TÉCNICO LISBOA

City Cam: Similarity based classification of vehicle count time series

Tomás Matos Fernandes da Cunha Cordovil

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. João Paulo Salgado Arriscado Costeira
Prof. Carlos Jorge Andrade Mariz Santiago

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. João Paulo Salgado Arriscado Costeira
Member of the Committee: Prof. Pedro Manuel Quintas Aguiar

November 2021

Affidavit

I hereby declare that the present document is an original work authored by me and that it abides to the requirements of the code of conduct and good practices of the University of Lisbon.

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do código de conduta e boas prácticas da Universidade de Lisboa.

Acknowledgments

To my advisor, without whom the completion of this project would not have been possible. Both for his advisory and motivational support I have nothing but appreciation. But especially for his patience, thank you.

To my parents, as this thesis probably meant more for them than for me. Their presence and insistence pushed me forward especially in the most critical moments. I know they worked hard to make my life simpler during this stage and all others. Logistically and emotionally, this would not have been possible without them.

To my girlfriend Beatriz who was there to hear me ramble about nonsense and witness my full range of emotions. For her caring patience and effort in preserving my sanity and for the serenity and energy she provided in the all times she kept me company while I was working.

Abstract

Classification of traffic patterns based on images. With a rise of utilization of vehicles in an urban context it also becomes increasingly important to be able to understand how traffic density varies in time.

Traffic surveillance cameras are a widespread and relatively cheap tool that can be used to analyse these traffic patterns. Image processing in a traffic context presents its own challenges.

We propose a method of converting the sequence of images acquired by traffic cameras into traffic state classification. The main contribution of this work is the application of Sparse Subspace Clustering algorithm to describe the traffic flow.

Vehicle counts are first extracted from the images. Vehicle density estimation through deep learning is employed to avoid using detection methods. The sequence of vehicle counts, segmented in same sized intervals, is fed to an optimization program which returns a sparse subspace representation of the data. This alternative representation allows the definition of a similarity measure. Considering a segment of vehicle counts as a node, and the similarity between two nodes being the weight of an edge connecting them, the data can be viewed as a graph. We apply spectral clustering to this graph to obtain classifications. The error values returned in the optimization program are indicators of how unique a segment is, easing the detection of anomalies in traffic density patterns.

There is no metric to analyze the results of this process, so a qualitative analysis is performed to evaluate the obtained results.

Keywords: traffic, image, density estimation, sparse subspace clustering, subspace representation, classification

Resumo Analítico

Classificação de padrões de trânsito baseado em imagens. Com o aumento da utilização de veículos também se torna cada vez mais importante compreender como a densidade de trânsito varia com o tempo.

Câmaras são ferramentas bastante difundidas e baratas que podem ser usadas com o intuito de analisar estes padrões de trânsito. Mas o processamento de imagem, no contexto de trânsito, tem dificuldades associadas. A principal contribuição deste trabalho é a aplicação do algoritmo de Sparse Subspace Clustering para caracterização do fluxo de veículos.

Propomos um método para converter sequências de imagens capturadas por câmaras em classificações de estado do trânsito.

A contagem de veículos é extraída da imagem. Usa-se ”deep learning” para obter estimativas de densidade de veículos numa imagem. A sequência de contagem de veículos, segmentada em partes de tamanho igual, é processada por um programa de optimização que devolve uma representação esparsa em sub-espaços dos dados. Esta representação dos dados permite a definição de uma métrica de semelhança. Considerando um segmento de contagens de veículos como sendo um nodo, e a semelhança entre dois nodos sendo o peso da aresta que os liga, a nova representação pode ser usada como um grafo. Aplicamos spectral clustering a este grafo de modo a obter classificações. O erro retornado no programa de optimização é indicador de quão singular um segmento é, facilitando detecção de anomalias em padrões de densidade de trânsito.

Não havendo métricas para análise dos resultados deste processo, fez-se uma análise qualitativa para poder avaliar os resultados obtidos.

Palavras chave: trânsito, imagem, estimação de densidade, sparse subspace clustering, sub-espaço, classificação

Contents

Acknowledgments v
Abstract vii
Resumo Analítico viii
List of Tables xi
List of Figures xiii

1 Introduction **1**
1.1 Objectives 1
1.2 Motivation 1
1.3 Work Highlights 2

2 Related Work **5**
2.1 Counting 5
2.1.1 Detection Based Methods 5
2.1.2 Regression Based Methods 6
2.1.3 Deep Learning 6
2.2 Clustering 7

3 Counting Framework **10**
3.1 Introduction 10
3.2 The U-Net 11
3.3 Analysis 13

4 Clustering **18**
4.1 Descriptor 18
4.2 Sparse Subspace Representation 20
4.3 Spectral Clustering 22
4.4 Single Camera Data Analysis 23

5 Application: a Case study of Tallinn **29**
5.1 The Cam Network 29
5.2 Processing 31
5.3 Data Processing Results 37

5.4 Sparse Spectral Clustering Results 40

6 Conclusions **49**

6.1 Achievements 49

6.2 Future Work 50

Bibliography **51**

List of Tables

3.1 Results of inference in training dataset 13

List of Figures

1.1 Main steps of this thesis 2

3.1 Example of annotated image and its corresponding target ground truth 11
3.2 U-Net architecture diagram 11
3.3 Example of U-Net inference in New York 1 13
3.4 Example of U-Net inference in New York 2 14
3.5 Example of U-Net inference in New York 3 15
3.6 Example of U-Net inference in New York 4 15
3.7 Example of U-Net inference in New York 5 15
3.8 Example of vehicle counts time series 16

4.1 Results of applying linear interpolation to the time series of cam031 19
4.2 Results of applying a moving average to interpolated data of cam031 19
4.3 Daily vehicle count descriptor of cam031 19
4.4 Transforming cam031's vehicle count descriptor into a similarity matrix 20
4.5 Reconstruction of cam031's 04/08/2020 using related days and reconstruction error . . . 21
4.6 Reconstruction of cam031's 04/08/2020 result and error 21
4.7 Classification with spectral clustering of cam031 23
4.8 Comparing moving average of 15min and 60min with cam031 23
4.9 Comparing daily vehicle count descriptors of a street camera (cam031) and a parking camera (cam017) 24
4.10 Analysing clustering classification of cam031 24
4.11 Analysing cam005 data obtained in March 25
4.12 Analysing pre and post lockdown traffic in cam005 25
4.13 Daily vehicle counts descriptor of cam005 26
4.14 Analysing clustering classification of cam005 in March 26

5.1 Grid showing cameras in Tallinn 29
5.2 Map of Tallinn with location of cameras 30
5.3 Example images obtained from the selected cameras 31
5.4 Diagram with sequence of processing steps 32
5.5 Example of inference on traffic images for four distinct cameras 32

5.6 Comparing the effects of using cubic spline and linear spline 33
5.7 Moving average using 3 bandwidths 34
5.8 Plotting moving averages and specifying day of the week 35
5.9 Moving average and sampled moving average 35
5.10 Descriptors of August and March for cam011 36
5.11 Matrices C and W of cam 112 in August 36
5.12 Example counting of cam 017 37
5.13 Moving averages of two parking cameras 38
5.14 Moving averages of three street cameras 39
5.15 Comparison of workday and weekend day's 39
5.16 Descriptors of August and March for 3 cams 40
5.17 Sparse subspace representation of cameras 005 and 017 41
5.18 Sparse subspace representation of cameras 005 and 017 in March 42
5.19 Joint descriptors of two cameras in August 42
5.20 Similarity matrices of cam005 and cam017 in August 43
5.21 Similarity matrices of cam005 and cam017 in March 43
5.22 Joint similarity matrices of two cameras in August 44
5.23 cam031 day 26's most similar days and reconstruction error 44
5.24 cam005 day 13's most similar days and reconstruction error 45
5.25 Classification of cam005, August, two groups 46
5.26 Joint classification of cam005 and cam031, August, two groups 47
5.27 Joint classification of cam005 and cam031 in similarity matrix, August, two groups, using similarity matrix 47

Chapter 1

Introduction

1.1 Objectives

Video cameras are a widespread tool for traffic monitoring by being mounted in roads, city streets, intersections and parking lots. As they are active at all times, these cameras generate a lot of data through the continuous acquisition of images. These images can be processed to obtain key indicators such as the number of cars per frame, or traffic density which is defined as the number of vehicles per unit of area of the road. Having a large record of these densities can enable the extraction of traffic patterns. Moreover, if a recurrent traffic density pattern can be found, anomalies can be more easily identified. In itself, the identification of traffic anomalies can help to infer if and how the anomaly will affect traffic in the other points in space and allow a faster response.

This thesis's objective is to process still image data to obtain both vehicle count and traffic density estimations which in turn are used to generate traffic descriptors. Using the obtained descriptors, a Sparse Subspace Clustering algorithm is applied ti them. The algorithm can be broken down into two steps: a sparse subspace representation method is used to reduce the dimensionality of the cumulative information obtained from images, followed by a clustering technique to classify time intervals of real data in terms of its traffic characteristics. Finally, analyse this clustering to find similarities between time segments of a single or multiple cameras. More specifically, we make use of Tallinn's camera network as a data source. As inferring vehicle counts from images is a problem that has already been investigated in the past, the main contribution of this work is the application of the Sparse Subspace Clustering algorithm to obtain a classification of vehicle count time series.

1.2 Motivation

With the present and assumed future rapid growth of inhabitants of urban areas, the number of road vehicles in these areas also witnesses a substantial increase. A direct result of this is a rise in the frequency of occurrences of traffic congestion situations. Since mobility in urban scenes is becoming more and more relevant because of its impact in the economy and the environment, being able to better

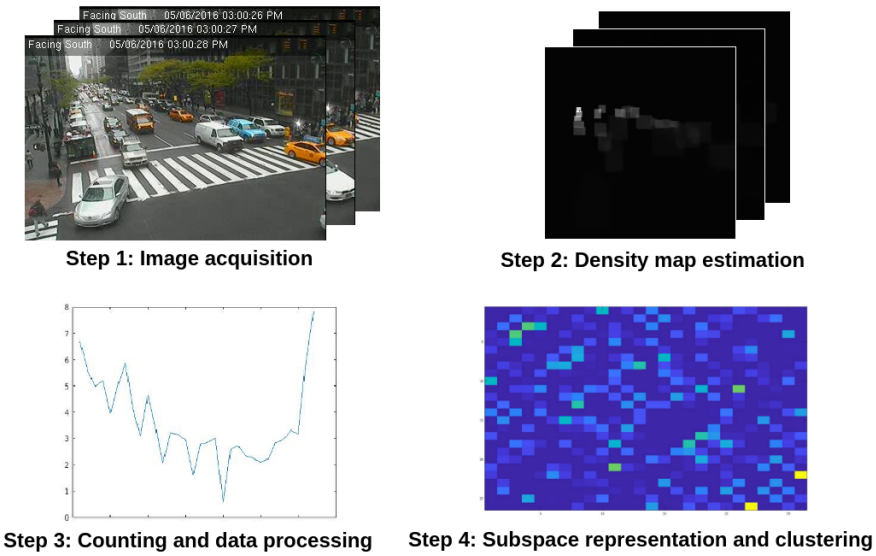

Figure 1.1: Main thesis steps.

characterize and predict traffic can be a valuable tool to improve circulation. Being able to classify traffic state in several points in a city can lead to a faster identification of unusual behaviours and help both individuals and governments to react faster to them. Traffic statistics can also help in finding interrelation between fixed locations if a connection can be found between their data collections.

Traffic characterization can have positive implications in congestion management as it can help in route planning, deciding best departure time, trip duration estimation and infrastructure management. To be able to perform this, traffic data acquisition is necessary, and finding a reliable source is a problem on its own. It is possible to obtain information regarding traffic with many devices, for example through loop detectors, piezoelectric sensors, radars, GPS devices ([1]) and cameras. Cameras have seen a peak in interest as it is the option which has yielded the best results in detecting cars. Moreover, camera based counting is a tool which has seen applications in other fields such as crowd counting, agricultural control and medical imagery.

Difficulties in obtaining vehicle counts appear when we try to accomplish this task based on typical surveillance videos as they can have low frame-rate, large perspective, changing background and point of views in their acquisition.

1.3 Work Highlights

This thesis's main objective is to classify traffic time intervals using a Sparse Subspace Clustering algorithm. The original data are images obtained from traffic surveillance cameras. Vehicle counts are extracted from images using a neural network. Its input are images and its output are density maps with a sum of pixel values equal to the number of vehicles contained in it. Pre-processing is applied to the data in order to obtain the desired format for the following steps. Then, an alternate representation of the series of vehicle counts is generated through the use of an optimization program. This alternate

representation consists in expressing a sequence of vehicle counts as a sparse linear combination of other sequences of vehicle counts. Thus, each of those sequences can be reconstructed with that linear combination added to a reconstruction error. With this alternative representation of the data, a similarity metric is obtained. The collection of similarity values between pairs of vehicle count sequences is in fact a graph, where each sequence of vehicle counts is a node and the similarity between two nodes is the weight of the edge connecting them. With the data now converted to a graph, it is possible to apply graph clustering methods which will classify time intervals of traffic.

The rest of this thesis will be organized into five chapters. In Chapter 2, we will introduce the state of the art. Chapter 3 contains a description of how vehicle counts are extracted from images. Chapter 4 explains how vehicle counts are classified. Chapter 5 is a real-life application of the proposed method, including a description of the acquisition of data. Chapter 6 describes the achievements of this thesis and possible future work.

Chapter 2

Related Work

In this chapter we review literature related to the two main parts of this project, namely counting in computer vision and clustering. In counting literature we will focus on image segmentation and density estimation. We will also provide a general view of the deep learning methods used in these contexts. In clustering we will introduce subspace representation methods and relevant clustering techniques.

2.1 Counting

Counting, in a computer vision context, consists in the estimation of the number of instances of an object in an image ([2]). Counting has a wide range of applications, from counting cars in a traffic surveillance context ([3]), to microscopy cells in medical imagery ([4]), trees in an agricultural control ([5]), crowd counting, and others.
With such a wide spectrum of possible applications, many methods have been developed in the course of the years to solve this problem. We can divide these methods into two categories: detection based methods and regression based methods ([6] [7]).

2.1.1 Detection Based Methods

Detection based methods rely on locating desired objects in an image to directly obtain an estimate number of their instances. This class encompasses morphological operations, transforms, image enhancement techniques, edge and contour detection and segmentation methods.
Segmentation itself is a task which consists in the grouping or classification of all pixels of an image. There are two types of image segmentation: semantic segmentation and instance segmentation. The main difference between these two is that in semantic segmentation, pixel classification is conducted immediately indicating if this pixel belongs to an object of a specific class without specifying boundaries between objects of the same class. While in instance segmentation, instances of that class are identified separately thus enabling the detection of every object by finding all its pixels in that image. This method (instance segmentation) will delineate the objects boundaries, allowing for a counting and localization of all objects([8], [6]).

Instance segmentation is used often in surveillance, crowd segmentation and cell segmentation ([9], [10]).

One of the main advantages of this method is that it gives spatial information on the objects, this will permit, for example, the tracking an instance of an object between frames or maybe even an estimation of real life distances. However, detection is a very hard task (there still is work being developed in this area), plus, it does not deal very well with occlusion or low resolution ([11]) and tracking becomes impossible when image acquisition has a low frame rate.

2.1.2 Regression Based Methods

Contrary to detection, regression is a method which makes use of image features, such as color histograms, texture or foreground pixel area ([12], [13]), to directly obtain an estimate of the number of instances of an object in an image. It is most useful in tasks where spatial information is not needed (as no object detection is used) and the input image has low quality or is overcrowded ([14]), such as in crowd or microscopy cell counting.

Based on this concept, the authors of [6] proposed a new method for counting based on regression: image density estimation which still retains spatial information from the original image. The authors do so by procuring a density function F which, when integrated over the entire image, outputs a density map representing the density of instances of objects in each pixel and whose integral outputs the count in that image.

2.1.3 Deep Learning

In recent years, deep learning methods have seen a rapid increase in their usage, and have been applied in a variety of contexts including image processing. A special interest in Convolutional Neural Networks (CNN or Convnets) has peaked as these have yielded excellent results in this field ([15]). They have been used for image classification ([16], [17]), detection ([18], [19]), segmentation ([20], [21], [22]) and count regression ([5]). Many variations of CNN have since been proposed, such as the You Only Look Once (YOLO) [23], U-Net ([24]) and others.

In the same microscopy cell counting context as mentioned in section 2.1.3, the authors of [4] consider a training set composed by a cell microscopy image and its corresponding density map and make use of a Fully Convolutional Neural Network (FCNN) to solve the density estimation problem. Their solution produced fast and accurate results.

Another example is the You Only Look Once or YOLO network. YOLO is a region proposal CNN used for object detection and classification which treats detection as a regression problem. This architecture has been applied to traffic counting ([25], [3], [26]) with very promising results. It works by dividing an image into a grid and proposing a classification and a bounding box for each cell grid. Although it is very fast, it has problems when dealing with low resolution, small objects or occlusion.

U-Net is a fully convolutional neural network commonly used for image segmentation. It follows the basic form of an auto encoder: a symmetrical architecture composed of a contracting side and an expanding

side. It has the particularity of forwarding the result of the convolution blocks from the contracting side to the expanding side. This forwarding is referred to as skips. Although it is mostly used in a biomedical context, it has also been used for vehicle counting tasks.
Another CNN architecture used for vehicle counting is proposed in [27]. Using a different method, the network proposed in that paper still combines features extracted from shallow layers (appearance features) with the ones extracted from deeper layers (semantic features). The authors of [27] affirm this helps producing denser feature maps thus solving the problem of reduced spatial resolution.

2.2 Clustering

Clustering is the task of grouping a set of objects in such a way as to maximize or minimize a specified metric. Objects inside the same cluster tend to be more similar (according to the predefined metric) between them than objects from different clusters. Clustering is crucial when trying to detect underlying patterns in data.

Many techniques have been developed over the years and they are used in a wide variety of research fields, including traffic analysis. In this context, they can be implemented with different objectives, depending on the input data, such as clustering the trajectories of vehicles in a sequence of images to better understand traffic flow in an intersection (see [28]). Another example of application is [29] where the authors try to classify traffic states of a single location, whereas in [30], the authors analyze data from a large network of road segments and try to identify spatial traffic prediction patterns. In sum, there is a multiplicity of ways of applying clustering techniques in a traffic analysis context. The methods which seem to appear recurrently are k-means and fuzzy C-means.

K-means is an unsupervised clustering method which aims to partition a population into k clusters such that, each point belongs to the one cluster whose centroid it is closest to. It minimizes within cluster variance, or inertia, by optimizing equation 2.1.

$$J(X, C) = \sum_{i=1}^{n} \sum_{j=1}^{K} m_{i,j} ||x_i - c_j||^2 \tag{2.1}$$

This is accomplished by iteratively calculating the mean of all clusters and attributing to a cluster, all points which are closer to it. After changing the composition of the clusters, its new mean is recalculated, and the process continues [31]. Although this method does not necessarily converge to a global optima, it always converges to a local optima. It is a non-deterministic method which depends on the initialization. Here, initialization is the process of defining the clusters before starting the iteration. Two popular initialization methods are for example: defining initial clusters as k samples chosen at random from the original samples (Forgy method), or, randomly assigning a cluster to every data point (random partition), as described in [32]. Overall, the K-means algorithm consists in three steps. Step one is initialization, we specify K, the number of clusters, and define what are the initial clusters based on the chosen method. Step 2 is assigning membership to every datapoint, which requires computing the euclidean distance from every point to each cluster center. As in K-means, each point can only belong

to one cluster (hard clustering), membership m is expressed as seen in equation 2.2.

$$m(c_l, x_i) = \begin{cases} 1, \text{ if } l = \underset{j \in 1, ..., k}{argmin}(||x_i - c_j||^2) \\ 0, \text{ otherwise} \end{cases} \tag{2.2}$$

A datapoint will be assigned to the cluster to whose center it closest to.

Step 3 is recalculating the center of every cluster based on the newly assigned memberships. This center or centroid is calculated as a mean of all the datapoints which compose it as seen in equation 2.3.

$$c_j = \frac{\sum_i^n m(c_j, x_i) x_i}{\sum_i^n m(c_j, x_i)} \tag{2.3}$$

Steps 2 and 3 are then repeated until convergence or maximum stipulated number of iterations is attained. Because this algorithm is dependent on initialization, it is often done several times with different initializations.

Fuzzy C-means [33] is an adaptation of K-means algorithm, which, unlike its counterpart, allows datapoints to partly belong to each of the existing clusters. It minimizes the within-groups sum of square errors function seen in equation 2.1 but with a different membership function and a different way of calculating cluster centers. The membership function for this method can be seen in equation 2.4.

$$\begin{aligned} m_{i,j} &= \frac{1}{\sum_{k=1}^{K} (\frac{d_{ik}}{d_{jk}})^{\frac{2}{m-1}}} \\ s.t. \quad & \sum_{j=1}^{K} m_{i,j} = 1 \end{aligned} \tag{2.4}$$

The centers are calculated with the expression in equation 2.5.

$$c_j = \frac{\sum_i^n m(c_j, x_i)^m x_i}{\sum_i^n m(c_j, x_i)^m} \tag{2.5}$$

The hyper-parameter m, called the fuzziness index, that controls the extent of sharing among fuzzy clusters [34]. The higher it is, the fuzzier the cluster will be in the end. Its value is usually set to two, but the decision is mostly case specific.

Fuzzy c-means seems to be a better choice than k-means when there are overlapping clusters [35].

Although these two are some of the most used methods, many others are relevant in traffic analysis context. The choice of clustering method depends mostly on the type of data. So, with the appropriate processing, any clustering method should be applicable to this context.

Chapter 3

Counting Framework

3.1 Introduction

The number of vehicles at given time and place is one of the relevant characteristics of traffic. Vehicle counts need to be extracted from whichever data source is available. This chapter describes how vehicle count is obtained from a still image or a video frame.
In vehicle counting using traffic cameras we are commonly faced with two major problems: low image quality and high occlusion. Both of these traits make it very difficult to detect objects in images, so, we entirely avoid detection based methods. Instead, we use a regression based method which still makes use of the spatial information in the image. This method has already been applied to the context of vehicle counting in images (see [7]). Counting is formulated as density map estimation. The estimator Θ, transforms an input image *I* into its corresponding estimated density map $\widehat{D}$:

$$\widehat{D} = \Theta(I), \quad I \, \epsilon \, R^{H \times W \times C}, \quad \widehat{D} \, \epsilon \, R^{H \times W} \tag{3.1}$$

H and *W* are the height and width of the input image and *C* is its number of channels. A density map is an alternative representation of the image where the integral over a segment of the map returns the number of objects contained in the corresponding image segment. As Figure 3.1 shows, in the case of digital images, the sum of the pixel values in a density map should return the object count in the analogous image with the constraint that the sum of pixels which represent one car, must return one.

Given its unbeatable performance, we relied on CNN's to estimate the regressor. This network is employed to directly transform an image into its corresponding vehicle density map which also allows the estimation of the number of vehicles in the image. A more detailed description of the neural network is provided in the 3.2 section. To recover an overall car count in an image, the corresponding density map output of this network is then integrated.

Figure 3.1: Example of annotated image and corresponding ground truth density map

3.2 The U-Net

Following the appreciation in Section 3.1, we use a U-Net architecture to solve the density estimation problem. This network is composed by an encoder (contracting side), a decoder (expanding side) and feature combination at multiple levels.

In total, the implemented network comprises 9 basic convolution blocks (ConvBlocks), max-pooling and transposed convolution layers between them and one final convolution layer.

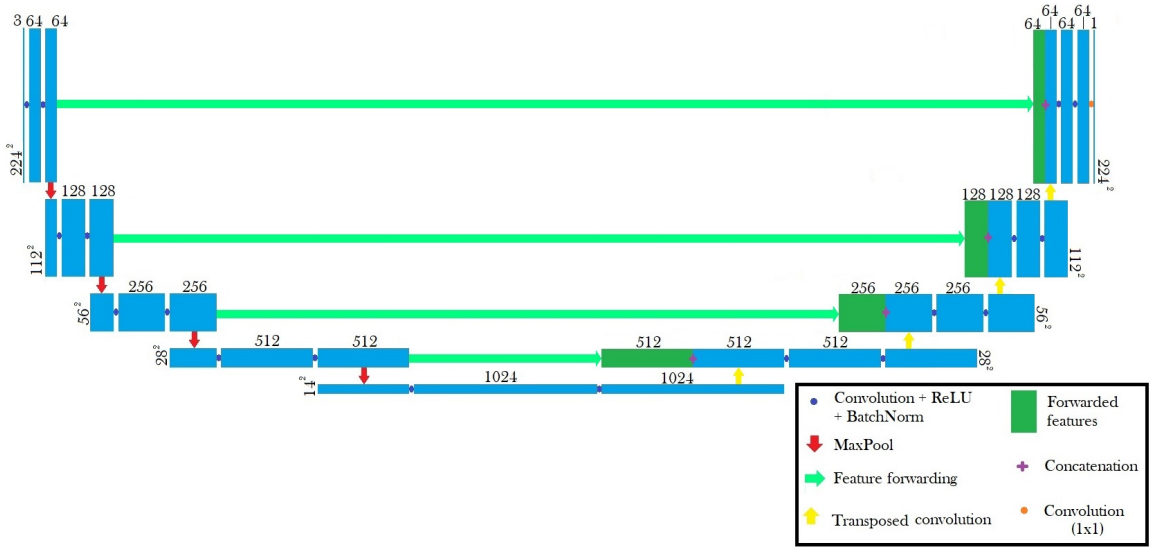

Figure 3.2: U-Net architecture

The term convolution block refers to a set of 4 layers and 2 activation functions. These layers are organized as two sets of convolution layers each followed by one rectified linear unit (ReLU) and one batch normalization layer. All convolution layers in every ConvBlock of the network use a 3x3 kernel and a stride of 2.

The contracting side is composed of 5 sequential convolution blocks, which we will individually designate by DownBlock(i), with i = 1, 2, ..., 5. This side uses a total of 4 max-pooling layers (with stride 2 and

kernel size 2x2), one between each DownBlock. DownBlock(i) has 2^{5+i} output channels, so, 64, 128, 256, 512 and 1024 channels respectively in their convolution layers.
The input of DownBlock(1) should be an RGB image with 224 by 224 pixels and three channels (corresponding to the three colors). For the following blocks, i = 2, 3, 4, 5, the input is the output of the MaxPool layer of DownBlock(i-1).
The decoder side is formed by 4 transposed convolutions, each followed by a ConvBlock. We will designate this sequence as UpBlock(j), with j = 1, ..., 4. At the end there is one final convolution to obtain the desired number of output classes (output channels is the number of classes). The transposed convolution layers use a 2x2 kernel and a stride of 2. The last convolution layer uses a kernel of size 1 and a stride of 1. Any UpBlock(j) has 2^{11-i} input channels and 2^{10-i} output channels The input of UpBlock(1) is the output of DownBlock(5) concatenated with the output of the MaxPool layer at the end of DownBlock(4). For every single other UpBlock(j), with j = 2, 3, 4, the input is the output of UpBlock(j-1) concatenated with the output of the MaxPool layer at the end of ConvBlock(5-j).
In the end, the structure of the neural network is as is shown in 3.2.
It contains a total of 31 043 520 trainable parameters.

The desired output representation of a vehicle is the equivalent of convolving its bounding box with a Gaussian Normal kernel. If a pixel is inside the bounding box of multiple vehicles, its contribution to each will add up. The value V_p of each pixel *p* in the ground truth density image *D* used is given by 3.2.

$$V_p = \sum_b \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}(\frac{(X-X_c)^2+(Y-Y_c)^2}{\sigma^2})} \tag{3.2}$$

In 3.2, we use *X* and *Y* as the coordinates for pixel *p* and X_c and Y_c the coordinates of the center of each bounding box *b* in which *p* is contained for a given image.

The loss function for this network takes into account the loss between ground truth and the estimated density map and the loss between the labelled number of vehicles and the estimated vehicle count (see 3.6). The total loss is given by 3.3.

$$L = L_{count} + L_{density} \tag{3.3}$$

$$L_{density} = \frac{1}{n} \sum_n \sum_p (V_p - \widehat{V_p})^2 \tag{3.4}$$

$$L_{count} = |C_I - \widehat{C_I}|^2 \tag{3.5}$$

Having obtained the estimated density map $\widehat{D}$ using our estimator Θ, the sum of the value of each pixel *p* should return the estimated number of vehicles $\widehat{C}$ in an image *I*. This idea is represented in formula 3.6.

$$\widehat{C}_I = \sum_p \widehat{Vp}, \quad p \, \epsilon \, D_I \tag{3.6}$$

3.3 Analysis

The dataset used to train the network is composed of 60 000 annotated images. These images come from 16 web cameras in Manhattan which acquired videos at a rate of around one frame per second. The annotations contain vehicle bounding box coordinates for each frame which were used to generate the ground truth density maps. The number of bounding boxes was used to directly infer the number of vehicles in each frame. These bounding boxes can be seen in Figure 3.1.

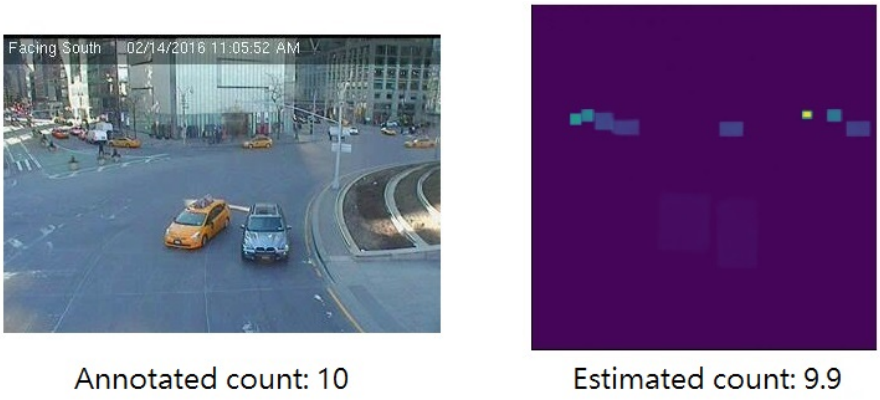

Figure 3.3: Example of inference over an image from training dataset in New York

To analyse the results of the neural net after being trained, an experiment was set up. This experiment consisted in collecting all images from dataset 1, where ground truth vehicle count was equal to some variable k, and process them with the U-Net. This experiment was repeated for $k \in [0,\ 1,\ 5,\ 10,\ 15,\ 17,\ 20,\ 25,\ 30]$ (see table 3.1). This was done to try and ascertain whether the results of the network's inference were more affected by the number of training samples for a certain k or the occlusion inherent to a great number of vehicles in an image. An example of an inference over an image from the training dataset can be seen in Figure 3.3

G. Truth	Mean	Std.	Min.	Max.	Median	Perct. 25-75	Perct. 5-95	Samples
0	-0.029	0.093	-0.109	0.731	-0.047	-0.058 : -0.033	-0.069 : 0.043	186
1	0.941	0.140	-0.019	2.295	0.932	0.899 : 0.967	0.811 : 1.127	878
5	4.899	0.339	2.815	14.59	4.904	4.816 : 4.998	4.553 : 5.198	2033
10	9.857	0.714	3.820	19.44	9.904	9.775 : 10.01	9.287 : 10.27	2936
12	11.87	0.806	5.447	22.30	11.90	11.78 : 12.02	11.26 : 12.30	3280
15	14.76	0.931	6.173	24.39	14.89	14.75 : 15.01	13.98 : 15.24	2989
17	16.74	0.914	8.667	24.03	16.89	16.73 : 17.02	15.48 : 17.27	2299
20	19.70	1.041	9.631	24.04	19.90	19.73 : 20.04	18.12 : 20.32	1784
25	24.75	1.006	16.83	28.92	24.91	24.78 : 25.05	23.64 : 25.34	767
30	29.69	1.359	20.80	32.33	29.93	29.77 : 30.09	28.11 : 30.48	444
35	34.87	0.564	30.85	36.62	34.94	34.77 : 35.10	34.33 : 35.34	177

Table 3.1: Results of inference in training dataset

In this table we present the ground truth values of vehicle count in an image used in the experiment above explained. Each row contains the mean, standard deviation, median and the number of samples tested in each case. It also includes the intervals which contain fixed percentages of the total

inferred counts for that experiment. These intervals are presented in order to show the inference values distribution succinctly without visual aid.

The neural network generally undershoots by a small amount the amount of vehicles in an image. The standard deviation seems to steadily increase with the number of cars in the image, this may be an indication that the results are more affected by occlusion than by the number of examples with k vehicles it is fed. In all the tested cases, most samples are very close to the median. Still there seem to exist cases where the inferred result is wrong by a large margin. To better understand what caused these errors, the images for which results were farther from the ground truth count were analysed. When referring to results we are strictly speaking in terms of estimated number of vehicles.

When the images with estimated count high above ground-truth where checked, a lot of them contained large busses which occupied a large portion of the image. Even though these were not even recognized as vehicles, there was an apparent excessive overestimation of the number of vehicles (see Figure 3.4). The error mostly came from identification of cars which were in the image but not in the annotations. This may be due to the fact that the annotations only contain bounding boxes for vehicles inside a certain area of an image but the image itself is not masked when used for inference (see Figure 3.5).

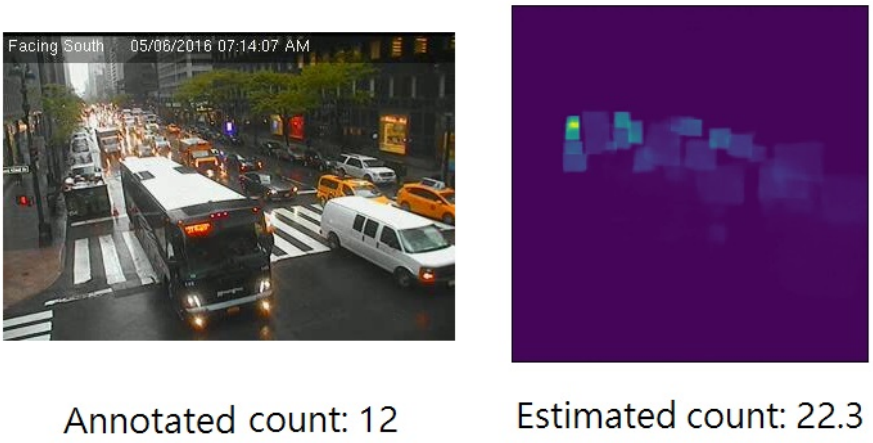

Figure 3.4: Example 1 of overestimation of number of vehicles

When the opposite case was analysed (predictions much below ground-truth), we found similar errors as the annotated images contained bounding boxes for vehicles which are barely recognisable as such, even for a human. This included cases where only a fraction of the vehicle was in the frame (see Figure 3.6) and others caused by high occlusion or bad lighting of the scene (see Figures 3.6 and 3.7).

In summary, starting from a sequence of images or video, we are able to produce a stream of data of counting estimations over time as shown in Figure 3.8

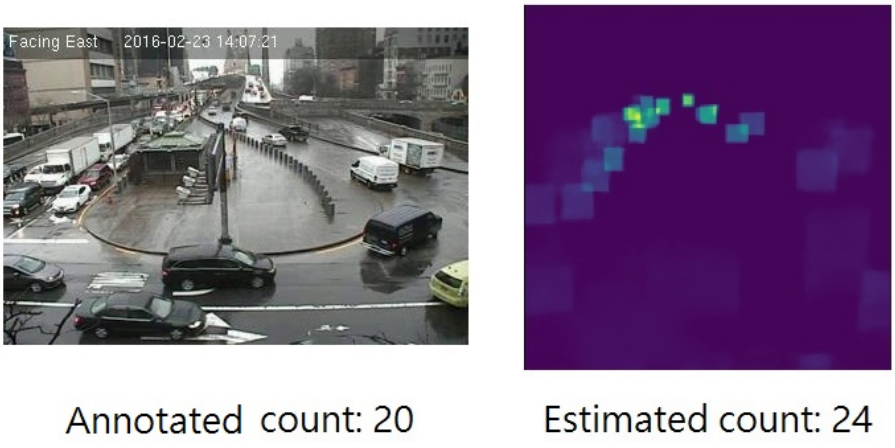

Figure 3.5: Example 2 of overestimation of number of vehicles

Figure 3.6: Example 1 of underestimation of number of vehicles

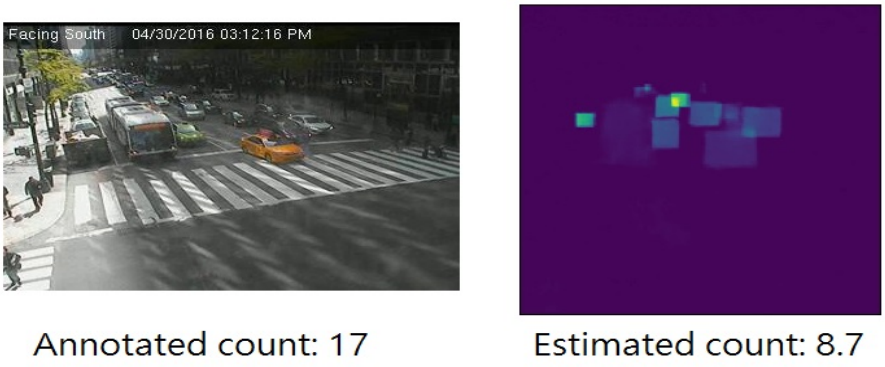

Figure 3.7: Example 2 of underestimation of number of vehicles

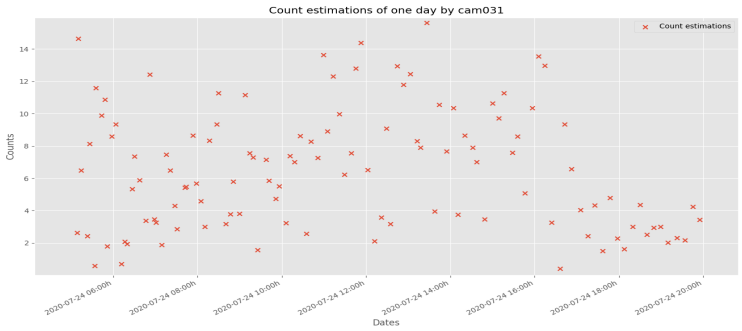

Figure 3.8: Example vehicle count estimations based on image acquisition of one camera on 24/07/2020

Chapter 4

Clustering

In this chapter we explain how vehicle count estimations generated with the method described in Chapter 3 are processed and clustered. Data acquisition was not linearly spaced and the data itself is noisy. Since for every cam multiple days were acquired, each containing a number of data points, the accumulation of vehicle counts form a high dimensional space. With this increase the dimensionality of data, the computational complexity of its analysis grows. The objective of this chapter is to process data, making it linearly spaced and with less fluctuations. Then, to find a low rank approximation of the high dimensional space in which that descriptor is contained. With this new representation, a clustering algorithm is used to group the observations and make it easier to find similarities between data points. To accomplish this objective, we split the work into three tasks: generation of the descriptor, sparse subspace representation and spectral clustering.

4.1 Descriptor

Since the acquisition of data has a low frequency and is not equispaced (as shown in Figure 4.1), before moving to the next step some pre-processing is needed. To obtain an equispaced set of counts, the sequence of vehicle count estimations is interpolated using linear, quadratic and cubic methods. By selecting equispaced values resulting from interpolating the original data, the problem of non-linear spacing between samples is solved. An example of the result of this operation can be seen in Figure 4.1.

A moving average is then applied to the interpolated data. This will act as a low-pass filter to reduce short term fluctuations in the counting acquisitions. The objective of this filtering is to remove the systolic dynamic caused by alternating green and red lights. The averages are obtained using 4.1.

$$A_i = \frac{\sum_{j=i-k}^{i+k} \widehat{C}_j}{2k+1} \tag{4.1}$$

Where A_i is the average corresponding to the count estimation $\widehat{C_i}$ and *k* is a variable parameter which defines the size of the subset used in the calculation.

Finally, a re-sampling is performed to obtain the dimensions needed for the next step.

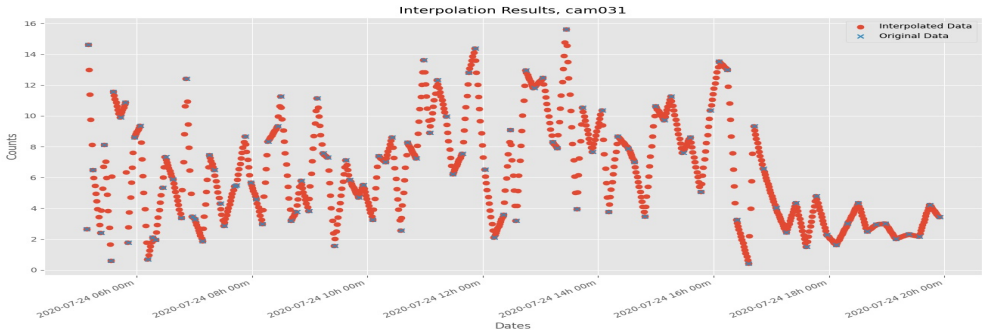

Figure 4.1: Linear interpolation of Cam 031 acquired data on 24/07/2020

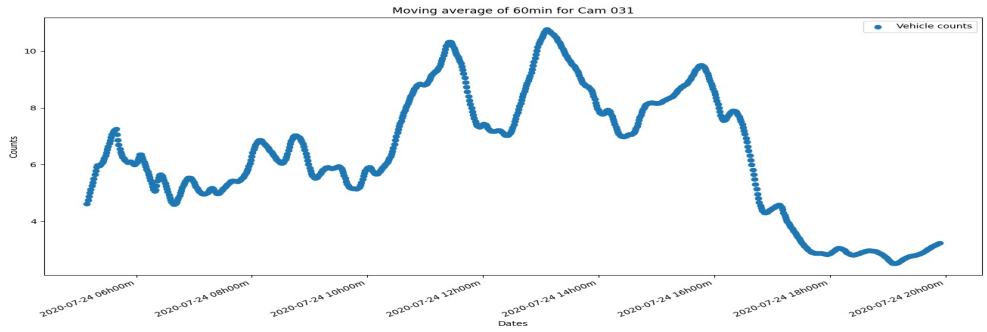

Figure 4.2: Moving average (60min) of Cam 031 interpolated data on 24/07/2020

Each cam descriptor (D) includes every vehicle count of that camera organized by day. A column contains all the observations of one day and a row contain observations of one specific hour of all days. The corresponding date and time of the vehicle counts in the descriptor are saved in an analogous matrix (T) with the same dimensions, in such a way that the observation $D_{i,j}$ has the associated time in $T_{i,j}$.

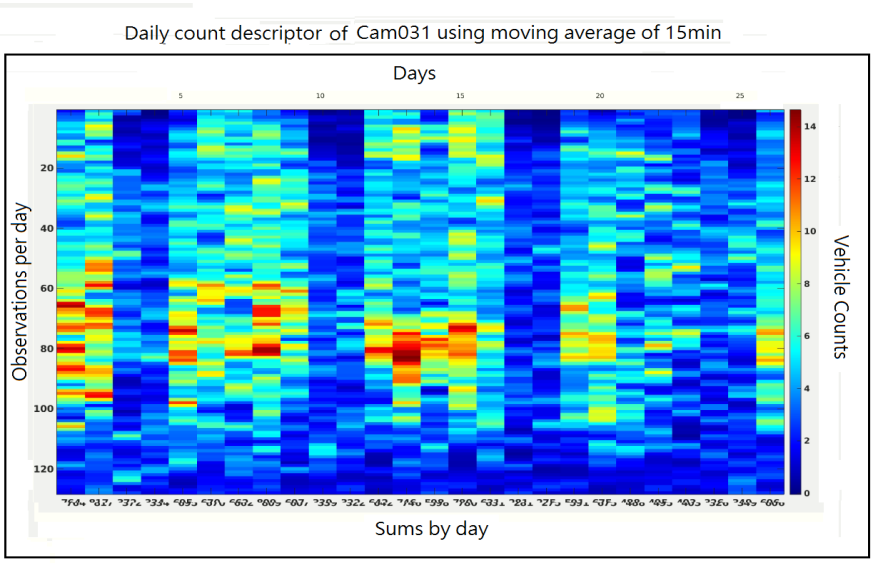

Figure 4.3: Daily count descriptor of Cam031

When reading these descriptors, an option is given to select one or multiple cams. If multiple cam descriptors are read simultaneously, they are concatenated horizontally to obtain a joint descriptor with as many columns as the acquired number of days times the number of selected cameras. The number of rows remains equal to the number of acquisitions in a day.

It is also possible to segment each day's worth of samples in equal parts. These segments, which will be called chunks, each contain $n^\circ\, of\, samples\, in\, a\, day / n^\circ\, of\, segments$ samples. For example, if one camera and two chunks are selected (splitting days in half), the descriptor will have $n^\circ\, of\, days \times 2$ columns, and $n^\circ\, of\, samples\, in\, a\, day/2$ rows.

4.2 Sparse Subspace Representation

In this section, we attempt to utilize the self-expressiveness of the data to obtain a sparse representation of the descriptor. The main idea is to take advantage of the existing similar patterns in traffic and to represent a section of data as a linear combination of n other sections. To impose sparsity, n must be a relatively small number. We use a sparse optimization program described in [36]. It consists in a minimization process which solves the expression 4.2.

$$\begin{aligned} & min\, ||C||_1 + \lambda_e\, ||E||_1 + \frac{\lambda_z}{2} ||Z||_F^2 \\ s.t. \quad & Y = YC + E + Z,\; 1^T C = 1^T,\; diag(C) = 0 \end{aligned} \tag{4.2}$$

In 4.2, Y is the descriptor, C is the sparse coefficient matrix, E is the sparse error and Z is the noise. C is a square matrix, with size equal to the number of columns of the descriptor. Its n^{th} column contain the sparse representation of the data in column Y_i. Each coefficient c_j in column C_i represents the fraction of Y_j that is used in the reconstruction of Y_i. Diag(C) is the diagonal vector of C and is forced to zero to prevent the solution from being an identity matrix.

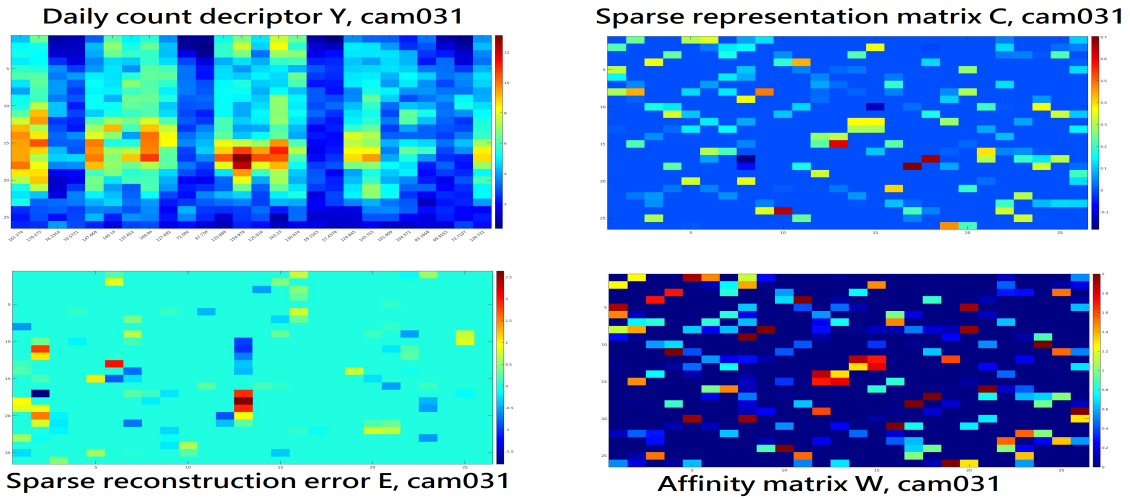

Figure 4.4: Top left: Daily count descriptor, Y, of Cam031. Top right: Sparse subspace representation C, using $\lambda_e = \lambda_z = 1 \times 10^{-3}$. Bottom left: Reconstruction error E. Bottom right: affinity matrix W (each with its own scale)

The affinity matrix W, where w_{ij} shows the degree of affinity between column Y_i and column Y_j, is calculated as $W = C + C^T$ but before proceeding to this calculation, an optional step suggested in [36] was implemented. This optional step consists in the normalization of each the columns of C using its corresponding l_∞ norm: $c_i \leftarrow \frac{c_i}{||c_i||_\infty}$. In this application, as each column of Y corresponds to a day's worth of observations for one camera, $w_{i,j}$ is the degree of affinity between day i and day j of the descriptor. Using the affinity matrix W, it is possible to find connected components in the descriptor Y by applying a clustering algorithm to it as it already takes the form of a graph. An application example of the matrices described above can be seen in Figure 4.4.

When observing a specific day, we can check its corresponding row or column in the affinity matrix to understand which other days are closely related to it as shown in Figure 4.5.

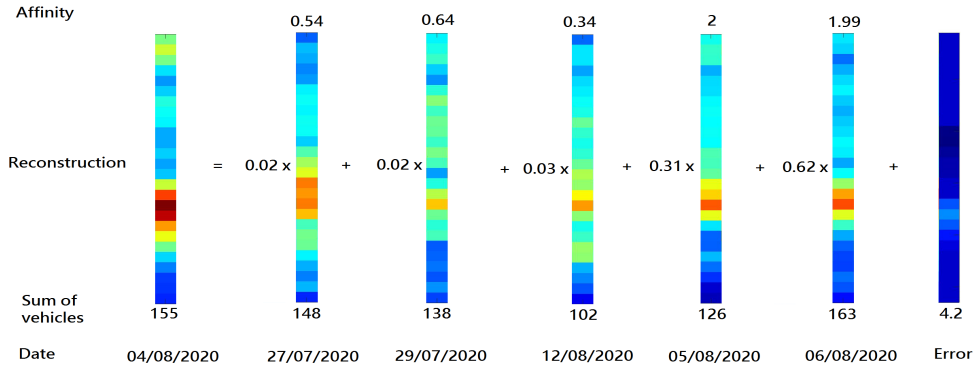

Figure 4.5: Day 13 (Tuesday 04/08/2020, cam031), its reconstruction using related days and the reconstruction error

We can see that sparse subspace associated with day 13 (04/08/2020) consists of 5 other days which have a peak in vehicle counts at around the same time but slightly shifted. This shift in time causes an error to exist right after (positive error) the center of the peak. By summing all these components multiplied by their reconstruction ratio, we obtain the estimated reconstruction of the original day. This is shown in Figure 4.6, here, the overall shape of the reconstruction is the same as that of the original data. As expected, the reconstruction error does have a greater amplitude at the time of the peak.

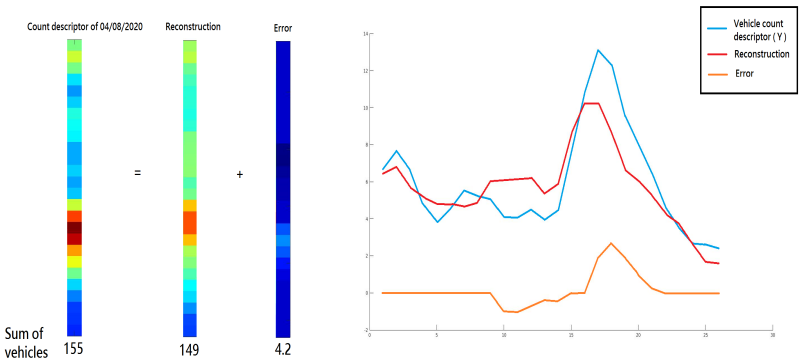

Figure 4.6: Vehicle count descriptor of Tuesday 04/08/2020 of cam031, its estimated reconstruction and its error (not including white noise)

4.3 Spectral Clustering

To find similarities between days or segments of days, we try to isolate connected components in the affinity matrix as these represent groups of similar days. But since there is noise in the data, the graph is fully connected and thus, non-partitionable. To obtain n connected components in the fully connected graph, we make use of spectral clustering. Spectral clustering is a clustering technique which is most related to the graph theoretic formulation of grouping. In this formulation, the data is represented as a weighted, undirected graph where the data points themselves are the nodes, and the edges linking pairs of nodes are a function of similarity between the nodes. In this specific application, the nodes are days of vehicle counts. The technique itself consists in utilizing a measure of the quality of data partitions, the normalized cut (N_{cut}). This criterion can be minimized when formulated as an eigenvalue problem and the resulting eigen-vectors can be used to construct good partitions.

When partitioning into two sub-graphs A and B, the normalized cut takes the form 4.3.

$$\begin{aligned} N_{cut}(A,B) &= \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \\ cut(A,B) &= \sum_{u \epsilon A, v \epsilon B} w(u,v) \\ assoc(A,V) &= \sum_{u \epsilon A, t \epsilon V} w(u,t) \end{aligned} \tag{4.3}$$

Where V contains all nodes in the graph, so that $assoc(A,V)$ is the total connection from nodes in A to all nodes in the graph. In this expression, $cut(A,B)$ is the total weight of the edges removed when partitioning A from B. The optimal solution is obtained by minimizing $min_x N_{cut}(x)$, where x is an indicator vector, as shown in 4.4.

$$x_i = \begin{cases} 1, \text{ if } i \in A \\ -1, \text{ otherwise} \end{cases} \tag{4.4}$$

Using spectral clustering in the graph (affinity matrix, W) will indicate the partition which minimizes the weight associated with removing edges to obtain the desired number of clusters. The number of clusters (k) is manually set and must be chosen in accordance with the problem at hand. In the specific case of traffic, we show an example where we set $k = 2$. The results are shown in Figure 4.7. These results show, for a single camera and using two clusters, an alternating pattern of sets of days attributed to group 1 followed by another set of days attributed to group 2. All of the days in group 2 appear to have one common factor: a lower density of traffic. When inspecting the dates of the days in group 2, we find that most of them are either Saturdays or Sundays. So, it seems that when partitioning a sequence of days into two clusters, this method mostly groups days by high or low traffic density, which incidentally also has a correlation with being a working day or a weekend day.

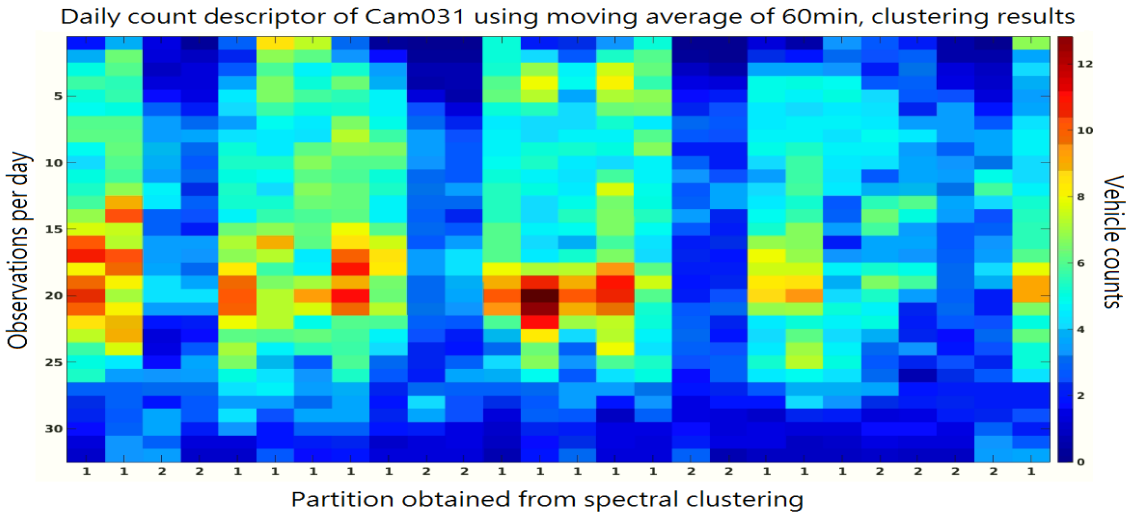

Figure 4.7: Results of applying spectral clustering to daily count descriptor of Cam031 (60min moving average) using two clusters

4.4 Single Camera Data Analysis

In this chapter we smoothed and formatted data obtained from a neural network's output to obtain a daily vehicle count descriptor for one or multiple cameras. We used a moving average to smooth the data. This filter had a period which varied between 15 minutes, 30 minutes and 60 minutes. We were able to observe the effects of changing the period of the filter by plotting its output (Figure 4.8), and since the data was very noisy due to having a small acquisition frequency, we opted to work with a 60 minutes period. These moving averages were re-sampled at double the frequency of the filter, separated

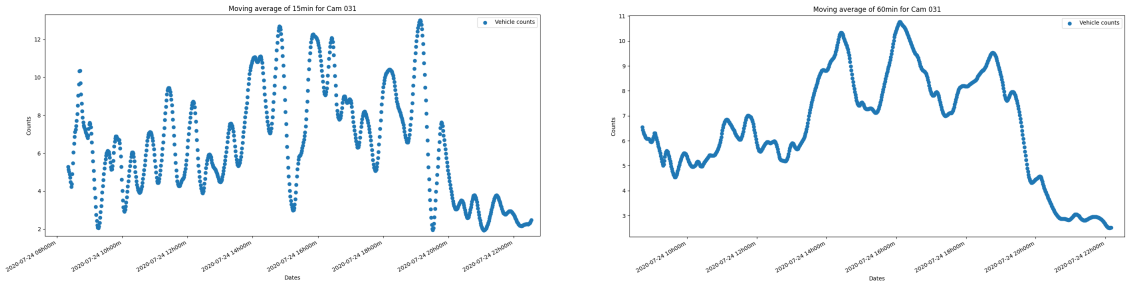

Figure 4.8: Results of applying moving average with period of 15 minutes (on the left) and 60 minutes (on the right) to interpolated data from cam031

in individual days and concatenated to form a vehicle count descriptor. These descriptors contained the days organized in columns. By presenting the descriptors as images we can visually identify some patterns, for example, a descriptor having greater traffic density in the afternoon gives the idea that its corresponding camera is placed in a road used as an exit from the city, and cameras placed in parking lots will be prone to have greater traffic density up until the beginning of the afternoon (see Figure 4.9). In addition, the sum over the columns of the descriptor was computed in order to gain more intuitive information on the patterns along a sequence of days. Having generated the descriptors, we used the self-expressiveness of the data to represent it using sparse subspaces. The sparse representation of

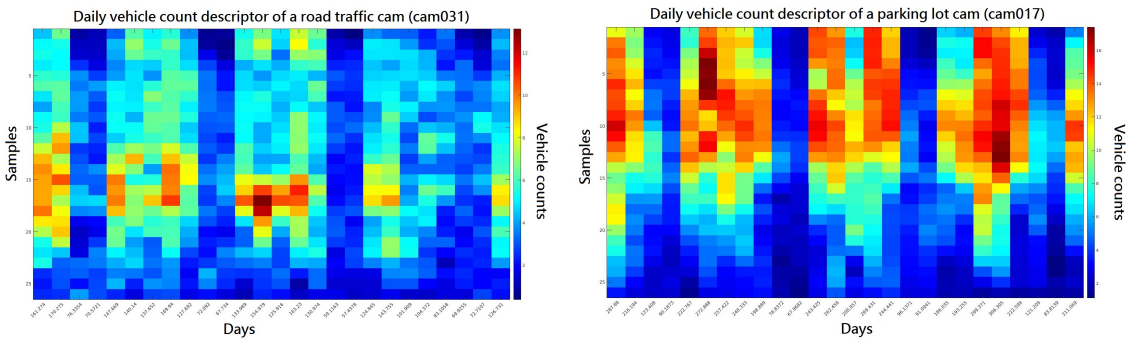

Figure 4.9: Left: Daily vehicle count descriptor of Cam031 (road intersection). Right: Daily vehicle count descriptor of Cam017 (parking lot), with different scales

the data, a sparse coefficient matrix, contained information on how to reconstruct any day by using a linear combination of n other days. The sparse error matrix which embodied the reconstruction error (not including white noise), allowed to identify when the traffic patterns differed substantially from the other existing samples. With the sparse coefficient matrix, we generate the affinity matrix. Since this matrix can be regarded as a graph, we applied spectral clustering in order to obtain k connected components in it. These connected components give us information on which days of the descriptor were most closely related. The result of applying spectral clustering to the descriptors is the classification of days into one of k clusters. We can also observe a lower traffic traffic density in the same days in both descriptors of Figure 4.9. Just like in Section 4.3, when checking the dates of these days, they turn out to correspond mostly to weekend days. With this information in mind, we plotted the moving average of one camera clearly identifying the weekends and another plot where the days were identified based on the group they belonged to, as shown in figure 4.10.

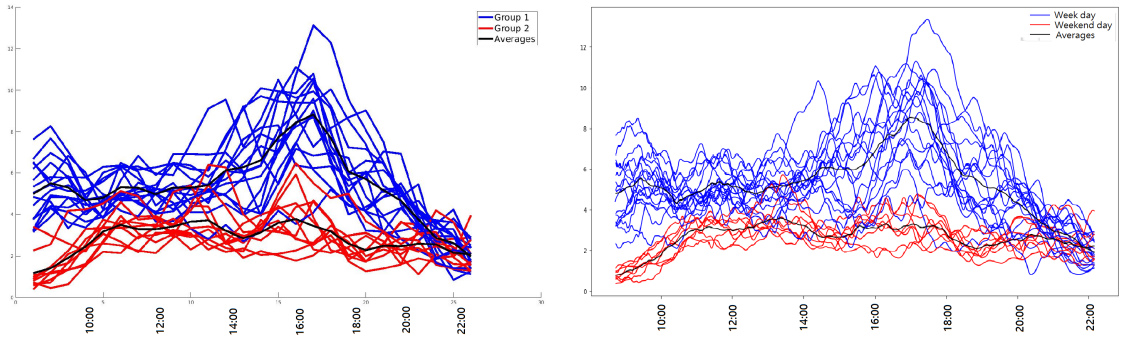

Figure 4.10: Left: Daily vehicle counts descriptors grouped by clustering classification. Right: Daily vehicle count moving averages, grouped by week days and weekend days (both from cam031)

By observing the amplitudes of the vehicle count curves, it is clear that weekends do generally tend to have a lower traffic density, especially at the time of the traffic peak but also in the morning. Here when using two clusters it seems that spectral clustering uses total traffic density as the key characteristic to differentiate between clusters. As so, two days which have exceptionally lower traffic density are grouped together with the weekend days.

Another portion of the data was acquired right before and right after the COVID lockdown of 2020 was imposed in Estonia.

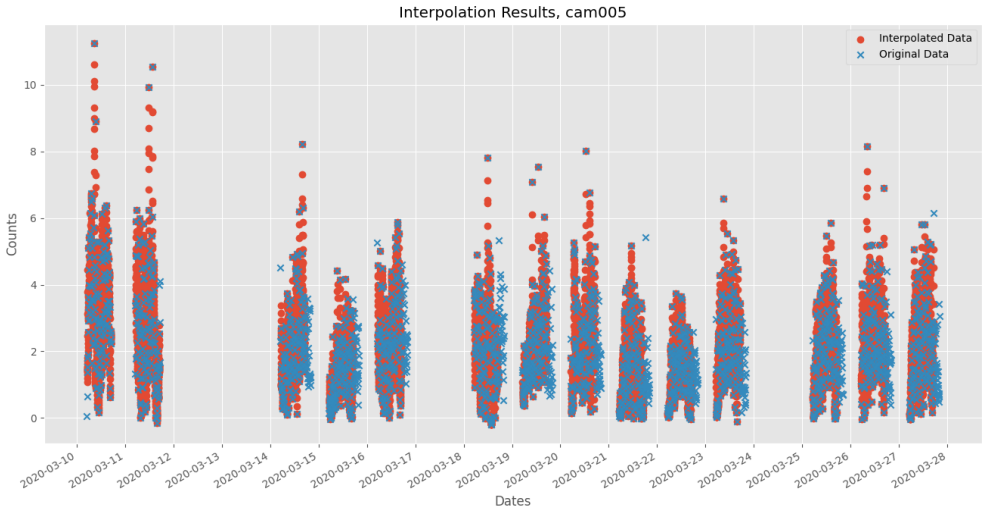

Figure 4.11: Vehicle counts of the acquired period, in blue is the original and in red is the interpolated data obtained from cam005 in March 2020

As can be seen in Figure 4.11, the first two days have a maximum value larger than the following acquired days. These first two days were acquired whilst the lockdown was not yet imposed, and the difference in maximum values reflects the change in traffic density caused by that same lockdown. To better visualize this change, we generated a plot which contains the results of applying a moving average to two specific days: Wednesday 11/03/2020 and Wednesday 18/03/2020. By superposing the data of these two days, we hope to determine, for this sample, if the lockdown did in fact cause significant reduction in traffic density. The results can be seen in Figure 4.12.

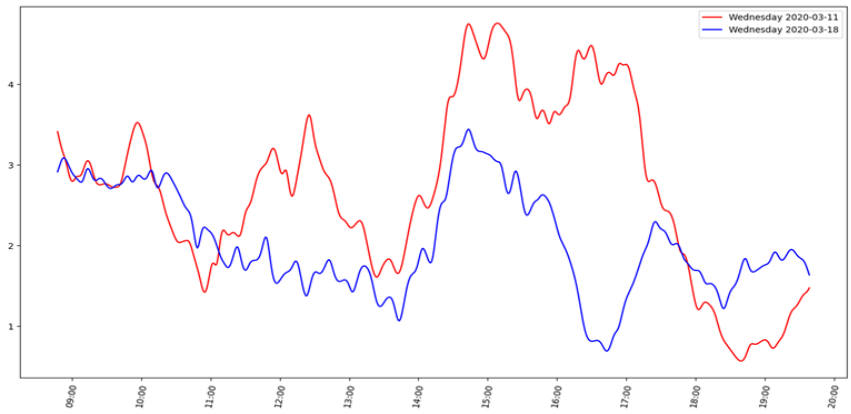

Figure 4.12: Daily vehicle counts of Wednesday 11/03/2020 (red) and Wednesday 18/03/2020 (blue) after smoothing with moving average (both from cam005)

What can be observed in Figure 4.12 is that the general shape of the curves is the same but at the

time of the traffic peaks, the number of vehicles detected is quite different. In fact, the first peak in the red curve (around 12h00) is non-existent in the blue curve. The larger peak, from around 14h00 to 18h00, exists in both days and starts and ends at around the same time, but is quite diminished on the curve corresponding to 18/03/2020. This seems to indicate that even though the overall traffic pattern did not change much, traffic density was lowered by the lockdown.

We then generate the daily count descriptor to visualize all the data after processing it, the daily sum of vehicles is also computed to give an additional indication of the variation in traffic density. The descriptor can be seen in Figure 4.13.

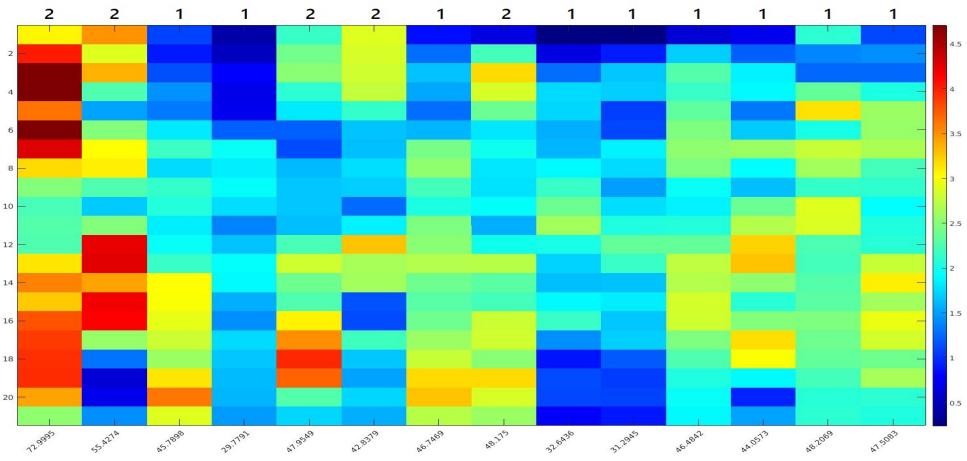

Figure 4.13: Daily vehicle counts descriptor of cam005 acquisition in March 2020 (sum of vehicles below each column and clustering results on top)

In this dataset, two weekends were captured. But unlike the descriptor generated with data from August 2020, here (Figure 4.13) weekends are not so distinguishable from weekdays. One of them (22 and 23 of March) can be visually identified but the other (14 and 15 of March) is harder to single out, mostly because Saturday the 14th has as much traffic density as the other post-lockdown weekdays. Spectral clustering is applied, with two clusters, to see how this method groups the days included in this dataset. The results are presented in the same fashion as in 4.10 and can be seen in Figure 4.14.

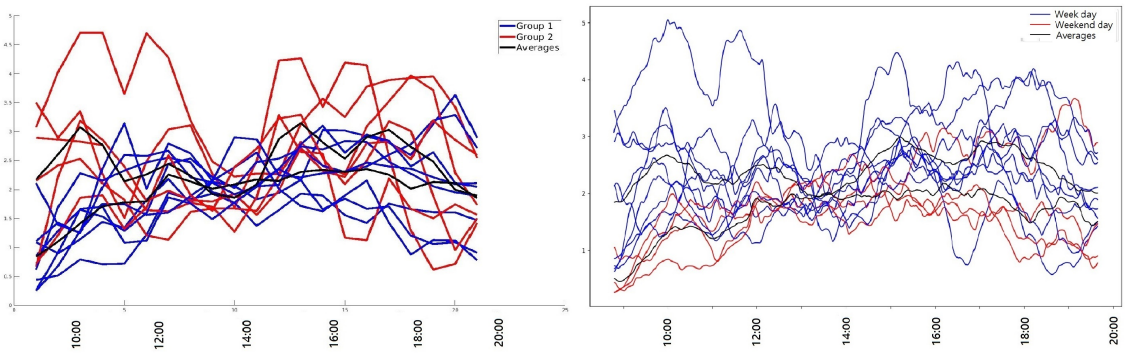

Figure 4.14: Cam005. Left: Daily vehicle counts descriptors grouped by clustering classification. Right: Daily vehicle count moving averages, grouped by week days and weekend days

Again, some differences can be noted between this figure and its analogous figure from the August dataset. Weekends are not as clearly separated from weekdays, and even though they are grouped together (in group 1), so are most of the weekdays. Group 2 seems to be composed of the days with the highest sum of vehicles (not including 14/03/2020).

Chapter 5

Application: a Case study of Tallinn

In this chapter we describe the results of the process in the aforementioned chapters. We first indicate how the data was obtained and then results of the two main steps (counting and clustering) are presented.

5.1 The Cam Network

The dataset used as input for the sequence of steps described in chapters 3 and 4 is composed of a total 1447577 images of two different time periods. The first one spans from 9 of March to 6 of April (except 11, 12 and 13 of March) 2020 and comprises 734284 images The second one spans from 23 of August to 17 of September 2020 and includes the remaining 713293 images. In March, images were acquired from 5:30 A.M. to 5 P.M and in August from 5 A.M to 7:30 P.M. The images are obtained from video streams of multiple cameras in the city of Tallinn, Estonia.

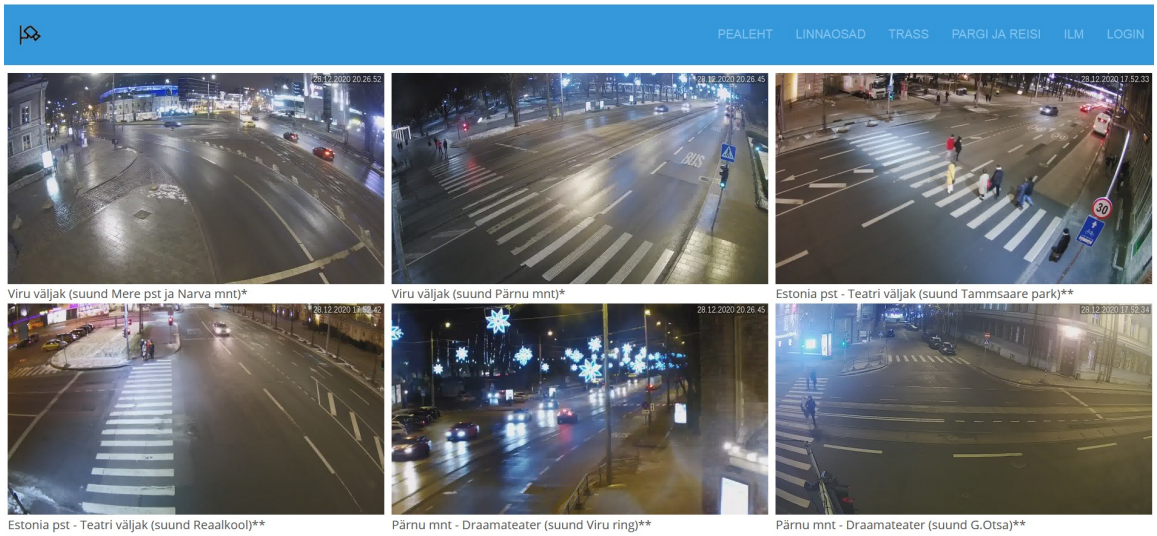

Figure 5.1: Grid showing cameras in Tallinn

The video feed from these cameras is made available in the city's own official website [37] where they are presented in a grid-like manner. In total, around 176 cameras are linked to this website, which

capture streets, intersections and parking lots all around the city. The feeds are always active and when a camera is not working or its connection is lost, a gray-scale image with the corresponding warning replaces the feed. The images provided have a high resolution (1920 by 1080 or 1280 by 720 depending on the camera) and the refresh rate is that of one image per second. Even though the original resolution and framework are relatively high, the acquisition forces the resolution to be the same as the input size for the U-Net described in 3.2 (224 by 224 pixels) and the image download for each camera was executed once every five minutes.

The download from the website was executed using a python script, more specifically using the urllib3 [38] library. For each camera, one image was downloaded every five minutes in a nonlinear time spacing fashion. Every five minutes, a routine was called which generated a random number between 1 and 299, this is the number of seconds to wait before downloading. This way, the download rate of one image every five minutes was ensured and an uneven acquisition was implemented. It was not possible to acquire at a faster rate as the source would create a buffer such that, if requests for images were too frequent, the buffer would simply hold one single image and thus, two image requests submitted at different times would return the same image. Of the total 176 available cameras, we selected 11 which are identified on Figure 5.2. We chose 2 cameras of parking lots (Cam017, Cam158), and the rest are cameras filming road segments. These road segments include streets both close and far from the city center. Most of these cameras capture street intersections but others cover bifurcations or merging of streets, and a few others are filming straight patches of road. There is no rule as to whether the camera is pointing towards or away from the city, and the individual direction of filming for each camera is shown in Figure 5.2. Unfortunately, two of the selected cameras did no capture images during the period of March, these were cameras ”cam112” and ”cam158”.

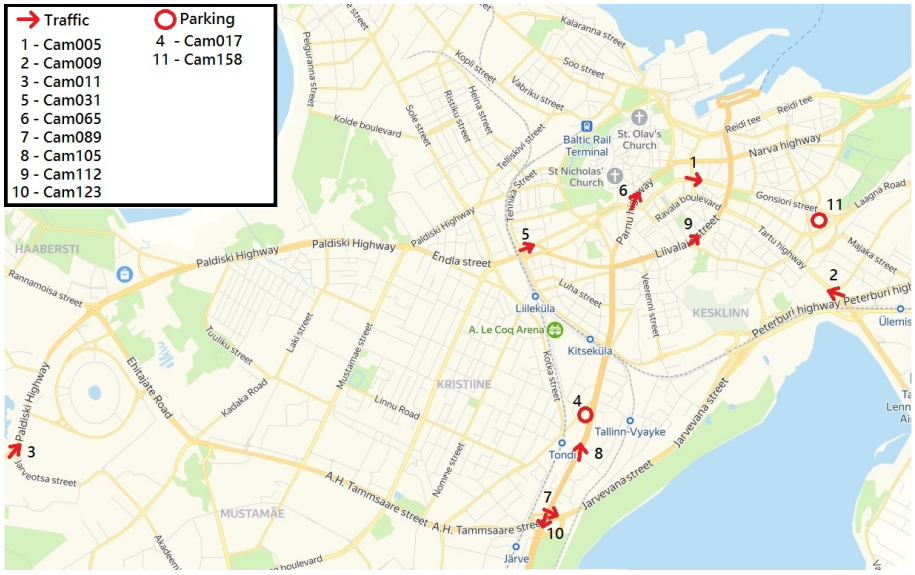

Figure 5.2: Selected cameras for results analysis

One image from each camera is shown in Figure 5.3. These cameras have different points of view and observe traffic flowing in multiple directions, for example, incoming towards the camera or flowing away from the camera. We included in our selection cameras located in parking spots. This was done

with the intention of comparing traffic patterns of road segments and parking lots.

Figure 5.3: Example images obtained from the selected cameras for results analysis

With the acquisition process finished, we end with a selection of eleven different cameras and data from two distinct time periods (March and August). This data is composed of sequences of images, obtained at an uneven rate between 8 A.M and 10 P.M or between 8:30 A.M and 8 P.M (local, Eastern European Time) depending on the period. We now need to process this data to make it usable for its intended purpose.

5.2 Processing

In this section we present the sequence of steps used to transform the images obtained from the cameras in Tallinn into the sparse subspace representation mentioned in Section 4.2.

The processing of the data comprises 8 main steps. These are inference, obtaining counts from traffic density maps or regression, interpolating, filtering with moving average, segmentation by date (each day's worth of vehicle counts for one camera is represented by an array of vehicle count values) and sub-sampling, concatenate the multiple days, calculate a sparse subspace representation and, finally, classification of days using spectral clustering. This sequence of steps can be seen in Figure 5.4.

The starting point for this process is to transform the downloaded images into usable, vehicle counts. In our approach, this transformation is implemented in two parts. First we transform the image into a traffic density map. We use a convolutional neural network to complete this task. This neural network follows the architecture of a U-Net (see 3.2), a widely used convolutional network. The input of the network are the images downloaded, and the output are their corresponding density maps. These maps contain information regarding both the location and number of vehicles in the image by representing them as patches of pixels with value different from zero. And then by summing the value of all of the pixels in any of these patches, we get the estimated number of vehicles in that patch. This way, we transform a given image into a datapoint, containing the date of its acquisition, and a value which is the number of cars in the acquired image.

Next, we make the sequence of datapoints linearly spaced. Because of the way the images were acquired (random process initiated every five minutes as described in Section 5.1) the samples were

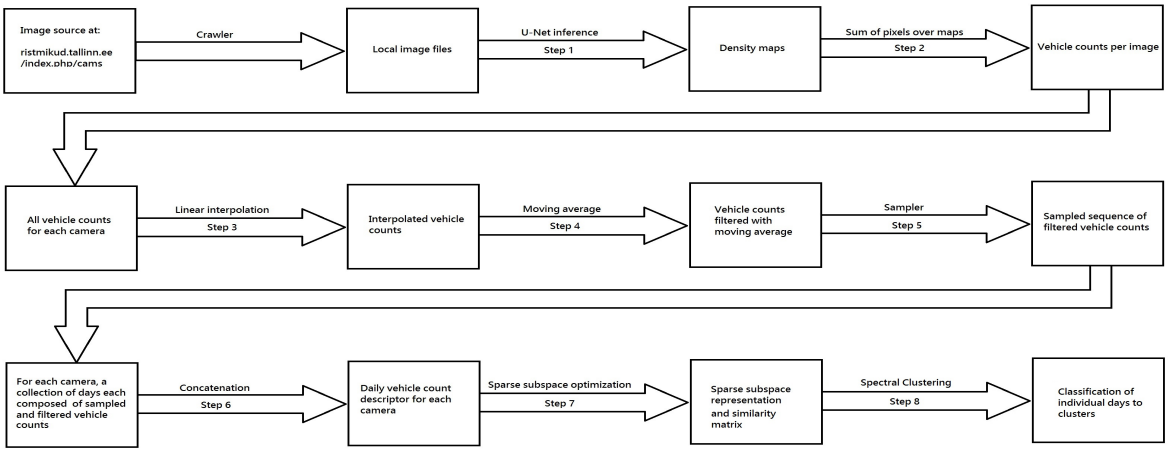

Figure 5.4: Sequence of steps in the processing of data

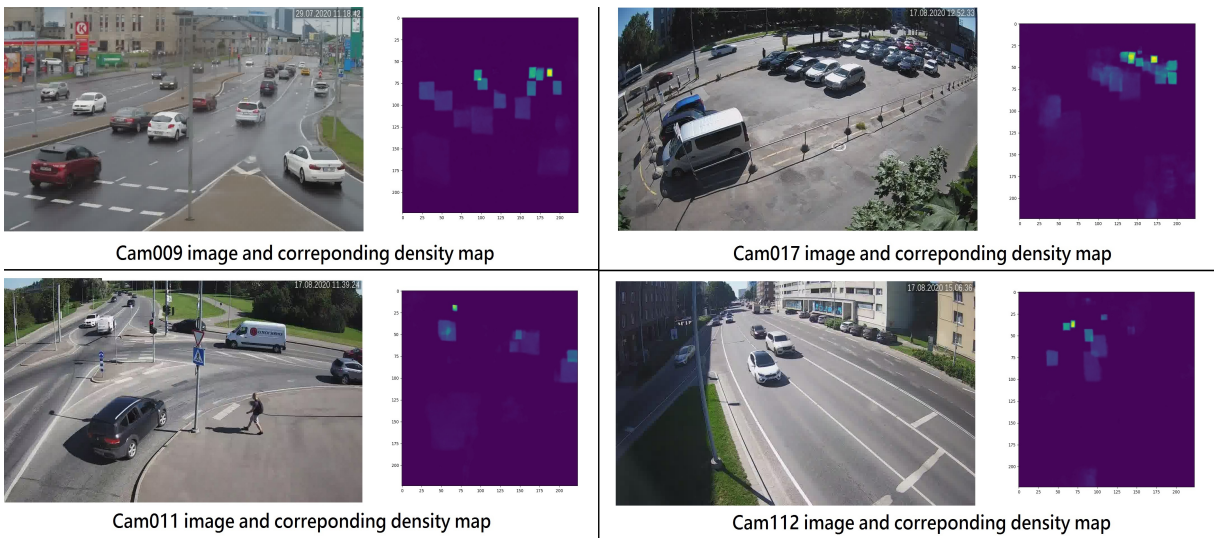

Figure 5.5: Example of inference on traffic images for four distinct cameras

not uniformly spaced. This meant that, when trying to apply a moving average directly to this oddly distributed data, a 30 minutes interval could contain either 5, 6 or 7 samples, which caused the results to be completely unreliable. So, to correct this, and to force all fixed size interval to have the same number of samples, the data was interpolated to obtain an evenly spaced collection of data points. This interpolation step was tested with three different methods: linear interpolation, quadratic spline and cubic spline. In some specific cases of poor distribution of datapoints, usually between two samples further away than usual or right after an abrupt change of curve tendency, both quadratic and cubic splines did bad approximations, generating values inconsistent with the problem at hand such as estimating vehicle count values of -10 (see Figure 5.6).

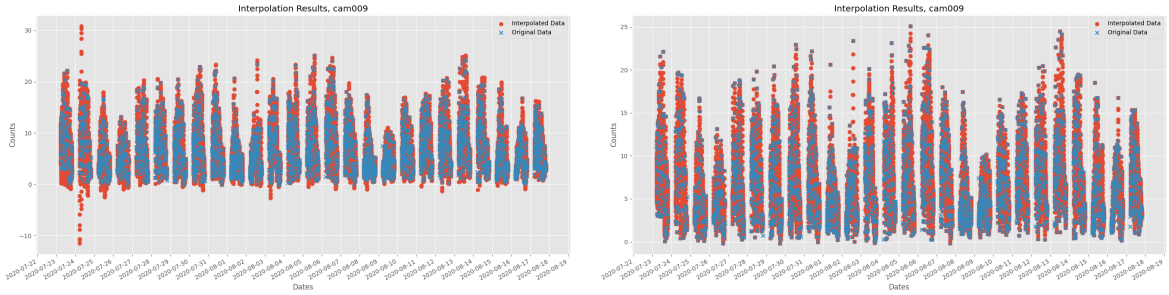

Figure 5.6: Effects of using cubic spline (on the left) and linear spline (on the right) on the same data

Since the collected data has a ”bumpy” nature, with many artifacts and non-linear spacing of samples, it was prone to this kind of errors. For this reason, we opted for the use of a linear spline interpolation. Although this method generated a data-curve with many acute angles, it was still more credible that the results obtained with the other two methods. After using linear spline interpolation to calculate the interpolating function, we selected a time interval of one minute to re-sample our data. Even though the original sampling rate had a mean of five minutes, in some extreme cases, two consecutive samples could be only one minute apart, so selecting a one minute interval should offer a more accurate representation of the original data.

The results of the interpolation mentioned above were polluted by artifacts caused by the large temporal spacing between sample acquisitions. Additionally, the traffic lights also influence the results of the acquisition, forming regular spikes in vehicle counts. This systolic pattern represents the difference of a higher flux of vehicles when a traffic light is green, versus the lower the stopped traffic caused by a red light. As the patterns we are more interested in are the ones that represent the overall flux of vehicles over time and not changes occurring within a small temporal window, we opted to apply a low pass filter to remove the influence of these shorter spanning variations. The chosen method was a moving average. The moving average can be thought of as a type of low pass filter where we don’t have control of the bandwidth besides choosing the number of samples. In this problem scenario, this translates as choosing the bandwidth by controlling the size of the temporal window analyzed. The moving average consists of selecting n samples, for example samples 0 through n, summing their values and dividing by the number of samples (see equation 4.1). This results in a single new average, which we placed the middle of the window. Then, the window is slid one sample forward and the process is repeated. As no

padding was used, the final number of samples obtained from applying the moving average is smaller than the number of samples this process started with. In fact, the number of samples after applying the moving average without padding is the number of samples that come as input, minus the number of samples contained in the used moving average window. We experimented applying moving averages with three different window sizes. We used windows of 15 minutes, 30 minutes and 60 minutes. The different results can be seen in Figure 5.7.

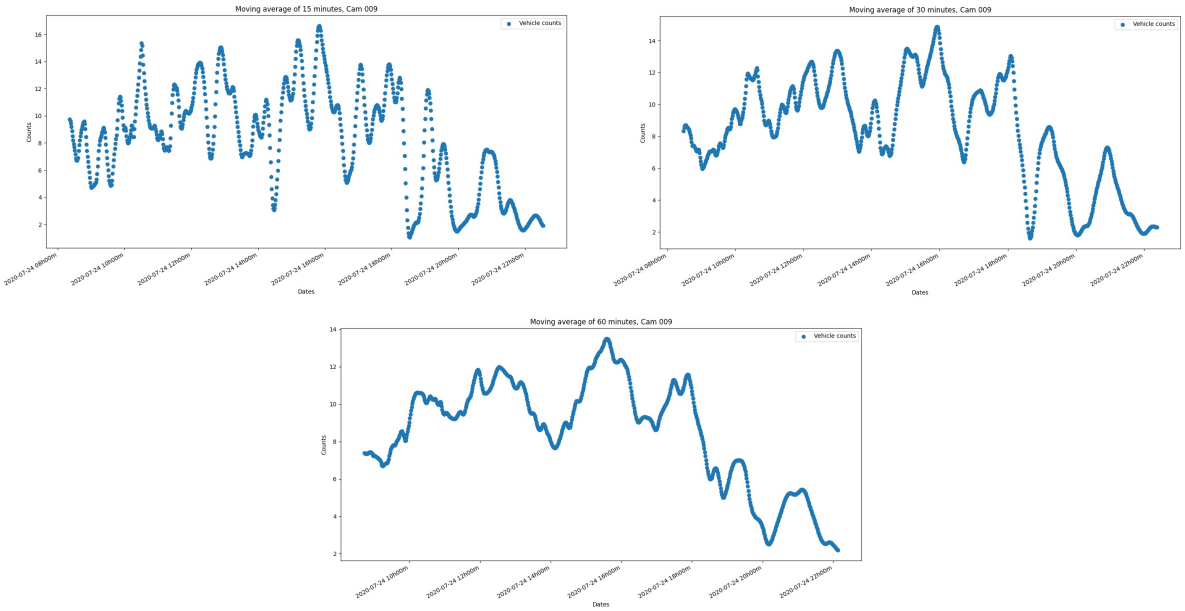

Figure 5.7: Applying a moving average with three different bandwidths to one day of vehicle counts. The camera shown is cam009 on the 24/07/2020. Top left is the result of using 15 minutes, top right is using 30 minutes and bottom is using 60 minutes

We chose the filter of 60 minutes as it produced smoother, more readable results. A larger window could be selected, but the loss of samples must also be a consideration, and using for example a window of two hours would lead to loosing two hours of samples after applying the moving average.

Then, data is separated by date such that instead of having one long array of data per camera that contains all of that camera's datapoints, we have multiple arrays, one for each day of each camera, that contains the sequence of vehicle counts obtained for that camera on that specific day. The visualization of the data in this format allows a better comparison of multiple distinct days as we can now plot them on the same figure, overlapping them. So, we tried plotting, for each camera, all of the acquired days, correctly identifying the day of the week of each day. The idea was to check if the days of the week had a static pattern. An example sample of this experiment can be seen in Figure 5.8.

Another test that was performed was to, instead of specifying the day of the week of each day, specify if a day was a business day or a weekend day. Sample results of this experiment can be seen in figure 5.8.

The following step in the data processing pipeline is to down-sample the data. As having one sample for every minute leads to a heavy processing load in the following steps, we down-sample the result of the moving average. The down-sampling rate chosen is half of the size of the window used when

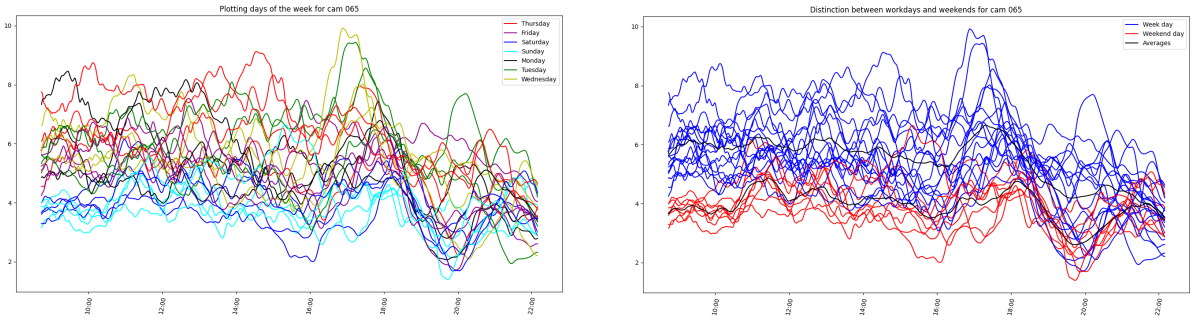

Figure 5.8: Moving averages segmented in days. On the plot on the left we specify for each day which day of the week it was captured on. On the plot on the right we specify for each day if was captured on a workday or weekend day. Both plots were based on cam065's captures.

calculating the moving average, for example, if a moving average with a window size of 60 minutes was used, we down-sample it by selecting one sample every 30 minutes. This down-sampling leads to a signal similar to the one seen in the results of the previous step (see Figure 5.7), the only difference being, the number of datapoints in the plot. An example can be seen in Figure 5.9.

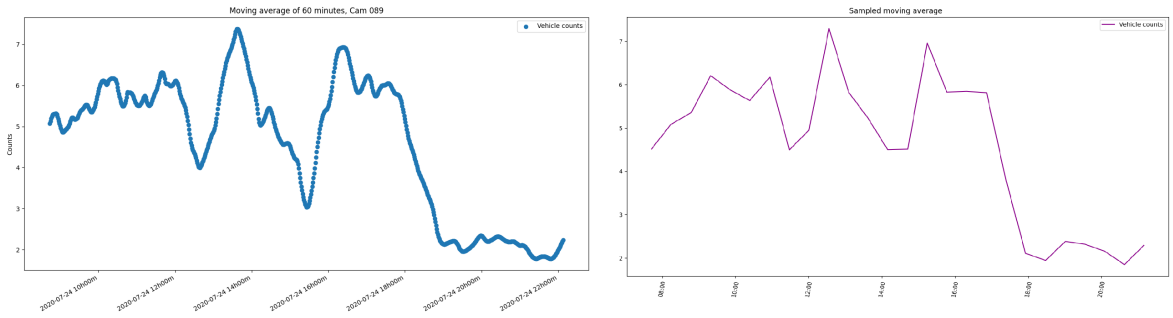

Figure 5.9: Sampling the results of applying moving average. Using cam 089 on 24/07/2020 after applying a moving average of 60 minutes. On the left are the results of applying the moving average and on the right is the sampled result (sampled every 30minutes).

All of the previous steps were implemented using Python3. The plots were generated using Matplotlib's Pyplot and Numpy. The results were saved in CSV files as matrices of size N by D, where N is the number of samples in a day, for one camera, and D is the number of days acquired for one camera. For each individual camera, two distinct matrices (with the same dimensions) were saved, one containing the vehicle count values, and the other containing the corresponding dates and times.

These CSV files are then read by a Matlab script. This Matlab script will read the data of each camera as a single matrix instead of reading it as multiple arrays, thus effectively concatenating the multiple days of vehicle counts horizontally. We call this matrix the daily vehicle counts descriptor of one camera, see Figure 5.10. Alternatively, multiple matrices from different cameras can be read simultaneously, generating a joint daily vehicle count descriptor of two cameras. These joint descriptors are the equivalent of concatenating two distinct descriptors horizontally. As the data was acquired in two parts (March and August) and they have different dimensions, they are provided separately and cannot be read simultaneously. One example of a daily vehicle count descriptor of each of these periods is shown in Figure 5.10.

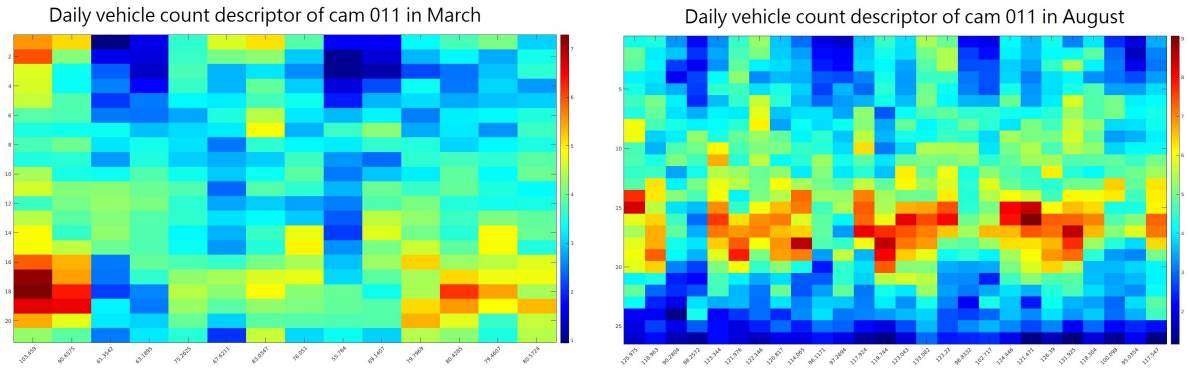

Figure 5.10: Daily vehicle counts descriptor of both acquisition periods (August on top and March at the bottom) for three cameras: cam 005 on the left, cam 017 (parking lot) in the middle, cam 123 on the right.

In this figure we can see how less days and less hours in each day were acquired during the acquisition period of March comparatively with August.

The daily vehicle count descriptors are fed to a Matlab script which calculates and returns its sparse subspace representation C. The subspace representation C is a sparse square matrix of size D by D, where D is the number of complete days in the read data, equivalent to the number of columns in the corresponding descriptor. For instance, if for example J cameras are read simultaneously, and the data of one camera contains K days, then D is J times K. Finally, the affinity matrix W, is calculated by adding matrix C to its transposed. The affinity matrix can be viewed as graph, thus allowing the application of spectral clustering. Both matrices C and W are shown in Figure 5.11.

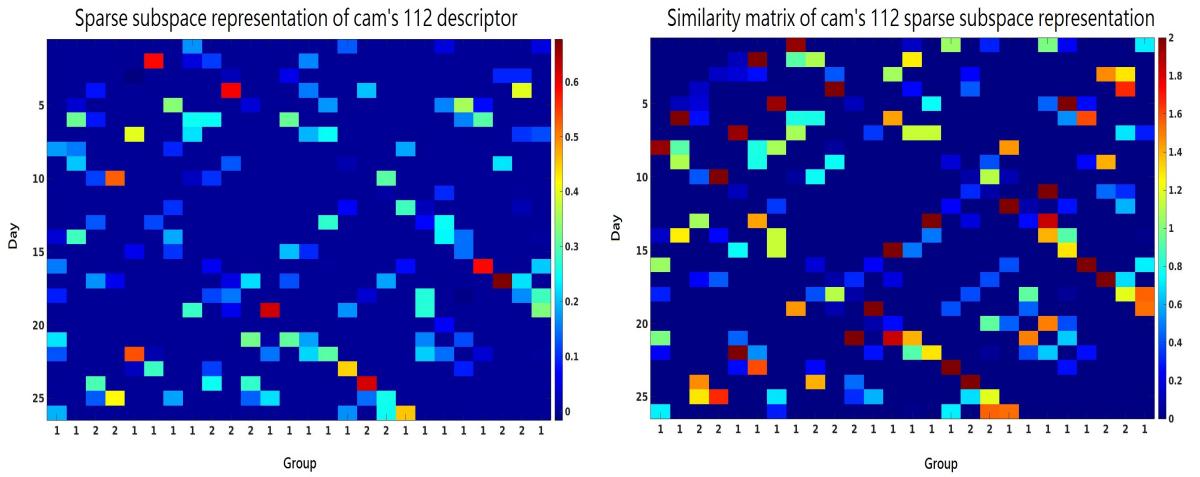

Figure 5.11: On the left: Sparse subspace representation (matrix C) of cam's 112 August daily count descriptor. On the right: similarity matrix (matrix W) of cam 112 in August.

Finally, spectral clustering is applied to the similarity matrix W. To run spectral clustering on this graph, the number of desired clusters is required. The result of this clustering is an array of size K (number of days) which in each entry contains the id of the cluster in which the day with the corresponding index was grouped.

5.3 Data Processing Results

In this section we analyse results of the process of transforming images to daily vehicle count descriptors. We briefly analyse the system's counting capabilities. But most of all, we focus on analysing this system's capacity of capturing the effects of real life events which might have impact on the overall traffic density of the city.

The results of the counting process are difficult to analyse as there is no ground truth to compare them to, so a visual analysis of the images and corresponding density map is needed to understand how the network performs in this application scenario. Since no kind of domain adaptation to the new point of view was performed, excellent results are not expected but instead a linear correlation between the real and estimated number of vehicles is expected. Some examples were created by manually counting the number of vehicles. We show two examples, one with many vehicles and another with none, captured by the same camera in Figure 5.12.

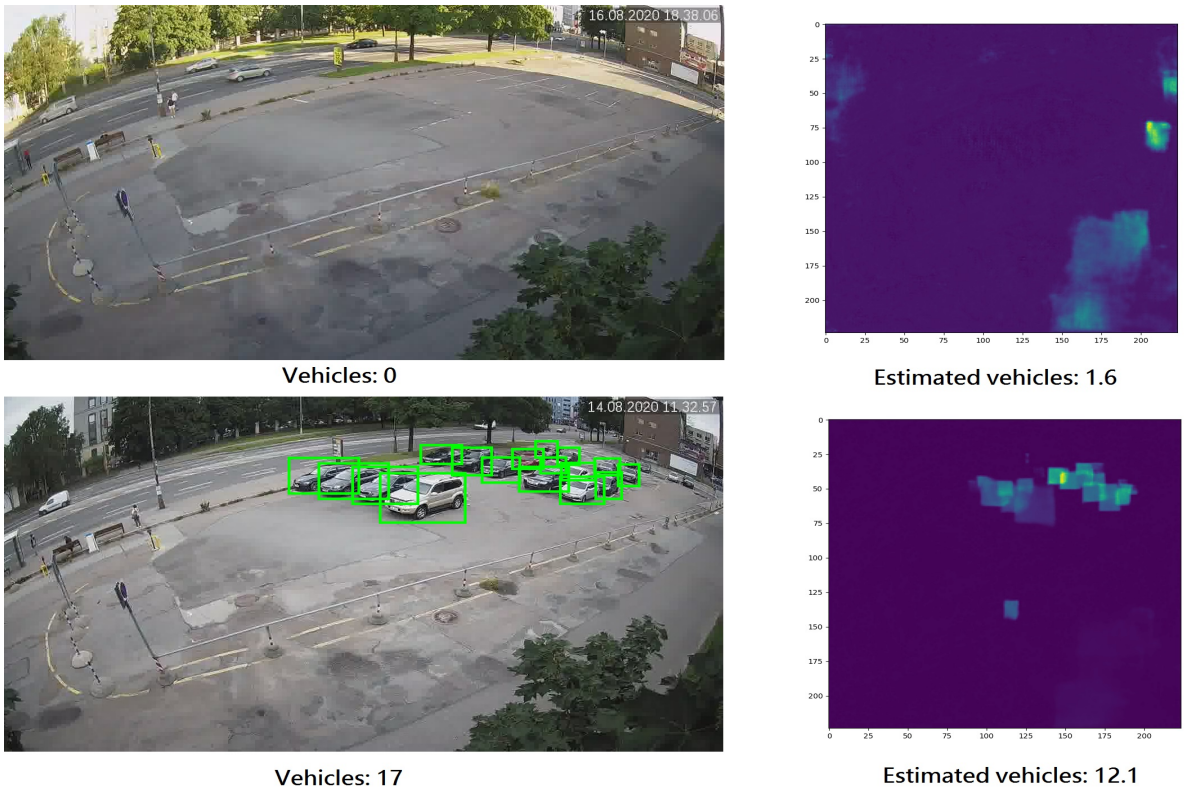

Figure 5.12: Example of density maps inferred using U-Net and respective estimation of number of vehicles (sum of pixel values). On top: image captured by cam 017 on 16/08/2020 with no vehicles. On bottom: image captured by cam 017 on 14/08/2020 with vehicles.

As expected, there is a considerable error, but the neural net is indeed capable of inferring the presence vehicles, and a correlation apparently exists between the real number of vehicles in the image and the number of vehicles estimated by the U-Net.

In the city of Tallinn, the government encourages people to leave their cars in parking lots built around the city and then use public transportation to move inside the city. This would generate a vehicle count plot, for a single day, which looks like a plateau: rising in the early hours of the day, staying stable

throughout the day and declining at the end of the afternoon. An example of a parking camera's vehicle count plot can be seen in Figure 5.13.

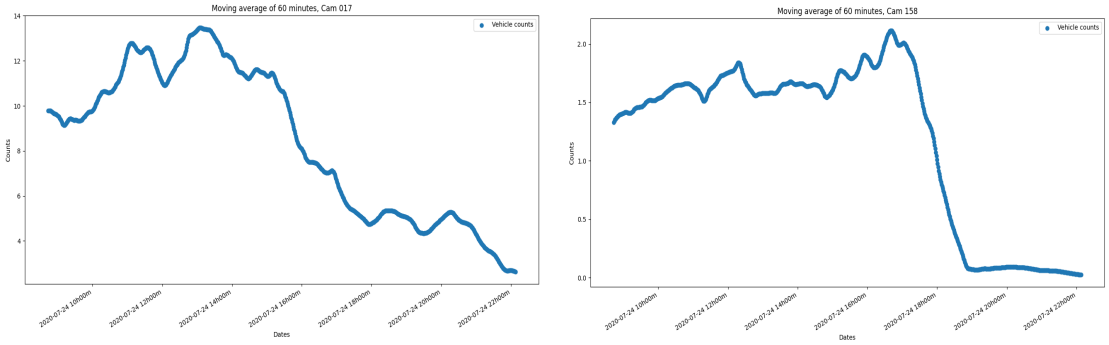

Figure 5.13: Moving Average of 60min for Cam017 (on the left) and for Cam158 (on the right) 24/07/2020

This plot shows a small increase in the number of vehicles in the morning (the camera started filming rather late), then a somewhat steady period until the afternoon and a decrease until the end of the afternoon. In the same figure (Figure 5.13), another example of a day's worth of vehicle counting by a parking lot camera can be seen. This second camera (cam158) captured less cars than cam017, but the pattern remains the same: a somewhat steady number of vehicles and a sudden drop in the afternoon ending at around the same time.

For cameras in streets and intersections, the plot should be more irregular in the sense that spikes in the number of vehicles are expected in the morning, lunch time and end of afternoon, when most of the circulation happens. But since the location of the chosen camera varies, including streets in the city center, others further away from the center, access roads and highways, the pattern should not forcibly be the same in all of these. To confer the location of the cameras, see Figure 5.2.
In Figure 5.14 we show the same day captured by cameras in different types of streets.

Even though these three plots do not resemble each other very much at first sight, they still have some elements in common. For example, all three of these plots seem to have in common a traffic density spike around 16:00h and another right after 18:00h. The second spike appears at around the same time as people usually finish working and start their return home. The camera closest to the city center is the only one out of the three to capture another spike at around 13:00h. This makes sense as at this time it is more expectable to have movement within the city than traffic flowing out or into the city. As mentioned in the previous chapter, a considerable difference in traffic density between a working day and a weekend day was noted. This difference was noted among all of cameras, with some having more accentuated differences and in other cases, some cameras where distinguishing a working day from a weekend day was harder. This can be seen in Figure 5.15.

If we look specifically into the images obtained in March, major differences are noted when comparing the moving averages of days before the quarantine was imposed, and after. The average traffic density is usually higher before the quarantine. This goes to show the effects of the quarantine were captured by this process. But again, the differences are mitigated in some cases. Notably, when using a camera

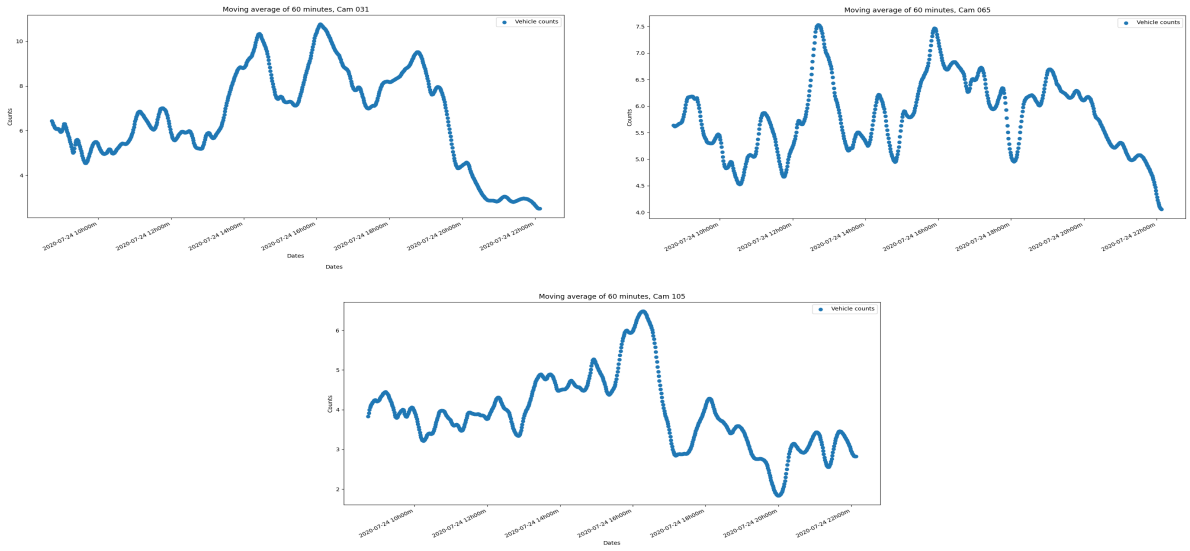

Figure 5.14: Moving Average of 60min for Cam031 (access street to the center of the city, top left), Cam065(close to the center of the city, top right) and Cam105 (on Parnu highway, bottom) on 24/07/2020

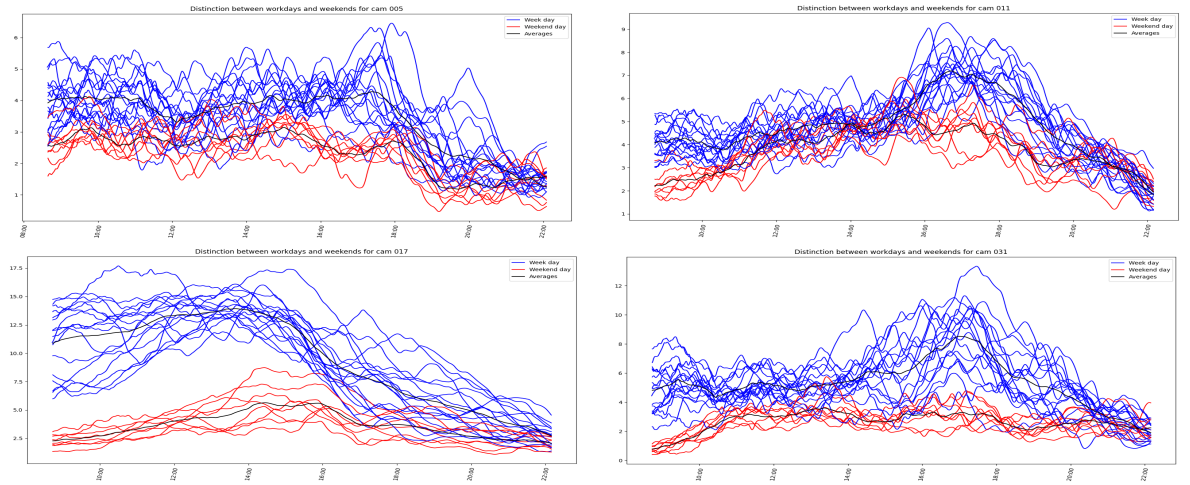

Figure 5.15: Comparison of workday and weekend day's moving average for Cam005 (close to the center of the city, top left), Cam009(Tartu highway, rop right), Cam017 (parking lot, bottom left), Cam031 (street access to the city center, bottom right) in August

which already did not capture many vehicles before the quarantine was implemented, the differences before/after quarantine are not so obvious.

Another way to visualize the processed data is through the use of the daily vehicle count descriptors (see Section 4.1). Because of the formatting of the data in these descriptors, we can better visualise the sequence of days and get a better sense of how a day's overall traffic density changes. An example with the daily cont descriptors of three cameras for both acquisition periods is shown in Figure 5.16.

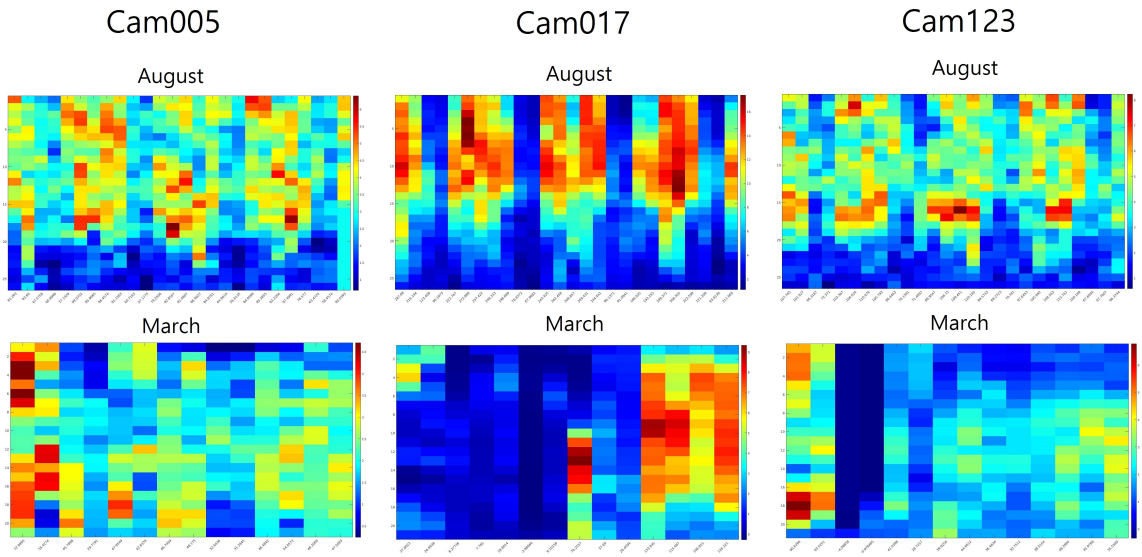

Figure 5.16: Daily vehicle counts descriptor of both acquisition periods (August on top and March at the bottom) for three cameras: cam 005 on the left, cam 017 (parking lot) in the middle, cam 123 on the right.

It is interesting to see in this figure the same results as in Figure 5.15 but with additional information. For example, in the period of August, the parking camera shows a steeper difference between workdays and weekends than the traffic cameras and how it seems that Saturdays usually have ever so slightly more traffic density than Sundays. Throughout the month, the pattern workday/weekend apparently remains the same. In the daily vehicle count descriptors of March, we note that the overall traffic density of a day tends to increase the further it is from the day quarantine was imposed. The shown parking camera's descriptor indicates that right before the quarantine there was already a reduction in traffic density, and here, traffic returns to normal after about a week and a half. In this camera's daily vehicle count descriptor (cam017) and in cam005 we also seem to perceive the influence of a weekend in traffic density, as traffic in two consecutive days is even lower than the traffic during the rest of the captured quarantine period.

5.4 Sparse Spectral Clustering Results

In this section we analyse the results obtained in the various steps of the sparse subspace clustering process. We will start by analysing and commenting some examples of sparse subspace representations, then move on to similarity matrices. At the end of this section we will also review a few clustering

results. All the example results provided in this Section were obtained using $L_e = 10^{-1}$ and $L_z = 10^{-1}$.

As was mentioned in Chapter 4 and Section 5.2, the daily vehicle count descriptors of the selected cameras are subjected to an algorithm which calculates and returns a sparse subspace representation of the corresponding descriptor. These sparse subspace representation matrices, or matrix C, are organized by columns. A deeper explanation of these matrices is provided in Section 4.2. To visualize the matrices C we display them as images with Matlab. We present two matrices C obtained with the daily vehicle count descriptors of individual cameras in Figure 5.17.

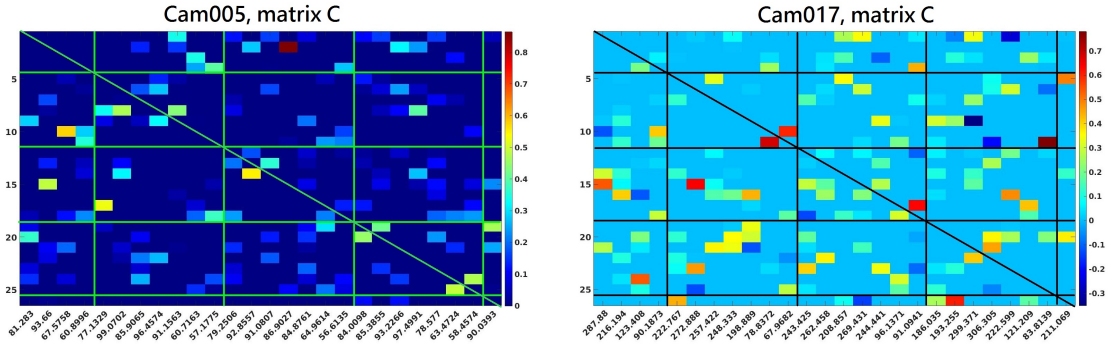

Figure 5.17: Sparse subspace representations (matrix C) of vehicle count descriptors in August. On the left is camera 005, a street camera. On the right is camera 017, a parking camera. Using different scales.

We chose a street camera and a parking camera and also drew lines to give a visual indication of where weeks start and end. These lines are always placed between Sundays and Mondays. The diagonal was also drawn to make it more evident. Below each column we placed the total number of cars detected in each day.

If all weeks shared a common pattern, that is, if all Mondays were alike, and Tuesdays all resembled each other, and so on, what we would see here would be the diagonals of the weeks with high reconstruction ratios, except for the diagonal which is forcibly set to zero. This would be an indication that every day of the week could be decomposed into a linear combination of every other corresponding day of the week, i.e Mondays, for example, could be recreated with a linear combination of all the other captured Mondays. Although these diagonals are not clearly solely populated with high ratio values, we do see that cells with a high reconstruction ratio do have a slight tendency to be closer to the diagonals. This may mean that week days which are closer to each, for example a Friday and Thursday even if not in the same week, have a tendency to resemble each other more than week days which are far apart. Another thing that is worth noting is the fact that most of the weekends are reconstructed using almost exclusively other weekends. There are of course a few exceptions but this seems to be a pattern throughout this period. It is also noteworthy that in the vehicle sums, displayed below each column, we can clearly notice that weekends have a total sum quite inferior when compared to the workdays.

We also show the sparse subspace representation of the same cameras in March in Figure 5.18.

First of all, since this acquisition was not continuous (there are a few missing days, see Figure 4.13) the lines drawn may not help identify all of the days of the week but they always help to identify weekends. Unlike the previous figure, in this one we don't see weekends being exclusively reconstructed with other

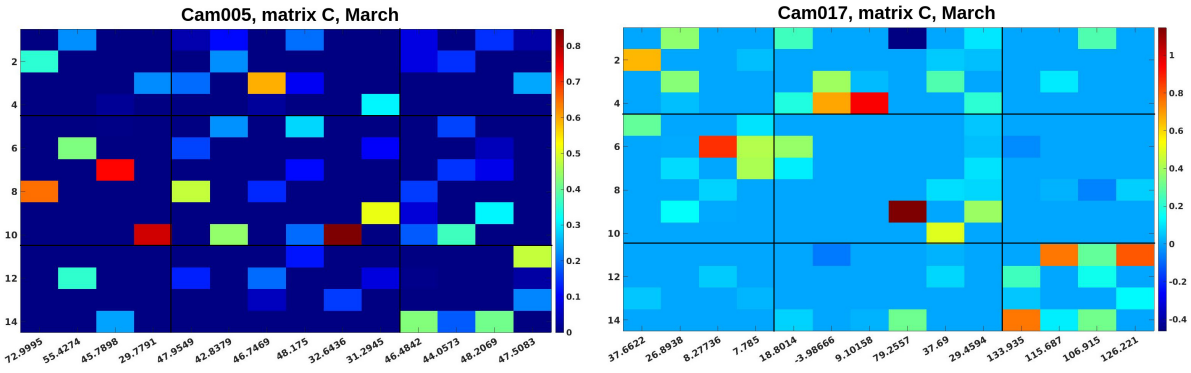

Figure 5.18: Sparse subspace representations (matrix C) of vehicle count descriptors in March. On the left is camera 005, a street camera. On the right is camera 017, a parking camera. Using different scales.

weekends. In camera 005, the first weekend's Saturday is reconstructed using mostly a Friday from the second acquired week. The other weekend days do seem to follow the pattern of being reconstructed with other weekends. As for camera 017, if we look at its descriptor in Figure 5.16, we can see that the last week is quite different from the others as it contains much more traffic density than the remaining weeks in this period. This is reflected here in Figure 5.18 as it is reconstructed using almost exclusively itself.

Next, in Figure 5.19 we show two sparse subspace representations of joint descriptors. The first one is a joint descriptor of two traffic cameras, and the second is a joint descriptor of a traffic camera and a parking camera.

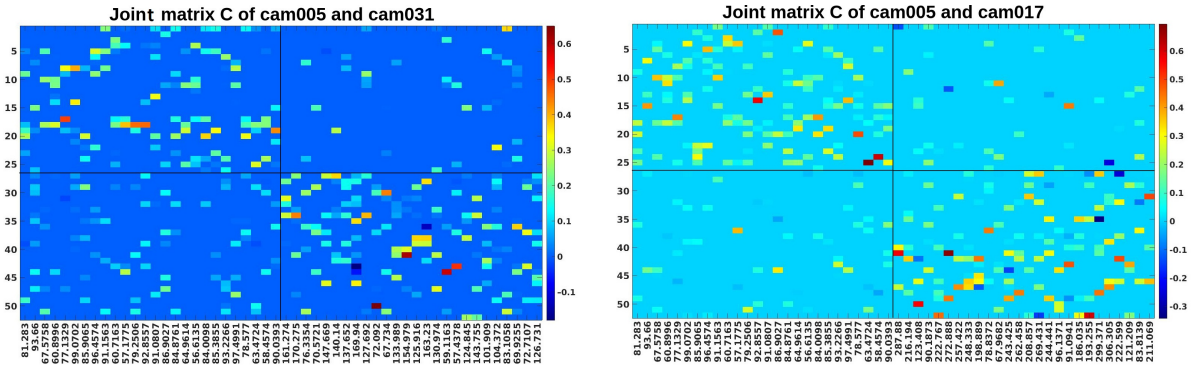

Figure 5.19: Sparse subspace representations (matrix C) of joint vehicle count descriptors in August. On the left is camera 005 and cam031, both street cameras. On the right is camera 005 and camera 017, a street camera and a parking camera. Using different scales in each image.

We drew lines separating the images into quadrants. These lines, drawn at the end of column 26 and row 26, give us an visual indication of which days belong to each camera. For example in the image on the left, the quadrant which includes columns 1 through 26 and rows from 27 through 52 shows which days from cam031 are used in the reconstruction of the days acquired for cam 005. The first thing we spot in both images of Figure 5.19 is that two of the quadrants in each seem to contain all of the nonzero values while the other two are almost empty. This means the days of each camera are mostly reconstructed with other days from the same camera, maybe because the patterns captured

by each camera are too distinct from each other. Two examples were given so that we are able to check if two street cameras influence each other more than a street camera and a parking camera. A slight difference can be spotted, it seems that the combination of camera 005 and 031 have more inter-dependency than cameras 005 and 017 as on the bottom left quadrant of the first image, there are still a considerable amount of non-zero reconstruction ratios. Although the difference exists, it is minimal, and we can conclude that the traffic patterns captured by the cameras are not very similar in general. The same happened when trying other camera combinations (not shown here).

The similarity matrices can be regarded as graphs obtained by adding matrix C to itself transposed. Unlike matrix C, the similarity matrix W is symmetric, such that day i is as similar to day j as day j is to day i. To be able to establish a comparison with their respective matrices C, we used in Figure 5.20 the same cameras as we used in Figures 5.17, 5.18 and 5.19.

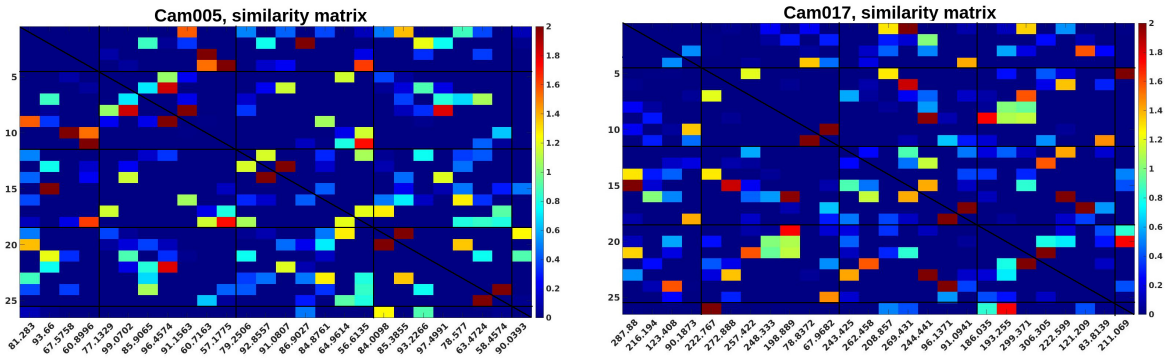

Figure 5.20: Similarity matrices of camera 005 (on the left), a street camera, and camera 017 (on the right), a parking camera, in August.

These similarity matrices resemble the sparse subspace representations shown in Figure 5.17 with the exception that since now the sum over a column is not restricted to 1. Overall the same comments that were done to that figure can be made to this figure. That is, weekends are mostly similar to other weekends and in general, the closer two days are within a week, the more similar they seem to be.

The similarity matrices for these two cameras in March can be seen in Figure 5.21.

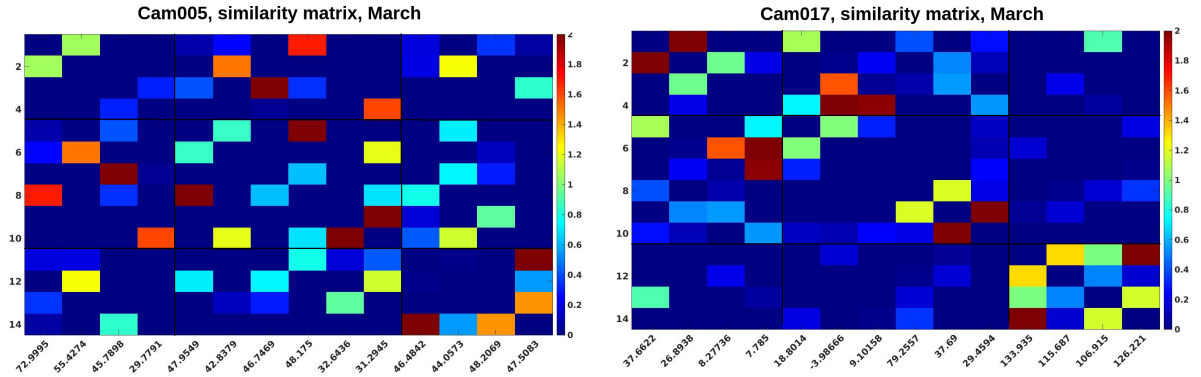

Figure 5.21: Similarity matrices of camera 005 (on the left), a street camera, and camera 017 (on the right), a parking camera, in March.

Although we expected to see a strong similarity between the first two days as these were the only

to be captured before the quarantine was put into effect, but this is not happens. These two days, a Tuesday and a Wednesday, are most similar to a Thursday and a Wednesday respectively.

Finally, in Figure 5.22, we show similarity matrices corresponding to the sparse subspace representations shown in Figure 5.19.

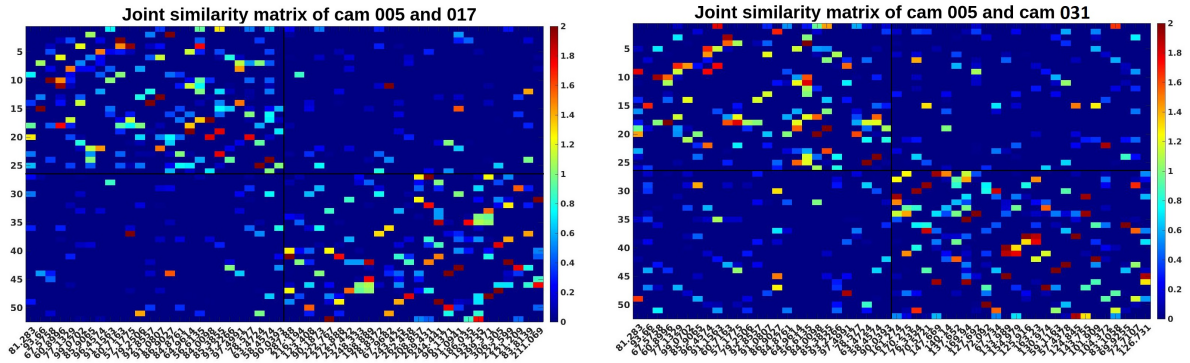

Figure 5.22: Joint similarity matrix (matrix W) of pairs of cameras in August. On the left is camera 005 and camera 017, a street camera and a parking camera. On the right is camera 005 and cam031, both street cameras. Using different scales in each image.

We confirm in this figure the conclusions made in Figure 5.19, the two street cameras are more similar with each other than a street camera and a parking camera.

To show some results of reconstructions, we created plots where we show for randomly selected days in a few daily vehicle count descriptors, the two days which are most similar to the selected one and the reconstruction error. With this plot we hope to understand if the days which are most similar do indeed have similar vehicle count plots, and if the reconstruction error reflects their similarity.

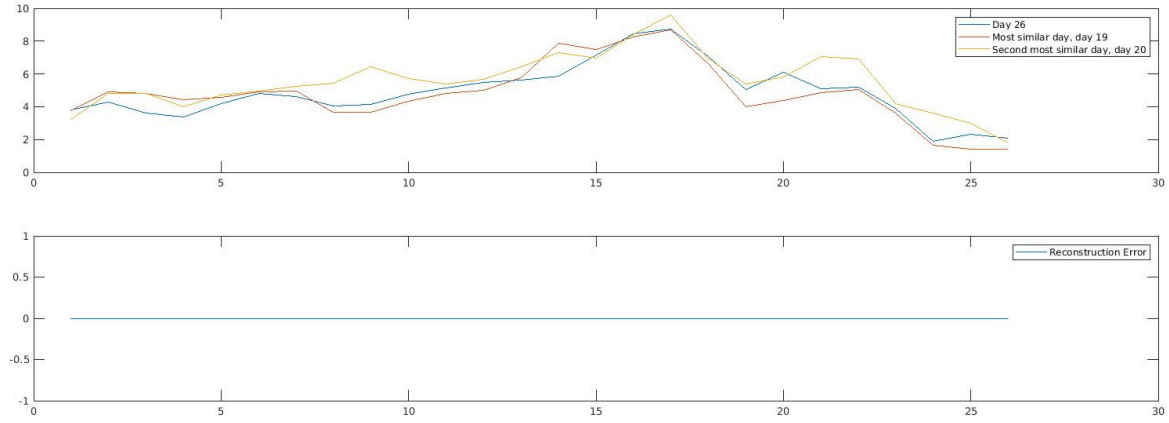

Figure 5.23: Plotting camera 031's vehicle counts. More specifically, day number 26 and its two most similar days. Below is the reconstruction error

In the example shown in Figure 5.23, the two most similar days (day 20, a Monday and day 21, a Tuesday) follow the same trend as the selected day (day 26, a Monday), this is a good example of three distinct days having the same base traffic density pattern, and thus, the reconstruction error is zero at all times. The slight differences which may exist in the reconstruction are considered white noise. But this is not always the case, and to showcase what happens when a day's pattern is too different from all

the others we have Figure 5.24.

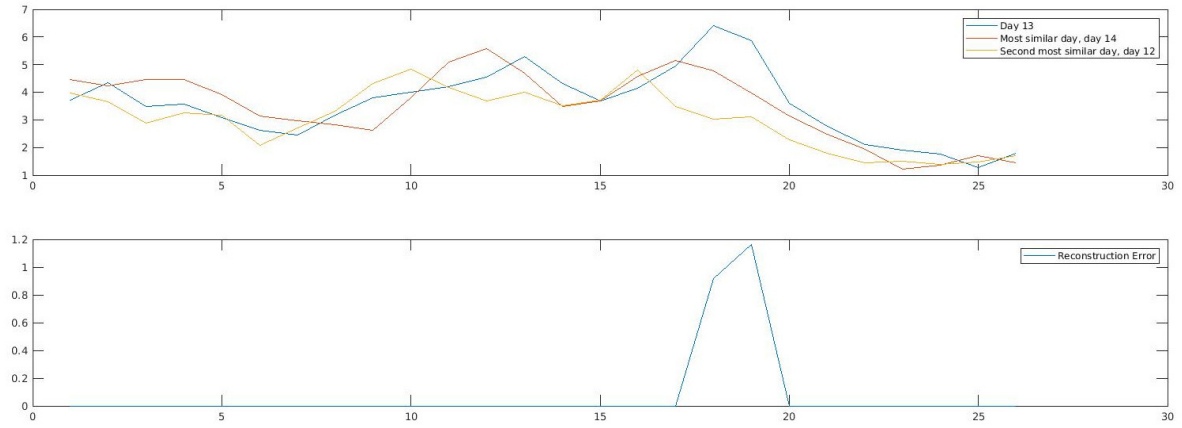

Figure 5.24: Plotting camera 031's vehicle counts. More specifically, day number 13 and its two most similar days. Below is the reconstruction error

In this figure the selected day, day 13, corresponds to Tuesday the 4th of August and its two most similar days are Wednesday the 5th of August and Monday 3rd of August respectively. This Tuesday has an unusually high peak of traffic density which is not matched by its most similar days. As such, when reconstructing it, a reconstruction error exists at the same time as the peak.

Finally, since the classification results are essentially arrays containing the group of each day, to analyze the results of the spectral clustering, we proceeded to plot all the days in each descriptor and attributing a color in the plot for each group. As spectral clustering requires us to select the target number of clusters, we started by choosing two clusters, and so, we have two colors in each plot. Additionally, for visualization purposes, we plot the average value of each group. An example of this kind of plot can be seen in Figure 5.25.

In this figure, we applied spectral clustering to the similarity matrix obtained from the vehicle count descriptor. The similarity matrix used to obtain this classification is the image on the left showed in Figure 5.20. The first thing we notice is that the two groups seem to be mostly separated by the amplitude of the curves: the red group has the same overall shape as the blue group but generally higher vehicle count values. This is corroborated by the averages plot. Both averages plots have vaguely the same appearance but one has higher vehicle counts than the other. This is an indication that spectral clustering mostly classified based on daily vehicle sum. So if we go back to Figure 5.20, where we see the sums at the bottom, we can already get an idea of how the days will be classified. Another interesting thing is that in both groups averages we can see a sharp decline at around the same time, which in reality corresponds to 18:00h (see Figure 5.16). When comparing the plot shown in Figure 5.16 with the one shown in Figure 5.25, we can spot similarities. Since in general the days with less traffic correspond to weekends, it seems that this classification ends up grouping the days into workdays and weekends, with some exceptions as other workdays days with less traffic density will be grouped together with weekends and weekends with higher traffic density will be grouped with the other workdays.

Another plot is shown in Figure 5.26 where we show the results of spectral clustering applied to the

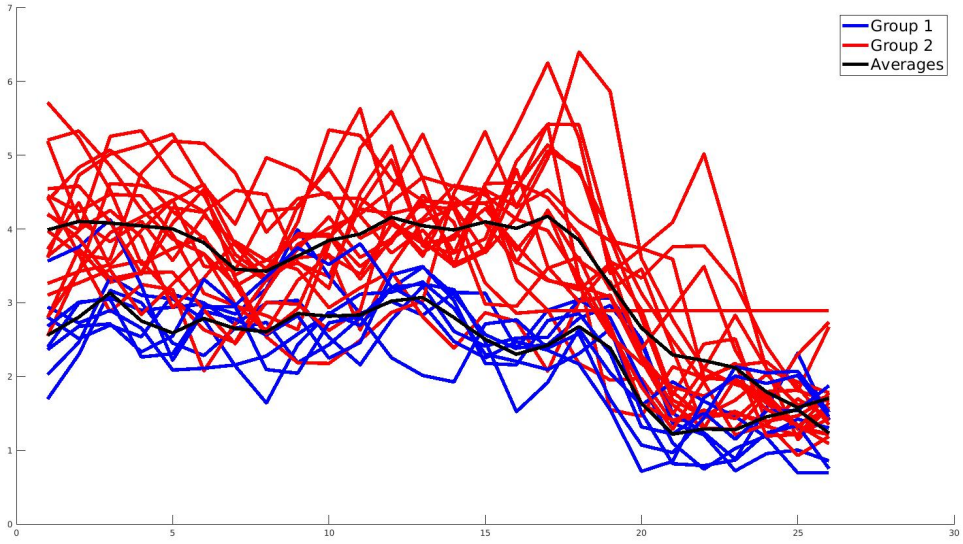

Figure 5.25: Plotting camera 005's vehicle counts in August. Using classification obtained from spectral clustering.

joint similarity matrix of camera 005 and camera 031 presented in Figure 5.22.

Similarly to Figure 5.25, it seems days were mostly classified based on the amplitude of the curve, which corresponds to the number of vehicles detected by each camera. When we compare the number of vehicles detected by each of the two cameras used here, it is clear that one of them, camera 031, detected more vehicles than camera 005. So, confirming the reasoning of the comments made about Figure 5.19, we expect this day classification to be based mostly on which camera it belonged to.

To help understand which days fell into each group, we show again the vehicle count descriptor, but instead of having the sum of vehicles below each column, we show the group into which that day was classified. This similarity matrix is the same as the image on the right presented in Figure 5.22.

What this figure shows is that most of the days obtained from camera 005 are classified into group 2, and most of the days from camera 031 are classified into group 1. The exceptions seem to be days with exceptionally high or low vehicle sum in comparison with the rest of the acquisition of their respective camera. For instance, in camera 031, weekends were classified into group 2.

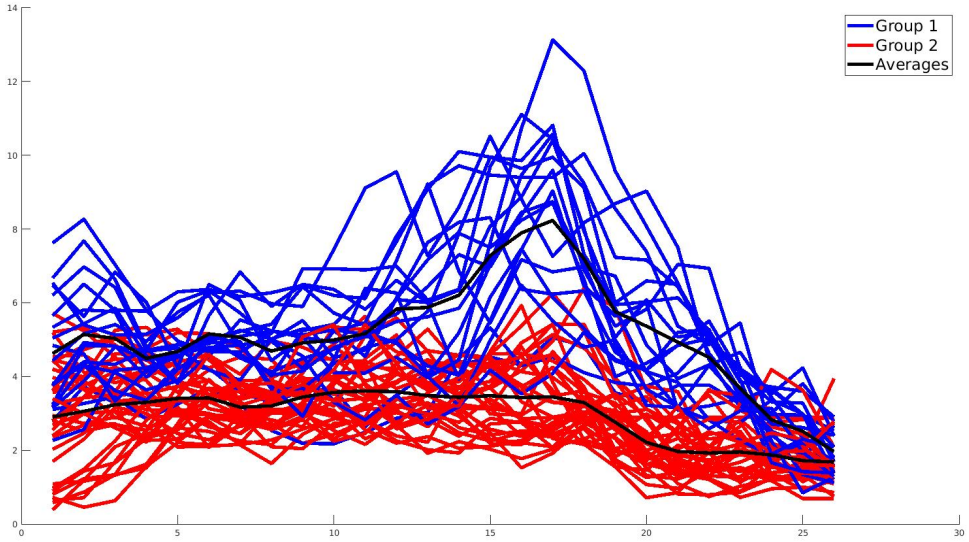

Figure 5.26: Plotting camera 005 and camera 031's vehicle counts in August. Using classification obtained from spectral clustering.

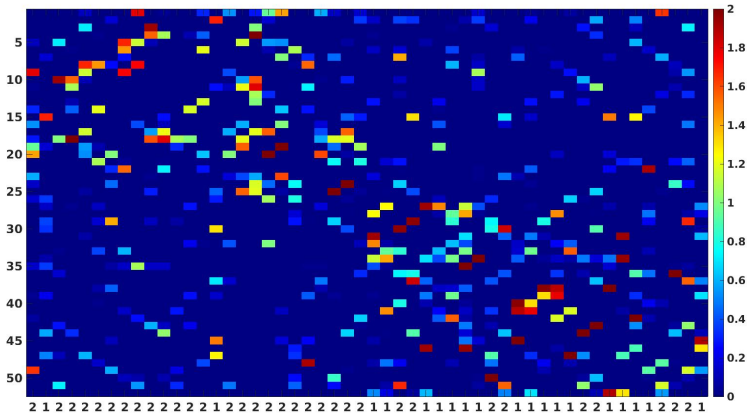

Figure 5.27: Plotting camera 005 and camera 031's similarity matrix in August. Using classification obtained from spectral clustering presented below each column.

Chapter 6

Conclusions

In this chapter we present the major achievements attained in this work and a few ideas for the future which would help refine the results of the present work.

6.1 Achievements

In this work, we managed to build a pipeline which transforms a sequence of traffic images into vehicle counts and propose a classification of that same sequence based on traffic patterns. We achieved this by first transforming each image to an estimate of the number vehicles contained in it through the use of a neural network. Then a pre-processing step was used to make these vehicle counts usable. The processed vehicle counts were used in an optimization program which, based on the self-expressiveness of the data, calculates a sparse subspace representation of the data. This is achieved by expressing each time segment of vehicle counts as a sparse linear combination of other time segments, thus, every segment can be reconstructed using the others. The sparse subspace representation itself is the collection of all the ratios used in the linear combination. In our work, the sparse subspace representation consists of a 2-D matrix of size $N \times N$ where N is the number of segments of image acquisition that is under analysis. Using the sparse subspace representation it is possible to obtain the so-called similarity matrix. A similarity matrix is a symmetric matrix which establishes degrees of similarity between our acquisition time segments. With such a tool, we can obtain an estimate of traffic similarity between, for example, two arbitrary days, in a principled manner. The similarity matrix can also be viewed as an undirected graph, where each node is a time segment, and the similarity value between two days is the weight of the edge connecting those two nodes. Such a representation opens the possibility of applying spectral clustering to the data. At the end of all these steps, we can effectively say that we are grouping days of traffic images based on a notion of similarity. As a byproduct, we also define a method of describing the traffic pattern captured by a camera in a time segment as a linear combination of other time segments of the same dimensions. Since not all time can be correctly reconstructed with a linear combination of others, the error is stored in a sparse error matrix. As this matrix captures events which were not previously observed, it could serve the purpose of detecting anomalies in traffic patterns.

6.2 Future Work

This work is open for further development in most of its parts, namely the dataset, the estimation of the number of vehicles in a traffic image, the pre-processing which precedes the Sparse Subspace Clustering algorithm and the clustering method.

The dataset employed in the case-study was relatively small, it comprised a total of forty complete days and included the school holiday season. This work could benefit from a larger dataset which also included acquisition during non-holiday or quarantine periods, although the latter is interesting in on its own. An dataset obtained during a regular working week could serve as a baseline for comparison with other acquired periods. The time interval between sample images could also be shorter which would yield more accurate information regarding traffic patterns and make the data smoother.

As was mentioned in Section 5.3, no domain adaptation was used in our neural network which was trained on different cameras than the ones used in the case-study. This meant that although the current method produced interesting results, these were populated with white noise which led to the need for additional pre-processing. Results could be improved if some form of domain adaptation was used or for example if the network had been trained on the same cameras that acquired the dataset in Chapter 5.

Further pre-processing could be implemented to allow a better comparison of traffic patterns. Although Sparse Subspace Clustering is invariant to scaling, we felt that the results had a tendency to classify days by total volume of traffic. This may be due to the fact that the shape of the vehicle count curve is indeed different, but it could be interesting to be able to group together days which have spikes in the same time-frame, even if the spikes differ in amplitude or duration.

Another point that could be improved in the pre-processing would be to make the process more invariable to small time-shifts. The results of the process as-is are very sensitive to time, if two days are exactly the same but shifted by thirty minutes, then their similarity metric should be close to zero as it is impossible to reconstruct the first one with the second one without correcting the time shift. Since traffic is susceptible to these changes in the time domain, it is possible that this process is not considering important similarities due to time shifts. Accounting for this possibility could make the results more consistent and the process more robust.

Finally, other clustering algorithms can be tested and their results compared in order to understand the benefits and shortcomings of each of them in this specific scenario.

Bibliography

[1] A. Daubaras and M. Zilys. Vehicle detection based on magneto-resistive magnetic field sensor. *Electronics and Electrical Engineering*, 118, 02 2012. doi: 10.5755/j01.eee.118.2.1169.

[2] J. Garcia Arnal Barbedo. A review on methods for automatic counting of objects in digital images. *IEEE Latin America Transactions*, 10(5):2112–2124, 2012. doi: 10.1109/TLA.2012.6362356.

[3] C. S. Asha and A. V. Narasimhadhan. Vehicle counting for traffic management system using yolo and correlation filter. In *2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pages 1–6, 2018. doi: 10.1109/CONECCT.2018.8482380.

[4] W. Xie, J. A. Noble, and A. Zisserman. Microscopy cell counting and detection with fully convolutional regression networks. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 6(3):283–292, 2018. doi: 10.1080/21681163.2016.1149104. URL `https://doi.org/10.1080/21681163.2016.1149104`.

[5] K. Djerriri, M. Ghabi, M. S. Karoui, and R. Adjoudj. Palm trees counting in remote sensing imagery using regression convolutional neural network. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 2627–2630, 2018. doi: 10.1109/IGARSS.2018.8519188.

[6] V. Lempitsky and A. Zisserman. Learning to count objects in images. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, NIPS'10, page 1324–1332, Red Hook, NY, USA, 2010. Curran Associates Inc.

[7] L. Ciampi, C. Santiago, J. P. Costeira, C. Gennaro, and G. Amato. Unsupervised vehicle counting via multiple camera domain adaptation. In *NeHuAI@ECAI*, 2020.

[8] R. Szeliski. *Computer Vision:Algorithms and Applications*. Springer, 2010.

[9] L. Wang and N. H. C. Yung. Crowd counting and segmentation in visual surveillance. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 2573–2576, 2009. doi: 10.1109/ICIP.2009.5413919.

[10] M. Moorman and A. Dong. Automated viral plaque counting using image segmentation and morphological analysis. In *2012 IEEE International Symposium on Multimedia*, pages 157–160, 2012. doi: 10.1109/ISM.2012.38.

[11] Y. Chen, X. Liu, and M. Yang. Multi-instance object segmentation with occlusion handling. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3470–3478, 2015. doi: 10.1109/CVPR.2015.7298969.

[12] A. N. Marana, S. A. Velastin, L. F. Costa, and R. A. Lotufo. Estimation of crowd density using image processing. In *IEE Colloquium on Image Processing for Security Applications (Digest No: 1997/074)*, pages 11/1–11/8, 1997. doi: 10.1049/ic:19970387.

[13] D. Kong, D. Gray, and Hai Tao. A viewpoint invariant approach for crowd counting. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 1187–1190, 2006. doi: 10.1109/ICPR.2006.197.

[14] Siu-Yeung Cho, T. W. S. Chow, and Chi-Tat Leung. A neural-based crowd estimation by hybrid global learning algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(4):535–541, 1999. doi: 10.1109/3477.775269.

[15] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651, 2017. doi: 10.1109/TPAMI.2016.2572683.

[16] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012. doi: 10.1145/3065386.

[17] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. doi: 10.1109/CVPR.2015.7298594.

[18] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *International Conference on Learning Representations (ICLR) (Banff)*, 12 2013.

[19] H. Yanagisawa, T. Yamashita, and H. Watanabe. A study on object detection method from manga images using cnn. In *2018 International Workshop on Advanced Image Technology (IWAIT)*, pages 1–4, 2018. doi: 10.1109/IWAIT.2018.8369633.

[20] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3376–3385, 2015. doi: 10.1109/CVPR.2015.7298959.

[21] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 2013. doi: 10.1109/TPAMI.2012.231.

[22] S. Gupta, R. B. Girshick, P. Arbelaez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. *CoRR*, abs/1407.5736, 2014. URL `http://arxiv.org/abs/1407.5736`.

[23] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL http://arxiv.org/abs/1506.02640.

[24] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

[25] J. Lin and M. Sun. A yolo-based traffic counting system. In *2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 82–85, 2018. doi: 10.1109/TAAI.2018.00027.

[26] G. Oltean, C. Florea, R. Orghidan, and V. Oltean. Towards real time vehicle counting using yolo-tiny and fast motion estimation. In *2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pages 240–243, 2019. doi: 10.1109/SIITME47687.2019.8990708.

[27] S. Zhang, G. Wu, J. P. Costeira, and J. M. F. Moura. Understanding traffic density from large-scale web camera data. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4264–4273, 2017. doi: 10.1109/CVPR.2017.454.

[28] M. Y. Choong, L. Angeline, R. K. Y. Chin, K. B. Yeo, and K. T. K. Teo. Vehicle trajectory clustering for traffic intersection surveillance. In *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pages 1–4, 2016. doi: 10.1109/ICCE-Asia.2016.7804776.

[29] C. Deng, F. Wang, H. Shi, and G. Tan. Real-time freeway traffic state estimation based on cluster analysis and multiclass support vector machine. In *2009 International Workshop on Intelligent Systems and Applications*, pages 1–4, 2009. doi: 10.1109/IWISA.2009.5073027.

[30] M. T. Asif, J. Dauwels, C. Y. Goh, A. Oran, E. Fathi, M. Xu, M. M. Dhanya, N. Mitrovic, and P. Jaillet. Spatiotemporal patterns in large-scale traffic speed prediction. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):794–804, 2014. doi: 10.1109/TITS.2013.2290285.

[31] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press. URL https://projecteuclid.org/euclid.bsmsp/1200512992.

[32] G. Hamerly and C. Elkan. Alternatives to the k-means algorithm that find better clusterings. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, page 600–607, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581134924. doi: 10.1145/584792.584890. URL https://doi.org/10.1145/584792.584890.

[33] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, USA, 1981. ISBN 0306406713.

[34] K. Zhou, C. Fu, and S. Yang. Fuzziness parameter selection in fuzzy c-means: The perspective of cluster validation. *Science China Information Sciences*, 57:1–8, 11 2014. doi: 10.1007/s11432-014-5146-0.

[35] Z. Cebeci and F. Yıldız. Comparison of k-means and fuzzy c-means algorithms on different cluster structures. *Journal of Agricultural Informatics. 2015 Vol. 6, No. 3 journal.magisz.org*, 6:13–23, 10 2015. doi: 10.17700/jai.2015.6.3.196.

[36] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781, 2013. doi: 10.1109/TPAMI.2013.57.

[37] C. of Tallinn. Tallinn cam network, 2020. URL `https://ristmikud.tallinn.ee/index.php/cams`.

[38] A. Petrov. Urllib3, 2020.