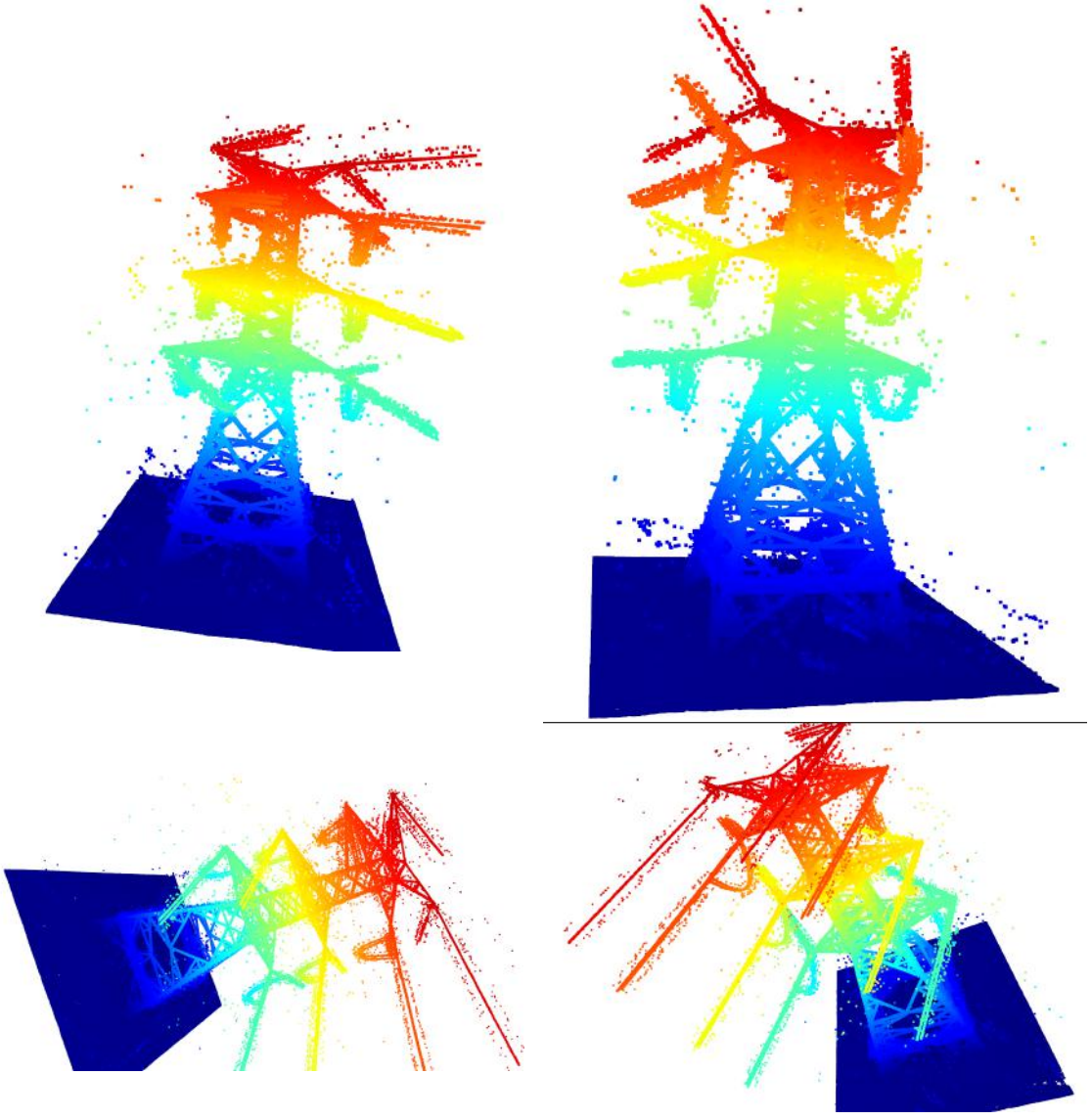


FIGURE 6.21: The 3D model of an electricity pylon. We present different views of the point cloud.

The point cloud of the electricity pylon comprises 49 million surface points. The scan captured the entire electricity pylon and is depicted in Figure 6.21. The dimensions of the electricity pylon is 40 metres high, 17 metres wide and 17 metres long.

Generation of a sequence

The power companies generally have control over electricity pylon data. For the work we need the 3D model of an electricity pylon. We use an electricity pylon that is located at Goodwood, Cape Town, South Africa. A 3D CAD drawing for the electricity pylon is adequate for the task, but the CAD model is held by the power companies and we could not obtain the drawings. Instead, we obtain the 3D model using a laser scanner.

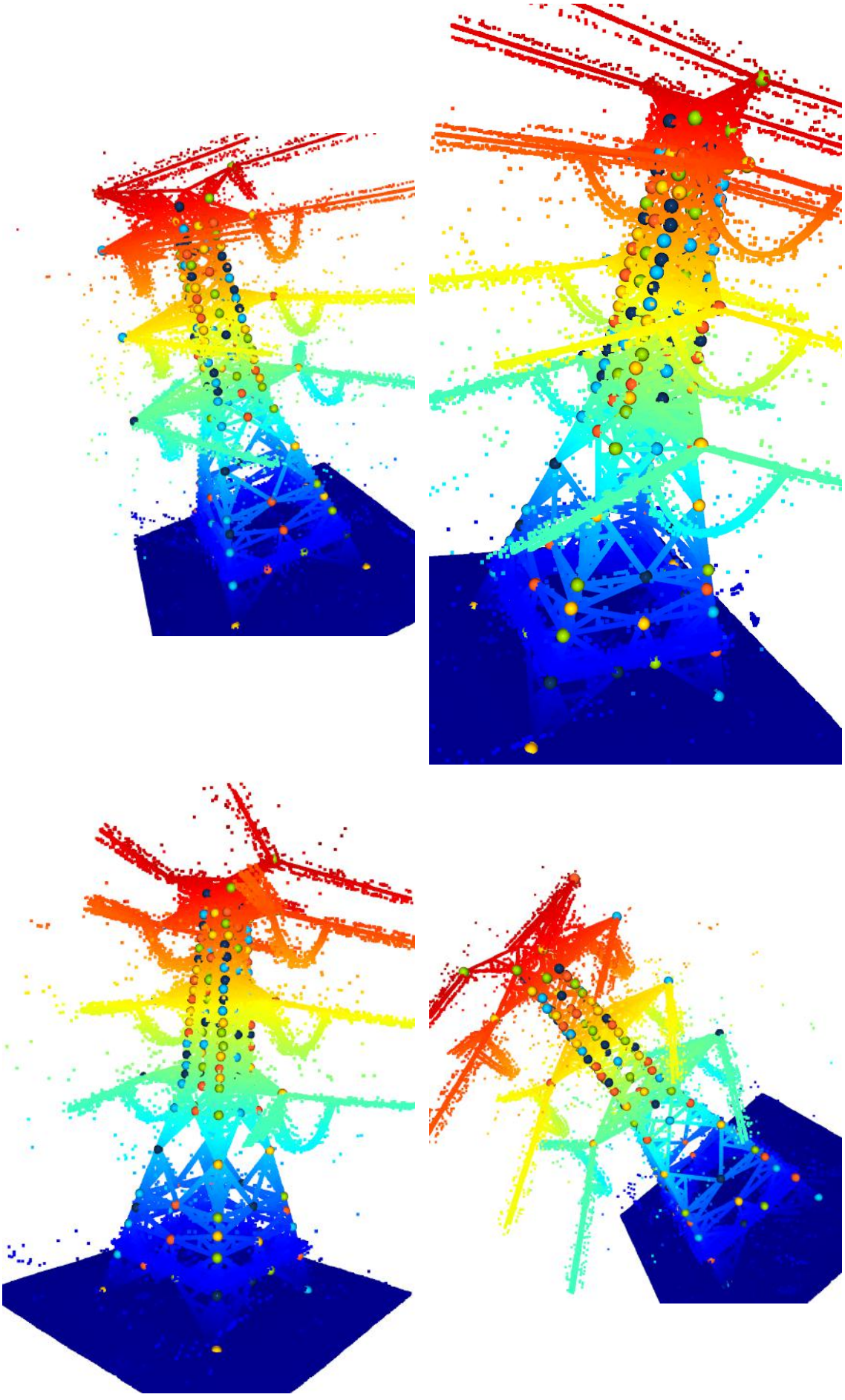


FIGURE 6.22: The 3D model point cloud of an electricity pylon with vertices marked.

We get the coordinates of the vertices from the 3D model. There are a total of 167 unique vertices on the pylon. The vertices selected are the main identifiable crossings on the pylon, we neglect the smaller crossings. For the neural network we only use the front side, which has 99 vertices. Figure 6.22 provides the 3D models and the vertices.

We generate a sequence of images by flying a drone over the location of the electricity pylon, and we are able to produce a video of the pylon. We use the DJI mini drone for capturing the data. The drone is fitted with a camera that can turn 360°. The camera operates at a frame rate of 30 frames per second.

For the sequence, we extract 300 images from the captured video. Each image in the sequence has a frame size of 1920×1080 pixels. The drone is operated in such a way that the electricity pylon is within the field of view of the camera. Figure 6.23 shows some of the images within the sequence.

FIGURE 6.23: Some of the images in the sequence.

To test the effectiveness of the tracking algorithm, we mark twelve vertices of the pylon and we use the vertices to estimate the ground truth pose. Figure 6.24 shows an image from the sequence in which the vertices are marked and the PnP algorithm is used to obtain the ground truth pose. We ensure the points are as far apart as possible for better measurement.

6.6.2 Method

The method we use is similar to the technique we introduced in the previous sections. In Section 6.4, geometric hashing was used for initialisation to improve

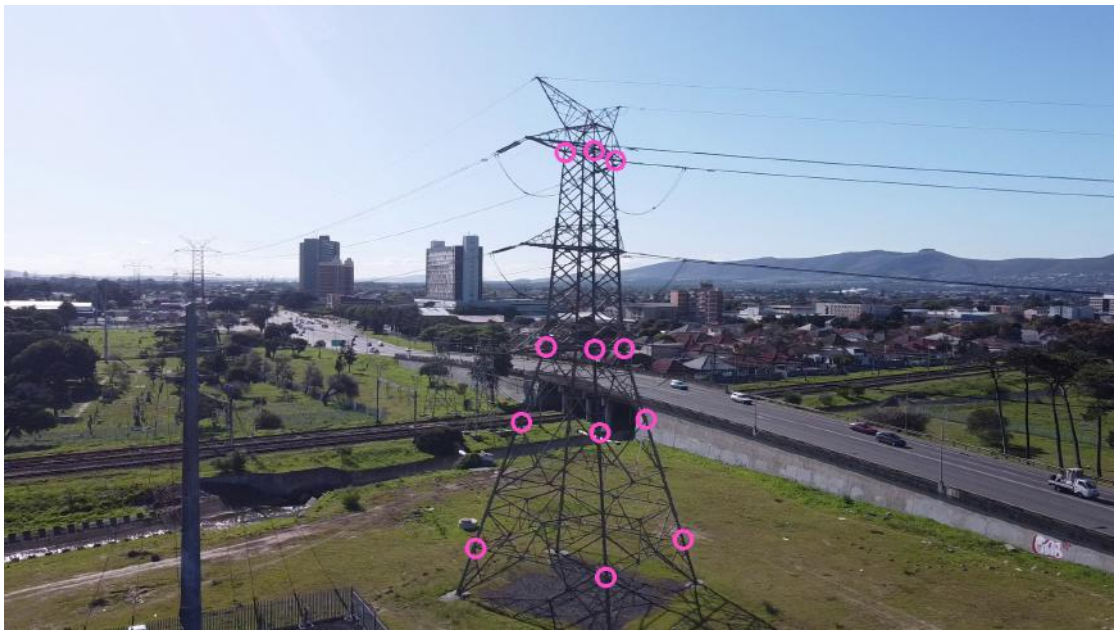


FIGURE 6.24: An image with the vertices marked to serve as ground truth.

the starting point of the tracking process as part of the framework. We presented the results of using geometric hashing in Section 5.8 for outdoor electricity pylons. The incremental tracking of a camera using the extended Kalman filter proved to be the most accurate tracker, so we use it for the tracking process.

We detect vertices using a convolutional neural network as described in Section 4.4.2. The detected vertices are the image keypoints used by the geometric hashing algorithm, and as measurements for the extended Kalman filter. The world model consists of the 3D coordinates of the pylon obtained from the reconstructed 3D model by the laser scanner. The features obtained through vertex detection and their model are stored in a hash table during the training for the geometric hashing. While at the recognition stage, the features of the image scene are obtained and used to get a match in the hash table with a voting technique. A match between the image scene and a world model means a correspondence has been established. This process serves as the initialisation of the tracker.

For the tracking of the pose estimate, we use the extended Kalman filter. The tracker has two stages namely the prediction and update stages. The a priori pose estimate and the a priori covariance are obtained at the prediction stage using the motion model introduced in Equation 6.11 of Section 6.4. The random walk serves as the motion model.

The measurement is input into the tracker at the update stage as the mean and the covariance are propagated. The measurement is the vertices from the output of the convolutional neural network. We use the a priori pose estimate to register the world coordinates of the vertices onto the 2D image scene. Subsequently, we determine the residual error component of the extended Kalman filter. The residual error component can only be accurately calculated if the registered points and the vertices can establish correspondence. Data association is how to establish correspondence between the registered points and the vertices.

Data association in Section 6.4 uses a RANSAC to solve the correspondence problem, while in Section 6.3 we use channel-to-channel assignment to solve the correspondence issue. With an outdoor electricity pylon, the vertices on the pylon especially towards the top are cluttered, and there are issues of mismatch between the world points and the image points. The goal is to ensure that points in close proximity are related to one another. So we assign an output of a channel of the network to a specific vertex on the pylon and we also ensure the channel and the vertex are within a certain distance. Therefore, we experiment with RANSAC and channel-to-channel methods.

Furthermore, we determine the Kalman gain with the inlier points from data association, and the Kalman gain is used to tune the residual error difference. The residual error difference is the observation model we use as introduced in Equation 6.12 of Section 6.4. Thus we obtain the a posterior pose estimate and covariance.

We refine the a posterior pose estimate to improve the tracking performance. A 3D-2D ICP is used as a pose refinement stage. This stage of pose refinement is similar to the stage of pose refinement we introduced in the previous experiments in this chapter.

6.6.3 Discussion and analysis

We present and discuss the results of validating the algorithms with real world pylon data and we demonstrate the performance of the algorithms. Previously, we collected data with the aid of a phone camera. However, in order to conduct this experiment we use a DJI mini drone to capture the data. The drone means that the data is captured from an overhead position with a clear field of view of

the electricity pylon. Also, the drone helps with getting accurate data because the vibration, motion stability, and wind tend to affect the trajectory of the movement of a drone.

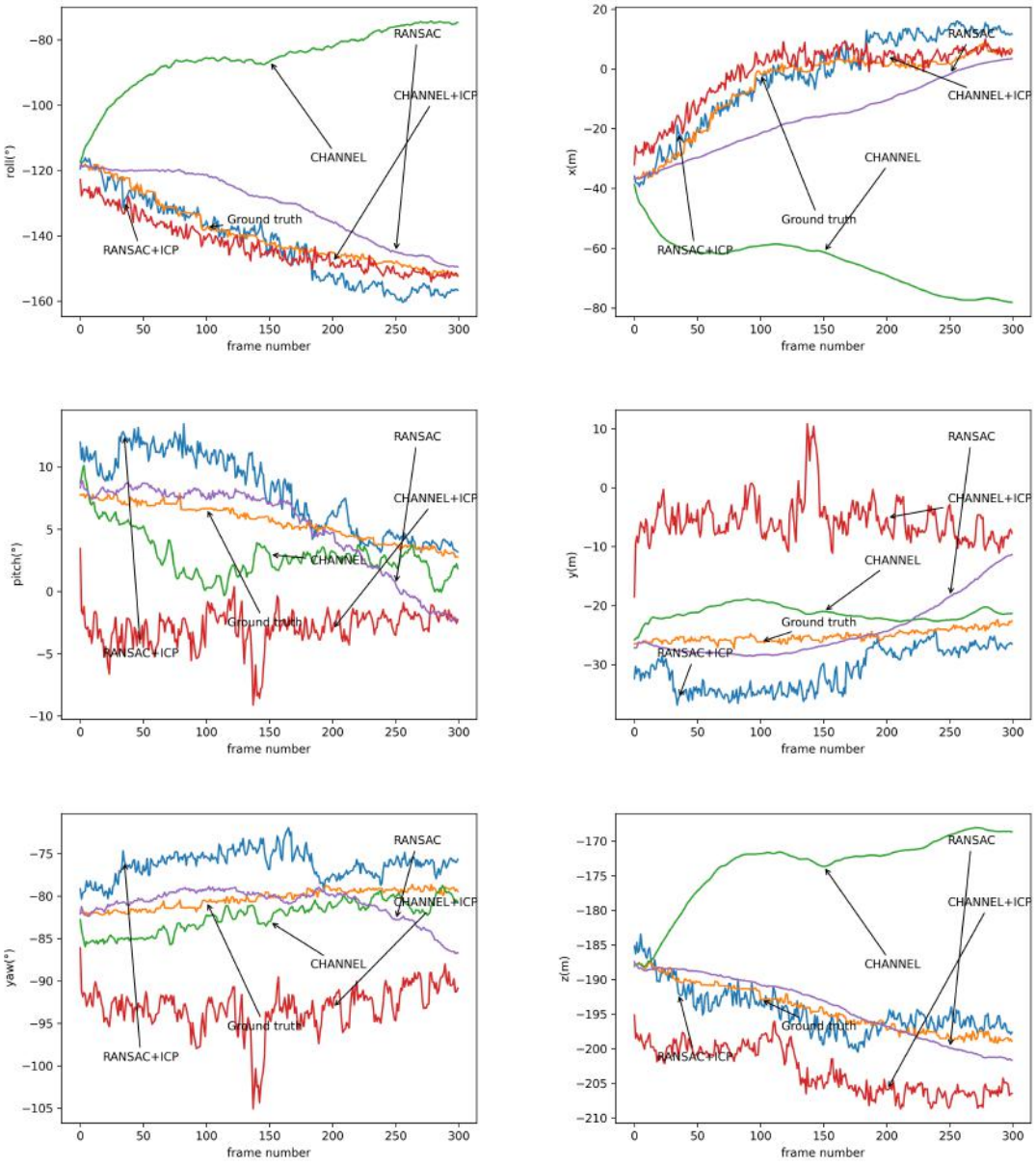


FIGURE 6.25: The position and orientation of the camera relative to the pylon during the tracking process. It represents the ground truth pose and the various estimated poses.

Figures 6.25 provides information on the trajectory of the camera as it moves. The result is for an extended Kalman filter using several data association techniques including RANSAC or the channel-to-channel method. We test both with and without pose refinement to check the effects on the pose estimate. The figures

demonstrate that when pose refinement is applied, the method have a tendency to enhance the results of the tracking process. Table 6.2 presents the average mean errors for the position and orientation of the camera relative to the pylon.

Method	x(m)	y(m)	z(m)	roll(°)	pitch(°)	yaw(°)
RANSAC without ICP	10.30	0.89	0.23	8.29	0.38	0.62
RANSAC with ICP	3.28	5.84	0.27	3.15	2.41	4.09
Channel-channel without ICP	59.47	3.78	20.67	53.42	2.63	2.06
Channel-channel with ICP	4.54	19.57	8.98	3.95	8.47	12.83

TABLE 6.2: Average mean camera errors.

We compare the error metric in 3D space using the APD metric and in the image plane using the reprojection error. Table 6.3 presents the error value in the 3D space and image plane with different data association and pose refinement techniques. We set the APD error metric in 3D as a way of measuring the accuracy of a successful detection, and the error value must fall within a certain limit to achieve the successful detection. We determine the threshold value by setting it to be 10 percent of the diameter or longest side of the object. All the methods were able to fall successfully within the allowed threshold value for the 300 frames in the sequence. Moreso, we see an average reprojection of less than 10 pixels for a frame size of 1920×1080 pixels for the sequence. Although, differences in the degree of accuracy as shown by the average error value using APD and reprojection errors are seen, the lower the error values, the better the performance of the algorithm. Even though the error of the RANSAC without the ICP is lower than the error of the RANSAC with the ICP in Table 6.3, when we compare with Table 6.2 we observe that the ICP spreads the error because without the ICP the result is skew to a certain position and orientation.

Method	APD(m)	Reprojection error(pixels)
RANSAC without ICP	0.19	1.61
RANSAC with ICP	0.47	2.32
Channel-channel without ICP	0.49	5.84
Channel-channel with ICP	0.42	1.86

TABLE 6.3: Average error value.

The speed of the tracking algorithms is found to be 1.76s, 1.73s, 1.55s and 1.62s for channel-to-channel with ICP, channel-to-channel without ICP, RANSAC without ICP and RANSAC with ICP respectively.

Figures 6.26 and 6.27 present samples of the images in the sequence with 3D coordinates of the vertices of the electricity registered onto the scene image. We see that the pose refinement tends to improve the registration as more vertices are registered onto the images.

For the sake of a reliable inspection, the drone will have to be flown at a slow inspection speed. During the inspection, the algorithm will provide localisation details to the robot, and inspection tasks such as fault analysis and missing components detection will be performed simultaneously.

6.7 Summary

In this chapter various algorithms for the pose estimation and tracking of the pylon were investigated. We explored the usage of the vertex detection and the extended Kalman filter for pose estimation. We showed the use of geometric hashing for single image pose estimation which can serve as the initialization for the algorithms.

We experimented with an extended Kalman filter and a vertex detection algorithm for tracking the camera relative to the pylon. The result shows the tracker stabilises after some frames but the 3D model could not fit properly. We made adjustments to the algorithm in experiment one to form the basis of experiment two.

The second experiment uses geometric hashing to obtain the single image pose estimation for initialization of the tracker. This provides a good starting time for the process. We adjusted the vertex detection to use RANSAC for outlier rejection. The result from experiment two shows successful initialization and an accurate tracking of the camera relative to the pylon. The successful implementation of experiment two can be used as a localisation component for the navigation of a robot.

The third experiment uses details from the heatmaps of the vertex detection algorithm and a gradient descent for tracking of the camera relative to the pylon.

.

FIGURE 6.26: Results of the 3D model of the pylon registering onto the image of the pylon with channel-to-channel as the means for data association .

FIGURE 6.27: Results of the 3D model of the pylon registering onto the image of the pylon with RANSAC as a means of data association.

We were able to achieve tracking of the camera without the need for RANSAC as outlier rejection. The experiment shows a successful tracking process has been achieved and can replace the extended Kalman filter in experiment two.

Finally in the fourth experiment we use real world pylon data to revalidate the tracking of the relative to the pylon. We use a laser scanner to build the 3D model, build a sequence from a video captured by a drone and test the performance of the algorithms with the data.

Chapter 7

Conclusion

In this thesis we present our work on pose estimation of a pylon. We provide the theory behind the work, undertake some experiments to establish the validity of our work, demonstrate the work with a small model of a pylon and revalidate the effectiveness of the algorithms using real world outdoor data.

This chapter concludes the thesis by summarising the work, and providing the significant contributions achieved.

The dissertation focused on designing a strategy for finding the pose estimate of a camera relative to a pylon.

We hypothesise that it is possible to find the localisation of weakly textured objects such as a pylon using computer vision techniques. In the problem definition in Chapter 1, we asked if we can find distinguishing features to enable us identify the pylon? We have answered this question by using a convolutional neural network and vertices as the distinguishing features.

We also asked if a search-based algorithm can be used to solve for the six degree-of-freedom pose estimate. We have been able to answer the question using a model-based recognition method called geometric hashing.

Finally, we asked if the movement of the camera can be tracked? We used an extended Kalman filter and a tracker using heatmap details to address this question.

In addressing the problem of pose estimation, we needed to find unique features on the pylon. Traditional hand-crafted feature methods such as SIFT, SURF, ORB and Harris corner detector tend to fail on the pylon. Therefore, we borrowed a

technique from human pose estimation with labelling of parts for the feature detection. We presented the ideas behind the technique in Chapter 4. The intersection points of the bars on the pylon are salient points and we call them the vertices. In Chapter 4, we discussed a convolutional neural network for finding a vertex.

Geometric hashing is a model-based technique that was used to match the 3D model pylon to the image scene. We used vertices obtained from Chapter 4 as the features for the algorithm. Chapter 5 presents the concepts of the algorithm. We implemented it at the initialization stage of the localisation of the tracking process.

We tracked the movement of the camera relative to the pylon and used the extended Kalman filter to achieve this task. In Chapter 3, we presented the theory behind the Kalman filter and discussed the motion model as a way of representing the movement of the camera.

Chapter 6 presents the central results of the research. We combined the ideas we presented in the other chapters to fulfill our need for a six degree-of-freedom pose estimation procedure. We successfully implemented three experiments for the pose estimation. Experiment one used vertex detection and an extended Kalman filter for tracking of the camera relative to the pylon. In experiment two, we incorporated geometric hashing for single image pose estimation used as initialization of the tracking process. The tracking was achieved using an extended Kalman filter and a vertex detection algorithm was used to find features for the geometric hashing and the extended Kalman filter. Experiment three was implemented using the heatmap details of the vertex detection algorithm for tracking of the camera relative to the pylon. Finally, in experiment four we implemented the extended Kalman filter on real world outdoor pylon data. The experiment ensures we revalidate the algorithms built with laboratory data with real world data. Also, it showed that the algorithms can be applied to different sets of pylons as long as there are crossings on the pylons.

Bibliography

[1] MultiMedia LLC. MS Windows NT kernel description, 1999. URL https://i.stack.imgur.com/yylFT.png.

[2] Andreas Wendel. *Scalable Visual Navigation for Micro Aerial Vehicles using Geometric Prior Knowledge*. PhD thesis, PhD thesis, Graz University of Technology, 2013.

[3] Manuel Hofer, Andreas Wendel, and Horst Bischof. Incremental line-based 3d reconstruction using geometric constraints. In *BMVC*, 2013.

[4] Bhavani Morarjee. Using multiple view geometry for transmission tower reconstruction. Master's thesis, University of Cape Town, 2016.

[5] MultiMedia LLC. MS Windows NT kernel description, 1999. URL http://aerossurance.com/helicopters/wirestrike-powerline-inspection/.

[6] Nicolas Pouliot and Serge Montambault. Geometric design of the LineScout, a teleoperated robot for power line inspection and maintenance. In *2008 IEEE International Conference on Robotics and Automation*, pages 3970–3977. IEEE, 2008.

[7] Paulo Debenest, Michele Guarnieri, Kensuke Takita, Edwardo F Fukushima, Shigeo Hirose, Kiyoshi Tamura, Akihiro Kimura, Hiroshi Kubokawa, Narumi Iwama, and Fuminori Shiga. Expliner-robot for inspection of transmission lines. In *2008 IEEE International Conference on Robotics and Automation*, pages 3978–3984. IEEE, 2008.

[8] Timothy Rowell and Ed Boje. Obstacle avoidance for a power line inspection robot. In *2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI)*, pages 114–119. IEEE, 2012.

[9] Xinyan Qin, Bo Jia, Jin Lei, Jie Zhang, Huidong Li, Bo Li, and Zhaojun Li. A novel flying–walking power line inspection robot and stability analysis hanging on the line under wind loads. *Mechanical Sciences*, 13(1):257–273, 2022.

[10] Aydin Tarik Zengin, Gokhan Erdemir, Tahir Cetin Akinci, Fahri Anil Selcuk, Mustafa Nizamettin Erduran, and S Serhat Seker. Rosetlinebot: One-wheel-drive low-cost power line inspection robot design and control. *arXiv preprint arXiv:1911.08173*, 2019.

[11] Tianzhe Zhang and Jun Dai. Electric power intelligent inspection robot: a review. In *Journal of Physics: Conference Series*, volume 1750, page 012023. IOP Publishing, 2021.

[12] Ahmad Bala Alhassan, Xiaodong Zhang, Haiming Shen, and Haibo Xu. Power transmission line inspection robots: A review, trends and challenges for future research. *International Journal of Electrical Power & Energy Systems*, 118: 105862, 2020.

[13] Fredrik Heintz, Piotr Rudol, and Patrick Doherty. From images to traffic behavior- A UAV tracking and monitoring application. In *2007 10th International Conference on Information Fusion*, pages 1–8. IEEE, 2007.

[14] Eduard Semsch, Michal Jakob, Dušan Pavlicek, and Michal Pechoucek. Autonomous UAV surveillance in complex urban environments. In *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 82–85. IEEE, 2009.

[15] Heikki Saari, Ismo Pellikka, Liisa Pesonen, Sakari Tuominen, Jan Heikkilä, Christer Holmlund, Jussi Mäkynen, Kai Ojala, and Tapani Antila. Unmanned Aerial Vehicle (UAV) operated spectral camera system for forest and agriculture applications. In *Remote Sensing for Agriculture, Ecosystems, and Hydrology XIII*, volume 8174, page 81740H. International Society for Optics and Photonics, 2011.

[16] Teodor Tomic, Korbinian Schmid, Philipp Lutz, Andreas Domel, Michael Kassecker, Elmar Mair, Iris Lynne Grixa, Felix Ruess, Michael Suppa, and Darius Burschka. Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. *IEEE robotics & automation magazine*, 19(3):46–56, 2012.

[17] Christian Eschmann, Chen-Ming Kuo, Chung-Hsin Kuo, and Christian Boller. Unmanned aircraft systems for remote building inspection and monitoring. In *Proceedings of the 6th European Workshop on Structural Health Monitoring, Dresden, Germany*, volume 36, page 13, 2012.

[18] Oualid Araar and Nabil Aouf. Visual servoing of a quadrotor UAV for autonomous power lines inspection. In *22nd Mediterranean Conference on Control and Automation*, pages 1418–1424. IEEE, 2014.

[19] Ian Golightly and Dewi Jones. Visual control of an unmanned aerial vehicle for power line inspection. In *ICAR'05. Proceedings., 12th International Conference on Advanced Robotics, 2005.*, pages 288–295. IEEE, 2005.

[20] Luis F Luque-Vega, Bernardino Castillo-Toledo, Alexander Loukianov, and Luis Enrique Gonzalez-Jimenez. Power line inspection via an unmanned aerial system based on the quadrotor helicopter. In *MELECON 2014-2014 17th IEEE Mediterranean electrotechnical conference*, pages 393–397. IEEE, 2014.

[21] Oscar Bowen Schofield, Kasper Høj Lorenzen, and Emad Ebeid. Cloud to cable: A drone framework for autonomous power line inspection. In *2020 23rd Euromicro Conference on Digital System Design (DSD)*, pages 503–509. IEEE, 2020.

[22] Serge Montambault, Julien Beaudry, Kristopher Toussaint, and Nicolas Pouliot. On the application of VTOL UAVs to the inspection of power utility assets. In *2010 1st International Conference on Applied Robotics for the Power Industry*, pages 1–7. IEEE, 2010.

[23] Robert Jenssen, Davide Roverso, et al. Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning. *International Journal of Electrical Power & Energy Systems*, 99:107–120, 2018.

[24] Xinyu Liu, Xiren Miao, Hao Jiang, and Jing Chen. Review of data analysis in vision inspection of power lines with an in-depth discussion of deep learning technology. *arXiv preprint arXiv:2003.09802*, 2020.

[25] Ian Golightly and Dewi Jones. Corner detection and matching for visual tracking during power line inspection. *Image and Vision Computing*, 21(9): 827–840, 2003.

[26] Wengang Cheng and Zhengzheng Song. Power pole detection based on graph cut. In *2008 Congress on Image and Signal Processing*, volume 3, pages 720–724. IEEE, 2008.

[27] Oualid Araar, Nabil Aouf, and Jose Luis Vallejo Dietz. Power pylon detection and monocular depth estimation from inspection UAVs. *Industrial Robot: An International Journal*, 42(3):200–213, 2015.

[28] Carlos Sampedro, Carol Martinez, Aneesh Chauhan, and Pascual Campoy. A supervised approach to electric tower detection and classification for power line inspection. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 1970–1977. IEEE, 2014.

[29] Rabab Abdelfattah, Xiaofeng Wang, and Song Wang. Ttpla: An aerial-image dataset for detection and segmentation of transmission towers and power lines. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[30] Xiaolong Hui, Jiang Bian, Xiaoguang Zhao, and Min Tan. Vision-based autonomous navigation approach for unmanned aerial vehicle transmission-line inspection. *International Journal of Advanced Robotic Systems*, 15(1): 1729881417752821, 2018.

[31] Jiang Bian, Xiaolong Hui, Xiaoguang Zhao, and Min Tan. A monocular vision–based perception approach for unmanned aerial vehicle close proximity transmission tower inspection. *International Journal of Advanced Robotic Systems*, 16(1):1729881418820227, 2019.

[32] Jingjing Zhang, Liang Liu, Binhai Wang, Xiguang Chen, Qian Wang, and Tianru Zheng. High speed automatic power line detection and tracking for a UAV-based inspection. In *2012 International Conference on Industrial Control and Electronics Engineering*, pages 266–269. IEEE, 2012.

[33] Xian Wang and Youmin Zhang. Insulator identification from aerial images using support vector machine with background suppression. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 892–897. IEEE, 2016.

[34] Markus Oberweger, Andreas Wendel, and Horst Bischof. Visual recognition and fault detection for power line insulators. In *19th computer vision winter workshop*, pages 1–8, 2014.

[35] Usiholo Iruansi, Jules R Tapamo, and Innocent E Davidson. An active contour approach to water droplets segmentation from insulators. In *2016 IEEE International Conference on Industrial Technology (ICIT)*, pages 737–741. IEEE, 2016.

[36] Xiren Miao, Xinyu Liu, Jing Chen, Shengbin Zhuang, Jianwei Fan, and Hao Jiang. Insulator detection in aerial images for transmission line inspection using single shot multibox detector. *IEEE Access*, 7:9945–9956, 2019.

[37] Haipeng Chen, Zhentao He, Bowen Shi, and Tie Zhong. Research on recognition method of electrical components based on yolo v3. *IEEE Access*, 7: 157818–157829, 2019.

[38] Ehab Ur Rahman, Yihong Zhang, Sohail Ahmad, Hafiz Ishfaq Ahmad, and Sayed Jobaer. Autonomous vision-based primary distribution systems porcelain insulators inspection using uavs. *Sensors*, 21(3):974, 2021.

[39] Bruno José Souza, Stefano Frizzo Stefenon, Gurmail Singh, and Roberto Zanetti Freire. Hybrid-yolo for classification of insulators defects in transmission lines based on uav. *International Journal of Electrical Power & Energy Systems*, 148:108982, 2023.

[40] Zhenbing Zhao, Zhen Zhen, Lei Zhang, Yincheng Qi, Yinghui Kong, and Ke Zhang. Insulator detection method in inspection image based on improved faster r-cnn. *Energies*, 12(7):1204, 2019.

[41] Mohsen Hejrati and Deva Ramanan. Analyzing 3d objects in cluttered images. In *Advances in Neural Information Processing Systems*, pages 593–601, 2012.

[42] Albert Haque, Boya Peng, Zelun Luo, Alexandre Alahi, Serena Yeung, and Li Fei-Fei. Towards viewpoint invariant 3d human pose estimation. In *European Conference on Computer Vision*, pages 160–177. Springer, 2016.

[43] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[44] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, volume 11, page 2. Citeseer, 2011.

[45] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[46] Silvio Savarese and Li Fei-Fei. 3D generic object categorization, localization and pose estimation. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[47] Amir Ghodrati, Marco Pedersoli, and Tinne Tuytelaars. Is 2d information enough for viewpoint estimation? *Proceedings BMVC 2014*, pages 1–12, 2014.

[48] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *2011 international conference on computer vision*, pages 858–865. IEEE, 2011.

[49] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3D object recognition. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 998–1005. Ieee, 2010.

[50] Bertram Drost and Slobodan Ilic. 3D object detection and localization using multimodal point pair features. In *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pages 9–16. IEEE, 2012.

[51] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D object pose estimation using 3D object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014.

[52] Alykhan Tejani, Rigas Kouskouridas, Andreas Doumanoglou, Danhang Tang, and Tae-Kyun Kim. Latent-class hough forests for 6 DoF object pose estimation. *IEEE transactions on pattern analysis and machine intelligence*, 40 (1):119–132, 2017.

[53] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3343–3352, 2019.

[54] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[55] Alberto Crivellaro, Mahdi Rad, Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. Robust 3D object tracking from monocular images using stable parts. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1465–1479, 2017.

[56] Hao Su, Charles R. Qi, Yangyan Li, and Leonidas J. Guibas. Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015.

[57] Cosimo Rubino, Marco Crocco, and Alessio Del Bue. 3d object localisation from multi-view image detections. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1281–1294, 2017.

[58] Daniel Glasner, Meirav Galun, Sharon Alpert, Ronen Basri, and Gregory Shakhnarovich. Aware object detection and pose estimation. In *2011 International Conference on Computer Vision*, pages 1275–1282. IEEE, 2011.

[59] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[60] Andrew H. Jazwinski. Adaptive filtering. *Automatica*, 5(4):475–485, 1969.

[61] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*, volume 1. MIT press Cambridge, 2000.

[62] Lawrence D. Stone, Roy L. Streit, Thomas L. Corwin, and Kristine L. Bell. *Bayesian multiple target tracking*. Artech House, 2013.

[63] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[64] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision*, pages 34–50. Springer, 2016.

[65] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017.

[66] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014.

[67] Alexander Mathis, Pranav Mamidanna, Kevin M Cury, Taiga Abe, Venkatesh N Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature neuroscience*, 21(9):1281–1289, 2018.

[68] Olga Barinova, Victor Lempitsky, and Pushmeet Kholi. On detection of multiple object instances using hough transforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1773–1784, 2012.

[69] Carl D. Meyer. *Matrix analysis and applied linear algebra*, volume 71. Siam, 2000.

[70] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[71] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[72] Yehezkel Lamdan and Haim J Wolfson. Geometric hashing: A general and efficient model-based recognition scheme, 1988.

[73] Yehezkel Lamdan, Jacob T Schwartz, and Haim J Wolfson. Object recognition by affine invariant matching. In *Proceedings CVPR'88: The Computer Society Conference on Computer Vision and Pattern Recognition*, pages 335–344. IEEE, 1988.

[74] Frank C.D. Tsai. Geometric hashing with line features. *Pattern Recognition*, 27(3):377–389, 1994.

[75] D.M. Gavrila and Frans C.A. Groen. 3D object recognition from 2D images using geometric hashing. *Pattern recognition letters*, 13(4):263–278, 1992.

[76] Xavier Pennec and Nicholas Ayache. A geometric algorithm to find small but highly similar 3D substructures in proteins. *Bioinformatics (Oxford, England)*, 14(6):516–522, 1998.

[77] Isidore Rigoutsos and Robert Hummel. A Bayesian approach to model matching with geometric hashing. *Computer vision and image understanding*, 62 (1):11–26, 1995.

[78] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[79] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[80] David A. Forsyth and Jean Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[81] Heinz Rüther, Christoph Held, Roshan Bhurtha, Ralph Schroeder, and Stephen Wessels. From point cloud to textured model, the zamani laser scanning pipeline in heritage documentation. *South African Journal of Geomatics*, 1(1):44–59, 2012.