Traffic Surveillance

Vehicle Tracking

Kumar Abhinav
Indian Institute of Technology Kharagpur

Summer Research Project

Under the guidance of
Prof. Lam Siew Kei
PhD. Kratika Garg

School of Computer Science and Engineering,
Nanyang Techological University, Singapore

May - July 2016

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my guide **Kratika Garg** for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. The help and guidance given by her time to time shall carry me a long way in the field of Computer Vision.

I take this opportunity to express a deep sense of gratitude to **Prof. Lam Siew Kei** for offering me the opportunity to work for summer project at School of Computer Science and Engineering, NTU and for his cordial support and guidance which helped me in completing this internship.

I am obliged to staff members of Hardware and Embedded Systems Lab, for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Kumar Abhinav
IIT Kharagpur

Contents

1 Introduction

A vision-based detecting and tracking vehicle in video streams is an important research in computer vision, and it plays an important role in ITS. The aim of motion detection is to get the changed region from the background image in video sequences, and it is very important for targets classification and tracking motion objects.

The vehicle tracks, or trajectories, are measured over a length of roadway, rather than at a single point,thus it is possible to measure true density instead of simply recording detector occupancy.The additional information from the vehicle trajectories could lead to improved incident detection, both by detecting stopped vehicles within the camera's field of view and by identifying lane change maneuvers or acceleration/deceleration patterns that are indicative of incidents beyond the camera's field of view. The trajectory data could be used to automate previously labor intensive trajectory studies, such as examining vehicle maneuvers in weaving sections or bottlenecks.

The representative works in vision based vehicle tracking are -

- Background subtraction and Mean shift algorithm
- Kalman filter based tracking
- SVM based particle filtering
- Feature based Optical Flow
- Spatio-temporal MRF

The various methods proposed above have their own advantages and some even track vehicles accurately. However existing techniques based on vehicle counting and tracking suffer from low accuracy due to sensitivity to illumination changes, occlusions, congestions etc. In addition, existing holistic-based methods cannot be implemented in real-time due to high computational complexity. The current work on vehicle tracking [1] is a block based holistic approach and employs variance as a means for detecting the occupancy of vehicles on pre-defined blocks . The tracking is done based on the occupancy data of the block .

2 Literature Survey

The various methods proposed above have their own advantages and some even classify vehicles accurately. However most of the techniques use Background subtraction or involve complex hardware. The objective of the study was to develop a classification algorithm as an extension to the existing research on Vehicle Tracking [1]. The feature extraction and classification process needed to be dynamic and computationally efficient for a traffic video as input.

3 Framework

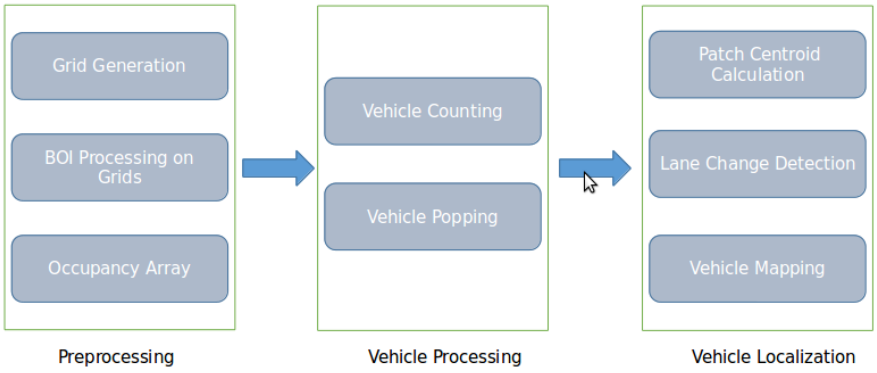


Figure 1: Tracking Framework

The proposed system mainly consists of three steps including Preprocessing, Vehicle processing and Vechile Localisation . The Preprocessing step is Vehicle region extraction based on method [1] employing variance as a means for detecting the occupancy of vehicles on pre-defined region of interest and has comparable accuracy with existing high complexity holistic methods.

The Vehicle Classification approach proposed relies on the centroid-based tracking and extracts the features based on the Region of Interest around each vehicle. Various approaches were implemented as an extension to the tracking method as follows -

- Contour Detection for each vehicle
- Bounding Box Detection for each vehicle
- Separating the Saturation layer and extracting vehicle features
- Convex Hull / Bounding Ellipse Detection

All the above mentioned approaches were based on the available packages in opencv - findContours, BoundingBox etc. None of the them was a promising approach since each suffered poorly in case of occlusion with

road markers, sidewalk or other vehicles. The most problematic scenario was to get correct vehicle dimensions for lanes near to sidewalk and vehicles near road-markers since that leaded to a very high degree of occlusion and the contour detection or convex hull detection resulted in unnecessarily large contours. The approach finally adopted is based on the idea of convolution and generates a box bounding box around a vehicle step-wise using local maxima matching at every step.

4 Vehicle Classification

The Vehicle Classification approach adopted is based on the idea of convolution considering local optimum. For each vehicle blob, an approximately fitting rectangular bounding box is generated stepwise. The overview of the entire algorithm can be obtained in the following sections.

4.1 Vehicle Features

To differentiate vehicles into three categories as mentioned in Table-1 different features can be considered for classification. The study involves considering the most basic set of features -

- Vehicle Width
- Vehicle Height

As an extension to the above features, many other features can be considered like **fitting an ellipse**, **image moments**, **convex-hull perimeter**, **solidity**, **compactness** etc however proper extraction techniques need to be devised for each feature. In this study, we have proposed a **unique approach for extracting** Vehicle Width and Vehicle Height (number of pixels)

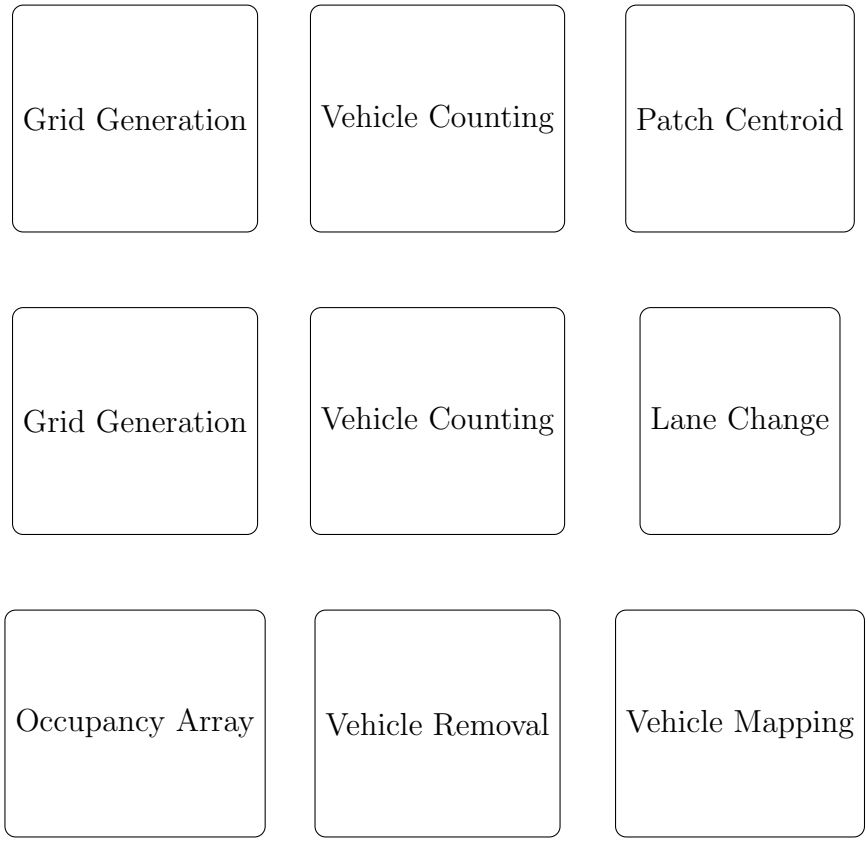
4.2 Feature Extraction

The overall aglorithm for feature extraction can be summarised as -

Step 1: The first step in our algorithm is to generate a custom **Region of Interest** around each vehicle present in every lane. This is generated based on the centroid of the vehicle obtained from tracking and mapping the centroid to the number of white patches derived from the Tracking Algorithm.

Step 2: On each of the custom generated ROI, we apply **Canny Edge Detection** algorithm to get the edges for each of the vehicles. The Canny Operator threshold set appropriately to get the best possible outcome.

Step 3: The detected edges are dilated in order to generate a **vehicle blob**. Erosion follows dilation in order to reduce the occlusion between two vehicles and between vehicle and the surrounding noise



Once an approximate white vehicle blob is obtained for each vehicle in the corresponding Vehicle ROI, we use a **convolution based local optimum** concept to generate a bounding rectangle for each vehicle.

Step 4: A square box is developed from the centroid of each vehicle. At each stage the similarity between the white box and the vehicle blob ROI is measured. The dimensions of the square box are increased until a certain level of similarity.

Step 5: In this step the square box is reduced from both ends to get the most optimum rectangular fit for each vehicle. This is based on the idea of local optima obtained in the similarity matching. The dimensions are reduced till a sudden spike is reached which denotes the matching-point.

Once the rectangle is obtained, the width and height of the vehicle are mapped against the corresponding Vehicle ID.

4.3 Unsupervised Learning

The vehicle features extracted from the traffic video are exported to a csv file for further analysis. Since none of the vehicle labels are known, unsupervised learning approach or **Clustering** has been used for vehicle classification.

The different Clustering algorithms that can be applied :

- k-Means Clustering
- Partioning Around Mediods
- Fuzzy c-means Clustering
- Hierarchical Clustering

The different Clustering algorithms that can be applied :

3.4 Implementation

AKS Primality test determines rigorously in polynomial time whether a number is prime or not. Since the algorithm has been presented many efforts are made to implement the algorithm. The major challenge is to implement the congruence, which is also most time consuming step in the algorithm. Most of the efforts are focussed on speeding up this congruence check

The basic algorithms used in the implementation are as follows

Algorithm Power check (n)

Algorithm Find Root (n,b) $n^{\frac{1}{b}}$ could be calculated by **integer Newton method**

Algorithm Largest Factor (p)

As mentioned earlier much efficient algorithms can be implemented but this does not have much effect on the time taken by the algorithm as most of the time is consumed by the congruence For loop.

Implementation of the Congruence

The evaluation of $(x-a)^n$ using repeated squaring could be represented as

$$y^n = (((((y_l^b)^2 y_{l-1}^b)^2 y_{l-2}^b)....)^2 y_1^b)^2 y_0^b$$

Where $y = x - a$, $n = \sum_{i=0}^{l} b_i 2^i$ and $l = \lfloor logn \rfloor + 1$

Let P(x) be a polynomial whose nth power is to be computed and PXP(x) be the polynomial which contains $P(x)^2$ .The following algorithm computes F(x)=$P(x)^n$.

Algorithm Square n multiply

We can observe that there is $\lfloor logn \rfloor$ squaring and at most $\lfloor logn \rfloor$ multiplication required by the above algorithm

Pseudocode for the implementation of congruence through basic polynomial multiplication and squaring:

Computation of the RHS of the congruence:

$(x^n - a)$ can be represented as
$(x^r - 1)(x^{n-r} + x^{n-2r} + ... + x^{n-kr}) + x^{n-kr} - a$

So the RHS = $x^{n \bmod r} - a \ (mod \ x^r - 1, \ n)$ as n-kr is n mod r.

Computation of the LHS of the congruence

$n = \sum_{i=0}^{l} b_i 2^i$ and $l = \lfloor logn \rfloor + 1$

$P[0] = -a; P[1] = 1; //P(x) = xa$

$F[0] = 1; F[i] = 0$ for all i;

$PXP[j] = 0$ for all j;

$degF = degPXP = 0;$

// degF and degPXP are the degrees of the F(x) and PXP(x) respectively

If one proceeds by the pseudocode above using grammar school multiplication and mod methods then the complexity of one iteration becomes $O(r^2 logn(logn)^2)$ and the total complexity of the algorithm becomes $O(logn)^{19}$.

But if FFT is used for both polynomial and integer multiplication then the complexity is reduced to $O(logn)^{12}$, this is due to the fact that $O(rlogr)$ and $O(logn)$ steps are used by FFT methods in polynomial and integer multiplication respectively.

Polynomial multiplication using Binary segmentation

A very interesting and efficient method of polynomial multiplication if one has large integer multiplication in hand is Binary segmentation (see [5]). Here the polynomial multiplication is achieved by multiplying two integers in which the coefficients are imbedded in binary form between pads of zero.

$$P(x) = \sum_{j=0}^{D-1} p_j x^j \quad Q(x) = \sum_{k=0}^{E-1} q_k x^k$$

$$P = 00....00p_{D-1}00...p_{D-2}...0p_0$$

$$Q = 00....00q_{E-1}00...q_{E-2}...0q_0$$

In the algorithm multiplication of huge integers is required, so FFT multiplication could be used effectively. One of the efficient FFT technique is given by Schnage in[7] and its implementation is given in detail in [8]. An interesting thing here to notice that squaring of a polynomial is faster than multiplication with another polynomial of the same degree, as it requires squaring of an integer. Some FFT methods perform squaring of integer almost 3/2 times faster than multiplication of integers of same bit length.

Enhanced powering ladder

An enhanced powering ladder can be used instead of ordinary ladder to speed up the performance. In this method the exponent is divided into words of k bits (k ¿ 1). This method is also known as windowing.

$$y^n = (((((y_l^b)^{2^k} y_{l-1}^b)^{2^k} y_{l-2}^b)....)^{2^k} y_1^b)^{2^k} y_0^b$$

Where $y = x - a$, $n = \sum_{i=0}^{l} b_i 2^i$ and $l = \lfloor logn \rfloor + 1$

So we can see therell be $k(l-1)$ squaring and $(l-1)$ multiplication. So if we increase base size k the computation will be driven towards squaring hence the time consumed will be reduced.

Though high values of k poses a threat to available memory, as all the coefficients of the polynomials from $(x-a)^2$ to $(x-a)^{2^k-1}$ have to be stored. Though this requirement could be still reduced if we just store the odd values and generate the even powers by their squaring.

5 Modifications on Original AKS Algorithm

Chris Caldwell[19] We will have to wait for efficient implementations of this algorithm (and hopefully clever restatements of the painful for-loop) to see how it compares to the others for integers of a few thousand digits. Until then, at least we have learned that there is a polynomial-time algorithm for all integers that both is deterministic and relies on no unproved conjectures!

There are two major approaches which can be used to reduce the complexity time:

- Restatement of the **for loop** to reduce the number of iterations.
- Finding more efficient ways to calculate the **useful prime r**.

The AKS-paper states the following lemma:

Lemma Let P (n) denote the greatest prime divisor of n. There exist constants $c > 0$ and n_0 such that for all $x \geq n_0$

$$| \{p \mid p\, is\, prime, p \leq x\, and\, P(p-1) > x^{\frac{2}{3}}\} \geq c\frac{x}{logx} \ .$$

An immediate corollary is there are many primes r such that $P(r-1) > r^{\frac{2}{3}}$.

This property is used in the proof of Lemma which determines there is a suitable $r = O(logn)^6$. As mentioned, the value of r plays a crucial role in determining the overall complexity of the algorithm. So if a stronger condition can be proved then a smaller value of r can be determined which will consequently reduce the running time of the algorithm.

4.1 Sophie German Conjecture

Sophie German Prime

A prime r is said to be a Sophie Germain prime if 2r + 1 is also prime.

The number of Sophie Germain primes less than n is asymptotic to $\frac{Dn}{(Logn)^2}$ where D is the twin constant (approximately 0.660161).

Lemma Assuming the Sophie Germain conjecture (Conjecture 1) there exists suitable r in the range $64(logn)^2$ to $a(logn)^2$ for all n ¿ n_0 , where n_0 and a are positive constants.
A proof for the Sophie Germain Conjecture therefore finds $\mathbf{r} = O(logn)^2$

4.2 Carmichael number Conjecture

Fermat's little theorem states that if p is a prime number, then for any integer a, the number $a^p - a$ is an integer multiple of p.

Carmichael numbers are composite numbers which have the same property of modular arithmetic congruence. In fact, Carmichael numbers are also called **Fermat pseudoprimes** or absolute Fermat pseudoprimes.

Carmichael numbers are important because they pass the Fermat primality test but are not actually prime. Since Carmichael numbers exist, this primality test cannot be relied upon to prove the primality of a number, although it can still be used to prove a number is composite.

Modify selection of r: It was observed that the congruence is satisfied for some composite numbers (Carmichael numbers). Based on the same, we need to modify the selection of r.

4.3 Multiplicative Order Conjecture

Given an integer a and a positive integer n with gcd(a,n) = 1, the **multiplicative order** $O_n(a)$ of a modulo n is the smallest positive integer k with

$$a^k \equiv 1 \bmod n$$

In other words, the multiplicative order of a modulo n is the order of a in the multiplicative group of the units in the ring of the integers modulo n.

$O_n(a)$ always divides $\phi(n)$. If $O_n(a)$ is actually equal to $\phi(n)$ and therefore as large as possible, then a is called a **primitive root** modulo n.

The order $O_n(a)$ also divides $\lambda(n)$, a value of the **Carmichael function**, which is an even stronger statement than the divisibility of $\phi(n)$.

Modify selection of r: Instead of finding q as the largest prime factor of r in the original AKS Algorithm, we shall try relate r to the multiplicative order of r modulo n and look for modification that can construct a more useful r.

4.4 AKS Conjecture Algorithm

We shall consider the following steps for the conjecture algorithm

- Algorithm Power Check
- Useful Prime r (While Loop)
  - If $(gcd(n, r) \neq 1)$ Output COMPOSITE;
  - Finding **Multiplicative Order** $O_r(n)$ to get range of r [Observation Table and Sophie German Conjecture]
  - Discarding **Carmichael Numbers** - Find r, such that r is not divide $n^2 - 1$ [Observation Table]
- The value of r from the above algorithm could be used in the congruence $(x-1)^n \not\equiv x^n - 1 \bmod (x^r - 1, n)$

6 Testing and Results

5.1 Sophie German Conjecture

We will investigate the Sophie German Conjecture which stated that there exists suitable r in the range $64(logn)^2$ to $a(logn)^2$ for all $n > n_0$, where n_0 and a are positive constants. So there should be an constant $C \geq 64$ such that r = $C(logn)^2$. A collection of primes from 1 bit to 18 bits was tested and the results are in the following table. N denotes the input number and C denotes the constant.

Table 1: Sophie German Conjecture

N	C
7	0.888
709	7.906
7103	43.393
11681	63.981
11689	64.015
71011	65.560
710009	64.060
70999997	64.106
70000000033	64.108
700000000009	64.018
7000000000009	64.205
70000000000009	64.041
10000000000000061	64.141
100000000000000003	64.011

- The value of C has very little variance once it reaches 64.
- The first prime which makes C bigger than 64 is 11689, and the next prime is 11699 which is last prime that AKS algorithm cannot find a useful prime r.
- **Once C reaches 64**, the AKS algorithm begins to find a useful prime r and **reduce the complexity** of the final congruence.

5.2 Carmichael Number Testing

It was observed that the congruence is satisfied for some composite numbers - **Carmichael Numbers**. The following table provides few composite number and r, which satisfies the congruence.

Table 2: Carmichael Numbers.

r	n Composite
2	561
3	1729
5	252601
7	1152271
11	1615681
13	2508013

In most of the cases to find whether a number n is prime or not, by AKS Algorithm we usually obtain larger values of r than those mentioned in the table.

- Choice of r should overcome the anomaly
- AKS Original Algorithm by its largest prime detection and selection algorithm for r takes care of the issue of Carmichael numbers in most cases
- However we are never certain about large prime numbers and their behaviour.
- **Conjecture** - It can be observed from the table that r**'s divide** n^2-1. So we can directly or indirectly build our selection of r around this.
- The **Carmichael function** of a positive integer n, denoted $\lambda(n)$, is defined as the smallest positive integer m such that $a^m \equiv 1 \pmod{n}$ for every integer a that is coprime to n.
- Carmichael function is also known as the **reduced Euler totient function**. Clearly Carmichael's theorem is related to Euler's theorem and Fermat's Little Theorem.

5.3 Useful Prime r Testing

A collection of primes from 1 bit to 18 bits were tested by the above two algorithms and the results are in the following table. N denotes the input number, T denotes time taken to find the useful prime r in milliseconds.

Table 3: Testing for Useful Prime r.

N	AKS Prime r	T	AKS Modified Prime r	T
5	5	0	2	0
37	37	0	9	0
977	37	0	36	0
2909	2909	0	53	0
11699	11699	0	91	0
11701	11699	0	93	0
86599	17327	1	126	0
123457	18443	1	128	0
2284423	28607	1	224	0
96447077	45083	1	348	0
484240567	53699	2	398	0
1307135101	58727	2	412	0
435465768733	96059	3	702	0
3435465768991	111263	6	795	0
65434218790277	134867	6	970	0
700000000000051	155699	9	1168	0
7000000000000037	177232	10	1344	0
10000000000000061	181199	11	1301	0
100000000000000003	204143	12	1536	1

Both tests found r very quickly, but our modification is more faster. On average the useful prime r found by our algorithm is about 130 times smaller than those found in AKS. The first useful r found by AKS original algorithm is when input number n is 11701. The time taken by both tests increases very slowly in terms of the bits of the input number n.

References

[1] Kratika Garg, Siew-Kei Lam and Thambipillai Srikanthan - Real-time Road Traffic Density Estimation using Block Variance

[2] Comparative Investigation of K-Means and K-Medoid Algorithm on Iris Data - International Journal of Engineering Research and Development - eISSN : 2278-067X, pISSN : 2278-800X - Volume 4, Issue 8

[3] Comparative Analysis between K-Means and K-Medoids for Statistical Clustering - 2015 Third International Conference on Artificial Intelligence, Modelling and Simulation

[4] R. Crandall and C. Pomerance. Prime Numbers: A computational Perspective. Springer Verlag, New York, 2001 [Binary Segmentation]

[5] Carmichael, R. D. (1912). American Mathematical Monthly 19 (2): 22-27.

[6] Shoup, J. Symbolic Comp. 20:363-397, 1995