Novel Classifier for Smart Traffic Network

Submitted June 2020, in partial fulfillment of
the conditions for the award of the degree **Bachelor of Computer Engineering/Computer Science.**

Thomas Courtney

3175353

Supervised by Kaushik Mahata ddd

School of Engineering

University of Newcastle

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature _____

Date _____ / _____ / _____

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Economies of the world have been industrializing unceasingly for the last two centuries. The consequent advent of machinery and the factory have seen a great migration of humanity from rural fields to urban cities. Such concentrations of people have posed infrastructural challenges leading to many iconic innovations like the skyscraper, subway train and something. One metropolitan trait, however, continues to cause grief for millions of urbanites every day, traffic.

Road transport contributes 16.5% of global CO2 emissions [1] and costs the US $305 billion of productivity per year alone [2] [3]. Any small improvement in this situation will yield great benefit to society and the future of the planet.

There is no single solution to the problem of traffic congestion, whether it be the building of tunnels, the conversion of traffic lights to roundabouts, the widening of roads or the narrowing of roads [4]. The integration of several lightweight and inexpensive techniques is a far more attractive and effective approach [4]. A Smart City is such an initiative [5], where in IoT devices are deployed in an urban area and used to collect data. This technology could be applied to collect traffic data and use it inform the nature of investment in a traffic network.

A versatile and inexpensive method of data collection in a traffic network is computer vision. This technique requires as input only raw images of a traffic network and could therefore piggyback onto existing surveillance cameras. The intention is to mimic the ability of human vision to rapidly identify distinct objects from the real world. A humans ability to separate a vehicle from its surroundings and track its movement is one example of this.

The implementation of a robust computer vision algorithm for traffic monitoring is not trivial. The dependence on only one feature in an image will not yield accurate results, however the combination of several different features will, features like object edges, object area, hue, contrast and reflectance. There are a great many and varying number of ways in which to infer meaning from these features. Whatever method of implementation is selected must provide consistent accuracy, and in the case of traffic monitoring, real time results.

Therefore, it is the objective of this report to explain the design and operation of a novel computer vision algorithm that can effectively count and estimate the speed of vehicles in traffic, in real-time, given raw images of a traffic network.

Chapter 2

Literature Review

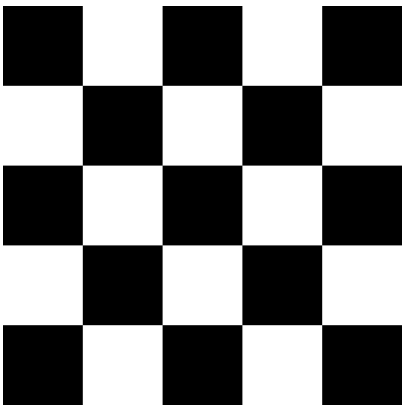
This section of the report expands upon the theory behind the design of the system. Digital image processing and computer vision are a broad fields and so only concepts relevant to this system are detailed.

2.1 Digital Images

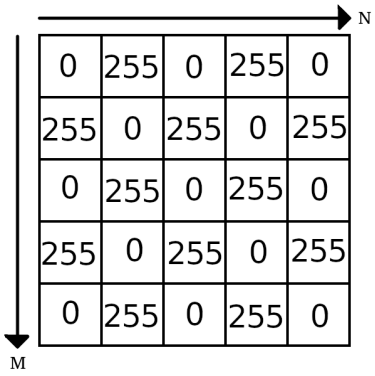
Unlike its analog predecessor the digital image consists of a discrete number of discrete valued data points called pixels. An image's resolution is defined how many pixels are in it and a pixel's value determines its colour. A pixel's value is also called its *intensity* and lies within a spectrum of values, for example an 8-bit image can have pixel values $[0, 255]$.

Digital images are often represented as a 2D array of values where each cell corresponds to an intesnity. In software this is how digital images are stored and manipulated. The image's array is $M\ pixels \times N\ pixels$ in size. Notice that in Figure 2.1, the lowest value is 0 (black) and the highest value is 255 (white). All other color that can be represented by 8-bits are between these two values.

The 2D array representation allows pixels to be referenced by their relative position in



(a)



(b)

Figure 2.1: (a) A Simple 8-bit Image (b) Array Representation of Image.

the M and N directions. Furthermore, M and N may be substituted for something more familiar like the x and y-axis of a Cartesian plane. In fact, a digital image is just a two dimensional function (Equation 2.1), where each pixel is described by a coordinate (x,y) and an intensity (z) at that point.

$$z = f(x, y) \tag{2.1}$$

This means that two dimnesional operations may be applied to a digital images. As a consequence images may be modelled as 3D surfaces. This is useful when trying to visualize rates of change in an image.

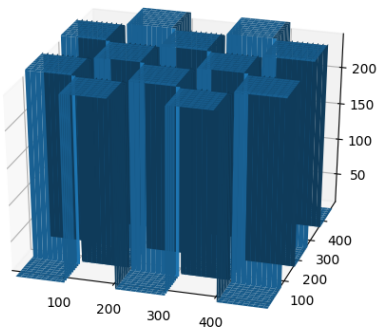


Figure 2.2: Surface Plot of Figure 2.2a

2.2 Linear Filtering

Filtering an image is a way to augment or extract information from it. Linear filters are a subset of operations that operate on a neighbourhood of pixels in an image. A filter's size determines the size of the neighbourhood to take as input for the operation.

Linear operations are characterized by two properties,

1. They are shift invariant, meaning it doesn't matter where you apply the filter to an image, its effect is consistent.
2. They preserve vector space operations, meaning it doesn't matter if they're applied before or after another linear operation, the result is the same.

All linear filters operate on the same prinicpal, that is, they output a weighted average of their input. It is the weightings of a filter, known as *filter coefficients*, that determine the effect of the filter. These weights are stored in a matrix called a *mask* or *kernel* that's then passed over the image. The mask is nearly always square so as to have a center which sits atop a reference pixel. The position of the reference pixel in the filtered image (output) will hold the result of the filter application that had that position at its center.

[SHOW FILTER OVER IMAGE, NEIGHBORHOOD, CENTER PIXEL, MASK]

For example, a box filter simply outputs the average of its inputs by having all filter coefficients equal and then scaling the result by the sum of all weights. This is useful for smoothing images.

[SHOW BOX FILTER MASK MATRIX]

Here, the weights are evenly distributed and equal to the chosen weight divided by the sum of all weights. By passing this filter over an entire image the sharpness of the image is reduced as all the values are averages of their neighbourhood, giving a smoothing or

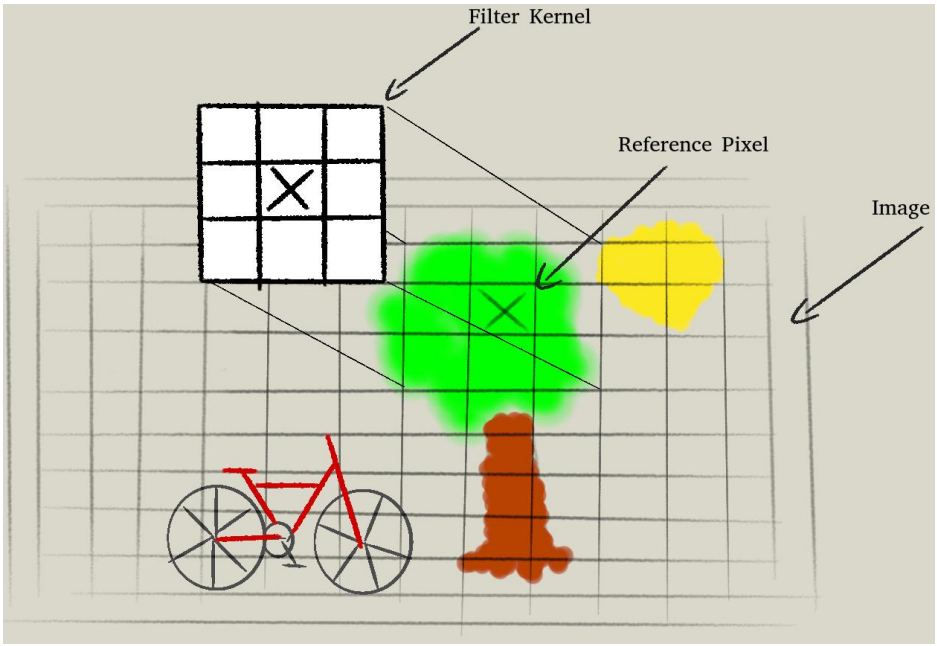


Figure 2.3: Application of a Filter Kernel to an Image

$$Kernel = \frac{1}{\sum_{i=0}^{M}\sum_{j=0}^{N} c_{i,j}} \begin{bmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,n} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m,0} & c_{m,1} & \cdots & c_{m,n} \end{bmatrix} \tag{2.2}$$

Figure 2.4: General Form of a Linear Filter

blurring effect.

[SHOW OUTCOME OF APPLYING BOX FILTER TO IMAGE]

The application of a linear filter $h(u, v)$ to an entire image $f(i, j)$, where $h(0, 0)$ is the center of the filter mask and $f(0, 0)$ is the target pixel beneath the masks center, may be described as follows

$$g(i, j) = \sum_{u=-k}^{k}\sum_{v=-l}^{l} f(i + u, j + v)h(u, v) \tag{2.3}$$

where the dimensions of the filter kernel are $2k + 1 \times 2l + 1$. Filter kernels are nearly

(a) Photo by Charles Deluvio on Unsplash

(b) Photo by Charles Deluvio on Unsplash

Figure 2.5: Application of Box Filter to RGB Dog

always odd dimensioned so as to have a center. This is because an asymmetric kernel (even numbered dimensions) will result in information being distorted because the output from the filter will be weighted unevenly by being placed in a reference pixel more one to one side than another.

[SHOW RESULT OF USING ASYMMETRICAL KERNEL]

2.2.1 Correlation

Linear filtering may be notated more concisely by the *correlation* operator.

$$g = f \otimes h$$

This operation measures the similarity between two signals (functions). We can think of both digital images and linear filters as signals. Performing correlation between them will yield an image where the highest values (pixel intensities) correspond to where the image and filter were most similar[1].

[SHOW EXAMPLE WITH SHARPENING FILTER]

This operation is *shift invariant*, which means that it does the same thing no matter where in an image it is applied. Therefore, correlation operations obey both the superposition principle

$$a(f_1 + f_2) = af_1 + af_2$$

and the shift invariance principle

$$g(i,j) = f(i+k, j+l) \Leftrightarrow \ (h \circ g)(i,j) = (h \circ f)(i+k, j+l)$$

This operation has the side effect of flipping both horizontally and vertically the location of output points relative to location the center point (*reference point*) in the original image which may be undesirable.

[SHOW EXAMPLE OF HOW CORRELATION FLIPS INPUT FOR OUTPUT]

2.2.2 Convolution

Convolution is also a linear operation that is shift invariant. It is very similar to correlation except that the output signal from a convolution operation is not flipped relative to the input signal. It is described mathematically by the expression,

$$g(i,j) = \sum_{u=-k}^{k} \sum_{v=-l}^{k} f(u,v)h(i-u, j-v)$$

We see that this is similar to equation 2.3 but that the filter $h(i,j)$ is the entity being shifted, furthermore it is flipped (note the negative signs), this results in the output's orientation being correct. Convolution may be notated with the following operator

$$g = f * h$$

[SHOW EXAMPLE OF CONVOLUTION NOT FLIPPING OUTPUT]

2.3 Clustering

This is a method of separating an image in disjoint sets (clusters). This is useful as we can semantically label different sections of an image according to their similarity.

[SHOW EXAMPLE OF IMAGE SEGMENTED BY CLUSTERING]

An entitys (for example a pixel) membership to a cluster is justified by it meeting some criteria that along with the rest of the clusters members. A simple example of this is a luster defined by its location in an image, all pixels in that region would belong to that cluster. This is a very simple criteria and often more sophisticated prerequisites must be met in order to classify a pixel's cluster. Entities like pixels or a neighbourhood of pixels'

characteristics are represented in something called a feature vector.

2.3.1 K-Means

There are many methods by which to segment an image with clustering but K-Means is very popular for its simplicity and speed. K-means produces K clusters implemented as follows

1. Place K points on the image randomly or with some educated guess. These points are the initial centroids for the K clusters.
2. Assign all data points to their nearest centroid.
3. Update the centroids' positions as the mean of all data points in that class.
4. Repeat steps 2 and 3 until cluster centroids no longer move (converge).

K-means is guided by trying to limit the variance within each class. Variance is described by the expression

$$V = \sum_{i=1}^{k} \sum_{x_j \in c_i} (x_j - \mu_i)^2 \tag{2.4}$$

Where x_j are the data vectors, c_i are the classes that the data point belong to and μ_i are the class centroids.

Because the K-Means algorithm is fast and the success of the segmentation depends on the initial placement of centroids, often the algorithm is performed many times with different initial conditions and the instance that yields the best results is used.

[SHOW EXAMPLE OF K-MEANS CLUSTERING BEING USED]

2.4 Gaussian Mixture Model

2.5 Morphology

2.6 Contours

2.7 Centroid Tracking

Chapter 3

Design

Containing a comprehensive description of the design chosen, how it addresses the problem, and why it is designed the way it is.

Chapter 4

Implementation

Containing a comprehensive description of the implementation of your software, including the language(s) and platform chosen, problems encountered, any changes made to the design as a result of the implementation, etc.

Chapter 5

Summary and Reflections

Including a discussion of results in a wider context (considering other work).

5.1 Project management

Covering the tasks as a part of your work plan and progress as well as how time and resources are managed.

5.2 Contributions and reflections

Providing the details of your achievements and contributions including innovation, creativity and novelty (if there is any) as well as a personal reflection on the plan and your experience of the project (a critical appraisal of how the project went).

Bibliography

[1] Sahoo, S. Deciding optimal kernel size for cnn.

Appendix A

User Manuals

Appendix B

User Evaluation Questionnaire