VIDEO OBJECT DETECTION AND MATCHING

A Thesis in

Computer Science and Engineering

by

Vladimir Y. Mariano

© 2003 Vladimir Y. Mariano

Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

December 2003
The thesis of Vladimir Y. Mariano was reviewed and approved* by the following:

Rangachar Kasturi  
Professor of Computer Science and Engineering  
Thesis Adviser  
Chair of Committee

Rajeev Sharma  
Associate Professor of Computer Science and Engineering

Octavia Camps  
Associate Professor of Electrical Engineering

Ageliki Elefteriadou  
Associate Professor of Civil and Environmental Engineering

Raj Acharya  
Professor of Computer Science and Engineering  
Head of the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School.
Abstract

The automatic analysis of video has come a long way from using global features to detection and matching of objects. We seek to understand two common classes of objects – text and vehicles. In unconstrained video, text occurs in a variety of ways including artificially embedded captions and those that naturally occur in the scene. We present a novel algorithm for detecting text in video frames. The constraints are that the text object is horizontally-aligned and the stroke pixels have uniform color. The color features derived by the detection algorithm are then used in a novel algorithm for finding similar text objects between two video frames. We believe that finding similar text objects is useful for finding related video clips and finding objects and scenes with similar text marks. Quantitative performance evaluation is important for optimizing, evaluating and comparing object detection algorithms. We present a set of six measures for quantifying different aspects of performance. Using these measures, the text detection algorithm was evaluated and compared to another text detection algorithm. Recognition of vehicles using visual features is useful for surveillance and traffic studies. We modified our text-detection method to find text marks on moving vehicles. We also explored a strategy for recognizing vehicles that repeatedly pass through a scene, utilizing feature clustering and a multi-object alignment framework.
# Table of Contents

List of Tables .......................................................... vii

List of Figures .......................................................... viii

Acknowledgments ......................................................... xii

Chapter 1.  Introduction ...................................................... 1
  1.1 Objects in Video ..................................................... 2
  1.2 Organization .......................................................... 3

Chapter 2.  Detection of Text ............................................. 6
  2.1 Introduction and Previous Work ................................. 6
  2.2 Observations .......................................................... 7
  2.3 A Color-Based Method .............................................. 7
    2.3.1 Clustering in $L^*a^*b^*$ space ............................... 7
    2.3.2 Determining bounding rows ................................... 8
    2.3.3 Finding text blocks ............................................. 9
    2.3.4 Fusing the detected text blocks .............................. 9
  2.4 Experiments and Preliminary Results ........................... 10
  2.5 Future Work ........................................................ 10

  3.1 Introduction ......................................................... 19
3.2 Previous Work ............................................. 20
3.3 Six Performance Measures ............................... 21
   3.3.1 Area-Based Precision for Frame .................... 21
   3.3.2 Average Fragmentation ............................... 22
   3.3.3 Average Object Area Recall ......................... 23
   3.3.4 Average Detected Box Area Precision ............... 24
   3.3.5 Localized Object Count Recall ..................... 25
   3.3.6 Localized Output Box Count Precision ............. 26
3.4 Implementation and Future Work ...................... 27
3.5 Combining Performance Measures ...................... 28
3.6 Some Cases and Computed Measures .................... 29

Chapter 4. Matching Text Objects .......................... 36
   4.1 Introduction ........................................... 36
   4.2 Previous Work ........................................ 38
   4.3 Matching Text Regions ................................ 38
   4.4 Experiments and Preliminary Results ................ 43
   4.5 Discussion and Future Work .......................... 44

Chapter 5. Detailed Vehicle Recognition .................... 46
   5.1 Detection of Text Marks on Moving Vehicles ........ 46
      5.1.1 Related Work .................................... 48
      5.1.2 Detection of Vehicle Text ....................... 49
         5.1.2.1 Vehicle Segmentation and Orientation .... 49
List of Tables

3.1 An evaluation and comparison of two text detection algorithms. The first is the method in Chapter 2 and the second is by the University of Maryland. . . . . . 28
List of Figures

2.1 Computing the bounding rows. One of the color clusters in row R are marked as short streaks and pixels of the text “tough crop” lie within the range of values of the cluster. Each segment of the bounding top ($R_a$) and bottom ($R_b$) rows are shown separated for clarity even when they are actually on the same row. 11

2.2 Analysis of a video frame and detected text block. 12

2.3 Pixels in text block belonging to color cluster in Fig. 2.2 13

2.4 Caption text on non-uniform background. The two small boxes in the first image are faint text watermarks. 14

2.5 Text on uniform and non-uniform background. 15

2.6 Examples of detected text with false alarm. 16

2.7 Text with different sizes and colors. 17

2.8 Big and small low contrast text. 18

3.1 Fragmentation measure. 30

3.2 Measures computed on different cases. 31

3.3 Measures computed on different cases. 32

3.4 Measures computed on different cases. 33

3.5 Measures computed on different cases. 34

3.6 Measures computed on different cases. 35
4.1 Detection of a text block. The intersection of row $R$ with the character strokes (dark streaks) are detected as one color cluster during the clustering of row $R$’s pixels. A search in $R$’s neighborhood for pixels within the cluster’s range allows the computation of the bounding rows and the bounding box. The cluster’s centroid becomes the box’s color feature. ............................. 39

4.2 Multiple boxes. Multiple rows passing through a text object and multiple colors composing the character strokes result in detection of multiple overlapping boxes. ............................................... 40

4.3 Some of the detected matches. The query frame is on the left side and the database frame is on the right. Each text regions is the union of a set of overlapping boxes. ............................................. 45

5.1 A typical street scene with vehicles and moving people. Our goal is to find any text marks on passing vehicles. .................................................. 68

5.2 A magnified view of the red text mark of the bus in Figure 5.1. ............................... 69

5.3 A box truck with text marks. Washington, D.C. police gave this image as a description of the vehicle linked to the area sniper. .............................. 70

5.4 Computing the bounding rows. One of the color clusters in row $R$ are marked as short streaks and pixels of the text “Ryder” lie within the ellipsoid of the cluster. The spacings between the streaks are used to compute the left and right boundaries of the text bounding box. ............................................. 71
5.5 Binary images generated by color ellipsoid membership. These overlapping text boxes are detected on the “Balfurd” text in Figure 5.7. For each text box, a single color cluster from the pixels of a single row is used to compute an ellipsoid in color space and spatially find the bounding box. The color ellipsoid is then used to generate the binary image (black = 1).

5.6 Binary image of a false alarm. In this example, the analyzed row passed through the vehicle windows whose sides were mistaken for character strokes.

5.7 Most text are parallel to the vehicle body. The moving vehicle is extracted from the scene and oriented horizontally before applying the text detection algorithm. The angle of rotation can be derived using the angle of the vehicle’s parallel edges, the direction of travel, or prior knowledge of the scene. The “Balfurd” image shows what the text looks like after rotation.

5.8 The test street scene, a moving truck segmented from the scene, and the detected text mark.

5.9 Overview of vehicle recognition system

5.10 Detection and alignment of correspondence clusters. Common spatial features are detected in the query object image and search space object images. The common features form a cluster, with a cluster prototype as the center. During re-alignment, the object image features are driven toward the cluster prototype.

5.11 Analogy of “outlier objects”. Points in this space represent objects in the reduced search space, clustered around the query. Outliers are the objects with the least correspondence with the query. Removing the outliers results in a smaller, more homogenous and better aligned search space.
5.12 Nine points for initial alignment. The three horizontal coordinates are the 25th, 50th, and 75th percentile in the X-coordinate spatial distribution. The three vertical coordinates are computed in the same way.

5.13 Three passes of the same car. A friend is asked to drive the same car through the imaged street scene. The second drive is used as the query. On the third drive, four black stickers are attached on the roof and side.

5.14 Outlier score vs. Iteration. In each iteration, the score of the removed object is recorded. The trend shows increasing correspondence with the query.

5.15 Intra-cluster standard deviation vs. Iteration. The shrinking clusters indicates smaller feature variation among objects.

5.16 Average object alignment error vs. Iteration. Average error decreases, indicating closer alignment of search.

5.17 Maximum object alignment error vs. Iteration. At each iteration, the maximum error among objects is recorded. The decreasing error shows closer alignment of objects.

5.18 Outlier removal history. For comparison, the query is shown in the last image. The two occurrences of the same white sedan were the last in the history. The white sedan with the black sticker was removed before the one without the sticker.
Acknowledgments

I am most grateful and indebted to my thesis advisor, Prof. Rangachar Kasturi, for taking me as his student and his patience, encouragement and generosity during my time at Penn State. I am also grateful for the financial support he has given me in the past two years and all the conferences we attended. I thank the other members of my committee, Dr. Rajeev Sharma, Dr. Octavia Camps and Dr. Lily Elefteriadou for their ideas, comments and interest in my topic. I thank the Philippine Department of Science and Technology for the financial support for my first four years at PSU. I also thank James Simpas for helping me with some video data collection and Surapong Lertrattanapanich for the discussions on image registration. I thank my fellow students Sameer Antani, David Crandall, Junghye Min, JinHyeong Park, Anand Narasimhamurthy for the times in the lab and opportunities to work together. Finally, I thank my wife Thennie, our Baby Aiden and the Lord Jesus Christ for the love, inspiration, and grace to finish.
Chapter 1

Introduction

Objects are everywhere. We see them everywhere we go. The presence of objects help us identify a certain place or scene. We recognize objects by their features – a combination of shape, color, marks, texture, specularity, their function, etc. These features allows us to make classifications of objects. Examples of general classes of objects are people, vehicles, computers, text. Beyond object classification, we go further into object association, activity detection, and formulation of higher-level concepts like “party”, “car crash”, or “advertisement”.

The amount of video data that the world is producing demanded methods and systems for management and retrieval. The cost of manual annotation and analysis is prohibitively expensive. Thus we need automated ways to do this.

The ever-increasing processing power available to us has allowed sophisticated methods to analyze images and video sequences. This, coupled with terabytes of storage distributed in a global network (i.e. the Internet) enable us to experiment with databases way beyond what was available a few years ago.

Video is the ultimate medium for communicating events. It brings us to where the action is, complete with sound. It has extended the informative power of images by adding the element of time. Photos were extended to moving pictures, and objects become animated.

The computer vision community has sought to model how humans understand video. A lot of work has gone into segmentation of video into logical units, organization of scenes, and
retrieval of similar video clips by example and by sketch. Hopefully, the methods developed might be able to help solve the problem of automatic video analysis.

1.1 Objects in Video

Analysis of video has ranged from global features to extraction of objects and their features. The need for object-based methods comes from the inherent complexity of the scene and the nature of video which allowed motion and temporal redundancy to be used as strong features for extracting objects.

Extracting the objects from the video is the first step in object-based analysis. Systems such as Netra-V [9] and VideoQ [35] used spatio-temporal segmentation to extract regions which are supposed to correspond to video objects. Others [21] have resorted to user-assisted segmentation. Although features were computed from the segmented regions, there is no attempt to establish the class of the objects that the regions represent. VideoQ enables queries of multiple objects going in different trajectories.

In the past few years, several efforts were undertaken to find objects belonging to specific classes. The presence of objects are useful indices in video. Recent systems like the Informedia Project [39] include in its generation of video summaries the detection of common objects like text and faces. This first step of detection is needed before further analysis like text reading, face recognition, and similarity.

An effective method for recognizing object in images is to specify an object model for each object class of interest and find regions in the image that conform to that model [14]. Different models were used in methods for finding objects that are common in video. These
include methods for finding text [27, 42], faces [33, 34] and vehicles [34]. Vailaya [38] used local features to detect sky and vegetation to identify outdoor scenes.

In this thesis, we seek to understand two important classes of objects found in video – text and vehicles. Occurrence of text in video is viewed as unconstrained whereas vehicles are studied in a constrained street scene.

Text is a very common object in video. It can be treated as an object in itself or a feature mark of a object (e.g. name on a football jersey, or a “Taxi” sign). In broadcast video, caption text is overlaid on scenes during video editing. The uniform editing of text for specific portions of TV programs (e.g. text to introduce people interviewed on CNN) make some text objects visually similar. We introduce methods for detecting and matching text objects in video. The detection method is constrained to finding text which is horizontally aligned and uniform in color.

Performance evaluation is an important issue in the development of object detection algorithms. Quantitative measures are useful for optimizing, evaluating and comparing different algorithms. For object detection, we believe there is no single measure for quantifying the different aspects of performance. In this thesis, we present a set of performance measures for determining how well a detection algorithm’s output matches the ground truth.

The observation of vehicles has gained some attention in the past few years [18, 5, 34, 37, 32]. Applications include license plate reading, surveillance and traffic studies. We explore methods for recognizing vehicles using visual features.

1.2 Organization

This thesis is summarized as follows:
In chapter 2 a method is presented for locating horizontal, uniform-colored text in video frames. It was observed that when a row of pixels across such a text region is clustered in perceptually uniform $L^*a^*b^*$ color space, the pixels of one of these clusters would belong to the text strokes. These pixels would appear as a line of short streaks on the row since a typical text region has many vertical and diagonal strokes. The method examines every third row of the image and checks whether this row passes through a horizontal text region. For a given row $R$, the pixels of $R$ are hierarchically clustered in $L^*a^*b^*$ space and each cluster is tested whether similar-colored pixels in $R$’s vicinity are possibly part of a text region. Candidate text blocks are marked by heuristics using information about the cluster’s line of short streaks. The detected text blocks are fused to come up with the text regions. The method was tested on key frames of several video sequences and was able to locate a wide variety of text.

In chapter 4, we introduce the first method that automatically detects text and finds similar text objects in a video database. A basic similarity algorithm takes two frames and finds text objects that are similar. The overlapping sets of boxes and color features computed during the detection process are used to determine similarity. A system is built that uses this algorithm to take a query frame, extract text from it, and look for similar text objects in a database of video frames. Experiments were performed on a diverse collection of key frames.

A set of six measures are presented (Chapter 3) for quantifying different aspects of a detection algorithm’s performance. These are implemented in the ViPER Performance Evaluation tool and will be used in the future to optimize and evaluate algorithms for detecting text, faces, moving people and vehicles. Results are presented for an evaluation run of two text detection algorithms.
Vehicle text marks are unique features which are useful for identifying vehicles in video surveillance applications. In chapter 5, we apply present a method for finding such text marks. Our text detection algorithm is modified such that detection is increased and made more robust to outdoor conditions. False alarm is reduced by introducing a binary image test which remove detections that are not likely to be caused by text. The method is tested on a captured video of a typical street scene.

Also in chapter 5, we explore a method for detailed recognition of vehicle images. Given the output of a general image retrieval system, an iterative scheme uses multi-object feature clustering and image alignment to reduce the number of images in the search space. In each iteration, the reduced search space becomes more homogeneous and better aligned. The reduction scheme uses the cluster membership of object features to decide which objects will be pruned from the search space. The method is applied to recognition of recurring vehicles in a street scene.

Finally, a summary of research challenges, contributions and publications is in Chapter 6.
Chapter 2

Detection of Text

2.1 Introduction and Previous Work

Large amount of stored video data has motivated much research in content-based video retrieval for several years. Detecting text in digital video has been an active area of research with the goal of identifying keywords [19, 36] for video characterization.

Different methods were used in recent work for detecting text. Kanade et. al. [19] combined interpolation, multi-frame integration, character extraction and recognition for text segmentation. Li, Doermann and Kia [23] used a hybrid wavelet/neural network segmenter to segment text regions. Zhong et. al. [10] used intensity variation information encoded in the DCT domain to locate caption text regions.

A few methods have used color information for detecting text in video frames. Jain and Yu [17] used multi-valued image decomposition on video frames to obtain subimages with different colors. Connected component analysis is used to find text regions. A method by Gargi, Antani and Kasturi [16] first creates a “slats” image where a slat is a segment of a scan line between two color edges of minimum height. Slats on consecutive scan lines that are similar in color and are aligned are merged to form stroke segments which correspond to character strokes.

In this chapter we present a method for locating text based on color uniformity of text pixels and horizontal alignment. An image is searched for rows that pass through vertical and diagonal strokes of text.
2.2 Observations

Caption text is embedded in video as a horizontal group of characters with the same color. Degradation due to digitization (ex. MPEG encoding) would still retain color uniformity among the text characters. An experiment was made by taking a row of pixels that goes through the middle of caption text on a non-uniform background. These pixels were clustered in $L^*a^*b^*$ space and each cluster was marked back on the row. One of the clusters clearly marked the pixels belonging to the text and appeared as short streaks. When other pixels in the image having the same range of color values as this cluster were marked, we observed that the top pixels of the characters are included in this group. Furthermore, these top pixels were aligned horizontally with deviation of at most a few pixels. The same is observed with the bottom pixels. Using these observations, heuristics were formed for detection of horizontal text.

2.3 A Color-Based Method

The algorithm visits every $Interval$ rows in the image. $Interval$ is set small enough to be able to detect very small text and large enough so as not to consume too much time. In our experiment we set $Interval = 3$. Given a row $R$ on the image, we want to find out whether or not $R$ passes through the middle of a text region.

2.3.1 Clustering in $L^*a^*b^*$ space

The pixels of $R$ are transformed and clustered in the perceptually uniform $L^*a^*b^*$ color space using hierarchical clustering. The algorithm first assigns each pixel as a cluster and the distance of pairs of clusters are stored in an array. Two clusters $A$ and $B$ are merged if $\|\mu_A - \mu_B\|$
is minimum and for each pixel \( p \) in \( A \cup B \), \( \| p - \mu_{A\cup B} \| < \text{MaxClusterRadius} \), where \( \mu_Z \) is the mean \( L^*a^*b^* \) vector of cluster \( Z \) and \( \| \cdot \| \) is the weighted Euclidean norm. The weighted norm was used to achieve a slight invariance to lightness (weights: \( L^* = 0.8, a^* = 1.1, b^* = 1.1 \)). In our experiments, we set \( \text{MaxClusterRadius} = 10 \) (ranges: \( L^* = 0 \ldots 100, a^* = -97 \ldots 88, b^* = -100 \ldots 88 \)). Merging continues until no two clusters can be merged.

### 2.3.2 Determining bounding rows

Each cluster \( C \) is tested to see if it contains pixels belonging to text. Locating the bounding rows (top and bottom rows of text) is the first step (Fig. 2.1). The cluster points are marked back on row \( R \) to create streaks \( S_i, i = 1 \ldots N_s \) (where \( N_s \) = number of streaks) of pixels in the row \( R \). Then all pixels in the entire image are examined and each pixel with a value within the range of values represented in the cluster are colored with a value of \( T \). All other pixels are marked \( T' \).

We now try to find out if there are bounding rows above and below \( R \) which may contain horizontal text. Given a pair of adjacent streaks \( S_i \) and \( S_{i+1} \), we find \( R_a \) – the first row above \( R \) in which the segment covering \( S_i \) and \( S_{i+1} \) is colored \( T' \). We also find \( R_b \) – the first row below \( R \) in which the segment covering under \( S_i \) and \( S_{i+1} \) is colored \( T' \). The \( R_a \) of each pair of adjacent streaks is computed and collected in an alignment histogram \( H_a \), where the bins are the rows of the image. \( H_b \) is computed in the same way by taking all the \( R_b \)’s. We declare the existence of a bounding row \( B_a \) if at least 60% of the elements in \( H_a \) are contained in three or fewer adjacent histogram bins. \( B_b \)’s existence is computed in the same way from \( H_b \). If \( B_a \) and \( B_b \) exists, \( \text{height} \) is defined as their difference.
If the cluster $C$ contains text pixels, then $B_a$ and $B_b$ would mark the text block’s upper and lower row boundaries, and $height$ would define its vertical dimension. Figure 2.1 illustrates the computation of $B_a, B_b$ and $height$.

### 2.3.3 Finding text blocks

We look for text blocks using heuristics on $height$ and the short streaks’ lengths and gaps. Streaks longer than $height$ are discarded and added to the gaps. Gaps longer than $height$ are considered not part of a text block. The remaining regions are now smaller blocks with short streaks. If a block’s width is greater than $1.5 \times height$ and the number of short streaks inside is greater than 3, then it is considered a text block, otherwise it is discarded. Finally, the text block is expanded a few pixels to the left and right to ensure full coverage of the characters at the ends.

Figure 2.2 shows how the text block "For generations" is detected. The pixels of row R (passing through the middle of text) are clustered in color space. One of the color clusters is marked black. Pixels in the image having similar color as the black ones are marked white. On the left side of the image, the two alignment histograms $H_a$ (above R) and $H_b$ (below R) are used to mark the bright bounding rows $B_a$ and $B_b$. The short streaks marked black and the $height$ between the bounding rows are used to find the text block. The two black streaks on the right were not included in the text block because their gap from the other streaks is greater than $height$. Figure 2.3 shows the text block binarized according to the color cluster.

### 2.3.4 Fusing the detected text blocks

It was observed that other color clusters were caused by the presence of text. The characters’ color “shadows” and the pixels in the transition from text foreground to background result in
other detected text blocks which largely overlap with the foreground text block. All the detected
text blocks are fused (set union) to come up with the final regions of text.

2.4 Experiments and Preliminary Results

The method was tested on a variety of video frames from news clips, sports reviews,
commercials and movie previews. Several video sequences totalling 14 minutes were manually
segmented and 213 key frames were extracted. The algorithm was able to find captions (Fig. 2.4),
text on uniform and non-uniform background (Fig. 2.5), text with different sizes and colors
(Fig. 2.7), and low contrast text (Fig. 2.8). Figure 2.6 shows typical false alarms which include
fences and vertical stripes. The overall detection rate was 94% while the false alarm rate was
39%.

2.5 Future Work

This chapter introduced a new method for locating horizontal, uniform-colored text in
video frames. Every third row is tested to determine whether it passes through text by analyzing
color clusters in $L^*a^*b^*$ space. The method is able to identify regions of text of different sizes,
in uniform and non-uniform backgrounds. We are currently working on modifying the method
to detect non-horizontal text.
Fig. 2.1. Computing the bounding rows. One of the color clusters in row R are marked as short streaks and pixels of the text “tough crop” lie within the range of values of the cluster. Each segment of the bounding top ($R_a$) and bottom ($R_b$) rows are shown separated for clarity even when they are actually on the same row.
Fig. 2.2. Analysis of a video frame and detected text block.
Fig. 2.3. Pixels in text block belonging to color cluster in Fig. 2.2
Fig. 2.4. Caption text on non-uniform background. The two small boxes in the first image are faint text watermarks.
Fig. 2.5. Text on uniform and non-uniform background.
Fig. 2.6. Examples of detected text with false alarm.
Fig. 2.7. Text with different sizes and colors.
Fig. 2.8. Big and small low contrast text.
Chapter 3

Performance Evaluation of Object Detection Algorithms

3.1 Introduction

Many algorithms have been developed for detecting objects in images and video. These include methods for detecting text [27, 23], faces [31, 34], vehicles [34] and moving people [8]. With many new algorithms emerging, rules for quantitative measurement of performance are necessary.

The performance of a detection algorithm can be measured by comparing its output with the ground-truth, where a human operator has marked the boundaries of the target objects. Tools for ground-truthing video sequences have been developed in [11] and [41].

In this chapter, we present several measures which can be used to determine how well the output of object detection algorithms matches the ground truth. Different measures capture different aspects of performance. The objects considered here are compact objects – those that can be covered by simple bounding shapes. For other applications, pixel-based representation of the data may be more appropriate.

The use of simple bounding shapes allows inexpensive ground-truthing. This would enable a large volume and variety of video data to be ground-truthed using an intuitive interface such as ViPER [11]. In contrast, ground-truthing of individual pixels (e.g. text pixels) is expensive which makes it prohibitive for large and diverse data sets. Furthermore, the output of most detection algorithms are presented as simple bounding shapes (e.g. boxes) which are consistent
with the ground-truth. Beyond bounding shapes, criteria for detection might include heuristics on specific object class features. For example, an algorithm’s coverage of two eyes and a mouth (ground-truthed features in addition to the face bounding box) could be used as a criteria for successful face detection [31]. This work covers only the use of bounding shapes which is applicable to many classes of objects in video.

Using the proposed measures, we can do the following:

- Algorithm parameters can be optimized for a particular set of measures.
- The performance of an algorithm for different kinds of data can be compared.
- Quantitative comparison of different detection algorithms is possible.
- In the course of an algorithm’s development, any performance improvement can be measured.
- Tradeoffs between performance aspects can be determined.

We believe no single measure can quantify all the different aspects of performance. Furthermore, developers may be interested in optimizing on a small set of performance aspects, thus a single universal measure would not be suitable.

3.2 Previous Work

Viper [11] was developed as a tool for ground-truthing video sequences. Objects are marked by a bounding box and object detection algorithms are evaluated using temporal and spatial measures. In [41], text objects are assigned a value for detection importance and detection difficulty which are factored in the computation of detection rate. In [3], five algorithms for text
detection were compared. The ground-truth and the algorithm output are considered as binary pixel maps (text = 1, non-text = 0). Recall and precision are computed using the intersection of the binary images.

### 3.3 Six Performance Measures

The following measures are described below along with their advantages and disadvantages. All the measures’ values range from zero to one (perfect). The different cases in Section 3.6 demonstrate their effectiveness and the tradeoffs between measures.

They are defined first for a single frame and extended to an entire video sequence. We use the term “ground truth object” to denote the ground-truth bounding box marked around image of the object. Detections are marked by an algorithm using “output boxes”. The bounding shapes can also be extended to ellipses (suitable for faces) or arbitrary polygons.

Let $G^{(t)}$ be the set of ground truth objects in a single frame $t$ and let $D^{(t)}$ be the set of output boxes produced by the algorithm.

#### 3.3.1 Area-Based Precision for Frame

The measure quantifies how well the algorithm minimized false alarms. This is a pixel-count-based measure. Initially it is computed for each frame, and for the whole data set it is the weighted average.

Let $U_{G^{(t)}}$ and $U_{D^{(t)}}$ be the spatial union of the boxes in $G^{(t)}$ and $D^{(t)}$:

\[
U_{G^{(t)}} = \bigcup_{i=1}^{N_{G^{(t)}}} G^{(t)}_i \quad \quad \quad U_{D^{(t)}} = \bigcup_{i=1}^{N_{D^{(t)}}} D^{(t)}_i
\]
For a single frame $t$, we define $Prec(t)$ as the ratio of the detected areas in the ground truth with the total detection:

$$
Prec(t) = \begin{cases} 
\text{undefined} & \text{if } U_{D(t)} = \emptyset \\
1 - \frac{|U_{D(t)} \cap U_{G(t)}|}{|U_{D(t)}|} & \text{otherwise}
\end{cases}
$$

$OverallPrec$ is the weighted average precision of all the frames.

$$
OverallPrec = \begin{cases} 
\text{undefined} & \text{if } \sum_{t=1}^{N_f} |U_{D(t)}| = 0 \\
\frac{\sum_{t=1}^{N_f} |U_{D(t)}| \times Prec(t)}{\sum_{t=1}^{N_f} |U_{D(t)}|} & \text{otherwise}
\end{cases}
$$

where $N_f$ is the number of frames in the ground-truth data set and the $||$ operator denotes the number of pixels in the area.

This measure treats the frame not as collection of objects but as a binary pixel map (object/non-object; output-covered/not-output-covered). A similar precision measure was used in [3] for comparing five text-detection algorithms.

### 3.3.2 Average Fragmentation

Detection of objects is usually not the final step in a vision system. For example, extracted text from video will go through enhancement, binarization and finally recognition by an OCR system. Ideally, the extracted text should be in one piece, but a detection algorithm could produce several boxes (e.g. one for each word or character) or multiple overlapping boxes which could increase the difficulty for the next processing step.
The measure is intended to penalize an algorithm for multiple output boxes covering a ground-truth object. Multiple detections include overlapping and non-overlapping boxes.

For a ground-truth object $G_i^{(t)}$ in frame $t$, the fragmentation of the output boxes overlapping the object $G_i^{(t)}$ is measured by:

$$Frag(G_i^{(t)}) = \begin{cases} \text{undefined} & \text{if } N_{D(t) \cap G_i^{(t)}} = 0 \\ \frac{1}{1+\log_{10}(N_{D(t) \cap G_i^{(t)}})} & \text{otherwise} \end{cases}$$

where $N_{D(t) \cap G_i^{(t)}}$ is the number of output boxes in $D(t)$ that overlap with the ground-truth object $G_i^{(t)}$. Figure 3.1 shows the function $Frag(N) = \frac{1}{1+\log_{10}N}$.

For a single frame $t$, $Frag(t)$ is simply the average fragmentation of all ground truth objects in frame $t$ where $Frag(G_i^{(t)})$ is defined.

$OverallFragmentation$ is defined as average fragmentation for all ground-truth objects in the entire data set.

### 3.3.3 Average Object Area Recall

This measure is intended to measure the average area recall of all the ground-truth objects in the data set. The recall for an object is the proportion of its area that is covered by the algorithm’s output boxes. The objects are treated equally regardless of size.

For a single frame $t$, we define $Recall(t)$ as the average recall for all the objects in the ground truth $G^{(t)}$.
Recall(t) = \sum_{\forall G_i(t)} \frac{ObjectRecall(G_i(t))}{N_{G(t)}}

where \( ObjectRecall(G_i(t)) = \frac{|G_i(t) \cap U_{D(t)}|}{|G_i(t)|} \)

and the || operator denotes the number of pixels in the area. Finally, \( OverallRecall \) is the weighted average recall of all the frames.

\[ OverallRecall = \frac{\sum_{t=1}^{N_f} N_{G(t)} \times Recall(t)}{\sum_{t=1}^{N_f} N_{G(t)}} \]

All the ground-truth objects contribute equally to the measure, regardless of their size. On one extreme, if a frame \( t \) contains two objects – a large object that was completely detected and a very small object that was missed, then \( Recall(t) \) would be 50%.

### 3.3.4 Average Detected Box Area Precision

This is a counterpart of the previous measure 3.3.3 where the output boxes are examined instead of the ground-truth objects. Precision is computed for each output box and averaged for the whole frame. The precision of a box is the proportion of its area that covers the ground truth objects.

For a single frame \( t \), we define \( Precision(t) \) as the average precision of the algorithm’s output boxes \( D(t) \):
\[ \text{Precision}(t) = \frac{\sum_{\forall D_i^{(t)}} \text{BoxPrecision}(D_i^{(t)})}{N_{D(t)}} \]

where \( \text{BoxPrecision}(D_i^{(t)}) = \frac{|D_i^{(t)} \cap U_{G(t)}|}{|D_i^{(t)}|} \)

and the \( \| \) operator denotes the number of pixels in the area. \( \text{OverallPrecision} \) is the weighted average precision of all the frames.

\[ \text{OverallPrecision} = \frac{\sum_{t=1}^{N_f} N_{D(t)} \times \text{Precision}(t)}{\sum_{t=1}^{N_f} N_{D(t)}} \]

In this measure the output boxes are treated equally regardless of size.

### 3.3.5 Localized Object Count Recall

In this measure, a ground-truth object is considered detected if a minimum proportion of its area is covered by the output boxes. Recall is computed as the ratio of the number of detected objects with the total number of ground-truth objects.

Let \( \text{Loc} \_ \text{Obj} \_ \text{Recall}(t) \) be the number of detected objects in frame \( t \):

\[ \text{Loc} \_ \text{Obj} \_ \text{Recall}(t) = \sum_{\forall G_i^{(t)}} \text{ObjDetect}(G_i^{(t)}) \quad \text{where} \]
\[ \text{ObjDetect}(G_i^{(t)}) = \begin{cases} 
1 & \text{if } \frac{|G_i^{(t)} \cap D_i^{(t)}|}{|G_i^{(t)}|} > \text{OverlapMin} \\
0 & \text{otherwise} 
\end{cases} \]

Here \text{OverlapMin} is the minimum proportion of the ground-truth object’s area that should be overlapped by the output boxes in order to say that it is correctly detected by the algorithm.

\text{Overall\_Loc\_Obj\_Recall} is the ratio of detected objects with the total number of objects in the ground-truth:

\[
\text{Overall\_Loc\_Obj\_Recall} = \frac{\sum_{f=1}^{N_f} \text{Loc\_Obj\_Recall}(t)}{\sum_{f=1}^{N_f} N_{G_i^{(t)}}}
\]

Again, the ground-truth objects are treated equally regardless of size.

### 3.3.6 Localized Output Box Count Precision

This is a counterpart of measure 3.3.5. The measure counts the number of output boxes that significantly covered the ground truth. An output box \(D_i^{(t)}\) significantly covers the ground-truth if a minimum proportion of its area overlaps with \(U_{G_i^{(t)}}\).

Let \textit{Loc\_Box\_Count}(t) be the number of output boxes that significantly overlap with the ground-truth objects in frame \(t\):

\[
\textit{Loc\_Box\_Count}(t) = \sum_{\forall D_i^{(t)}} \text{BoxPrec}(D_i^{(t)}) \quad \text{where}
\]
\[
\text{BoxPrec}(D_i^{(t)}) = \begin{cases} 
1 & \text{if } \frac{|D_i^{(t)} \cap G(t)|}{|D_i^{(t)}|} > \text{OverlapMin} \\
0 & \text{otherwise}
\end{cases}
\]

Here \(\text{OverlapMin}\) is the minimum proportion of the output box’s area that should be overlapped by the ground truth in order to say that the output box is precise.

\(\text{Overall Output Box Prec}\) is the ratio of precise output boxes with the total number of output boxes produced by the algorithm:

\[
\text{Overall Output Box Prec} = \frac{\sum_{f=1}^{N_f} \text{Loc Box Count}(t)}{\sum_{f=1}^{N_f} N_D(t)}
\]

Again, in this measure the output boxes are treated equally regardless of size.

### 3.4 Implementation and Future Work

The measures were implemented in ViPER [11] and were used in the development and evaluation of algorithms for detecting text, faces, moving people and vehicles. In the development stages, the measures are used to optimize algorithm parameters for a training data set. Evaluation on a test data set is then performed using the same set of measures. Periodic evaluation is done to monitor the progress of algorithm development. The detection algorithms currently considered for evaluation look for object instances in single images. The evaluation techniques need to be extended to algorithms that detect objects that persist in time and space (i.e. objects in video).
We used the measures to evaluate and compare two text-detection algorithms, one from Penn State University [27] (also described in Chapter 2) and the other from the University of Maryland (UMD) [23]. These were run on 1291 video key frames ground-truthed at UMD using the ViPER Ground-Truthing Tool. The following overall results were obtained:

<table>
<thead>
<tr>
<th>Measure</th>
<th>PSU</th>
<th>UMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area-Based Recall for Frame</td>
<td>0.59</td>
<td>0.56</td>
</tr>
<tr>
<td>Area-Based Precision for Frame</td>
<td>0.29</td>
<td>0.32</td>
</tr>
<tr>
<td>Average Fragmentation</td>
<td>0.97</td>
<td>0.98</td>
</tr>
<tr>
<td>Average Object Area Recall</td>
<td>0.58</td>
<td>0.57</td>
</tr>
<tr>
<td>Average Detected Box Area Precision</td>
<td>0.27</td>
<td>0.31</td>
</tr>
<tr>
<td>Localized Object Count Recall</td>
<td>0.58</td>
<td>0.57</td>
</tr>
<tr>
<td>Localized Output Box Count Precision</td>
<td>0.22</td>
<td>0.21</td>
</tr>
</tbody>
</table>

Table 3.1. An evaluation and comparison of two text detection algorithms. The first is the method in Chapter 2 and the second is by the University of Maryland.

3.5 Combining Performance Measures

The measures described above gives us a good set of measures to quantify performance. This measure set is actually designed as a model of how a user perceives performance. The reason we have a set is that each measure tries to capture just one aspect of performance – credit is given when the output boxes satisfy the measure’s criteria and a low score is given otherwise. Regardless of how good this criteria is, we can always find a case where either (1) visually, the algorithm did well yet recieved a low score for the measure, or (2) the algorithm got a high
score even if, visually, the output is nowhere near good. Thus each measure has advantages and disadvantages.

3.6 Some Cases and Computed Measures

Figures 3.2 to 3.6 show some cases of ground-truth objects and output boxes. Computed values of the measures are listed.
Fig. 3.1. Fragmentation measure.
Fig. 3.2. Measures computed on different cases.
Fig. 3.3. Measures computed on different cases.
Fig. 3.4. Measures computed on different cases.
Fig. 3.5. Measures computed on different cases.
Fig. 3.6. Measures computed on different cases.
Chapter 4

Matching Text Objects

4.1 Introduction

The exploding amount of digital video has pushed the demand for tools to extract important information used for indexing and organizing content. A class of tools for understanding video are methods which find classes of objects like faces, vehicles, and text. Beyond detection, finding similarities between objects is valuable for establishing semantic links between different video clips. For example, the visual similarity between two text objects of two different video clips could be the result of an “event” that is common to both clips. The following are some “events” that lead to similarity of text objects:

- Two different broadcasts of a TV program are edited the same way making their text visually similar.

- Two video clips were shot in the same place. An incidental background text appears on both clips.

- Two video clips contain the same object. Surface text identifies the object.

The two text objects share the same visual features as a result.

Text similarity can identify other clips that came from the same source. Same source could mean that they came from the same program or they were shot in the same location. In the first case, the text in the two broadcast programs have the same characteristics even if the actual
characters are different. For example, when people are interviewed at CNN News, the person’s name appears as caption text. This is generated when the video was edited and is also done when other “interview” or “talking person” footage are edited.

It can also help find objects that reoccur in other video clips. For example, a company could try to find the video clips where their bathroom cleaner product is seen. One of the identifying features of the product is its text label. This also applies to finding other objects like billboards and vehicles.

Text similarity can also be used as a tool in creating a mosaic of text objects. This mosaic can be presented to a user as a summary of the objects of interest. The user picks those that are interesting. The grouping makes the organized mosaic easier to search.

Finding similar text is also useful to find text that an operation is designed to work on. For example, when a method M is developed to binarize and read text constrained to a certain set of features, a system for finding similar text can be used to look in other clips for text that M can read.

In [27] we presented a method for detecting horizontal, uniform-colored text. Its invariance to color and size make it able to detect a wide variety of text in video, including fine-print, captions, and horizontal scene-text. Features like text color and size were extracted during the detection process.

In this chapter we present the first algorithm that would find text in a query frame and look for similar text in a database of video frames. It uses our text detection method [27] to find the bounding boxes and extract color features. During detection, very third line of pixels is examined and every text block that it passes through are bounded by a box. We have observed that after detection, a text block is highlighted by several overlapping boxes. This is caused by
multiple lines passing through the text and the multiple colors that compose the character strokes (Figure 4.2). The matching algorithm works by matching overlapping sets of boxes in the query frame with overlapping sets of boxes in a frame in the database. Within the matching of the two sets of boxes, $L^*a^*b^*$ color features are used in comparing pairs of boxes.

4.2 Previous Work

The literature on text in video have envisioned systems that would find text and read the characters using an OCR. Success in binarizing and reading characters [19] has been limited to text constrained in size, stroke color, and background. Problems that make this goal elusive include noise, character degradation due to encoding (ex. MPEG), variety of text fonts and sizes, unknown character color, and changing background.

In this paper we treat text as a video object whose features can used to compare with other text objects. Text has a wide variety in appearance, and our aim is to put as few constraints to its features as possible.

Text detection algorithms look for features such as texture [42], edges [1, 40], and color [16, 17, 27].

In [27] we introduced a method to detect text whose constraints are horizontal alignment and uniform color. Color features and size features are extracted in the detection process. It was shown to be invariant to size, color and background.

4.3 Matching Text Regions

First of all, we briefly describe our earlier text-detection method (referred to as TextD) and explain how the color features are extracted. More details are found in [27].
TextD examines every third row of pixels and determines whether it passes through a text object (Figure 4.1, top). Given a row R, its pixels are clustered in \( L^*a^*b^* \) space. Each color cluster is examined and the pixels close to R and within the cluster’s range of values are marked (Figure 4.1, bottom). The bounding rows and the bounding box are computed from the marked pixels. A computed bounding box comes from the examination of a color cluster, thus the center of this cluster is assigned to the box as its color feature.

![Detection of a text block](image)

Fig. 4.1. Detection of a text block. The intersection of row R with the character strokes (dark streaks) are detected as one color cluster during the clustering of row R’s pixels. A search in R’s neighborhood for pixels within the cluster’s range allows the computation of the bounding rows and the bounding box. The cluster’s centroid becomes the box’s color feature.

The size of a text block determines the number of rows that pass through (Figure 4.2, top) because every third row of pixels is examined. For example, a 6-pixel-high fine print will have two rows passing through whereas much larger text will have more. This causes multiple overlapping boxes computed on a text object.
Another cause of multiple boxes is the multiple colors that compose the character strokes (Figure 4.2, bottom). A clear example would be a typical light caption text with a dark shadow. These two colors (two clusters) would cause two boxes for a given row. For most text objects, the clustering algorithm typically produces two to four colors which would result in overlapping boxes.

Fig. 4.2. Multiple boxes. Multiple rows passing through a text object and multiple colors composing the character strokes result in detection of multiple overlapping boxes.

With these observations, we represent a text object as a region covered by a set of overlapping boxes, each with a color feature. This set is matched against other sets of overlapping boxes in the video frame database.

The algorithm looks for similar text regions by finding matches between two sets of overlapping boxes. Similarities between pairs of boxes are computed using the distance between \( L^*a^*b^* \) color vectors.
We denote the text boxes detected by TextD in frame A and B as

\[ A_i, i = 1, \ldots, N_a \]
\[ B_j, j = 1, \ldots, N_b \quad \text{where} \]

\[ N_a = \text{number of boxes in frame A} \]
\[ N_b = \text{number of boxes in frame B} \]

We define a boxpair \( P_{ij} \) as a pair of boxes in frame A and frame B

\[ P_{ij} = (A_i, B_j) \]

and their similarity denoted by

\[ \text{Sim}(P_{ij}) = \left| \text{Color}(A_i) - \text{Color}(B_j) \right| \]

where \( \text{Color(textbox)} \) is the \( L^*a^*b^* \) color vector feature of textbox when it was computed by TextD.

For all boxpairs, their similarities are tested against a threshold \( SimT \). A boxpair \( P_{ij} \) is rejected if \( Sim(P_{ij}) > SimT \). The remaining boxpairs are considered for the next step, where the overlapping of boxpairs is tested. Two boxpairs that overlap is defined by

\[ P_{ij} \cap P_{xy} \neq \emptyset \iff A_i \cap A_x \neq \emptyset \land B_j \cap B_y \neq \emptyset \]
A match $M^v_k$ is a set of $P_{ij}$’s, defined by

$$M^v_k = \{ P_{ij} \}, \bigcap_{P_{ij} \in M^v_k} P_{ij} \neq \emptyset$$

where $v$ is the number of boxpairs $P_{ij}$ in $M^v_k$, (i.e. $v = |M^v_k|$) and $k$ is simply an index into the list of all computed matches. The superscript $v$ denotes the overlap level for a match. In simple terms, a match is a set of boxpairs, with the condition that all the boxpairs overlap.

The algorithm starts by computing the matches at overlap level equal to 2, and proceeds to higher levels. When all the matches at all levels are computed, some at the lower levels are subsets of higher-level matches. The subsets are removed, since they are considered “redundancies” of the higher-level supersets. This means that from the list of matches, $M^u_{k_1}$ is removed if there exists another match $M^v_{k_2}$ such that $M^u_{k_1} \subset M^v_{k_2}$ (it also follows that $u < v$).

Finally, a parameter $MinOverlap$ determines the minimum overlap level in order to accept a match. That is, a match $M^v_k$ is rejected if $v < MinOverlap$. Clearly, we favor matches with higher overlap levels.

The result is a list of matches where each match highlights a region in frame A and its matching text region in frame B. A text region is presented as the union of boxes within the match.

The intuition behind the matching algorithm is simple: If two text objects are similar, then we can find a mapping between the two sets of boxes (whose color feature is used for similarity computation). The boxpair is the basic element of this mapping. It is necessary to constrain possible mappings to overlapping boxpairs because we have modelled detected text
objects as overlapping boxes. The overlapping of boxpairs can also be thought of as a “voting” of pairs of similar regions. Higher overlap levels mean a higher vote for a particular pair of regions.

4.4 Experiments and Preliminary Results

The method is tested on a database of 200 key frames from a diverse collection of video clips. The clips are in MPEG-1 format (4.15 Mbps) with 320 x 240 frame sizes. Each key frame is run through the TextD algorithm and the set of detected boxes are stored as an index.

The input is a set of 10 query frames. Each frame is run through TextD and the output is compared with the index of every frame in the database. Whenever the algorithm finds two similar text regions, this pair of regions are displayed separately (as in Figure 4.3).

We define a correct match as two similar text regions (in query frame and database frame) that are correctly matched by the algorithm. False matches include two dissimilar but matched regions, and matches which include false text detections by TextD. For a database frame, a misdetection is a text object with a similar text object in the query frame but is not found by the matching algorithm.

There are a few frames in the database that are shot within the same broadcast program as the query frames, and a successful matching algorithm should find the matches within these frames. The matching algorithm was able to find 68% of the correct matches in these frames. Examples are shown in Figure 4.3. There were also false matches, occurring in 31% of the database frames. Most of these false matches are text regions whose character stroke colors are similar to those in the query frames but the appearance (stroke width, size) are different. The 32% misdetection rate includes text that are too small or too faint to be detected.
4.5 Discussion and Future Work

In this chapter we have shown the possibility of using text as a video object whose visual features can be used to find other similar text objects. The detection and matching of these objects were done automatically. If fully developed, this can be used in content-based video retrieval systems for finding semantic links between video segments. Although success has been limited to a reasonable recall (correct matches), we believe that this can be improved. For some applications like finding similar text captions, spatial and size constraints can be used to filter the matches generated by a matching algorithm. We believe that the features extracted by other text-detection methods (texture, edge or color features) can also be used to create methods for finding similar text.
Fig. 4.3. Some of the detected matches. The query frame is on the left side and the database frame is on the right. Each text region is the union of a set of overlapping boxes.
Chapter 5

Detailed Vehicle Recognition

In this chapter, we explore two strategies for recognizing vehicle images using visual features. The first is by detecting the presence of text on the vehicle body. The second strategy takes the whole vehicle image and uses a multi-object alignment strategy to determine if a particular vehicle was seen before.

5.1 Detection of Text Marks on Moving Vehicles

The need for increased security has spurred a great deal of research to develop intelligent tools for automatic analysis of video. Advances in scene classification, object detection, object classification and activity recognition have proved the eventual feasibility of video-based security systems. Nowadays, inexpensive security cameras can be easily mounted above a street scene to observe activity. A particular object of interest is the vehicle. Vision systems have been envisioned to constantly monitor traffic and observe passing vehicles, extracting important features such as vehicle type, color and distinct marks. Text is a distinct mark that can be found in many vehicles, especially commercial vehicles. We present an approach for finding text marks on vehicles in a large street scene as captured by a video camera. It uses a typical wide view of a street scene (Figure 5.1) where vehicles can be observed in full length as well as moving people. Our plan to later use this method as a part of a larger system that creates an index and identifies vehicles by their type, color and distinct marks. In a surveillance application, this could be
used for recognizing suspicious activity. An example of suspicious activity is when a certain car repeatedly passes in front of a building (e.g. embassy), or a “Ryder” truck that unusually slows down in front of a federal building.

Another application is in law enforcement, where police can mount a system for finding vehicles of a certain description. One such example is when authorities in the Washington, D.C. area were looking for a box truck that is linked to a sniper. This truck is described as having text marks on its side (Figure 5.3). The system’s ability to find text marks would be useful in this case.

Finding text in video has been a challenging problem. Factors like poor resolution, noise and compression artifacts degrade the video images. In the imaged scene, text can have different fonts, size, colors and background. The most common constraint among previous text detection methods is the assumption that text lines are oriented horizontally. Limited success has been achieved without this assumption.

On vehicles, text marks are very small compared to the imaged scene (Figure 5.1) and are typically aligned with the vehicle’s body or its direction of travel. Motion blur causes character edges to degrade, although the transition between character pixels and background are still visible (Figure 5.2).

In [27], we have developed a method for detecting text where the only assumptions are that it is horizontally aligned and the stroke pixels have uniform color. It was proven to be invariant to scale and performed well on caption text and horizontal scene text. We modified this method to increase detection and make it more robust to small off-horizontal angles and greater color variation. To reduce the false alarm caused by the increased detection, a binary image test is added which favors detections having text-like properties. The method is applied to the
vehicle image. This image of the vehicle is obtained by segmenting moving objects from the scene and rotating the object image such that the vehicle body (and the text marks, if any) are oriented close to horizontal.

5.1.1 Related Work

The problem of detecting text in video has been approached using three kinds of methods: edge-based, texture-based, and color-based methods. Edge-based methods [1, 24] use the observation that text has strong edges between the character and background pixels. Texture-based methods [23, 42] rely on the texture feature uniformity across the text object.

Color-based methods [17, 16, 27] assume that the text pixels are similar in color or intensity. The three cited methods particularly have the advantage of making no assumption about the color and intensity of the text pixels, whereas others assume a white or bright text foreground color. Color features are first extracted and spatial rules are applied to find the spatial boundaries. The major difference between the three methods is the amount of information used in extracting the color features. Jain and Yu [17] used the information in the whole image in obtaining a multivalued image decomposition. Connected components is used to determine the spatial boundary. This method would work well if either the text occupies a large portion of the image or if the image has a few distinct colors, such as a magazine. In contrast, Gargi et al [16] examined very small features – small segments of the same color are fused together and grown to become text regions. The work of Mariano and Kasturi [27] strikes a balance between the two by examining a single row of pixels at a time. For every third row of the image, color clusters are computed, and a local search is used to find text boundaries in the row’s vicinity. This method works well for horizontal, uniform-color text such as captions.
5.1.2 Detection of Vehicle Text

The scene is a typical wide view of a street where both vehicles and people are observed. A video camera is mounted overhead in such a way that the sides of passing vehicles are visible and rarely occluded.

5.1.2.1 Vehicle Segmentation and Orientation

The first step is to extract the vehicle image and normalize its orientation. Given an image sequence, moving objects are extracted using frame differencing followed by a series of morphological operations. The vehicle image is then rotated such that it is as close to being horizontal as possible. The assumption is that most text found on vehicles are aligned horizontally with the vehicle’s body. The angle of rotation can be computed using the angle of the vehicle’s parallel edges (as in [37]), the direction of travel, or prior knowledge of the scene. For our experiments, we use the angle of the vehicle’s parallel edges.

5.1.2.2 Color Feature Extraction

Text is detected using the method in [27] which is briefly described below. Section 5.1.2.3 describes how the method is modified. The basic assumption is that the pixels within the characters’ strokes are similar in color, regardless of the background. The rectified vehicle image is scanned every second line of pixels. We use an ellipsoid in the $L^* a^* b^*$ color space to model the color variation of stroke pixels. A maximum distance (between ellipsoid points and the center) limits the size of each ellipsoid. Candidate ellipsoids are computed from the clusters resulting from a hierarchical clustering of row pixels. For each candidate ellipsoid, a local spatial search
finds other pixels that fall in the ellipsoids range of colors, and heuristics are defined to test
whether the pixels form a text-like structure.

The algorithm visits every row in the extracted vehicle image. Given a row $R$ on the
image, we want to find out whether or not $R$ passes through the middle of a text region.

The pixels of $R$ are transformed and clustered in the perceptually uniform $L^*a^*b^*$ color
space using hierarchical clustering and the weighted Euclidean norm as the distance function.
The weighted norm was used to achieve a slight invariance to lightness (weights: $L^* = 0.8, a^* =
1.1, b^* = 1.1$). Later, this weighted norm will be used to test membership into the formed
clusters. Ellipsoids in color space are defined as a result.

Each cluster $C$ is tested to see if it contains pixels belonging to text. Locating the bounding
rows (top and bottom rows of text) is the first step (Fig. 5.4). The cluster points are marked
back on row $R$ to create streaks $S_i, i = 1 \ldots N_s$ (where $N_s =$ number of streaks) of pixels in the
row $R$. Then pixels above and below $R$ are examined and each pixel that falls within the ellipsoid
of cluster $C$ are colored with a value of $T$. All other pixels are marked $T'$.

We now try to find out if there are bounding rows above and below $R$ which may contain
horizontal text. Given a pair of adjacent streaks $S_i$ and $S_{i+1}$, we find $R_a$ – the first row above
$R$ in which the segment covering $S_i$ and $S_{i+1}$ is colored $T'$. We also find $R_b$ – the first row
below $R$ in which the segment covering under $S_i$ and $S_{i+1}$ is colored $T'$. The $R_a$ of each pair
of adjacent streaks is computed and collected in an alignment histogram $H_a$, where the bins are
the rows of the image. $H_b$ is computed in the same way by taking all the $R_b$’s. We declare the
existence of a bounding row $B_a$ if at least 60% of the elements in $H_a$ are contained in a window
of three or fewer adjacent histogram bins. $B_b$’s existence is computed in the same way from $H_b$.
If $B_a$ and $B_b$ exists, $height$ is defined as their difference.
If the cluster \( C \) contains text pixels, then \( B_a \) and \( B_b \) would mark the text block’s upper and lower row boundaries, and \( \text{height} \) would define its vertical dimension. Figure 5.4 illustrates the computation of \( B_a, B_b \) and \( \text{height} \).

We look for text blocks using heuristics on \( \text{height} \) and the short streaks’ lengths and spacings. Streaks longer than \( \text{height} \) are discarded. Spacings that are longer than \( \text{height} \) are considered not part of a text block. The remaining regions are now smaller blocks with short streaks. If a block’s width is greater than \( 1.5 \times \text{height} \) and the number of short streaks inside is greater than 3, then it is considered a text block, otherwise it is discarded. Finally, the text block is expanded a few pixels to the left and right to ensure full coverage of the characters at the ends.

5.1.2.3 Detection Increase and Binary Image Test

The detection algorithm assigns the boundaries of a text box based on an extracted color, a local spatial search above and below the analyzed row \( R \), and the spacings between the streaks. This has worked quite well in previous work on artificially-placed caption text [27] where there is little variation in color across text pixels. Furthermore, caption text is almost perfectly horizontal, which benefits localization. On vehicle text, however, conditions are less favorable. Even if the vehicle image is rotated to be horizontal, the text can still be off-horizontal by a few degrees. On the vehicle surface, the colors of the text pixels still appear similar but there is greater variation compared to caption text. The reason is that this is on an object surface that is imaged outdoors.

To make the detection more robust to these problems, the parameters of the detection algorithm are relaxed in order to increase detection. To relax the horizontal constraint, we increase the number of adjacent bins in the alignment histograms which are used to find the bounding
rows $B_a$ and $B_b$. To account for the greater variation in text pixel color, the maximum distance (between cluster elements and cluster center) is increased in the clustering algorithm.

These modifications would result in better recall but would also increase the number of false alarm. A binary image test is used to reduce the false alarm by distinguishing between boxes that are likely caused by text and those that are not. The binary image of each text box is computed by testing each pixel whether it is inside the color ellipsoid which was computed in the clustering step. Figure 5.5 shows binary images of boxes caused by the “Balfurd” text in Figure 5.7. Consider any of these binary images. If we observe most of the pixel rows, there is repeated transition between consecutive zeros and consecutive ones as you go from left to right. This is caused by the characters’ vertical and diagonal strokes. Now comparing these to the binary image of a typical false alarm in Figure 5.6, this is not the case.

The binary test examines each row of the text box’s binary image and counts the number of segments (countiguous groups of ones). If enough of the rows have segments greater than a threshold then the text box is accepted. The threshold should be dynamic because the number of segments is directly related to the number of characters covered by the box. Since the number of characters is not known, we use as an alternative the text box’s aspect ratio (width/height) multiplied by a factor to compute the threshold. We define the test as follows: Let

$$\text{MedSeg} = \text{Median}(\text{NumSeg}[\])$$

$$\text{NumSeg}[\] = [\text{NumSeg}_1, \text{NumSeg}_2, \ldots, \text{NumSeg}_m]$$
where $\text{NumSeg}_j$ is the number of segments across row $j$ of the binary image and $m$ is the height of the binary image. A segment is defined as a group of row-contiguous pixels with binary value equal to 1.

The text box is accepted if its binary image satisfies:

$$MedSeg > \left(\frac{\text{Width}_{\text{binary}}}{\text{Height}_{\text{binary}}}\right) \times \text{RatioFactor}$$

This is a comparison of $MedSeg$ with a dynamic threshold. It is a function of the box’s aspect ratio and controlled by a fixed $\text{RatioFactor}$ parameter.

Finally, all the detected boxes that passed the binary test are fused together to form the text regions.

In summary, the detection algorithm extracts multiple hypotheses on the color of the text, and applies rules to test each hypothesis and find the spatial boundaries. The ability to generate good hypotheses makes it possible to create the binary image, have the binary test and weed out false alarm. This ability sets it apart from the other color-based detection methods in the literature and obviously from edge-based and texture-based methods.

### 5.1.2.4 Tests on Image Sequences

The method was tested on a 20-minute video of a street scene with pedestrians and vehicles. Of the 285 vehicles that appeared on the scene, 24 had text that is large enough to be read of which 23 had their text marks detected. Figure 5.8 shows an example of a detected text mark on a segmented vehicle. Of the vehicle which didn’t have text in them, there were 33 false alarm.
5.1.3 Conclusion and Future Direction

We have presented a method for detecting text marks on vehicles using color features. A modified existing method was applied to extracted vehicle images and a binary image test is used to reduce the false alarm.

When a text mark is covered by multiple detected boxes, the binary images (like those in Figure 5.5) seem to suggest that the hypothesized colors are accurate even if we are not sure which ones are foreground. Perhaps a method can be formulated for grouping and fusing these images in order to create a binary which can be passed to an OCR for recognition.

5.2 Recognition of Recurrent Vehicles

Vehicles are one of the most common objects found outdoors. They are used to go from place to place and transport all kinds of cargo. Human activity directly affects their occurrence and behavior. Part of studying and monitoring human activity is recognizing the objects that we commonly use and interact with.

Each vehicle is unique. In order to distinguish each one, a license plate is attached to the front or the rear. Like a person’s nametag, a license plate gives a vehicle its identity. Visual features can also give a vehicle its unique identity. A good example would be finding our own car in a parking lot when we forgot where he parked it. Our first visual cues are typically shape and color, then if a match is found we look for more smaller features like stickers and dents. And finally to make sure, we look at the license plate to verify that it is my car.

Throughout the life of a vehicle, its visual identity gets established. The owner can attach new wheels, put membership stickers like AAA, repaint the body, and the car may develop rust
in lower areas. The acquired features enables us to distinguish vehicles even of the same make, model and color. As a human can distinguish vehicles by visual features, we seek to understand how we can make a computer do it using computer vision.

The application of interest is a system for monitoring a limited view of a street scene. The goal is to observe each passing vehicle and determine whether the currently observed vehicle has appeared before in the same scene. In the light of recent heightened security in major buildings like airports, embassies and other federal buildings, this application can be used to observe vehicles and detect any suspicious activity. We explore a computer vision strategy towards realising this application. The challenges presented by this problem includes video noise, low video resolution, object specularity plus the variability in imaging outdoor environments.

5.2.1 Previous Work on Vehicle Features and Recognition

In the literature, there has been significant work on extracting vehicle features. Work has been focused on extracting the general shape and classification into the major vehicle types (sedan, truck, wagon, etc.). Color features are also extracted for vehicle matching. Identification of vehicles using body features were limited to recognition of license plates [7, 6].

Collins, et al [5] developed a system for detecting, tracking and classifying moving blobs. A neural network is trained to to classify the moving blobs into single human, human group, vehicle, and clutter. The input features to the network are blob dispersedness, blob area, blob aspect ratio and camera zoom. Vehicles are further classified into different types (van, truck, or sedan) and color using linear discriminant analysis. For vehicle type, the feature vector consists of blob features: area, center of gravity, width, height, and the first three moments taken along the row and column pixel axes. The color feature vector has three dimensions (I1,I2,I3). They
claim that the method was also trained to recognize specific vehicles like a UPS truck and the campus police car.

Tan and Baker [37] presented algorithms for localizing and classifying vehicles (high-roof van, minibus, saloon/sedan) based on image gradients in a small window. The vehicle pose is computed using a ground-plane constraint and the fact that the appearance of most vehicles is dominated by two sets of parallel lines.

Fung, et al [15] showed a method for approximating vehicle shape using motion observed using a high video camera. The height of feature points (corners) are estimated to create a height profile. The basic idea is that feature points that are higher (thus closer to the camera) move faster than the lower feature points. The height profile can then be used later for vehicle classification.

Jolly, et. al. [18] used shape, color and edge features to match vehicles between two sites for measuring time of travel. Deformable templates were used to classify vehicles based on shape information. Side-view, 2-D deformable templates of five vehicle types (sedan, pickup, hatchback, station wagon and van) were fitted on the front vehicle’s edge image. Color and edge features were taken from their earlier work on vehicle matching [12]. 3-D, RGB histograms of the vehicles were compared using histogram intersection. The set of edge points within the fitted template were compared to those of other vehicles using a modified Hausdorff distance between point sets.

Zeng and Crisman [32] also used RGB histograms to match vehicles between two sites. Color information was used to track vehicles from one camera site to another to measure the time it takes to travel from point to point. The histograms were modified to compensate for differences in illumination between the two sites. Statistics based on bin-to-bin differences were used to compute the similarity between vehicles.
The literature on vehicle feature extraction (excluding license plate reading) works on general features like shape and color to classify and compare vehicle images. While these may be effective for reducing the search space to a few similar images, it is not enough to determine if two images are that of the same object. A more detailed image comparison would be needed toward this goal.

Detailed object comparison makes sense when corresponding object parts are compared. The images of the compared objects show a subset of these parts. The object observed in the scene (Figure 5.18) are almost in the same pose and corresponding objects parts are visible as image features. The logical first step would then be to align the object images with the goal of moving the corresponding image features in the same spatial location.

In searching for the most similar objects, the natural approach would be a serial search where the query image is aligned with each candidate image. The candidate images can then be ranked according to their spatial differences with the query.

Image registration methods seek to find the set of transformations necessary to align images. Most of work on registration [4] deal with alignment of images of the same object or scene. While work on aligning multiple images has been done in shape learning [13] and super-resolution from video [22], multiple-image alignment of different objects for recognizing a query object image has not been realized.

### 5.2.2 An Alignment Approach to Recognition

We present a multiple-object alignment strategy coupled with a search-space reduction scheme utilizing learned feature correspondences. The main idea is that given a set of candidate images output from a general image-retrieval system, the images are aligned in parallel and
outlier objects are iteratively removed, making the set more homogeneous and better aligned. The strategy uses a multi-object feature clustering and alignment approach.

Figure 5.9 shows the framework for recognition. A high-recall image retrieval system utilizing general features such as color and general shape then retrieves a loose set of similar images from the database of observed vehicles. The database is thus reduced to a smaller search space.

The contribution of this work is the iterative reduction of the search space by detecting correspondences, re-alignment of the search space with the query, and removing outlier objects. Correspondences are detected not just between the query and each search space image but across all images in the search space. In effect, we are looking for correspondence clusters (Figure 5.10) where each cluster is a spatial feature common to a subset of the objects. Our hypothesis is that the outlier object will have an image with the least correspondence with the query and the rest of the search space (Figure 5.11). We also hypothesize that removing the outlier objects would allow a more precise re-alignment and allow a more accurate detection of correspondence clusters. Prior to this iterative process, the method starts with the retrieved set of images. An initial alignment step roughly aligns the image set to the query image.

The main challenge in this strategy is to define what is an “outlier” object in a set of object images. This definition is modeled using information in the correspondence clusters.

The reduction of the search space can later be used to learn “groupings” of similar objects. This could enable faster searches as more object groupings are learned.
5.2.3 Detailed Recognition of Vehicle Images

The recognition method starts with a set of images retrieved by a high-recall image-retrieval system. General features such as size, color and shape can be used to achieve high-recall. For example, the query image could be a white sedan thus the initial query could be the set of “light-colored, medium-sized” vehicle images. The set of retrieved images would referred to as our initial search space.

Detailed recognition is done by aligning the images in the search space and iteratively reducing it by removing outlier objects. At each iteration, the objects in the search space becomes more homogenous and more closely aligned to each other. The alignment strategy was chosen because detailed object comparison is possible when corresponding features are moved closer spatially.

Corresponding features are learned using a clustering algorithm that works spatially and across multiple images. A cluster could thus contain multiple features from multiple objects. The learned clusters are centered around the prototypes (or cluster means) which serve as the reference points toward which the cluster elements gravitate in the alignment step. The properties of the clusters are used to determine the outlier-ness of a each object.

We first define how alignment is performed. Given a set of source points

\[ [(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)] \]

and a set of corresponding reference points
the goal is to align each point \([x, y]\) with its corresponding \([x', y']\) using the following transformation:

\[
\begin{bmatrix}
x'
y'
\end{bmatrix} =
\begin{bmatrix}
S_x & 0 \\
0 & S_y
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix} +
\begin{bmatrix}
T_x \\
T_y
\end{bmatrix}
\]

where \(T_x\) and \(T_y\) are parameters for translation in the \(x\) and \(y\) directions, and \(S_x\) and \(S_y\) are scaling factors in the \(x\) and \(y\) directions. Instead of one factor, scaling was separated into \(S_x\) and \(S_y\) because the aspect ratio of the vehicle image varies with the vehicle’s lateral position on the road. Our chosen view of the scene is above and beside the road which made the scaling separation necessary. For some views, a single scaling factor would suffice.

A set of source points are aligned to a set of reference points by estimating the transformation parameters defining the alignment. Each source-reference pair of points represent an equation. The \(N\) equations are collected in the following matrix equation:
where $R$ is the reference and $M$ is the source. $P$ is the transformation parameter vector which can be estimated using Least Squares:

$$P = (M^tM)^{-1}M^tR$$

If the source points are transformed using the estimated $P$ vector, the source points may not be perfectly aligned with the reference points. The alignment error measures this misalignment:

$$AlignmentError = \frac{1}{N} \sum_{i=1}^{N} \sqrt{(x_i^t - x_i')^2 + (y_i^t - y_i')^2}$$
where \((x_i^t, y_i^t)\) is the transformed source point and \((x_i', y_i')\) is the corresponding reference point.

5.2.3.1 Feature Extraction and Initial Alignment

Image gradients are stable image features. Under varying illumination conditions, the image gradients remain significant. In the case of a vehicle, the structural edges produce image gradients useful for aligning and differentiating objects. We used a Sobel edge detector and a gradient magnitude threshold that sufficiently highlights the structural edges of the object.

Given the retrieved set of images, edge features are detected to produce a point set for each image. Each point set is roughly aligned to the query’s point set prior to the iterative search space reduction. Nine points are computed from the spatial distribution of the edges. The three horizontal coordinates are the 25th, 50th, and 75th percentile in the X-coordinate spatial distribution. The three vertical coordinates are also computed as the 25th, 50th, and 75th percentile of the y-coordinates. Figure 5.12 shows the nine points for an object image.

For each image \(M\) in the search space (the retrieved image set), the transformation parameters are estimated by aligning the nine source points of \(M\) with the nine target points of the query image. The estimated parameters are then used to transform the edge image of \(M\), thus roughly aligning it with the query edge image.

5.2.3.2 Detection of Correspondence Clusters

Once the objects are initially aligned with the query, the iterative loop starts. The purpose of the initial alignment is to bring the objects’ corresponding features close enough to start the convergence of multi-object alignment. When sufficiently aligned, features common to some
objects lie in close spatial proximity. When the objects are superimposed, the common features appear as clusters in image coordinates.

A K-means clustering algorithm is employed to detect features common to objects, including the query. The coordinates of the edge features of all the object images are gathered in one list and the $K$ cluster prototypes are allowed to gravitate toward significant edge clusters. The number of cluster prototypes $K$ is made sufficiently large to detect major features as well as small and isolated features. Each cluster prototype is computed as the mean $(x, y)$ coordinate of the edge features belonging to that cluster.

5.2.3.3 Re-alignment of Object Features and Cluster Prototype Adjustment

The purpose of re-alignment is to bring corresponding features, which are detected by the clustering algorithm, closer together. With each iteration, the goal is to more accurately detect feature correspondences and make the remaining objects better aligned.

Upon convergence of the $K$ cluster prototypes’ locations, each object is aligned with the cluster prototypes. For a given object, the edge features are allowed to gravitate toward their respective cluster prototypes. The transformation parameters are estimated using the edge feature coordinates as the source points and the cluster prototype coordinates as the reference points.

The transformation of the objects re-positions the elements of the clusters and renders the prototype locations invalid. Thus the cluster prototypes need to be adjusted to the new feature locations. This is done by allowing the K-means algorithm to run a few more iterations to allow the prototypes to converge.
5.2.3.4 Search Space Reduction by Outlier Removal

The initial retrieved set of images is reduced iteratively by removing outlier objects. We define an outlier as the object with the least correspondence with the query object. Each cluster is observed and the amount of correspondence of a candidate with the query is computed and summed over all clusters.

Let $C_{ji}$ be the set of feature points belonging to object $j$ and classified into cluster $i$. An object’s score is computed as:

$$\text{ObjScore}(D_j) = \sum_{i=1}^{K} \text{ClusterScore}(C_{ji})$$

where $\text{ClusterScore}(C_{ji})$ is the amount of correspondence, within cluster $i$, between object $j$ and the query object. This is defined as:

$$\text{ClusterScore}(C_{ji}) = \min \left( \frac{|C_{ji}|}{N_j}, \frac{|C_{qi}|}{N_q} \right)$$

where $N_j$ and $N_q$ are the number of feature points in object $j$ and the query, respectively.

The $\min()$ operation acts as an intersection of the two objects within a certain spatial locality. A high $\min()$ means there is strong correspondence in that cluster’s location. In each iteration, the object with the minimum $\text{ObjScore}()$ is declared as outlier and removed from the search space.
5.2.4 An Experiment

An experiment was performed to test the iterative reduction scheme. A 15-minute video of a street scene was captured on a sunny day. To create a vehicle recurrence event, a friend is asked to drive his white sedan three times through the street scene in four-minutes intervals. Prior to the third drive-through, four black stickers are attached to the roof and the shaded side of the car. Figure 5.13 shows the white sedan in the three drive-throughs. Using the second occurrence of the white sedan as the query object, the other two occurrences should be among the last ones to be removed from the search space.

The output of an image-retrieval system is simulated by manually picking out all the light-colored vehicles that crossed within the 10-minute interval. Nineteen vehicles were picked, including the first and the third drive-through of the white sedan. The second drive-through is the query object. The image of the white sedan is about 400x200 pixels.

Figure 5.18 shows the history of outlier object removal. Starting with nineteen vehicles, the search space is iteratively reduced to one car. The first drive-through of our white sedan was the last object in the search space, with the black-stickered sedan coming in second to the last. The removal sequence starts with vehicles that are least similar, and ending with the most similar.

Figure 5.14 shows the $ObjScore()$ (defined in Section 5.2.3.4) of the removed object at each iteration. The trend shows an increase in the amount of per-cluster correspondence as the search space is reduced. With each iteration, the trend shows the search space’s increase in similarity with the query. Intra-cluster deviation, defined as the standard deviation of all feature point distances from their respective cluster prototypes, is used to measure the variation...
of features among the search space objects. Figure 5.15 shows a general decrease of the intra-cluster deviation as the search space is reduced. The decrease of feature variation around the clusters may be attributed to fewer, more similar, and aligned set of objects.

A good alignment scheme allows a more accurate detection of common feature because corresponding features are brought closer to each other. To characterize the performance of our alignment method, we define the average alignment error as the per-object average distance of the feature points from their respective cluster prototypes. This is computed after the re-alignment of objects’ feature points. Figure 5.16 shows the decrease of this alignment error as the search space becomes smaller. For each iteration, the maximum alignment error among all objects is shown in Figure 5.17. Both graphs show continuous improvement of the alignment of objects as the search space is reduced.

5.2.5 Conclusion and Possible Extensions

The feature clustering and alignment framework worked well for recognizing vehicles recurring in an outdoor street scene. Our simple experiment proved the feasibility of recognizing particular objects even among other very similar objects. The framework can be useful in outdoor surveillance applications like suspicious activity detection and example-based queries.

The feature points used are produced by an edge detector. Ideally, only a few salient points are needed but selecting them under unpredictable outdoor conditions would be a challenge. 2-D and 3-D object models (like those used in [37] and [18]) can use known object model to guide the detection of salient features. Features need not be limited to points but can be extended to higher features like lines, islands and polygons. The challenge will be to define the
clustering criteria and distance measures. Our work is also limited to transformations of scaling and translation. This can be extended to 3-D transformations since the objects are in 3-D.
Fig. 5.1. A typical street scene with vehicles and moving people. Our goal is to find any text marks on passing vehicles.
Fig. 5.2. A magnified view of the red text mark of the bus in Figure 5.1.
Fig. 5.3. A box truck with text marks. Washington, D.C. police gave this image as a description of the vehicle linked to the area sniper.
Fig. 5.4. Computing the bounding rows. One of the color clusters in row $R$ are marked as short streaks and pixels of the text “Ryder” lie within the ellipsoid of the cluster. The spacings between the streaks are used to compute the left and right boundaries of the text bounding box.
Fig. 5.5. Binary images generated by color ellipsoid membership. These overlapping text boxes are detected on the “Ballurd” text in Figure 5.7. For each text box, a single color cluster from the pixels of a single row is used to compute an ellipsoid in color space and spatially find the bounding box. The color ellipsoid is then used to generate the binary image (black = 1).

Fig. 5.6. Binary image of a false alarm. In this example, the analyzed row passed through the vehicle windows whose sides were mistaken for character strokes.
Fig. 5.7. Most text are parallel to the vehicle body. The moving vehicle is extracted from the scene and oriented horizontally before applying the text detection algorithm. The angle of rotation can be derived using the angle of the vehicle’s parallel edges, the direction of travel, or prior knowledge of the scene. The “Balfurd” image shows what the text looks like after rotation.
Fig. 5.8. The test street scene, a moving truck segmented from the scene, and the detected text mark.
Fig. 5.9. Overview of vehicle recognition system

Fig. 5.10. Detection and alignment of correspondence clusters. Common spatial features are detected in the query object image and search space object images. The common features form a cluster, with a cluster prototype as the center. During re-alignment, the object image features are driven toward the cluster prototype.
Fig. 5.11. Analogy of “outlier objects”. Points in this space represent objects in the reduced search space, clustered around the query. Outliers are the objects with the least correspondence with the query. Removing the outliers results in a smaller, more homogenous and better aligned search space.
Fig. 5.12. Nine points for initial alignment. The three horizontal coordinates are the 25th, 50th, and 75th percentile in the X-coordinate spatial distribution. The three vertical coordinates are computed in the same way.
Fig. 5.13. Three passes of the same car. A friend is asked to drive the same car through the imaged street scene. The second drive is used as the query. On the third drive, four black stickers are attached on the roof and side.
Fig. 5.14. Outlier score vs. Iteration. In each iteration, the score of the removed object is recorded. The trend shows increasing correspondence with the query.
Fig. 5.15. Intra-cluster standard deviation vs. Iteration. The shrinking clusters indicates smaller feature variation among objects.
Fig. 5.16. Average object alignment error vs. Iteration. Average error decreases, indicating closer alignment of search search.
Fig. 5.17. Maximum object alignment error vs. Iteration. At each iteration, the maximum error among objects is recorded. The decreasing error shows closer alignment of objects.
Fig. 5.18. Outlier removal history. For comparison, the query is shown in the last image. The two occurrences of the same white sedan were the last in the history. The white sedan with the black sticker was removed before the one without the sticker.
Chapter 6

Summary

6.1 Challenges and Future Directions

Detection of unconstrained text in video is still a very open problem. Though there has been moderate success in the literature for detecting horizontal and good contrast text, their assumptions do not hold true for most scene text. The least that the current algorithms are able to do is to identify candidate text regions where further analysis can be made. For some applications such as interactive video browsing, the candidate regions may be enough. If the ultimate goal is to recognize the characters, the detection algorithm would need to get feedback from a character recognition module in order to refine the text regions. Thus we find that the classic paradox, “To recognize an object, you need to find it. To find the object, you need to recognize it,” would certainly apply to text as well as to other object classes. The problem of the large variation in text size, color and font suggests that multiple detectors are needed and the results intelligently combined. Matching text objects, like the method described in this thesis, would be useful for finding similar text in a large video database that a trained recognizer could read with high accuracy. Our method used only color features to find matches, but other features and their combinations can be explored for certain types of text.

When we first started with performance evaluation of object detection methods, it appeared to be a trivial task since there were a few papers in the literature. Each research group had its own method of evaluating their algorithms and there are few public data available. It
turned out that performance evaluation is not simple at all. Aside from the usual issue of quality of training and testing data, there is no single measure that quantifies how well an algorithm performs. The six measures presented in this thesis, though able to capture different performance aspects, are still too many when it comes to summarizing performance, algorithm comparison and especially algorithm optimization. The challenge is to come up with a model for combining these performance measures into a much smaller set. The model would have to be tweaked for different applications which have different performance evaluation requirements.

Recognizing vehicles, whether simply to determine its type or recognize its distinguishing features, is very challenging because of unpredictable outdoor imaging conditions and the fact that the object is 3-D. The aligning transformations in this thesis, namely translation and scaling, are not enough to account for the range of object pose variations. An initial estimation of the 3-D object shape could allow additional 3-D transformation. Fitting a 3-D model would also allow more robust detection of features where edge detection does not find object features.

6.2 Contributions and Publications

The contributions of this thesis are the following:

- A new method for detecting text in video.
- A new method for comparing text objects using visual features.
- A set of six new measures for performance evaluation of object-detection algorithms.
- An exploration of methods for recognizing vehicles, first by detecting text marks and second by detailed recognition through image alignment.
6.3 Publications

The work in this thesis and results have appeared in the following publications. Also included are publications during the author’s time as a student at Penn State.


References


Vita