12-19-2022 11:00 AM

Towards Parking Lot Occupancy Assessment Using Aerial Imagery and Computer Vision

John Jewell, *The University of Western Ontario*

Supervisor: Yalda Mohsenzadeh, *The University of Western Ontario*
: Michael Bauer, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science

Abstract

Advances in Computer Vision and Aerial Imaging have enabled countless downstream applications. To this end, aerial imagery could be leveraged to analyze the usage of parking lots. This would enable retail centres to allocate space better and eliminate the parking oversupply problem. With this use case in mind, the proposed research introduces a novel framework for parking lot occupancy assessments. The framework consists of a pipeline of components that map a sequence of image sets spanning a parking lot at different time intervals to a parking lot turnover heatmap that encodes the frequency each parking stall was used. The pipeline of components includes Image Stitching, Vehicle Detection and Heatmap Generation. The focus of this work is Image Stitching and Vehicle Detection, while Heatmap Generation is left for future work. Beyond proposing a novel framework for parking lot occupancy assessments, several contributions are made to the Computer Vision field. In particular, a novel method for initializing the pose of images based on the metadata from the acquisition system is introduced. Additionally, a novel comparative study of object detection models applied to the vehicle detection task is presented. Extensive experiments are used to validate the proposed contributions on both public and private datasets.

Keywords: Object Detection, Image Stitching

Summary for Lay Audience

Vision is fundamental to how humans perceive and act in the world. Thus, in the pursuit of creating intelligent systems, it is intuitive to endow computers with a similar sense of perception. This is the focus of Computer Vision - a scientific field seeking to develop systems that analyze images and videos. In this thesis, Computer Vision is leveraged to work toward a system that performs parking lot occupancy assessments. More specifically, the system uses aerial images of parking lots taken across time intervals to generate a heatmap that encodes the number of times each parking stall is used. The system consists of a pipeline of three components: Image Stitching, Vehicle Detection, and Heatmap Generation. First, Image Stitching is used to map a set of overlapping images to a consistent mosaic. Subsequently, vehicle instances in the mosaic are localized during Vehicle Detection. Lastly, Heatmap Generation uses the mosaics along with the detected vehicle instances to generate a heatmap. The focus of this work is on Image Stitching and Vehicle Detection. However, a preliminary discussion of Heatmap Generation is included to provide some initial direction.

Acknowlegements

I want to start by expressing gratitude to my supervisor Yalda Mohsenzadeh. I have had the pleasure of working with her since my undergraduate studies. Throughout this time, she has helped me grow as both a researcher and a person. I will always be indebted to her for the countless opportunities she has provided me with. Furthermore, I would like to thank Steven Beauchemin and Michael Bauer for co-supervising me. Their thoughtful feedback was constructive in writing my thesis. Many thanks to my co-author and friend Vahid Reza Khazaie as well.

I would also like to thank my loved ones - without them this work would not be possible. In particular, thank my parents for everything they have done for me. I would be remiss if I did not also acknowledge the tremendous support of my grandparents. Lastly, thanks to my partner Lauren for being by my side every step of the way.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Literature Review

1.1 Introduction

With the advent of unmanned aerial vehicles (UAV) such as dynamic remotely operated navigation equipment (drones), high-resolution aerial image data has become widely available [11]. In parallel, the Computer Vision field has matured to provide a wide range of image-based algorithms for tasks like scene reconstruction and object detection. This presents an opportunity to apply recent advances in Computer Vision to aerial imagery to develop novel systems. For example, Computer Vision algorithms can be used to map a set of images spanning a scene to a precise 2D model wherein various objects of interest can be detected. Thus, numerous applications in urban planning, traffic planning, surveillance and agriculture are enabled.

Specifically, within the realm of urban planning, large open-concept retail centers could benefit from leveraging aerial image data to analyze how their parking facilities are being used. This would allow retail centers to optimize the allocation of parking facilities, thereby increasing efficiency and reducing the parking oversupply problem. With this use case in mind, the proposed research offers a significant step towards an end-to-end framework for generating parking lot turnover heatmaps. These heatmaps specify the amount cars parked in each parking stall over the entire day. The proposed framework is a pipeline that maps a sequence of image sets spanning a parking lot at different time intervals to a parking lot turnover heatmap. The resulting output gives us the distribution of parking lot utilization. This is an order-of-magnitude improvement over occupancy assessment systems that may only detect aggregate vehicle counts.

As depicted in Figure 1.1, the framework consists of three components: Image Stitching, Vehicle Detection and Heatmap Generation. This thesis will focus on the Image Stitching and Vehicle Detection component while the Heatmap Generation is left for future work. To our knowledge, this is the first work to work towards an end-to-end assessment of parking utilization. In doing so, several novel contributions are made to Image Stitching and Vehicle Detection. The following chapter seeks to familiarize the reader with the foundations of the proposed research. This includes an overview of the dataset, a discussion of related work and a look ahead to future chapters.

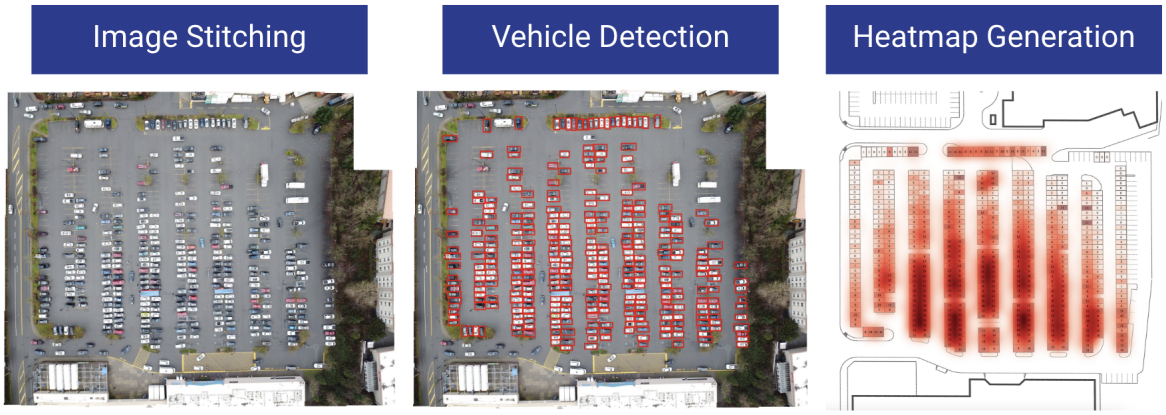

Figure 1.1: An example of the three main steps of the project: Image Stitching, Vehicle Detection and Heatmap Generation.

1.2 Dataset

The data for the parking lot utilization assessment framework is sourced from a large private dataset of high-resolution drone images and associated metadata. Specifically, the dataset contains sequences of image sets that span a particular parking lot across several time intervals throughout the day. Each parking lot is surveyed 11 times at half-hour intervals from 12:00 to 17:00. A figure illustrating a set of images at the first and last time interval for the same parking lot is available in Figure 1.2. The image set at each interval spans the parking lot and its size can vary across time intervals. In total, there are 26 parking lots that were surveyed across Canada using various DJI drones. As such, there is substantial variation in the characteristics of the parking lots.

1.3 Formalizing Image and Metadata

Each image i from parking lot p at time t can be represented as a tensor $I_{ipt} \in \mathbb{R}^{HxWx3}$ where H is the height of the image, W is the width of the image and 3 is the number of channels in a color image. I_{ipt} also has associated metadata tuple M_{ipt}:

$$M_{ipt} =< \psi, \theta, \phi, t_x, t_y, t_z, f_x, f_y, c_x, c_y > \tag{1.1}$$

For brevity of notation, the index ipt is omitted from components of M. ψ, θ and ϕ are the yaw, pitch and roll for the drone, respectively. The angles are measured in degrees. t_x and t_y are utm coordinates representing the easting and northing, respectively. t_z is the relative altitude of the drone. t_x, t_y and t_z are measured in meters. f_x and f_y are the focal lengths of the camera, expressed in pixel units. (c_x, c_y) is the coordinates of the principal point in pixel units, which is assumed to be at the center of the image. Thus, M_{ipt} contains several parameters that help recover the pose of the drone in the world coordinate system when capturing I_{ipt}. This information can be leveraged to help build a precise 2D model of the parking lot.

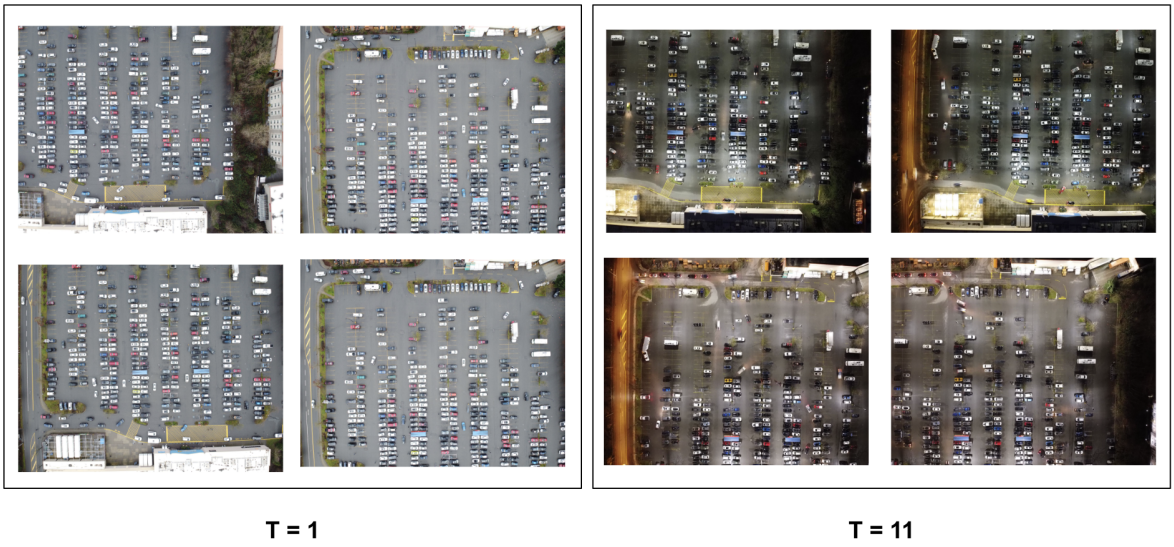

Figure 1.2: A set of images at the first and last time interval for the same parking lot.

1.4 Related Work

The related work section provides the reader with an overview of research related to the current work. The proposed end-to-end parking utilization assessment framework is novel and no existing research addresses this specific problem—however, sub-components of the framework fall inside well-established fields of research in computer vision. In particular, Image Stitching and Vehicle Detection are two popular Computer Vision tasks that are relevant to the proposed framework. Image Stitching provides the foundation for generating mosaics from sets of images. It also provides the capability to register multiple mosaics across time. Vehicle Detection is also a core component of the proposed framework. It is used to detect vehicle instances in aerial images. The following sections highlight related work for Image Stitching and Vehicle Detection, respectively.

1.4.1 Image Stitching

Image stitching involves aligning images of the same scene captured from different locations and orientations [27]. At its core, image stitching involves two high-level steps: homography estimation and global alignment. In the homography estimation step, the transformation that maps one image to the coordinate system of another is estimated. In the subsequent global alignment step, the estimated homographies from the previous step are refined jointly.

The first step of the homography estimation process involves detecting and describing the key points for each image with a vector. The goal is to generate keypoints with descriptors that only match the corresponding point in other images. Keypoint detectors select the points in the image representing corners as keypoints. The rationale for selecting corners is that they correspond to distinct parts of objects. Thus, corners are more easily identifiable across images. Subsequently, a descriptor could be generated for the keypoint using a neighbourhood of pixels. However, this representation is not invariant transformations in the image plane. Scale

Invariant Feature Transform (SIFT) extends this simple paradigm with a more sophisticated description algorithm that generates representations that are invariant to scaling and rotations [20]. Additionally, Speeded-Up Robust Features (SURF) [2] and Oriented FAST and Rotated BRIEF (ORB) [22] features are alternatives to SIFT that is more efficient to compute. Since these features are faster to compute, they are favored in low-latency applications. However, when the quality of keypoints is of utmost concern, SIFT features are the better choice.

Following keypoint detection and description, correspondences across images are determined via a keypoint matching algorithm. In the simplest case, a brute-force approach involves computing a distance between each pair of keypoints across images. A more efficient alternative involves approximate k-d tree matching [1]. Once keypoints have been matched, the transformation can be determined by solving the system of equations specified by the correspondences. Due to the presence of false correspondences, the system of equation is inconsistent and no unique solution exists. Thus, methods like Random Sample Consensus (RANSAC) are employed to yield an approximate global solution. RANSAC is an iterative method for solving an overdetermined system of equations that contains outliers. In each round, a random subset of correspondences are chosen to fit a model of the data. The model is validated across all datapoints and the number of outliers is recorded each round. The procedure is repeated a specified number of times and the model with the lowest number of outliers is returned. In this way, RANSAC provides an approximate global solution for the overdetermined system of equations. The aforementioned procedure is repeated for each pair of images in the set to yield the pairwise relative transformations. A threshold on the number of feature matches [27] is used to filter out pairs of images that do not overlap. This ensures that only pairs of images that are likely to be overlapping will have transformations computed. In the subsequent global alignment step, the estimated homographies from the previous step are refined globally, often using a least squares solution like bundle adjustment [29].

1.4.2 Vehicle Detection

Vehicle detection involves localizing vehicle instances in aerial images. Object detection has been a highly active area of research in recent years with the advent of deep learning and its applications in computer vision. The research focuses on a class of deep learning models called convolutional neural networks (CNN) [14]. CNN combines the traditionally disjointed process of feature extraction and classification/regression, allowing the model to be optimized end-to-end for the prediction task. Current state-of-the-art object detection approaches leveraging deep learning have been able to achieve near-human level performance over a multitude of benchmark datasets [13].

Traditionally, object detection has been approached using sliding window template matching. Template matching involves collecting a set of templates that are small images that depict the object that we are looking to detect. The image is divided into overlapping grid cells the same size as templates. The template is then applied to each grid cell and the correlation is computed. Locations of the image with high correlation are more likely to contain the object specified by the template. A threshold is applied to the correlation map to determine actual detection. The grid cells give us the bounding box of an object and the template gives us the class. The template matching and sliding window approach to object detection has several limitations. The search space of object instances across poses within a class is extremely large.

Templates only characterize a single instance of an object with a specific pose. Thus, a finite number of templates cannot adequately search the space of object instances.

Subsequently, the Viola-Jones [31] approach to object detection was proposed. This method addressed some limitations of the sliding window template matching approach by avoiding explicitly defining object templates. The Viola-Jones detector involves two steps: feature extraction and classification. In the feature extraction step, a window is moved over the input images and features are computed at every location. An integral image is used for efficient feature evaluation. This involves determining which areas of the image have a high cross-correlation with handcrafted feature templates. Each feature template is considered weak learning because it only captures certain characteristics of an object. By using multiple feature templates, we can define a linear model as a strong learner that maps the predictions of weak learners to a final joint prediction. The Viola-Jones detector found moderate success in downstream tasks such as face detection. However, the accuracy of the Viola Jones Detector is still strongly dependent on the pose of objects. In the case of face detection, only people directly facing the camera have their faces reliably detected.

The feature extraction and classification object detection paradigm were extended in subsequent works. Methods to enhance both the feature extraction and classification steps have been proposed. In particular, Histogram of Oriented Gradient (HOG) features [4] enhanced object detection by providing better feature detection and description. HoG features are computed by defining a histogram over the quantized orientation of image gradients for a particular location. A Support Vector Machine (SVM) is then trained to classify keypoints that are extracted from a specific object class. Thus, an SVM is trained for each object class. The positive samples are extracted from images containing the specified object class and negative samples are extracted from images not containing a specified object class. This approach to object detection has been shown to be more robust then the Viola Jones detector.

The seminal CNN-based object detection approach is OverFeat [24]. OverFeat uses feature maps extracted from the intermediate layers of a pretrained backbone to classify and localize objects in images. More specifically, the feature maps are input into the network's classification and localization branches to yield the objects' detections. OverFeat is fed the input image at multiple resolutions to account for objects at multiple resolutions. Lower-resolution inputs are geared toward detecting large objects and high-resolution inputs are geared toward detecting small objects. The spatial resolution of the feature maps is proportional to the size of the input image. Thus, the classification and detection branches of the architecture must handle feature maps with a variable spatial resolution, which is impossible in the case of fully connected layers. OverFeat uses 1x1 convolutional layers in place of fully connected layers to overcome this constraint. The R-CNN [7] architecture for object detection was subsequently proposed and surpassed the performance of OverFeat. R-CNN uses an auxiliary region proposal algorithm such as Selective Search [30] to generate candidate regions of various sizes. The candidate regions are resized to a consistent resolution and fed to a CNN for classification. Since the candidate generation algorithm outputs the coordinates of the objects, no localization branch of the architecture is necessary. A downside of the R-CNN architecture is that it requires a forward pass per regional proposal which is computationally intensive. Fast R-CNN [6] improves the efficiency of the R-CNN architecture by mapping the regional proposal directly to the feature maps produced by CNN. Region of Interest (ROI) pooling was introduced to pool features across region proposals. The feature maps are then fed to classification and localiza-

tion branches to generate the labels and coordinates of objects, respectively. Most recently, Faster R-CNN [21] was introduced as a more efficient and accurate alternative to Fast R-CNN. The fundamental contribution of this approach was the introduction of a CNN-based region proposal network in place of the Selective Search algorithm.

Although object detection is a popular area of research, the application of object detection to detecting vehicles in aerial images has been explored less [12]. Out-of-the-box object detection methods tend to generalize poorly because vehicle instances in aerial imagery are smaller than the objects they are optimized to detect. Recent works have addressed this issue by adapting state-of-the-art object detection architectures to deal with small object instances using higher resolution feature maps, smaller anchor boxes and other minor adaptations [26]. Other recent approaches to detecting vehicles in aerial imagery incorporate inductive bias related to the specific object detection task into the model [15].

1.5 Overview

The chapters in the thesis are as follows: Background, Image Stitching, Vehicle Detection, and Conclusion. The Background chapter offers an overview of the technical preliminaries that are helpful for future chapters. The Image Stitching chapter outlines the algorithm to map a set of images to a mosaic. First, ground truth transformations are estimated between pairs of images using a feature alignment-based approach. In the subsequent global alignment step, the pose of the images in the mosaic is jointly optimized to approximate the ground truth relative transformations from the first step. In the Vehicle Detection chapter, the object detection algorithm to detect vehicle instances in aerial images is outlined. A preliminary discussion of Heatmap Generation and its relation to existing components of the framework is also included. Lastly, the Conclusion chapter offers a summary of the work as well as Limitations and Future Work.

Chapter 2

Background

2.1 Coordinate Systems

Coordinate systems are used to represent the location of points on a manifold. The points in the space are specified using sets of coordinates that index into reference lines or curves [27]. Many different coordinate systems represent the same geometric structures, and we can often define generic mappings between these different representations. The most common coordinate systems are Cartesian coordinates and Homogeneous coordinates. Given a coordinate system, we can apply transformations that map points in the space to their updated coordinates. In Computer Vision, coordinate systems are used extensively to define a 2D (ex. image) or 3D (ex. real world) space in which points reside. Transformations allow us to augment the pose of a structure with respect to a space. For example, an image can have transformations applied to it, such as rotations, translation and scaling [3]. A depiction of the various classes of transformations that can be applied to images can be found in Figure 2.1.

2.1.1 Cartesian Coordinate System

The Cartesian Coordinate system represents a n-dimensional euclidean space using n perpendicular basis vectors [3]. Thus, points in the space can be addressed using an n-dimension tuple. We can represent a generic 3D transformation consisting of rotation followed by scaling and translation as follows:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} a_x & 0 & 0 \\ 0 & a_y & 0 \\ 0 & 0 & a_z \end{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.1}$$

$$P' = ARP + t \tag{2.2}$$

where P is the coordinate of the original point and P' is the corresponding point after the transformation has been applied. A is the scaling matrix, R is the rotation matrix and t is the translation vector. The rotation matrix can be decomposed into three matrices, each representing the rotation around a single axis. Positive yaw, pitch and roll represent a counterclockwise rotation around the Z, Y and X axis, respectively. Given yaw ψ, pitch θ and roll ϕ we can decompose the series of rotations as follows:

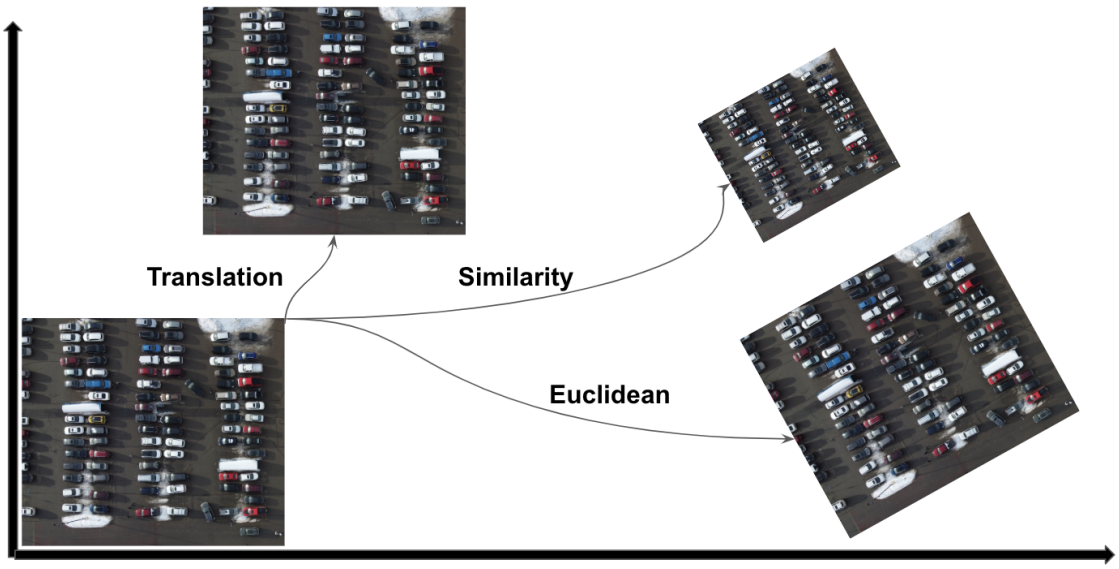

Figure 2.1: An example of various classes of transformation being applied to an image.

$$\begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \quad (2.3)$$

$$R = R_\psi R_\theta R_\phi \quad (2.4)$$

where R is the total rotation matrix, R_ψ is the yaw rotation matrix, R_θ is the pitch rotation matrix and R_ϕ is the roll rotation matrix.

2.1.2 Homogeneous Coordinate System

Homogeneous coordinates are a system of coordinates used in projective geometry. In general, they represent euclidean coordinates $P \in \mathbb{R}^N$ with an equivalent parameterization $\hat{P} \in \mathbb{R}^{N+1}$ [3]. Given a 3D euclidean coordinate $P \in \mathbb{R}^3$, the equivalent homogeneous coordinates $\hat{P} \in \mathbb{R}^4$ as follows:

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (2.5)$$

$$\hat{P} = c \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (2.6)$$

where c is a scaling parameter. By factoring out c from the last element of $\hat{P}$, we recover the original euclidean parameterization P in the first three elements of $\hat{P}$. The advantage of using

homogeneous coordinates is combining affine transformations and perspective transformations in the same matrix. This allows us to conveniently chain transformations by taking the product across a sequence of transformation matrices. We can represent a generic transformation consisting of rotation followed by scaling and translation as follows:

$$c\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_x & 0 & 0 & 0 \\ 0 & a_y & 0 & 0 \\ 0 & 0 & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.7}$$

$$\hat{P}' = ATP \tag{2.8}$$

where $\hat{P}$ is the coordinate of the original point and $\hat{p}'$ the corresponding point after the transformation has been applied. A is the scaling matrix, T is the matrix that encodes a rotation followed by a translation.

2.1.3 North East Down Coordinate System

With a preliminary understanding of coordinate systems in place, we can define some domain-specific conventions for coordinate systems in aerial imagery. In particular, the North East Down Coordinate System (NED) is often used in aviation. The X, Y and Z axis point North, East and downwards. Rotation around the X, Y and Z axis in the counterclockwise direction represents a positive Roll, Pitch and Yaw, respectively [3].

2.2 Image Formation

With an understanding of some basic geometric primitives in place, image formation is described in this section. At its core, the image formation process involves mapping a 3D scene to a 2D image. The mapping is conditioned on the scene geometry, lighting conditions, surface properties and camera optics. A camera model provides a mathematical formulation of this process. In this way, a camera model is a fundamental building block in obtaining information about the physical environment from images.

2.2.1 Pinhole Camera Model

Although several camera models exist based on the camera's characteristics, the pinhole camera model is often used because of its generality and simplicity. Using the parameter definitions in Section 1.3, a pinhole camera maps a 3D point P_O to its corresponding pixel location p_I as follows:

$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & -t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & -t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & -t_z \end{bmatrix} \begin{bmatrix} X_O \\ Y_O \\ Z_O \\ 1 \end{bmatrix} \tag{2.9}$$

$$p_I = {}^{I}K_C \begin{bmatrix} {}^{C}R_O & {}^{C}t_O \end{bmatrix} P_O \tag{2.10}$$

$$p_I = {}^{I}K_C {}^{C}M_O X_O \tag{2.11}$$

where ${}^{I}K_C$ is the intrinsic matrix and ${}^{C}M_O$ is the extrinsic matrix. The intrinsic matrix ${}^{I}K_c$ relates a camera's internal properties to an ideal pinhole-camera model. The intrinsic parameters include the focal lengths f_x and f_y as well as the principal point coordinates (c_y, c_y). Alternatively, the extrinsic matrix ${}^{C}M_O$ maps coordinates in the world coordinate system P_O to coordinates in the camera coordinate system P_C. ${}^{C}M_O$ includes a rotation matrix ${}^{C}R_O$ and a translation vector ${}^{C}t_O$. ${}^{C}R_O$ is the inverse of the drone's orientation in the world coordinate system ${}^{O}R_C$:

$${}^{O}R_C = R_\psi R_\theta R_\phi \tag{2.12}$$

following the definition in Equation 2.3. The translation vector ${}^{C}t_O$ is the drones position vector ${}^{O}t_C$ scaled by -1.

2.3 Deep Learning

Artificial Intelligence (AI) is a long-standing field that involves developing systems to perform tasks that generally require human intelligence. A central feature of AI-based systems is their ability to improve performance based on collected data iteratively. Traditionally, AI-based systems were underpinned by knowledge bases curated by in-the-loop humans. Although these systems successfully completed tasks specified by a list of formal rules, they failed in cases involving more intuitive reasoning. A fundamental limitation of rule-based systems is their reliance on humans to hand engineer the knowledge base. Machine learning was introduced as a way for systems to automate knowledge acquisition using data to extract patterns. This enabled AI systems to tackle a broader range of problems involving intuitive reasoning.

The success of machine learning depends heavily on the features, otherwise known as representation, input into the model. This is because the representation contains the information we use to generate the prediction. A substantial amount of time is spent on feature engineering to ensure the representation input into the model is optimal for the prediction task. However, in some cases, it's difficult to know the features that should be incorporated into the representation. To overcome this challenge, representation learning is used. It involves learning a mapping from raw inputs to representations with strong predicitve power. The learned representations can then be used by machine learning models to generate predictions.

Representation learning is a difficult task, especially in the case of high-dimensional features such as images or natural language. The high dimensional input space makes it difficult to disentangle the factors of variation, which are essential for the prediction task. Deep learning

surmounts this by decomposing a mapping from input features to representation into a series of nested mappings. Generally, these mappings consist of layers that apply a nonlinear function to a linear combination of the inputs to produce an output. By constructing a model with nested layers, the input is iteratively refined from low-level features to a high-level representation. The final layer is responsible for projecting the refined representation into the output space.

2.3.1 Deep Learning Architectures

The architecture of a deep learning model is a description of the operations of each layer, along with how the layers are arranged. A multitude of deep learning architectures exists with different characteristics. The choice of architecture depends on various factors, including the characteristics of the input data and the prediction task at hand. Two deep learning architectures relevant to this project are the Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN).

MLP is a deep learning architecture consisting of a series of nested, fully connected layers that map input and output nodes. The output nodes are given by a linear combination of the input nodes with learned coefficients followed by an activation function. The activation adds a non-linearity that is necessary to approximate a broader class of functions. Without a non-linear activation function, an MLP with an arbitrary amount of layers can be re-expressed as a two-layer MLP. The number of nodes in the first and last layers equals the model input and output dimensions, respectively. Conversely, the number of nodes in intermediate layers are hyperparameters of the model. Although successfully applied to various prediction tasks, the MLP has some drawbacks. Namely, the fully connected nature of MLP makes them prone to overfitting. This is especially the case in high-dimensional input spaces such as images.

CNN have several strong inductive biases that make them suitable for image data - the focus of this analysis. In contrast to MLP, CNN avoid overfitting through weight sharing, which acts as regularization. At its core, the CNN consists of layers in which learned kernels are convolved with the layer input to yield the layer output. The convolution operation in this discrete image-space setting involves applying a 2D filter to select indices of the 2D input signal and computing the Frobenius norm. The stride of the convolution controls how sparsely the indices are sampled from the input signal. A nonlinear activation function is applied to the feature map to generate the output. Max Pooling is often used at the end of the convolutional layers. The max pooling operation involves downsampling the spatial resolution of feature maps by only selecting the maximal activation across local regions of the feature map using a sliding window approach.

The input to a CNN is an image that is represented 3D tensor. A series of convolutional layers map the input image to a representation fed to the head to generate the output. Each layer learns a set of filters that capture different elements of the input signal. The number of convolutional filters is increased in consecutive layers. At the same time, the spatial resolution of the feature maps is down-sampled either via Max Pooling or by increasing the stride of the convolution operation. Thus, across layers, the spatial resolution of feature maps is decreased while the depth is increased. Throughout this process, the feature maps are refined from low-level to high-level features that are passed to the head.

The head of the CNN projects high-level features to the output space. The architecture of the head depends on the prediction task. In the case of image classification, the prediction head

is a fully connected layer that maps a representation to a distribution over class labels. For the object detection task, the architecture of the head consists of two convolutional components that localize and classify object instances, respectively.

2.3.2 Neural Network Training

During the training phase, the neural network parameters are optimized to minimize a specified loss function. The loss function characterizes the difference between the model's prediction and the corresponding ground truth labels. Several loss functions are popular, including Mean Squared Error (MSE) and Cross Entropy Loss. MSE is often used in regression tasks and is calculated as the average squared error between prediction and ground truth. The cross-entropy loss is used in classification as it is a measure of divergence between two distributions.

A Stochastic Gradient Descent (SGD) based algorithm is used to optimize the parameters of the network. SGD involves iteratively sampling batches of data from the train set and computing updates that are applied to the model parameters. The model updates are computed as the negative gradient of the loss with respect to the parameters. The size of the update is controlled by a parameter called the learning rate which is multiplied to the negative gradient prior to being applied to the parameters. This update process occurs for each batch in the train set and is repeated for a specified number of epochs. Several variants of the SGD algorithm have been proposed that have better convergence properties. Central among these innovations is the inclusion of a momentum term and the introduction of adaptive learning rates.

The backpropagation algorithm is used to efficiently compute the gradient of the loss with respect to the parameters. Backpropagation makes it tractable to train deep learning models with potentially billions of parameters. The operations in the model are used to define a computation graph. The input is mapped to the output in the forward pass through the network. The result of each computation in the forward pass is cached in a node in the computation graph. During the backward pass, the gradient of the objective with respect to the parameters is calculated via the chain rule. To this end, the gradient is computed one layer at a time, starting from the final layer and iterating backward. The use of reverse mode differentiating avoids the redundant computation of intermediate terms.

Chapter 3

Image Stitching

3.1 Overview

An essential aspect of the proposed parking utilization framework is the ability to generate mosaics given sets of input images. This is the responsibility of the image stitching component, which is described in this section. Image stitching is a well-studied topic in computer vision that involves aligning a set of images of the same scene captured from different locations and orientations [27]. In this thesis, we used an image-stitching algorithm that involves two high-level steps: homography estimation and global alignment. In the homography estimation step, the transformations that map one image to the coordinate system of another are estimated. In the subsequent global alignment step, the initial pose of each image in the mosaic is updated to approximate the relative transformations determined in the homography estimation step. As part of the global alignment step, a novel algorithm is proposed for initializing the pose of the images in the mosaic using metadata from the drone. Specifically, the drone's extrinsic and intrinsic parameters are used to initialize the mosaic images effectively.

The following chapter provides a detailed overview of the image stitching algorithm. This includes outlining the homography estimation and global alignment step. As a novel contribution, the image pose initialization algorithm is described in detail. The algorithm is assessed in numerous experiments on the SkyDeploy dataset. These experiments seek to evaluate the image stitching algorithm's effectiveness and find the best setting of hyperparameters. Additionally, experiments are used to evaluate the effectiveness of the proposed image initialization algorithm. Finally, the chapter concludes by summarizing the experiments' quantitative and qualitative results.

3.2 Homography Estimation

3.2.1 Keypoint Detection

The first step of the homography estimation process involves detecting and describing the key points for each image with a vector. The goal is to generate keypoints with descriptors that only match the corresponding point in other images. Scale Invariant Feature Transform (SIFT) [20] is a method for robustly detecting and describing image keypoints. The location of key-

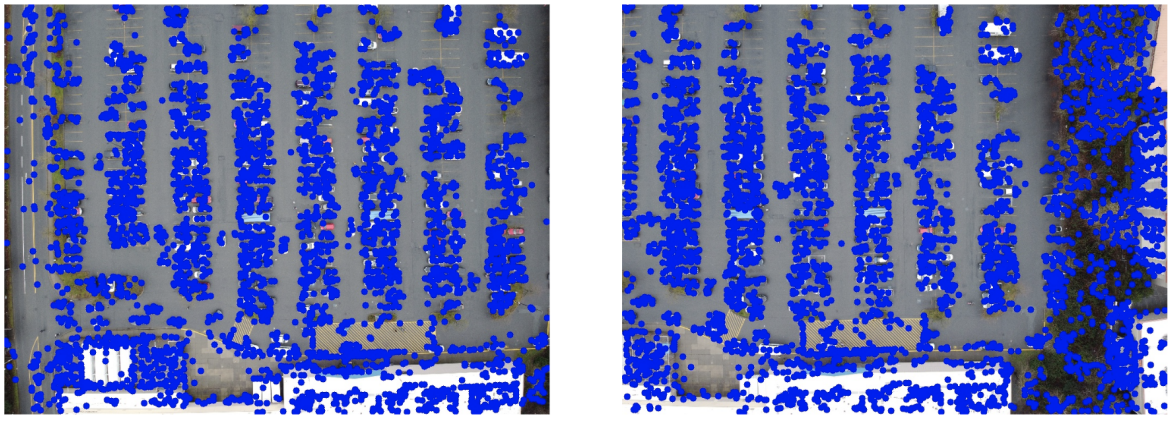

Figure 3.1: An example of keypoints detected on two sample images.

points are maxima and minima of the Difference of Gaussians (DoG) computed at multiple scales. The DoG approximates the Normalized Laplacian of Gaussians (NLoG) - a second-order method to detect edges that uses gaussian smoothing to mitigate noise. The DoG is computed for successive gaussian blurred images with various variance scaling parameters. With the resulting stack of DoG, the 3D space is traversed with a 3D filter that finds extrema. These points are considered to be interest point candidates. Certain heuristics filter out weak extrema to generate the final set of interest points. The size of the keypoint depends on the variance of images in the gaussian scale space that was used to generate the DoG. Thus, the DoG generated by low-variance gaussian blurred images will detect fine grain keypoints and the DoG generated by high variance gaussian blurred images will detect large keypoints. This allows the detection of keypoints of different sizes.

Since the objects we are trying to find correspondences across images may be of different scales and orientations, we must have a representation invariant to these factors. Fortunately, keypoints are detected at multiple scales to achieve scale invariance. To achieve rotational invariance, the image gradient is computed for a neighborhood of pixels around the keypoint. For each pixel, there is a gradient magnitude and direction. By quantizing the direction of gradients into several bins and allocating pixels to bins according to their gradient direction, we can obtain a histogram of gradients. The bin with the highest count is chosen to be the principal orientation and the remaining gradient directions are expressed with respect to it. It is important to note that the magnitude information associated with the image gradients is not used. This is because the gradient magnitudes will be susceptible to changes in color or brightness. Thus, the algorithm is invariant to these conditions by discarding the gradient magnitudes and only using the gradient directions. An illustration of the keypoints detected by the SIFT algorithm on two sample images is available in Figure 3.1.

3.2.2 Keypoint Description

The keypoint detection step generated the location, scale and orientation for a set of keypoints in the image. Subsequently, each keypoint needs to be described by a representation known as a descriptor. The descriptors are generated using similar techniques employed in the key-

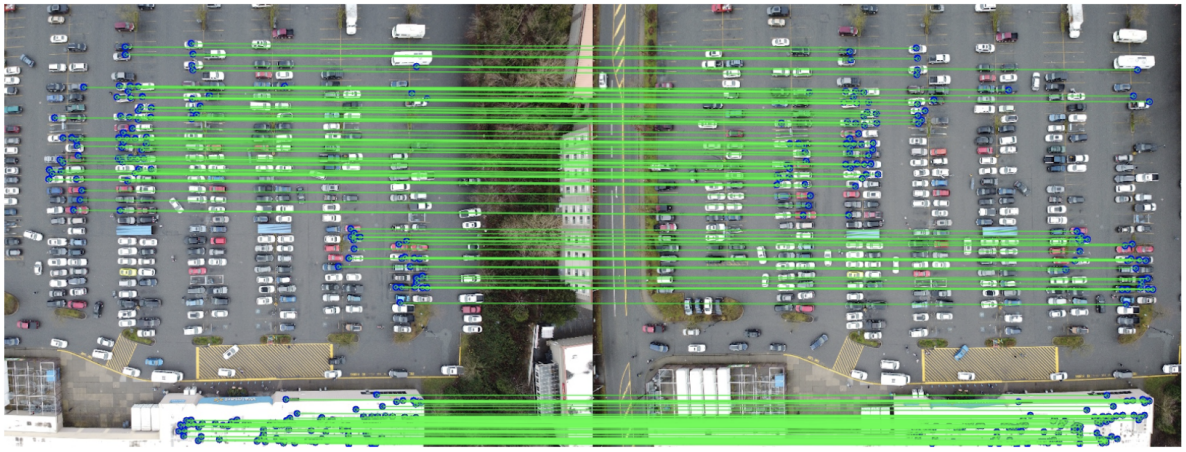

Figure 3.2: An example of feature matches on a pair of overlapping images.

point detection phase. Specifically, image gradients from a neighborhood of pixels around the keypoint are used to generate Histogram of Gradient (HOG) [4] based representations. Except in this case, this process will be repeated across four square sub-quadrants in the neighborhood of pixels surrounding the keypoint. The HOG representations of the four quadrants are concatenated into a single vector to form the SIFT descriptor.

3.2.3 Keypoint Matching

Following keypoint detection and description, correspondences across images are determined via a keypoint matching algorithm. In the simplest case, a brute-force approach involves computing a distance between each pair of keypoints across images. The distance metric of choice is often the L2 distance. The L2 distance is optimal, assuming the data is generated from a Gaussian distribution. A more efficient alternative involves approximate k-d tree matching [1]. A k-d tree is a binary tree where each node represents a d-dimensional point that acts as a hyperplane splitting the space into two parts. As such, the k-d tree allows quick querying of the approximate nearest neighbor of a given feature. An example of the matches generated using approximate k-d tree matching on a pair of overlapping images is available in Figure 3.2.

3.2.4 Pairwise Transformation

Once keypoints have been matched, the transformation can be determined by solving the system of equations specified by the correspondences. Due to the presence of false correspondences, the system of equation is inconsistent and no unique solution exists. Thus, methods like Random Sample Consensus (RANSAC) [5] is used to yield an approximate global solution. RANSAC is an iterative method for solving an over-determined system of equations that contains outliers. Each round, a random subset of correspondences are chosen to fit a model of the data. The model is validated across all data points and the number of outliers are recorded each round. The procedure is repeated for a specified number of rounds and the model with the lowest number of outliers is returned. In this way, RANSAC provides an approximate

Figure 3.3: An example of the pairwise transformation computed between a pair of overlapping images.

global solution for the over-determined system of equations. The aforementioned procedure is repeated for each pair of images in the set to yield the pairwise relative transformations. To account for the situation where all images in the set may not be overlapping, a heuristic such as specifying a minimum number of feature matches [27] between pairs of images may be implemented. This ensures only pairs of images that are likely to be overlapping will have transformations computed. An example of the pairwise transformation computed between a pair of overlapping images is illustrated in 3.3.

3.3 Global Alignment

In the homography estimation stage, the relative transformations between each pair of images are estimated using the procedure above. These will act as ground truth relative transformations between images. In the global alignment step, the parameters of each image that describe its pose in the mosaics coordinate system are optimized so that the relative transformations approximate the ground truth relative transformations from the homography estimation step. A fundamental prerequisite to this procedure is the accurate initialization of the parameters describing each image pose in the mosaic coordinate system. Fortunately, each image cap-

tured by the drone has associated metadata, including the location, orientation, and intrinsic parameters of the camera. By leveraging this information, each image can be initialized with informative parameters. This ensures that the optimization procedure converges to an optimal local minimum.

3.3.1 Image Pose Initialization

In the following section, we propose a novel algorithm to yield an initial set of parameters that will be optimized during the global alignment step. This set of parameters describes the location and orientation of each image in the mosaic. In particular, four parameters are used for each image: x_M, y_M, ψ_M and s_M. (x_M, y_M) is used to describe the location of the image center in the mosaic coordinate system. Alternatively, ψ_M is the yaw and s_M is the scale of the image in the mosaics coordinate system. The image pose initialization process is illustrated in equation 3.4. The metadata M_{ipt} associated with each image, as specified in 1.1, is used to generate image initializations.

First, the mapping ${}^{I}T_{O}$ that defines each image is recovered using the camera's extrinsic parameters and intrinsic parameters. The extrinsic parameters are used to define a transformation from the world coordinate system to the camera coordinate system, which includes a rotation and translation. The rotation is represented by a matrix ${}^{C}R_{O}$. The translation is represented by a vector ${}^{C}t_{O}$. Alternatively, the intrinsic parameters relate a camera's internal properties to an ideal pinhole-camera model via a matrix ${}^{I}K_{c}$. Using the definition of the metadata in Section 1.3 and the pinhole camera model in Section 2.2.1, the mapping ${}^{I}T_{O}$ from the world coordinate system to the image coordinate system is defined as:

$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & -t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & -t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & -t_z \end{bmatrix} \begin{bmatrix} X_O \\ Y_O \\ Z_O \\ 1 \end{bmatrix} \tag{3.1}$$

$$p_I = {}^{I}K_C \begin{bmatrix} {}^{C}R_O & {}^{C}t_O \end{bmatrix} P_O \tag{3.2}$$

$$p_I = {}^{I}K_C {}^{C}M_O P_O \tag{3.3}$$

$$p_I = {}^{I}T_O P_O \tag{3.4}$$

Each image is projected into a different camera coordinate system. The images must be expressed in a common coordinate system to account for this. We can define a virtual camera that maps 3D points to mosaic coordinates p_M:

$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\hat{t}_x \\ 0 & 1 & 0 & -\hat{t}_y \\ 0 & 0 & 1 & -\hat{t}_z \end{bmatrix} \begin{bmatrix} X_O \\ Y_O \\ Z_O \\ 1 \end{bmatrix} \tag{3.5}$$

$$p_M = {}^{M}K_V \begin{bmatrix} {}^{V}R_O & {}^{V}\hat{t}_O \end{bmatrix} P_O \tag{3.6}$$

$$p_M = {}^{M}K_V {}^{V}M_O P_O \tag{3.7}$$

$$p_M = {}^{M}T_O P_O \tag{3.8}$$

The absolute orientation of the drone is chosen such that the optical axis is perpendicular to the plane that defines the parking lot being captured. This is the case when $(\psi, \theta, \phi) = (0, 0, 0)$. The resulting rotation matrix ${}^{V}R_O$ is the identity matrix. The virtual camera is chosen to have ${}^{V}t_O$ that is the mean across all position vectors ${}^{O}t_C \in \mathbf{I}$ scaled by -1. The intrinsic matrix is parameterized by focal lengths f_x and f_y. The principal point is omitted since we want to have the mosaic centered at the origin.

To obtain the mosaic coordinates p_M from image coordinates p_I, we must first project p_I to the 3D point corresponding to P_O. To this end, pixel coordinates p_I are mapped to direction vectors $\vec{p}_I$ by decomposing and inverting the mapping ${}^{I}T_O$:

$$\vec{p}_I = {}^{C}R_O^{-1}K^{-1}p_I - {}^{C}t_O \tag{3.9}$$
$$\vec{p}_I = {}^{O}R_C K^{-1}p_I + {}^{O}t_C \tag{3.10}$$

These direction vectors are rays from a pixel in the image to the real-world point corresponding to P_O. This point can be computed as the intersection between $\vec{p}_I$ and the plane B, which describes the ground of the parking lot. B is defined by a normal vector $b = [0, 0, 1]$ and a point that lies on plane $q = [0, 0, 0]$. Given B and $\vec{p}_I$, the intersection point P_O is trivial to compute.

In turn, P_O can be projected into the mosaic coordinate system via ${}^{M}T_O$. Correspondences between $\hat{p}_I$ and p_M can be used to solve a system of equations that yields the affine transformation parameterized by the scale s_M, the location (x_M, y_M) and the yaw ψ_M. $\hat{p}_I$ is the pixel coordinates p_I after being translated, so the center of the image aligns with the origin of the coordinate system. Given that there are four unknowns and each correspondence consists of an (x, y) coordinate, at least two correspondences are needed to solve for the unknown. In practice, the correspondences between the corners of the images are used to solve for the unknowns. The resulting transformation H mapping image coordinates p_I to mosaic coordinates p_M is defined as follows:

$$s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_M\cos\psi_M & -\sin\psi_M & x_M \\ \sin\psi_M & s_M\cos\psi_M & y_M \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & -\frac{w}{2} \\ 0 & 1 & -\frac{h}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$
$$p_M = ATp_I \tag{3.12}$$
$$p_M = Hp_I \tag{3.13}$$

where T is a matrix that centers the image in the mosaics coordinate system using the image's width w and height h. A is an affine transformation encoding the pose of the image in the mosaics coordinate system. This process is repeated for $I_i \in \mathbf{I}$ to obtain H_i. x_M, y_M, ψ_M and s_M parameterize H_i and are optimized over in the subsequent global alignment step.

3.4 Optimization Procedure

During the global alignment step, the initial set of parameters s_M, x_M, y_M and ψ_M that encode the pose of images in the mosaics coordinate system H_i are refined. They are optimized so

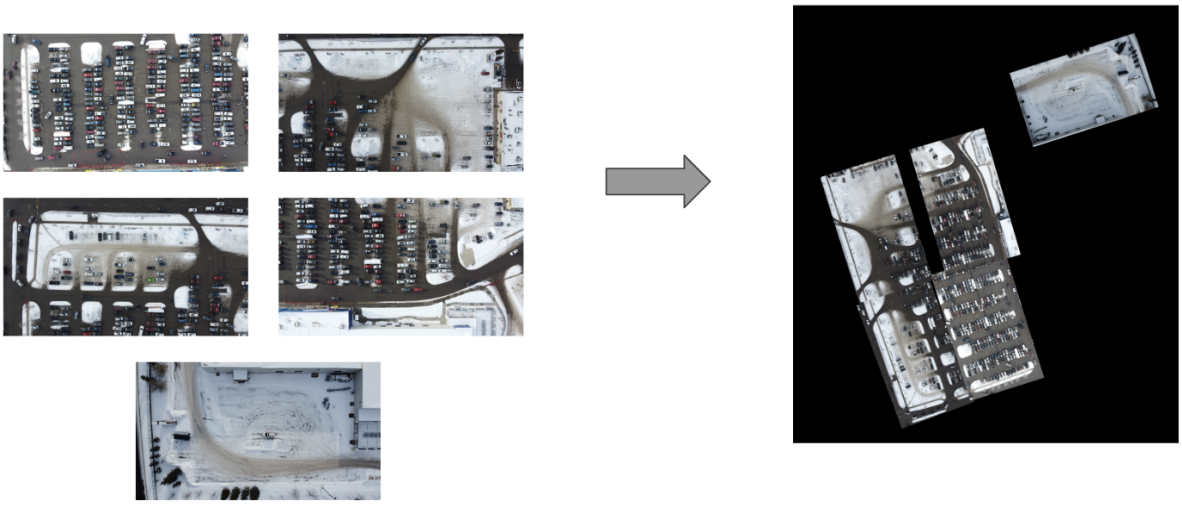

Figure 3.4: Overview of the image initialization process.

that the relative transformations approximate the ground truth relative transformations from the homography estimation step. In order to accomplish this, relative transformations $E_{i,j}$ must be derived for all pairs of global transformations H_i and H_j in which the underlying images overlap:

$$H_i p_{I,i} = H_j p_{I,j} \tag{3.14}$$
$$p_{I,i} = H_i^{-1} H_j p_{I,j} \tag{3.15}$$
$$p_{I,i} = E_{i,j} p_{I,j} \tag{3.16}$$

where $p_{I,i}$ is a point in I_i and $p_{I,i}$ is a point in I_j. Let $\Theta \in \mathbb{R}^{4N}$ be a parameter vector formed by concatenating the scale s_M, location (x_M, y_M) and yaw ψ_M of each image. Thus, the objective is to minimize the squared distance between the points projected by the ground truth relative transformations $G_{i,j}$ and the points projected by estimated relative transformations $E_{i,j}$.

$$\min_{\Theta} \sum_{i=1}^{N} \sum_{j=1}^{N} \|E_{i,j} p_{I,i} - G_{i,j} p_{I,i}\| \tag{3.17}$$

As a differentiable function, the object is minimized using gradient descent. Gradient descent involves iteratively calculating the gradients with respect to the objective and updating the parameters in the opposite direction. The gradient of a function represents the direction of the most rapid ascent, so iteratively updating the parameters by stepping in the opposite direction results in convergence to a local minimum.

3.5 Implementation Details

The aforementioned Image Stitching algorithm was implemented using the Python Programming Language. In particular, the OpenCV package is used extensively in the homography

estimation step to detect keypoints, compute descriptors, find correspondences between images and solve for transformations via RANSAC. In the subsequent global alignment step, the Automatic Differentiation engine of the PyTorch package is used to perform the gradient descent procedure. Other notable packages include matplotlib for plotting utilities, numpy/scipy for scientific computing and pandas to store metadata for images in the set.

3.6 Experiments

In this section, the experiments used to test the efficacy of the image stitching algorithm are described. The experiments are performed using a private dataset of images introduced in Chapter 1. Specifically, the dataset contains sequences of image sets that span a particular parking lot across several time intervals throughout the day. Each parking lot is surveyed 11 times at half-hour intervals from 12:00 to 17:00. No ground truth labels exist for the pose of images in the mosaic. Accordingly, we developed a suite of quantitative and qualitative tools to assess the stitched images. In the Quantitative Analysis section, experiments are conducted to evaluate the quality of stitched images over different hyperparameters settings. These experiments aim to find the optimal hyperparameter configuration for the image stitching algorithm. Subsequently, the results of the image stitching algorithm are interpreted visually in the quantitative analysis section. This includes generating a visualization of the mosaic, image initialization, and optimization process. The purpose of the qualitative analysis is to support the results in the quantitative analysis and offer some interpretability.

3.6.1 Hyperparameter

This section outlines experiments that aim to discover the optimal hyperparameters for the image stitching algorithm. Each setting of the hyperparameters is evaluated on a benchmark image stitching task. The configuration of hyperparameters that achieves the best performance on the benchmark is chosen as the optimal set of hyperparameters. The benchmark task involves producing mosaics for a dataset of 55 sets of images sampled from the SD dataset. For each setting of hyperparameters, the mean squared link loss is reported. The link loss measures how consistent the pose of images in the mosaic is with the relative transformations derived in the homography estimation step. By taking an average of the link losses across an image, we can estimate the quality of the stitched image. A lower mean squared link loss indicates the estimated relative transformation between pairs of images in the mosaic approximates the relative transformations that we solved using SIFT and RANSAC. Intuitively, this translates to a more cohesive fit among the images in the mosaic. Thus, the hyperparameters with the lowest mean squared link loss are considered the optimal set. In essence, a grid search is being performed across hyperparameters. A set of six hyperparameters are chosen for the grid search that relates to the global alignment process of the algorithm. The hyperparameters involved in the pairwise image stitching process include:

- **Theta Learning Rate**: The learning rate of the optimizer for the theta (yaw) parameters. A higher Theta Learning Rate will lead to larger updates to the theta parameters throughout the optimization process.

- **Scale Learning Rate**: The learning rate of the optimizer for the scale parameters. A higher Scale Learning Rate will lead to larger updates of the scale parameters throughout the optimization process.

- **Translation Learning Rate**: The learning rate of the optimizer for the translation parameters. A higher Translation Learning Rate will lead to larger updates of the translation parameters throughout the optimization process.

The aforementioned hyperparameters represent a set of levers that control various aspects of the optimization procedure. For the grid search, three values per hyperparameter are specified. The values are chosen to be in the neighborhood of the default value of the hyperparameter. Specifically, the values for the theta learning rate as well as the scale learning rate $\in \{.0001, .001, .01\}$ and the values for the translation learning rate $\in \{.005, .05, .5\}$. Lower values are used for the theta and scale learning rate parameters because they are sensitive to optimize over. In total, $3^3 = 27$ configurations of hyperparameters will be evaluated.

The results for this experiment are available in Table 3.1. Overall, the best performance is obtained when the theta learning rate is .0001, the scale learning rate is .001 and the translation learning rate is .05. Furthermore, for each type of hyperparameter, the best performance on average is obtained with the same values. In general, the median value of each hyperparameter yields the best overall performance. This reflects the need to carefully set the learning rate low enough to find the global minimum accurately but high enough to avoid getting stuck in local minimums. In the case of the theta learning rate, the best performance on average is obtained when the parameter is set to its minimum value. This can be attributed to the fact that this parameter corresponds to an angle, which makes it a sensitive parameter to optimize over.

3.6.2 Image Initialization Ablation Study

The following section outlines an ablation study to assess the effectiveness of the proposed image initialization algorithm. In order to do so, the algorithm's performance on the benchmark dataset is compared with and without using the image initialization algorithm. In the case where it is not used, images are initialized in the center of the mosaic with constant scale and no yaw. If using the image initialization algorithm is beneficial, it would follow that the mean squared link loss would be lower than the baseline. The results from this experiment are available in Table 3.2. The image initialization algorithm leads to a substantial increase in the performance. This validates its use within wider image stitching algorithm.

3.6.3 Visual Results

The previous sections outlined quantitative experiments to optimize and validate various components of the image stitching algorithm. The following section explores visual results from the image stitching algorithm. Figure 3.8, 3.10 and 3.9 illustrate mosaics generated by the image stitching algorithm from the corresponding image sets 3.5, 3.7, 3.6. In almost all cases, the image stitching algorithm produces a mosaic that is geometrically consistent with the scene. This qualitatively further validates the results in the previous sections that demonstrate the effectiveness of the image-stitching algorithm.

Theta LR	Scale LR	Translation LR	Mean Squared Link Loss
0.0001	0.0001	0.005	178.35
0.0001	0.0001	0.05	165.69
0.0001	0.0001	0.5	210.21
0.0001	0.001	0.005	167.42
0.0001	0.001	0.05	156.81
0.0001	0.001	0.5	210.76
0.0001	0.01	0.005	194.23
0.0001	0.01	0.05	181.09
0.0001	0.01	0.5	225.36
0.001	0.0001	0.005	199.80
0.001	0.0001	0.05	193.79
0.001	0.0001	0.5	210.73
0.001	0.001	0.005	198.04
0.001	0.001	0.05	189.12
0.001	0.001	0.5	230.68
0.001	0.01	0.005	209.35
0.001	0.01	0.05	201.28
0.001	0.01	0.5	218.95
0.01	0.0001	0.005	214.39
0.01	0.0001	0.05	205.18
0.01	0.0001	0.5	223.81
0.01	0.001	0.005	208.50
0.01	0.001	0.05	202.72
0.01	0.001	0.5	223.46
0.01	0.01	0.005	230.97
0.01	0.01	0.05	225.21
0.01	0.01	0.5	257.89

Table 3.1: The results of the Hyperparameter experiments.

In general, image stitching results are promising across different parking lots. However, in a few cases, the algorithm may fail to produce a geometrically consistent mosaic. This occurs when the parking lot consists of a large number of images. A prime example is the mosaic in figure 3.11, which consists of 28 sub-images captured during poor weather conditions.

Other useful visualization utilities were developed to monitor the image stitching process. This includes the initialization diagrams and the link loss diagram. The initialization diagrams

Initialization Algorithm	Mean Squared Link Loss
	156.81
	321.54

Table 3.2: The results of the image initialization ablation study.

Figure 3.5: Langley parking lot consisting of 4 images.

illustrate the position and orientation of images in the mosaic. An example of the Mosaic Initialization Diagram is available in Figure 3.12. Each node in the graph represents the initial pixel location of the centre of an image in the mosaic. The green line that emerges from each node encodes the image's orientation in terms of the yaw angle. Nodes connected via a pink link denote that the underlying images overlap. Alternatively, nodes that are connected via a grey link denote that the underlying images are not overlapping. This diagram can be used to verify if the locations of images are initialized reasonably.

In addition to the initialization diagram, the link loss diagram is a helpful visualization for monitoring the image stitching process. The Link Loss Diagram is generated at the end of the global alignment step and depicts the loss between each pair of images. The loss is defined as the sum of squared errors over the same set of points projected by the ground truth relative transformation and the estimated relative transformation, respectively. A high loss reflects that the estimated relative and ground truth transformations are very different. Alternatively, a low loss reflects that the transformations are close together. This visualization can be used to determine which pairs of images are contributing most to the inconsistencies in the mosaic. Figure 3.13 shows an example of the link loss diagram.

Figure 3.6: Kelowna parking lot consisting of 6 images.

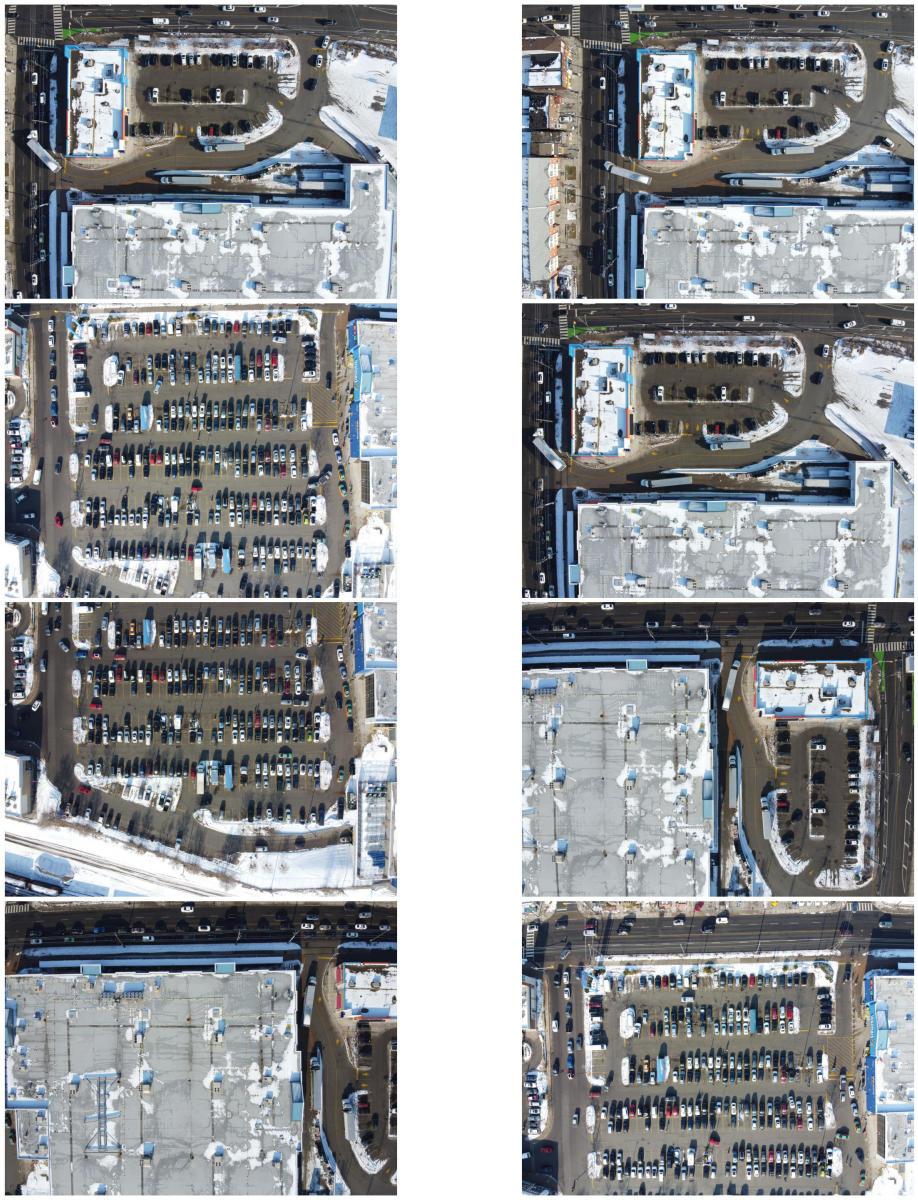

Figure 3.7: Stockyards parking lot consisting of 8 images.

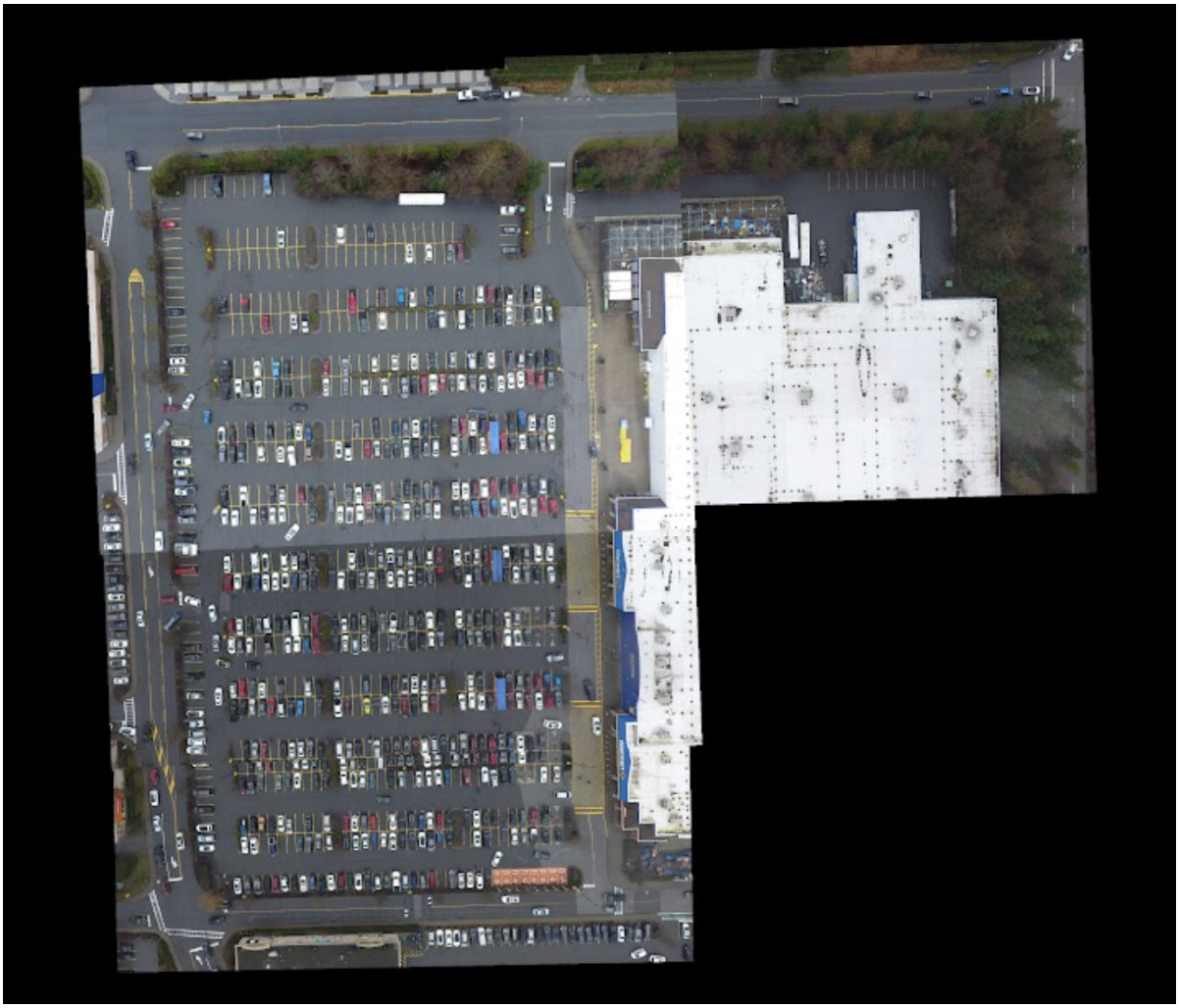

Figure 3.8: Result of the image stitching algorithm on the Langley image set.

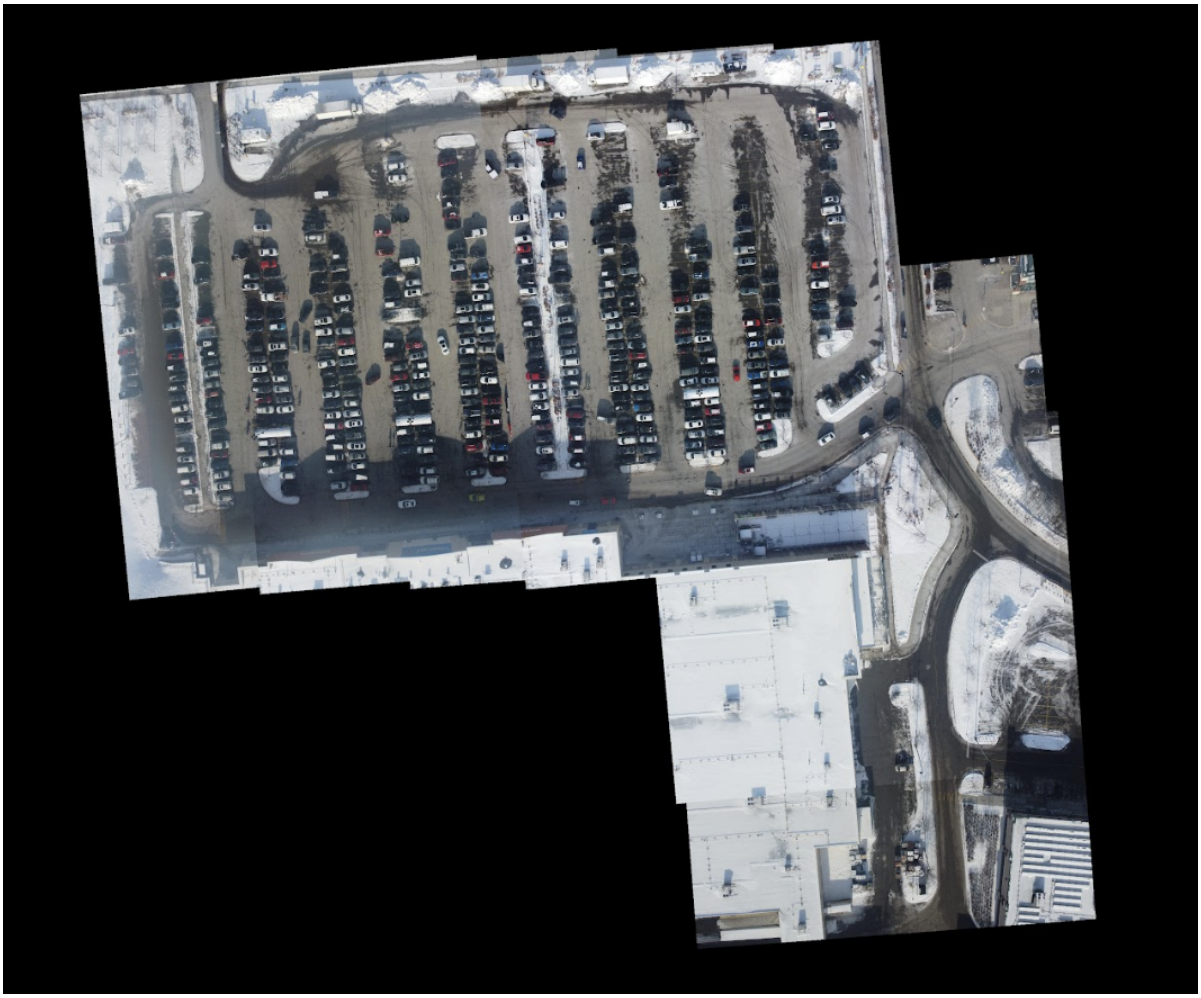

Figure 3.9: Result of the image stitching algorithm on the Kelowna Image Set.

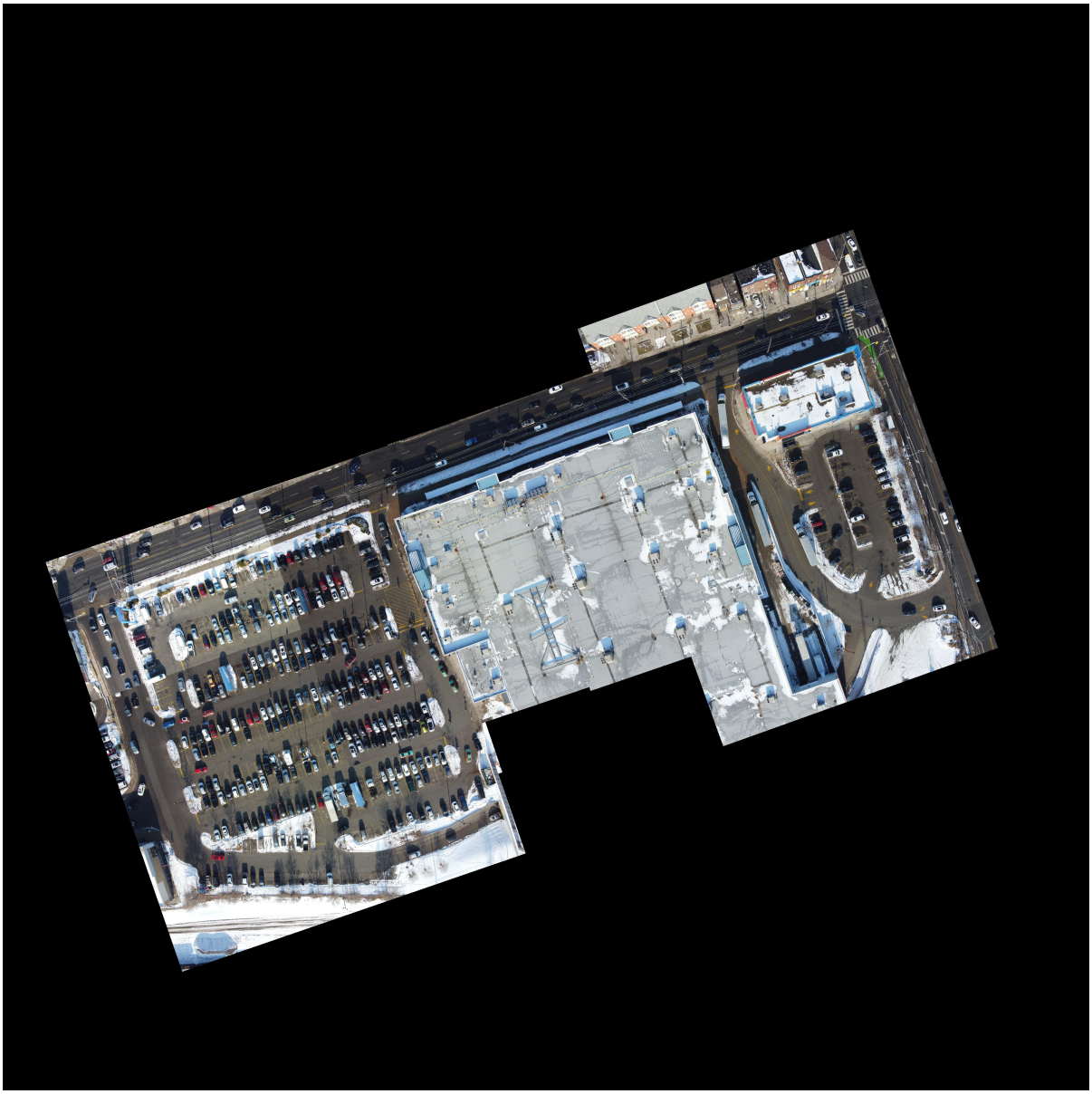

Figure 3.10: Result of the image stitching algorithm on the Stockyards Image Set.

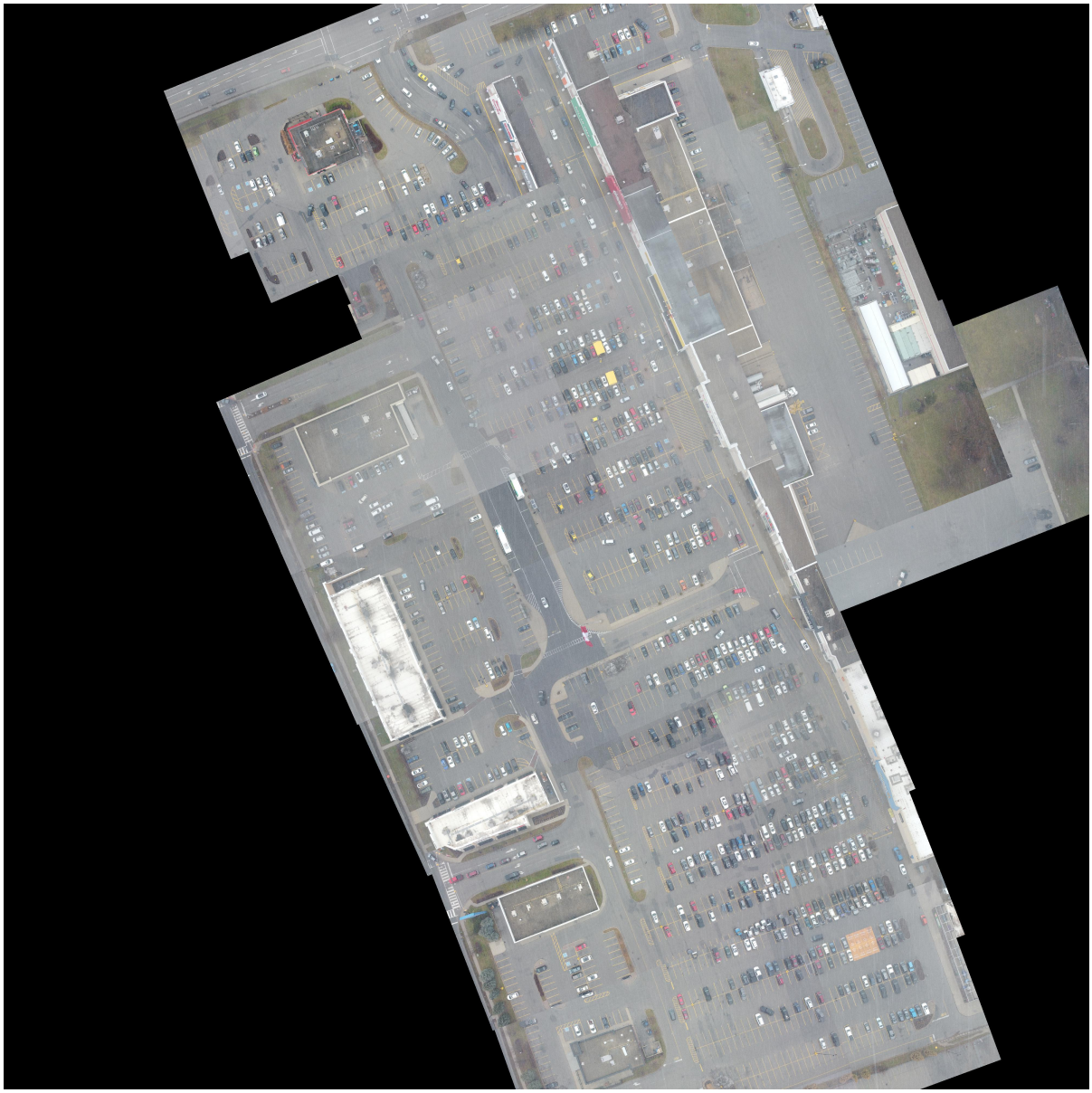

Figure 3.11: Subpar image stitching results obtained in case of large mosaic.

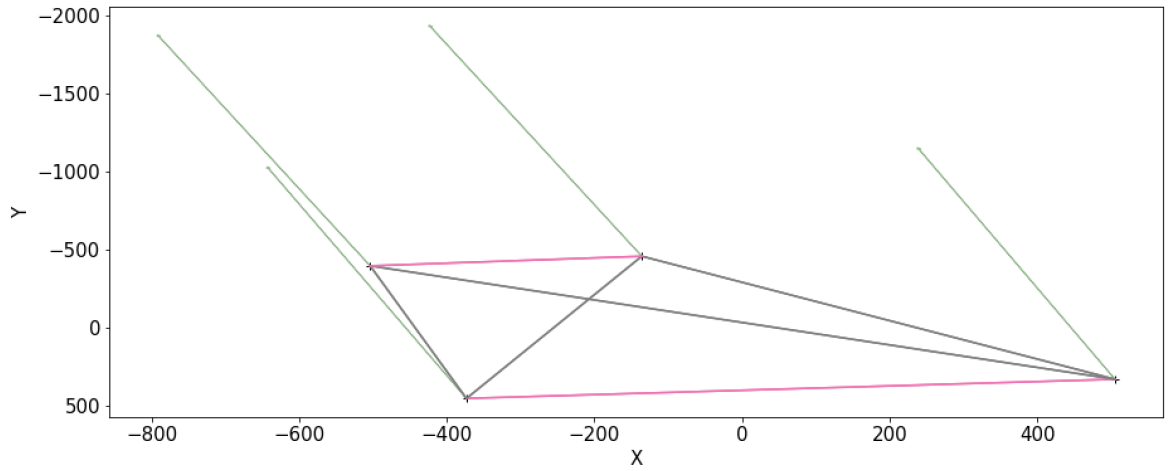

Figure 3.12: Graph view of image initialization. Nodes represent image centers and edges represent transformations between images. A pink (gray) edge indicates transformations exist (don't exist) between a pair of images.

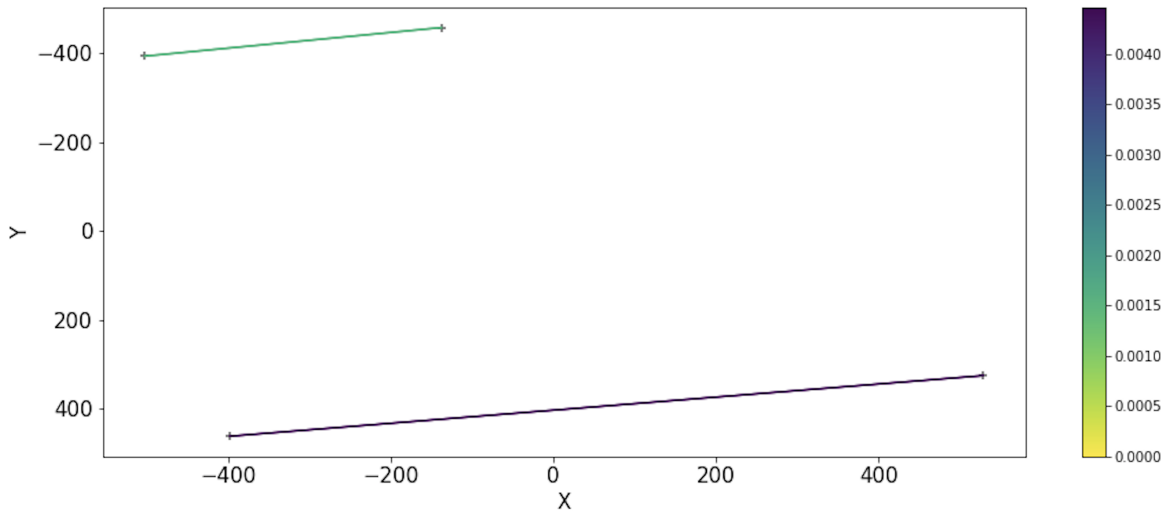

Figure 3.13: Graph that shows link loss for each link in the graph. The link loss is simply the l2 norm between points projected by the estimated relative transformation and true relative transformation.

Chapter 4

Vehicle Detection

4.1 Overview

A critical aspect of the proposed parking utilization framework is the ability to localize vehicle instances in aerial images. This is the responsibility of the vehicle detection component, which is described in this section. The vehicle detection component aims to map an input image to a series of coordinates describing the location and size of vehicle instances therein. The vehicle detection task is an instance of the object detection task - a well-researched topic in computer vision. Recently, a variety of state-of-the-art methods for object detection have been proposed that leverage Convolutional Neural Networks (CNN). CNN combines the traditionally disjointed process of feature extraction and classification/regression, allowing the model to be optimized end-to-end for the prediction task. These approaches enable downstream tasks such as vehicle detection.

This chapter will outline the experiments that benchmark state-of-the-art object detection methods. This analysis aims to determine the best method for vehicle detection to use within the context of our parking utilization assessment framework. The architectures involved in the comparative study include Single Shot Detector SSD [19] RetinaNet [17], MobileNet V3 [9], Fully Convolution One Stage Object Detection (FCOS) [28], Faster Regional Convolutional Neural Network (Faster RCNN). Additionally, the variants of the RetinaNet and Faster RCNN architecture presented in [32] and [16] are included. These approaches will be denoted RetinaNet V2 and Faster RCNN V2, respectively. The methods are evaluated on both the Sky-Deploy Dataset (SD) and the Car Parking Lot Dataset (CARPK) [11]. The SDY dataset is a private dataset consisting of aerial images of parking lots captured by drones. The task is to detect vehicle instances within the images. CARPK is a similar open-source dataset that is a popular benchmark for vehicle detection.

This is the first work to perform a comparative study on the specified object detection approaches using multiple parking lot datasets. Thus, the experiments are a valuable contribution to the object detection literature. Researchers and practitioners can use the results of the experiments to inform what architecture best fits their use case. The following will explain the architectures, datasets and experiments involved in the vehicle detection stage. Furthermore, a high-level discussion of heatmap generation is provided following the vehicle detection experiments. This includes high-level details of how the image stitching, vehicle detection and

heatmap generation component are orchestrated to generate heatmaps.

4.2 Object Detection

The object detection task involves localizing and classifying objects in an image. An object j in image i is characterized by a bounding box $b_{i,j}$. $b_{i,j}$ is a tuple that describes the location and class of an object:

$$b_{i,j} = < u, v, w, h > \tag{4.1}$$

where (u, v) is the pixel coordinates of the bottom left corner of $b_{i,j}$. w and h are the width and height of $b_{i,j}$ in pixels. Each $b_{i,j}$ has a corresponding label c that specifies the object's class. It is often represented as one hot encoding with dimension K where K is the number of classes. The set of bounding boxes for a given image is denoted B_i and the set of corresponding labels is denoted C_i.

Thus, an object detection method is specified as a function that maps images to a predicted set of bounding boxes $\hat{B}_i$ and class labels $\hat{C}_i$. Object detectors are trained on a dataset and evaluated on a held-out test set of samples. The training phase involves iteratively sampling images with their corresponding bounding boxes and labels. The object detector generates a predicted set of bounding boxes $\hat{B}_i$ and labels $\hat{C}_i$. Subsequently, a loss is calculated using the ground truth labels. The total loss consists of a regression loss calculated between B_i and $\hat{B}_i$ as well as a classification loss between C_i and $\hat{C}_i$. The total loss is back-propagated through the object detector with respect to the learnable parameters. Throughout the training phase, the model converges to a state where it can accurately localize and classify objects in input images. In the testing phases, the model's performance is assessed on samples the model has yet to encounter. Since we are evaluating the model's generalization performance, we do not update the parameters during the testing phase.

4.3 Architectures

Generally, the architecture of deep learning-based approaches to object detection consist of a backbone, neck and prediction head as depicted in 4.1. The backbone is a pretrained convolutional neural network that extracts rich features from multiple resolutions. The backbone is often ResNet [8] or VGG [25]. The features extracted from the backbone are input into the neck module, which aggregates features across resolutions. In particular, high-resolution, low-level features from earlier layers are supplemented with information from low-resolution, high-level layers from later layers. This process enhances the predictive power of layers earlier in the network. Lastly, the head produces the bounding box predictions at each resolution. The multi-resolution approach to prediction allows for detecting an object at multiple scales. Typically, high-resolution feature maps yield detections for small objects and low-resolution feature maps yield detections for large objects.

Deep-learning-based approaches to object detection can be classified into one-stage and two-stage detectors. One-stage detectors map the input image directly to a set of bounding

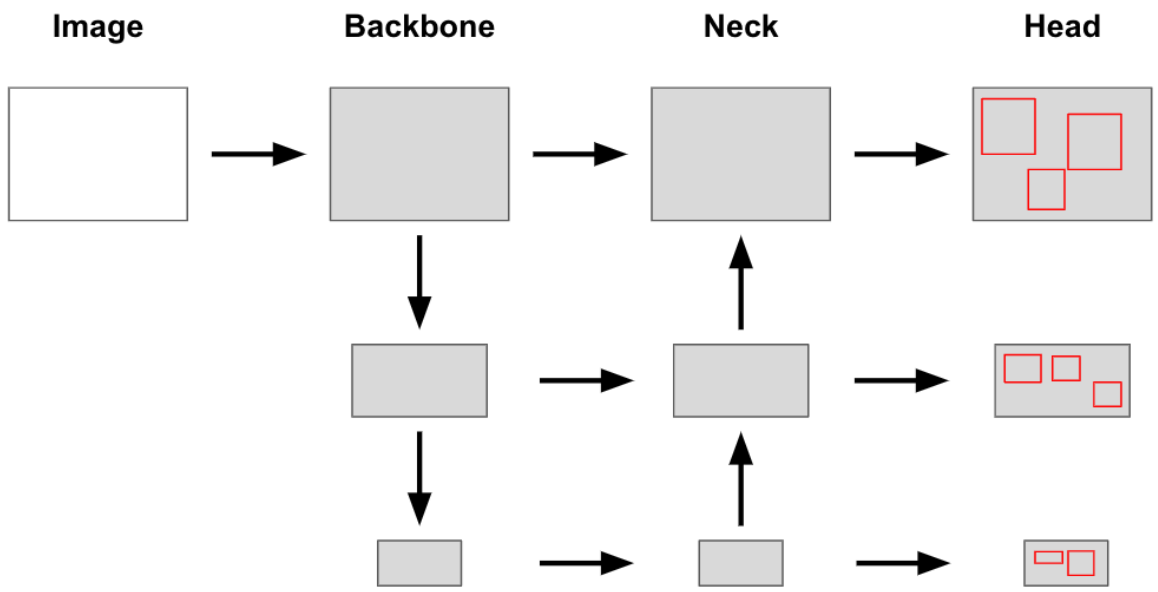

Figure 4.1: A high level illustration of deep learning based object detection methods.

boxes $\hat{B}_i$. On the other hand, two-stage detectors first map an image to a set of object proposals that localize parts of the image that are likely to contain an object. The object proposals are then refined and classified to generate the final prediction B_i. An illustration of one-stage detectors and two-stage detectors is available in Figure 4.2. The following sections outline the comparative study's one- and two-stage object detection methods.

4.3.1 One Stage Detectors

One stage object detectors map input images to bounding box coordinate $\hat{B}_i$ and corresponding labels $\hat{C}_i$. Because of this, one-stage detectors are generally much more efficient than two-stage detectors. The one-stage detectors explored in the vehicle detection comparative study are the Single Shot Detector(SSD) [19], RetinaNet [17], RetinaNet V2 [32], FCOS [28] and MobileNet V3 [9].

SSD: SSD is one of the seminal approaches to one stage object detection. It leverages a pretrained VGG-16 [25] to extract features from input images. The output from several layers is used to generate features of multiple resolutions. Multi-resolution features facilitate the detection of objects at multiple scales. A series of convolutional layers are used to map the features at a given resolution to the corresponding bounding box predictions. Dilated convolutions are used to increase the receptive field of the filters while keeping the number of parameters constant. The number of potential outputs across resolutions is very large. This sets SSD apart from previous one-stage detectors, which generate a limited number of outputs at a single resolution. SSD is trained with a loss function that is the summation of an L1 localization loss and a classification confidence loss. Throughout the training process, data augmentation, such as random cropping, is used and is shown to improve performance substantially.

FCOS: In [28], a fully convolutional one-stage object detector (FCOS) is proposed. FCOS

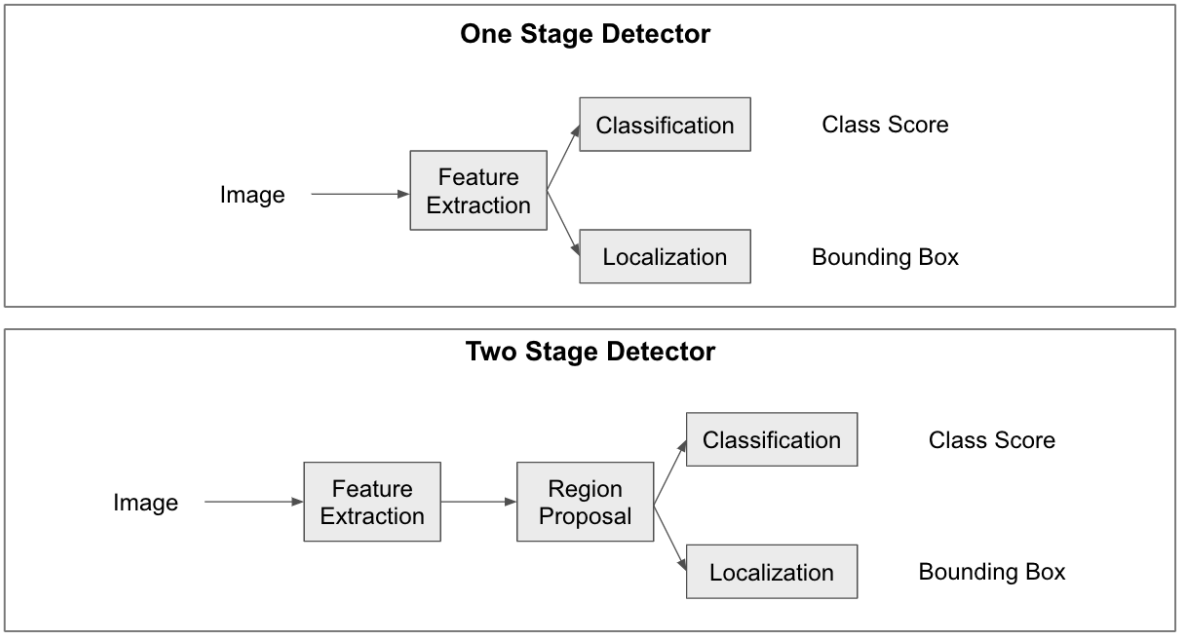

Figure 4.2: An illustration of one stage detectors and two stage detectors.

introduced a novel anchor-free paradigm for object detection. This is a departure from previous anchor-based methods for object detection that produce bounding boxes by refining predefined anchors tiled on the feature map of the input image. Instead, FCOS predicts a bounding box for each foreground pixel in an input image. This formulation of the object detection task avoids the pitfalls of anchor-based approaches. Namely, the complex computation and vast amount of hyperparameters anchor boxes introduce. Experimentally, FCOS outperformed previously proposed methods for one-stage object detection despite the simpler architecture.

RetinaNet: RetinaNet is a recently proposed one stage object detection method. Prior to its introduction, two-stage object detectors consistently outperformed one-stage detectors in terms of accuracy. RetinaNet demonstrated superior performance both in terms of accuracy and efficiency when compared to leading two-stage object detection methods. The backbone consists of a ResNet [8] and takes a standard anchor-based approach to object detection. The primary contribution of the method is introducing a novel loss function. This stemmed from the observation that the foreground-background class imbalance is the primary cause of the inferior performance of one-stage object detection methods. To address this, a Focal Loss is introduced that modifies the standard cross entropy loss function to down-weight the contribution of well-classified samples. Thus, the loss focuses on a small set of hard examples. This prevents the large number of easy negatives from dominating the loss term during training.

RetinaNet V2: In [32], the authors unify anchor-based and anchor-free detectors by demonstrating that the difference in the approaches lies in the definition of positive and negative training examples. Similar results are obtained if the same sampling procedure is used across the different classes of approaches. This result highlights the importance of the sampling procedure used to obtain positive and negative samples. To this end, the authors propose an Adaptive Training Sample Selection (ATSS) to automate the process selection process. Positive and negative samples are chosen according to the statistical characteristics of objects. ATSS signif-

icantly increases performance for both anchor-based and anchor-free detectors. When ATSS is applied to the RetinaNet, the resulting architecture is referred to as RetinaNetV2.

MobileNet V3: MobileNet V3 is the latest iteration of MobileNet architectures - a family of efficient one-stage object detection methods. As the name implies, MobileNet architectures are optimized for inference on mobile devices. In the first iteration of the MobileNet architecture, [10], the standard convolution is substituted for a newly introduced depth-wise separable convolution. The depth-wise separable convolution factorizes the standard convolution operation into a depth-wise convolution followed by a point-wise convolution. The depth-wise convolution filters information by applying a standard convolution with a separate learned kernel for each channel in the input feature map. The point-wise convolution consists of a 1x1 convolution that combines information across channels output from the outputs of the depth-wise convolution operation. Although the depth-wise separable convolution filters and combines information in the same way as the standard convolution operation, it is 8 to 9 times more efficient [9]. MobileNet V2 [23] extends MobileNet by optimizing the efficiency of inner blocks with novel inverted residuals and linear bottlenecks. Most recently, MobileNet V3 [9] introduced a novel architecture using complementary search techniques and a novel architecture design. In particular, hardware-aware network architecture search and the NetAdapt algorithm are employed.

4.3.2 Two Stage Detectors

Two-stage object detectors use region proposals to narrow the space of possible object locations. This allows the classification head to classify region proposals directly. Furthermore, the localization head has only to refine the coordinates of existing bounding boxes. The two-stage detector explored in the comparative vehicle detection study is Faster RCNN [21] and Faster RCNN V2 [16].

Faster R-CNN: Faster R-CNN is a popular method for two-stage object detection. R-CNN stands for Regional Convolutional Neural Networks, which alludes to the architecture's use of region proposals. Faster R-CNN outperforms the previously proposed R-CNN [7] and Fast R-CNN [6] both in terms of accuracy and efficiency. Following Fast R-CNN, the spatial coordinates of region proposals are mapped from the image to the feature map. Thus, the region proposal features are represented as the subset of the feature map specified in the region proposal coordinates. Region of Interest (RoI) pooling is used to map region proposals of various spatial dimensions to a uniform size that can be input into the classification and localization head. The primary contribution of the Faster R-CNN paper is to parameterize the region proposal algorithm as CNN. Prior works relied on algorithms such as Selective Search [30], which are computationally inefficient. Using a CNN as the region proposal network, the entire framework can be optimized jointly, enhancing the overall accuracy.

Faster RCNN V2: In [16], the authors propose a variety of enhancements for object detection approaches such as Faster R-CNN. The enhancements to the architecture include the addition of convolutional layers and batch normalization throughout the architecture. Namely, the Region of Interest (RoI) classification and box regression head are modified to have four convolutional layers followed by a fully connected layer in place of a two-layer MLP in the original architecture. Additionally, after the convolutional layers in these modules, batch normalization is included, along with the Feature Pyramid Network (FPN). Lastly, the Region

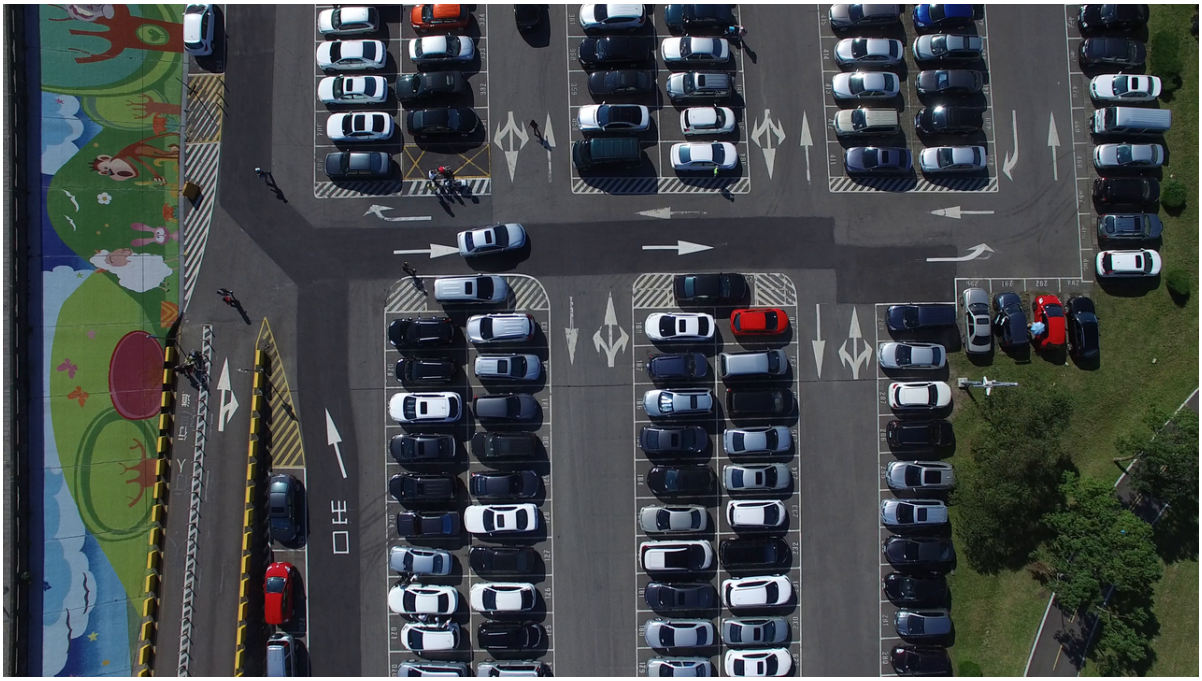

Figure 4.3: Image from the Car Parking Lot (CARPK) Dataset .

Proposal Network (RPN) is augmented to have two convolutional layers instead of one. Along with enhancements to the architecture, an updated training procedure is introduced. In particular, a data augmentation technique called large-scale jitter (LSJ) is used. LSJ is shown to prevent overfitting and improve model performance when training many epochs.

4.4 Experiment

In the experiments section, the aforementioned approaches SSD, MobileNet V3, RetinaNet, RetinaNet V2, FCOS, Faster RCNN and Faster RCNN V2 are compared on the vehicle detection task. First, the open-source dataset used for the comparative study is described. Subsequently, the setup used for training and evaluation is outlined, along with implementation details of the method. Lastly, the results of the experiments are discussed and the methods are ranked in terms of their vehicle detection accuracy.

4.4.1 Dataset

CARPK: CARPK is a dataset that contains aerial images of parking lots captured by a DJI Phantom 3 Professional drone. It was chosen due to its similarity to the SD dataset. The dataset contains images and annotations for object detection and vehicle counting. In total, 989 images with a resolution of 1280 x 720 were captured at approximately 40 meters in height. Across the images, there are 90,000 vehicle instances from 4 parking lots. Each vehicle instance is described by a bounding box that specifies its location and size within the context of the input image. An example of a raw image in the CARPK dataset can be found in 4.3.

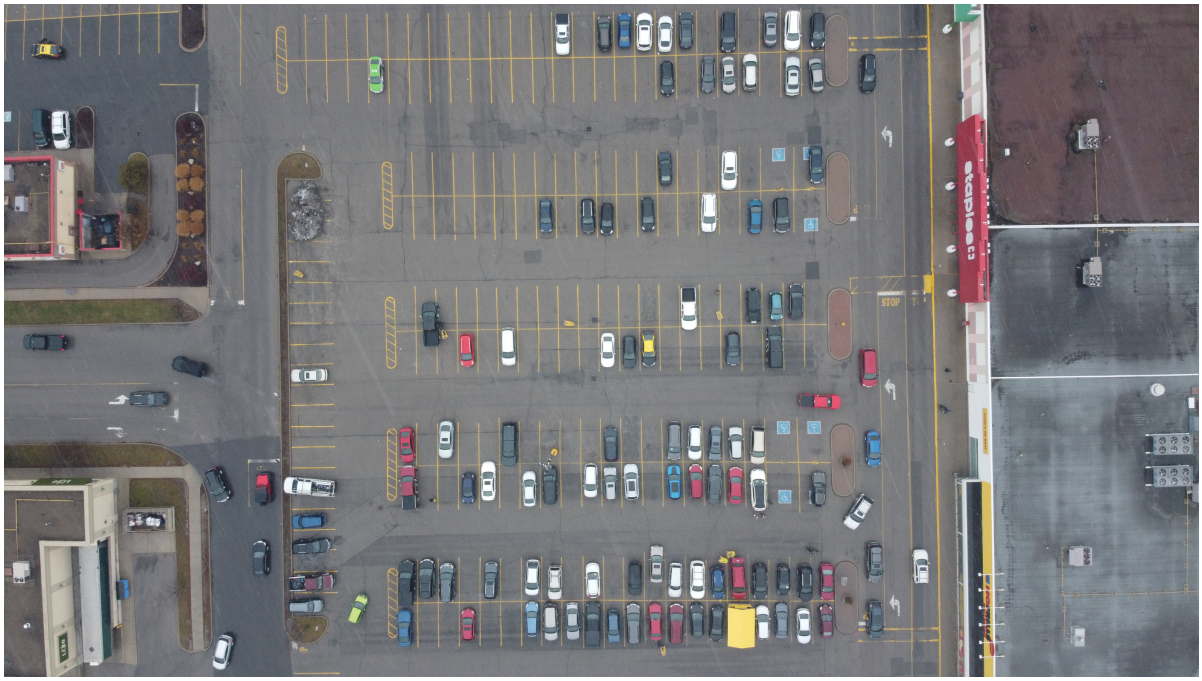

Figure 4.4: Image from the SkyDeploy (SD) Dataset .

SD: The SkyDeploy dataset (SD) is a private dataset consisting of 5000 aerial images of parking lots. The images were captured across 26 parking lots across Canada using the DJI Phantom 3 Profession drone. Each image has a resolution of 4000 x 2250 and was captured at a height ranging from 40 to 60 meters. At the time of the experiments, only a small subset of 250 images are annotated with bounding boxes. The bounding boxes specify the size and location of vehicle instances in an image. An example of a raw image in the SD dataset is available in 4.4.

4.4.2 Implementation Details

The experiments were implemented in Python using the PyTorch Framework and conducted on 1 NVIDIA Telsa P100 GPU. The architecture for each approach is consistent with that specified in the original papers. Each method is trained for 30 Epochs using stochastic gradient descent with a momentum of .9, weight decay of .0005 and a learning rate of 0.005. The models are initialized with pretrained weights from the COCO dataset [18]. Random seeds are used to strive for consistency in evaluation and reproducible experiments.

4.4.3 Experiment Details

CARPK: The CARPK dataset is divided into training (80%), validating (10%) and testing (10%) sets. Images are normalized using the mean and standard deviation of the Imagenet dataset [13]. The proposed object detection models are trained on the training set, while the validating set is used to determine stopping criteria. Lastly, the trained model is evaluated on

the testing set. The model with the highest performance on the validation set is used during the testing phase.

SD: Due to the limited amount of labels, a model could not be fine-tuned from scratch on the SD dataset. Accordingly, the experiments on the SD dataset involve evaluating the performance of models pretrained on the CARPK dataset. Because of the similarity of the dataset, it is reasonable to assume that model trained on the CARPK dataset will generalize to the dataset. Images in the dataset are normalized using the mean and standard deviation of the Imagenet dataset [13]. Images are also sliced into subimages in the preprocessing stage due to their high resolution. Precisely, images are sliced into eight subimages of size 1125 x 1000.

Using the entire preprocessed dataset, each method is evaluated with the parameters that yield the highest performance on the CARPK dataset across epochs.

4.4.4 Evaluation Metrics

On both datasets, the models are evaluated using the Average Precision (AP) and Average Recall (AR). These metrics were chosen because of their popularity in the object detection literature. To start, it is helpful to outline the set of possible outcomes when making (or not making) a prediction for an object. In particular, the relevant outcomes are true positives, false positives and false negatives. True positives describe the case in which an object was detected correctly. In contrast, false negatives and false positives correspond to failing to detect an existing object and incorrectly detecting an object that does not exist. The Intersection over Union (IoU) metric is used to determine whether a predicted bounding box is a true positive or a false positive. The IoU is a normalized measure of overlap between the predicted bounding box and ground truth. Typically, an IoU of .5 or.7 is used as a threshold. With this information, precision and recall are defined as follows:

$$precision = \frac{tp}{tp + fp} \tag{4.2}$$

$$recall = \frac{tp}{tp + fn} \tag{4.3}$$

where tp, fp and fn represent true positives, false positives and false negatives, respectively. Intuitively, precision measures the proportion of positive predictions that are correct. Alternatively, recall measures the proportion of correct positive predictions over all predictions made. There is a fundamental trade-off between precision and recall. Different prediction score thresholds yield different precision and recall values. The precision-recall curve encodes the trade-off between precision and recall for different score thresholds. The average precision (AP) is calculated as the area under the precision-recall curve. In contrast, The average recall (AR) is calculated as the mean recall value across IoU thresholds. AP and AR range from 0 to 1, where a higher score represents better performance.

4.4.5 Results

The AP and AR for each method are reported in Table 4.1 for both the CARPK and SD datasets. Due to their similarity, the relative ranking of methods remains relatively consistent across

datasets. Specifically, Faster RCNN achieves the highest AR and AP on CARPK and SD. Faster RCNN V2 and FCOS achieve slightly lower performance, followed by MobileNet V3 and RetinaNet. The worst-performing methods are SSD and RetinaNet V2 which lag in AR and AP by a considerable margin.

In both cases, RetinaNet and Faster RCNN outperform their newer counterparts RetinaNetV2 and Faster RCNN V2. This result highlights the fact that the modifications to the architecture are not beneficial in the case of the vehicle detection task. This can be attributed to underlying differences between the generic object detection and vehicle detection tasks. The vehicle detection task differs from the generic object detection task in that the object instances are smaller and more numerous. As such, the introduction of additional convolutional layers, as proposed in Faster RCNN V2 and RetinaNet V2 may result in the degradation of performance on the vehicle detection task. This is because adding convolutional layers reduces the spatial resolution of feature maps, making it difficult to detect small object instances. A future study is warranted to discover the exact reason for the performance degradation of the vehicle detection task with the Faster RCNN V2 and RetinaNet V2 architectures.

Figure 4.5 reports the training and validation performance across epochs. In general, the BCE loss of methods decreases across epochs for the training and validation set. This reflects that the models can iteratively improve their performance without overfitting the training data. The train and validation loss curves look similar for all methods except for SSD. The SSD train and validation loss curves start with large values but eventually converge to those similar to other methods. One of the potential reasons for this discrepancy is that the SSD architecture uses vgg16 as the backbone, whereas other methods use the ResNet backbone. The ResNet backbone has been shown to outperform the vgg16 backbone on a wide range of computer vision tasks, so it is conceivable that the SSD method would incur higher losses earlier in the training process.

Visual results from the two strongest performing methods, Faster RCNN, are available in Figure 4.6 and Figure 4.8. This is contrasted with visual results of the second strongest performing method, Faster RCNN V2, in Figures 4.7 and 4.9. Faster RCNN and Faster RCNN V2 demonstrate accurate vehicle detection across the CARKPK and SD datasets. In each case, every car is detected, and only a few false positives exist. These visual results support the quantitative results that both Faster RCNN and Faster RCNN V2 are effective methods for vehicle detection.

4.4.6 Conclusion

This chapter presented a comparative study of object detection methods applied to the vehicle detection task. Seven state-of-the-art methods were benchmarked in a series of experiments. The experiments involved evaluating the approaches on both an open-source dataset (CARPK) and a private dataset (SD). In the CARPK experiments, the models were finetuned and evaluated on a held-out test set. In the subsequent experiment, the pretrained models on the CARPK dataset are evaluated on the SD dataset. In both cases, Faster RCNN achieved the best results in terms of AP and AR. This indicates that the method is most well-suited for the vehicle detection task.

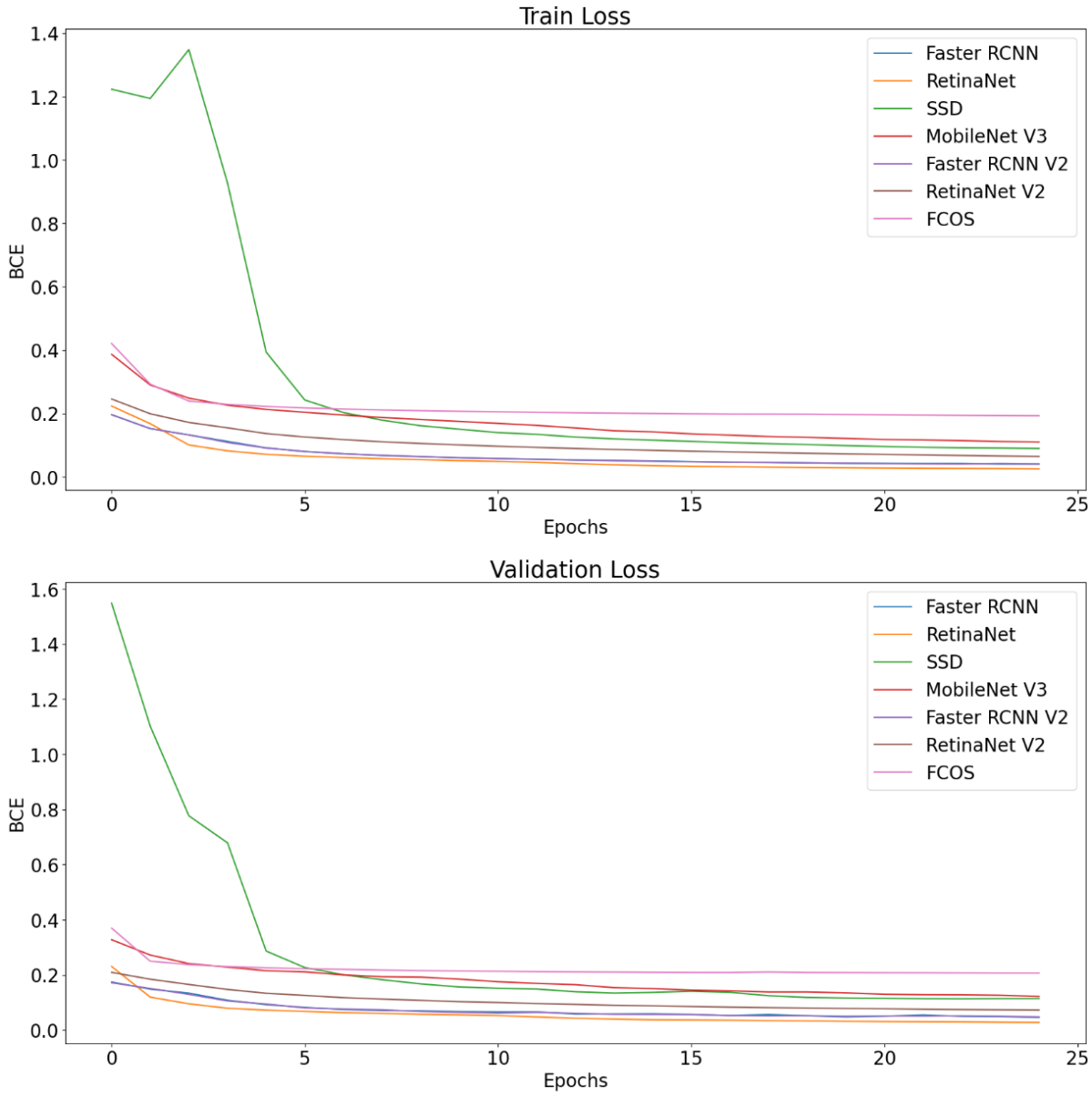

Figure 4.5: Binary cross entropy (BCE) loss for training set (top) and validation set (bottom) across epochs.

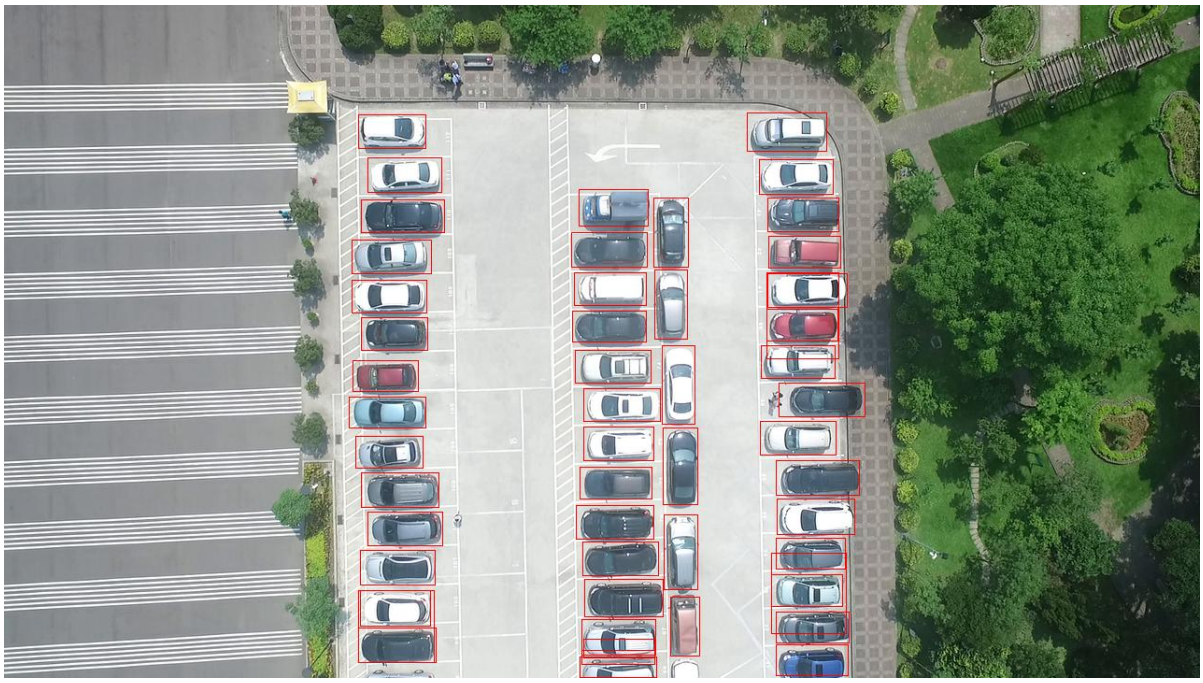

Figure 4.6: Visual Results for the Faster RCNN method on the CARPK Dataset

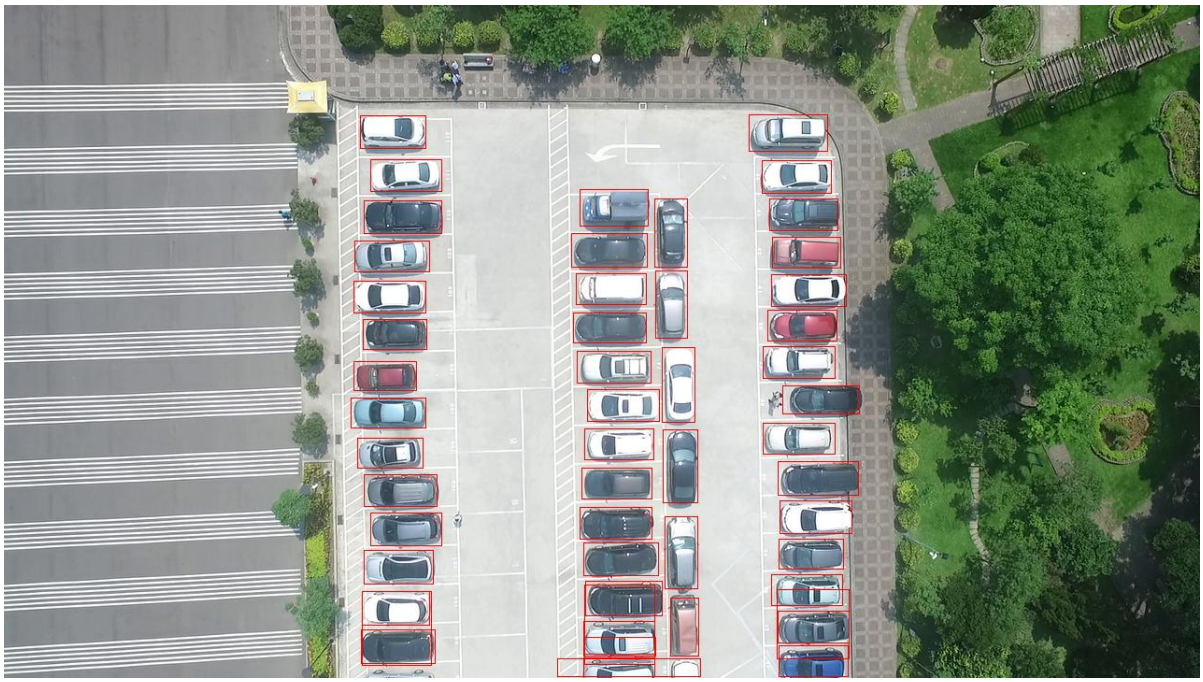

Figure 4.7: Visual Results for the Faster RCNN V2 method on the CARPK dataset

Figure 4.8: Visual Results for the Faster RCNN method on the SD dataset

Figure 4.9: Visual Results for the Faster RCNN V2 method on the SD dataset

	CARPK		SD	
Model	AP	AR	AP	AR
Faster RCNN	0.7792	0.8298	0.7195	0.7589
Faster RCNN V2	0.7689	0.8226	0.7147	0.7568
RetinaNet	0.7356	0.7882	0.6573	0.6925
RetinaNet V2	0.6429	0.7058	0.6043	0.6417
SSD	0.6395	0.7141	0.5981	0.6340
FCOS	0.7685	0.8256	0.7098	0.7435
MobileNet V3	0.7415	0.7479	0.7045	0.7134

Table 4.1: Average Precision and Average Recall on test set for each approach

4.5 Towards Heatmap Generation

The proposed research is a significant step towards an end-to-end framework to generate parking lot turnover heatmaps. Specifically, this work focused on the Image Stitching and Vehicle Detection component of the framework. The Heatmap Generation component is left for future work. The Heatmap Generation step is the last component in the pipeline. It involves registering mosaics across time intervals and summing the number of vehicles that are parked in each stall. The resulting output is a heatmap showing how many times vehicles parked in various stalls throughout the day.

In order to provide some initial direction on how to realize the proposed framework, the end-to-end pipeline is discussed in this section. Given a sequence of image sets that span a parking lot at different intervals, the objective is to generate a parking lot turnover heatmap that encodes the frequency at each parking stall used. The first step in the pipeline is to generate the mosaic for each time interval for the parking lot. The pose of the virtual camera used to generate the mosaics should be consistent across intervals to help with the consistency across mosaics. An illustration of this first step in the pipeline is depicted in Figure 4.10

In the subsequent vehicle detection stage, each mosaic is sliced into lower-resolution sub-images before being fed into the vehicle detection model. This ensures that the vehicle detection can efficiently perform inference without excessively downsampling the mosaic. Each image slice is fed to the vehicle detection model to get the local of the vehicle instances. The bounding box coordinates of the image slices can be mapped back onto the corresponding mosaics by considering the size and index of the image slice. An example of the resulting mosaic annotated with detected vehicle instances is available in Figure 4.11.

In the final step of the pipeline, heatmap generation, the mosaics with vehicle annotations are used to generate parking lot turnover heatmaps. This involves registering each mosaic with a CAD blueprint of the parking lot that includes the parking stalls. Once a mosaic is registered to the blueprint, each detected vehicle instance is assigned to the parking stall in the blueprint, which has the highest overlap. This process is repeated for each mosaic, and the count of all parking stall are computed. This process is depicted in 4.12. The final output of the heatmap generation phase is a parking lot turnover heatmap that is shown in 4.13

Figure 4.10: An illustration of the Image Stitching step of the pipeline.

Figure 4.11: An illustration of the Vehicle Detection step of the pipeline.

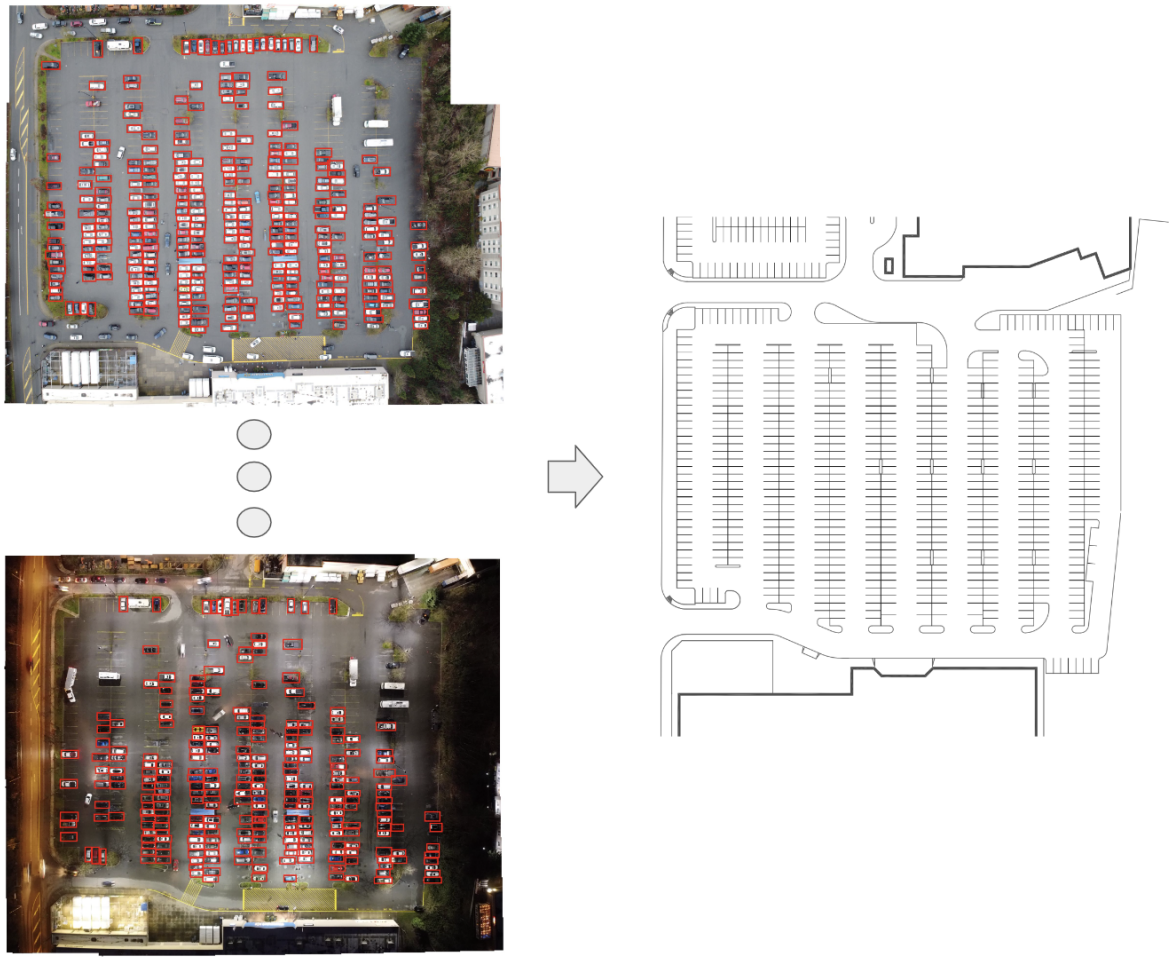

Figure 4.12: An illustration of the Heatmap Generation step of the pipeline.

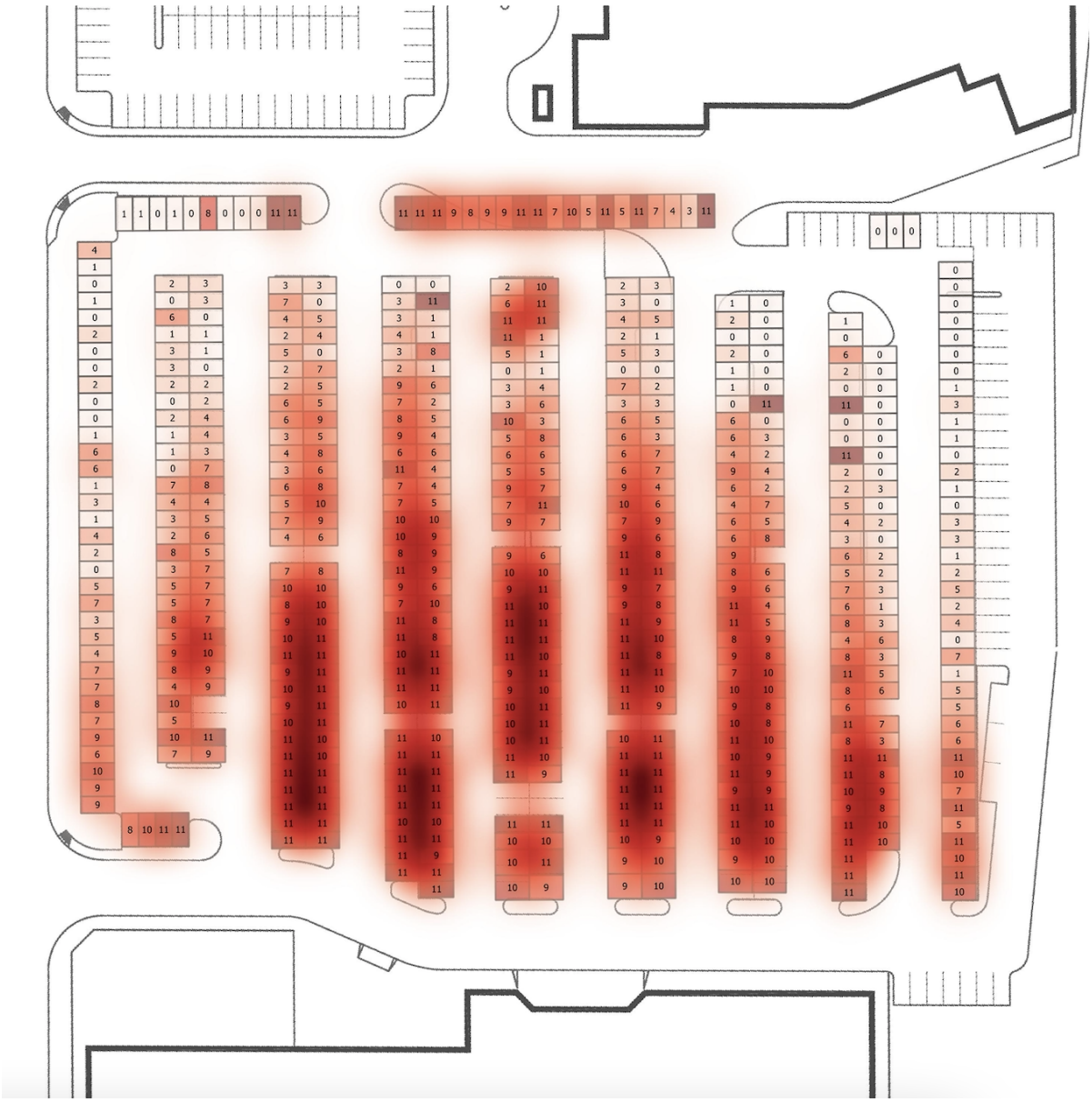

Figure 4.13: An example of the Parking Lot Turnover Heatmap.

Chapter 5

Conclusion

5.1 Conclusion and Discussion

Advances in aerial imagery and computer vision have enabled various commercial applications in urban planning. Specifically, large open-concept retail centres could benefit from systems to analyze parking lot usage. This would allow retail centers to optimize the design of parking lots to efficiently utilize space and eliminate the parking oversupply problem. This research works towards developing a framework to generate parking lot turnover heatmaps that encode the number of vehicles parked in each stall during a given day. The proposed framework is a pipeline of components that maps a sequence of image sets spanning a parking lot at different time intervals to a parking lot turnover heatmap. In particular, the framework consists of three components: Image Stitching, Vehicle Detection and Heatmap Generation. This thesis focuses on Image Stitching and Vehicle Detection, while Heatmap Generation is left as future work.

Chapter 3 provides an overview of the algorithm to map a set of aerial images spanning a parking lot to a mosaic. This involves first determining the pairwise transformations among images in the set. Next, a global alignment procedure is used to optimize the pose parameters of images in the mosaic to approximate the pairwise transformations. A novel algorithm was introduced for initializing images using the pose of the drone and characteristics of its camera. In experiments, the optimal set of hyperparameters for the image stitching algorithm were determined. Additionally, an ablation study was used to show the efficacy of the novel method for image initializations that was proposed.

The Vehicle Detection component of the framework is described in Chapter 4. Vehicle detection is an instance of the object detection task that involves localizing and classifying vehicle instances given input images. A comparative study of state-of-the-art object detection approaches is presented. In particular, seven methods are compared on two benchmark datasets. These datasets contain aerial images of parking lots with corresponding annotations of vehicle instances. The purpose of the experiments was to determine the best method for vehicle detection. This is a novel comparative study that is useful for both researchers and practitioners. As a result of the experiments performed, Faster RCNN achieved the best results in terms of precision and recall.

5.2 Limitations

Although the proposed research includes beneficial novel contributions, there are various limitations. Specifically, in Chapter 2, the evaluation of image stitching algorithms is complex because the ground truth labels do not exist. It is also nontrivial to generate them, even using human annotation. To surmount this, the mean squared link loss is used to compare the algorithm with different sets of hyperparameters. In practice, this evaluation technique worked well and matched the visual results obtained.

There are also several limitations to the image stitching algorithm itself. Namely, the relative transformation we solve using SIFT and RANSAC is prone to false correspondences between keypoints that may yield inaccurate relative transformations. Furthermore, the image initialization algorithm is sensitive to sensor noise. Errors in the drone's location, orientation and intrinsic parameters may contribute to inaccurate initializations. Lastly, partial affine transformations are used for the mappings between image coordinates. This class of mappings is not expressive enough to accurately represent some transformations.

In Chapter 3, the vehicle detection experiments face a few minor limitations. First, some popular object detection methods were not included in the comparative study. This is because some object detection methods do not have proper open-source implementations available. This is a reasonable limitation, given that a relatively large set of vehicle detection methods are explored. Another limitation of the Vehicle Detection experiments is that the private dataset only contained a small number of annotations. To overcome this, both a private and open source dataset are used for evaluation.

5.3 Future Work

This work is a significant step towards an end-to-end framework for parking lot occupancy assessments. The research focuses on developing the Image Stitching and Vehicle Detection components of the framework. The future work includes realizing the Heatmap Generation component. Section 4.5 provides a high-level discussion of Heatmap Generation and its integration with existing components. Further work must be done to flush out the Heatmap Generation component and the design of the end-to-end pipeline. Once the framework is complete, it would be beneficial to build an application around it to enhance its usability.

Bibliography

[1] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[3] Guowei Cai, Ben M. Chen, and Tong Heng Lee. *Coordinate Systems and Transformations*, pages 23–34. Springer London, London, 2011.

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[5] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[6] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[8] 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.

[9] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019.

[10] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[11] Meng-Ru Hsieh, Yen-Liang Lin, and Winston H Hsu. Drone-based object counting by spatially regularized regional proposal network. In *Proceedings of the IEEE international conference on computer vision*, pages 4145–4153, 2017.

[12] Ersin Kilic and Serkan Ozturk. An accurate car counting in aerial images based on convolutional neural networks. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–10, 2021.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[14] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999.

[15] Qingpeng Li, Lichao Mou, Qizhi Xu, Yun Zhang, and Xiao Xiang Zhu. R^3-net: A deep network for multi-oriented vehicle detection in aerial images and videos. *arXiv preprint arXiv:1808.05560*, 2018.

[16] Yanghao Li, Saining Xie, Xinlei Chen, Piotr Dollar, Kaiming He, and Ross Girshick. Benchmarking detection transfer learning with vision transformers. *arXiv preprint arXiv:2111.11429*, 2021.

[17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[19] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

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

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[22] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[23] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[24] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] Lars Sommer, Tobias Schuchert, and Jürgen Beyerer. Comprehensive analysis of deep learning-based vehicle detection in aerial images. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(9):2733–2747, 2018.

[27] Richard Szeliski. Computer vision algorithms and applications, 2011.

[28] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019.

[29] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[30] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[31] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.

[32] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9759–9768, 2020.

Curriculum Vitae

Name: John Jewell

Post-Secondary: Bachelor of Computer Science
2016 - 2020
University of Western Ontario
London, ON

Experiences: Associate Applied Machine Learning Specialist
Vector Institute
Toronto, ON
Jan 2022 - Present

Applied Machine Learning Intern
Vector Institute
Toronto, ON
Sep 2020 - Dec 2020

Software Developer Intern
Royal Bank of Canada
Toronto, ON
May 2019 - Aug 2019

Publications:

1. Jewell, John Taylor, Vahid Reza Khazaie, and Yalda Mohsenzadeh. ”One-Class Learned Encoder-Decoder Network with Adversarial Context Masking for Novelty Detection.” Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2022.

2. Khazaie, Vahid Reza, Anthony Wong, John Taylor Jewell, and Yalda Mohsenzadeh. ”Anomaly Detection with Adversarially Learned Perturbations of Latent Space.” 2022 19th Conference on Robots and Vision (CRV). IEEE, 2022.