Abstract

In this report, we present an approach for the automatic detection and tracking of vehicles in aerial video. The proposed system is comprised of three stages namely; Region of Interest (ROI) Selection, Classification and Tracking. In the first stage we expedite detection by identifying image regions that are most likely to contain vehicles. To achieve this we use a fast corner detector combined with an efficient feature density estimation technique. In the classification stage the system uses shape encoding local features and robust machine learning techniques to perform efficient vehicle detection. In the final stage we propose a tracking algorithm, which employs particle filtering to exploit vehicle dynamics, and improve tracker-detector associations. We evaluate the proposed system by performing a number of experiments and use the results to show that the system is capable of successfully detecting and tracking a variety of differing vehicle types under varying rotation, sheering and blurring conditions.

Table of Contents

List of Figures

List of Tables

1. Introduction

The automatic detection and tracking of people and vehicles in aerial imagery has seen significant attention from the research community in computer vision. In the past this research was primarily motivated by military applications, such as aerial surveillance and automatic target recognition. However, in recent years there has been considerable adoption of Unmanned Aerial Vehicle (UAV) systems and their associated technologies for civil and commercial uses. These applications include commercial aerial surveillance, automatic traffic monitoring and security related tasks, such has domestic policing, border control and search and rescue.

With the continual increase of the number of vehicles on the roads, there has been a growing requirement to efficiently monitor and manage the transportation networks. To achieve this, vast quantities of data need to be collected and analysed, often in real-time, and hence automatic traffic monitoring is becoming an increasingly important task. Automatic traffic monitoring has many applications; in traffic research it is being used extensively to automate previously labour intensive studies (Cheng, 2009). It also has an important role in the development of Intelligent Transportation Systems (ITS), where it is used to monitor traffic activities and detect congestions to assist in the automation of traffic flow. Traditionally traffic monitoring is accomplished using surveillance systems positioned in specific locations in the road network, however these systems have several drawbacks such as being fixed and sparsely located. For this reason there has been an increased investment into the use of UAV systems for aerial traffic surveillance. These systems have many advantages (Cheng, 2009) and are capable of capturing wide-area aerial imagery that can provide a more accurate indication of the current traffic situation. One of the main objectives in the development of these aerial surveillance systems is the ability to reliably detect and track vehicles.

1.1 Goals & Challenges

The goal of the work presented in this report is to develop a system that can automatically detect and track a variable number of vehicles in top-down aerial imagery captured above complex urban environments. In particular, we are interested in the detection and tracking of both static and moving vehicles. The aerial imagery is obtained from a low altitude UAV that has a fixed downwards facing camera capable of capturing low resolution video. The framework is developed without any prior knowledge of the sensory environment and is required to function independent of the UAV platform. As a result we have no access to any additional information provided by the other UAV sensors, such as GPS, accelerometer and gyroscopic information.

There are many factors that make the detection of vehicles in aerial imagery a very challenging task. Depending on the UAV's approach angle, flight altitude and position; the size, shape and orientation of vehicles can vary considerably. Another challenge is that, due to the low image resolution and the relatively small size of the vehicles in the image many of the details of the vehicles are not visible; this makes them difficult to distinguish from similarly shaped objects. In addition, due to its size, the UAV is very susceptible to vibrations caused by atmospheric turbulence; as a result the images captured often have substantial motion blur. The background of the scene can also have a considerable effect on the difficulty of identifying a vehicle; low contrast between vehicles and the background, the possibility of vehicles having similar colour and intensity as the background objects, and changing lighting conditions, requires the detection method to be invariant to colour and illumination. Background scenes can be cluttered and complicated, and in urban environments there may also be many objects that might resemble vehicles such as windows and building rooftops. A vehicles appearance might also be heavily influenced by shadows from neighbouring objects, and in addition, obstructed by environmental structures such as trees and lamp posts.

Tracking of vehicles involves the recognition of individual vehicles across consecutive video frames and has its own set of challenges. The difficulties associated with tracking arise due to a multitude of reasons; the first is related to the non-linear dynamics of both ground vehicles and UAVs. Vehicles can move at variable speeds and change direction over/between frames, to intensify this problem the vehicles may also be moving fast relative to the cameras frame rate. In addition, the UAV's approach angle and speed may also vary relative to the movement of the ground vehicles, therefore, a vehicles motions across frames is the combination of both the movement of the vehicle and the movement of the UAV. Furthermore, the tracking method employed must also be capable of tracking vehicles in potentially clustered and complicated scenes, where there might be multiple vehicles with similar appearance moving in close proximity to each other and a potential for complete vehicle occlusions.

1.2 Proposed Solution

In this work, we develop a vehicle tracking system that builds on the vehicle detection framework proposed by Gleason et al. (2011). The system uses shape encoding local features and machine learning techniques to learn a model of the appearance of vehicles. We expedite vehicle detection using a fast corner detection technique to perform region of interest selection, and employ particle filtering to model vehicle dynamics to improve tracking.

1.3 Report Outline

The remainder of the report is organised as follows:

In **Chapter 2**, an overview of the two general vehicle detection strategies is presented followed by a comprehensive summary of the related work on vehicle detection and tracking.

Chapter 3 gives an overview of the structure of the system and details the design of the approach developed. Furthermore it provides reasoning for the design choices and describes how the proposed system addressed the challenges identified above.

Chapter 4 summarises the implementation strategy used. It includes a brief description of how each subsystem works and the associated data structures.

Chapter 5 specifies the experiments carried out on the systems, it includes details of the obtained results and a discussion on the performance of the system.

Chapter 6 closes the report and provides a conclusion on the outcomes of the project followed by suggestions of future work to improve the system.

2. Literature Review

In recent years there has been an abundance of research into the detection of vehicles in aerial imagery. This however is not true for vehicle tracking in aerial imagery. Therefore, for this reason, we focus this chapter on providing a comprehensive summary of the current vehicle detection approaches in academic literature.

The approaches to vehicle detection can be arranged into two groups; explicit modelling approaches and implicit modelling approaches (Hinz, 2003). In explicit modelling approaches geometric features of a car are used to create a generic model of a vehicle constructed by representing the shape of the vehicle with a 2D/3D model e.g. a wire-frame representation. Detection is performed by extracting features such as edges and grouping them together to create structures that can be compared to the model. These methods have proven very robust (Zhao and Nevatia, 2001, Hinz, 2003) however because of the small size of vehicles in aerial imagery models must be simplistic and are therefore in danger of matching many regions in the image.

In recent years there has been increased research into implicit approaches. Implicit or appearance-based modelling approaches use machine learning to perform statistical classification. Vehicles are represented by local or textural features, from which distribution–based descriptors are used to create a model. Detection is achieved by computing feature vectors for regions of interest and classifying them against this model (Nguyen, 2007). The inherent weaknesses of this approach are that the models generated by the classifiers are dependent on the training data and often struggle to generalise well as it cannot be assured that the training data captures changes in illumination, pose and other possible influence on the vehicles appearance by neighbouring objects and the relative position of the UAV (Hinz, 2003).

In the next section, we provide a review of the most recently proposed implicit modelling approaches and where possible a summary of their performance.

2.1 Related Work

Brecken et al. (2009) proposed a two stage approach to the automatic detection of vehicles within aerial imagery. Their approach is based on using multiple cascaded Haar classifiers (Viola and Jones 2001) for vehicle classification and a secondary verification stage that attempts to eliminate non-vehicle candidates based on UAV altitude driven vehicle size constraints. The cascaded Haar classifier provides a reliable detector that is invariant to vehicle colour, type and configuration. To achieve vehicle orientation invariance they use four separate cascaded Haar classifiers trained on sample vehicle images categorized into four positional orientations. The four classifiers are then evaluated over a query image at multiple scales and positions using a sliding window approach and multiple-classifier detections and detection overlaps are resolved using a spatial merging technique. Gąszczak et al. (2011) later extended this work with the use of additional thermal imagery for thermal signature confirmation, which improved performance considerably.

Gleason et al. (2011) focused on automatic vehicle detection in rural environments. Their approach consists of a cascade detection algorithm which is comprised of two stages. In the first stage a Harris corner detector is used to identify image features of interest, the authors arguing that “vehicles in particular have a large number of edges and corners compared to natural objects”. Next an efficient sliding window approach is used to determine regions with a feature density higher than a predetermined threshold, overlapping regions are grouped together and are further refined based on the colour characteristics of background areas. The regions selected by this stage are then presented to image classification techniques in the second stage which determine the presence of a vehicle. Their research compared the performance of two image patch descriptors, a modified Histogram of Oriented Gradients (HOG) feature and Histogram of Gabor Coefficients features. They also investigated the performance of three statistical classification techniques, K-Nearest

Neighbours (k-NN), Random Forests (RF) and Support Vector Machines (SVM). From the first stage of their algorithm they obtained an average detection rate of 85% and found the top performing classifier to be Random Forests using the Histogram of Gabor Coefficients features, which was capable of classifying 98.9% of vehicles and 61.9% of background images correctly.

Sahli et al. (2010) proposed a local feature-based approach for automatic vehicle detection in low-resolution aerial imagery. Their approach was developed with the aim of being free from the constraints related to detection methods based on a vehicles visual appearance i.e. a vehicles rectangular shape and the presence of frontal and/or rear windows (Zhao and Nevatia, 2001). Their approach is based on the extraction of Scale-Invariant Feature Transform (SIFT) features from vehicle and background images. These features are used to train a Support Vector Machine (SVM) classifier to define a model that can be used to classify SIFT features extracted from the cars and background in a query image. The collection of SIFT features that are predicted to belong to a vehicle are then clustered in the 2D image space into subsets associated with individual cars. Their clustering method is based on a modified Affinity Propagation (AP) algorithm that is bounded by the spatial constraints related to the geometry of vehicles at the given resolution. They obtained a classification accuracy of 95.2% on aerial imagery of a parking lot containing 105 cars with no false-positive detections.

Shi et al. (2012) developed a context-driven framework that uses scene context to improve the detection of moving vehicles in urban environments. Their approach is comprised of three stages; motion detection, vehicle detection and an on-line method of road network estimation for detection filtering. To detect moving objects they use an image stabilisation technique, by which SURF (Speeded Up Robust Features) features are registered across successive frames and fitted to an affine model for image warping. With stabilised frames background subtraction is used to detect regions that do not move according to the estimated homography. A cascade of Support Vector Machines (SVM) classifiers that use shape and size, and Histogram and Oriented Gradient (HOG) features are then used to detect vehicles in the

candidate regions. In the final stage they use multi-object tracking to track the trajectories of the detected objects to estimate the road network and use this information to discard vehicle detections that are not on the roads. They evaluated their system on the CLIF dataset and obtained a positive and negative classification rate of 0.843 and 0.797 respectively for their cascade classifier.

Nguyen et al. (2006) proposed a boosting based framework for vehicle detection for aerial images. They use Haar features, Histogram of Oriented Gradient (HOG) features and Local Binary Patterns (LBP) with an on-line version of the Adaboost training algorithm to select the most informative features. In this approach each feature resembles a single weak classifier and boosting is used to select an informative subset from these features to create a strong classifier. To create a weak classifier from the features they model the probability distribution of each feature in vehicle and background images using a Bayesian learning algorithm for the Haar features and a nearest neighbour learning algorithm for the two histogram-based features. Detection is performed using a sliding window approach and thresholding the outputted confidence values. Overlapping detections are grouped together using mean shift clustering applied to the probability density distribution of the classifier outputs.

2.2 Conclusion

Many implicit modelling approaches have been proposed for the task of automatic vehicle detection in aerial images. The general approach is to decompose the problem into two stages; a detection stage followed by a filtering and verification stage. In most cases detection is achieved using a sliding window strategy, in which an exhaustive search is performed across position and scale. To accelerate detection some approaches make use of integral images and integral histograms for real-time computation of features. Features that encode shape information are the most common method for vehicle representation and appear to perform well. The classification approaches used vary, however most choose to create discriminative learning models as opposed to probabilistic models, with the most prominent classifier being the Support Vector Machine.

3. Design

In this chapter, the design of the proposed vehicle detection and tracking system will be described. An overview of the structure of the system will be provided, followed by a detailed explanation of the stages in the approach. This will include reasoning for the design choices and will discuss how the proposed system addresses the challenges identified in Chapter 1.

3.1 System Overview

The proposed vehicle detection and tracking system comprises three stages: region of interest selection, classification and tracking. Each stage is decomposed into a number of steps; the interaction between these steps is illustrated in **Figure 1**.

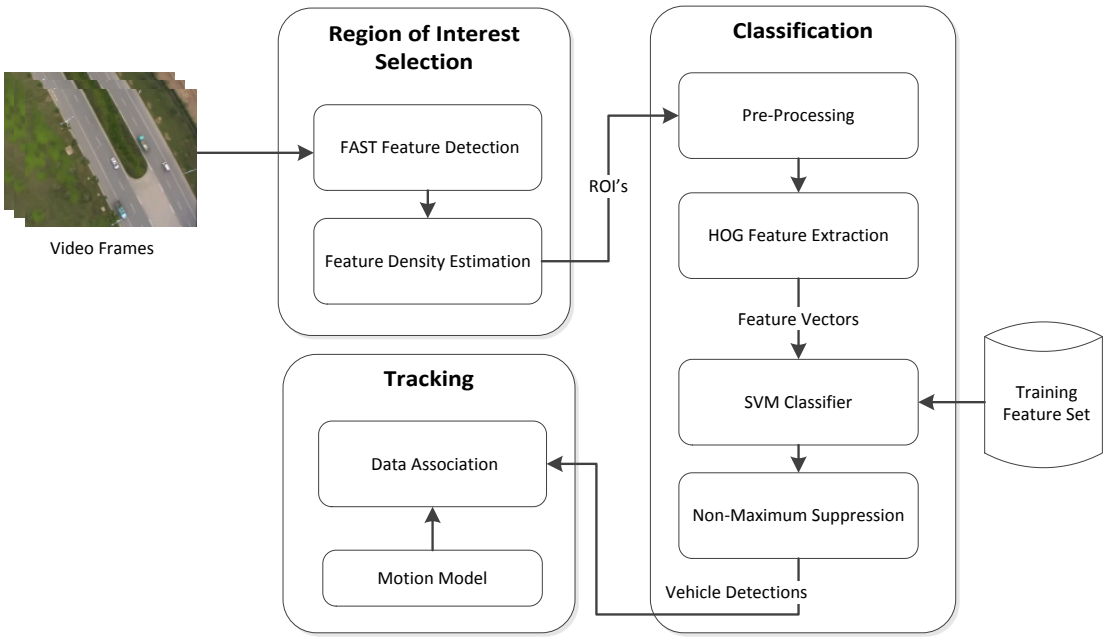

Figure 1: Overview of proposed Vehicle Detection and Tracking framework.

A brief summary of each stage in the system is provided below:

Region of Interest Selection: The vehicle detection and tracking system is initialised by the region of interest selection process. This process attempts to expedite detection by identifying regions in the image that are most likely to contain vehicles. The process uses a fast corner detector and an efficient feature density estimation technique to identify regions with a high concentration of features.

Classification: The classification stage categorises the regions of interest selected in the previous stage into vehicle and non-vehicle classes. Histogram of Oriented Gradients (HOG) features are extracted from the selected image patches and presented to a collection of four support vector machines for classification.

Tracking: The tracking process attempts to locate a vehicle (or multiple vehicles) in a sequence of frames. We employ a particle filtering technique to model vehicle dynamics and use a greedy matching algorithm to make vehicle associations between frames.

3.2 Region of Interest Selection

Performing an exhaustive search over all image positions and multiple scales is computationally expensive and often excessive as we usually only expect vehicles to occupy a small area of the image. Therefore, selecting regions of interest prior to classification is an important stage in the development of a real-time system.

To select regions of interest an approach similar to that proposed by Gleason et al. (2011) is used. The approach depends on the observation that man-made objects, specifically vehicles, have a large number of edges and corners compared to other natural topological features such as roads, trees and grasslands. With this observation in mind the algorithm uses an efficient sliding window technique to select regions in the image with a high density of corner features and consequently selects regions that are most likely to contain vehicles. To detect corners Gleason used the Harris corner detector, however in this report we propose the use of the Features from Accelerated Segment Test (FAST) corner detector as it has a similar corner response but it is many times faster (Rosten and Drummond, 2005). This provides improved real-time capabilities and comparable detection results (see Chapter 5).

3.2.1 Features from Accelerated Segment Test (FAST)

The Features from Accelerated Segment Test (FAST) detector was developed by Rosten and Drummond (2005, 2006) and is based in principle on the SUSAN corner detector. The FAST detector classifies a pixel *p* as a corner by performing simple brightness tests on a discretized circle of sixteen pixels around *p* (**Figure 2**). A corner is detected at *p* if there are twelve contiguous pixels in the circle with intensities that are all brighter or darker than the center pixel *p* by a threshold *t*. For this test condition to be satisfied three of the four pixels at the circle positions 1, 5, 9 and 13 must have intensity above or below the intensity of *p* by the threshold *t*. This allows the test condition to be optimised by only testing these four pixels first before examining all pixels in the circle.

In contrast to the Harris corner detector the FAST detector does not compute a corner response function. Therefore to perform non-maximal suppression the following score function must be evaluated for each candidate corner:

$$Score(p) = \max\left(\sum_{q \in S_{bright}} |I_q - I_p| - t, \sum_{q \in S_{dark}} |I_p - I_q| - t\right) \quad (1)$$

where S_{bright} is the subset of pixels in the circle that are brighter than *p* by the threshold *t* and S_{dark} is the subset of pixels that are darker that *p* by *t*.

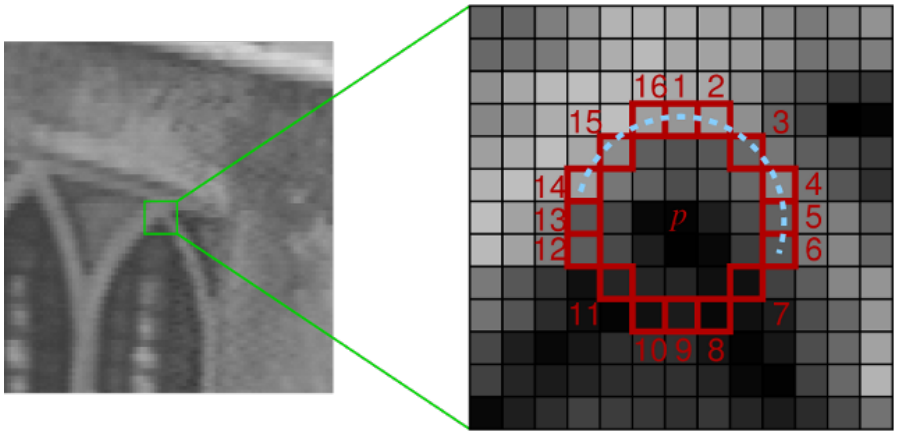

Figure 2: FAST Corner Detector - discretized circle of sixteen pixels around pixel *p* (Rosten and Drummond 2005).

3.2.2 Feature Density Estimation

Having detected the FAST corner features the next stage of the algorithm is to identify regions within the image that have a high concentration of these features. This is achieved by sliding a window over every location in the image and selecting the windows that have a feature density greater than some fixed threshold (shown in **Figure 3**). Given a window with the top left-hand corner *(x,y)*, width *w* and height *h* the feature density for this window is calculated by the score function used by Gleason el al. (2011):

$$Score(x, y, w, h) = \frac{S_{x,y,w,h}}{w \times h} \tag{2}$$

where $S_{x,y,w,h}$ is the number of features detected within the window. By sliding the window over the image at multiple scales and normalising $S_{x,y,w,h}$ by the area of the window, this allows us to build a scale invariant detector with the ability to detect vehicles at multiple scales in the image.

The efficient computation of the number of features in a window is determined using integral images (Viola and Jones, 2001). Given a set of FAST corners for a query image, a corner image $i(u, v)$ can be created where the value at any point is 1 or 0 indicating the presence or absence of a corner respectively. For this corner image, the corresponding integral image $I(i, j)$ is the image where each point (i, j) is computed as the sum of all the corners above and to the left of (i, j) inclusive:

$$I(i, j) = \sum_{u \leq i, v \leq j} i(u, v) \tag{3}$$

This image can be computed efficiently in a single pass over the corner image. Once computed this image allows for the evaluation of any window in constant time with only four lookups, where by the number of features in the window is computed as $S_{x,y,w,h} = I(x + w, y + h) + I(x, y) - I(x + w, y) - I(x, y + h)$.

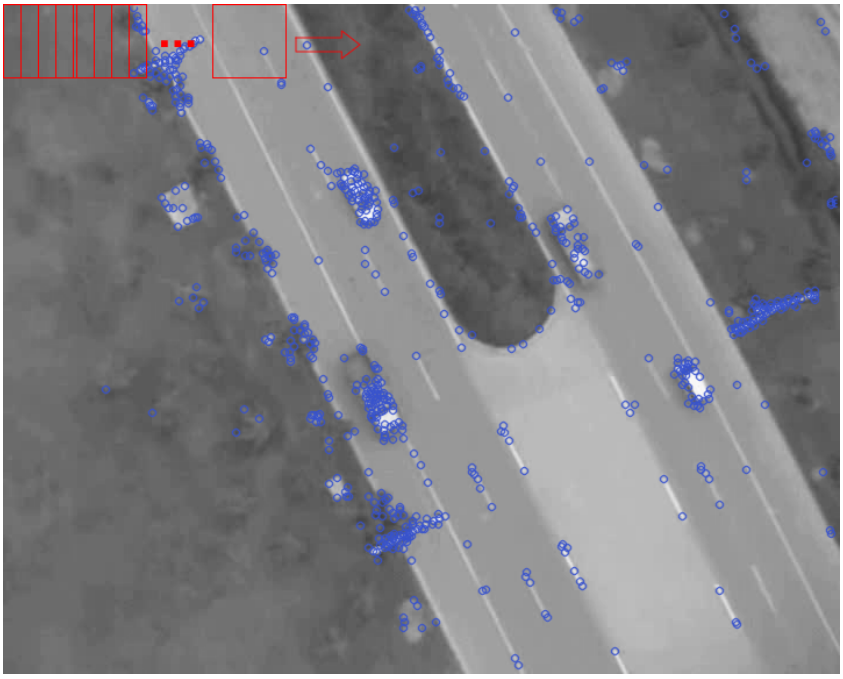

Figure 3: Feature Density Estimation – FAST corner features in blue.

3.3 Feature Extraction

Feature Extraction is a special form of dimensionality reduction; it is the process of transforming an image into a reduced representation that describes the most informative features in the image. In the proposed system feature extraction is used to transform the image patches selected by region of interest detection into a representation that can be presented to a classifier.

As discussed in Chapter 1, this representation should have several desirable properties. The representation must be invariant to changes in scale and rotation to enable vehicles to be recognised irrespective of their size and orientation in the image. In addition this representation must also be invariant to illumination changes, colour and motion blur as well as intra-class variations and partial occlusions. These properties are necessary to ensure that the classifier generalises well and is capable of identifying a diverse variety of vehicles under a wide range of condition. Furthermore the representation also needs to be sufficiently distinctive in order to maximise the classification accuracy and due the real-time requirement of the system it must also be computationally fast.

Humans are very good at recognising vehicles in aerial imagery, for this reason Zhao and Nevatia (2001) carried out a series of psychological tests to find out what features of vehicles humans used to allow us to make the decision about the presence of a vehicle. They found that the two most significant features where the rectangular shape and the presence and position of the frontal and rear windshields. Shape is recognised by the presence and position of edges, for a vehicle these edges consist of four primary edges that identify the outline of the vehicle and a set of secondary edges for the windshields. Therefore, to capture this information we require a feature descriptor that is capable of encoding this edge information.

The idea behind the Histogram of Oriented Gradients (HOG) descriptor is that the shape of objects can often be well described by the distribution of edge directions even without precise information of the edges themselves (Dalal and Triggs, 2005). This makes is a good choice for vehicle detection because, edges on vehicles can generally be grouped in two major edge directions, the direction of the sides and the direction of the back, front and windshield edges. Furthermore these edge orientations are largely perpendicular and therefore this gives a common distribution of edge directions among vehicles (Gleason et al, 2011). In addition the HOG descriptor is advantageous as it is relatively invariant to the geometric and photometric changes described above. A weakness of the HOG descriptor is that it is not rotationally invariant however this functionality is provided by the classification approach and is addressed in section 3.4. The remainder of this section describes the theoretical aspects and the problem specific formulation of the HOG feature vector.

3.3.1 Histogram of Oriented Gradients (HOG)

The Histogram and Oriented Gradients (HOG) feature was developed by Dalal and Triggs (2005) and was originally proposed for the task of human detection. The extraction of a HOG feature vector from a detection window is decomposed into five steps:

1. Normalize gamma and colour
2. Compute gradients
3. Weighted vote into spatial and orientation cells
4. Contrast normalise over overlapping spatial blocks
5. Collect HOG's over detection window

(Dalal and Triggs, 2005)

After colour and gamma normalisation, edges are detected by convolving the image patch with the simple 1D $[-1, 0, 1]$ mask both horizontally and vertically. In the second step the image patch is subdivided into rectangular regions called cells, within each cell we compute the gradient for each pixel; in colour images the gradient is computed separately for each channel and the largest gradient is chosen for the gradient for that pixel. In the next step each pixel within the cell then computes a weighted vote for the orientation of the cell, where the vote is weighted by the gradient magnitude (i.e. the L^2 norm). These votes are accumulated into orientation bins, whereby a vote is cast into the closest bin in the range 0 to 180 (for unsigned gradient angle), and stored in a histogram. In the penultimate step local contrast normalisation is used to suppress the effects of changes in illumination and contrast with the background on the gradient magnitude. This stage was found to be essential for good performance and is achieved by grouping cells into large blocks and normalising within these blocks ensuring that low contrast regions are stretched. In addition, to ensure consistency across the image patch but still keep local variations, overlapping blocks can be used (Lacey, 2009). Finally the normalised orientation histograms for each cell are collected together and result in a $b \times c_x \times c_y$ –dimensional feature vector where b is the number of orientation bins is, $c_x \times c_y$ is the number of image cells.

The HOG approach provides a reasonably flexible descriptor that can be tuned for different applications, naturally this leads to a range of choice to be made such as the cell size, block size and number of orientation bins, the choice of these parameters is discussed in Chapter 4.

3.4 Classification

In this stage, HOG feature vectors extracted from the regions of interests are passed to a binary classifier which determines the presence of a vehicle in the image patch. As mentioned in the previous section, the HOG features are not rotationally invariant, therefore to facilitate the detection of vehicles at all orientations this functionality must be provided at the classification stage. The conventional method to achieve this is to train one classifier on images of vehicles at a single "norm" orientation and then evaluate each region of interest at multiple orientations by rotating the detection window. However, the drawback of this method is that the HOG features have to be recomputed for each rotation which is computationally expensive. Instead, we use an approach similar to Breckon et al. (2009), in which four separate Support Vector Machines (SVM) are trained on sample vehicle images that are categorised into one of four angular offsets to the horizontal: 0°, 45°, 90° and 135°. These four SVM's are then combined to construct a single classifier that evaluates a rotationally invariant response for a single HOG feature vector. Support vector machines were chosen as the learning algorithm used in classification as they demonstrated a very high accuracy in previous vehicle detection research (see Chapter 2).

Figure 4: Subset of training data used to train the 45° offset SVM.

3.4.1 Support Vector Machine (SVM)

In this section I will give a brief introduction to support vector machines. A support vector machine is primarily a discriminative two-class classifier. The support vector machine creates an optimal hyper-plane that separates two classes with a maximum margin, that is, it maximises the distance between the decision boundary and the closest positive and negative training examples. As such, the support vector machine attempts to minimise the expected generalisation loss on the training data.

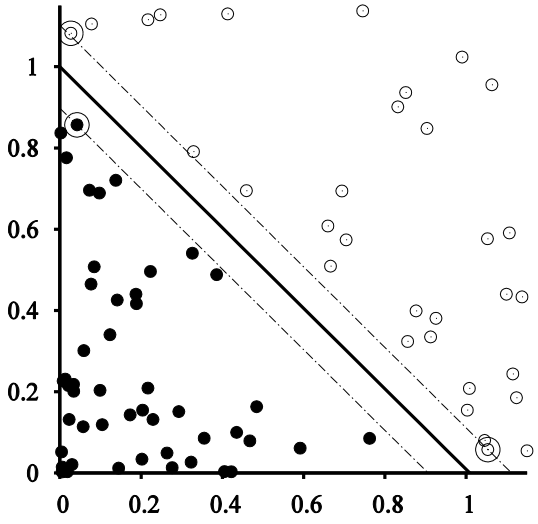

Figure 5: An example of a linearly separable problem in two dimensional space. The maximum margin separator (heavy line), is the midpoint of the margin (area between dashed lines). The support vectors are circled. (Russell and Norvig, 2010)

The support vector machine constructs a linear separating hyper-plane; however, by using the so called *kernel trick*, they have the ability to map feature vectors non-linearly into higher-dimensionality feature spaces. This allows them to efficiently perform linear classification on data that was not linearly separable in the original feature space.

The training examples closest to the separating hyper-plane are known as the support vectors – because they "hold up" the optimal hyper-plane (Russell and Norwig, 2010). As these support vectors completely define the separating hyper-plane, they are the only points retained by the support vector machine. This

therefore allows support vector machines, which are a nonparametric method, to gain some of the advantages of parametric models.

Given a set of N training features that belong to two classes $\{(x_i, y_i), \dots, (x_N, y_N)\}$, where the feature vector $x_i \in R^m$ and the class label indicated by $y_i \in \{1, -1\}$, the support vector machine attempts to solve the following minimisation problem:

$$\begin{aligned} minimise \quad & \frac{1}{2}\boldsymbol{w} \cdot \boldsymbol{w} + C\sum_{i=1}^{N}\xi_i \\ subject\ to \quad & y_i(\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i) + b) \geq 1 - \xi_i \\ and \quad & \xi_i \geq 0. \end{aligned} \tag{4}$$

where ξ_i is a set of slack variables and C is the penalty parameter. The function φ maps the input vector $\boldsymbol{x_i}$ into the higher-dimensionality feature space. The SVM only deals with inner products of feature vectors and hence the function φ is never used explicitly but is instead used to compute a kernel function $K(x_i, x) = \varphi(x_i)^T\varphi(x)$. Many SVM kernel function exist however in testing our classifier we only considered the three most popular kernels; the linear kernel (**Equation 5)**, the Polynomial kernel (**Equation 6)**, and the Radial Basis Function (RBF) kernel (**Equation 7)**.

$$K(x_i, x) = (x_i^T x) \tag{5}$$

$$K(x_i, x) = (1 + (x_i^T x))^P \tag{6}$$

$$K(x_i, x) = \exp\left(-\frac{1}{2}|x_i - x|^2\right) \tag{7}$$

3.5 Tracking

Tracking is the process of locating a vehicle (or multiple vehicles) in a sequence of images. In the proposed system tracking is achieved by the continuous application of the vehicle detection algorithm developed in the previous sections and the association of the detector responses across consecutive frames. This is an instance of the general tracking strategy known as tracking by detection.

3.5.1 Tracking Algorithm

This section gives an overview of the algorithm used to track vehicles. To begin, we introduce the concept of a vehicle tracker; a vehicle tracker is used to encode the state of a single vehicle and track its location in the video sequence. Assuming that the systems has been tracking from some time, the first step of the algorithm is to associate vehicle detections with existing trackers – this process is described in detail in section 3.5.3. There is a one-to-one mapping between trackers and detections however this mapping is not surjective, in other words in any given frame there may be fewer or more detections than the last and hence both trackers and detections can go unallocated. Therefore, the next step in the algorithm is to initialise new trackers for those vehicle detections that were not allocated in the previous step. A new vehicle tracker is initialised for all unallocated detection in a frame; however, these trackers do not become 'active' until they have received some number of detections over a defined number of frames without being terminated. This ensures that trackers are not visually initialised for persistent false positives and in cases when multiple detections are returned for a single vehicle. The final stage of the algorithm is tracker termination; at the end of each cycle we automatically prune trackers that have not received a detection for a defined number of frames. This ensures that trackers do not survive for vehicles that have left the field of view. The above algorithm is summarised in **Figure 6**.

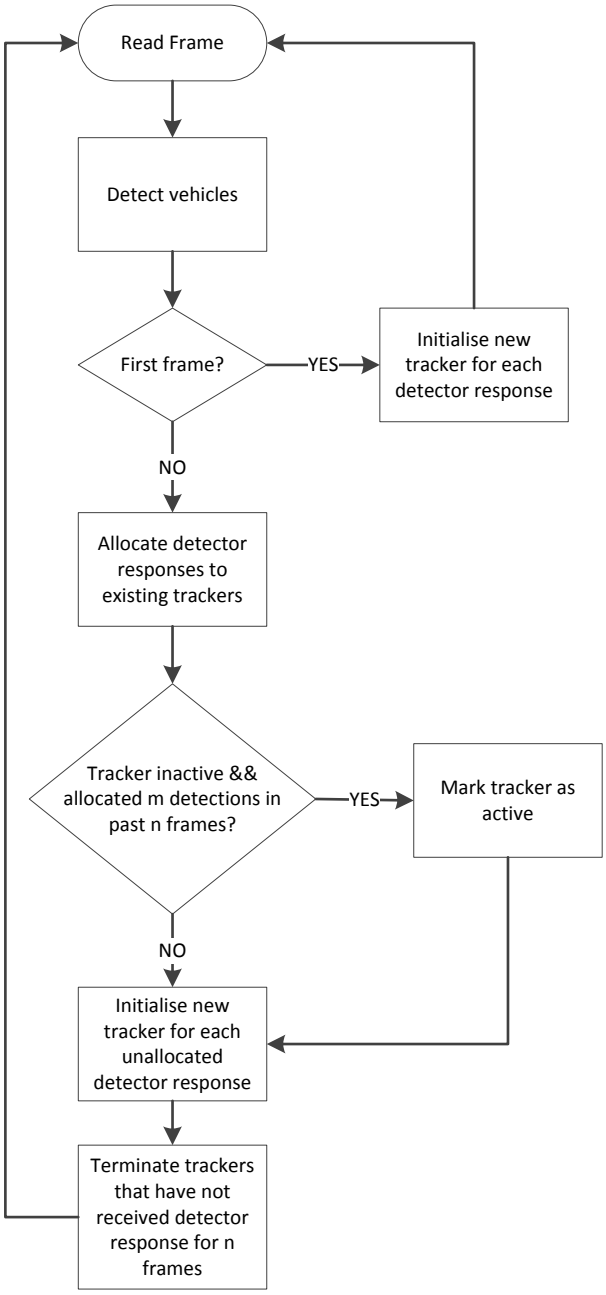

Figure 6: Tracking Algorithm

3.5.2 Particle Filtering

The association of detector responses to vehicles trackers is a very difficult task, in order to simplify this problem it is useful to generate some inference about the motion of a vehicle through the imagery. To achieve this task we use a Particle Filter.

Particle filtering is a general Monte Carlo (sampling) method for performing model estimation using simulation. Within this system particle filtering provides a probabilistic framework for encoding the state and dynamics (i.e. some model of how this state will change from frame to frame) of a vehicle. The goal of the Particle filter is to maintain an accurate representation of the posterior of a vehicles state by exploiting both the observations provided by the vehicle detector and a dynamical model of a vehicles motion. This representation is then used to evaluate the likelihood that a predicted detector response is associated with a given tracker. We choose to use a particle filter over other object tracking techniques such as the Kalman filter as particle filters are capable of modelling non-linear systems. As discussed in Chapter 1 the motion of a vehicle through the video sequence is highly non-linear and hence this makes linear dynamic models unsuitable for this application. Particle filters are a good choice as they are extremely versatile and robust. In addition, particle filters only rely on past observations and therefore are more appropriate for online applications such as ours.

In our approach we initialise a separate particle filter for each vehicle tracker. The state of a vehicle $x = (x, y, u, v)$ is composed of the 2d image location (x, y) and the vehicle velocity components (u, v). To model the vehicles motion we use a constant velocity motion model, shown in **Equation 8**.

$$(x, y)_t = (x, y)_{t-1} + (\Delta t)(u, v)_{t-1} + \varepsilon_{(x,y)}$$
$$(u, v)_t = (u, v)_{t-1} + \varepsilon_{(u,v)} \quad (8)$$

The process noise $\varepsilon_{(x,y)}, \varepsilon_{(u,v)}$ for each state variable is individually drawn from a zero-mean Gaussian distribution. For details about particle filtering we refer to Forsyth and Pounce (2012).

3.5.3 Data Association

Data association is the problem of deciding which vehicle detector response should be associated to which tracker. To make this decision, we use a greedy algorithm to solve a maximum weighted bipartite matching problem, where by the trackers form one side of the bipartite graph and the detections the other. The edges are weighted by a matching score that is calculated by evaluating a matching function for each tracker-detection pair.

The matching algorithm used in the proposed system is similar to that proposed by Breitenstein et al. (2009). The first step in the algorithm is to create a matching score matrix *S* where the rows correspond to existing trackers and the columns to the vehicle detector responses for the current frame. For each tracker-detection pair a matching score is calculated as described below. If the score is above some fixed threshold then the matching score is input into the corresponding cell in the *S*, otherwise a null value is input. By thresholding the matching score we ensure that we only associate detector responses that are good match for the trackers. Next the tracker-detection pair with the maximum score in *S* is iteratively selected and the row and column belonging the selected tracker and detection are removed. This process is repeated until only a null valued matrix is left, at which point the remaining rows and columns indicate the trackers and vehicle detections that have not been allocated respectively.

The matching score for a tracker-detection pair (tr, d) is calculated by evaluating the matching function $s(tr, d)$. This function considers both the vehicle appearance and the distance between the detection and the trackers expected position. The function calculates the colour histogram correlation between the detection d and the previous tracker detection tr_d which provides us with a correspondence measure between the two vehicles. In addition it also evaluates the distance between the detection d and each of the particles p of the tracker tr to determine the probability of the detection measurement given the predicted state of the tracker. The matching function is calculated as follows:

$$S(tr,d) = \mathrm{I}\left(H_{tr_d}, H_d\right) \cdot (\alpha \cdot \sum_{p \epsilon tr}^{N} p_{N(d-p)}) \tag{9}$$

Where $p_N(d-p) \sim N(d-p, \sigma^2)$ denotes the Normal distribution evaluated for the distance between d and p and $\mathrm{I}\left(H_{tr_d}, H_d\right)$ is the colour histogram correlation calculated as:

$$I(H_1, H_2) = \frac{\sum_i (H_1(i) - \overline{H_1})\,(H_2(i) - \overline{H_2})}{\sqrt{\sum_i (H_1(i) - \overline{H_1})^2 \sum_i (H_2(i) - \overline{H_2})^2}}$$

$$where\ \overline{H_k} = \frac{1}{N} \sum_j H_k\,(j)$$

$$And\ N\ is\ the\ total\ number\ of\ histogram\ bins. \tag{10}$$

4. Implementation

In this chapter, a detailed description of the implementation of the proposed systems is presented. A brief description of how each subsystem works is provided and the associated data structures are also defined.

4.1 System Specification

The vehicle detection and tracking system is implemented in C++ and makes extensive use of the OpenCV (Open Source Computer Vision Library) library. OpenCV is a library of programming functions that focuses primarily on real-time image processing and computer vision. A decision was made to use C++ with OpenCV as opposed to the popular Matlab programming language and supporting image processing toolbox, as C++ provides an object oriented programming environment that is implemented on a wide variety of hardware and operating system platforms. In addition OpenCV is widely supported by academia with many researches choosing to implement there algorithms using its functions. In particular the OpenCV library offers class wrappings for both the FAST feature detector and the HOG feature descriptor. The version of C++ used in this system is Visual C++ 10.0. The computer used for the implementation and testing of the system has the following specifications:

- 2.4GHz Intel Core i5 Processor
- 4.00GB RAM
- Windows 7 (64-bit) Operating System

4.2 Vehicle Detection

Vehicle detection comprises the first two stages of the proposed system in Chapter 3, specifically, the region of interest selection stage and the classification stage. Given a new frame, the first task of the systems is to perform vehicle detection. To perform this task we implement a *VehicleDetector* class that provides this functionality. The *VehicleDetector* class provides a *detect* method that is passed as an argument the new frame and returns a set of *VehicleDetections*. A *VehicleDetection* is a data structure that encapsulates the information about a single

detection in the frame. It contains a rectangular bonding box which indicates the location and scale of the vehicle in the frame, a confidence level – indicating the sureness of the presence of a vehicle, and a copy of the image patch in the detected region. The remainder of the section will explain in detail the implementation of the *VehicleDetector* class focusing mainly on the *detect* method.

4.2.1 Region of Interest Selection

The region of interest selection algorithm is described in the previous Chapter. The first step in the algorithm is to detect corner features using the Features from Accelerated Segment Test (FAST) corner detector. This is achieved using the OpenCV implementation of the FAST algorithm.

As discussed in Chapter 3 the FAST algorithm requires an intensity threshold to determine whether a pixel should be classified as a corner. During the development and testing of the system, it was found that this threshold needed to be low to facilitate reliable detection and hence is defaulted to 10. This allows for the detection of vehicles that have a similar intensity to the surrounding background and improves detection in situations when there is low contrast between the vehicles and the background. As a result of choosing a low threshold considerably more corner features are detected, however, due to the efficiency of the FAST algorithm this has little effect on the performance of the system.

The resulting corner features are then used create a corner image, which is then passed to the *integralImage()* function to create a corresponding integral image. Using the algorithm described section 3.2.2 a sliding window approach is taken to calculate a feature density score for all regions in the frame and return regions with a density greater than some fixed threshold *densityThreshold.* These regions are then processed and passed to the feature extractor.

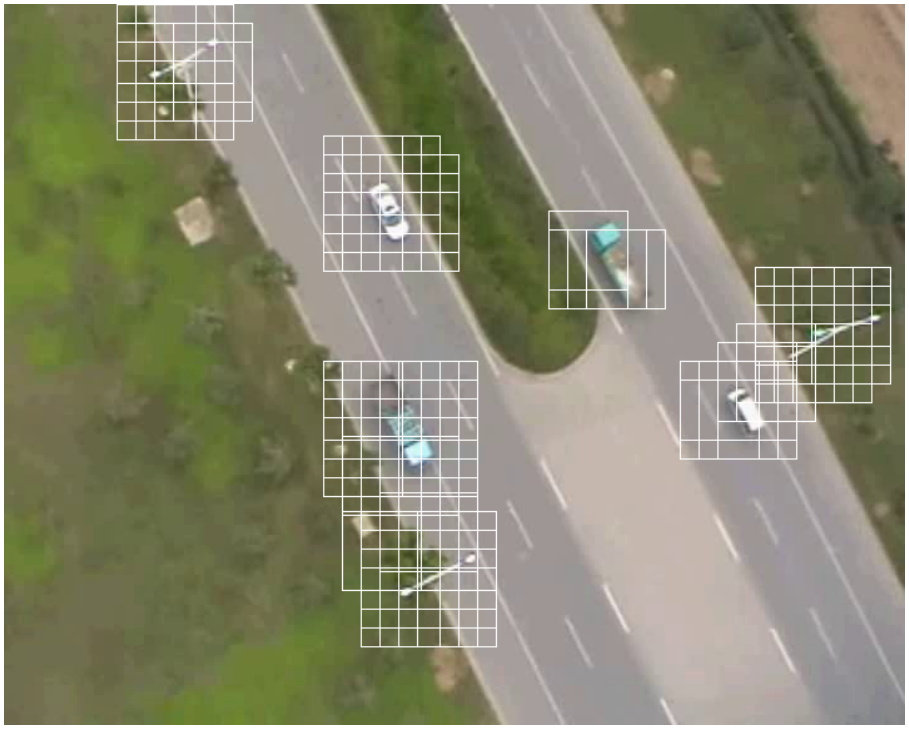

Figure 7: Region of Interest Selection – Regions with high feature density in white.

4.2.2 Feature Extraction

The Histogram of Oriented Gradients (HOG) features are extracted from the regions of interest using the OpenCV CPU HOG implementation. This implementation is based on the Dalal and Triggs (2005) approach and follows the methodology described in Chapter 3. The OpenCV *HOGDescriptor* class provides an interface to this implementation and has a *compute* method to calculate the HOG feature vector for a given image (patch). This method outputs the feature vector as a C++ float vector; therefore before these vectors can be classified they are converted to the format expected by the SVM package, i.e. a row vector representation of OpenCV matrix type.

As mentioned at the end of Section 3.3.1 the HOG approach provides a reasonably flexible descriptor that has a number of variable parameters such as the cell size, block size and number of orientation bins. We chose the number of cells experimentally and found that subdividing the image patch into 16 cell (4x4 cell) achieved the highest classification accuracy. Similarly after testing we found that the best number of orientation bins was 21, this provided a reasonably low

dimensionally feature vector that delivered good descriptive power and resulted in the best classification accuracy.

4.2.3 Classification

Having extracted the HOG features from the regions of interest these features are then presented to a classifier to be categorised into two classes, vehicle and background. The classifier is composed of four, separately trained, support vector machines as described in Chapter 3. In this project we use the n-class SVM implementation provided in OpenCV, which is based on the LibSVM library (Chang and Lin, 2011). During experiments we tested all three kernels presented in Chapter 3 and found that in this approach a Radial Basis Function (RBF) kernel provided the best results for vehicle classification.

To classify an image patch we iteratively test the feature vector against the four oriented SVM's, if any SVM returns a positive response the image patch is classified as a vehicle, otherwise it is consider to belong to the background. In the case when an image patch is classified as a vehicle, a new *VehicleDetection* is created and assigned the highest confidence level returned by the positive response SVMs. In the current system the orientation information provided from the positively responding SVM's is not used. However, an extension would be to use this information to constrain the direction of the velocity vector assigned to the particles used to estimate the vehicles state. Before the classifier can be used to categorise an image patch the four SVMs must be trained; the remained of this section described how this is completed.

4.2.3.1 Training the Classifier

Before we can begin training the classifier a large set of training images needed to be collected. Currently there is no existing datasets that contains top-down aerial imagery of vehicles therefore these images had to be manually extracted from the available UAV captured video. Vehicle images were cropped from the imagery and categorised into one of four possible orientations as described in Chapter 3.

Due to the limited amount of training and testing imagery, there was only around a fifty unique vehicles in the video, therefore to increase the number of vehicle training images, images of the same vehicle across different frames were extracted and categorised. In addition, vehicles in the available video were only moving in approximately four different directions, therefore to ensure that we capture a wide variety of vehicle orientations; we have added the same vehicles multiple times but rotated it by an arbitrary angle. As well as the vehicle images, a large amount of background images were also manually extracted. To ensure that the classifier generalises well we attempted to capture a large variety of background features.

The *train* method of the *VehicleDetector* class is used to train the classifier. This method accepts four *VehicleTrainData* structures that each stores the training images for one orientation category as well as their associated class labels. Subsequently this data is converted to the formatted expected by the OpenCV SVM implementation and used to train the individual SVM's.

As mentioned above, the RBF kernel function is selected for the SVM's. The RBF kernel has two parameters C and γ which need to be selected before classification. However, it is difficult to estimate which values for these parameters are best for a given problem (Hsu and Chang, 2003). As a result some form of parameter search must be done to find the best values. To achieve this the OpenCV SVM implementation selects the parameters using k-fold cross validation. In our implementation k is set to 10 and gives good results. Once selected, the parameters are then used to train the classifier on the whole training set.

4.2.4 Non-Maximum Suppression

As expected, due to the scanning resolution used when selecting regions of interest, the above procedure returns a large number of *VehicleDetections* around the vehicles in the frame. Therefore to attempt to reduce these numbers to a single detection per vehicle we reject overlapping detections using non-maximum suppression. This process is implemented in the *VehicleDetectors nsm* method. Starting with the *VehicleDetection* with the highest confidence level, this function

iteratively removes all other *VehicleDetections* that overlap with the detection significantly.

4.3 Vehicle Tracking

The main tracking algorithm described in section 3.5.1 is implemented in the *trackvehicles()* function. This function accepts two arguments; the first is the file location of the video file in which we desire to track vehicles and the second is the file location of the *VehicleDetector* model to be used by the detector to initialise the classifier.

Each step of the tracking process begins by reading a new frame from the video and detecting vehicles using the *VehicleDetector detect* method. For the first frame the association of detector response to vehicle trackers is easy, since no trackers have been initialised yet we simply assign each of the returned *VehicleDetections* to new vehicle tracker. This is achieved by initialising a new *VehicleTracker* object for each detection. The *VehicleTracker* class implements the vehicle tracker abstraction introduced in Chapter 3; the functionality provided by this class will be described in the subsequent section.

For subsequent frames the data association process is more difficult. We begin this process by firstly calling the *predict* method for each of the existing *VehicleTrackers.* This method updates the vehicle trackers internal particle filter to provide a new predicated state for their associated vehicle. With an estimated state for each of the trackers the next step in the algorithm is attempt to associate detector responses to the correct tracker. To do this we begin by computing a matching score for each of the tracker-detection pairs and constructing the corresponding matching score matrix described in section 3.5.3. Using this matrix we then assign the detector responses to trackers by iteratively allocating the highest scoring tracker-detection pair. In practice this is done using a two-stage approach, in the first stage we allocate the active trackers only and then in the following stage we allocate the remaining inactive trackers. This two-stage approach was found to be a very important step the development of a reliable tracking algorithm; if we consider the

situation when multiple detections are returned for a vehicle that is currently being tracked by a single tracker, one of these detection will be associated with the existing tracker and the other will result in a new in-active tracker being initialised. If in the next frame a single detector response is returned for the vehicle, then without the two-stage approach i.e. by assigning detections in a single pass based on the largest matching score, this detection may be better matched to the in-active tracker and hence be assigned according; which will eventually result in the vehicle changing tracker. The two-stage approach minimises the effects of these situations by prioritising the assignment of active-trackers and therefore ensures longer enduring trackers. The assignment of a detection to tracker is achieved using the *VehicleTracker update* method, this method ascribes the detector response to the tracker and then updates the trackers internal particle filter with a new measurement provided from the detections image location. In the final steps of the algorithm we initialise new trackers for each unallocated detector response, terminate the trackers that have not been allocated a detection in 3 frames and finally display the active trackers that were assigned a detection in a current frame - by drawing the detection window to the frame with an associated vehicle number (**Figure 8**). The above implementation of the tracking algorithm is illustrated in **Figure 9**.

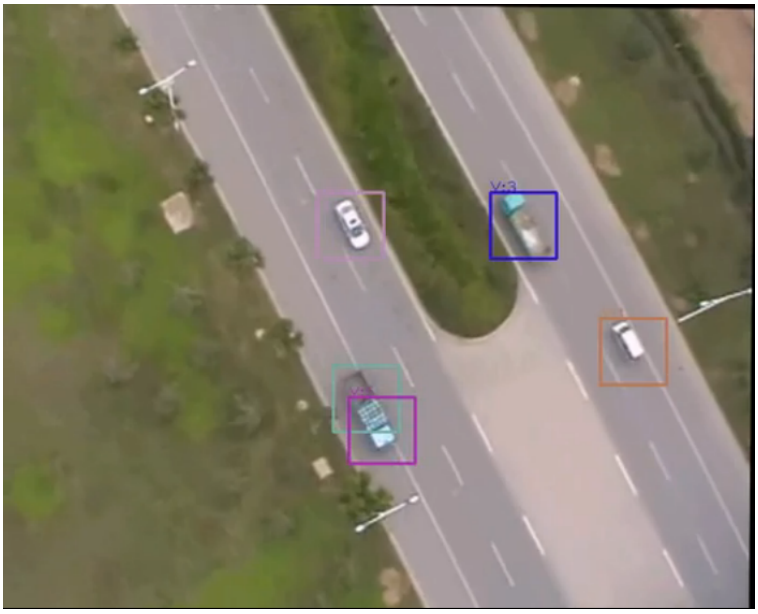

Figure 8: Output of tracking algorithm – Vehicle detection are show boxed with their associated vehicle tracker number.

4.3.1 Vehicle Tracker

The *VehicleTracker* class provides the functionality necessary to encode the state of a single vehicle and track its location in the video sequence. This functionality is essential in the operation of the above tracking algorithm and is provided by three methods that are described below:

- The score method implements the matching function described in section 3.5.3 and is used to calculate the matching score for the tracker and a given detector response. To calculate the colour histogram correlation component of the matching function we begin by converting the detector response and previous tracker detection image patch to the HSV colour space. In this colour space we then compute the normalised H-S 2-channel histogram for each image patch using the OpenCV *calcHist()* and *normalise()* functions and then finally compute the correlation between the two using the *compareHist()* function. To calculate the distance measure component we use the *ParticleFilter score* method. Finally these values are combined to produce a matching score.
- The *predict* method provides an interface to the *ParticleFilter predict* method and is used to predict the state of the vehicle in the current frame given past measurements.
- The *update* method is used to assign a detector response to the tracker and calls the *ParticleFilter update* method which performs the correction and resampling steps of the particle filter given the new measurement.

4.3.2 Particle Filter

As discussed in Chapter 3, it is useful to generate some inference about the motion of a vehicle to guide its tracker. To achieve this we maintain a *PartilceFilter* for each tracker. When a new tracker is initialised a new *ParticleFilter* is created. This *ParticleFilter* maintains a set of 100 particles; where each particle represents an estimated state of the vehicle. At first, these particles are initialised at random by independently drawing them from a Gaussian distribution around the center of the detection window. Once initialised the goal of the *ParticleFilter* is to maintain an accurate representation of the posterior of a vehicles state by exploiting both the

observations provided by the vehicle detector and a dynamical model of a vehicles motion. This functionality is provided by the *predict* and *update* methods.

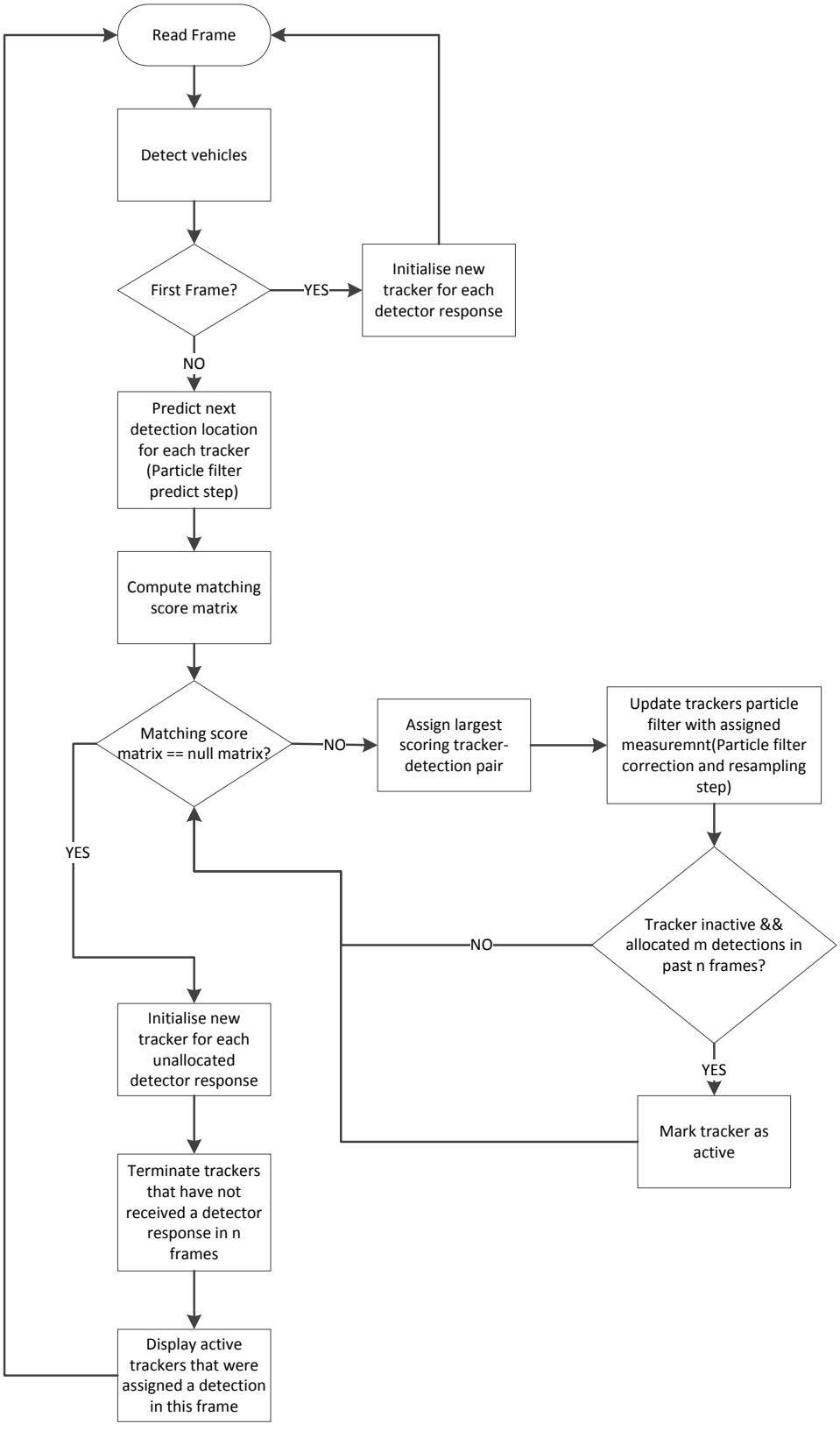

Figure 9: Tracking algorithm implementation

5. Results

In this chapter, an explanation of the experiments carried out on the system will be provided. A detailed analysis of the results obtained from these experiments will be presented and the overall performance of the system will be discussed.

5.1 Vehicle Dataset

To the authors knowledge there is no existing public dataset that comprises UAV captured aerial imagery which specifically contains road vehicles. Therefore, the experiments carried out on the proposed system use a video dataset that was provided to the author at the beginning of the project. The video dataset comprises five video clips that capture in total 59 seconds of aerial imagery. The video clips are extracted from a set of larger video sequences that were taken in multiple flights of a low-altitude UAV platform. These clips were captured with a resolution of 720 x 576 pixels, at a frame rate of 25fps and are all stored in AVI file format. The video clips contain aerial imagery of a motorway segment that is largely surrounded by a rural environment and together contain a total of 47 unique vehicles.

The provided dataset contains some of the challenges identified in Chapter 1, for example, the sequence contains a diverse set of vehicles that range from small passenger cars and vans to larger lorry and coach type vehicles. These vehicles are a variety of different colours and some have similar intensity to the surrounding road surface. In addition the video contains vehicles moving in close proximity to each other at different speeds and in varying direction. Furthermore the sequence also contains partial vehicle occlusions by lamp post structures. These above challenges consequently allow us to the test the proposed framework under some of the conditions that it would likely encounter in real world use.

5.2 Region of Interest Selection Results

To test the performance of the region of interest (ROI) selection stage we evaluated its detection rate (i.e. its ability to correctly identify regions containing vehicles) across the entire video dataset. To make comparison we performed the same experiment using both the FAST corner detector used in our system and the Harris corner detector used in the approach proposed by Gleason et al. (2011). To assess the detection rate of the ROI selection stage we computed the detection accuracy across all the video sequences. Assuming that for each frame t the number of positive detections is indicted by p_t and the number of ground-truth vehicles is indicted as $N_G^{(t)}$, we computed the detection accuracy as:

$$Detection\ Acc = \frac{\sum_{t=1}^{N_{frames}} p_t}{\sum_{t=1}^{N_{frames}} N_G^{(t)}} \tag{11}$$

Where we count a positive detection when there exists an identified ROI that overlaps the ground-truth vehicle detection by more than 80%. This metric gives us a percentage measure of the accuracy of the ROI selection stage. When evaluating this metric using the FAST corner detector and the Harris corner detector we found that they obtained a detection accuracy of 0.983 and *0.99* respectively across all the video dataset. As you can see the Harris detector performed marginally better, however when evaluating the average speed of detection of both detectors we found that the FAST corner detector performed approximately 5x faster.

5.3 Classification Results

To evaluate the performance of the classification stage we carried out a number of experiments; the testing procedures and the results from these experiments are detailed below.

5.3.1 Experiment 1

In the first experiment we were interested in obtaining an overall performance measure for the classification stage. This was achieved by evaluating its ability to correctly differentiate vehicle and non-vehicle regions in the video imagery. In order to do this we began by dividing the video dataset described above into a training and testing dataset; 80% of the video was used for training and the other 20% for testing. From the training dataset, vehicle and background images were manually cropped from the video using the method described in Chapter 4 and subsequently used to train the classifier.

We evaluated the accuracy of the classification stage over 178 frame of test video; for each frame the number of correctly detected vehicles (True Positives), the number of background regions that were incorrectly classified as vehicles (False Positives) and the number of vehicles that were not detected (False Negatives) was recorded. These results are presented in the confusion matrix below (**Table 1**). In this experiment we only considered vehicles that were successfully identified by the region of interest selection process; this ensured that we were isolating the performance of the classification stage in our evaluation. To summarise this data and provide a more informative measure we use the F-*measure* metric. The F-measure is a harmonic mean of precision P and recall R and is given by:

$$F_1 = 2 \cdot \frac{PR}{P + R} \tag{12}$$

The F-*measure* metric gives a value in the interval $(0,1)$ with larger value F-*measures* corresponding to a higher classification rate. The classification rate obtained by the proposed system was *0.78* and we believe that this measure provides a reasonably accurate estimate of the overall performance of the classification stage.

The above performance indicates a relatively high accuracy of the classifier correctly identifying a vehicle, however, we feel it is important to note that given the large number of regions returned by the ROI selection stage the obtained classification performance can still result in a poor overall system performance. For example, if you look at the test results you can see that in total 350 false positives were retuned by the classification stage in the experiment. Across 178 frames this results in approximately 2 background regions per frame that were incorrectly identified as vehicles, which in total accounts for around a third of the total detections per frame.

The high number of incorrect detection in this experiment can be accredited to the SVM's over generalising, as most of these false positives came from a large number of rectangular shaped, vehicle resembling landmarks in the test video. This highlights the weakness in using a shape based feature descriptor.

		Actual Class	
Predicted Class		Vehicle	Background
	Vehicle	725	350
	Background	57	-

Table 1: Classification - Experiment 1 Confusion matrix.

5.3.2 Experiment 2

In the second experiment we were interested in obtaining a performance measure for the classifier and also for the individual support vector machines. To achieve this we evaluated the classification accuracy for the classifier and the comprising SVMs in isolation. For this task vehicle and background images were extracted from all the videos in the dataset using the method described in section 4.2.3.1. The vehicle images were then categorised into the four vehicle orientation sets and then further divided into training and testing samples. To evaluate the performance of the classifier we trained and tested it on all vehicle orientation sets; for the SVM's, we trained and tested them only on the vehicles belonging to their associated orientation. The results of these tests are summarised in the confusion matrices in **Table 2** and **Table 3**. The classification accuracy and related F-*measure* for the

classifier and comprising SVM's is presented in **Table 4**. It is important to note that the accuracy obtained in these result may have a significant bias due the method used to extract the vehicle image from the dataset, and the homogeneity in the appearance of vehicle and background objects due to the limited size of the dataset. To produce more truthful results concerning the performance of the classifier, more tests would have to be conducted on a larger, more varied dataset.

		Actual Class	
Predicted Class		V	B
	V	114	1
	B	0	113

(a)

		Actual Class	
Predicted Class		V	B
	V	36	0
	B	0	114

(b)

		Actual Class	
Predicted Class		V	B
	V	33	0
	B	1	114

(c)

		Actual Class	
Predicted Class		V	B
	V	17	0
	B	0	114

(d)

Table 2: Classification Experiment 2 Results: Confusion matrices for SVMs trained and tested on vehicle images categorised into angular offsets (a) 0°, (b) 45°, (c) 90° and (d) 135° to the horizontal. (V – Vehicle, B –Background)

		Actual Class	
Predicted Class		Vehicle	Background
	Vehicle	114	1
	Background	0	113

Table 3: Classification Experiment 2: Confusion matrix for classifier evaluated on all test vehicle and background images.

	Accuracy	**F-*measure***
Classifier	0.9956	0.9956
0° offset SVM	0.9956	0.9965
45° offset SVM	1	1
90° offset SVM	0.9932	0.9851
135° offset SVM	1	1

Table 4: Accuracy and F-*measure* performance metrics for classifier and comprising support vector machines.

5.4 Tracking Results

To evaluate the overall performance of the system we performed an experiment to assess the system's ability to accurately detect and track vehicles. To perform this experiment we used the same separation of the video dataset into training and testing sets as used in the first classification experiment. In the experiment we obtained an accuracy measure for the system using the Multiple Object Tracking Accuracy (MOTA) metric proposed by Bernardin et al. (2006). This metric provides an accuracy score that considers the number of missed detections, the false positive rate and mismatches between tracker and vehicle pairs. Assuming for frame t, the number of missed detections is indicted by m_t, the number of false positives is indicted by fp_t and the number of tracker mismatches considering the association of detections in frame $(t-1)$ is indicated by $Tr\_SWITCHES_t$, we computed the MOTA as:

$$MOTA = 1 - \frac{\sum_{t=1}^{N_{frames}} \left(c_m(m_t) + c_f(fp_t) + c_s(Tr\_SWITCHES_t) \right)}{\sum_{t=1}^{N_{frames}} N_G^{(t)}} \quad (13)$$

Where the values used for the weighting functions in this evaluation were $c_m = c_f = c_s = 1$.

Using the MOTA metric our system obtained an accuracy of 0.46. This result would suggest relative poor overall system performance however we believe that this accuracy could be considerably increased with more training data and a more representative test set. For example, when we took a cross-validation approach to the analysis of the system performance, iteratively dividing the video dataset into

training and test sets and using the average of these results we found that this gave a MOTA of 0.67, which we believe gives a more accurate estimate of the overall system performance and also indicates that the results in the first instance may be biased due to the large number of false positives detected in the test set.

5.5 Real-time Performance

To evaluate the real-time performance of the system we measured the average processing time per frame for the entire video dataset. We obtained a result of 0.55 seconds per frame, approximately 2 frames per second. This result is far from the desired 25 frame per second that is needed for real-time, online performance. However, this result is highly depended on the processing power of the computer system used to run the test and we have no doubt that real-time performance could be achieved on a higher specification system.

6. Conclusion

In this report, we present an approach for the automatic detection and tracking of vehicles in aerial video. The proposed system is comprised of three stages namely; Region of Interest (ROI) Selection, Classification and Tracking. In the first stage we expedite detection by identifying image regions that are most likely to contain vehicles. To achieve this we use a fast corner detector combined with an efficient feature density estimation technique. In the classification stage the system uses shape encoding local features and robust machine learning techniques to perform efficient vehicle detection. In the final stage we propose a tracking algorithm, which employs particle filtering to exploit vehicle dynamics, and improve detector association.

To evaluate the proposed system we performed a number of experiments to test both the overall system performance, and the comprising stages individually. To evaluate the region of interest selection stage we tested its detection rate across the entire video dataset; this stage performed very well and achieved a detection rate of 98.3%. To evaluate the classification stage we assessed its ability to correctly classify the regions of interest selected in the previous stage, to do this we tested its classification accuracy over a subset of the dataset. This stage performed well and achieved an f-score of 0.78 which indicates a fairly high classification rate. Although these two stages performed well individually, when we evaluated the overall performance of the system, using the Multiple Object Tracking Accuracy (MOTA) measure our system only achieved a score of 0.67. This measure considered all the stages of the system and accounted for the number of missed detections, number of false positives and the number of tracker mismatches. Therefore, we consider this measure to be a good indicator of the overall performance of the system.

The proposed system demonstrates a reasonably high accuracy and is capable of successfully detecting and tracking a variety of differing vehicle types under varying rotation, sheering and blurring conditions. For these reason we believe the project

has been a success. That said, the system is far from a complete and in the following section we will recommend some areas for future work.

6.1 Future Work

The developed system is far from complete and there are a lot of opportunities for future work, below I identify aspects of the system that could be improved or extended:

- Improving the real-time capabilities of the system, potentially by further optimising the region of interest selection process and by parallelising the evaluation of the support vector machines that comprise the classifier. In addition a density based clustering method could be used to reduce the total number of regions presented to the classier.
- Investigating the performance of additional shape encoding features such as the Gradient Location and Orientation Histogram (GLOH) and the Local Energy based Shape Histogram (LESH).
- Investigating other machine learning techniques to classify vehicles such as Random Forests and boosting methods, that both demonstrated very high classification rates in previous literature (Chapter 2).
- Training the system with a larger more diverse dataset that has more variety in vehicle and background appearance, which would likely result in improved generalisation of the classifier and better performance in both the detection and tracking aspects of the system.
- Using the orientation information provided from the classifier to constrain the velocity direction of the particles that initially model the state of a vehicle in the particle filter. This would provide a better estimation of the dynamics of the vehicle and hence improve the precision and reliability of tracking.
- Using scene context to further improve the region of interest selection and classification stages of vehicle detection.

7. References

Bernardin, K., Elbs, A., & Stiefelhagen, R. (2006). Multiple object tracking performance metrics and evaluation in a smart room environment. *Sixth IEEE International Workshop on Visual Surveillance, in conjunction with ECCV*, **90**, 91.

Breckon, T. P., Barnes, S. E., Eichner, M. L., & Wahren, K. (2009). Autonomous real-time vehicle detection from a medium-level uav. *Proc. 24th International Unmanned Air Vehicle Systems, **29,*** 1.

Breitenstein, M. D., Reichlin, F., Leibe, B., Koller-Meier, E., & Van Gool, L. (2009). Robust tracking-by-detection using a detector confidence particle filter. *Computer Vision, 2009 IEEE 12th International Conference*, 1515-1522.

Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST),* **2(3),** 27.

Cheng, P., Zhou, G., & Zheng, Z. (2009). Detecting and counting vehicles from small low-cost UAV images. *ASPRS 2009 Annual Conference, Baltimore,* **3**, 9-13.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference*, **1**, 886-893

Forsyth, D.A. & Ponce, J. (2012) *Computer Vision: A Modern Approach*. 2nd edition. Pearson Education, Essex.

Gąszczak, A., Breckona, T. P., & Hana, J. (2011). Real-time people and vehicle detection from UAV imagery.

Gleason, J., Nefian, A. V., Bouyssounousse, X., Fong, T., & Bebis, G. (2011). Vehicle detection from aerial imagery. *Robotics and Automation (ICRA), 2011 IEEE International Conference*, 2065-2070

Hinz, S. (2003). Detection and counting of cars in aerial images. *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference* , **3**, III-997.

Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to support vector classification.

Lacey, G. (2004). Histogram of oriented gradients. [online] lecture notes distributed in CS4053 Course at Trinity College Dublin. Available at: <https://www.scss.tcd.ie/Gerard.Lacey/Gerard_Lacey_Homepage/CS4053_Course_files/Lecture%20009%20HOG-HOF.pdf> [Accessed 22 March 2013].

Nguyen, T. T., Grabner, H., Bischof, H., & Gruber, B. (2007). On-line boosting for car detection from aerial images. *Research, Innovation and Vision for the Future, 2007 IEEE International Conference,* 87-95 .

Rosten, E., & Drummond, T. (2005). Fusing points and lines for high performance tracking. *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference,* **2**, 1508-1515.

Rosten, E., & Drummond, T. (2006). Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006* , 430-443.

Russell, S. J., Norvig, P. (2010) *Artificial Intelligence: International Version: A Modern Approach,* 3rd edition, Prentice Hall

Sahli, S., Ouyang, Y., Sheng, Y., & Lavigne, D. (2010). Robust vehicle detection in low-resolution aerial imagery. *Proc. SPIE* , **7668**, 76680G.

Shi, X., Ling, H., Blasch, E., & Hu, W. (2012). Context-driven moving vehicle detection in wide area motion imagery. *Pattern Recognition (ICPR), 2012 21st International Conference*, 2512-2515.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference*, **1**, I-511.

Zhao, T., & Nevatia, R. (2001). Car detection in low resolution aerial image. *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference,* **1**, 710-717.