The Pennsylvania State University

The Graduate School

College of Engineering

DATA ASSOCIATION USING VIEW-BASED MODELS

A Thesis in

Electrical Engineering

by

Gaurav Vijaykumar Vaidya

© 2009 Gaurav Vijaykumar Vaidya

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2009
The thesis of Gaurav Vijaykumar Vaidya was reviewed and approved* by the following:

David J. Miller  
Professor of Electrical Engineering  
Thesis Advisor  

Robert T. Collins  
Associate Professor of Computer Science and Engineering  

Ken Jenkins  
Professor of Electrical Engineering  
Head of the Department of Electrical Engineering  

*Signatures are on file in the Graduate School
ABSTRACT

The problem of video tracking has been widely researched in the Computer Vision community. Identifying a target and keeping continuous track of it requires robust techniques which haven’t been definitely found yet. This thesis proposes a solution for the scenario where a tracker has had to reinitialize and needs to identify which among the potential objects of interest is the desired target. This is obviously an object recognition problem and the thesis proposes an object recognition approach. The proposed method uses an effective feature view clustering approach involving SIFT features to create a model of the desired target from multiple viewpoints. This model has the feature that it can be used to identify previously unseen poses of the target by interpolating between seen poses. Even if a new pose is far off from the seen poses, partial matches to existing poses are enough for good recognition. The contribution of this thesis is to apply this ‘object recognition in images’ technique to the video domain, and handling of the various issues that crop up due to this application. Experimental results show that the technique is effective in video-based object recognition and hence is a viable solution for the scenario mentioned above.
# TABLE OF CONTENTS

LIST OF FIGURES .......................................................................................... vi

LIST OF TABLES ........................................................................................... ix

Chapter 1 Introduction ...................................................................................... 1

1.1 Problem definition ...................................................................................... 1

1.2 Related work ............................................................................................... 1

1.3 Proposed approach ..................................................................................... 2

1.3.1 Failure modes ......................................................................................... 4

1.4 Contribution of this thesis ......................................................................... 5

Chapter 2 Data Association Using View-Based Models: Preprocessing Steps ...... 6

2.1 Outlier Removal Through Interframe Matching ....................................... 7

2.1.1 Color handling ....................................................................................... 8

2.1.2 Nearest neighbor keypoint matching ................................................... 11

2.1.3 Calculating affine transformation between consecutive frames .......... 12

2.1.3.1 The Hough transform ...................................................................... 13

2.1.3.2 Affine fit, outlier removal and top-down matching ......................... 18

2.1.3.3 Determining the winning affine transformation ............................ 22

2.1.4 Detection of tracker jump ................................................................. 23

Chapter 3 Data Association Using View-Based Models: Modeling and Recognition ........................................................................................................... 25

3.1 Modeling using local feature view clustering .......................................... 25

3.1.1 Linking of keypoints ............................................................................. 35

3.1.1.1 Implementation details ................................................................... 36

3.1.1.2 Special effect of inheriting keypoints while merging with a model view ......................................................................................... 37

3.1.2 Probabilistic verification ...................................................................... 38

3.1.3 Merging current frame keypoints with winning model view ............... 41

3.2 Recognition algorithm ............................................................................ 42

Chapter 4 Testing and results ......................................................................... 44

4.1 Data set for experiments .......................................................................... 44

4.2 Creating the modeling and testing sequences ........................................ 46

4.2.1 The modeling sequence ...................................................................... 46

4.2.2 The testing sequences ......................................................................... 50

4.3 Modeling and recognition ..................................................................... 53
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.1 Getting data ready for experimentation</td>
<td>54</td>
</tr>
<tr>
<td>4.3.2 Experimentation</td>
<td>56</td>
</tr>
<tr>
<td>4.4 Results</td>
<td>57</td>
</tr>
<tr>
<td>4.4.1 Short training sequence recognition</td>
<td>58</td>
</tr>
<tr>
<td>4.4.2 Long training sequence recognition</td>
<td>62</td>
</tr>
<tr>
<td>4.5 Analysis of failure modes</td>
<td>64</td>
</tr>
<tr>
<td>4.5.1 Incorrect recognition failures</td>
<td>65</td>
</tr>
<tr>
<td>4.5.2 Bad set of keypoints</td>
<td>67</td>
</tr>
<tr>
<td>4.5.2.1 Modeling for low-res data</td>
<td>69</td>
</tr>
<tr>
<td>4.6 Modeling complexity</td>
<td>70</td>
</tr>
<tr>
<td>Chapter 5 Conclusion</td>
<td>81</td>
</tr>
<tr>
<td>5.1 Future work</td>
<td>81</td>
</tr>
<tr>
<td>Bibliography</td>
<td>83</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Fig. 2.1: One frame from a video sequence showing overlaid SIFT vectors. (Clockwise from top left) Keypoints in the monochrome version, R plane keypoints, B plane keypoints and G plane keypoints. ........................................... 10

Fig. 2.2: Nearest neighbor matching result showing a spurious match (the diagonal line). ........................................................................................................ 12

Fig. 2.3: Voting for closest bins in Hough transform. Simplified example for a two dimensional Hough transform. The bins with centers closest to the match hypothesis are D, E, G and H. For two dimensions there are 2^2=4 such bins. For 4 dimensions there will be 2^4=16 closest bins. ............................................. 17

Fig. 3.1: Model hierarchy for each target in the feature view clustering approach ..... 26

Fig. 3.2: The situation at a point where n frames have been modeled.................. 32

Fig. 3.3: Frame n+1 comes in and its nearest neighbor matches are found for its keypoints. Each model view with a match receives a vote for it and links are traversed while doing so. Thus, even though existing model views 2, 3 and 4 each have one direct match, traversing the links makes the vote count as 1, 2 and 3 for model views 2, 3 and 4 respectively. ......................................................... 33

Fig. 3.4: The case where error of fit is greater than the threshold. Current frame keypoints are used to create the new model view 5 and keypoints which have matches with the winning model view 4 are linked to keypoints in that model view. Note that certain keypoints in model view 5 which had matches with keypoints in model views 2 and 3 are now linked to those keypoints in model view 4 which were linked to the matching keypoints in model views 2 and 3 ... 34

Fig. 3.5: The case where error of fit is lower than the threshold. The current frame keypoints are merged with the winning model view 4. Current frame keypoints which had matches in model view 4 inherit the links of their matches. .................................................................................................. 35

Fig. 3.6: Structure of the link data for each model view........................................ 36

Fig. 4.1: (On the left) The entire frame from the video sequence with superimposed bounding box from an off-the-shelf tracker (mean-shift based in this case). On the right is the extracted vehicle which is used as the frame for all modeling and testing. ............................................................... 45

Fig. 4.2: Target 1 in normal resolution (left) and low resolution (right). ............. 46
Fig. 4.3: Hierarchy of the modeling database ................................................................. 55

Fig. 4.4: Hierarchy of the recognition database .............................................................. 55

Fig. 4.5: Target 1 (recognition sequence, left) vs. target 3 (training sequence, right) ................................................................. 66

Fig. 4.6: Target 2 (recognition sequence, left) vs. target 4 (training sequence, right) ................................................................. 66

Fig. 4.7: Target 8 (recognition sequence, left) vs. target 7 (training sequence, right) ................................................................. 67

Fig. 4.8: Target 11 (recognition sequence, left) vs. target 10 (training sequence, right) ................................................................. 67

Fig. 4.9: (Clockwise from top left) Low-res frames of targets 1, 2, 4, 3 with superimposed keypoint locations for all three color planes ........................................... 68

Fig. 4.10: Modeling behavior with respect to addition and discarding of keypoints... 73

Fig. 4.11: Target 1 modeling behavior showing addition and discarding of keypoints ................................................................. 75

Fig. 4.12: Target 2 modeling behavior showing addition and discarding of keypoints ................................................................. 75

Fig. 4.13: Target 3 modeling behavior showing addition and discarding of keypoints ................................................................. 76

Fig. 4.14: Target 4 modeling behavior showing addition and discarding of keypoints ................................................................. 76

Fig. 4.15: Target 5 modeling behavior showing addition and discarding of keypoints ................................................................. 77

Fig. 4.16: Target 6 modeling behavior showing addition and discarding of keypoints ................................................................. 77

Fig. 4.17: Target 7 modeling behavior showing addition and discarding of keypoints ................................................................. 78

Fig. 4.18: Target 8 modeling behavior showing addition and discarding of keypoints ................................................................. 78
Fig. 4.19: Target 9 modeling behavior showing addition and discarding of keypoints ................................................................. 79

Fig. 4.20: Target 10 modeling behavior showing addition and discarding of keypoints ................................................................. 79

Fig. 4.21: Target 11 modeling behavior showing addition and discarding of keypoints ................................................................. 80
LIST OF TABLES

Table 2-1: Hough transform parameters for interframe matching.................................16
Table 2-2: Error thresholds used for outlier removal during affine fit. ..........................20
Table 3-1: Hough transform parameters for modeling................................................28
Table 3-2: Parameters for probabilistic verification......................................................40
Table 4-1: Poses covered during modeling and testing (in degrees approximate).............53
Table 4-2: ‘Short’ training sequence ‘seen’ recognition results......................................58
Table 4-3: ‘Short’ training sequence ‘seen+unseen’ recognition results..........................59
Table 4-4: Recognition failures for ‘short’ training sequence ‘seen – non-low-res’ recognition .................................................................................................................................59
Table 4-5: Recognition failures for ‘short’ training sequence ‘seen – low-res’ recognition .................................................................................................................................60
Table 4-6: Recognition failures for ‘short’ training sequence ‘seen+unseen – non-low-res’ recognition .................................................................................................................................60
Table 4-7: Recognition failures for ‘short’ training sequence ‘seen+unseen – low-res’ recognition .................................................................................................................................61
Table 4-8: ‘Long’ training sequence ‘seen+unseen’ recognition results............................63
Table 4-9: Recognition failures for ‘long’ training sequence ‘seen+unseen – non-low-res’ recognition .................................................................................................................................63
Table 4-10: Recognition failures for ‘long’ training sequence ‘seen+unseen – low-res’ recognition .................................................................................................................................64
Chapter 1

Introduction

1.1 Problem definition

The problem discussed in this thesis is to create a system for vehicle recognition that would come into play when a tracker (of any sort) loses the target or has little confidence in the current tracking. When the tracker is reinitialized, e.g. by segmenting out the vehicles based on motion, the proposed algorithm will perform the task of data association; determine which target gets which label.

1.2 Related work

This problem is the same as that researched in [7]. Just like this previous approach, the current approach uses SIFT keys [3, 5] as features for modeling. In [7] however, a complex SVM-based technique is used for modeling each target. The modeling was based on only an initial set of training frames and was carried out using the combined set of keypoints from all frames for all targets. Since there was no explicit attention towards target pose during modeling, the model was not effective for recognition beyond a certain pose range. The approach proposed here concentrates heavily on exactly this issue of pose and does so without the use of slow SVM-based techniques.
A very similar approach is used in [9]. Here, a model is created from multiple views of a target using a description based on Harris corners. However, this method doesn’t exactly perform classification and can only identify the target which was being tracked. SIFT keypoints are much more descriptive for matching across multiple viewpoints as compared to Harris corners because of the invariance to scaling, translation and rotation. Moreover, the associated scale, location and orientation of each keypoint make it easy to create model hypotheses, as will be shown later. The use of Harris corners instead of SIFT may not be suitable for the low resolution frames that are used for experimentation here. A dense set of Harris corners is definitely easier to extract than a dense SIFT set, but most of the Harris corners have very low ‘R’ values, indicating that they aren’t strong enough and may not be consistent across multiple frames.

[8] describes another approach towards object modeling and recognition from multiple views. This heavily depends on the training and test images being of high resolution and less cluttered, characteristics that most aerial imagery doesn’t possess. The modeling using partial model patches is also very slow, with numbers as high as 75 minutes to match a pair. The technique used here is much faster and uses a much simpler and straightforward approach towards multiple-view object modeling.

1.3 Proposed approach

The method used for target modeling closely follows the feature view clustering technique described in [4]. Enhancements have been made to this method to keep under
check the number of keypoints in the model. These enhancements will be described later in section 3.1.3. The basic approach is as follows. Initially, during the training phase, use the bounding boxes provided by a tracker to extract SIFT features over the target vehicle. Assuming that the tracker is tracking well, for each frame, the SIFT keys are used to create and update model views as described in [4]. Each frame either updates a particular model view, creates a new model view with features linked to existing model views or creates a new object sub-model altogether. The decision of which of these three tasks to perform is done based on comparing an error measure to a threshold. At the end of the training phase, each target model will consist of multiple sub-models which in turn are made of multiple model views with linked features, describing the target from different viewpoints.

One scenario during recognition could be that the tracker has lost its target and after some time, it has been reinitialized. However, it doesn't know the association of each target with its label. In this case, the same procedure used during modeling is followed for recognition. The SIFT features extracted from the bounding boxes are compared to all target models to give a matching model identifying the target. Apart from this scenario, when the tracker is tracking normally, the SIFT features can be used to update the target's model views. This way, new appearance information is constantly added to the models, adding to their robustness.
1.3.1 Failure modes

Two failure modes can potentially crop up in this implementation. The first relates to signature management. In [4], Lowe creates view-based models of objects using training images sampled at intervals of different viewpoints. However, when applied to each frame of video as is proposed here, the algorithm could continuously receive multiple training images of the same pose at a rate equal to the frame rate. This could lead to model views which are unnecessarily dense with keypoints, which would in turn increase the storage size of the database, thus affecting searching performance. Two techniques are used to avert this, one using Lowe’s method of discarding similar keypoints as described in [4] and another new technique proposed here based on the error of fit of a similarity transformation. The effects of these have been experimentally verified and will be described in later chapters.

The second failure mode would arise when, during recognition, the algorithm sees a totally unobserved pose of a known target. This would arise when the training phase covers only a small range of the possible viewpoints and the test image is from a viewpoint far away from these viewpoints. In this case, no match would be found to any model view of any target. The rate of occurrence of this event is minimized by the use of ‘linked’ model views and the fact that there is always some part of a vehicle which is common to any pair of viewpoints (e.g. the vehicle roof). Experimentation is carried out to see the recognition results on test poses which are far away from poses used for modeling and the overall results look promising.
1.4 Contribution of this thesis

Apart from the application of Lowe’s feature view clustering technique to the video context, the main contribution of this thesis is the modification in implementation to handle corner cases specific to video. An important addition is a new technique based on the similarity transformation error of fit to reduce the number of keypoints being added to the target model as modeling progresses. This greatly helps in keeping the model size and keypoint count under control and reducing search complexity during recognition.

Another contribution is the addition of a preprocessing step before the actual modeling. The video sequence frames that were used for experimentation were usually low resolution images with a lot of spurious keypoints being extracted. The preprocessing step eliminates these spurious keypoints and provides a relatively clean keypoint set for the modeling phase.

Special experimentation was carried out to test the characteristics of the modeling algorithm towards signature explosion, i.e. to see how effective it is in discarding keypoints from already seen poses and appearances and not adding them to the model. The testing database was designed such that the recognition behavior for poses not seen during modeling can be observed. Also, numerous metrics were tracked to exactly identify the cause of a failure, thus aiding in identifying any trends in the results.
Chapter 2

Data Association Using View-Based Models: Preprocessing Steps

The scale invariant method of extracting features described in [3, 5] can be effectively used for modeling targets for tracking. The problem of tracking, however, requires that one or multiple targets be tracked robustly despite major viewpoint changes. A global appearance model is useful in this case, one reason why color-based tracking is so effective. But a major argument against such models is that they ignore local features which could identify the distinctive structure of the target, which could be useful in discriminating it from other objects. Consider a scenario in which there are large changes in scale of a target in the video frame, possibly due to the camera zooming in on it. It could be that at one instance the target covers a minute area in the frame whereas at the next, it spans majority of the frame. A color model, for example, is prone to fail in this case, since now each part of the target could have a different appearance not accounted by the global model.

Local features like SIFT are capable of solving problems like above but only for a limited variation in pose [3, 5]. In order to employ them to create a truly global representation, multiple models must be created for each target from different viewpoints. These sets of view-based models for each target would provide both a global appearance model as well as local structure information, thus making the target distinctive at any scale and aspect. This technique forms the core of the algorithm discussed in this thesis. This chapter
details the preprocessing steps before the procedure for view-based data association, along with related design issues.

2.1 Outlier Removal Through Interframe Matching

The use of SIFT brings with it its own design issues. Targets extracted from video sequences for tracking often have very low resolution. The low contrast often results in very few keypoints being extracted on the target. Moreover, many of the keypoints are spurious, with no repeatability from one frame to another. It is obviously necessary to discard such keypoints before using the set to create a target model.

Preliminary observation of the keypoints extracted in these video sequences leads one to make the following definition:

**Consistent keypoints:** These keypoints are those which maintain roughly the same location, orientation and scale on the object over a wide range of frames. Still, they may not be present in every frame of this wide set and can appear and disappear momentarily. This is as opposed to spurious keypoints, which are random and are usually not observed again after their initial appearance. It is also observed that keypoints which are consistent in one pose of the object may not even exist in other poses, no matter how less the difference between the poses. There are very few keypoints which remain consistent over a wide range of poses. This is further motivation for creating view-based models of the objects to be recognized.
The scheme presented here for outlier removal is based on the following two facts. A good keypoint should be at least present in one of the two immediate neighbor frames of the current frame, and the locations of the set of good keypoints between two consecutive frames are related by an affine transformation. It should be noted that the temporal dependence of the scheme is limited only to the two neighboring frames, thus minimizing any effects of camera motion and bringing the problem as close as possible to simple object recognition. Not surprisingly, Lowe’s object recognition scheme [3, 5] can be employed for the task. The following sub-sections describe how this technique is applied to the outlier removal task. All steps from section 2.1.2 to 2.1.3.2 are directly from Lowe’s method. Since this is interframe matching with only a small change in viewpoint, if any, the parameters for object recognition are set to be more rigid in this case.

2.1.1 Color handling

Before moving on to the outlier removal scheme, one needs to decide how to handle the color information associated with each keypoint. The papers by Lowe [3, 4, 5] operate on monochrome images and do not explicitly mention of any technique to harness the color information. Methods for incorporating color with SIFT don’t seem to be readily available in the literature either, as authors either tend to ignore stating this issue or just operate on monochrome versions instead (e.g. [9] which uses Harris corners for vehicle modeling).
Color can be a very discriminative feature for targets, especially when the target structures are similar. Converting color images to grayscale for processing averages out all the variations in color, just leaving the intensity info. [8] describes a method to match feature vectors using color as an initial filtering tool. A 10x10 color histogram is extracted from a patch surrounding the keypoint. Only the UV portion of the YUV color space is considered and two histograms are compared using a $\chi^2$ metric. While finding keypoint matches, both the $\chi^2$ distance between the histograms as well as the Euclidean distance between the SIFT descriptors is measured. The SIFT descriptors here are found from a monochrome version of the image. Preference is given first to a lower $\chi^2$ distance and then to the Euclidean distance. Results mentioned there show that this is an effective measure in finding best match for color keypoints.

In our experiments, where the video stream is mainly characterized by low resolution, the following main observations can be made. As mentioned earlier, even consistent keypoints aren’t present in all frames. Also, as can be seen from fig. 2.1, there is a possibility that each color plane has a different number of keypoints. E.g. here, the vehicle in the frame has a major blue color component and as such, many more keypoints are extracted in the B plane than the R and G planes. Further, it can be seen that in the monochrome version, a lot of the keypoints extracted in the B plane (and even some major ones from the G plane) aren’t present. These lost keypoints could be valuable for model creation. Further, the only three keypoints that survive in the monochrome version can result in a very weak affine transformation relating this frame with the previous or
the next frame; whereas the B plane keypoints will provide in an overconstrained equation and thus a stronger affine relation.

Fig. 2.1: One frame from a video sequence showing overlaid SIFT vectors. (Clockwise from top left) Keypoints in the monochrome version, R plane keypoints, B plane keypoints and G plane keypoints.

From the above observations, a design decision is made in this approach to maintain the separation between keypoints from each color plane. Processes like finding matching keypoints will be carried out separately for each color plane. However, calculating global parameters like affine relations between consecutive frames will combine the keypoint data from all color planes and use this as one single set. These considerations will be elaborated further.
2.1.2 Nearest neighbor keypoint matching

After having decided on the color handling scheme, the next step towards outlier removal is to find matching keypoints between consecutive frames. At this first stage, the matching criterion is the smallest Euclidean distance between the SIFT descriptors. Specifically, for each keypoint from the current frame, the keypoint from the next frame with the closest descriptor is selected. To discard spurious matches, the distance ratio technique described in [3, 5] is used. For each current frame keypoint, the ratio of distances to the closest next-frame neighbor and the second-closest next frame neighbor is taken. If this ratio is above a certain threshold, the match is discarded. A distance ratio of 0.8 is used in the experiments at this stage.

As an added layer of filtering out spurious matches, a bidirectional best-match heuristic is used along with the above process. This is an approximate solution to the Linear Assignment Problem and has also been employed by the author in [9]. Basically, a match between two keypoints is accepted only if one keypoint is the best match for the other and vice-versa. This is a very strong condition for matching, and although it results in discarding a lot of matches, the remaining matches are generally very good.

It should be again noted here, that the matching is performed separately for each color plane. E.g. current frame keypoints from the R plane are matched with those from the R plane of the next frame. At the end of this procedure, the resultant data for each frame contains the following:
1. Keypoint descriptor and location information for each color plane

2. For each color plane, for each keypoint, index of the best match in the previous and the next frames. Keypoints with no matches have a best match index of ‘0’.

### 2.1.3 Calculating affine transformation between consecutive frames

Although the above procedure of nearest matching with bidirectional best-match heuristic gives pretty good matches, the whole set of matches doesn’t necessarily conform to any transformation between consecutive frames. E.g. fig. 2.2 shows a spurious match which clearly doesn’t follow the same transformation as the rest of the matches.

![Fig. 2.2: Nearest neighbor matching result showing a spurious match (the diagonal line).](image)

Secondly, and more importantly, keypoints on the background and those on the target have a slightly different transformation due to the relative motion. Also, sometime, parts of undesired targets are visible in the bounding box for the desired target. An effective method to distinguish between the background and spurious targets is to identify the transformation which relates the keypoints on the desired target between consecutive
frames. An affine transformation is used for this geometric verification task as it models to a good approximation the variation between consecutive frames.

2.1.3.1 The Hough transform

The Hough transform [2] is a fast technique to identify clusters of features bound together by a particular relation. It is especially effective because of its robustness and performance in presence of a higher ratio of outliers as well as due to the fact that it can detect multiple such clusters. Here, the nearest neighbor matching process previously described provides us with keypoint matches for each frame pair. Since each SIFT keypoint has embedded location, scale and orientation information, these matches can be used to create hypotheses for the next-frame pose of the object with respect to the current frame. In other words, for each match, these three parameters imply a similarity transformation between the current frame and the next frame. The Hough transform will be used to cluster together keypoint matches according to the similarity transformation they imply. Note that the similarity transformation is only an approximation to the affine transformation that actually relates the matches in the two frames. However, using broad error measures to cluster features according to similarity transformations is a good starting point before further refinement.

A similarity transformation can be represented in homogenous coordinates as shown in eq. 2.1.
\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} =
\begin{bmatrix}
    s \cos \theta & s \sin \theta & tx \\
    -s \sin \theta & s \cos \theta & ty \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]

(2.1)

Here, \( s \) is the scale factor, \( \theta \) is the change in orientation and \( tx \) and \( ty \) are the changes in \( x \)- and \( y \)-coordinates respectively. Note that this equation is written with respect to image coordinates, i.e. with the \( y \)-axis inverted. \( s, \theta, tx \) and \( ty \) are the parameters to be determined. The scale factor \( s \) can be determined by taking the ratio of the next-frame keypoint scale and the current-frame keypoint scale. Similarly, \( \theta \) is calculated simply by subtracting the current frame keypoint orientation from the next frame orientation. Having known \( s \) and \( \theta, tx \) and \( ty \) can be easily calculated. A summary of these calculations is given in eq. 2.2.

\[
\begin{align*}
    s &= s_{2} / s_{1} \\
    \theta &= \theta_{2} - \theta_{1}, \theta \in \left[-\pi, \pi\right] \\
    tx &= x' - s(x \cos \theta + y \sin \theta) \\
    ty &= y' - s(-x \sin \theta + y \cos \theta)
\end{align*}
\]

(2.2)

where \( s_{2}, s_{1} \) are the next-frame and current-frame keypoint scales respectively, and \( \theta_{2}, \theta_{1} \) are the respective orientations of the next-frame and current-frame keypoints.

In this step, each match can vote for a Hough transform bin according to these calculated values. Further matches from all color planes are combined for this voting process. However, the color information of each match is maintained and recorded when assigning matches to Hough transform bins. At this stage, keeping the bin sizes broad as mentioned in Lowe’s papers isn’t advisable. Since we are considering consecutive frames here, such broad variations aren’t expected. We still want to separate keypoints located
on the target from those on the background. However, again as a result of this consecutive frame comparison, there is only a very slight difference between the transformations relating the target keypoints and that relating the background keypoints. The result is that one has to experiment with bin sizes such as they aren’t so broad that keypoints on the target and those on the background aren’t differentiated, and neither are they so narrow that even keypoints on the target fall into different bins.

The setting of bin sizes in this step is somewhat crucial. We have to minimize the possibility that a background keypoint match falls in the same bin as a set of on-target keypoint matches. This is because, in the next step, an affine transformation will be fit to the matches in each bin. Due to the low resolution and small dimension of the vehicle bounding box images, we don’t really get a dense set of good keypoint matches. As a result, even a single background match present with good on-target matches can ruin the calculation of the affine transformation. The affine fit is calculated using least squares which is very sensitive to outliers.

The nature of keypoint matches between two consecutive frames needs a special note here. In two consecutive frames, there of course is relative motion between the target and the background. However, depending on the relative velocity between the two, this can be very small. Coupled with the inaccurate localization of keypoints due to the low resolution of images, this leads to difficulties in separating the two sets of keypoint matches. To put it in a different way, the background keypoints aren’t exactly outliers. They are just a different set of keypoint matches with a very slightly different affine
relation. The Hough transform, with its approximate similarity transformation, isn’t always able to distinguish such minute differences in affine relation.

For our purposes, the Hough transform bin sizes were set as shown in table 2-1.

Table 2-1: Hough transform parameters for interframe matching

<table>
<thead>
<tr>
<th>Scale bin size</th>
<th>Scale factor of 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle bin size</td>
<td>15 degrees</td>
</tr>
<tr>
<td>x location bin size</td>
<td>0.05 * (Scale factor) * (Reference image x dimension)</td>
</tr>
<tr>
<td>y location bin size</td>
<td>0.05 * (Scale factor) * (Reference image y dimension)</td>
</tr>
</tbody>
</table>

Different keypoint matches will hypothesize model locations based on different predicted scales. The dependence of the x and y location bin sizes on the predicted scale and a reference image dimension allows the comparison of such predictions on an equal footing. In this step of interframe matching, the reference image is arbitrarily chosen to be the one with the smallest area.

Voting for the Hough transform bins is done by creating a hash table. For a given keypoint match, first the bin to which it belongs to is determined. The ‘key’ for that bin is the index of the bin in each dimension (in this case, the 4 dimensions). This list of 4 bin indices is fed to a hash function, which gives the index of an entry in the hash table. The keypoint match is then assigned to this entry, i.e. the color, current frame keypoint index and next frame keypoint index are stored in the entry. Hash collisions are resolved by the chaining technique. Moreover, as mentioned by Lowe [3, 5], to avoid boundary effects in bin assignment, each keypoint match votes for the $2^n$ closest surrounding bins, where
`n` is the dimensionality of the Hough transform. For \( n = 4 \), this gives 16 votes per keypoint match (see fig. 2.3).

Fig. 2.3: Voting for closest bins in Hough transform. Simplified example for a two dimensional Hough transform. The bins with centers closest to the match hypothesis are D, E, G and H. For two dimensions there are \( 2^2 = 4 \) such bins. For 4 dimensions there will be \( 2^4 = 16 \) closest bins.

The hash function used in this implementation uses a simple bit masking technique. The function works by finding the power of 2 that the hash table size is. Then some number of bits are taken from the trailing edges of the 4 bin indices (represented as 8-bit unsigned integers), so that the total bits taken is equal to the power of 2 just found out. These bits are then concatenated to give the index into the hash table. Note that this hash function is
created just to generate a hash quickly and its efficiency with respect to collisions has not been investigated.

The hash table is a very efficient way to implement the Hough transform. It allows better use of memory as compared to creating a 4-D matrix for the bins. At the end of this procedure, each entry in the hash table typically consists of multiple bins (due to collisions). Each bin consists of the data of the matches that voted for that bin. Before moving on to the next step, the hash table ‘dechained’ and sorted. Basically, the hash table is restructured such that each of its entries now contains only one bin and then it is sorted in the descending order of bin votes.

2.1.3.2 Affine fit, outlier removal and top-down matching

After this preprocessing step involving Hough transform to segregate matches according to an approximate similarity transformation, the actual interframe affine fit is calculated using the matches in each of the bins. This section describes how that is done.

Starting from the bin with the most votes, an affine transformation is fit to the matches in that bin. An affine transformation can be described in homogenous coordinates as shown in eq. 2.3.

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} = \begin{bmatrix}
    a & b & c \\
    d & e & f \\
    0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]  

(2.3)
\(a, b, c, d, e, f\) are the affine transformation parameters to be determined. We need to have enough equations to solve for these six unknowns. Each keypoint match has four degrees of freedom due to the relative scale, orientation and location information. Thus, two matches should usually be enough. However, using scale and orientation to calculate the affine fit leads to a non-linear set of equations. So to simplify things, the scale and orientation information is dropped and instead a requirement of at least three keypoint matches is set. This gives us the required number of equations to solve for the six unknowns. If there are more matches, the system is overconstrained and a least squares solution is found. The system of equations can be written as in eq. 2.4.

\[
\begin{pmatrix}
x_1 & y_1 & 0 & 0 & 1 & 0 \\
0 & 0 & x_1 & y_1 & 0 & 1 \\
x_2 & y_2 & 0 & 0 & 1 & 0 \\
0 & 0 & x_2 & y_2 & 0 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{pmatrix}
\begin{pmatrix}
a \\
b \\
c \\
d \\
e \\
f \\
\end{pmatrix}
= 
\begin{pmatrix}
x'_1 \\
y'_1 \\
x'_2 \\
y'_2 \\
\vdots \\
\end{pmatrix}
\]

(2.4)

The least squares solution \(x\) can be written as in eq. 2.5.

\[x = [A^TA]^{-1}A^Tb\]  

(2.5)

After calculating the affine transformation, the next step is to refine the set of keypoints that was used for the affine fit. This means removing those keypoints that do not conform to the calculated affine transformation and adding new keypoints that do. Removal of outlier keypoints is necessary since we have used the approximate similarity transformation to cluster these keypoints during the Hough transform step. For the same
reason, some matches may have been discarded at that stage, and they need to be added now.

The outlier removal process is done by first applying the calculated affine to the current frame keypoints in the bin. The resulting predicted keypoint locations are compared to the actual locations of the corresponding keypoints in the next frame. The error in scale, angle and location is then compared with a threshold to decide whether the match conforms to the calculated affine. The predicted scale and angle is found by representing each current frame keypoint (i.e. the SIFT vector) in terms of its head and tail location. The affine transformation is then applied to these two location values per keypoint, and the predicted scale and angle are found by taking the length of the predicted SIFT vector and its orientation.

The thresholds used at this step are set to half the bin sizes used during the Hough transform step. Thus, for the values in table 2-1, the thresholds used here are shown in table 2-2.

Table 2-2: Error thresholds used for outlier removal during affine fit.

<table>
<thead>
<tr>
<th>Scale bin size</th>
<th>Scale factor of √2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle bin size</td>
<td>7.5 degrees</td>
</tr>
<tr>
<td>x location bin size</td>
<td>0.025 * (Predicted scale) * (Reference image x dimension)</td>
</tr>
<tr>
<td>y location bin size</td>
<td>0.025 * (Predicted scale) * (Reference image y dimension)</td>
</tr>
</tbody>
</table>

For the scale bin size, instead of half the Hough transform bin size, the threshold used is √2 times the bin size. Note that errors are compared in absolute terms, i.e. only their magnitude is used. For this reason, if dividing by √2 gives a scale bin size less than 1,
then its reciprocal is taken so that the scale error threshold is always greater than 1. This is done because there cannot be a negative error in scale factor; it can be either less than or greater than 1.

Those matches for which the next-frame keypoints are beyond the error threshold are discarded as outliers. Then, the same affine transformation is used to identify matches which do not belong to the current Hough transform bin but still conform to the calculated affine. This is done by first applying the affine transformation to the keypoints outside the current bin. The scale, orientation and location of each predicted keypoint is compared to that of each next-frame keypoint which is not already classified as an inlier. If a next-frame keypoint is close within the above thresholds (in terms of all scale, angle and location) to a predicted keypoint, then it is classified as ‘conforming’. Finally, all such new conforming keypoints are added to the set of the original inliers and the affine transformation is recalculated using this set.

The above procedure of outlier removal and top-down matching is repeated until there is no change in the set of keypoint matches, or until a limiting number of iterations is reached. It is important to note that in this step, all processing is done separately on each color frame, except for the affine calculation. Thus, predicted keypoints of R plane are compared only to actual next-frame keypoints of R plane and so on. Once the set of keypoint matches stabilizes and the affine transformation for this set is determined, these keypoints are marked as ‘matched’. The whole procedure of affine fitting is then repeated on the Hough transform bin with the next highest number of votes, until all bins with >3
keypoint matches are covered. However, keypoints that have already been marked as ‘matched’ are not considered in these subsequent iterations. The affine transformations calculated in each of these iterations are stored separately along with the corresponding matches, to be used in the next step.

2.1.3.3 Determining the winning affine transformation

At the end of the affine fit, outlier removal and top-down matching phase, what we get is a collection of affine transforms and associated keypoint matches. Typically in a video frame, there are multiple clusters of keypoints, each associated with an ‘object’ in the frame. ‘Object’ here can refer to the desired target, the background, or any other spurious target or structure present within the frame. Each of these ‘objects’ can have keypoint matches which have their own independent affine transformation. The previous phase of affine fitting will typically give all these affine transformations and the matches that conform to them. We need to determine which among these clusters of matches belongs to the desired target, so that this cluster can be used in the next big step of target modeling. Also, the affine transformation that relates the matches in that cluster needs to be known. This affine relation is called the ‘winning’ transformation, and the reason for that will become clear shortly.

The procedure to determine the winning transformation is a simple voting-based approach. Initially, it is assumed that for the first frame, the winning transformation is known. This could be the transformation that contains keypoints located on the desired
target. Keypoints belonging to this transformation are noted. For the next frame, after applying the affine fit procedure, we have a set of affines and the corresponding next-frame keypoints. Then, each keypoint in the winning cluster in the first frame votes for that affine in the next frame, which corresponds to the next-frame match of this keypoint. At the end of this voting, the next frame affine relation that has the most number of votes ‘wins’ and is used as the reference for repeating this process on subsequent frames. The motivation behind this procedure is to roughly determine the next-frame locations of the current frame winning keypoints, and hence the target. If none of the next frame affine relations get any votes, this could mean that the target that is being tracked isn’t present in the next frame. Such a situation can be handled by signaling to the tracker to reinitialize and redetect the target.

2.1.4 Detection of tracker jump

An interesting byproduct of calculating the affine fit and the winning affine transformation is that it can be used to detect when a tracker jumps from the desired target to another object. A method to do that can be as follows. The winning affine transformation is applied to the current frame bounding box to get a projected bounding box. This projected bounding box is then superimposed on the next-frame bounding box and the amount of overlap is calculated. If the area of overlap is below a certain threshold, then this could mean that the next-frame bounding box isn’t covering the target entirely and that the tracker is drifting.
Note that this technique cannot be applied to detection of tracker drift, which usually happens slowly. This is because, if the bounding box is drifting very slowly off the target, there is a high chance that some new conforming keypoints not located on the target may get added to the winning affine cluster of matches. Once these new keypoints get added and their number grows, the current procedure won’t sense tracker drift so long as these keypoints are present within the bounding box, irrespective of whether the box still encompasses the target or not. A slight change can be done for possibly detecting tracker drift. By using the projection of the convex hull of the current-frame winning affine keypoints instead of the whole current-frame bounding box, the threshold for overlap can be reduced further, thus leading to earlier detection of slow drift. This is due to the smaller area of the convex hull as compared to the bounding box.
Chapter 3

Data Association Using View-Based Models: Modeling and Recognition

The previous chapter described the essential preprocessing steps before creating view-based models of each target. After obtaining a set of outlier-free keypoints per frame, we are now ready to use them for modeling. The heart of this process is Lowe’s local feature-view clustering algorithm [4]. Although Lowe has described this algorithm in the context of object recognition from fixed images, it has been applied here for video object recognition. Although it is perfectly logical to do so, there are certain aspects like signature explosion and database size that are more pronounced in the context of applying it to video. These points will be described as we move on through the chapter. Lastly, after modeling has been done, this chapter describes how the models are used to recognize targets.

3.1 Modeling using local feature view clustering [4]

As discussed in the initial paragraphs of the previous chapter, creating view-based models has many advantages towards robust recognition. In this feature view clustering approach each target has a model hierarchy associated with it, as shown in fig. 3.1.
Fig. 3.1: Model hierarchy for each target in the feature view clustering approach

Here, each target consists of multiple sub-models, which in turn consist of ‘linked’ model views. Each of these hierarchy components will now be explained.

Modeling of a particular target is started with the first frame of the training sequence. Before this, the preprocessing steps to remove outlier keypoints described in the previous chapter are performed on the whole training sequence. Such a clean set of keypoints for the first frame forms the initial model view of the target. As we proceed, multiple such model views will form one sub-model of the target. For the second frame onwards, we need to decide how to integrate the keypoints in this frame with the target model. This could involve either of merging the second frame keypoints to the model view, creating a new model view, or creating a new sub-model altogether.
The general modeling algorithm is described below. This is implemented exactly as described in [4] and uses all the steps. Certain aspects such as linked model views, merging of keypoints and probabilistic verification will be explained in detail later.

1. Initialize the target model with keypoints from the first frame. At this point, the model hierarchy is just ‘target model’->’sub-model 1’->’model view 1’, where model view 1 is the set of keypoints from the first frame.

2. From second frame onwards, do the following for each frame:

   a. Find best match for each keypoint in the whole model. That is compare with each keypoint in every model view of every sub-model of the target. Note that at this stage, only keypoints of the same color are compared. The distance ratio criterion is to be used here to determine if the nearest match is to be kept or discarded. This criterion mentions that a match is to be kept if the ratio of distances to the nearest match and that to the second-nearest match is less than a threshold. A value of 0.8 is used here.

   b. Perform Hough transform on the matches found. From this step onwards until step (g) below, all processing is done on the combined set of keypoints from all colors. This categorizes each match hypothesis into separate bins based on the relative scale, orientation and position of the two matching keypoints. Refer to section 2.1.3.1 for details on this. In this step we use broader bin sizes to allow for greater variation in the poses clustered. The parameters used are the same as in [4] and are listed in table 3-1.
Table 3-1: Hough transform parameters for modeling

<table>
<thead>
<tr>
<th>Scale bin size</th>
<th>Scale factor of 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle bin size</td>
<td>30 degrees</td>
</tr>
<tr>
<td>( x ) location bin size</td>
<td>( 0.2 \times \text{(Scale factor)} \times \text{(Current frame ( x ) dimension)} )</td>
</tr>
<tr>
<td>( y ) location bin size</td>
<td>( 0.2 \times \text{(Scale factor)} \times \text{(Current frame ( y ) dimension)} )</td>
</tr>
</tbody>
</table>

c. Now it is possible that there were simply no matches found, or there was no Hough transform bin with \( > 2 \) matches. We are going to fit a similarity transformation to these matches later. Since each match has four degrees of freedom because of the relative scale, orientation and location, one keypoint is enough to calculate the similarity fit. However, to add some robustness, a requirement of at least 2 matches is imposed. In the case of less than 2 matches, we simply create a new sub-model using all the keypoints from the current frame. Alternately, if enough matches are found, we can move to the next step.

d. Consider the Hough transform bin with the maximum number of matches. Each match has one keypoint coming from one of the model views in the target model. This information is used to vote for model views to find the winning model view to which most of the matches in that Hough transform bin belong. This process is a bit more involved due to the consideration of ‘links’. Details of linking are given in section 3.1.1, but a brief introduction follows.

During creation of the model, each keypoint in a model view can be ‘linked’ to other keypoints in another model view under the same sub-
model. These are usually keypoints belonging to the same feature on the
target, but viewed from a different pose. So while voting for the model
views, these links are traversed, so that a model view containing a
keypoint which is linked to a keypoint in the match also gets a vote from
that match.

e. Fit a similarity transformation to the matches in the highest Hough
transform bin belonging to the winning model view.

f. Probabilistic verification: The inverse of the calculated similarity
transformation is used to transform keypoints in the model view to the
coordinates of those in the current frame. Probabilistic verification, as
described in [4], is then carried out on this set of keypoints. The details of
this are described in section 3.1.2. If probabilistic verification fails, the
whole set of keypoints from the current frame are used to create a new
sub-model of the target. Alternately, if verification passes, we move on to
the next step.

g. From step (e) above, an error of fit can be obtained during calculation of
the similarity transformation. This error of fit is compared with a threshold
and a decision is made as to whether the current frame keypoints are to be
merged with the winning model view, or whether a new model view is to
be created.
The error of fit used here is the average distance between the projected current frame keypoints and the model view keypoint matches. As in [4], the expression of error of fit is as shown in eq. 3.1.

\[ e = \sqrt{\frac{2\|Ax - b\|^2}{r}} \]  

(3.1)

Here, the variables have the same meaning as in eq. 2.3 to 2.5 and follows from the same analysis, except that it is applied to the similarity context of eq. 2.1. ‘r’ is the number of rows in the matrix A. The error expression is basically the square root of the sum of squared distances divided by the number of matches, hence the factor 2 in the numerator.

This error is compared to a threshold \( T \), which is set to be 5% of the maximum dimension of the current frame.

h. If \( e > T \), a new model view is created using the current frame keypoints under the current sub-model (the sub-model to which the winning model view belongs). Here, keypoints from different color planes are kept separate. This involves ‘linking’ each current frame keypoint to its match in the winning model view, if such a match exists. The details are in section 3.1.1, but simply put, a link just consists of the serial number of the linked keypoint in its model view. Thus, the link data of matching keypoints in the winning model view also have to be updated with this new current frame keypoint information.
i. On the other hand, if $e < T$, the current frame keypoints are to be ‘merged’ with the winning model view. Again, color planes are kept separate, i.e. keypoints from the R plane in the current frame are merged with the R plane keypoints in the winning model view. This basically involves the following steps:

   i. Transform all current frame keypoints to the winning model view coordinates using the similarity transformation already calculated.

   ii. For all matches between the model view and current frame, discard all new keypoints which are very similar to existing ones. There are two techniques which are employed here that are used to discard similar keypoints and are described in section 3.1.3. This step is very important to keep the model keypoint count and hence the search complexity during recognition under control.

   iii. All new current frame keypoints that are added and have a match in the winning model view inherit the links of their matches. Also, link data of the already existing keypoints they are linking to is updated.

   j. Repeat steps (2a) to (2i) until all frames are processed.

Figures 3.2 to 3.5 explain the above steps of merging vs. new model view creation for a simple case where the target model contains just one sub-model.
For frame n

Fig. 3.2: The situation at a point where n frames have been modeled.
Fig. 3.3: Frame n+1 comes in and its nearest neighbor matches are found for its keypoints. Each model view with a match receives a vote for it and links are traversed while doing so. Thus, even though existing model views 2, 3 and 4 each have one direct match, traversing the links makes the vote count as 1, 2 and 3 for model views 2, 3 and 4 respectively.
Case $e > T$:
Create new model view with linked features.

Fig. 3.4: The case where error of fit is greater than the threshold. Current frame keypoints are used to create the new model view 5 and keypoints which have matches with the winning model view 4 are linked to keypoints in that model view. Note that certain keypoints in model view 5 which had matches with keypoints in model views 2 and 3 are now linked to those keypoints in model view 4 which were linked to the matching keypoints in model views 2 and 3.
Case $e < T$: Combine new frame keypoints with winning model view

Fig. 3.5: The case where error of fit is lower than the threshold. The current frame keypoints are merged with the winning model view 4. Current frame keypoints which had matches in model view 4 inherit the links of their matches.

3.1.1 Linking of keypoints

The linking of keypoints is an important aspect of Lowe’s algorithm. The main reasoning behind linking is that certain target features are visible in multiple poses. Each pose will typically be represented by a model view in the target model. If there was no linking, an intermediate pose containing some of these common target features would vote for multiple model views, thus dispersing the votes. Linking keypoints corresponding to these common target features across model views and traversing them during voting
allows the accumulation of model view votes in one particular model view rather than dispersing them among multiple model views. Thus a new pose intermediate to poses represented by existing model views can be matched and identified. The result of linking is that the target model gets some fuzzy characteristic with respect to pose. Although each model view represents a discrete pose, the links among them create sort of an ensemble which interpolates the discrete poses and represents all the intermediate poses.

3.1.1.1 Implementation details

The link structure for a model view is a very simple one as shown in fig. 3.6.

![Diagram showing the link structure for a model view]

Fig. 3.6: Structure of the link data for each model view

Each keypoint in each color plane in the model view has an associated 2-D array which contains a set of pointers to other keypoints in other model views that this keypoint is linked to. Thus, when we have to ‘traverse’ links for voting model views, the link data for each keypoint is seen, the model view and index of the linked keypoint is read, the model
view indicated gets one vote and the process is repeated for the indicated keypoint in that model view. Now if keypoint 1 is linked to keypoint 2, both have pointers to each other in their link data. So while traversing links, care is taken not to follow a link backwards. This is done by temporarily discarding the pointer to the ‘calling’ keypoint from the link data of the keypoint currently being processed.

During creation of new model views, the keypoints in the current frame get linked to their matches in the winning model view. This is implemented simply by adding pointers to the winning model view keypoints in the link data of the current frame keypoints. During merging of model views, current frame keypoints which are added to the winning model view inherit the links of their matches. This is done by copying the link data of the winning model view keypoints to the link data of the new keypoints added. Note that in both the above cases, the link data of the keypoints that the current frame keypoints link to must also be updated. Recall that if keypoint 1 and keypoint 2 are linked, each has a pointer to the other.

3.1.1.2 Special effect of inheriting keypoints while merging with a model view

While merging current frame keypoints with a model view, all new keypoints inherit the links of their matches. An effect of this inheritance is that the model view could get a special bias during voting from now on. Refer fig. 3.5 for an explanation. Consider one of the two new keypoints which inherit links and are now linked to keypoints in model view 3. If a new frame comes in and a new keypoint matches one of these keypoints, model
view 3 will get one vote because of the linking. This behavior is normal. However, what if the new keypoint matches one of those model view 3 keypoints? These model view 3 keypoints are now linked to two keypoints in model view 4. Thus, model view 4 gets two votes even though both these model view 4 keypoints are similar. Hence the bias.

Although this could lead to model view 4 incorrectly winning the vote, getting multiple votes for a feature isn’t drastically wrong. Here’s why. A new keypoint is added to a model view only if it has new information. Thus, even though the two model view 4 keypoints are similar, they do have some difference in information. In fact, this difference in information could be better than having just one model view 3 keypoint, as it can match variations in appearance of that feature. In this case it is actually better to use model view 4. Considering this argument, no special effort is taken to handle this case in the implementation presented here. Also, the actual occurrence rate of this case and its effect on modeling will have to be experimentally verified as a part of future work.

3.1.2 Probabilistic verification

The use of similarity transformation and the wide bin sizes in Hough transform is advantageous in a way since greater variations in pose can be accommodated. However, this also increases the chances of false recognition. To deal with this, a novel probabilistic verification procedure is described in [4]. This estimates the probability that the model view is present in the frame given the set of keypoint matches obtained.
It can be written as $P(m|f)$, where $m$ is the winning model view and $f$ is the set of $k$
keypoint matches found between the current frame and the winning model view. The
analysis to find $P(m|f)$ given in [4] leads to the expression given in eq. 3.2.

$$P(m|f) \approx \frac{P(m)}{P(m) + P(f \not| m)} \tag{3.2}$$

Here,

$P(m)$ = A priori probability that the model view is present in the current frame.

$P(f \not| m)$ = Probability that the set $f$ of keypoint matches was found even if the model view
wasn’t present.

It can be seen that $P(m|f)$ is inversely related to $P(f \not| m)$. Thus, we have a higher
certainty in the matches if the probability of random appearance of the set of matches $f$
is very low. The expression for $P(f \not| m)$ is as shown in eq. 3.3.

$$P(f \not| m) = \sum_{j=1}^{n} \binom{n}{j} p^j (1 - p)^{n-j} \tag{3.3}$$

Here,

$n$ = Number of current frame keypoints present within the convex hull formed by back-
projecting the winning model view keypoints using the similarity transformation found.

This represents the number of keypoints that can lead to false matches.

$p$ = Probability that a current frame keypoint accidentally matches with a keypoint from the
winning model view.

$k$ = Number of matches between current frame and winning model view.

$p$ can be calculated using the expression given in eq. 3.4.

$$p = dlr$s \tag{3.4}$$
where,

\( d = \) Probability of randomly selecting the matching model view keypoint from the entire database. This is simply the ratio of number of keypoints belonging to the winning model view to the number of keypoints in the whole database.

\( l = \) Probability of matching the position constraint.

\( r = \) Probability of matching the orientation constraint.

\( s = \) Probability of matching the scale constraint.

The factors \( l, r, s \) are required because, along with randomly selecting a winning model view keypoints, the two keypoints in the match should also have the right combination of relative position, orientation and scale to be categorized into the highest voted Hough transform bin.

Since we are using the same Hough transform parameters as in [4] (refer table 3-1), we use the same values for \( l, r \) and \( s \), as shown in table 3-2.

**Table 3-2: Parameters for probabilistic verification**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L )</td>
<td>0.04</td>
<td>Location is constrained to 20% of the current frame size along each dimension. Thus, ( 0.2^2 = 0.04 ).</td>
</tr>
<tr>
<td>( R )</td>
<td>0.085</td>
<td>Orientation constrained to 30 degrees, hence ( 30/360 = 0.085 ).</td>
</tr>
<tr>
<td>( S )</td>
<td>0.5</td>
<td>Scale is constrained to a factor of 2.</td>
</tr>
</tbody>
</table>

Following the reasoning in [4] for the a priori probability \( P(m) \), it is calculated by assuming that one of the matching keypoints is used to determine the model view. Thus
\(P(m)\) is the probability that this keypoint is a correct match. This is just the ratio of correct matches to all matches between the current frame and the model view. In our application since the target spans most of the frame, this ratio is high enough. A value of 0.5 is used as an estimate for this.

The final decision of whether to accept the model view hypothesis is done by thresholding \(P(m|f)\). A threshold of 0.95 is used here, just as in [4].

It may seem that if this verification fails, i.e. \(P(m|f) < 0.95\), then the model view hypothesis ought to be discarded. However, since this is being used in the modeling process, we are sure that the current frame contains a view of the target being modeled. Probabilistic verification failure typically indicates that too few a matches were found compared to the expanse of the target in the frame (i.e. variables \(n\) and \(k\) in eq. 3.3). This could be due to various reasons including an as yet unseen pose, a change in appearance due to illumination, etc. So if verification fails, the current frame keypoints are used to create a new sub-model to account for these variations.

### 3.1.3 Merging current frame keypoints with winning model view

When the fitting error \(e\) is less than the threshold \(T\), the current frame keypoints are merged with the winning model view. However, since this is a video context, a new set of keypoints comes in every new frame. If all new frame keypoints were to be added to the model view, it could lead to a linear growth in the keypoint count of the model as
modeling progresses, thus leading to a very high search complexity during the recognition phase.

To avoid this, current frame keypoints which are very similar to existing model view keypoints are not added to the model view. To do this, for every current frame keypoint with a match, the second closest match is found in the model view. If, for a keypoint, the ratio of the distances to the closest match and that to the second closest match is less than 1/3, the keypoint is discarded. Else it is added to the model view.

During experimentation it was observed that this criterion wasn’t enough to control the growth of keypoints in the target model. Hence an additional technique was used to discard keypoints. This involved comparing the ratio of similarity transformation fitting error $e$ and threshold $T$ to another threshold $S$. If $(e/T) < S$, it is assumed that the new frame holds very little new information and all keypoints of this frame are discarded. A threshold of $S = 0.3$ was used in experiments. As will be described in section 4.6, this criterion was pretty effective in controlling the growth of keypoints without affecting recognition results.

### 3.2 Recognition algorithm

The recognition algorithm is simply a truncated version of the modeling algorithm. All steps described in section 3.1 up to the probabilistic verification stage are used during recognition. At the end of the modeling process, what we have is a target model structure
for each target as shown in fig. 3.1. The recognition algorithm uses this and operates on a
given test frame and the keypoints extracted from that frame. A quick algorithm listing is
as follows:

1. For each keypoint in the test frame a nearest neighbor match is found by
   comparing with all the keypoints in all the target models. Thus we have a match
   for each test frame keypoint from one of the target models.
2. Perform Hough transform over these matches found. Use the same parameters as
given in table 3-1.
3. For the highest vote Hough transform bin, find the target - sub-model - model
   view combination with the greatest number of votes from matches.
4. Fit a similarity transformation to the matches in the highest vote Hough transform
   bin belonging to the winning target - sub-model - model view combination.
5. Perform probabilistic verification on this set of keypoints.

If probabilistic verification passes, we know that the test frame matches with the winning
target - sub-model - model view combination. The target number from this combination
is returned as the recognized target. If any of the above 5 steps fail, either because of very
few matches found or verification failure, a ‘no decision’ result is returned.
Chapter 4

Testing and results

This chapter will describe the experiments carried out to verify the working of the proposed approach. The chapter has been divided into sections introducing the data set, creation of modeling and training sequences and then the results. An analysis of the results follows after that.

4.1 Data set for experiments

Video sequences from the VIVID tracking testbed [1] were used for all experiments. These videos consist of aerial recordings of multiple vehicles on the ground. Target vehicles are visible in a wide range of poses, from different viewing angles relative to incident light, as well as at different distances from the camera, which affects their apparent size relative to the frame dimension. The result of so much available variation is that the behavior of the algorithm can be observed not only with respect to pose change, but also illumination and slight color change, and spatial dimensions.

The presented algorithm is meant to operate on top of a tracker. The testing scenario here is that a tracker has lost track of its target, but has regained control and managed to segment out potential targets from the frame. The tracker now should correctly identify which among these is the target it was initially tracking. All experimentation was thus
carried out by first extracting bounded boxes around vehicles present in the video sequences. These bounded boxes are outputs of a typical off-the-shelf tracker and contain only the vehicle to be tracked. An example is shown in fig. 4.1. Bounded box data for most targets was obtained by the author of [7] as part of that research and used here directly. Additional bounding box sequences for some more targets were created by this author using a modified version of the Matlab tracker available as part of the VIVID Tracking Testbed available from www.vividevaluation.ri.cmu.edu.

Fig. 4.1: (On the left) The entire frame from the video sequence with superimposed bounding box from an off-the-shelf tracker (mean-shift based in this case). On the right is the extracted vehicle which is used as the frame for all modeling and testing.

So to begin with, what we have is a set of bounding box video sequences for each vehicle, where the vehicle can be seen in different poses, sizes (due to distance from camera) and illumination. In sequences where the target is far away from the camera, the
target is visible in low resolution because of its small size. An example is shown in fig. 4.2, which shows the normal resolution and low resolution versions of the same target vehicle, obtained from two different video sequences. This video sequence has to be split into sequences for modeling (training) and recognition (testing). The approach taken for doing this is described in the next section.

![Fig. 4.2: Target 1 in normal resolution (left) and low resolution (right).](image)

### 4.2 Creating the modeling and testing sequences

#### 4.2.1 The modeling sequence

A good model is one that can be used to recognize the target under varying poses, scale and illumination. To create such a model, the modeling sequence should be able to ‘teach’ the modeling algorithm something about how the vehicle looks like under these conditions. Hence, here, the objective while creating modeling sequences was to include enough variation in pose and appearance of the target. The steps and considerations while creating the modeling sequences are listed below:
1. First, all available video sequences were manually observed and classified into 11 targets.

2. Then, each video sequence was manually observed and a log was created of which target it belongs to, the appearance of the target in the sequence and the range of poses covered in the sequence. This data was needed to determine which frames to include in the modeling sequence. Here, pose was noted as the angle that the front of the vehicle made with the imaginary X-axis. Note that all vehicles were observed from the top since the video sequences were all aerial views. Pose estimation was manual and crude, so an error of +/- 10 degrees is expected. There wasn’t a need to know this exactly, and the reason is explained in a subsequent point.

3. Secondly, each video sequence was classified into ‘normal resolution’ and ‘low resolution’, based on the appearance of the target. If the features on the target vehicle were clearly visible, the sequence was classified as ‘normal resolution’, else ‘low resolution’. There were no sequences in the database which had a normal resolution view in some of the frames and a low resolution view in the rest. This made classification even easier. At the end, the modeling sequence for each target had a mixture of low-res and normal-res frames, so that the target model should contain variations in appearances.

4. After knowing the target, pose range and resolution information for each video sequence, a modeling sequence was created. This consisted of ‘stitched’ frame sequences, each of which showed a particular pose of the target. E.g. If the video sequence for a target had the following description from step (2) above:
Target 1: Frames 1-220: Car pose makes an angle of 20 degrees with X axis
  : Frames 220-480: Car pose is roughly parallel to X axis
  : Frames 480-839: Car pose makes an angle of -20 degrees with X axis
Then the modeling sequence would consist of, say, frames 100-150, frames 250-270, frames 500-600, stitched into one straight sequence. The goal was to choose these individual components such that the vehicle pose remains roughly constant during the duration of that component.
Now we needed a variety of poses to be modeled. However, the goal of testing was also to examine the behavior of the algorithm towards unseen poses, i.e. whether the algorithm can ‘interpolate’ or ‘extrapolate’ already seen poses to recognize unseen poses. Towards this goal, modeling poses were chosen such that there was a maximum of 20-30 degree difference between consecutive modeling poses as well as between a modeling pose and any testing pose, as far as possible. This was a rough demarcation of poses, and as such the rough pose estimation in step (2) was enough for this.
5. During modeling, the effect of adding new keypoints to the model with each new frame has to be tested. Since this is an application of Lowe’s method to the context of video, the number of keypoints in the target model can quickly grow out of hand, leading to increased search complexity during recognition. This is the problem of signature explosion. To see how modeling handles this, two categories of modeling sequences were created – a ‘short’ sequence and a ‘long’ sequence. Two different target models were created using these two modeling sequences.
The ‘short’ sequence consisted of exactly 20 frames of each pose chosen in step (4). Now there were certain video sequences where, for an extended duration, the same pose was visible. An example is the description for target 1 listed in step (4). Here, the target pose remains roughly the same for the three sets of frames. The ‘long’ sequence was created using such constant-pose sequences provided that some frames of the sequence were used for the ‘short’ sequence as well. Effectively, the short sequence was a subset of the long sequence, with the long sequence having no new poses than the short sequence, just longer durations for each pose. Of course, there was always the case that the long modeling sequence is the same as the short one, due to unavailability of constant-pose sequences. The way the long sequence was useful is that, at the end, it can be observed whether or not the algorithm is adding new keypoints to the model even if it is seeing an already seen pose, and also whether the long sequence models have too many keypoints for the recognition phase to handle.

Thus, at the end we have two modeling sequences – a ‘short’ and a ‘long’, each containing multiple poses for each target as well as a mixture of normal-res and low-res frames. A separate target model was created for each target using these two sequences, so that performance for both can be observed.
4.2.2 The testing sequences

All remaining video sequences which were not utilized for modeling were used for testing. During testing, the recognition algorithm received an individual frame and associated keypoint data. As such, the concept of video ‘sequence’ isn’t really applicable to the testing sequences. Having said so, the pose and resolution of the vehicle in each frame was still known (to an estimate) from step (2) in the previous section. Although all frames not used for modeling were clubbed together for testing, certain categorization was made, both because of the way the modeling sequences were created as well as for gathering performance metrics.

The first demarcation was made between normal-res and low-res frames. Recognition was performed separately on groups of frames depending on their resolution, i.e. separate numbers were obtained for sets of normal-res frames and sets of low-res frames. Secondly, a demarcation was made between ‘seen’ poses and ‘seen + unseen’ poses. This has to do with creation of separate ‘short’ and ‘long’ training sequences. Note that the ‘long’ sequence is a superset of the ‘short’ sequence with no new poses, just with longer durations of already existing poses. Those frames which were present in the ‘long’ sequence but not present in the ‘short’ sequence were clubbed together to form a ‘seen’ testing sequence, obviously because all these poses were already seen during modeling. All other frames from all other video sequences, i.e. the complementary set of the ‘long’ sequence, constituted the ‘seen + unseen’ testing sequence, because it contained a mixture of both types of poses.
The reason for having a ‘seen + unseen’ sequence as opposed to an only ‘unseen’ sequence was that it was too tedious to filter out all seen poses from the video sequences manually. Most of these sequences are up to 2000 frames long, as is listed later, and have a continuous variation in pose. This is exactly why keeping the seen poses is not expected to skew the performance numbers, since seen poses are only a discrete set which were hand-picked for modeling (see step (4) in previous section), whereas the testing sequence has a continuous series of poses.

The ‘seen’ testing sequence is useful as a ‘verification’ set to verify that the modeling and recognition is working sensibly. Since these are seen poses, the recognition accuracy on this sequence is expected to be very high.

The two demarcations mentioned above – normal-res vs. low-res and ‘seen’ vs. ‘seen + unseen’ are then combined to give four categories of testing sequences, namely:

1. Seen poses / Normal-res
2. Seen poses / Low-res
3. Seen + Unseen poses / Normal-res
4. Seen + Unseen poses / Low-res

Note that all these four sequences are mutually exclusive, i.e. there are no frames present in more than one of these sequences. Moreover, such sequences are created separately for each target, except for targets which simply don’t have the right kind of frames. E.g. if a target has no low-res frames, then category (2) and (4) sequences are not created for that
target. One more thing to note is the usage of these categories with models generated from the ‘short’ and the ‘long’ modeling sequences. It can be seen that there is an overlap between the ‘long’ modeling sequence and the ‘seen’ testing sequences (categories (1) and (2) above). To circumvent this, two separate runs of recognition are performed, where in the first, the ‘short’ modeling sequence is used for modeling and frames from all four testing categories are used to get the recognition accuracies. In the second run, the ‘long’ training sequence is used for modeling and frames from only categories (3) and (4) are used during recognition. Thus, there is never an overlap between the training set and the testing set.

Table 4-1 summarizes the previous two sections and details the poses covered by the modeling and training sequences for each target.
Table 4-1: Poses covered during modeling and testing (in degrees approximate)

<table>
<thead>
<tr>
<th>Target</th>
<th>Seen</th>
<th>Seen+Unseen</th>
<th>Modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Non-low res</td>
<td>Low res</td>
<td>Non-low res</td>
</tr>
<tr>
<td>1</td>
<td>-40, -10, 20, 30, 40</td>
<td>45</td>
<td>(-70)-135</td>
</tr>
<tr>
<td>2</td>
<td>60, 120</td>
<td>No poses</td>
<td>(-45), (-30), 30-170</td>
</tr>
<tr>
<td>3</td>
<td>20, 50</td>
<td>No poses</td>
<td>(-10)-135, 170</td>
</tr>
<tr>
<td>4</td>
<td>0, 30, 45, 135</td>
<td>No poses</td>
<td>(-50), (-25), (-20), 0-65, 85-100</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>No poses</td>
<td>(-60), (-30), (-20), (-10), 5-10, 30, 40, 45-60</td>
</tr>
<tr>
<td>6</td>
<td>(-30), (-120)</td>
<td>No poses</td>
<td>(-30), (-120)</td>
</tr>
<tr>
<td>7</td>
<td>30, 160</td>
<td>No poses</td>
<td>(-160), (-135), (-130)-(-110), (-95)-(-70), (-65)-0, 30, 160, 170-180</td>
</tr>
<tr>
<td>8</td>
<td>(-150), (-90)</td>
<td>No poses</td>
<td>(-180), (-150), (-140), (-30), 150-175</td>
</tr>
<tr>
<td>9</td>
<td>(-45), 40</td>
<td>120</td>
<td>30, 40, 170</td>
</tr>
<tr>
<td>10</td>
<td>No poses</td>
<td>No poses</td>
<td>0-20, 30</td>
</tr>
<tr>
<td>11</td>
<td>No poses</td>
<td>No poses</td>
<td>No poses</td>
</tr>
</tbody>
</table>

4.3 Modeling and recognition

This section describes the experimental details of how the modeling sequences are used to create the target models, how the recognition is performed and how the performance numbers are obtained.
4.3.1 Getting data ready for experimentation

Before we start in this section, what we have are short and long modeling sequences for each target. Each of these sequences is passed through the following steps. These steps are those that have been described in the previous chapters.

1. Perform preprocessing. In this step, SIFT keypoints are extracted from each frame. These keypoints are then processed using the procedure described in section 2.1 to give a set of good keypoints for each frame.

2. Modeling is carried out using Lowe’s feature view clustering approach as described in section 3.1. We thus get two separate target models (short and long) for each target.

3. The target models for all targets are combined to form two model databases labeled ‘short’ and ‘long’ corresponding to the type of modeling sequence they were derived from.

As described in the previous section, we have four categories of recognition sequences. For each such sequence for each target, the preprocessing step described above is carried out, so that we have good keypoint data for each frame in the sequence. All this per-frame keypoint data for each target is combined into one huge recognition database. There’s one such recognition database for each category of recognition sequences. Note that the recognition database still holds information as to which frame belongs to which target. This is important for the experiments which will be described further.
From an implementation point of view, the model databases and recognition databases have the following hierarchy as described in figures 4.3 and 4.4.

Fig. 4.3: Hierarchy of the modeling database

Fig. 4.4: Hierarchy of the recognition database
4.3.2 Experimentation

Once we have the target model databases and the recognition databases ready, the experimentation can begin. The results were obtained using the following algorithm:

1. Select a model database – either ‘short’ or ‘long’.
2. Select a recognition database, within the constraints that the ‘seen’ categories aren’t used with the ‘long’ modeling sequences.
3. For each target in the recognition database, cycle through all frames and associated keypoint data.
   a. For each frame, feed this frame and keypoint data to the recognition algorithm described in section 3.2.
   b. The recognition algorithm returns a decision on which target is visible in the frame or, if no decision was made, the reason for indecision.
   c. Note the result of the recognition algorithm and update the numbers in a recognition log.
   d. Move on to the next target and repeat steps (3a) – (3d) until all targets are processed.

The recognition accuracy was taken as the ratio of correctly recognized frames to the total number of frames tested. Along with the correct recognition accuracy, the percentage of incorrect recognition and no decision was also measured. Now there are four cases which can lead to indecision –

a) Failure in probabilistic verification.
b) No similarity transformation found due to a bad set of keypoint matches. An example case for this is where only one keypoint was extracted from each color plane, but each was at the same location in the frame. The number of keypoints is then three, but since it is the same keypoint, it’s an underconstrained equation to solve for calculating the similarity transformation.

c) Less than two model matches found. No similarity transformation can be found in this case.

d) No model matches found.

To better understand why an indecision occurred, the percentage occurrence of each of the above cases is also noted. The results and their analysis are given in the next section.

4.4 Results

Tables 4-2 to 4-10 detail the results of recognition performed on different recognition sequences using different modeling sequences. Tables 4-2 to 4-7 show recognition results using the short modeling sequence whereas tables 4-8 to 4-10 are for the long modeling sequence. Entries which say ‘NaN’ indicate no data due to no available frames. Each table shows the number of frames tested for each target along with the number of frames used for modeling. This can be used to better analyze the percentages.
4.4.1 Short training sequence recognition

Here’s a detailed description of the short training sequence tables. Tables 4-2 and 4-3 summarize the correct and incorrect recognition results for each target and type of recognition sequence. Tables 4-4 to 4-7 describe the split among recognition failures for the different recognition sequences.

Table 4-2: ‘Short’ training sequence ‘seen’ recognition results

<table>
<thead>
<tr>
<th>Target</th>
<th>Non-low res</th>
<th>Seen</th>
<th>Low res</th>
<th>Number of modeling frames</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Correct %</td>
<td>Incorrect %</td>
<td>Frames</td>
<td>Correct %</td>
</tr>
<tr>
<td>1</td>
<td>99.78</td>
<td>0.22</td>
<td>917</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>99.74</td>
<td>0.26</td>
<td>381</td>
<td>NaN</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>0</td>
<td>862</td>
<td>NaN</td>
</tr>
<tr>
<td>4</td>
<td>91.95</td>
<td>1.24</td>
<td>969</td>
<td>NaN</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>0</td>
<td>129</td>
<td>NaN</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>0</td>
<td>248</td>
<td>NaN</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>0</td>
<td>150</td>
<td>NaN</td>
</tr>
<tr>
<td>8</td>
<td>99.62</td>
<td>0</td>
<td>262</td>
<td>NaN</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>0</td>
<td>464</td>
<td>80.56</td>
</tr>
<tr>
<td>10</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
<td>NaN</td>
</tr>
<tr>
<td>11</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
<td>NaN</td>
</tr>
<tr>
<td>TOTAL</td>
<td>98.13</td>
<td>0.34</td>
<td>4382</td>
<td>93.04</td>
</tr>
</tbody>
</table>
### Table 4-3: ‘Short’ training sequence ‘seen+unseen’ recognition results

| Target | Non-low res |  | Low-res |  |  |  | Number of modeling frames |
|--------|-------------|----------------|--------|--------|--------|---------------------------|
|        | Correct %   | Incorrect %   | Frames | Correct % | Incorrect % | Frames |
| 1      | 87.77       | 7.21          | 2542   | 43.73   | 10.04   | 827           | 160       |
| 2      | 85.86       | 8.23          | 2706   | 50      | 22.97   | 370           | 100       |
| 3      | 94.03       | 3.64          | 1485   | 48.19   | 18.8    | 500           | 120       |
| 4      | 88.95       | 0.75          | 266    | 60      | 3.67    | 572           | 140       |
| 5      | 86.44       | 1.06          | 752    | 77.63   | 3.02    | 431           | 60        |
| 6      | 92.83       | 6.95          | 463    | NaN     | NaN     | 0             | 40        |
| 7      | 87.21       | 4.82          | 2686   | 79.21   | 1.82    | 329           | 140       |
| 8      | 87.85       | 5.87          | 1645   | NaN     | NaN     | 0             | 100       |
| 9      | 85.38       | 0             | 294    | 83.84   | 0       | 99            | 60        |
| 10     | 96.27       | 2.36          | 888    | 88.56   | 0.54    | 1285          | 100       |
| 11     | NaN         | NaN           | 0      | 39.61   | 8.44    | 474           | 20        |
| TOTAL  | 88.59       | 5.46          | 13727  | 64.14   | 7.14    | 4887          |

### Table 4-4: Recognition failures for ‘short’ training sequence ‘seen – non-low-res’ recognition

<table>
<thead>
<tr>
<th>Target</th>
<th>Probabilistic verification %</th>
<th>No similarity transformation found %</th>
<th>Less than 2 model matches %</th>
<th>No model matches %</th>
<th>Incorrect %</th>
<th>Total frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.22</td>
<td>917</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.26</td>
<td>381</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>862</td>
</tr>
<tr>
<td>4</td>
<td>4.33</td>
<td>0</td>
<td>2.48</td>
<td>0</td>
<td>1.24</td>
<td>969</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>129</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>248</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>150</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0.38</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>262</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>464</td>
</tr>
<tr>
<td>10</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>0.96</td>
<td>0.023</td>
<td>0.55</td>
<td>0</td>
<td>0.34</td>
<td>4382</td>
</tr>
</tbody>
</table>
Table 4-5: Recognition failures for ‘short’ training sequence ‘seen – low-res’ recognition

<table>
<thead>
<tr>
<th>Target</th>
<th>Seen verification %</th>
<th>No similarity transformation found %</th>
<th>Seen less than 2 model matches %</th>
<th>No model matches %</th>
<th>Incorrect %</th>
<th>Total frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>15.28</td>
<td>0</td>
<td>4.17</td>
<td>0</td>
<td>0</td>
<td>72</td>
</tr>
<tr>
<td>10</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>5.47</td>
<td>0</td>
<td>1.49</td>
<td>0</td>
<td>0</td>
<td>201</td>
</tr>
</tbody>
</table>

Table 4-6: Recognition failures for ‘short’ training sequence ‘seen+unseen – non-low-res’ recognition

<table>
<thead>
<tr>
<th>Target</th>
<th>Seen+Unseen verification %</th>
<th>No similarity transformation found %</th>
<th>Seen+Unseen less than 2 model matches %</th>
<th>No model matches %</th>
<th>Incorrect %</th>
<th>Total frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.62</td>
<td>0.47</td>
<td>1.93</td>
<td>0</td>
<td>7.21</td>
<td>2542</td>
</tr>
<tr>
<td>2</td>
<td>3.63</td>
<td>0.96</td>
<td>1.32</td>
<td>0</td>
<td>8.23</td>
<td>2706</td>
</tr>
<tr>
<td>3</td>
<td>1.44</td>
<td>0.4</td>
<td>0.49</td>
<td>0</td>
<td>3.64</td>
<td>1485</td>
</tr>
<tr>
<td>4</td>
<td>6.77</td>
<td>0.75</td>
<td>2.03</td>
<td>0.75</td>
<td>0.75</td>
<td>266</td>
</tr>
<tr>
<td>5</td>
<td>9.18</td>
<td>0.13</td>
<td>3.19</td>
<td>0</td>
<td>1.06</td>
<td>752</td>
</tr>
<tr>
<td>6</td>
<td>0.22</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6.95</td>
<td>463</td>
</tr>
<tr>
<td>7</td>
<td>5.51</td>
<td>0</td>
<td>2.46</td>
<td>0</td>
<td>4.82</td>
<td>2686</td>
</tr>
<tr>
<td>8</td>
<td>4.46</td>
<td>0.24</td>
<td>1.58</td>
<td>0</td>
<td>5.87</td>
<td>1645</td>
</tr>
<tr>
<td>9</td>
<td>7.48</td>
<td>0</td>
<td>7.14</td>
<td>0</td>
<td>0</td>
<td>294</td>
</tr>
<tr>
<td>10</td>
<td>0.65</td>
<td>0</td>
<td>0.72</td>
<td>0</td>
<td>2.36</td>
<td>888</td>
</tr>
<tr>
<td>11</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>3.81</td>
<td>0.37</td>
<td>1.75</td>
<td>0.015</td>
<td>5.46</td>
<td>13727</td>
</tr>
</tbody>
</table>
Table 4-7: Recognition failures for ‘short’ training sequence ‘seen+unseen – low-res’ recognition

<table>
<thead>
<tr>
<th>Target</th>
<th>Probabilistic verification %</th>
<th>No similarity transformation found %</th>
<th>Less than 2 model matches %</th>
<th>No model matches %</th>
<th>Incorrect %</th>
<th>Total frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23.82</td>
<td>3.14</td>
<td>19.27</td>
<td>0</td>
<td>10.04</td>
<td>827</td>
</tr>
<tr>
<td>2</td>
<td>15.14</td>
<td>0.27</td>
<td>11.62</td>
<td>0</td>
<td>22.97</td>
<td>370</td>
</tr>
<tr>
<td>3</td>
<td>15.8</td>
<td>7</td>
<td>10.21</td>
<td>0</td>
<td>18.8</td>
<td>500</td>
</tr>
<tr>
<td>4</td>
<td>19.23</td>
<td>0.7</td>
<td>16.05</td>
<td>0.35</td>
<td>3.67</td>
<td>572</td>
</tr>
<tr>
<td>5</td>
<td>10.89</td>
<td>0.46</td>
<td>8</td>
<td>0</td>
<td>3.02</td>
<td>431</td>
</tr>
<tr>
<td>6</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>9.18</td>
<td>0.91</td>
<td>8.88</td>
<td>0</td>
<td>1.82</td>
<td>329</td>
</tr>
<tr>
<td>8</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>13.13</td>
<td>0</td>
<td>3.03</td>
<td>0</td>
<td>0</td>
<td>99</td>
</tr>
<tr>
<td>10</td>
<td>7.16</td>
<td>0.39</td>
<td>3.35</td>
<td>0</td>
<td>0.54</td>
<td>1285</td>
</tr>
<tr>
<td>11</td>
<td>26.58</td>
<td>2.11</td>
<td>22.42</td>
<td>0.84</td>
<td>8.44</td>
<td>474</td>
</tr>
<tr>
<td>TOTAL</td>
<td>15.35</td>
<td>1.76</td>
<td>11.48</td>
<td>0.12</td>
<td>7.14</td>
<td>4887</td>
</tr>
</tbody>
</table>

From tables 4-2 and 4-3, the following observations can be made:

1. **‘Seen’ recognition results**: The recognition accuracy for the ‘seen’ recognition sequences is consistently high and close to 100%. This is expected since all poses in the ‘seen’ recognition are already present in the model. This can be seen as a validation of the modeling and recognition algorithm.

2. **‘Seen + Unseen’ recognition results**: The recognition results for the non-low-res section of ‘seen + unseen’ testing are consistently above 85%. However, for the low-res section, the accuracy was much lower. Note that the incorrect recognition percentage is still typically less than 10%. Thus most of the failure is because of indecision.
Tables 4-4 and 4-5 show a split-up of the recognition failures for the ‘seen’ recognition sequences. Since the recognition accuracy for these sequences is very high, most of this table is empty. Tables 4-6 and 4-7 for the ‘seen + unseen’ recognition sequences on the other hand are much more informative as to where recognition failures occurred. From both the non-low-res and low-res sections, it can be seen that most indecision is because of failures during the probabilistic verification phase or simply not enough matches were found to calculate a similarity transformation. The incorrect recognition percentage is almost consistently less than 10%. For the non-low-res section, indecision failures are typically around 4%.

4.4.2 Long training sequence recognition

Tables 4-8 to 4-10 show the results for recognition using the long training sequence. Table 4-8 summarizes the correct and incorrect recognition results. It can be seen that there are no results for the ‘seen’ recognition sequence since this was not to be used along with the long modeling sequence. The recognition results for both non-low-res and low-res sections are generally better than the corresponding results using the short modeling sequence. However, no big jump is seen in the numbers. Hence it can be concluded that repeatedly modeling poses which have already been seen is not very essential to the recognition process, although it does result in a slight improvement.
### Table 4-8: ‘Long’ training sequence ‘seen+unseen’ recognition results

<table>
<thead>
<tr>
<th>Target</th>
<th>Seen+Unseen</th>
<th>Low-res</th>
<th>Number of modeling frames</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Non-low res</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Correct %</td>
<td>Incorrect %</td>
<td>Frames</td>
</tr>
<tr>
<td>1</td>
<td>91.27</td>
<td>3.5</td>
<td>2542</td>
</tr>
<tr>
<td>2</td>
<td>87.87</td>
<td>7.06</td>
<td>2706</td>
</tr>
<tr>
<td>3</td>
<td>96.27</td>
<td>2.21</td>
<td>1485</td>
</tr>
<tr>
<td>4</td>
<td>90.95</td>
<td>0.38</td>
<td>266</td>
</tr>
<tr>
<td>5</td>
<td>87.18</td>
<td>1.86</td>
<td>752</td>
</tr>
<tr>
<td>6</td>
<td>97.18</td>
<td>2.16</td>
<td>463</td>
</tr>
<tr>
<td>7</td>
<td>92.34</td>
<td>1.12</td>
<td>2686</td>
</tr>
<tr>
<td>8</td>
<td>88.03</td>
<td>9.42</td>
<td>1645</td>
</tr>
<tr>
<td>9</td>
<td>82.32</td>
<td>1.7</td>
<td>294</td>
</tr>
<tr>
<td>10</td>
<td>98.58</td>
<td>0.68</td>
<td>888</td>
</tr>
<tr>
<td>11</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>91.21</td>
<td>3.89</td>
<td>13727</td>
</tr>
</tbody>
</table>

### Table 4-9: Recognition failures for ‘long’ training sequence ‘seen+unseen – non-low-res’ recognition

<table>
<thead>
<tr>
<th>Target</th>
<th>Seen+Unseen</th>
<th>Non-low res</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Probabilistic verification %</td>
<td>No similarity transformation found %</td>
<td>Less than 2 model matches %</td>
<td>No model matches %</td>
</tr>
<tr>
<td>1</td>
<td>2.81</td>
<td>0.43</td>
<td>1.99</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3.36</td>
<td>0.3</td>
<td>1.41</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1.42</td>
<td>0.07</td>
<td>0.03</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>4.89</td>
<td>0.75</td>
<td>1.9</td>
<td>1.13</td>
</tr>
<tr>
<td>5</td>
<td>7.96</td>
<td>0.27</td>
<td>2.73</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0.58</td>
<td>0</td>
<td>0.08</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>4.84</td>
<td>0.11</td>
<td>1.59</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1.09</td>
<td>0</td>
<td>1.46</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>8.71</td>
<td>0</td>
<td>7.27</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0.41</td>
<td>0</td>
<td>0.333</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>TOTAL</td>
<td>3.18</td>
<td>0.20</td>
<td>1.50</td>
<td>0.023</td>
</tr>
</tbody>
</table>
Table 4-10: Recognition failures for ‘long’ training sequence ‘seen+unseen – low-res’ recognition

<table>
<thead>
<tr>
<th>Target</th>
<th>Probabilistic verification %</th>
<th>No similarity transformation found %</th>
<th>Less than 2 model matches %</th>
<th>No model matches %</th>
<th>Incorrect %</th>
<th>Total frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21.77</td>
<td>0.12</td>
<td>20.82</td>
<td>0.12</td>
<td>10.52</td>
<td>827</td>
</tr>
<tr>
<td>2</td>
<td>7.57</td>
<td>0.81</td>
<td>19.29</td>
<td>0</td>
<td>16.76</td>
<td>370</td>
</tr>
<tr>
<td>3</td>
<td>8.8</td>
<td>3.6</td>
<td>15.8</td>
<td>0</td>
<td>20.6</td>
<td>500</td>
</tr>
<tr>
<td>4</td>
<td>6.64</td>
<td>1.22</td>
<td>30.23</td>
<td>0.35</td>
<td>6.29</td>
<td>572</td>
</tr>
<tr>
<td>5</td>
<td>7.89</td>
<td>1.16</td>
<td>8.37</td>
<td>0</td>
<td>5.58</td>
<td>431</td>
</tr>
<tr>
<td>6</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>9.45</td>
<td>0</td>
<td>6.95</td>
<td>0</td>
<td>2.43</td>
<td>329</td>
</tr>
<tr>
<td>8</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>8.08</td>
<td>0</td>
<td>2.02</td>
<td>0</td>
<td>0</td>
<td>99</td>
</tr>
<tr>
<td>10</td>
<td>4.67</td>
<td>0</td>
<td>11.98</td>
<td>0</td>
<td>1.17</td>
<td>1285</td>
</tr>
<tr>
<td>11</td>
<td>9.28</td>
<td>0</td>
<td>39.83</td>
<td>0.84</td>
<td>10.34</td>
<td>474</td>
</tr>
<tr>
<td>TOTAL</td>
<td>9.56</td>
<td>0.70</td>
<td>18.40</td>
<td>0.14</td>
<td>7.86</td>
<td>4887</td>
</tr>
</tbody>
</table>

4.5 Analysis of failure modes

This section looks at the failure numbers in the results and analyses their causes. Here we will concentrate mainly on tables 4-4 to 4-7 and 4-9 to 4-10 to see where failures occurred. To come up with the analysis, data was collected on which frames in the recognition sequence failed and for what reason. Then, the appearance of the target and location of descriptors in that frame was observed and compared to the recognized target, if any. Characteristics of the comparison that were noted included the following:

a) Similarity of appearance between test frame and recognition result.

b) Number of descriptors and their location in the test frame.

c) Pose of the target in the test frame, especially relative to the pose range of that target present in the modeling sequence.
Most of the failures could be explained using the above three metrics. The following general observations were made:

a) In general, recognition failures, either incorrect recognition or indecision, occurred at the edges of the pose range modeled. E.g. if for a target, poses corresponding to 10, 20, 30 degrees were modeled and the testing poses ranged from (-10) degrees to 50 degrees, it was observed that there were more failures for frames with poses around (-10) degrees and 50 degrees compared to other poses.

b) Incorrect recognition failures mostly occurred because the test frame target had an appearance and pose very similar to another target modeled.

c) Failures in low-res sequences occurred because there weren’t enough good descriptors or descriptors did not have a strong localization.

Points (b) and (c) are elaborated with examples in the next sections.

4.5.1 Incorrect recognition failures

Most incorrect recognition failures were observed to be due to similarity in appearance and pose of the test frame target to another target present in the model. Figures 4.5 to 4.8 show such examples for various targets. These examples are frames taken from a set of recognition frames for a target where the same incorrect recognition result was returned for all frames in that set. E.g. fig. 4.5 is an example frame taken from target 1 recognition sequence frames 496-516, where the recognition result was (incorrectly) target 3 for all these 21 frames.
Fig. 4.5: Target 1 (recognition sequence, left) vs. target 3 (training sequence, right)

Fig. 4.6: Target 2 (recognition sequence, left) vs. target 4 (training sequence, right)
4.5.2 Bad set of keypoints

Most of the failures in the low-res sequences were due to no decision being returned by the recognition algorithm. And as can be seen from tables 4-6, 4-7, 4-9, 4-10, most of the
indecision is because of failures during probabilistic verification and not enough keypoint matches found.

Low-res frames in general had poor contrast and definition of vehicle features. This led to very few keypoints being found and those that were found had poor localization and weren’t consistent across consecutive frames. Some of the worst-performing targets were targets 1-4, whose low-res sequence versions are shown in fig. 4.9. These images were of such low resolution because the target was very far away from the camera and as such had a very small spatial expanse in the frame.

Fig. 4.9: (Clockwise from top left) Low-res frames of targets 1, 2, 4, 3 with superimposed keypoint locations for all three color planes.
Sequences for targets 1-4 were the worst performing low-res sequences. The other target sequences had better accuracies but they were still worse than those for the non-low-res sequences.

4.5.2.1 Modeling for low-res data

During modeling, it was assumed that for a particular target pose, consistent keypoints extracted from a normal-res image would be a superset of those extracted from a low-res version. The low-res version will of course have numerous other inconsistent keypoints.

Because of the above assumption, if a particular target pose was modeled using a normal-res frame, that pose wasn’t modeled again using a low-res version, as it was thought that the normal-res modeling would suffice. However, looking at the results of low-res recognition, it seems that this assumption may be wrong. Although it is still possible that consistent keypoints in normal-res version are a superset of those in low-res version, it is not enough to simply model the normal-res version. Having separate target models for normal-res and low-res frames may improve accuracy for the low-res sequences. However, this experiment hasn’t been carried out as part of this work.

In this work, the basis of classifying an image as being low-res has been the total number of pixels on the target (the target spatial support). An image would be low-res if the total number of pixels on the target is below a certain threshold. This criterion accounts for almost all cases such as low pixel count in the frame itself, high pixel count in the frame
but target is too far from camera, etc. Here, since the classification as low-res and normal-res was done manually, no value is available for the threshold, but one can be assigned in automated low-res/normal-res identification methods. The number of keypoints extracted from the frame could also be of help to identify the resolution. If there are very few keypoints in the frame, the frame is usually low-res. However, there could be a case where there are many keypoints extracted but most of them are on the background because the target spatial support was very small. This may get wrongly classified as normal-res. Some more information will be required when using keypoints as the criterion for low-res/normal-res identification.

4.6 Modeling complexity

Lastly, this section takes a look at the question of signature explosion and how complex the model for a given target is, in terms of number of keypoints present. More the number of keypoints in the model, larger the search space for keypoint matches during recognition and hence slower the process. Apart from performance, a model with too many keypoints is difficult to store and update. Also, since this is a video context where new frames are modeled as they come in, the number of keypoints in the model can easily get out of hand and needs to be kept under check.

In section 3.1.3, techniques employed to handle signature explosion were described. These were:
a) Lowe’s method, where, when keypoints from new frames are to be merged with keypoints in the model view, discard those new frame keypoints for which the ratio of distance to the closest model match and that to the second-closest model match is less than 1/3.

b) An additional constraint that new frame keypoints should not be merged (added) to the model view if the fitting error during similarity transformation calculation is below a certain threshold.

The main goal of this approach was to prevent a linear increase in the number of keypoints in the model as new frames are modeled.

During modeling the fitting error threshold for technique (b) above was set to 0.3 times the error threshold for deciding whether or not to merge the new frame keypoints with the winning model view. As an initial experimentation to test the above techniques, a test sequence was created from one of the video sequences containing target 3. There was nothing special about target 3 and this test sequence was unrelated to any modeling or recognition sequence. This test sequence was 470 frames long. Manual observation showed that the vehicle pose was constant till frame 390 and then undergoes continuous change till frame 470. To see the effect of modeling frames containing poses already modeled, this test sequence was stitched to itself back-to-back, so that it was now a 940 frame sequence with two identical 470-frame halves. This was then fed to the modeling algorithm and the following parameters were noted:

a) Number of keypoints added to the model vs. number of frames modeled.
b) Number of new frame keypoints discarded due to violation of either of the above two signature explosion handling techniques, again vs. number of frames modeled.

c) Ratio of similarity transformation fitting error and model-view merge error threshold vs. number of frames modeled.

These parameters are plotted in the graphs shown in fig. 4.10.
Fig. 4.10: Modeling behavior with respect to addition and discarding of keypoints

The following observations can be made from these graphs:
a) In the first plot, the number of keypoints added to the model flattens out for frames from ~50 to ~400 and again from frame ~500 to ~900.

b) There are three growths in the plot, from frame 0 to ~50, ~400 to ~500 and ~900 to ~940.

c) The number of keypoints discarded (the second plot) is two orders of magnitude greater than the number of keypoints added to the model.

The following conclusions can be made from the above observations:

a) If the modeling algorithm sees a continuous series of frames with the same pose, it won’t add new keypoints to the model.

b) Whenever there is a change in pose, keypoints will be added to the model.

It can be seen that the aforementioned techniques weren’t completely successful in stopping the linear growth of keypoints in the model. However, considering that the number of keypoints that were discarded was two orders of magnitude more than those added, and looking at the good recognition results, it can be said that the system can work accurately with the minimal amount of keypoint data. Also, since the keypoint growth is not drastic, there is scope for post-processing and compression of model data once a good set of poses are observed, and this can be done before the number of keypoints in the model grows to a huge number. However, these aspects aren’t covered in this work.

The long modeling sequences were specifically created to observe how the modeling algorithm handles the new keypoints coming in with each frame. Figures 4.11 to 4.21 show the number of keypoints added to the model and those that were discarded, for each target plotted against number of frames modeled.
Fig. 4.11: Target 1 modeling behavior showing addition and discarding of keypoints

Fig. 4.12: Target 2 modeling behavior showing addition and discarding of keypoints
Fig. 4.13: Target 3 modeling behavior showing addition and discarding of keypoints

Fig. 4.14: Target 4 modeling behavior showing addition and discarding of keypoints
Fig. 4.15: Target 5 modeling behavior showing addition and discarding of keypoints

Fig. 4.16: Target 6 modeling behavior showing addition and discarding of keypoints
Fig. 4.17: Target 7 modeling behavior showing addition and discarding of keypoints

Fig. 4.18: Target 8 modeling behavior showing addition and discarding of keypoints
Fig. 4.19: Target 9 modeling behavior showing addition and discarding of keypoints

Fig. 4.20: Target 10 modeling behavior showing addition and discarding of keypoints
Fig. 4.21: Target 11 modeling behavior showing addition and discarding of keypoints
Chapter 5

Conclusion

An application of Lowe’s feature view clustering approach to video object recognition was described in this thesis. The main thrust of this work was towards obtaining an object recognition method which is robust towards pose change. As can be seen from the results, this goal was achieved. A new technique for avoiding signature explosion was proposed and it was effective without adversely affecting recognition results.

5.1 Future work

An advantage of this approach is that it is pose-robust without having to rely on any pose estimation technique. However, adding a pose estimation phase could greatly improve recognition performance. Currently, an exhaustive search is carried out while matching current frame keypoints to the target model. If each model view could have an associated pose and there was a way to estimate the pose in the current frame, one could directly index the model using this pose information, thus reducing processing time.

A quick pose estimation technique can be as follows. Consider the entire video frame and note the location of the bounding box centre in that frame. Draw a line between the bounding box centers of the previous and the next frame and see if the current bounding box center lies close to this line. If it does, then this line indicates the general motion
vector of the target and its angle with respect to the X-axis can be considered as its pose since it is an aerial video sequence. Note that until now we have only been concerned with the contents of the bounding boxes. But this technique requires us to store the location of the bounding box in the full video frame as well.

Another extension of this work could also be an on-the-fly modeling and recognition scheme, where the target model is updated with new appearances during periods of good tracking, thus keeping the model up-to-date with latest changes. This can be seamlessly implemented without major changes in the proposed algorithm.
Bibliography


