9-6-2005

Object detection and tracking using a parts-based approach

Daniel S. Clark

Object Detection and Tracking using a Parts-Based Approach

Daniel S. Clark

September 6, 2005

Object Detection and Tracking using a Parts-Based Approach

Daniel S. Clark

Roger Gaborski 9/27/2005

Dr. Roger Gaborski, Chair

Roxanne Canosa 9/28/2005

Dr. Roxanne Canosa, Reader

Carl Reynolds 9/27/2005

Dr. Carl Reynolds, Observer

Thesis Author Permission Form

Title of thesis	Object Detection and Tracking using a Parts-Based Approach
Name of author	Daniel S. Clark
Degree	Master of Science
Program	Computer Science
College	Golisano College of Computing and Information Sciences

I understand that I must submit a print copy of my thesis or dissertation to the RIT Archives, per current RIT guidelines for the completion of my degree. I hereby grant to the Rochester Institute of Technology and its agents the non-exclusive license to archive and make accessible my thesis or dissertation in whole or in part in all forms of media in perpetuity. I retain all other ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Print Reproduction Permission Granted

I, Daniel S. Clark, hereby **grant permission** to the Rochester Institute of Technology to reproduce my print thesis or dissertation in whole or in part. Any reproduction will not be for commercial use or profit.

Signature of Author: Daniel S. Clark

Inclusion in the RIT Digital Media Library Electronic Thesis & Dissertation (ETD) Archive

I, Daniel S. Clark, additionally grant to the Rochester Institute of Technology Digital Medial Library (RIT DML) the non-exclusive license to archive and provide electronic access to my thesis or dissertation in whole or in part in all forms of media in perpetuity.

I understand that my work, in addition to its bibliographic record and abstract, will be available to the world-wide community of scholars and researchers through the RIT DML. I retain all other ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation. I am aware that the Rochester Institute of Technology does not require registration of copyright for ETDs.

I hereby certify that, if appropriate, I have obtained and attached written permission statements from the owners of each third party copyrighted matter to be included in my thesis or dissertation. I certify that the version I submitted is the same as that approved by my committee.

Signature of Author: Daniel S. Clark

Contents

List of Figures

List of Tables

Acknowledgements

I'd like to thank my advisor, Dr. Gaborski, for helping me learn about research and guiding me through this thesis; all the members of my committee for their input throughout the writing of this thesis; and Tim Lebo, Gavin Page and Dr. Zack Butler for being around to bounce ideas off of.

Abstract

One of the main goals of artificial intelligence is to allow computers to understand the world around them. As humans we extract a large amount of knowledge about the world from our visual perception, and the field of computer vision is determined to give computers access to this same wealth of knowledge. One of the fundamental steps in understanding the world is finding specific objects within our field of view, and the related task of following these objects as they move.

In this thesis the Implicit Shape Model algorithm, a local feature-based object detection algorithm, is implemented and used to develop an appearance model and object tracking algorithm based on it. This algorithm is very robust to intra-class variation, and can successfully track objects when both occlusion and non-stationary backgrounds are present. The usefulness of the proposed appearance model is analyzed, and results of the algorithm on real video sequences are presented. Several enhancements to the method are also proposed, and performance in terms of recall and precision is analyzed.

Chapter 1

Introduction

One of the main goals of artificial intelligence is to allow computers to understand the world around them. As humans, we extract a large amount of knowledge about the world from our visual perception, and the field of computer vision is determined to give computers access to this same wealth of knowledge. One of the fundamental steps in understanding the world is finding specific objects within our field of view, and the related task of following these objects as they move (known as *object detection* and *object tracking* in the literature). Both object recognition and tracking are fields in and of themselves, but both are necessarily part of allowing computers to understand visual scenes.

The main challenge that object recognition algorithms have traditionally faced is the wide intra-class variation that natural classes allow. Even in a moderately restricted class such as "automobiles," there are hundreds of different models, variations on those models, and paint colors, not to mention the infinite variety of viewing angles, lighting conditions and occlusions. When you consider all this variation, it is quite natural to see why computers have such a hard time making the generalizations necessary to accurately classify objects.

In the general literature, there are several common distinctions among approaches to object recognition. One division is between top-down and bottom-up algorithms. Briefly, the bottom-up approach works by finding low-level features, and attempts to build these into objects, while the top-down approach begins by imposing an object model onto the scene, and searching for matches.

Another division among methods is between component-based (or local

feature-based) and holistic (or global feature-based) methods. In component-based methods, individual features of the object in question are searched for within a scene, and assembled to form an object if enough exist. In the holistic approach, the whole model is searched for at once. Although the descriptions of these two divisions sound similar, they are not in fact. Top-down vs. bottom-up refers to an analysis of a scene with or without an expectation of what will be found. Component vs. holistic refers to the level of granularity at which the scene is searched – fine or coarse, respectively.

Holistic methods tend to be very mathematical in their approach. Various types of wavelet transforms exist which can be used for feature extraction and pattern classification, and are invariant to scale or rotation. Brooks et al. [8] provide an excellent survey of the use of wavelets in pattern recognition. Another method currently being researched is Independent Component Analysis. A summary of ICA by Leach can by found in [15].

Component-based approaches have the advantage that they are conceptually simple to understand. If an object is present in a scene, all or most of its recognizable components parts will be present. An algorithm that can find the component parts and analyze their spatial layout will be able to recognize the object. If the problem can be constrained to a single viewpoint, it becomes almost trivial for humans, and much easier for computers.

In this work, I propose to extend the Implicit Shape Model (ISM) algorithm described by Leibe et al [18, 17]. This is a component-based viewpoint-dependent still object detection algorithm that incorporates both bottom-up and top-down processing. In this thesis, I will describe how the ISM algorithm can be used to implement an object tracker with an unique and natural appearance model. Characteristics of video data are examined in terms of the ISM method, and a tracking algorithm and appearance model are developed. Results of the tracker on stereotypical video footage is presented and analyzed.

1.1 Related Work

Object recognition has been a heavily researched topic for quite some time. Traditionally, object recognition has been in the context of static images, but more and more, researchers are looking at video data, driven both by hardware and software advances and also by world events that have made automated video security increasingly desirable. Although object recognition

is a very general area, with the security focus of many applications much of the work in recognizing objects in real scenes has attempted to recognize vehicles or humans.

Several different component-based object recognition techniques have been proposed over the last several years. Ullman et al. [28, 29] propose a probabilistic technique that uses image fragments of varying size and resolution to detect objects. Mutual information is calculated to determine the most valuable image fragments, which are then categorized according to their semantic type. Objects are detected if enough semantic types are detected in the correct spatial arrangement. The disadvantage of this technique is that human intervention is required to guide the selection of suitable image fragments. Further work by Ullman et al. in [7, 9] extend the technique to perform image segmentation using a hierarchical fragment matching approach and automatic classification of semantic types.

Agarwal et al. [2, 1] propose another component-based method, which moves away from the probabilistic method suggested by Ullman. This method detects objects by finding co-occurring image patches centered at interest points. The use of interest points reduces the computational complexity of this algorithm by only extracting image patches with high information content instead of gathering patches first, then computing information, as in [28, 29]. Objects are detected in a window that can be moved over the image, creating a binary feature vector encoding detected patches and their spatial relationships, then using a learned classifier over this vector to classify the window as object or non-object. This approach suffers computationally from moving the window over the whole image.

Leibe et al. [18, 17] present an approach similar to Agarwal's. In his ISM method, extracted patches are centered on interest points, which are related to the object center instead of other patch locations. Objects are detected by performing a Generalized Hough Transform [3] on the location of the center of the object specified by the detected image patches. This method will be discussed in further detail in Chapter 2.

Research into the human visual system has suggested that humans may use a component based method as well. Biederman's Recognition-By-Components method [6] proposes that the human visual system uses edge information to create 2D and 3D primitives (geons) which are the basic components of recognition. Further work in this area by Tarr et al. [25, 26] supports a component-based theory, but asserts that it is also highly viewpoint dependent.

The features used to describe the components of an object are one of the defining characteristics of a component-based recognition approach. Some simple approaches include various ways of using image patches or color histograms. Many different kinds of feature vectors for describing image points have been proposed, such as Lowe's SIFT measure [20, 21].

Much of the current work on object tracking requires a stable camera overlooking a relatively static scene. Hu et al. [12] provide a very good survey of current object tracking and visual surveillance techniques. In particular, background subtraction algorithms are popular due to their speed of operation, and have found many commercial applications. Many such techniques have been developed, several of which are outlined by McIvor [22]. Tracking using a stationary camera is useful for highway monitoring, security cameras, wildlife monitoring, and so on. While the constraint of a stationary camera is acceptable for many applications, a general purpose object tracker should not be limited in this way.

Methods based on optical flow offer one possibility to overcome the problem of requiring a stationary background. These algorithms have been used to track moving objects and to track the motion of an observer through a stable environment. This also suggests the possibility of tracking moving objects through environments in which the observer is also moving. Neumann [23] and Barron et a. [4] provide summaries of optical flow techniques.

Image registration to allow background subtraction. Other methods to allow moving backgrounds...

Another possibility to remove the need for a stationary background model is to use object detectors that are designed for static images. Because these algorithms do not assume a prior motion segmentation, they can be successfully applied to a series of video frames as if each frame were an independent image. The disadvantage of this approach is that it does not take advantage of any temporal information, and objects cannot be tracked, merely detected in subsequent frames.

1.2 Outline

Chapter 2 describes the ISM method proposed by Leibe. In Chapter 3, several enhancements to the static image method are proposed. Chapter 4 explains the object tracking process developed for this thesis. Chapter 5 presents the results of both static image detection on a standardized

database, and the object recognition and tracking algorithm on a series of videos collected by the author. Chapter 6 discusses interesting findings regarding the temporal stability of the local features, and in Chapter 7, alternative corner detectors are examined. Finally, possible future work is suggested in Chapter 8.

Chapter 2

Description of Implicit Shape Model

The Implicit Shape Model (ISM) method described by Leibe et. al. [18, 17] forms the basis for the method of object detection described in this thesis. ISM is a trained object detector for static images that uses distinctive components of an object and their spatial relationship to the object center to describe objects. This spatial relationship only to the center of the object is the basis of the "implicit" part of ISM. Nowhere in the training process is an object described in a holistic sense – although a side view of a car may have "a wheel to the lower-left of the center" and "a wheel to the lower-right of the center," (among other components) the ISM's model of a car does not explicitly describe cars as having two wheels. A car with only one wheel in a correct location could still be recognized as a car. Similarly, the ISM model of a human might store many different configurations for the possible positions of the arms, but would nowhere state explicitly that a human must have two arms. This would allow it to recognize humans with no arms, one arm, two arms, or even more than two arms, given that they are all found in "possible" locations.

This property of the ISM model makes it very robust to occlusion. As we will see in the discussion of object detection, an object is given a score based on how many object parts are found. An object with missing pieces (occluded) will have a lower score than a completely visible object, but still much greater than simple background clutter. Inference can be made by the computer in a very natural way given this score – whereas the visible object may be "definitely" of a certain type, an occluded object may be "probably"

Figure 2.1: Examples of training data and associated segmentation masks.

or "possibly" of that type, depending on how much of the object is visible, as reflected by its score.

What is described in this chapter is the ISM method for detecting objects based directly on Leibe's work. Changes or additions to the method are noted where appropriate. The original contributions of this thesis are described in Section 3 and Section 4.

2.1 Training

The result of the training phase is the Implicit Shape Model (ISM), which consists of the *codebook*, a collection of *codebook entries*, and the *codebook vectors*. Codebook entries are distinctive image patches describing an object, and codebook vectors are a mapping from codebook entries to vectors describing possible object centers. The method of creating the codebook and associated vectors is described in detail in this section, and their use in Section 2.2. Fig. 2.1 contains examples of the training data used.

2.1.1 Extraction of Image Patches

The first step is to extract distinctive components from the training objects. The component descriptions used by Leibe, also used here, were 25x25 pixel image patches. In order to find only truly distinctive image patches, an interest point detector is used.

The detector chosen for this was the Harris corner detector[10], due to its speed, simplicity, and high rate of returning useful locations. The Harris corner detector works by finding areas in an image that have a high value for the second derivative of the image in both directions. A high value in one direction corresponds to an edge, and in both directions, a corner. In Chapter 7, other corner detectors are discussed. To eliminate patches describing background clutter from the ISM, only interest points lying on the object are used. Hand drawn segementation masks, examples of which are also shown in Fig. 2.1, are used to indicate which points lie on the object.

2.1.2 Clustering of Image Patches

Once the distinctive image patches have been extracted from all the training examples, they are clustered in order to reduce the number of patches that will have to be considered in future stages of the algorithm. More importantly, clustering similar patches results in clusters that represent a distinctive feature of an object class, rather than of a specific object. Combining multiple image patches results in codebook entries that have varying levels of detail, depending on how many image patches form each cluster.

Clustering is performed in an agglomerative fashion, using the Normalized Grayscale Correlation (NGC) as a similarity measure. The pairwise NGC is calculated for all patches, where p and q are the patches being compared and p_i and q_i are the individuals pixels within each patch.

$$\mathrm{NGC}(p,q) = \frac{\sum_i (p_i - \overline{p_i})(q_i - \overline{q_i})}{\sqrt{\sum_i (p_i - \overline{p_i})^2 \sum_i (q_i - \overline{q_i})^2}}$$

Significant speedup can be achieved by normalizing both patches to mean 0 and unit standard deviation before comparison (p^n and q^n refer to the normalized patches), reducing this equation to

$$\mathrm{NGC}(p^n,q^n) = \sum_i p_i^n \times q_i^n$$

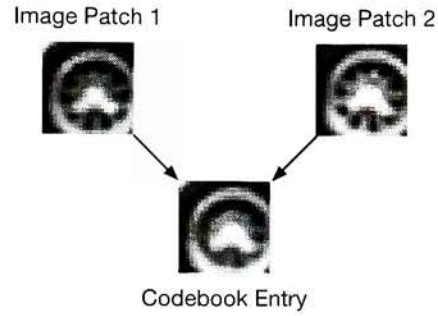

Figure 2.2: Similar image patches are clustered together, then combined to form a codebook entry.

This correlation produces a value between -1 and 1, indicating 100% negative and positive correlation respectively. A threshold of 0.7 was chosen for use in both the clustering stage, and later in the object detection stage. This was the value used by Leibe, and minor experimentation revealed that values between 0.6 and 0.8 performed acceptable levels of clustering, while values outside that range created clusters with too wide a range of image patches, thus loosing their power to describe distinctive image features for low values; or clustered very few patches, which lacked a general enough description of the object category for high values.

Clustering begins with each patch in its own cluster. The similarity between clusters C_1 and C_2 is calculated by the following formula, where p and q are patches within their respective clusters, and $|C_x|$ is the size of a cluster:

$$\mathrm{sim}(C_1, C_2) = \frac{\sum_{p \in C_1} \sum_{q \in C_2} \mathrm{NGC}(p, q)}{|C_1| \times |C_2|}$$

The patches in the pair of clusters with the highest value (most similar clusters) are combined to form a new cluster, and similarities are recalculated for this cluster. This process is repeated until the highest similarity between any pair of clusters has a similarity value less than the aforementioned threshold of 0.7. The resulting cluster centers – calculated by taking the normalized mean of patches within that cluster – make up the *codebook entries*. Figure 2.2 shows a cluster containing two patches and the resulting codebook entry. The original patches are very similar, so the resulting code-

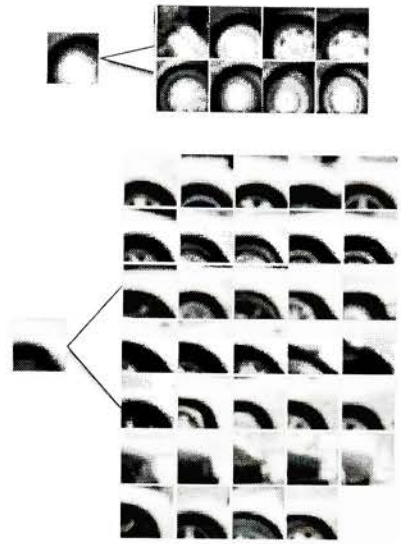

Figure 2.3: Codebook entries, and the patches from which they were formed.

book entry is also very similar, but has lost some detail. A certain amount of loss of detail is beneficial, as it prevents codebook entries from being too specific to be useful. More examples of images patches and codebook entries are shown in Fig. 2.3.

2.1.3 Calculation of Codebook Vectors

The codebook described in the previous step is the part of the shape model which identifies individual image patches as belonging to an object, but does not give any indication of the location of the object center relative to any discovered image patches. To localize each codebook entry, the training images are reprocessed. Image patches at Harris corners are compared to every codebook entry using NGC, and matches are recorded. A vector describing the x and y displacement between the matching image patch and the center of the object is added to the codebook vectors. The center of the object is known because the same segmented images from Section 2.1.1 are used. This will result in at least one codebook vector for each codebook entry and possibly many vectors, if the patch is very generic.

The codebook entries and corresponding codebook vectors are stored for

use during the detection phase of the algorithm.

2.2 Object Detection

The object detection phase of the ISM method is very closely related to the training method. In both phases, corners and image patches are extracted, then codebook entries are used, and finally codebook vectors. Note that this algorithm employs true object detection, rather than just object recognition. For example, Agarwal's detection algorithm [2, 1] is really a recognizer that can be moved across an image to find objects of a specific size and scale. I bring up this point because object detection is a more difficult task than object recognition. In object recognition, the question is "Is this an object of type X?" but in object detection, the question is "Where are the objects of type X?" The ISM method is a type of object detection. It has the capability to identify and localize multiple objects in a single image. The following description is of an object detection algorithm that can localize objects invariant to location, but not to scale or rotation. Scale and rotation invariance are commonly desired properties of object detectors, and in fact this algorithm can achieve scale invariance using image pyramids, as shown by Leibe in [19]. Presumably rotation invariance could be achieved in a similar manner, but neither scale nor rotation invariance are inherent in the algorithm.

2.2.1 Extraction of Image Patches

Extraction of image patches proceeds in the same manner as during the training phase, except that it is not known whether an image patch lies on an object or not. In fact, the vast majority of extracted image patches do not lie on objects, since the Harris corner detector finds corners in backgrounds, other object categories, etc., as well as the object category of interest.

2.2.2 Generalized Hough Transform

The method for localizing objects is based on the generalized Hough transform by Ballard [3]. Each image patch is matched against each codebook entry, and if the NGC is higher than the threshold, entries are added to the Hough-transform space. The Hough-transform space is an accumulator array

slightly larger than the actual image (to accommodate objects whose centers are outside the image), and each matched image patch adds a value equal to

$$\frac{\text{NGC}(c,p)}{|\text{vectors}(c)|}$$

to the locations $\{c + v_i : v_i \in \text{vectors}(c)\}$, where $\text{NGC}(c,p)$ is the normalized grayscale correlation between image patch p and codebook entry c, and $\text{vectors}(c)$ is the set of codebook vectors for c, as collected in Section 2.1.3. (x,y) locations in the accumulator are analogous to (x,y) locations in the original image.

Large clusters of data will form at locations in the accumulator around the true center of the object of interest. To understand how this works, consider first a simple example in which there is one codebook entry (for a car wheel) with two vectors associated with it (Fig. 2.4). In Fig. 2.5, the image patches containing the wheels of the car match the codebook entry, and each casts two votes, with score 0.5[1]. This causes three locations within the accumulator to be incremented – the left and right locations once, and the center location twice. Picking the largest value, we find the correct center of the car. Additional image patches on the car will match other codebook entries and further strengthen the center hypothesis. How the extra, unwanted hypotheses are eliminated will be discussed in Section 2.2.4.

The same principle holds for more complex examples. Fig. 2.6 shows an image from the UIUC test dataset[2]. Fig. 2.7 shows a representation of the Hough transform space for this image. Darker areas indicate a higher concentration and value of votes. Of the four areas of high concentration, the strongest two represent the locations of the cars in the image, and the weaker two are artifacts of the same type as seen in Fig. 2.5.

To find areas of high concentration, the Hough transform space is partitioned into squares roughly 10% of the object's maximum dimension, and non-maximal suppression applied to remove points next to each other. The remaining maxima are the initial object center hypotheses. The score of each object center hypothesis is the total weight of the Hough transform space in a circle of radius 1.5 times the square width. This yields an expected deviation from the true center of $< 5\%$ the maximum dimension of the object.

Leibe instead uses a mean-shift search starting from the object center

[1]The score would actually be slightly less than 0.5, since the codebook entry does not match the car wheels exactly

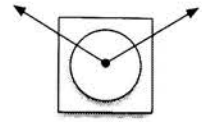

Figure 2.4: Codebook entry and codebook vectors for a stick-figure wheel.

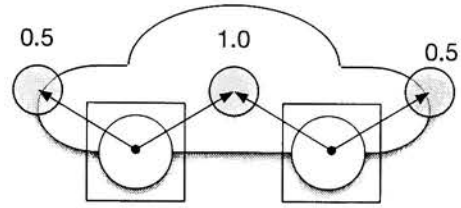

Figure 2.5: A stick-figure car with three location hypotheses.

hypotheses, instead of only looking at the weight of the area immediately surrounding the hypothesis. Mean-shift searching is more precise than the method used in this thesis, but imposes an additional computational burden. Section 5.4 compares these methods.

2.2.3 Simultaneous Segmentation of Object

Leibe's work also involves simultaneously providing a probabilistic segmentation for objects that have been localized. This is only mentioned here in support of the next step. It is important to note that acquiring an object segmentation using this method requires uniform sampling over the region containing the car, which is a computationally intensive process. I determined that for this thesis, object segmentation was not required, in order to increase the throughput of the algorithm.

2.2.4 Eliminating Extra Hypotheses

Liebe's method for deciding the number of objects in an image is based on the principle of Minimal Description Length (MDL), which states that the best description is one that minimizes the description length for the image, model and error. Hypotheses found in Section 2.2.2 are each tested to determine the saving they offer, based on each pixel's probability of being a specific

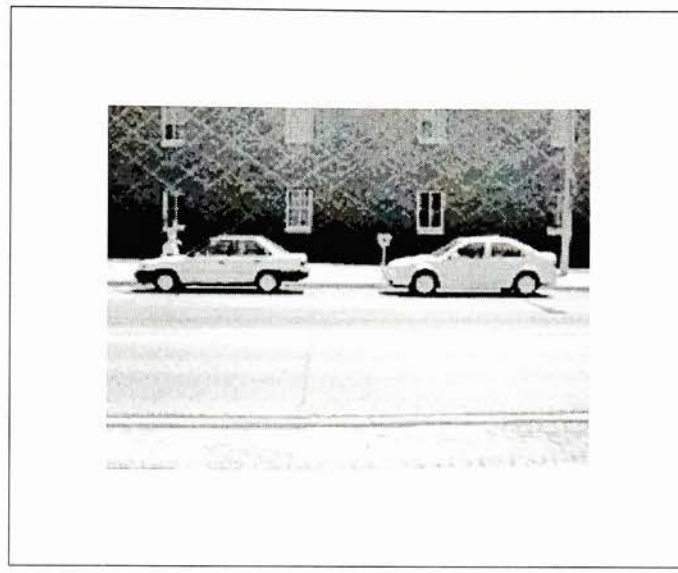

Figure 2.6: UIUC Test Image #10.

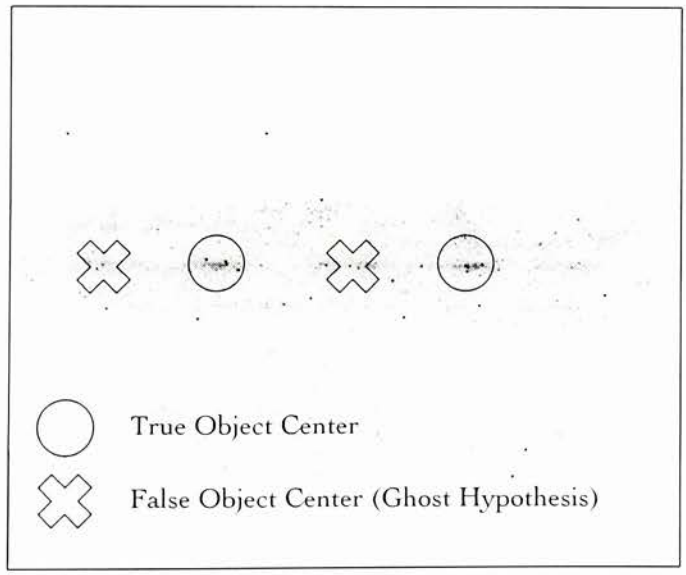

Figure 2.7: Hough-space with true hypotheses labeled for Fig 2.6.

object. The probability of an individual pixel being part of a specific object is determined during segmentation of the object. More detail on MDL and how it is used by Leibe can be found in [17].

2.2.5 Decision Making

In order to decide how many objects are in a scene, some decision needs to be made about which hypotheses are correct and which are just noise. Thresholds are commonly used when a numeric score is calculated. Although each hypothesis has a single numeric score, the values vary greatly between images, primarily dependent on the amount of background noise and interest points detected. The score cannot be normalized for this problem, because that would cause images with no detectable objects to return false detections. As a solution to this problem, I look at the derivative of the plot of sorted hypothesis scores. The rationale for this is that the true objects will have very high scores in relation to non-objects, regardless of the amount of background noise in the image. Taking the first derivative eliminates the effect of constant background noise, and leaves peaks where there are large differences between scores. The highest peak is taken as the separation between hypotheses describing objects and noise hypothesis. In Fig. 2.8, the largest difference is between object hypotheses 1 and 2, which indicates a single object in this frame.

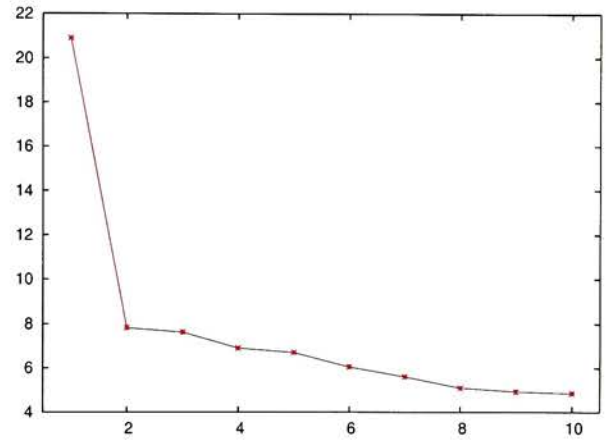

Figure 2.8: Scores of the top ten hypotheses from frame 50 of `RIT_cars_4_150px.avi`

Chapter 3

Enhancements to ISM method

This chapter describes enhancements to the core ISM method that were implemented. These enhancements were planned with several goals in mind: the first was to keep detection accuracy up while avoiding some of the more time-intensive stages of Leibe's ISM algorithm. The second was to handle some special cases, such as highly-occluded objects. For example, objects which are not fully inside the image area. The effect of applying these enhancements is analyzed in Chapter 5.

3.1 Elimination of Mean-Shift Clustering

Unlike what was used in this thesis, Leibe uses mean-shift clustering to localize the object centers in the Hough transform space. This gives very good localization of the object in question, but is computationally intensive, nearly tripling the running time of the algorithm. As shown in Section 5.4, using mean-shift search marginally improves the results for the still image database, however, the improvement was not deemed to be significant enough to justify the amount of time spent. It is possible to get almost as good detection by finding windows with high values in the Hough-transform space.

3.2 Hypothesis Starving

One of the issues with the ISM method, and generalized Hough transform methods in general, is the problem of areas of high concentration in the Hough-transform space that do not indicate the presence of objects. This

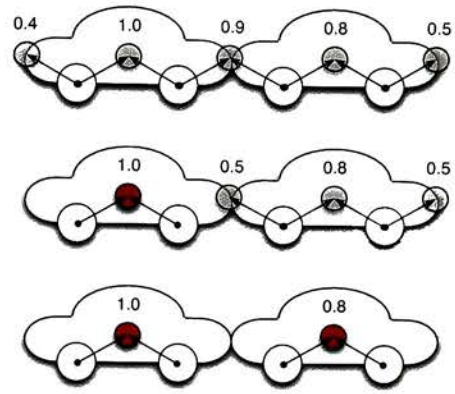

Figure 3.1: Illustration of Hypothesis Starving.

issue is particularly evident when dealing with images of cars, because one of the primary features of a car is the wheels, but this feature has two major interpretations, as either the front wheel or the back wheel. Figs. 2.4 and 2.5 illustrate this idea for the car example, and Fig. 2.7 shows the Hough-transform space of an image containing concentrations that do not correspond to objects (ghost hypotheses).

Leibe deals with this issue by verifying each hypothesis independently using the MDL criterion (Section 2.2.4). Given the segmentation associated with each hypothesis, and the assumption that objects opaquely occlude one another, the strongest hypotheses will be accepted first, and only if there is sufficient support will additional occluded hypotheses be accepted.

Hypothesis starving accomplishes much the same effect, but at an earlier stage of the process. Instead of considering an object's segmentation, it is possible to look back at the individual pieces of evidence. Using the same assumption that objects opaquely occlude each other, a single image patch can only belong to a single object in the image. If it can be determined that a patch belongs to a certain object, the support that the patch gave to other hypotheses can be removed, since it is now known that the patch is not part of that object. Fig. 3.1 illustrates this process. In this example, without hypothesis starving, the center of the image would incorrectly be chosen as the second object. However, after the votes from the first object are removed, the score for this false hypothesis falls under the other true object in the

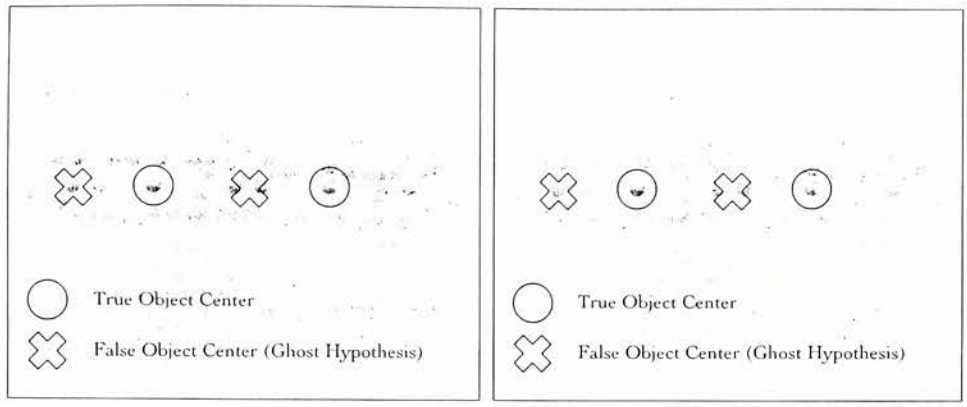

Figure 3.2: Hough-transform space, before and after hypothesis starving. These Hough-transform spaces are found after an initial filtering of Fig. 2.7.

scene. Fig. 3.2 shows that the false hypotheses have had their scores greatly reduced through hypothesis starving, while the correct hypotheses remain intact.

The primary advantages of hypothesis starving versus MDL verification is that it is patch based versus pixel based, and that it can occur earlier in the decision making process. Patch-based hypothesis starving operates a lower level of granularity, which can reduce the processing time needed to perform the calculations. Hypothesis starving also does not require any additional image processing beyond what has been performed up to that point in the process. In contrast, MDL verification requires the object segmentation, which is generated by uniformly sampling of the object region, a computationally intensive task.

Both hypothesis starving and MDL verification suffer from one major weakness. If the first hypothesis determined to be an object is incorrect, it will remove certain patches or pixels from true hypotheses, possibly causing missed detections in addition to the false detection. An example of this is shown in Fig. 3.3. In this example, the center ghost hypothesis in the center has a slightly larger value than either of the true centers. However, once the votes are removed, the true centers are reduced by so much that they cannot be distinguished from the background. For a comparison of results with and without hypothesis starving enabled, see Section 5.3.

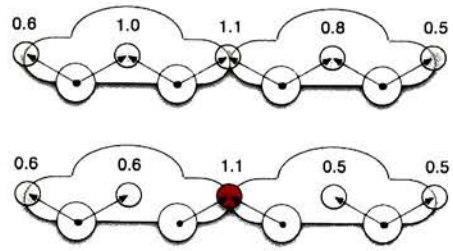

Figure 3.3: Example of Hypothesis Starving with incorrectly chosen primary hypothesis.

Due to time constraints, MDL verification was not implemented for this thesis, so a side-by-side comparison could be performed.

3.3 Hypothesis Boosting

Hypothesis boosting is a technique to aid in the object decision-making process described in Section 2.2.5. Because the simple decision-making process used in this work is based on a single number – the weight of votes in the Hough-transform space – the scores of occluded objects are reduced approximately proportionally to the amount of occlusion. In order to compensate for this, hypothesis boosting is performed on any objects suffering from occlusion. The most significant sources of occlusion in the video images are from the edges of the image, i.e. an object that is partially outside the image. In these cases, an object's true score (as if it were fully within the image) can be estimated by boosting the score in proportion to the amount of occlusion.

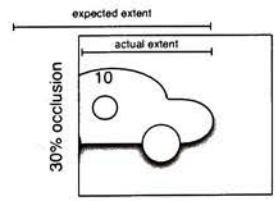

Figure 3.4: Example of occluded object suitable for Hypothesis Boosting.

The amount of occlusion can be determined by comparing the extent of the object to the center. Since the object extent is symmetric, the amount of occlusion can be determined by comparing the actual extent to the expected extent. Figure 3.4 shows an object that is 30% occluded by the edge of the image. This amount of occlusion is calculated by subtracting the actual extent of the object from the expected extent. If this object had an initial score of 10, its score would by boosted by 30% for a final score of 13.

Chapter 4

Object tracking using ISM

Leibe's work, described in [18, 17] and summarized in Chapter 2, presents some very interesting and promising results in the field of object detection in static images. The first part of this research focused on the static aspect of object detection, but the main focus is object detection in dynamic scenes. Leibe's approach has many advantages when dealing with dynamic scenes. As noted previously, it deals well with occlusion, a common occurrence in dynamic scenes. Also, the description of objects based on matched codebook entries provides a natural appearance model for an object tracker to use. This chapter contains a description of the object tracking algorithm, and provides discussion of the methods used.

In the next sections, *past objects* refer to objects that have been identified in frames prior to the current one. These objects may have been tracked over several frames, in which case they have a history of both appearances and locations. If the past object was detected recently, and evidence indicates it should still be present in the scene, it is described as an *active object*. *Present objects* refer to any objects that have been detected in the current frame. The goal of the object tracking portion of this algorithm is to find the nearest present object for each active past object. A minimum threshold must also be met, in order to prevent spurious matches.

4.1 ISM object detection

The first phase of the object tracking algorithm is the ISM static image detection algorithm. This algorithm returns the location of object of interest in

the image, and a list of matching codebook entries and their general location relative to the center of the object. The location is divided into quadrants to allow for some flexibility regarding the patch location. The codebook entries and quadrant numbers make up the *evidence* for an object hypothesis.

4.2 Object Tracking

All the present objects are compared against the active objects to determine if they match. There are two parameters for matching: the appearance of the present object, described by the evidence, must match the appearance model of the past object; and the current location must be consistent with the past locations. The next two sections describe the process for developing and matching these parameters.

4.2.1 Appearance Model

Once evidence has been collected for an object, this becomes the basis of its appearance model. Two different methods for matching past objects to present objects were created, based on the temporal stability properties discussed in Section 6.2. The results are shown in Section 5.5.2.

The patch frequency model favors present objects that have the same evidence as past objects. The score of a present object/past object match is the sum over the present object's evidence of the number of times that piece of evidence has appeared in the previous object, normalized by the number of active frames of that object. The number of active frames of an object is used instead of the age to prevent bias against often occluded objects. See Section 4.3 for how occluded objects are handled. pr is the present object, pa is the past object, $pa.e$ and $pr.e$ are the associated evidence, and pa.active_frames is the number of pa's active frames.

$$\text{pf\_score}(pr, pa) = \sum_{e \in pr.e} \sum_{e \in pa.e} \frac{1}{pa.\text{active\_frames}}$$

This appearance model is biased towards matching present objects with past objects of similar appearance throughout their lifetime, regardless of recent fluctuations of the appearance model of the past object.

The other appearance model, patch runs, favors objects that have had a large portion of the appearance model remain consistent over the lifetime of

the object. The similarity score between a present object and a past objects is the sum of the past object's run lengths for each piece of evidence in the present object. The run length of a piece of evidence is the number of immediately preceeding consecutive frames that piece of evidence has appeared in.

$$\text{pr\_score}(pr, pa) = \sum_{e \in pr.e} \text{runlength}(pa.e, e)$$

This appearance model favors past objects that not only have consistent evidence, but are longer running as well. It is heavily weighted by the most recent appearance of the object, which may cause problems with reaquiring the object if it undergoes significant occlusion.

4.2.2 Location Model

The other parameter with which present objects must match past objects is the location. The expected location can be calculated using a Kalman filter[14][1], which is used as an estimator of the object's velocity. Kalman filters are often used to estimate noisy linear systems, such as the position and velocity of moving objects. Figure 4.1 shows an example from the Matlab package. In this example, a series of points representing movement (*actual*) are measured using a noisy process (*measured*). After applying Kalman filtering and smoothing the *smoothed* position and velocity values can be estimated. From this smoothed path, the next point can be approximated (*estimate*) by adding the smoothed velocity to the last measured position.

The velocity of objects in the system is assumed to be approximately linear, so the next expected position can be estimated using the Kalman filter. The variance of an object's velocity can be calculated using the previous velocity data. The distance between an object's estimated position and its actual position can be found using

$$D = \sqrt{\left(\frac{E_x - A_x}{\sqrt{v_x}}\right)^2 + \left(\frac{E_y - A_y}{\sqrt{v_y}}\right)^2}$$

where E is the expected position, A is the actual position, v is the variance of the position, and the result D is the normalized distance of the object from

[1]The Matlab package used for running Kalman filters can be found at Kevin Murphy's website: http://www.cs.ubc.ca/~murphyk/Software/Kalman/kalman.html

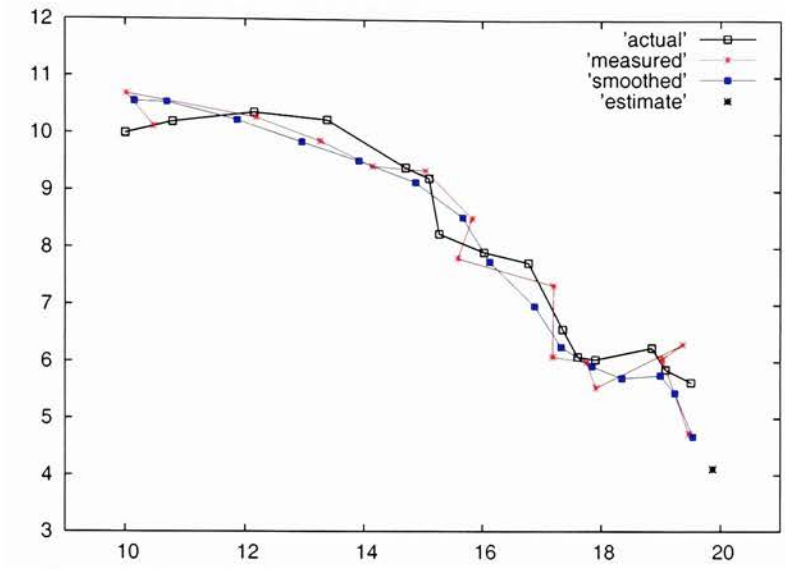

Figure 4.1: Example of Kalman filtering.

the expected position. The unit of D will be the number of standard deviations from the expected location. Assuming a normal curve, the probability of a match can be calculated from these two positions.

4.3 Tracking Procedure

Both the appearance and location models are used when determining which past objects match which present objects. The pr or pf score is multiplied by the probability of D to get the final score. This weights the appearance model more heavily than the location if the present object's position is close to the expected position, but the location model more heavily if present object's position is far away. Each present object is matched with the past object that is most similar, provided no other present object has a higher score, and that its score is above a minimum threshold.

4.3.1 Dealing with Occlusion

Occasionally, due to real or imagined occlusion, an object cannot be found near its expected location in a frame. If there is reasonable evidence to believe that the object should still exist, it is placed at its expected location, and marked as occluded. For purposes of this thesis, objects were eligible to be marked as occluded after they had been detected in three consecutive frames. Objects with lifetimes of one or two frames were not eligible to become occluded, and were simply removed from the list of active objects if they were not found in the next frame. The object's evidence is not changed if it becomes occluded, although there is a maximum lifetime of any occluded object of four frames past the last detection. Four frames is sufficient for minor interruptions due to inaccurate detections, and also allows for many vehicles to be accurately tracked as they exit the viewing field.

4.4 Motion Invariance

One of the great strengths of this method over many other methods of object detection and tracking in video is that it does not require a stable camera. Because the first stage of the algorithm is a static object detection algorithm, it can localize objects anywhere in the scene, regardless of whether the background is consistent from frame to frame. This algorithm can be used to track objects even when the background is itself moving. Many popular object tracking algorithms assume a relatively stable background, and find objects in the foreground only. Requiring a stable background prevents these algorithms from working with pan-and-zoom cameras, vehicle-mounted cameras, or in any kind of television or movie footage.

Chapter 5

Results

5.1 Training Data

The training data for all the results reported here was a series of 50 images of cars collected by Bastian Leibe[1], mirrored to represent cars facing in both directions. A variety of cars is represented, including sedans, sports cars, SUVs and compact cars, most of which are European styled cars. Hand drawn segmentation masks are also provided, and used during training.

5.2 UIUC results

Agarwal et al. [2] have collected a series of 170 images containing 200 instances of cars, which is used as an evaluation measure for this work. Leibe also uses this database in [17], which makes direct comparisons simple. The baseline configuration used for testing against the UIUC database used the simple decision method making described in Section 2.2.5 with Hypothesis Starving enabled, and both mean-shift searching and hypothesis boosting disabled. Because this method of decision making is somewhat primitive, results are also presented when the algorithm was provided with the correct number of cars in an image. This causes the number of false detections to equal the number of missed detections, yielding the Equal Error Rate of the algorithm (See Appendix A). The EER point has a higher F-measure than the simple decision making for all tests, giving an approximate maximum

[1]Images are available at http://www.vision.ethz.ch/leibe/data/

F-measure, assuming a better decision-making process.

The baseline configuration produced 14 false positives and 32 missed detections of the 200 cars in the dataset. Many of these missed detections were due to the decision-making process only choosing one car in an image with multiple cars. When the algorithm was told the number of cars present in a scene, the number of missed detections dropped to 21, and false positives rose to 21 as well. With simple decision making this translates to a recall of 0.840, precision of 0.935 and F-measure of 0.885 (See Appendix A). When the correct number of cars is known, R,P,F = 0.895 (all three are the same when the correct number of cars is known beforehand).

5.3 Hypothesis Starving

When hypothesis starving was disabled, there were 30 false positives and 17 missed detections. R = 0.915, P = 0.859 and F = 0.886. When the number of cars was given, there were 9 mistakes (R,P,F = 0.955). Hypothesis starving increases precision at the cost of lowering recall. The recall/precision tradeoff is well-understood, and the effect of lowering recall to raise precision seen here is typical.

5.4 Mean-Shift Search

When mean-shift search was enabled, recall and precision were both increased above the baseline, with only 16 missed detections and 13 false positives (R = 0.920, P = 0.934, F = 0.927). When the number of cars was known, there were only 8 errors (R,P,F = 0.960). However, enabling mean-shift searching nearly tripled the algorithm's processing time, which is why it was disabled in the baseline. This 3x slowdown occurred even when the number of votes was reduced by only processing the 5000 most significant votes. A typical image in the UIUC database generates around 100,000 votes, most of which are very weak. Mean-shift searching is approximately $O(n^3)$, so fully processing the votes would be prohibitive.

	Simple Detection			Known #
	R	P	F	R,P,F
Harris	0.840	0.935	0.885	0.895
No hypothesis starving	0.915	0.859	0.886	0.955
Mean-shift search	0.920	0.934	0.927	0.960

Table 5.1: Summary of test results on UIUC database.

5.5 Video

5.5.1 Method

To test the object detection and tracking algorithm, several videos of cars in natural scenes were shot at Rochester Institute of Technology in June 2005. The videos are stationary but hand-stabilized, which means that there is image motion due to the motion of the camera. A variety of different cars, trucks and SUVs are filmed, along with a number of other non-car vehicles. Videos `RIT_cars_3_225px.avi`, `RIT_cars_4_150px.avi`, `RIT_cars_4_100px.avi` were used to test the algorithm's ability to track cars in a simple, mostly stationary scene. `RIT_cars_3_225px_short.avi` and `RIT_lot_cars_5_300px.avi` were used to test the ability to track cars in a scene where the camera is moving. The former video was generated by clipping the video to the area surrounding one vehicle (Car #12 in Table 5.4), simulating a camera tracking the vehicle. In the latter video, the camera operator actually tracks the vehicle. Also in that video, the angle between the camera and car varies, and the camera is pointed downward onto the car (which results in an image quite a bit different than the training data). Videos were recorded using a DV camcorder at 720x480 pixel resolution, at 30 frames per second. The original frames were interlaced, so the video was made progressive by duplicating every other scan line. In all cases, the videos were scaled so the average size of the cars matched the training data, in which the cars had an average length of 225 pixels. The average car length in each video is recorded in its filename (ex. `RIT_cars_4_100px.avi` was scaled by 2.25 before processing).

The object tracking algorithm used the ISM created from Leibe's training data, and the top ten "objects" from each frame were kept (no decision making was used), even though there was usually only one actual object in the scene at a time. This was done to allow for analysis even when decision-

making did not necessarily choose the correct hypothesis for a car. Hypothesis boosting was used, and hypothesis starving and mean-shift search were disabled. Each video was analyzed manually by examining the actual cars in the video, and categorizing the tracker into one of six categories, described in Table 5.2.

Near Perfect	The object was tracked in $> 95\%$ of the frames it was on screen.
Very Good	The object was tracked in $> 75\%$ of the frames it was on screen.
Good	The object was tracked in $> 50\%$ of the frames it was on screen.
Fair	The object was tracked in $> 25\%$ of the frames it was on screen.
Poor	The object was detected, but not tracked, in $> 25\%$ of the frames it was on screen.
None	The object was not detected, or was detected in $< 25\%$ of the frames it was on-screen (to account for noise).

Table 5.2: Video Tracking Result Descriptions

The number of frames of object lifetime includes all frames where more than 10% of the vehicle is visible. This is why Table 5.2 is quite liberal in the percentages, because even a "perfect" detector could not reasonably be expected to detect the vehicle in 100% of its frames under these conditions. The number of frames the object is tracked in includes frames where the the object was marked as "occluded." This occurs when an object has been tracked for a number of frames and its position can be estimated, but the vehicle was not found in that frame. This often occurs as vehicles are exiting the frame, resulting in some vehicles being tracked for one or two frames after they have exited the field of view.

The level of occlusion is also reported in order to give an accurate account of the videos. Most of the cars were not significantly occluded during their time on-screen, aside from entry and exit from the field of view. Occlusion levels are described in Table 5.3.

5.5.2 Results

The results for PR and PF trackers are highly correlated, because they both use the same initial set of possible hypotheses per frame. In some cases, choosing the correct hypothesis per frame may be easy, because there is only one reasonable choice. In other frames, this is more difficult, because there

Light	Almost no occlusion. Small tree trunks, sign-posts, etc.
Lower	Significant occlusion of lower half of vehicle. Bushes covering the lower $\frac{1}{3}$ of the vehicle. Light occlusion of the rest of the vehicle.
Out-of-scale	This vehicle was at the wrong scale for detection.
Non-car	This vehicle was not a car/pickup/SUV.

Table 5.3: Video Tracking Occlusion Level Descriptions

are multiple hypotheses in the general vicinity of the "correct" one. These are the cases where the two different trackers have the possibility of diverging.

One of the interesting phenomena discovered when analyzing these results was the possibility of a vehicle to "capture" an existing object hypothesis. Because the top ten hypothesis are kept for each frame in this experiment, several weak hypotheses are present in each frame. Since the background is generally stable, they form weak, stationary objects at those locations. When a car occludes the weak object, it will either remove the weak object from the active objects, or the tracking algorithm may decide that the weak object matches the actual object. This is possible if the weak object has very little evidence, and will score relatively high when compared to a strong object containing many pieces of evidence, since the strong object may have a superset of the weak object's evidence. When an object is "captured", it is marked in the table.

Accuracy of the algorithm between patch frequency and patch runs is similar, with patch runs scoring slightly higher on some vehicles. However, breaks in the object path were much more common in the patch frequency results than in the patch run results. This occurs when an object is tracked, the system looses track of it, then locates it again in a later frame but labeled as a different object. These occurrences are marked in the table.

5.6 Speed

The work described here was performed on an Intel Pentium 4 3GHz CPU with hyperthreading enabled, and 2GB of RAM. The code was written in Matlab. The speed of the algorithm depended upon the size of the codebook and the number of interest points detected per frame, which in turn was related to the scaled size of the image or frame. Each image in the UIUC

Car #	Description	Occlusion	P.F. Result	P.R. Result
1	Black 2-door sedan	Light	Good(3)[a,b]	Good
2	Light Blue 4-door sedan	Light	Good(2)[a]	Good(4)[a,b]
3	Red 4-door sedan	Light	Very Good(2)[a,b]	Very Good[b]
4	Black 2-door sedan	Light	Fair(2)[a,b]	Fair[b]
5	White 4-door sedan	Light	Good(2)[a,b]	Good[b]
6	Black SUV	Lower	None	Poor
7	Light Blue 4-door sedan[c]	Out-of-scale	—	—
8	White SUV	Light	Very Good(3)[a]	Very Good
9	Black 4-door sedan	Lower	Poor	Poor
10	White FedEx truck[d]	Non-car	—	—
11	Red & White Cement truck[d]	Non-car	—	—
12	White pickup truck	Light	Near Perfect(2)[a]	Near Perfect
13	Black 4-door sedan	Light	Poor	Fair

Table 5.4: Results for `RIT_cars_3_225px.avi`

[a]There were breaks in the tracking of this object. The number of segments is given
[b]This object "captured" an existing weak object
[c]The wheel-well areas of this car were tracked for a portion of the lifetime of this object
[d]The various wheels of these trucks were tracked through portions of the video

Car #	Description	Occlusion	P.F. Result	P.R. Result
1	Yellow Construction Shovel	Non-car	—	—
2	Green Station Wagon	Light	Good	Good
3	White 4-door sedan	Light	Very Good	Very Good
4	Black 4-door sedan	Light	Good(2)[a]	Good
5	Black 4-door sedan	Light	Near Perfect	Near Perfect
6	Red SUV	Light	Near Perfect	Near Perfect
7	White Construction Truck[b]	Non-car	—	—
8	Silver 4-door sedan	Light	Near Perfect	Near Perfect
9	Light Green Minivan	Light	Very Good	Very Good
10	Bicycle	Non-car	—	—
11	Gray Construction Truck[b]	Non-car	—	—

Table 5.5: Results for `RIT_cars_4_150px.avi`

[a]There were breaks in the tracking of this object. The number of segments is given
[b]The various wheels of these trucks were tracked through portions of the video

Car #	Description	Occlusion	P.F. Result	P.R. Result
1	Grey Construction Truck[a]	Non-car	—	—
2	Grey 4-door sedan	Light	Very Good	Very Good
3	Black 4-door sedan	Light	Near Perfect(7)[b]	Near Perfect(2)[b]
4	White Minivan	Light	Very Good	Very Good
5	Grey 4-door sedan	Light	Very Good(3)	Very Good(2)[b]
6	White 4-door sedan	Bottom	Very Good(3)	Good(2)[b,c]
7	White 4-door sedan	Bottom	Good(2)	Very Good(2)[b,c]
8	White SUV	Bottom	Good(2)	Good
9	Black Compact	Light	Good	Very Good
10	Beige Station Wagon	Light	Near Perfect(2)	Near Perfect
11	Red 4-door sedan	Bottom	None(2)	None(2)[b]
12	Blue 4-door sedan	Light	Near Perfect	Near Perfect

Table 5.6: Results for `RIT_cars_2_100px.avi`

[a]The various wheels of these trucks were tracked through portions of the video
[b]There were breaks in the tracking of this object. The number of segments is given
[c]This object "captured" an existing weak object

Car #	Description	Occlusion	P.F. Result	P.R. Result
1	White pickup truck	Light	Very Good(3)[a]	Very Good

Table 5.7: Results for `RIT_cars_3_225px_short.avi`

[a]There were breaks in the tracking of this object. The number of segments is given

Car #	Description	Occlusion	P.F. Result	P.R. Result
1	Red 2-door sports car	Light	Very Good(5)	Good(5)

Table 5.8: Results for `RIT_lot_cars_5_300px.avi`

database and each frame of video took 30-120 seconds to process. The UIUC images were consistent within 5-10 seconds of each other, and each frame of the same video took a similar amount of time, varying by 15-30 seconds, primarily depending on the number of interest points detected per frame.

Chapter 6

Discussion

6.1 Usefulness of Object Patches

In my analysis of video data, the question arose of which patches were most useful for tracking objects. Data from the three videos was analyzed to determine which codebook entries were most commonly found in objects, and which were most commonly found in background clutter. If the most useful patches could be identified, and the least useful removed, this could help improve the ISM method's accuracy. The videos were analyzed by manually circumscribing a box around the true vehicles, and recording which codebook entries were found inside and outside the box in each frame. The number of times each patch appeared inside the box was compared to the number of times it appeared outside, normalized by the total number of patches in each group to account for the fact that there were significantly more non-object matches than object matches. Codebook entries that appeared significantly more inside the box (matching the object) were marked as *useful*, and compared to codebook entries that appeared significantly more outside the box (matching the background), marked *not-useful*.

The two groups of codebook entries, *useful* and *not-useful*, were compared by looking at the number of training image patches that formed them, the number of codebook vectors associated with them, and the image entropy of the codebook entries. Image entropy is the entropy of an image histogram containing 256 bins, and is an indicator of how much information is stored in that patch.

Table 6.1 summarizes the results of this analysis. There is essentially no

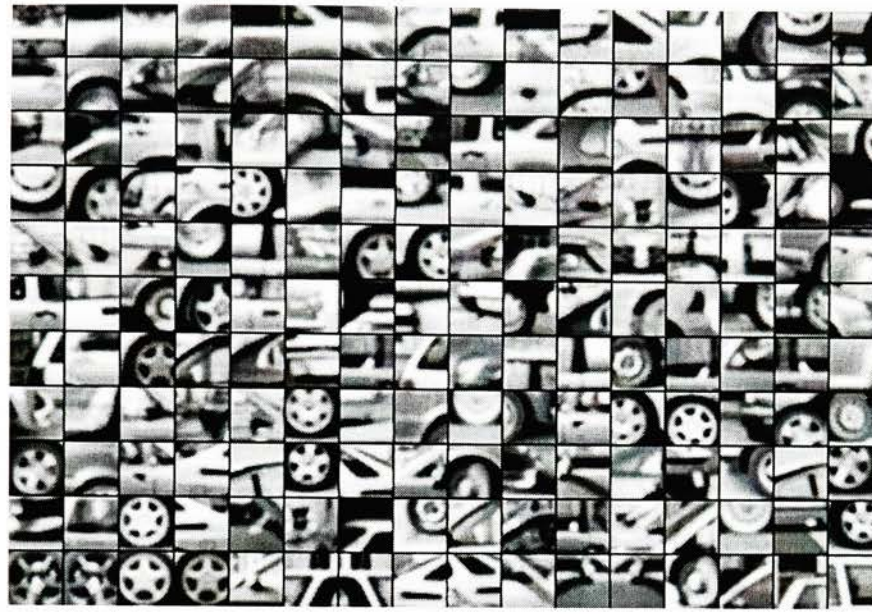

Figure 6.1: Best 150 *useful* patches.

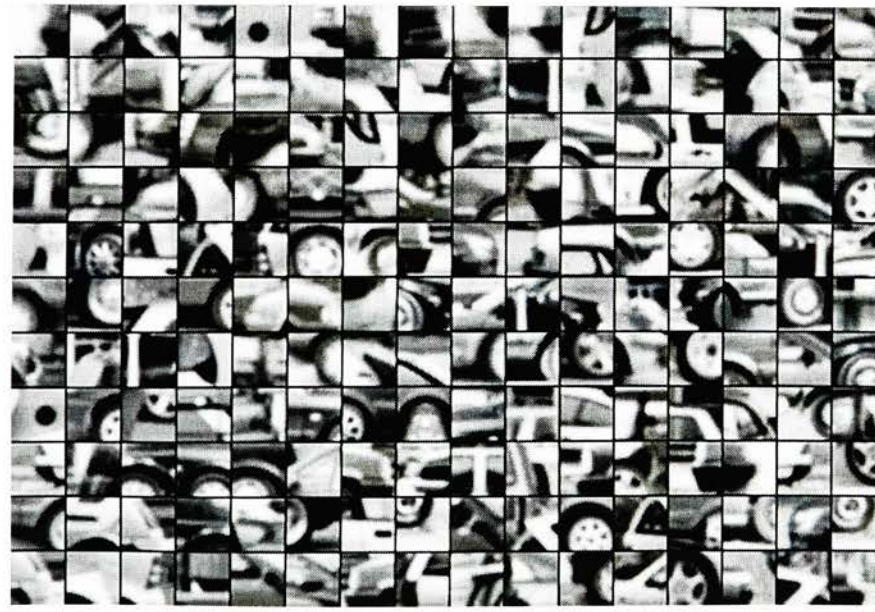

Figure 6.2: Worst 150 *not-useful* patches.

	#	# patches		# vectors		entropy	
		μ	σ	μ	σ	μ	σ
useful	160	1.863	4.103	9.550	18.826	0.741	0.024
not-useful	429	1.620	1.865	6.692	11.100	0.741	0.024

Table 6.1: Comparison of *useful* and *not-useful* codebook entries.

difference between the numerical descriptions of *useful* and *not-useful*. In Fig. 6.1 and Fig. 6.2, the patches that had the highest and lowest object to non-object ratios are shown. When these patches are examined manually, the difference appears to be slight. In general, *useful* seems to contain more parts that are easily recognizable as car parts than *not-useful*. Both classes contain about the same percentage of parts containing wheels, 24% for *useful* and 23% for *not-useful*, but the wheel parts in *not-useful* are usually more blurred, or a less common type of wheel.

In one sense, this result is quite surprising. I expected, prior to running this experiment, that either the numerical analysis or the manual analysis would uncover some distinctive trait that separates the two categories. For example, early results indicated a significantly higher percentage of wheel patches in *useful*. On the other hand, this result does show the flexibility of the ISM method. Even given a flawed codebook (that contains numerous *not-useful* entries), the algorithm is still able to localize objects with remarkable accuracy. It also shows that any analysis of the goodness of patches would have to take place outside the current system. Possible future work in this regard is mentioned in Section 8.5.

6.2 Temporal Stability of Object Patches

Given that a category of *useful* patches cannot be predetermined for a particular codebook, other properties of patches were investigated. One of the properties that is directly analogous to human vision is temporal stability. This means that the same patch appears in a similar location in sequential frames of the same object. Patches with high temporal stability can be used to track objects of interest.

Two types of temporal stability were tested in the three test videos. The first was simple patch frequency. Figure 6.3 shows the percentage of object frames patches were observed in. Because a patch can appear multiple times

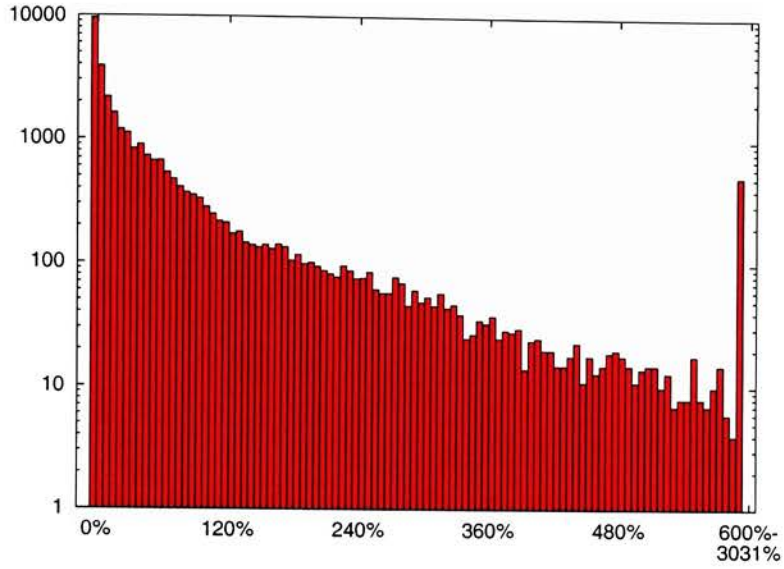

Figure 6.3: Patch Frequencies

in the same quadrant in the same frame, the percentage of frames can exceed 100%. The vast majority (notice the logarithmic scale) of patches appear in less than 100% of an object's frames. Only a few patches show a very high consistency.

The second type of temporal consistency that was measured was patch run length. The run length of a patch is the number of successive frames in which the same patch appears in the same location on the same car. In order to allow for small errors, locations were divided up into object quadrants. The percentage of frames a patch was found in the same quadrant in sequential frames is recorded in Figure 6.4. Note that this does not record only the maximum run length of a patch, so patches with high consistency contribute to all the preceding run lengths. A very fast fall-off was observed, especially after a run length of 60% of the object's lifetime. Figure 6.5 shows the maximum run length of patches. This graph shows that the number of patches with maximum run lengths between about 20 and 80 percent of the object lifetime are roughly the same.

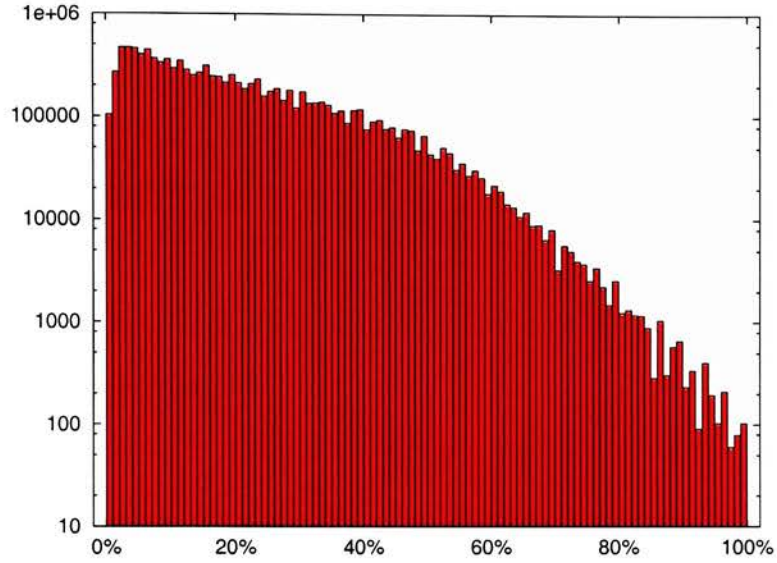

Figure 6.4: Run lengths

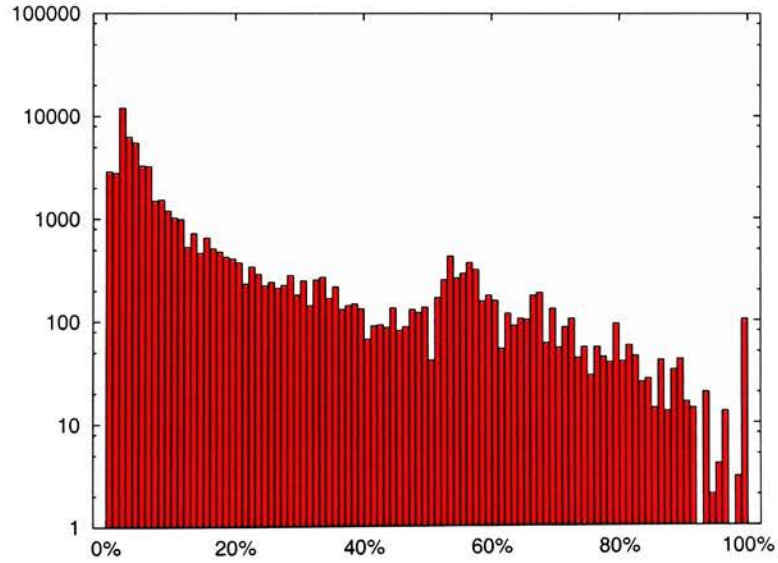

Figure 6.5: Maximum run lengths

Chapter 7

Comparison of Corner Detectors

Based on the results of Chapter 5, one possible area for improvement discussed was in the selection of image patches for processing. The Harris corner detector was originally chosen as an interest point detector due to its use in Leibe's original work, along with speed and ease of implementation. In addition, in [24], Schmid et al. rate an improved version of the Harris detector as the best interest point detector in terms of repeatability and information content. The goodness metric of information content is applicable to the ISM method, as it is important that interest points are located in regions of high information content. Repeatability is also very important, as similar interest points must be detected in different images in order for patches to be matched.

There are, of course, other interest point detectors that can be used, many approaching the problem from a completely different angle than the Harris detector. Contour based corner detectors are another branch of detectors. The Horaud detector, a contour based detector, performs well in the information content portion of Schmid's tests, although it is the worst performer in most of the repeatability tests. To see how a contour based corner detector compares, I chose to test an enhanced version of the Curvature Scale Space (CSS) algorithm by He and Yung [11]. In the CSS algorithm, T-corners (intersections of edges) and points of maximum curvature along contours are detected. The version by He and Yung has the additional advantage of finding scale-independent corners, a feature the standard Harris detector does not have.

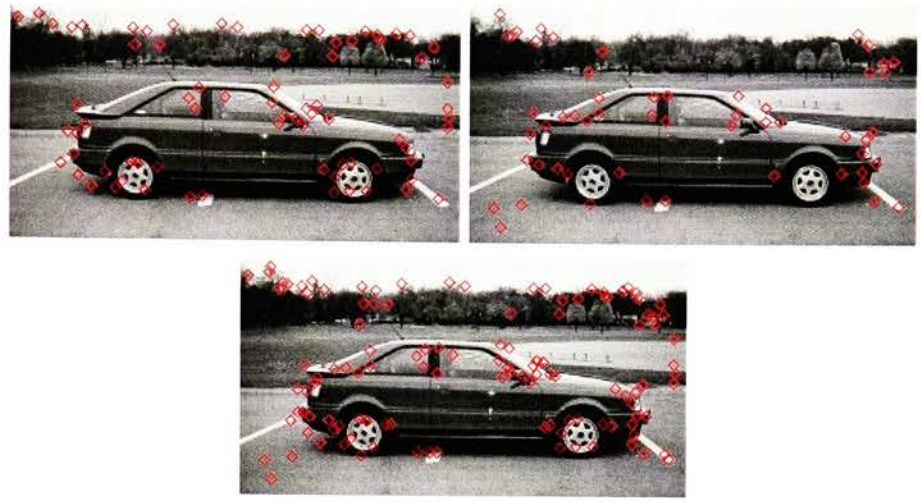

Figure 7.1: The three different interest point detectors tested.
Upper left: Harris, *upper right*: Enhanced CSS, *lower*: joint detector.

A joint corner detector was also tested, which returned as corners points identified by either the Harris or CSS detector, with nearby corners merged.

7.1 ISM comparison

The comparison of the different corner detectors begins with a comparison of the codebook entries and vectors that were generated with the different detectors. The results are summarized in Table 7.1. Between the Harris and CSS methods, the number of patches extracted from the training data, and the number of codebook entries generated from the patches were nearly identical. To compare the content of the codebooks, the number of similar codebook entries was recorded. Codebook entries were defined as an entry that has a high correlation coefficient (See Section 2.1.2) with any entry in the other codebook. About 55% of the Harris and CSS codebook entries were determined to have a similar entries to each other, and most of the Harris and CSS codebook entries were also present in the joint detector.

	Harris	CSS	Joint
Patches Extracted	3618	2549	5497
Codebook Entries	1210	1149	1919
Codebook Vectors	23001	13715	42506
Codebook Entropy	6.2494	6.4068	6.6886
Shared Patches (w/ Harris)	–	646	1427
Shared Patches (w/ CSS)	661	–	1382
Shared Patches (w/ Joint)	1181	1119	–

Table 7.1: Summary of ISM properties created by different corner detectors

7.2 UIUC database

7.2.1 CSS Detector

When tested on the UIUC database, the version of ISM based on the CSS corner detector was noticeably inferior to the Harris version. With simple detection, this ISM missed 52 cars in 49 images, and had 99 false detections in 51 images. This gives R = 0.740, P = 0.599 and F = 0.662. The number of false detections is particularly high because of four images with 5 or more false detections. With those images removed, there are 59 remaining false detections in 47 images, yielding R = 0.740, P = 0.711 and F = 0.725. When the number of cars was known beforehand, 56 cars were missed for R,P,F = 0.720. These results are significantly worse than those generated by the Harris ISM.

7.2.2 Joint Harris/CSS Detector

The ISM created from the joint detector that combined the results of the Harris corner detector and the CSS corner detector performed similarly to just the Harris detector alone. On the UIUC database, there were 37 missed detections and 11 false detections. Most of the missed detections (25) were instances where, like in Section 5.2, only one car was detected in an image containing multiple cars. When the number of cars per image was known beforehand, the algorithm made only 11 mistakes, which perhaps more accurately reflects an improvement over just the Harris alone. This improvement did not come without a price, as the ISM and test set each took nearly twice as long to run as either of the other two methods.

	Simple Detection			Known #
	R	P	F	R,P,F
Harris	0.840	0.935	0.885	0.895
CSS	0.740	0.599	0.662	0.720
CSS*	0.740	0.711	0.725	
Joint	0.815	0.937	0.872	0.945

CSS* is CSS with the four images with high numbers of false positives removed.

Table 7.2: Error measures on UIUC database for different corner detectors.

Chapter 8

Future Work

This research has raised nearly as many questions as it has answered, and suggested many ideas for future work. Unfortunately, time constraints prevented me from implementing these ideas in this thesis.

8.1 Improved Feature Descriptors

During the course of this research, the validity of using image patches as feature descriptors was brought into question. This is primarily due to the difficulty in determining how similar image patches are to each other. The normalized correlation used in this work has a number of issues, for example, otherwise similar patches may differ greatly in only one or two locations and cause the patches to be marked as different. Bhat et al. [5] suggest the use of ordinal measures to alleviate this problem. Another problem with correlation based measures that Bhat's solution doesn't help with is the large number of calculations required each time a patch comparison is performed – O(patch size^2) multiplications per comparison. Possibilities to reduce this computation cost include using a different type of descriptor altogether. In defense of using image patches however, their use is very intuitive and they seem to work well enough for the algorithm to function.

8.2 Multiple Object Categories

The ISM method is not limited to just cars – as Leibe shows in [17], this detection method can be used on categories as disparate from cars as cows.

Extending the ISM model to simultaneously detect multiple object categories is both straightforward and economical. In another feature-based algorithm, Torralba et al. [27] show that the computation and memory growth of their local feature based algorithm is logarithmic rather than linear in the number of categories. This is a huge advantage over systems that must use a separate detector for each object category.

8.3 Intelligence Decision Making

A notable weakness of the method implemented in this thesis is the decision-making process to decide the number of objects in an image. Leibe uses his segmentation as a verification step. Lebo [16] suggests the use of a trained classifier over various hypothesis properties, such as the pre- and post-starvation scores.

8.4 Probabilistic Segmentation

As described in Section 2.2.4, Leibe's original work combined object detection with a probabilistic segmentation. The extension of that technique to this work might yield additional temporal information that could be exploited.

8.5 Patch Testing

Section 6.1 shows that certain patches are more useful to the algorithm than others, and that just looking at the image patches themselves, or at their ISM properties is not enough to determine which ones fall into either category. An external measure of usefulness would likely boost the effectiveness of the algorithm.

8.6 New Training Database

Leibe's database contains only 50 cars, and does not accurately reflect the types of cars that a North American video surveillance system is likely to see. A larger database containing a more representative sample of cars would increase the algorithm's performance. Another possible weakness of the train-

ing phase of this algorithm is the lack of negative training exemplars. No information about non-object image patches is recorded. Including negative training examples in the database, and learning from them, might be used as the useful measure mentioned in the previous section.

The database also holds more information about image patches that has not yet been exploited. Including the joint probabilities of patches could lead to a more robust Implicit Shape Model. Lebo [16] accomplishes something similar with his co-activation networks.

8.7 Object tracking

The Kalman filter is an old standby of object tracking, but other techniques exist. Perhaps one of the most useful properties of some of the other tracking algorithms that exist is the ability to simultaneously track multiple hypotheses. CONDENSATION [13] is one of most popular algorithms for this purpose. In [12], Hu et al. list some other search strategies as well.

Appendix A

Precision, Recall and F-measure

Some common metrics for evaluating retrieval algorithms are the recall, precision and F-measure values [30], which are explained below.

Let r = # of objects correctly identified
a = # of non-objects identified as objects
R = # of actual objects

Recall is the ratio of objects correctly identified to actual objects:

$$\mathrm{R} = \frac{r}{R}$$

Precision is the ratio of objects correctly identified to total objects found:

$$\mathrm{P} = \frac{r}{r + a}$$

F-measure is their weighted average:

$$\mathrm{F} = \frac{(\beta^2 + 1)\mathrm{RP}}{\beta^2\mathrm{P} + \mathrm{R}}$$

where β indicates the relative importance of recall and precision. $\beta = 1$ corresponds to equal weighting, and is used throughout this paper.

Equal Error Rate (EER) is the point at which R = P.

Bibliography

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, November 2004.

[2] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *Proceedings of the European Conference on Computer Vision*, volume 4, pages 113–130, Copenhagen, Denmark, May 2002. Springer-Verlag.

[3] D. H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 12(2):111–122, 1981.

[4] J. Barron, D. Fleet, S. Beauchemin, and T. Burkitt. Performance of optical flow techniques. *CVPR*, 92:236–242, 1992.

[5] D. Bhat and S. Nayar. Ordinal measures for image correspondence. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(4):415–423, april 1998.

[6] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

[7] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part II*, pages 109–124, London, UK, 2002. Springer-Verlag.

[8] R. R. Brooks, L. L. Grewe, and S. S. Iyengar. Recognition in the wavelet domain: a survey. *Journal of Electronic Imaging*, 10:757–784, July 2001.

[9] B. Epshtein and S. Ullman. Identifying semantically equivalent object fragments. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*

(CVPR'05) - Volume 1, pages 2–9, Washington, DC, USA, 2005. IEEE Computer Society.

[10] C. Harris and M. Stephens. A combined corner and edge detector. In *Fourth Alvey Vision Conference*, pages 147–151, 1988.

[11] X. C. He and N. H. C. Yung. Curvature scale space corner detector with adaptive threshold and dynamic region of support. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2*, pages 791–794, Washington, DC, USA, 2004. IEEE Computer Society.

[12] W. Hu, T. Tan, L. Want, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 34(3):334–352, 2004.

[13] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[14] E. Kalman, Rudolph. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[15] K. Leach. A survey paper on independent component analysis. In *Proceedings of the Thirty-Fourth Southeastern Symposium on System Theory*, pages 239–242, 2002.

[16] T. Lebo. Guiding object recognition in video. Master's Thesis, Rochester Institute of Technology, 2005.

[17] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Proceedings of the Workshop on Statistical Learning in Computer Vision*, Prague, Czech Republic, May 2004.

[18] B. Leibe and B. Schiele. Interleaved object categorization and segmentation. In *British Machine Vision Conference (BMVC'03)*, pages 759–768, Norwich, UK, Sept. 2003.

[19] B. Leibe and B. Schiele. Scale invariant object categorization using a scale-adaptive mean-shift search. In *DAGM'04 Annual Pattern Recognition Symposium*, volume 3175, pages 145–153. Springer LNCS, August 2004.

[20] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision ICCV, Corfu*, pages 1150–1157, 1999.

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[22] A. M. McIvor. Background subtraction techniques. In *Proceedings of Image & Vision Computing New Zealand 2000*, Auckland, New Zealand, 2000. Reveal Limited.

[23] B. Neumann. Optical flow. *SIGGRAPH Comput. Graph.*, 18(1):17–19, 1984.

[24] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

[25] M. J. Tarr, H. H. Bl̈thoff, M. Zabinski, and V. Blanz. To what extent do unique parts influence recognition across changes in viewpoint? *Psychological Review*, 8(4):282–289, 1997.

[26] M. J. Tarr, P. Williams, W. G. Hayward, and I. Gauthier. Three-dimensional object recognition is viewpoint dependent. *nature neuroscience*, 1(4):275–277, 1998.

[27] A. Torralba, K. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection, 2004.

[28] S. Ullman and E. Sali. Object classification using a fragment-based representation. In *BMVC '00: Proceedings of the First IEEE International Workshop on Biologically Motivated Computer Vision*, pages 73–87, London, UK, 2000. Springer-Verlag.

[29] S. Ullman, E. Sali, and M. Vidal-Naquet. A fragment-based approach to object representation and classification. In *IWVF-4: Proceedings of*

the 4th International Workshop on Visual Form, pages 85–102, London, UK, 2001. Springer-Verlag.

[30] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition.* Dept. of Computer Science, University of Glasgow, 1979.