Video Object Counting in Unconstrained Environments Using Density-Based Clustering

by

Onalenna Junior Makhura

A thesis submitted for the degree of PhD
Department of Computer Science and Electronic Engineering
University of Essex
Date of Submission for Examination: May, 2019

Abstract

In this thesis, we present a video object counting approach using multiple local feature matching. We explain the development of a dataset with which to test our approach. Our dataset uses a new approach which we designed to extract object ground truth. We also provide a comparison of common single object trackers. We develop a multi-object tracker named Learn-Select-Track and use it to track the colours of objects of interest to filter out false positive object localisations.

We discuss the implementation of the HDBSCAN algorithm which we use in our novel approach for matching multiple local feature descriptors. We show that the detected clusters provide very good matches for the features and demonstrate our approach to cluster analysis and validation. We develop a simple yet efficient way of learning the features of the object of interest which is independent of the number of objects in the frame.

We also develop a computationally simple way of detecting the other objects in the frame by using a combination of the detected clusters, the features of the object of interest and vector algebra. Our approach is capable of detecting partially visible and occluded objects as well. We present three ways of extracting object count estimations from videos and provide empirical evidence to show that our approach can be used in a wide variety of scenarios.

To my lovely son, Michael.

"The most valuable gift you have to offer is yourself - The Go-Giver, Burg and Mann"

Declaration of Authorship

I, Onalenna Junior Makhura, declare that this thesis titled, 'Video Object Counting in Unconstrained Environments Using Density-Based Clustering' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ______________________________

Date: ______________________________

Acknowledgements

First, I would like to acknowledge my supervisor Dr. John Woods. He has been a rock steady guide for me in this journey. From the moment I contacted him to be my supervisor, he has been kind and supportive. He offered me great advice and encouragement which I will forever be grateful for. I pray that he may forever be such a wonderful supervisor for many more aspiring PhD student like me.

I would also like to acknowledge my family. My mother, Leah Masilo, has always been a symbol of calm strength. She has been an example of how to weather the difficulties without losing who you are. Her faith in me has never wavered, which has greatly influenced me to give my best. My son, Bonno Michael Mokgobi has always been a beacon of hope and direction in my life. He always helps me remember what is truly important in my life.

I would be forgetful if I do not acknowledge my friends, who have always encouraged and believed in me. Obakeng and Lebogang, who have been my friends for more than 15 years. Without them, my affairs at home would not have been taken care of while I pursue my PhD. Dr. Thabiso Maupong has been an inspiration for me. I am so glad that he was 2 years ahead of me which served as a perfect template of how to approach my PhD studies. His guidance and advice were truly amazing. To Kabo, Kagiso, Ella, Khana and Wada, I hope to repay your kindness one day. You guys are awesome.

My sponsor, the University of Botswana has literally given me my dream of pursuing PhD and a career in academics. Through the Training Department, my studies have been a smooth road. Even though I was a pain at times, the staff handled things brilliantly. Special recognition to Neo Seroke who has handled my interactions with them with grace and always with a smile on her face.

Last, and most importantly, I would like to appreciate God for all the blessings I mentioned above and many more. The presence of all these wonderful people in my life could only have been by His grace. He has always been a voice in my heart guiding me to be where I am today. I will forever be grateful and thankful to Him.

Contents

List of Figures

List of Tables

Listings

Glossary

ANN - Artificial Neural Network designed for classification tasks by mimicking the way the brain learns patterns in data..

CNN - Convolutional Neural Network designed for image processing. Useful for object recognition.

DBSCAN - Density based clustering algorithm for finding clusters in data..

FLANN - Fast Library for Approximating Nearest Neighbours.

HDBSCAN - Hierarchical DBSCAN designed to have the ability to find varying density clusters by finding all possible DBSCAN solutions..

HODV - High Object Density Videos are videos where object count are more than 50. These videos provide the highest chance for object occlusions and false positive object locations..

LODV - Low Object Density Videos have object count less than 10. These videos often have very detailed objects that are close to the camera recording the video. They are often subject to over clustering of the features leading to some objects being identified multiple times..

MODV - Moderate Object Density Videos have object count greater than 10 but less than 50. These videos have a less chance of object occlusions. The object count estimations are subject ot whichever end of the scale the ground truth count is..

OpenCV - A suite of open source computer vision libraries. The library is implemented in C++ with python, java and matlab bindings. For this thesis, this library was used to handle video file frame extraction, object tracking and local feature extraction..

SIFT - Scale Invariant Feature Transform.

SURF - Seeded-Up Robust Features.

Publications

1. O. Makhura and J. Woods, "Multiple feature instance detection with density based clustering," in *2018 IEEE International Conference on Consumer Electronics (ICCE) (2018 ICCE)*, (Las Vegas, USA), Jan. 2018.
2. O. J. Makhura and J. C. Woods, "Learn-select-track: An approach to multi-object tracking," *Signal Processing: Image Communication*, vol. 74, pp. 153 – 161, 2019.
3. O. Makhura and J. Woods, "Video object counting dataset," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, (San Jose, USA), pp. 1 – 4, Mar 2019.

Chapter 1

Introduction

The ability to quantify objects is one of the ways of extracting semantic information about a scene. Therefore, object counting using computer vision techniques is a highly active and evolving research field. The algorithms developed have been applied in a wide variety of problems across the whole spectrum of human lives. For example, in large scale industries where thousands of products have to be counted, object counting algorithms are critical to industrial efficiency and profitability. In such industrial applications, computer vision based object counting can benefit from the controllability of the environment. The counting environment background and lighting are some of the factors that can be tightly controlled. The objects to be counted are often inanimate, as such, their size, colour and texture are static, allowing for counting algorithms that are tailor made for specific environments.

In such environments, it is common to see algorithms that rely on background subtraction in use. In those situations, since the background is known and static, its value can be removed from the images or video frames, and whatever colour is left can be considered to belong to the objects of interest. Histogram analysis can be accurately used in such environments since the size and colour of the objects are known. Other common algorithms include shape detectors, blob detectors and contour extraction algorithms.

Object counting is also being employed by animal conservation societies like Birdlife International [1] and Convention on International Trade on Endangered Species of Wild Fauna and Flora (CITES) [2]. Unlike the industrial environments, the animals are often in remote areas that are not easily accessible and the environment is even less controlled. Technological advances in areas such as satellite imagery and unmanned aerial vehicles (UAV) are giving researchers and conservationists access to such hard to reach places. A good example of use of such technologies can be found in [3], where the authors demonstrated using super-high resolution NASA satellite images to count albatrosses in the remote region of Chatham Islands.

In Botswana, Birdlife Botswana [4] published a guideline for monitoring bird population [5]. The process requires people to walk around in a set pattern and recording the count estimations of the birds they see. The obvious downsides to this approach is the cost in human capital and the inaccuracy of relying on humans to count birds in motion. Other factors such as fatigue and object density can affect the reliability of human object counting [6,7]. The document even notes the need for “scientifically sound low-tech monitoring methodology” while noting the need for their approach to be implemented by “as many participants as possible”. With vultures across Africa facing extinction [8,9], Birdlife Botswana also has signs across the country asking the public for information on vulture sightings.

The increase in accessibility of low cost but high quality imaging devices such as UAVs and smartphones can provide powerful tools to address the challenges highlighted above. The bird population monitors can use these devices to record the videos of birds for analysis later instead of just estimating the numbers based on what they are seeing then. With algorithms such as the one detailed in this thesis, the videos can be analysed to obtain more accurate and consistent count estimation. Even the public can use their smartphones to record the vulture sightings and send the videos to Birdlife Botswana for analysis.

The other problem in wildlife population monitoring is the widely diverse environments where the animals can be found. In order to be successful, object counting algorithms would have to be either designed for specific environments e.g. water, sky, clouds, forest etc., or they would need an in-built capability to adapt to these ever changing parameters. The approach described in this thesis focusses on the latter which means the algorithm can be deployed in different environments without any changes to how it processes the videos. Usually algorithms designed for non-constrained environments require a lot of time and data to train them. With infinite classes of objects that one may be interested in counting, the problem becomes more than just counting, the problem extends to counting *any* object.

In this thesis, the focus is on object counting for videos taken in unconstrained environments. The idea behind this research is to take a random video containing multiple instances of an object of interest, and using image and video processing computer vision algorithms, to identify all the object instances and give a count of the objects in the video frame. With consumer generated video data, the environments under which the object count must be extracted is unconstrained which provides a challenge for algorithms to be generic yet accurate. The scope of this thesis does not include aggregating the number of objects in the video, but rather, the number of objects in each video frame.

1.1 Review of Object Counting Approaches

In this section we review some of the recent object counting literature. In this thesis, we recognise three groups of object counting approaches. We first discuss the density estimation approaches which work by using image density maps and local feature mapping to estimate number of objects. Then we review object counting using object detection which aim to detect and localise objects of interest in images. Finally, we discuss counting by trajectory clustering approaches which are designed for video object counting through object motion detection. We discuss the findings and the shortcomings of these approaches which work by detecting object movements in videos.

1.1.1 Counting By Object Density Estimation

Density estimation based counting methods estimate a real valued density function of pixels in a given image by mapping local features of the image to its density map. These methods often require a set of manually labelled training images where they generate the ground truth density map, then extract local features and finally apply a regression model to learn the mapping between the local features and their corresponding density maps. The regression model is then used to estimate the density map of any image and the object count estimation is extracted by calculating the integral of the density map.

Density estimation based methods such as the one outlined in [10], tend to degrade in performance when new objects and scenes are encountered. In [11], the authors address this problem by proposing a manifold based density estimation counting method. Their approach is based on an assumption that the neighbouring image patches are more likely to share similar density patches, hence, images of objects share information as their density maps regarding the local geometries. As such the counting problem is converted into the problem of characterising the local geometry of a given image patch which makes their approach robust against features used and image resolution.

For this method, [11], the training requires annotated images and their ground truth density maps. The method then extracts image patches and their counterpart density patches. These are used in a feature engineering stage to create a hierarchical clustering tree structure using K-Means algorithm. The tree structure is then used to estimate the density map for each image patch. The object count can then be extracted by an integral of the density map. This form of training is a weakness to this approach as the annotation and ground truth density map generation means that the learning has to be done offline. The approach has been tested on cells [10], bees, fishes, birds [12] and pedestrian [13–15] datasets.

Convolutional neural networks (CNN), which are a type of artificial neural networks (ANNs) designed with image processing in mind, are also a big part of object recognition and classification and have proven in recent years to have state of the art performances. CNNs work by using artificial neurons to learn weights that are needed to recognise objects in the training dataset. They have been used with great success in challenges such as the ImageNet LSVRC-2010 challenge where the design outlined in [16] successfully deployed a CNN achieved top-1 and top-5 error rates of 37.5% and

17.0% on a test data comprised of high resolution images. The accuracy of CNNs does not only rely on the dataset, they also rely on the design of the neural networks itself.

In [17], an adaptive Counting Convolutional Neural Network (A-CCNN) is proposed which has the ability to handle large scale variations in people sizes and the facility to generate local density maps within a crowd scene. In the paper, the authors do not try to use different CCNN architectures [18], instead, they only try to select the most effective hyper parameters to generate the CNN model. The approach uses as inputs, an image divided into 16 part and head sizes from different parts of the image detected using tiny-face detection [19]. The image patches are then fed to the appropriate CCNN model with a proper hyper parameter by using Fuzzy Inference System. The CCNN model then produces density maps for each image section which are merged to obtain final density map. The authors trained and tested their approach on the UCSD dataset [18], UCF-CC Dataset [13] and the Sydney Train Footage dataset.

1.1.2 Counting By Object Detection

Object recognition/detection is the identification of the presence of a “known object”. With a near infinite number of objects, the main challenge is how to best describe what any particular object looks like in such a manner that a computer can then be able to recognise it in any context. In order to define a particular object, there is need to break down the object into a set of *features* that together can be used to identify an occurrence of the object. The features can then be matched in target images and frames to locate the objects of interest. The total number of object locations identified therefore represent the number of objects detected.

Investigations into fish population estimation and species classification is addressed in [20]. The authors use background subtraction to find the fish and Zernike moments to classify the species. In [21] a probabilistic approach is proposed where multiple object instances are detected using Hough Transform. The authors show how to detect instances without dealing with the problem of multiple peak identification in Hough images and without invoking non maximum suppression heuristics. Their approach, instead, detects multiple object instances using maximum-a-posteriori in the probabilistic model which contrasts the heuristic peak location and non-maximum suppression used in traditional Hough Transform.

In [22], a Density-based Clustering of Applications with Noise (DBSCAN) [23] is applied to Speeded-Up Robust Features (SURF) [24] in order to match a template to multiple inventory items. The approach extracts SURF features from both the prototype and the scene. Then DBSCAN is applied to the prototype features before applying feature mapping of the detected clusters to SURF points from the scene image. Sum of Squared Difference (SSD) and Nearest Neighbous Ratio (NNR) are used as matching factors.

DBSCAN is applied again to the scene features that have been matched to the template clusters and box information is extracted from the resulting clusters. Each of the clusters represents an instance of the prototype. Since there are multiple features within each cluster, there will be multiple boxes per instance. Box fixing is achieved by applying DBSCAN to the centroid locations to find the final box polygon. The approach uses hyper parameter tuning to determine the best values for $minPts$ and ϵ. This approach is designed for handling images of inanimate inventory objects. As such, it is not tested on videos or and living objects. The paper also relies on the prototype and scene objects having enough features to form clusters.

CNNs and other deep learning approaches have two setbacks; dataset requirements and training requirements. These approaches require a lot of training data, time and computation. Therefore, they are not suitable for use in real-time to handle new object types. The datasets often have to be split into training, testing and validations subsets [25]. These datasets have to be prepared beforehand often through annotations which takes a lot of time and effort. It also limits their use in object counting to the learned objects.

The authors of [26] offer an approach towards generic object counting which aims to minimise the problems encountered when using CNNs. Their approach uses a fully convolutional network architecture where the input is an image and a hierarchy of image divisions. While there is an extra need to divide the image into localised divisions and hierarchies, the approach reduces the need for annotating training images by global-image counts instead. They test their approach on MS-COCO [27], Pascal-VOC2007 [28] and CARPK [25] datasets. As a generic object counter, the authors did not eliminate the need for offline training and dataset preparation.

1.1.3 Counting By Trajectory Clustering

Movement of objects in a video has also been used for object counting. In [29], an algorithm for counting crowded moving objects by clustering a rich set of extended tracked features without requiring background subtraction is presented. The approach uses a highly parallelised KLT tracker to populate the spatio-temporal volume with a large set of feature trajectories. Normally, the feature trajectories would be fragmented and noisy, hence, a conditioning algorithm is used to smooth and extend raw feature trajectories. The conditioned trajectories are clustered using agglomerative clustering into candidate objects using a local rigidity constrained learned from a small set of training frames. The authors note that the training frames have to be prepared beforehand by labelling them only with the ground truth count. This prohibits their algorithm from working with a completely new object.

In [30], an algorithm of counting people (pedestrians) using trajectory clustering is proposed. The authors address the fragmentation and noise problems in trajectories by applying a combination of linear transformations, i.e. independent component analysis (ICA) plus rotation. The approach explores different sets of data representations and distance/similarity measures. The data representations are ICA, time series and maximum of cross-correlation (MSS). ICA is used with Euclidean distance, time series with longest common subsequence (LCSS) and MSS with Hausdorff distance.

Their approach in [30] is multi-layered with three levels using agglomerative clustering to avoid having a priori knowledge of how many pedestrians are in the video. The first two layers are basically pre-processing levels where the first level is a length-based clustering in which trajectories of similar lengths are grouped together. The second level is spatial clustering where they acknowledge that different individuals enter the video frame at different locations, unless two or more individuals are walking together. The actual counting happens in level three where the discrimination between oversampled individuals and different individuals walking together happens.

The datasets used in [29] provide object counts of varying size. The USC [31] dataset has between zero and twelve people. The other two datasets are from the library where there are between twenty to fifty people and the cells dataset has fifty to a hundred blood cells moving at different speeds. These two datasets would provide a challenge for preparing the training frames as there is a lot of people and cells. However, this challenge is not addressed in the paper. In [30], the videos used have less than eleven pedestrians.

With trajectory clustering approach, the objects have to be decomposed to their movements within the video. This means that a lot of information about the objects' shape, texture and colour are lost. As such, the approaches are commonly used in videos where the camera is stationary and there is only one class of objects. Non-stationary cameras provide a challenge because everything will appear to be moving including the background. While motion compensation can be used, the objects of interest must still be moving which is still a limitation.

1.2 Proposed Approach to Object Counting

Generally, object counting approaches are concerned with specific types of objects. This specificity is often enforced by either the type of features used, such as Circular Hough transforms to notice circular shapes of blood cells [32], or by the need for offline training such as in [29] and [17]. In this thesis, we broaden the parameters for object counting to reduce this specificity. While we do not address all the challenges that object counting presents, we aim to encompass a lot of the variables as outlined below:

- We aim to use any video including those that have unconstrained environments as long as there are multiple instances of the object of interest. In this thesis we address video properties such as changing background, foreground and lighting. The restriction to this aim is that the object should be visible enough to have the features we use for matching.

- Our approach must learn object features online. This means that we do not have to know what object we will be counting beforehand. The learning process should be independent of the number of objects in the video. It also means that the user interaction in the learning process must also be independent of the number of objects in the video.

- We aim to count any object regardless of shape, colour and texture. This property of our algorithm means that we have to be able to notice instances of the object even through self occlusion and deformities due to animate nature of live objects. In this thesis, however, we restrict the objects to have the same colour within the object class. This removes objects such as people who can wear different coloured clothing and have different skin colours.

The concept of object counting in this thesis is conceptualised as shown in Figure 1.1. On a frame by frame basis, the key inputs are a region of interest (ROI) of a known object of interest and the Speeded-Up Robust Features (SURF) [24] of a video frame. The features of the frame are then compared with features of the known object to find matching features. For this phase, variations in lighting, scale and rotation provide a big challenge. Another challenge for this phase comes from the feature representation. Often, instances of the same feature are not represented by exact values, as such, a margin of error needs to be established for matching. This margin of error can be difficult to ascertain and has no one size fits all solution. If the margin of error is too small, false negatives will be introduced into the final object count estimation and if it is too big, false positives are introduced instead.

In this thesis we develop an approach of matching multiple feature instances using a density based clustering algorithm which bypasses the need for determining the feature matching margin. Once the matches have been found, the locations of the objects of interest have to be determined. In this phase, scale and rotation have to be taken into consideration as well. Another challenge is occlusion. While humans can easily determine and properly identify occluded objects, computers have a difficult time with it. The question is often about when should objects in occlusion be considered a single object and when to count them separately. The localisation techniques also suffer from any inconsistencies from matching phase. This often means some

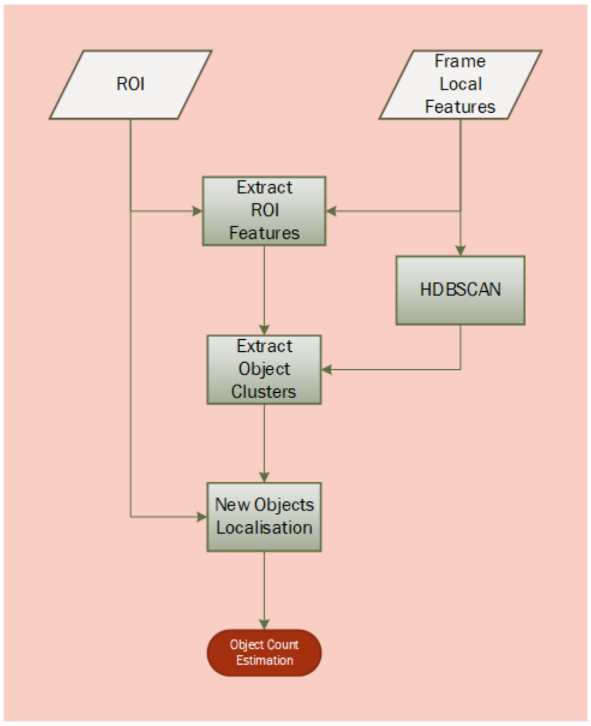


Figure 1.1: Concept of object counting for this thesis. On a frame-by-frame basis, the approach takes as input an ROI identifying one of the objects, and the SURF features from the frame and produces an estimation of the number of object in the frame by locating each instance of the object.

techniques are needed to detect the false positives and false negatives.

With object localisation complete, the number of locations represents the number of object instances detected in the frame. The problems highlighted above mean there is almost always a margin of error between the object count from computer vision based algorithms and the ground truth. The object counting algorithms have to be able to give an error bound for the count estimation. When comparing the count estimation to ground truths, it is easy to find the error bound. However, when dealing with the counting for a new video that has no ground truth, there needs to be a way to provide the certainty measurement of the count estimations.

The first part was to obtain the dataset to be used for testing the approach. As it turned out, there is no readily available video counting dataset for testing. Most research in object counting use images, and often focus on specific objects. In this thesis, we set out to create a video dataset dedicated specifically to object counting. The dataset was designed to be as general as possible. The main benchmark of the dataset is the frame by frame object count that was created using a human-computer hybrid approach which utilises people's ability to identify objects in complex environments and the computer's ability to quickly and efficiently count well-defined objects.

The aim for the approach was not just to count, it was to count any object irrespective of size, colour, texture or rotation. The easiest way to get the dataset needed was to look for videos on youtube. The dataset was designed not only to contain the videos with the objects to count, but also the ground truth of each frame in the video. The ground truths can then be used to verify the accuracy of the object counting algorithms.

Another requirement for this research was the low cost training for object detection and localisation. In this thesis, low cost training means that learning necessary features does not take a long time nor does it require a lot of data. In this context, cascade classifiers [33] and CNNs and other offline training methods are not viable options. For this thesis, low level local features, specifically (SURF) are used. These features can be extracted from each frame and processed frame by frame. The feature algorithms provide points where the frame has significant features and provide vectors called descriptors to define the feature.

For feature training, the user provides a region of interest (ROI) as a rectangle around one of the objects to be counted. Any local feature inside the ROI is then considered as part of a training set for the current frame while the other features in the frame are the query dataset. Those features are then matched to other features in the frame to find matches and possible locations of other objects of interest. In order to avoid asking the use to select the ROI for each frame, an online tracking algorithm is used on the initial ROI to track the selected region from one frame to the next.

In order to match the training dataset to the query dataset, a notion of density based clustering is applied. Local feature matching calculates distances between local features descriptors and considers two features to match if the distance between them is within a specified margin. Applying that to a frame with multiple instances of the same object, the features within those objects will be close to each other in the descriptor space, thus forming clusters. The matching task in this work, therefore, is to identify the clusters that contain the points from the training dataset.

As an example, a common approach is to use Lowe's ratio test [34] where the ratio between the two best matches has to be above 0.75. However, during experimentation, it was found that good matches can easily fall below that limit. The uncertainty in the margin of error creates the first problem to accurate object feature counting as discussed earlier. In this thesis, the problem is solved by using a clustering algorithm that discovers the margin of error for each cluster of features. The algorithm chosen in this thesis is the Hierarchical Density-based Clustering of Applications with Noise (HDBSCAN) [35]. The algorithm does an exploratory data analysis on the feature descriptors to discover the clusters that exist within the dataset and discovering the densities within the clusters.

The second problem to the accuracy of our approach is the object localisation. The features within the clusters that contain the features from the ROI give estimation of where the other objects of interest may be in the frame. The locations are extrapolated from the cluster intersections with the ROI. If a point inside the ROI is in a valid cluster, the location of that point inside the ROI is used to determine the possible positions of the objects represented by the other points within that cluster. To get the final object locations, template matching is applied around the locale of the matched feature points. The higher the number of objects in the video frame, the more likely there will be inter-object occlusion. This problem is solved by discarding new locations if there is a 50% or more intersection with other object locations.

The exploratory data analysis approach to feature matching yielded three types of results. The first type of results is the over-segmented clusters. These results provided clusters which after inspection were noted to be of one feature type and should have been one cluster. The counting estimation from such results leads to a lot of false negatives and produces much lower object count than desired. The second type of results produced is the properly clustered, where examination of the clusters showed no breaking up of valid cluster into sub-clusters. These results still yielded lower than desired count estimations for High Object Density Videos (HODV) but significantly higher than the first type of results. The final set of results is such that the detected clusters are noisy leading to a lot of false positive locations which gives much higher object count estimation.

Whatever results were produced from the clustering results, object count estimations can still be determined. When dealing with the first results, some of the objects in the frames are skipped because their features are not in the same clusters as the training features. To discover the missing objects, the clusters that do not intersect the initial ROI are checked against the newly detected object locations and used to find new object locations to be added to the count. When dealing with bad clustering results, the localisations algorithm yielded a lot false negative locations. In this case, we use one of the contributions in this thesis to detect and remove such locations. When the ROI is first selected, a colour model learning algorithm is used to find the colours that best describe the object of interest. The colour model, which evolves from one frame to the next, is then applied to the detected object locations to detect and eliminate the false negatives.

The main contributions of this thesis are summarised as follows:

1. We designed a dataset for testing object counting algorithms. While the dataset currently contains two object classes, birds and cells, the bird class contains videos with various challenges common to object counting. The dataset contains videos in different categories with LMDB databases for the ground truths for each video. We also provide the code for extracting the ground truth. The structure of the dataset is such that it is easy to add more object classes.

2. We designed a simple yet fast algorithm for learning objects of interest features online. We designed the algorithm to be independent of the number of objects to be counted which allows our counting algorithm to handle any object. We also designed a state-of-the-art online algorithm for learning and tracking colours of objects of interest. While the colours are located all over the frame, the user interaction is independent of the number of colour points to track.

3. We demonstrate an efficient way for detecting multiple feature instances in video frames using density-based clustering. We also designed a way of analysing clustering results to determine the validity of the clusters. We show how the clustering is able to recognise similar features without specifying a measure of similarity between cluster points.

4. We developed a simple yet powerful algorithm for detecting the locations of objects as well as handling inter-object occlusions. The algorithm is able to recognise the location of the object even from only one matched feature. We developed ways in which to combine the feature learning algorithm, colour model tracking algorithm and matching algorithm in different ways to handle different variations in the videos.

1.3 Thesis Outline

The chapters highlighted below form the major contributions of this thesis.

- Chapter 2: We discuss the shortcomings of some available datasets. We also explain the dataset that was put together for this thesis. We explain the challenges that the dataset provides as well as the creation of the ground truth. The output of this chapter is a dataset that is used to test our counting approach.

- Chapter 3: We review and compare the performaces of single object trackers (SOTs) implemented in the OpenCV library. We also discuss the local feature extraction and detection algorithms used in this thesis. Finally we review the density-based clustering algorithms as they pertain to multiple feature matching used in this thesis as well as analysing cluster quality and validity of the cluster results.

- Chapter 4: We explore multiple feature instance detection as a density-based clustering problem. We demonstrate that the clusters detected within the local features have high degree of accuracy even as they fall below the accepted measure of matching quality. We also show how to explore various values of *minPts* and use the number of clusters detected and the validity of each clustering results to select the best value to use in the frame.

- Chapter 5: In this chapter, we introduce an approach to multiple object tracking that trains online with the user interaction independent of the number of objects to track. We show how to use it to track the colour model of the objects of interest which we later use to filter out false positives.
- Chapter 6: We describe the simple yet powerful approach of extracting the object instance locations from the clusters in Chapter 4. We use a combination of vector algebra and localised template matching to find object locations and localised moments to score the similarity between the template object and the located object instances. We demonstrate a simple and accurate method of training the template features for matching the sample object with the local feature clusters.
- Chapter 7: We discuss the results of using our approach to estimate the object counts on our dataset. We show how different parameters can be used to obtain a more stable count estimations. We compare the object count estimations from the three different approaches used in this thesis and show the average times needed to process each frame.
- Chapter 8: We provide an objective analysis of success and shortfalls of the approach in this thesis. Future improvements are suggested in this chapter as well.

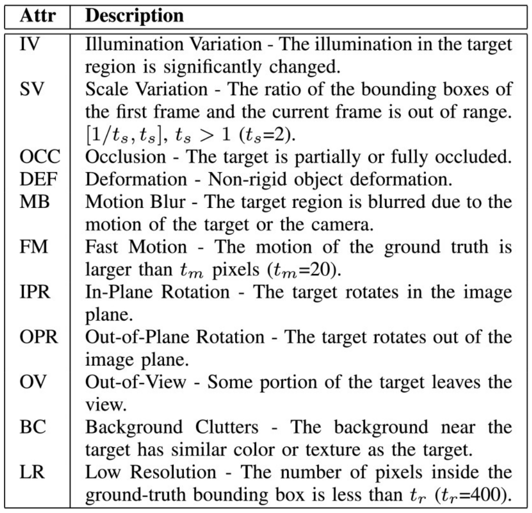
Chapter 2

Development of a Video Object Counting Dataset

Standardised datasets are a common tool for evaluating performance of object recognition, tracking and counting algorithms. Their widespread use highlight the importance of having common benchmarks on which to compare algorithm performances. Over the years, there have been different datasets and benchmarks developed for computer vision and image processing, but most of them are for tracking and recognition. There is little work on object counting datasets, let alone video object counting.

In object recognition, datasets have played a key role from the beginning. The Yale Face Database [36] was one of the early datasets that provided a collection of images of different facial expressions. As years went by, more facial recognition datasets were introduced including video based ones like Youtube Faces DB [37] and 300 Faces in-the-Wild [38]. These datasets are restricted in variety as they are about people's faces. As such, they are not well suited for the task of counting being detailed in this thesis.

Caltech-256 [39] provided labelled images for the task of object detection and classification. The dataset was designed only for object recognition and as such the images only contain a few objects per image. Recent years have seen video based datasets



Attr	Description
IV	Illumination Variation - The illumination in the target region is significantly changed.
SV	Scale Variation - The ratio of the bounding boxes of the first frame and the current frame is out of range. $[1/t_s, t_s]$, $t_s > 1$ (t_s=2).
OCC	Occlusion - The target is partially or fully occluded.
DEF	Deformation - Non-rigid object deformation.
MB	Motion Blur - The target region is blurred due to the motion of the target or the camera.
FM	Fast Motion - The motion of the ground truth is larger than t_m pixels (t_m=20).
IPR	In-Plane Rotation - The target rotates in the image plane.
OPR	Out-of-Plane Rotation - The target rotates out of the image plane.
OV	Out-of-View - Some portion of the target leaves the view.
BC	Background Clutters - The background near the target has similar color or texture as the target.
LR	Low Resolution - The number of pixels inside the ground-truth bounding box is less than t_r (t_r=400).

Figure 2.1: Different challenges presented by a dataset used in [42].

such as in [40], where the dataset was designed for video segmentation and contains variety of scenes including videos from moving cameras. As such, the dataset provides no benchmarks for object counting and also the object count in the dataset do not reach the numbers that we are interested in for this thesis. The dataset in [41] provides 150 video sequences with 376 annotated objects also dedicated to video object segmentation but has the same shortfalls as [40].

Object tracking benchmarks and datasets such as [42,43] and [44] were designed with modern object tracking challenges in mind. These datasets provide challenges and performance metrics for researchers to test their approaches and compare them to others. In [42], the dataset provides various challenges shown in Figure 2.1 which are also challenges encountered in object counting. However, the dataset is not geared towards multiple objects as such the number of objects in the videos is generally very low. The multiple object tracking (MOT) dataset MOT16 [43] provides a tracking dataset for people tracking aimed towards MOT approaches. The dataset however is not diverse enough for use in general object counting.

The widespread need for quantifying objects in videos has led to research across multiple disciplines. Lou *et al.* [45] developed a red blood cell counting approach based on spectral angle mapping and support vector machines. Venkatalakshmi and

Thilagavathi [46] undertook the same task by relying on Hough transforms to detect the circular shape of red blood cells. In both of these cases, the authors relied on images as their input but do not provide easy access to the images they used.

With urban settings being a major focus of a lot of object recognition and tracking in recent years, crowd counting has also seen increased research focus. Ryan *et al.* [47] proposed crowd counting by using multiple local features. They tested their approach on a large pedestrian dataset provided by Chan *et al.* [13]. The dataset lacks the diversity to test the different scenarios outside crowd monitoring. Yoshinaga *et al.* [48] used blob descriptors for real-time people counting. They tested their approach on the PETS2006 dataset [49] which has been used widely for object tracking approaches and provides good benchmarks, but the sequences are from stationary cameras and predominantly feature people.

While object counting is based on object recognition, and can be used with both object recognition and object tracking datasets, there has been a lack of focus on creating object counting datasets and benchmarks. Most of the datasets used in testing object counting algorithms are spatial based and do not allow temporal based analysis. The problem with using object tracking video datasets is that the benchmarks are mainly on tracking, not counting.

In this chapter, we set out to create a video dataset that is primarily designed to test object counting algorithms. The main aim is to provide videos that can be classified into three categories; low object density videos (LODV), medium object density videos (MODV) and high object density videos (HODV). For this thesis, we define LODVs as videos that contain less than 10 objects. With LODVs, a person can comfortably count the objects as the video is running. MODVs contain between 11 and 50 objects which allows people to make accurate estimation. HODVs contain more than 50 objects which is considerably harder to estimate especially when the number of objects reach hundreds. We created a dataset that for all these categories, there are different scenes, from simple backgrounds to complex and cluttered background.

The rest of this chapter is structured as follows; Section 2.1 explains the challenges the dataset provides for object counting algorithms. Section 2.2 details the process of creating the object count ground truth on a frame by frame basis. In Section 2.3, we explain the structure of the dataset including the video files and their basic statistical ground truth values.

2.1 Object Counting Dataset Challenges

In this section we discuss the challenges our dataset provides and the process of creating the ground truth for each of them. This dataset is freely available on github[1] under the GNU General Public License v3.0 and has 20 bird videos and 4 blood videos at the time of writing this thesis. The videos were downloaded from youtube and edited to reduce the lengths. When compiling the videos in this dataset, we aimed to include many of the challenges encountered by object recognition, object tracking and object counting algorithms. We also set out to include a lot of the videos where the camera is also in motion.

2.1.1 Inter Object Occlusion and Partial Visibility

Inter-object occlusion is as old a problem as computer vision. It appears in tasks such as object recognition and tracking. Since object counting is an extension of object recognition, it has also become an issue in object counting [50]. In terms of the object counting dataset, this created a ground truth problem where a person has to decide how much occlusion the two objects should have before they can no longer be counted individually. This is such a common occurrence that many of our videos have the challenge. When creating the ground truth, we opted to count the object as long as some identifiable part is visible. Figure 2.2 shows the occlusion problem in *voc-18-bd-1* video. Even though there are 3 birds, the occlusion is so much that an argument can be made for counting only 2 of them. However, there is still a significant part of the middle bird visible that we encourage counting it in the ground truth.

Partial visibility is similar to occlusion in that part of the object is invisible. However, unlike inter-object occlusion, the invisible part of the object is occluded by either the

[1]https://github.com/ojmakhura/voc-18.git

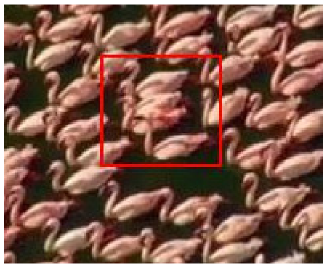


Figure 2.2: The occlusion challenge as seen in *voc-18-bd-1* video.

edge of the frame or some part of the background. Just like the occlusion problem, the challenge for both ground truth creation and counting algorithms is how much of the object should be visible to be considered. In our dataset, we counted every bird visible no matter how little it is visible. For the counting algorithms, this capability relies on the type of features used to identify the objects. Features that work at the pixel levels have a better chance of detecting small objects than those that use groups of pixels.

2.1.2 High Object Density

High object density presents a challenge because when there is a lot of objects, a person trying to count would have difficulty getting an accurate count. When objects start reaching hundreds, this can become a near impossible task as concentration lapse and fatigue become hindering factors. It also results in an almost certainty that there will be an occlusion problem. Often, the higher the density, the smaller the objects. This often means any object counting algorithm has to deal with fewer discernible features from the objects.

An example of this in our dataset is the *voc-18-bl-1* video as shown in Figure 2.3. The high density in the number of blood cells results in a lot of occlusion and truth count challenges. It is because of such videos that we created the human-computer hybrid ground truth extraction process; a person identifies and marks the objects and the computer counts the marks.

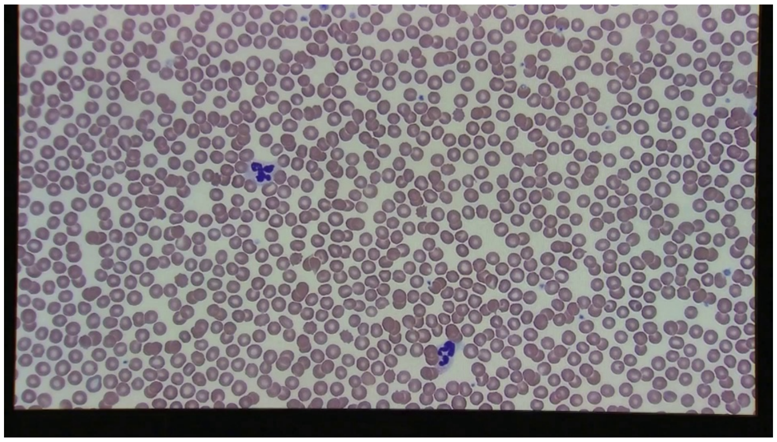


Figure 2.3: The high object density challenge from *voc-18-bl-1* video.

2.1.3 Scale

The problem of scale results when some of the objects of interest are close to the video recording equipment while others are further away. While the objects are the same, their size and amount of features available differs. The objects closer to the camera appear bigger while the ones further away appear smaller. As such, the success of object counting algorithms in detecting this relies on the features used to identify the objects.

The challenge in creating truth count comes from not knowing how small the objects should be before they are ignored. Figure 2.4 shows the scale problem in *voc-18-bd-14* video where some of the birds are flying and some of them are sitting down on the water further from the camera. This video actually presents a very extreme case of this problem as some of the birds are so small that they show no discernible features. In this thesis, the ground truth for this video was created by only tagged the birds which were flying as their features were clearly visible. But even some of the flying birds had to be ignored as they were too far from the camera.

2.1.4 Vanishing Point

Situations where the camera is tilted forward as it moves across the objects of interest does not only create a scale problem, it also creates a vanishing point problem.

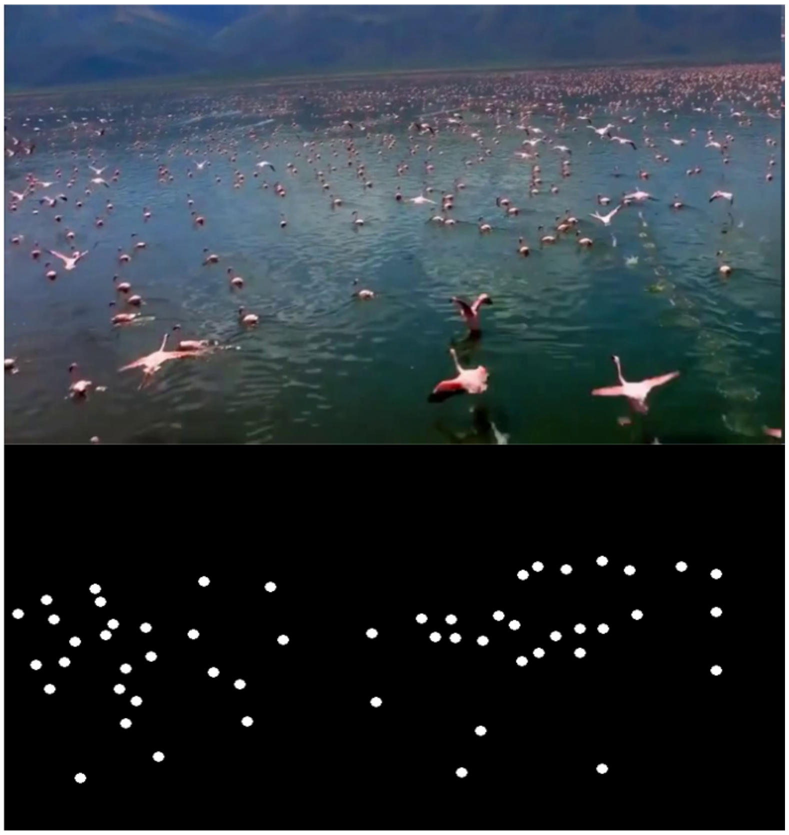


Figure 2.4: The scale challenge as seen in *voc-18-bd-14* video with the binary image representation of the ground truth.

Depending on the number of objects, the inter-object occlusion and partial visibility occurs. This challenge can be seen in Figure 2.5 where the video has so many birds that the vanishing point problem merges the birds into a mass of pink. This results in a problem for both ground truth and counting algorithms. It is impossible to estimate the number of birds in the pink mass. For this video, the ground truth was created by using a similar approach to the occlusion problem discussed above. We tagged the birds as long as they can be visually separated from the pink mass.

2.1.5 Complex Background

In this thesis, we refer to complex backgrounds as backgrounds that have a wide range of characteristics such as colours, texture and other objects. This complexity

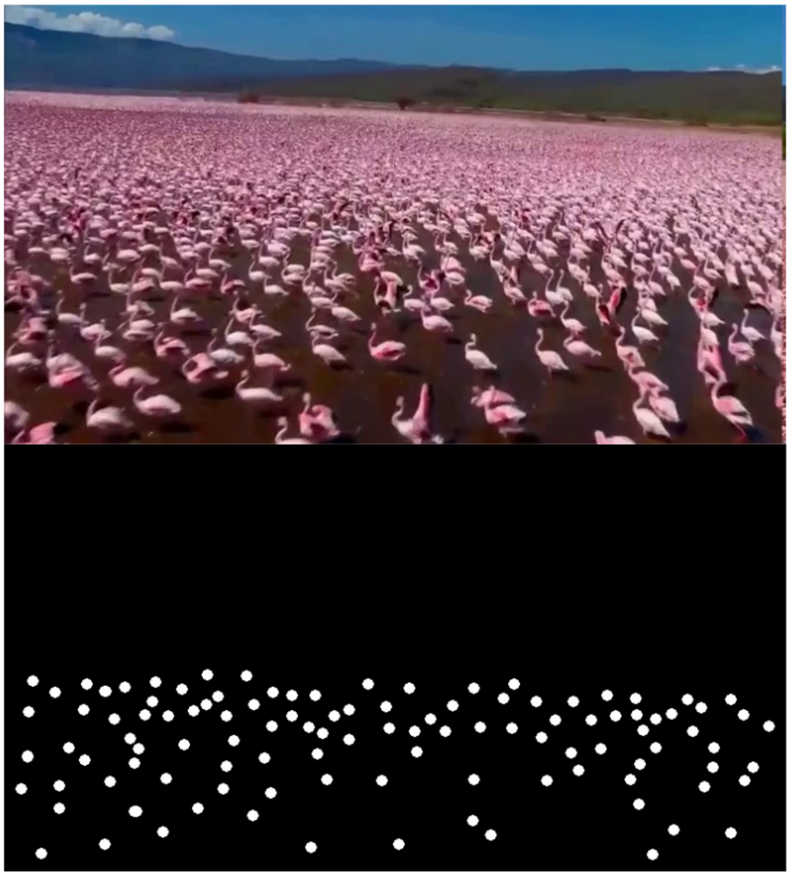
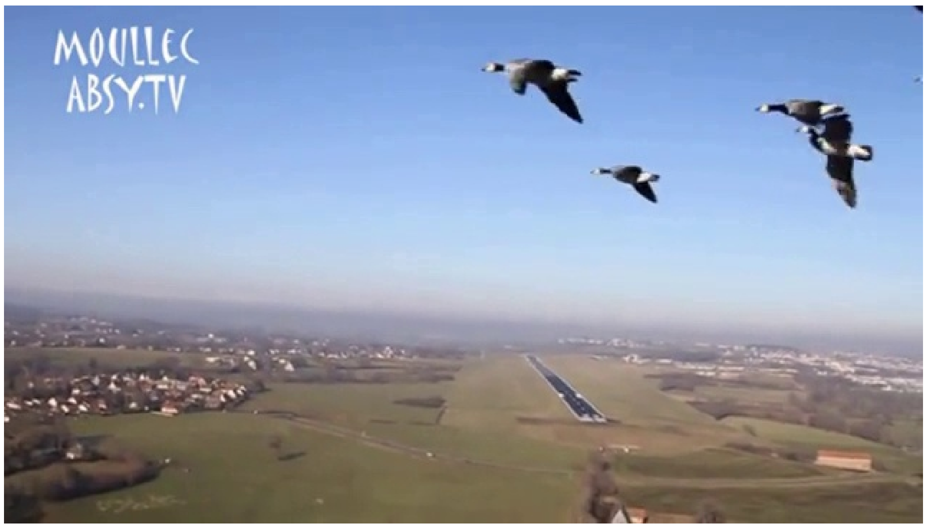
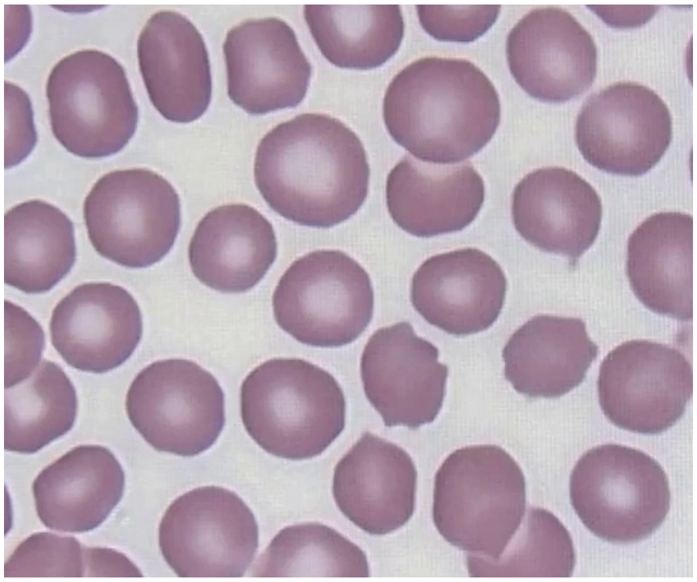


Figure 2.5: The vanishing point challenge in *voc-18-bd-18* video with the binary image representation of the ground truth.

becomes a problem when the features in the background are similar to some of the features in the objects of interest. An example of such situation can be seen in Figure 2.6. In Figure 2.6a, there is a lot of a black and white colours in the background which are also present on the ducks. This is also the case in Figure 2.6b which has the centre of the blood cell being very similar to the background. This problem does not affect the creation of the ground truth, the object counting algorithms would have to handle such problems and be able to separate the background from the foreground.



(a) The complex background from clutter in *voc-18-bd-6* video.



(b) Complex foreground in *voc-18-bl-3* video when the object colours are similar to the background.

Figure 2.6: Examples of complex background challenges in our dataset.

2.1.6 Complex Foreground

A complex foreground problem commonly appears when dealing with animate and deformable objects. In our dataset, flying birds offer this challenge and a lot of the videos have flying birds. The challenge is that since objects are deformable, they can have different appearances in the frames. Also since they are animate, they can rotate and change shape providing different features. As such, depending on how flexible the features used by object counting algorithms are, some of the objects can be missed.

This problem does not necessarily provide a challenge for ground truth extraction as humans are very good at identifying objects even poorly defined ones. In addition to showing scale challenges, Figure 2.4 also presents a good example of of complex foreground. While the birds sitting in the water are a good example of this problem, the flying birds also present the challenge depending on the position of their wings in the frame.

2.2 Ground Truth Benchmark

Extracting the ground truth from the video frames requires human interaction. While the option to use people to count the objects was available, it would suffer from the usual human related problems such as fatigue, mistakes and slowness, especially when counting objects in high object density videos such as in Figure 2.3. These problems were circumvented by using a hybrid human-computer process to extract the ground truth. This process relies on the ability of humans to identify accurately objects in a wide variety of situations including inter-object occlusions while computer algorithms can quickly and accurately count well defined objects. The task of getting well defined and accurately identified objects was done by a person. While a person has to actually identify the objects, they do not have to actually count them. Identifying the objects is considerably easier for humans than actually counting them.

In creating the ground truth, we first extract he frames from the video using Listing A.5. Each frame needs to be processed independently using an image editing software to tag each object with a white dot. The tagged image is then thresholded to remove all the other colours save for the white dots. The final product of this process is a binary image that can be processed by a computer to count the number of white dots on a black background (Figure 2.7).

The binary image representations of each video frame are saved in a single directory.

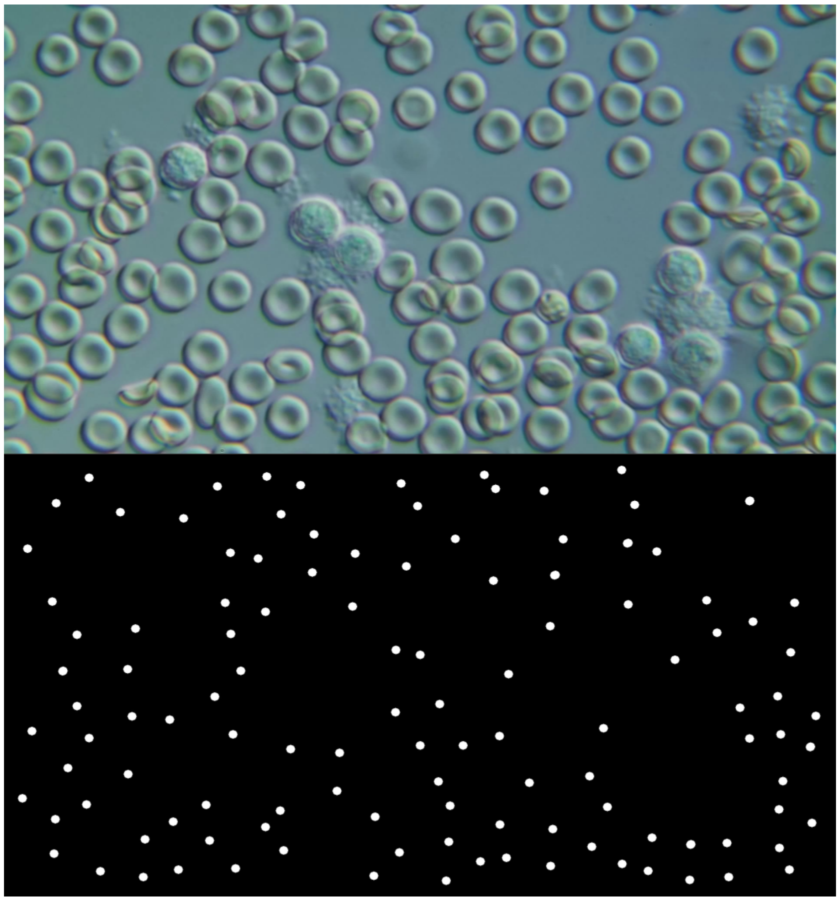


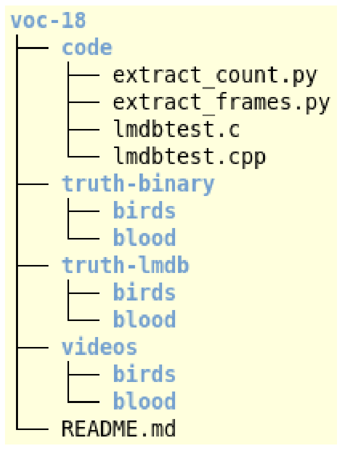
Figure 2.7: A frame and its binary version for video file voc-18-bl-4.

The algorithm for extracting the ground truth from them is connected components labelling (Listing A.6). The algorithm looks through the image and finds all white pixels that are neighbours and gives them the same label. Each group of pixels is given a different label. The highest label in the image represents the total number of white dots thus giving the ground truth for the number of objects in the video frame. The ground truths are stored in a Lightning Memory-Mapped Database (LMDB) [51] with the frame number as the key and the ground truth as the value. Even though the binary images are created by a person, there is always the chance that they might make a mistake. Therefore, when extracting the ground truth, we used a moving average over 10 frames to compensate for any human errors during object tagging.

2.3 Dataset Structure

This section describes the structure of the dataset as it appears on github. We also show the videos in the dataset and the statistical descriptions of the ground truth (See Tables 2.1, 2.2 and 2.3) as well as the challenges each video provides (See Table 2.4). Some of the videos have minimum and maximum values that fall in different categories. We therefore used the count average to decide the categories for each video. The statistical data can help object counting algorithms to measure their success. However, it is worth noting that these values are not only influenced by errors in human identifications of the objects, but also by the actual number of objects as the camera may be in motion, or the objects in motion or a combination of both.

The main directory contains four subdirectories. The *videos* directory contains the videos. The *truth-lmdb* directory contains the LMDB databases for the ground truth. The *truth-binary* directory contains the binary images for each of the videos. While we encourage using the LMDB databases, we include this directory in case the object counting algorithm does not have access to LMDB libraries. The inclusion of the binary images does not affect the size of the dataset much since the JPEG compression algorithms works very well for such low detail images. Each of the *videos*, *truth-lmdb* and *truth-binary* has *birds* and *blood* subdirectories for the different classes of objects in the dataset. This structure allows for a simplified addition of new object classes.



```
voc-18
├── code
│   ├── extract_count.py
│   ├── extract_frames.py
│   ├── lmdbtest.c
│   └── lmdbtest.cpp
├── truth-binary
│   ├── birds
│   └── blood
├── truth-lmdb
│   ├── birds
│   └── blood
├── videos
│   ├── birds
│   └── blood
└── README.md
```

Figure 2.8: The structure of the VOC-18 dataset as it appears on github.

Table 2.1: Low Object Density Videos

Video	**Number of Frames**	**Min**	**Max**	μ	σ
voc-18-bd-4	65	4	6	5.49	0.77
voc-18-bd-5	56	2	4	3.09	0.95
voc-18-bd-6	37	3	5	3.89	0.91
voc-18-bd-7	87	3	6	4.82	0.795
voc-18-bd-8	85	7	9	7.98	0.48
voc-18-bd-9	143	3	7	5.54	1.08
voc-18-bd-11	155	4	8	5.68	0.21
voc-18-bd-20	93	3	3	3	0

Table 2.2: Medium Object Density Videos

Video	**Number of Frames**	**Min**	**Max**	μ	σ
voc-18-bd-2	71	4	23	11.23	5.43
voc-18-bd-10	121	9	18	13.82	2.96
voc-18-bd-16	75	32	40	35.43	1.03
voc-18-bd-19	117	25	37	33.33	3.05

Table 2.3: High Object Density Videos

Video	**Number of Frames**	**Min**	**Max**	μ	σ
voc-18-bd-1	78	257	277	266.69	6.35
voc-18-bd-3	103	84	106	92.39	6.68
voc-18-bd-12	73	211	322	296.34	29.63
voc-18-bd-13	99	47	62	51.83	3.89
voc-18-bd-14	110	48	83	69.23	11.73
voc-18-bd-15	115	33	62	52.23	7.23
voc-18-bd-17	85	106	125	116.45	4.47
voc-18-bd-18	97	87	106	97.57	3.81
voc-18-b1-1	73	698	722	698.93	3.36
voc-18-b1-2	94	460	460	460	0
voc-18-b1-3	80	167	186	176.49	6.49
voc-18-b1-4	49	115	130	122.89	5.13

Table 2.4: Challenges posed by each video.

Video	**Occlusion**	**Scale**	**Vanishing Point**	**Background**	**Foreground**
voc-18-bd-1	✓	✗	✗	✗	✗
voc-18-bd-2	✓	✗	✗	✗	✗
voc-18-bd-3	✓	✗	✗	✗	✓
voc-18-bd-4	✗	✗	✗	✓	✓
voc-18-bd-5	✓	✗	✗	✗	✗
voc-18-bd-6	✗	✗	✗	✓	✓
voc-18-bd-7	✓	✓	✗	✗	✗
voc-18-bd-8	✓	✓	✗	✓	✓
voc-18-bd-9	✓	✓	✓	✓	✓
voc-18-bd-10	✓	✗	✗	✗	✗
voc-18-bd-11	✓	✓	✗	✓	✓
voc-18-bd-12	✓	✗	✗	✗	✓
voc-18-bd-13	✓	✓	✓	✗	✓
voc-18-bd-14	✓	✓	✓	✓	✓
voc-18-bd-15	✗	✓	✓	✗	✓
voc-18-bd-16	✓	✓	✓	✓	✓
voc-18-bd-17	✓	✓	✓	✗	✓
voc-18-bd-18	✓	✓	✓	✗	✗
voc-18-bd-19	✓	✗	✗	✓	✗
voc-18-bd-20	✗	✗	✗	✗	✗
voc-18-b1-1	✓	✗	✗	✓	✗
voc-18-b1-2	✓	✗	✗	✗	✓
voc-18-b1-3	✓	✗	✗	✗	✓
voc-18-b1-4	✓	✗	✗	✓	✓

2.4 Conclusion

In this chapter, we have presented a video object counting dataset with a simple benchmark which we use in the following chapters to test different parts of our object counting approach. We explained the novel process by which the ground truth benchmark was extracted even for HODVs and how we counteract the human errors that might be introduced during the process by using moving averages. The dataset structure was also explained and it is also available freely.

In Section 2.1, we presented the challenges that the dataset offers. We aimed to compile the dataset that provided the main challenges common to computer vision. The dataset provided six challenges which were explained in terms of both the object counting algorithms and ground truth creation. We noticed that some challenges such as complex foreground and occlusion affect the counting algorithms and ground truth differently. Other challenges such as vanishing point and scale offer the same challenges for both algorithms and ground truth the same.

The creation of the ground truth was covered in Section 2.2. We presented a novel human-computer hybrid approach to creating frame by frame ground truth. We leveraged the incredible object recognition ability of humans and the counting speed of computers. A person creates binary image representations of the video frames and using connected components analysis we extract the frame object count. We employed a ten frame running average which we found offsets any errors a person may make identifying objects on a frame which means only one person can be used per video.

The dataset structure describes in Section 2.3 offers an easy way to add more object classes. We believe the simplicity yet well laid out structure will help in further developments on the dataset. We provided python code for extracting frames from videos and for counting the objects in the binary images and creating the LMDB database. We also provided C and C++ code for reading the LMDB database.

At the time of writing this thesis, the dataset had 24 videos. In the future, we plan to increase the number of videos as well as the classes of objects to be counted. This will give an opportunity to bring different combinations of the object counting challenges. We also plan to introduce multi-class counting where a video contains more than one type of object. The other powerful benchmark that will be included in the future is the number of distinct objects in the video. This benchmark will require algorithms to utilise long-term multi-object tracking to track objects that come in and out of video frames. In this case, the ground truth will be a single number defining how many distinct objects have been encountered in the videos. The usual benchmarks for object tracking will also be included in terms of the object locations. This benchmark will provide the ability of the the algorithms to determine how accurate they can locate the objects

Chapter 3

Object Tracking, Local Features and Density-Based Clustering

In this chapter we discuss the research literature that is key to the different parts of the object counting approach outlined in this thesis. We start by looking at the single object tracking (SOT) algorithms implemented in the OpenCV [52] library. We then review some of the multi object tracking (MOT) literature to highlight the main weaknesses of available approaches. We do not do any experimental comparisons, but rather highlight the problems that we will address when we explain the design of our own MOT in the coming chapters.

Then we review local feature extraction and feature matching. As with object tracking, the focus is on OpenCV implementations, specifically, Scale Invariant Feature Transform (SIFT) [53] and Speeded-Up Robust Features (SURF) [24]. These two algorithms have been shown to be very resilient to scale, lighting and rotational variance. The matching algorithms reviewed are the brute force matcher and the Fast Library for Approximating nearest Neighbours (FLANN) [54].

The final review section looks at density based clustering as well as explain why it is important in this thesis. The algorithms reviewed in this section are the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [55] and its hierarchical extension HDBSCAN [35]. While DBSCAN is included because it is the main feature matching algorithm for the work in this thesis, DBSCAN is included to give context to HDBSCAN since it forms the basis on which HDBSCAN is built. HDBSCAN had already been implemented in python and Java, but since our work in this thesis was in C++, we wrote a new C/C++ implementation with python and Java bindings.

3.1 Single Object Tracking

As highlighted in Section 1.2, we want to count an unknown number of unknown objects while keeping the user interaction independent of the number and type of objects being counted. However, some information needs to be provided by the user on the objects of interest. In this thesis, the information is provided as a rectangle around one of the objects of interest in the video frame. We then employ an SOT [56, 57] to track the object of interest from one frame to the next. This approach ensures that the user interaction is kept to only one frame in the video.

We tested the SOTs implemented in OpenCV on our dataset which we discussed in Chapter 2. In our tests, we looked at the ability of the SOTs to properly track the objects with minimal drift. We also looked at the tracking speed of the algorithms. In this section, we give an overview of the SOTs and show the final tracking times and discuss the tracking failures and successes as well as the algorithm of choice for this thesis.

3.1.1 BOOSTING

Real-time tracking via Online boosting was proposed in [58] to address two main challenges to object tracking; variation of appearance of target and coping with background clutter. The given region of interest (ROI) is treated as the positive sample while negative samples are extracted from same size regions around the positive sample. The algorithm uses a version of online Ada Boost algorithm to update the tracking features. For their paper, they used Haar-like features [33], orientation histograms [59] and local binary patterns [60]. The approach generates a set of weak classifiers, which have to only perform slightly better than random guessing. The weak classifiers are grouped together into a single “global weak classifier” pool. The pool is then shared among a set of selectors. Each selector is randomly initialised with a set of weak classifiers from the pool and selects one of them. The selected weak classifiers are then linearly combined to create a strong classifier.

3.1.2 MIL

Multiple Instance Learning (MIL) tracker [61, 62] addresses the challenge of how to select positive and negative samples when updating the appearance model that is encountered by a lot of trackers that attempt to model both the background and the foreground. The authors argue that it is difficult for a human labeler to be consistent with respective to how the positive and negative samples are selected. As such, they propose to use MIL to remove the ambiguity from a person to the learning algorithm.

MIL tracker is composed of three components: image representation, appearance model and the motion model. The image representation is made up of Haar-like features computed for each image. The appearance model is made up of discriminative classifiers that can select the image patches most likely to contain the object of interest. The motion model is used to update the tracker location which is then used to update the object model.

3.1.3 KCF

Henriques *et al.* [63] presented a tracker that uses discriminative learning methods, but, unlike other trackers, has linear complexity. The virtually limitless amount of negative samples often leads some trackers to under-sample which limits their performance. In their paper, the use of the Fourier domain allowed the authors to demonstrate a tracker that is based on kernel ridge regression but does not suffer from the “curse of kernelisation”.

Using convolution filters is another way the algorithm achieves fast tracking capabilities. This is because convolving two image patches is equivalent to element-wise product in the Fourier domain. Using cyclically shifted samples for ridge regression, a connection is made with correlation filters which enables fast learning at *Olog(n)* with Fast Fourier Transform (FFT) instead of expensive matrix algebra.

At the centre of the KCF tracker is the use of the "kernel trick" which allows for the use of the more powerful, non-linear regression, the draw-back of which is increased complexity. In order to achieve fast kernel regression, the paper provides proof that the kernel matrix is circulant for datasets of cyclic shifts. Since this does not hold in general, the paper imposes Theorem 1.

Theorem 1 *Given a circulant data, $C(x)$, the corresponding matrix K is circulant if the kernel function satisfies $k(x, x^{'}) = k(Mx, Mx^{'})$ for any permutation matrix M.*

3.1.4 CSRT

In [64], a discriminative correlation filter (DCF) with spatial and channel reliability is proposed. Standard DCF trackers use FFT for efficient learning which brings a restriction that the filter and the search region size should be equal. Efforts to counteract this limitation such as learning the filter from a larger region degrades the performance of DCF trackers. Another limitation of DCF tracker is the assumption that the target shape is well approximated by the axis-aligned rectangle.

In the paper, the authors introduce a spatial reliability map which adapts the filter to the part of the object worth tracking thus overcoming the problems of circular shift thereby allowing for arbitrary search region size and the rectangular assumption. They also introduce channel reliability to filter design which is then used to weight the per-channel filter responses in localisation.

3.1.5 Median Flow

Median flow tracker was introduced in [65] with the ability to detect tracking failures. The tracker uses the forward-backward consistency assumption that correct tracking is independent of the direction of time-flow. The tracker operated in 3 steps:

1. Given the location of a target object at time t a forward trajectory is estimated at time $t + 1$ to obtain a new object location.

2. A validation trajectory is obtained from the $t+1$ estimated location by tracking the location back to t.
3. The two trajectories are compared, if they differ significantly, the tracker is proved to have failed.

3.1.6 MOSSE

Minimum Output Sum of Squared Error (MOSSE) [66] tracker is a regularised version of the Average of Synthetic Exact Filters (ASEF) [67] that uses adaptive correlation filters to model the target and tracks it via convolution. The review of ASEF is outside the scope of this thesis. As such, the interested reader is directed to the full article on ASEF for full details.

Given a target, MOSSE trains by correlating the filter over a search window in the next frame and extracts the location corresponding to the maximum value in the correlation output. Correlation is an element-wise multiplication in the Fourier domain, as such, the correlation output G, takes the form $G = F \odot H^*$, where F is the 2D Fourier transform of the image H^* is the complex conjugate of the Fourier transform of the filter. Including the transformation back to spatial space of the correlation output, the online process is $O(PlogP)$, where P is the number of pixel in the tracking window.

The FFT convolution algorithm maps the image and the filter to a torus topological structure. The effect is then reduced by transforming the pixels with a log function, normalising them to $0.0 - 1.0$. The results are then multiplied by a cosine window to gradually reduce pixels near the edge to 0 which has an added advantage of putting the emphasis on the centre.

3.1.7 TLD

In long-term tracking, the main challenge is the detection of the target when it leaves the camera view and then reappears. Also, the object may have changed shape, which makes previous detection information irrelevant. In [68], a new approach called Tracking-Learning-Detection (TLD) is proposed that approaches long-term tracking by decomposing the problem into three components.

The first component is the tracker, which makes use of the assumption that the target moves smoothly through the video and that it is visible. Using the appearance information learned from the past, the detector scans the whole frame to localise the appearances. As can be expected, the detector will produce false positives as well as false negatives. The learning component of TLD observes the performance of both the tracker and the detector and estimates detector errors. By assuming that both the detector and the tracker can fail, the learning component can generate training examples which help the detector to generalise to more target appearances while avoiding similar errors in the future.

In frames where there are multiple target instances, some of those instances may be part of the false positive detections by matching the previous appearance information. This behaviour can be seen in Figure 3.1 where in three consecutive frames, the detector localised to three different target instances. An improvement on this could be for the detector to use the same assumption of smooth motion used by the tracker to give higher priority to instances detected close to the the previous known location.

3.1.8 GOTURN

Generic Object Tracking Using Regression Networks (GOTURN) [69] is a neural network based generic SOT. While most deep learning SOTs are slow, GOTURN has been tested by the authors at 100 fps. The improvement in speed is due to two factors. First, GOTURN is trained offline, which in itself is a restriction to the versatility of the algorithm. During testing, the learned weights are frozen and no fine tuning is necessary during tracking. Second, instead of classifying different image patches to find the best match, GOTURN uses a feed-forward pass through the network to regress directly to the object. Both the OpenCV and author implementations of the tracker are based on the Caffe [70] architecture. In this thesis we tested the OpenCV implementation and we use the same weights provided by the authors of the paper.

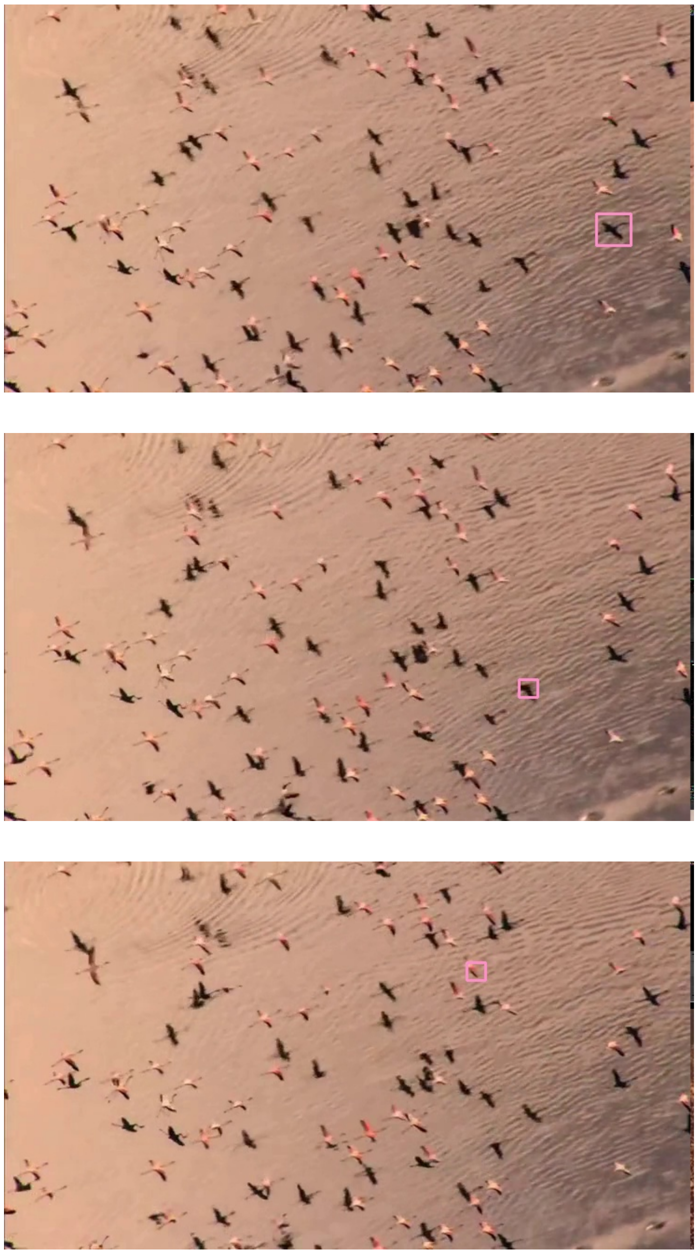


Figure 3.1: The detector "jumping" problem in TLD with *voc-18-bd-3* video.

3.1.9 Single Object Tracker Comparisons

In our time tests, we used the videos from our datasets discussed in Chapter 2, but here, we show the average Frames-per-Second (FPS) that each method was able to process for the *voc-18-bd-1* video. As can be seen in Table 3.1, MOSSE provided the fastest tracking while MEDIAN FLOW was the slowest. BOOSTING tracker ran at 419 FPS while GOTURN was running at 30 FPS. With GOTURN, the speed was way lower than the speed advertised in the paper, but this is attributed to not using the GPU for running the tracker.

In terms of tracker drift, BOOSTING provided the most stable tracking while still running at 99 FPS. The design of the algorithm to handle variation in appearance and background clutter is evident in *voc-18-bd-11* (see Figures 3.2a and 3.3a) in which the birds are in flight and flapping their wings which dramatically changes their appearance and the background is also complex. However, the tracker managed to keep track of the object even through some partial occlusion, only completely loosing to full occlusion.

GOTURN had the worst drift even with videos that have simple background and foreground. This is because we did not retrain it for our dataset. In a test with *voc-18-bd-1*, the tracker started drifting on the second frame and completely failing to detect the object by frame 22. In comparison, the next worst performer in that video was TLD, which did not fail to provide an estimation of the object location, but rather drifted off the object. As a deep learning based tracker, the problem of drift for GOTURN can be solved by more training, however, that is out of scope of this thesis.

Other trackers had varying degrees of successes and failures for various videos. MOSSE and MIL trackers showed promising stability in tracking but still had some localisation and drifting problems. In Figure 3.2, MIL showed some vulnerability by drifting off the object of interest while BOOSTING and MOSSE did not. However, the tracker then adjusted to one of the other birds and continued to track it properly. Figure 3.3 show the drifting problem for both MOSSE and MIL. In the figure, MOSSE has lost the actual object and the ROI is now off the target while MIL is still tracking

Table 3.1: Average FPS of the OpenCV SOTs using the *voc-18-bd-1* video.

MIL	**BOOSTING**	**MEDIAN FLOW**	**TLD**	**KCF**	**GOTURN**	**MOSSE**	**CSRT**
16.814	99.46	419.112	11.633	442.68	30.715	4464.85	48.976

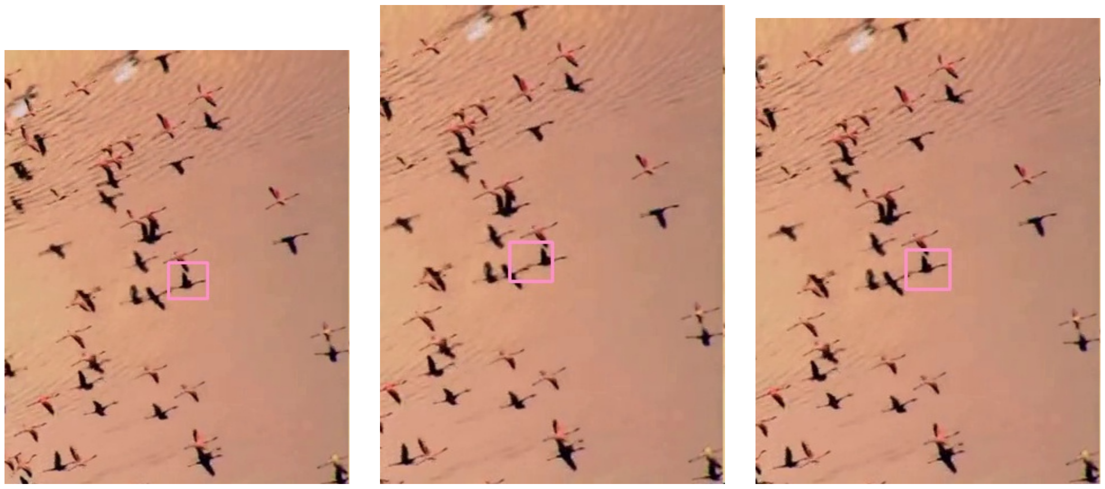
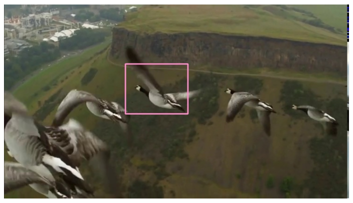
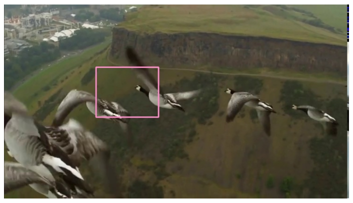

(a) BOOSTING tracker at frame 59 of *voc-18-bd-3*.
(b) MIL tracker at frame 59 of *voc-18-bd-3*.
(c) MOSSE tracker at frame 59 of *voc-18-bd-3*.

Figure 3.2: Comparisons of BOOSTING, MIL and MOSSE on *voc-18-bd-3* video.

just some part of the object. The problem of appearance variation is evident in a lot of the videos. This can be seen in Figure 3.4 where the CSRT tracker was switching between two ducks in order to preserve the appearance. As such, in this thesis, we use BOOSTING tracker, although MIL and MOSSE can work well.


(a) BOOSTING tracker at frame 58 of *voc-18-bd-11*.


(b) MIL tracker at frame 58 of *voc-18-bd-11*.

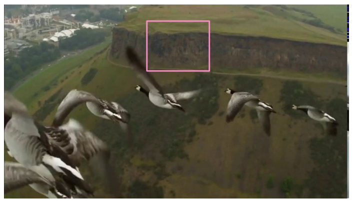
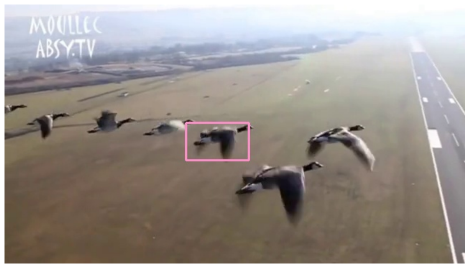
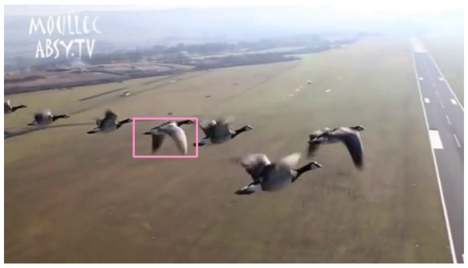

(c) MOSSE tracker at frame 58 of *voc-18-bd-11*.

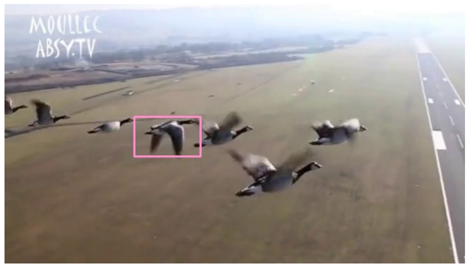
Figure 3.3: Comparisons of BOOSTING, MIL and MOSSE on *voc-18-bd-11* video.



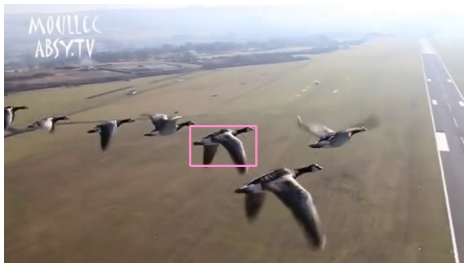
(a) CSRT tracker at frame 8 of *voc-18-bd-9*.



(b) Frame 9: CSRT losing the original target object in frame 9 because it changed appearance.



(c) Frame 10: Still on the same object because the appearance has not changed yet.



(d) Frame 11: CSRT tracker changed back to the original target.

Figure 3.4: The susceptibility of CSRT to appearance variations.

3.2 Multiple Object Tracking

Tracking algorithms built around single object tracking discussed in the previous section rely on the user-given ROI around the object. Unique features are learned from that ROI and used to track the occurrence of the object in subsequent video frames. The obvious downside to applying this training to multiple object tracking is that the user has to provide more ROIs which can become a big challenge when the objects number in the dozens and downright impractical when they reach hundreds. The advantage of these algorithms, however, is that the object to be tracked can be learned on the fly making them more adaptive to new objects.

The other approach is to train the objects to be tracked offline. The features learned can then be applied online to track the objects. Haar-like features [71] are one such algorithm that have gained widespread use. Deep learning approaches [72–74] have also gained traction for multi-object tracking and offer some of the best performances in the field. While these approaches often achieve state of the art performances with accurate object detection and tracking, the need for offline training is still a major problem. The objects to be tracked have to be known beforehand, which usually requires a lot of data on the objects.

Recent publications such as Zhang et al. [72], Wancun et al. [75] and Chen et al. [73] have demonstrated how convolutional neural networks can provide state of the art performances in multi-object tracking. Deep learning platforms such as YOLO [74], Caffe [70] provide a way of simplifying the training process. However, they introduce a problem of limiting the number of objects to be tracked to only those that have been trained on. Any new objects require collecting a lot of data on the new object including a lot of time to train on the new model. The offline nature of these approaches is often due to the time it takes to train and the computational power requirements.

Wu et al. [76] leveraged the power of discriminate correlation filters (DCF) [66] and the Markov decision process (MDP) to develop an MOT. The use of DCF provides resilience to occlusion and scale variation in addition to improved accuracy in single object tracking. They use MDP to integrate two DCF based trackers into the multi-object tracker and address the update problem of the appearance model.

Lan et al. [77] propose an MOT approach that exploits interactions between tracklets. They introduce *close* and *distant* tracklet interaction. *Close interaction* imposes physical constraint on the temporally overlapping tracklets and *distant* interactions handles appearance and motion consistency between two temporally separate tracklets. While both Lan *et al.* and Wu *et al.* as well as scores of other publication in this field do obtain good performances on dozens of objects, they do not address the issue of initialisation of targets or propose ways of simplifying that process when the number of objects to be tracked reaches hundreds.

Other researchers have attempted to address the MOT problem of initialisation in a variety of ways. In [78], Türetken *et al.* proposed a way of tracking elliptical cell populations in videos by using image segmentation and elliptical fitting to find cell candidates. They create a hierarchy of these candidates and use network flow integer programming to select the most temporally consistent cell candidates. While this approach promises good results for cells, it does not generalise well to arbitrary shapes especially when dealing with live objects which can change shapes in videos.

Object trajectories have been used to successfully track multiple objects. Wang *et al.* [79] applied generalised minimum cost flows (MCF) algorithm to jointly optimised consecutive batches to generate a set of conflicting trajectories. They then apply MCF again to obtain optimal matching between trajectories from consecutive batches. Maksai *et al.* [80] used a non-Markovian approach to impose global smoothness constraints on the trajectory segments. The main weakness of trajectory based methods is the loss of visual features of the objects such as colours and texture. This loss of information means that differentiating objects from their motion can be a challenge. Another challenge is that they can only perform well in motionless cameras as any motion from the camera will make all the objects including the background to generate trajectories.

3.3 Local Feature Extraction and Matching

Local feature detectors and descriptors have formed an important part of image processing with wide spread use from image representation, image classification and retrieval, motion tracking and object recognition [81]. Over the years there has been a lot of detectors and descriptors developed. The detectors considered in this thesis are SIFT [34] and SURF as they have been proven to outperform other local feature detectors and descriptors in terms of time and computational requirements [82] while SIFT performs better in terms of accuracy in matching [83]. Between the two, SURF has been shown to be faster than SIFT, in part because SURF feature descriptors are 64 dimension vectors while SIFT's are 128, therefore, SURF is the algorithm that was used for experiments in this thesis. SURF finds keypoints by using a basic Hessian matrix approximation which is achieved by finding blob features where the determinant is maximum. This matrix is shown in Eqn 3.1.

$$\text{Hess}\,(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & -L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \tag{3.1}$$

$L_{xx}(x,\sigma)$ is the convolution of Gaussian 2^{nd} order derivative $\frac{d^2}{dx^2}g(\sigma)$. The determinant of the approximate Hessian matrix is then calculated as $det(Hess_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2$ where w is the relative weight of the filter responses, and D_{xx}, D_{yy}, D_{xy} are the approximations of Gaussian with $\sigma = 1.2$ and represent the lowest scale for computing blob response maps.

SURF descriptors describe the distribution of intensity content within the neighbourhood of the interest point. Each descriptor is extracted by overlaying a 4x4 grid over the interest point as shown in Figure 3.5. For each square, a Haar wavelet response is calculated in the horizontal (d_y) and the vertical (d_x). Their absolute values ($|d_y|$ and $|d_y|$) are also calculated to get the information about the polarity of the intensity changes. This leads to each square having a vector $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. With 16 squares, this leads to a keypoint descriptor vector with 64 dimensions. A consequence of this is that our counting approach relies on the presence of the features, and since SURF features are not at the pixel level, it also affects the sizes of the objects we can detect.

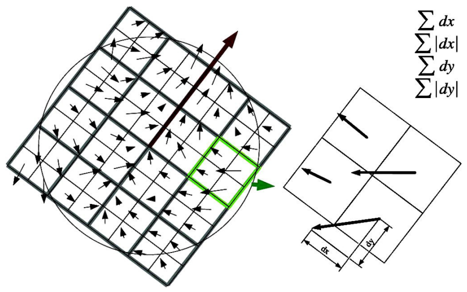


Figure 3.5: Oriented quadratic grid with 4x4 square sub-regions is laid over the interest point and the $\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|$

In OpenCV, running SURF on a frame gives two sets of data; the keypoints, and the descriptors. The keypoints provide information on the location of the feature, the angle from which the feature is calculated, and the size of the meaningful keypoint neighbourhood while descriptors are 64 dimension vectors describing the distribution of intensity within the feature size given by the keypoints. For the rest of this thesis, the following notations are used:

- $\mathcal{F}_i$: the i^{th} frame where $i \in \{1, ..., L\}$ and L is the length of video.
- $\mathcal{X}_i = \{(\vec{k_1}, \vec{d_1}), ..., (\vec{k_N}, \vec{d_N})\}$: SURF local feature for frame i. k is the keypoint, d is the corresponding descriptor and N is the number of features in the frame.
- $\vec{k} = \{p, a, s\}$: p is the location of the feature within the frame, a is the angle and s is the size.
- $\vec{d} = \{d^1, ..., d^{64}\}$ is the descriptor vector.
- $\vec{\mathcal{K}_i} = \{k_1, ..., k_N\}$: a set of keypoints in frame i.
- $\vec{\mathcal{D}_i} = \{d_1, ..., d_N\}$: a set of descriptors in frame i.

In this thesis, we describe the basis of the object counting approach in this thesis which uses local features for object recognition and matching. We achieve this by matching features of the object of interest to other features from the object instances in the video frame. Commonly, brute-force and Fast Library for Approximating

Nearest Neighbours (FLANN) [84], are used to match feature descriptors from a query set to feature descriptors from the training set.

Brute-force matcher offers the simplest feature matching approach. This algorithm finds the distance between training and query features and returns best matches by selecting the smallest distances. The distance is normally calculated using the L1 or L2 norm, although other distance metrics can be used as well. The problem with this approach is that the matches are only selected because they are the closest match, not because they are correct. FLANN is a suite of algorithms that was designed to solve nearest neighbours matching in high-dimensional spaces. This approach though, still requires the user to specify the degree of precision for the nearest neighbour estimation. The common approach is to use Lowe's Ratio Test [53] which takes the distance of the closest match and the second closest match. If the ratio is above 0.75, the detected match is accepted.

In this thesis, we are looking to match multiple feature instances without the burden of providing the degree of precision as required by FLANN or resorting to one-to-one matching of brute-force matcher. We view this task in terms of the distribution of the features in the descriptor space and as such, we develop Hypotheses 1 and 2 below:

Hypothesis 1 *: Given multiple object instances in a frame, SURF will detect a set of descriptors, $\mathcal{D}$, such that matching descriptors will form clusters within the 64-dimension descriptor space.*

Hypothesis 2 *: The clusters within $\mathcal{D}$ can have distance ratio below Lowe's Ratio Test while maintaining high degree of similarity.*

3.4 Density Based Clustering

The clusters alluded to above can therefore be detected using cluster recognition algorithms. In this section, we discuss a density-based cluster detection algorithm that we use to match multiple local features. The criteria we used to select the algorithm to use was driven by two problems:

1. We do not know how many local features will be detected and how many clusters

will exist within the descriptor space.

2. We do not know the density of the points within each cluster.

These two problems dictate that the algorithm to be used must be 1) able to find the clusters without any supervision on the number of clusters, and 2) able to find clusters of varying densities. Therefore, the algorithm must have parameters that have nothing to do with the structure of the data. In this section, we discuss the theory behind the density based clustering algorithms of DBSCAN and HDBSCAN and explain their importance to the work in this thesis. We also discuss the most commonly used clustering algorithm and explain why it is not suitable for the work in this thesis.

3.4.1 K-Means

k-means [85] cluster recognition algorithm is fast, easy to understand and has been widely implemented. It works by requiring a parameter k, which denoted the expected number of clusters. Then it randomly selects k points as initial cluster centroids and assigns the dataset points to the nearest centroid while also optimising centroids by taking the mean of all the points in the clusters.

As popular as *k-means* is, it is not a cluster detection algorithm, but rather a data partitioning algorithm. It simply assigns the data points to one of the k expected clusters. In this thesis, it is impossible to know how many clusters are in the descriptor dataset and therefore cannot select the proper value of k. *k-means* also does not have a concept of 'noise', therefore all the data points will be assigned to one of the k clusters. In a set of descriptors, some of the data points will not have any matches and must be labelled as such. The random initialisation of the first data points also means that for different initial points, one can get different clustering results.

In comparison, both DBSCAN and HDBSCAN require parameters that have nothing to do with number of clusters in the dataset. Furthermore, they have an inbuilt capability to recognise noise features. Another important feature is that they are able to recognise arbitrarily shaped clusters whereas *k-means* clusters assumes the clusters are globular.

3.4.2 DBSCAN

DBSCAN [23] is a density-based clustering algorithm that was designed for finding clusters in large spatial databases. It works by detecting clusters when there are at least a certain number of points, *minPts*, within a certain distance ε, of a certain point, p. The algorithm therefore requires *minPts* and ε as parameters. Given a spatial database D, the following definitions are given (See Figures 3.6 and 3.7):

Definition 1: *ε-neighbourhood* This is defined as $N_\varepsilon(p) = \{q \in D | dist(p, q)| \leq \varepsilon\}$. This definition gives rise to *core points*, which are points whose ε-neighbourhood contains at least *minPts*. The points that are within that neighbourhood but are not core points are referred to as *border points*.

Definition 2: *directly density-reachable* A point p, is directly density-reachable from a point q if p is in the ε-neighbourhood of q and there are atleast *minPts* in that neighbourhood.

Definition 3: *density-reachable* A point p, is density-reachable from a point q if there are points $p_1, ..., p_n$ where $p_1 = q$ and $p_n = p$ such that p_{i+1} is directly density reachable from p_i.

Definition 4: *density connected* A point p, is density-connected to point q if there is a point o such that both p and q are density-reachable from o.

Definition 5: *Cluster* A cluster C is a non-empty set of the database D such that all points in C are density-reachable and density connected to each other.

Definition 6: *Noise* For all clusters in D, *Noise* is a set of points that do not belong to any cluster.

The main weakness of DBSCAN is the parameter ε which essentially describes the acceptable cluster densities [35]. Even when dealing with low dimension data, selecting the best ε is a challenging task. In our case, we are working with 64 dimensions in the case of SURF or 128 dimension in the case of SIFT, which makes it even more challenging.

Ye et al. [86] applied a modified version of DBSCAN to image segmentation. In their

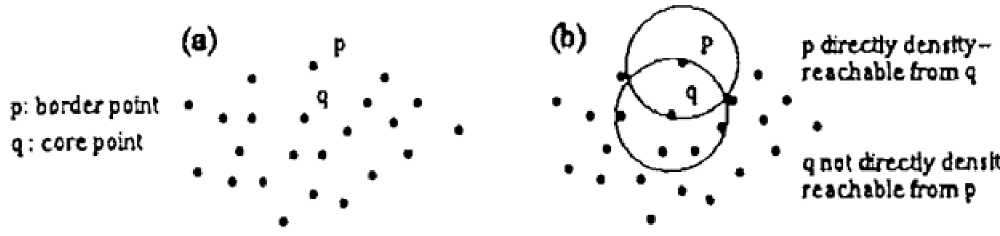

(a)
p
q
p: border point
q : core point
(b)
p
q
p directly density-reachable from q
q not directly density-reachable from p

Figure 3.6: Part *a* shows the concept of border points and core points in a cluster. Part *b* shows the *ε-neighbourhood* of both p and q. In a cluster, the points are such that the *ε-neighbourhood* of core points contain the number of points less than or equal to the value of *minPts*.

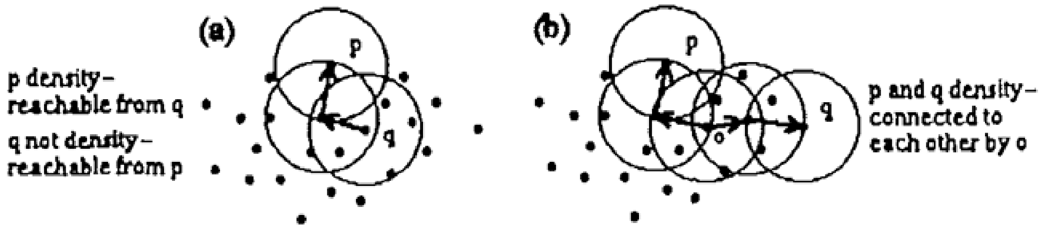

(a)
p
q
p density-reachable from q
q not density-reachable from p
(b)
p
o
q
p and q density-connected to each other by o

Figure 3.7: Part *a* illustrates how p is density-reachable from q while the reverse is not true. Part *b* shows how point p is density connected to point q.

approach, images were treated as special spatial dataset. They aimed to segment the images not only using colour but also using the spatial separation. They modified the algorithm to cater for spatial separation of the points. Instead of using ε, they introduced two properties, ϵ_c and ϵ_s. ϵ_s is the spatial neighbourhood around a given pixel and ϵ_c is the colour neighbourhood for a given pixel. A core point was defined as pixel whose ϵ_s contains *minPts* of similar color. Pixel p was said to be directly reachable from pixel q if p is within ϵ_s of q and q is a core pixel.

The value of ϵ_s was selected based on the size of the image and *minPts* was set to be half the value of ϵ_s. Determining ϵ_c required color conversion from RGB to Munsell (HVC) colour space. ϵ_c was then formatted as an ellipsoid within that colour space with radiuses H_{Radius}, V_{Radius} and C_{Radius}. Within that colour space, a core pixel p_o, with colour (H_o, V_o, C_o) is colour similar to pixel p with colour (H, V, C) if it satisfies equation 1 below:

$$\frac{(H-H_o)^2}{H_{Radius}^2} - \frac{(V-V_o)^2}{V_{Radius}^2} - \frac{(C-C_o)^2}{C_{Radius}^2} \leq 1 \tag{3.2}$$

The radius values are determined by using histogram connectivity analysis in H, V and C bands respectively. Their approach however still relies on a human to define colour similarity and the spatial neighbourhood. This means that there is a chance that when a different image is used those parameters would not work as well. Even more problematic is applying this to videos because of the ever changing scene which makes it difficult to know beforehand what conditions will be like in the video.

3.4.3 Hierarchical DBSCAN

Hierarchical DBSCAN (HDBSCAN) was developed by Campello et al. [35] to provide less supervision in cluster detection. While DBSCAN relies on the inputs, ε and *minPts*, HDBSCAN only takes *minPts* and creates all possible clusters for different values of ε and uses the concept of cluster stability to choose the final clusters. This leads to clusters that have different values of ε that relies on the points distribution within it. HDBSCAN's proper formulation adds the concepts of *core* and *reachability distances* from the OPTICS algorithm [87]. These concepts are defined as follows:

Definition 7: *Core Distance* Core distance is the minimum ε around point p that satisfies the condition $|N_\varepsilon(p)| \geq minPts$.

Definition 8: *Mutual Reachability Distance* The minimum ε for which points p and q are ε-reachable.

Definition 9,: *Mutual Reachability Graph* The graph created with vertices as all the points in D and the edges as the mutual reachability distances between respective points.

In essence, HDBSCAN produces a tree of possible clusters obtainable for all $\varepsilon \in [0, \infty)$. Obviously that clustering tree would be infinite, so the tree only consists of ε's where clusters undergo significant changes, i.e. when a point changes from core to noise, when a cluster splits into smaller clusters, and when a cluster completely disappears.

A minimum spanning tree (MST) is created from the mutual reachability graph which is then extended with self-edges. The HDBSCAN hierarchy is then extracted from the extended MST as a dendogram. Making a horizontal cut through the dendogram to get final clustering would correspond to setting a single density threshold which may not detect clusters with largely varying local densities. This is not a desirable outcome so the concept of cluster stability defined below is used instead.

$$S(C_i) = \sum_{x_j \in C_i} \left(\frac{1}{\varepsilon_{min}(x_j, C_i)} - \frac{1}{\varepsilon_{max}(C_i)} \right) \tag{3.3}$$

If we have a set $\{C_2, ..., C_\kappa\}$ as a collection of all clusters in the hierarchy tree excluding the root C_1, and $S(C_i)$ as the stability of each cluster, extracting prominent non-overlapping clusters can be treated as an optimisation problem expressed as:

$$\begin{aligned} \max_{\delta_2 \dots \delta_\kappa} \quad & J = \sum_{i=2}^{\kappa} \delta_i S(C_i) \\ \text{subject to} \quad & \begin{cases} \delta_i \in \{0,1\}, i = 2, ..., \kappa \\ \text{exactly one } \delta_{(.)} = 1 \text{ in each path from leaf} \\ \text{ to node} \end{cases} \end{aligned} \tag{3.4}$$

where δ_i indicates whether Cluster C_i is included into the final cluster solution or not. The solution can be found by propagating the total stability $S(C_I)$ from the leaf cluster to the root using:

$$\hat{S}(C_i) = \begin{cases} S(C_i) & \text{if } C_i \text{ is a leaf node} \\ max\{S(C_i), \hat{S}(C_{il}) + \hat{S}(C_{ir})\} & \\ \text{if } C_i \text{ in an internal node} & \end{cases} \tag{3.5}$$

where $\hat{S}(C_{il}) + \hat{S}(C_{ir})$ is the summation of the total stabilities of the left and right nodes of cluster C_i in the case of a binary tree. If the tree is not binary, the sum would be for all the nodes of the cluster. In this thesis, we consider the set of prominent clusters $\vec{\mathcal{C}} = \{C_1, ..., C_j\}$ and the corresponding labels $\vec{\mathcal{L}} = \{l_1, ..., l_j\}$ such that for $1 \leq \hat{j} \leq j$, the label for $C_{\hat{j}}$ is $l_{\hat{j}}$ and for $\hat{j} = 1$, C_1 is the noise cluster and $l_1 = 0$.

3.4.4 HDBSCAN Implementation

The first part of HDBSCAN is the distance calculations, where, given a dataset, $\mathcal{D}$, the algorithm requires the distances between the elements of $\mathcal{D}$. This process is both memory and computationally intensive. If $\vec{\mathcal{D}}$ has length N, the resulting matrix of distances will have dimensions N x N. Both the memory and computational time are $\mathcal{O}(n^2)$ operations. In order to reduce the amount of memory needed, we need to take a closer look at the resulting distance array. Given a dataset $\vec{\mathcal{D}} = \{1, 4, 9, 7\}$, the distance matrix will be:

$$D = \begin{bmatrix} 0 & 9 & 64 & 36 \\ 9 & 0 & 25 & 9 \\ 64 & 25 & 0 & 4 \\ 36 & 9 & 4 & 0 \end{bmatrix}$$

The distance matrix is symmetric with 0 on the principal diagonal such that such that $D_{i,j} = D_{j,i}$. This property of the distance matrix allows for significant reduction in the memory requirements by only storing the top half of the matrix. If the length of the dataset is denoted by N_x, then the new distance matrix as a vector will have length $N_d = N_x(N_x - 1)/2$ which is the triangular number of N_x. In our implementation, the distance calculations are sped up by using OpenMP [88] with the parallelism provided by multi-core processors available in most computers. The core distances are extracted using a separate function that also uses OpenMP for parallelism. We found out that this approach offers higher computational speed than combining distance calculation and core distance extraction as that would require full N_d^2 iteration over the dataset.

Since only the part of the original distance matrix above the diagonal is being saved, there needs to be a way to encode and decode the distances into the proper locations in the new distance vector. Using the example above, the new distance vector will be $\vec{D_n} = \{9, 64, 36, 25, 9, 4\}$. Considering only those values where $i < j$, $distance(\mathcal{D}_i, \mathcal{D}_j)$ will be encoded into location $idx = i * rows + j - triangular(i + 1)$, where *rows* is the length of the dataset $\vec{\mathcal{D}}$, and *triangular(i+1)* is the triangular number of *i+1*. The decoding of the distance location is based on the Algorithm 1.

Algorithm 1: Decoding the index of the distance in the new distance vector for a given point (i, j)

```
Data: Point (row, col)
Result: location of the distance
if $row < col$ then
    $idx \leftarrow (rows * row + col) - triangular(row + 1)$
else
    if $row == col$ then
        $idx \leftarrow 0$
    else
        $idx \leftarrow (rows * col + row) - triangular(col + 1)$
    end
end
```

Our implementation of HDBSCAN expects a 2-dimensional matrix where each row is a data point within the dataset. In Chapter 4, we use the descriptors as the dataset and in Chapter 5, we use the RGB values at the feature locations. The optimised distance calculations are done using Algorithm 2 and the corresponding code sample A.4.

In our implementation, we realised that in some cases we wanted to vary the value of the parameter *minPts* without changing the dataset. We therefore, implemented HDBSCAN in such a way that the distances are only calculated once and then re-used with various *minPts* values. Since the distance calculations occupy the bulk of memory and computational requirements, this approach speeds up our algorithm in subsequent values of *minPts*.

3.4.5 Cluster Analysis and Validation

The post clustering stage in cluster detection is concerned with evaluation and interpretation of the clusters and clustering results validity [89]. This stage can be seen as

Algorithm 2: Distance calculation

```
Data: $\vec{\mathcal{D}}$, numNeighbours
Result: $\vec{D}$ :distance vector
for $i \leftarrow 0...length(\vec{\mathcal{D}})$ do
    $x_i \leftarrow row(\vec{\mathcal{D}}, i)$
    for $j \leftarrow (i+1)...length(\vec{\mathcal{D}})$ do
        $x_j \leftarrow row(\vec{\mathcal{D}}, j)$
        $d_{i,j} \leftarrow L2_norm(x_i, x_j)$
        $offset \leftarrow i * length(\vec{\mathcal{D}})$
        $idx \leftarrow offset - triangular(i+1)$
        $\vec{D}(idx) \leftarrow d_{i,j}$
    end
end
```

answering three issues; cluster quality, cluster interpretability and cluster tendency. Good quality clusters will be very different from each other while the data points within will be very similar. The interpretability of clusters is concerned with the common features shared by the points within a cluster, and cluster tendency is about whether there are actually clusters within the dataset. In this section, we review common approaches to the above issues and propose a new way of calculating cluster quality and clustering results validity.

3.4.5.1 Cluster Quality

Variations in the points within a cluster are a good indication of the quality of a cluster. In [89], this is called 'within-cluster' variation and is defined by Equation 3.6. In this equation, given a set of clusters $\vec{\mathcal{C}} = \{C_1, ..., C_k\}$, $C_k \in \vec{\mathcal{C}}$ is the cluster, x is a member of the cluster and r_k is the centroid of the cluster.

$$wc(C_k) = \frac{1}{|C_k|} \sum_{x \in C_k} d(x, r_k)^2 \tag{3.6}$$

where $d(x, r_k)$ is the distance between any point in the cluster and the cluster centroid. The overall quality of the clustering results also need to be determined. Equation 3.7

has been used to determine the overall clustering results quality.

$$\frac{BC}{WC} \tag{3.7}$$

where WC is the sum of within-cluster variations and BC is the sum of squared distances between the cluster centroids:

$BC = \sum_{i \leq j \leq k \leq K} d(r_j, r_k)$ and $WC = \sum_{k=1}^{K} wc(C_K)$

In this thesis, we take a different approach to cluster analysis and validation. With equation 3.6, the smaller the value of $wc(C_k)$, the higher the similarity of the points within the cluster. This is useful when comparing the clusters in $\vec{\mathcal{C}}$. However, the value of $wc(C_k)$, like WC, is unbounded. As such, in this thesis, they do not provide proper measure of cluster quality. Our approach to cluster quality is about measuring the variations within the cluster as a function of the overall cluster differences such that we get a percentage value signifying how similar the points within the each cluster are.

We use both the intra-cluster distances and the cluster core distances to calculate the quality of the clusters. Using intra-cluster distances allows for measuring the quality of cluster based on the cluster densities while using core distances allows for measuring the cluster quality based on the localised distribution of the features. For each cluster C_k, we calculate the minimum and maximum core distances (D_{min} and D_{max}). We then used them to calculate the $max : min$ ratios (*coreRatio* 3.8 and *disRatio* 3.9) for each of the clusters.

$$R_c = \frac{C_{max}}{C_{min}} \tag{3.8}$$

$$R_d = \frac{D_{max}}{D_{min}} \tag{3.9}$$

The maximum core distance ratio ($R_{c,m}$) and maximum intra-cluster distance ratio ($R_{d,m}$) are then used to calculate the percentage confidence in each cluster (C_k)

according to the core distances and the intra-cluster distances using equations 3.10 and 3.11 respectively. In these equations, $R_{c,k}$ and $R_{d,k}$ represent the core distance ratio and intra-distance ratio for cluster C_k respectively. The results obtained showed that the cluster confidence values above 50% can be trusted to describe clusters that have high feature similarity (Details in Chapter 4). It was also observed that any cluster that has a ratio that is greater than the ratio of the noise cluster will be visually noisy.

$$F_c = \frac{R_{c,m} - R_{c,k}}{R_{c,m}} * 100 \tag{3.10}$$

$$F_d = \frac{R_{d,m} - R_{d,k}}{R_{d,m}} * 100 \tag{3.11}$$

3.4.5.2 Clustering Validity

Validation of overall cluster detection results is a challenging task as most of the time it requires an in-depth knowledge of the dataset space. One way of determining if the dataset has clusters is to use Hopkin's equation (See Equation 3.12) [89]. A random sample of the data $\vec{S}$, is extracted from the main dataset, and a set of randomly generated data, $\vec{P}$ with the same size as $\vec{S}$ is created. For both $\vec{S}$ and $\vec{P}$, the distance between each point and its nearest neighbour is calculated and all the distances are summed. If the ratio yields a value close to 0.5, it shows that the sample data $\vec{S}$, and the random data $\vec{P}$ have roughly the same nearest neighbour distances which is a strong signal that the data may not have clusters.

$$H(\vec{P}, \vec{S}) = \frac{\sum\limits_{p,t_p \in S} dist(p, t_p)}{\sum\limits_{m,t_m \in P} dist(m, t_m) + \sum\limits_{p,t_p \in S} dist(p, t_p)} \tag{3.12}$$

The use of Hopkin's equation has two challenges. Firstly, it assumes that the sample data $\vec{S}$, will be a proper representation of the dataset. There is no guideline as to the appropriate size for $\vec{S}$. Furthermore, the distance calculations between $\vec{P}$ and S would increase execution time. Secondly, the suggested ration of 0.5 does not have a upper or lower limit. It also relies on whether the sample data is a proper representation of the whole dataset. These two challenges highlight the inadequacy of Hopkin's equation in this thesis. The descriptor dataset being used in this thesis is generated for each frame on the fly. As such, there is no way of assessing the selection criteria for each frame and each video.

In this thesis, we use Algorithm 3 to calculate bounded discreet values between -1 and 4 inclusively to represent how valid the clustering results are. The algorithm uses skewness (Equation 3.14) and kurtosis (Equation 3.13) statistical values on the core and intra-cluster distance ratios explained in the previous sub-section. Incidentally, equation 3.13 equation requires that at least 3 clusters be detected and equation 3.14 requires at least 2 clusters.

In this thesis, we use skewness to measure the symmetry (or lack thereof) of the distribution of the distance rations. The work in this thesis assumes that the noise cluster will have more features that the other clusters. In this case, the noise cluster will have the largest distance ratio, thus creating positive skewness. We use kurtosis to measure the peakedness of the distribution of the distance ratios. Working off of the same assumption as for skewness, kurtosis measures how large the distance ration of the noise clusters is compared to the other clusters which indicates how much tail the distribution has. Positive kurtosis therefore indicates that the distribution has a tail and negative kurtosis indicates there is no difference between the ratios of the noise cluster and other clusters.

In the context of our distance ratios, when skewness is positive and the kurtosis is also positive, it indicates that the noise cluster has the largest *max:min* ration and also creates the tail in the distribution. This outcome proved to be a good indication of clustering validity. Using Algorithm 3, we therefore reward positive skewness and kurtosis and punish negative values. Core distances are less susceptible to cluster sizes but more susceptible to noise while intra-cluster distance are more susceptible to cluster sizes but less susceptible to noise. As such, our validity algorithm uses the ratios from both distances in order to balance out the weaknesses.

$$K = \frac{N(N+1)}{(N-1)(N-2)(N-3)} \sum_{j=1}^{N} \left(\frac{x_j - \overline{x}}{s} \right)^4 - \frac{3(N-1)^2}{(N-2)(N-3)} \tag{3.13}$$

$$\varsigma = \frac{N}{(N-1)(N-2)} \sum_{j=1}^{N} \left(\frac{x_j - \overline{x}}{s} \right)^3 \tag{3.14}$$

Algorithm 3: Calculating validity for overall clustering results for a particular *minPts* value.

```
Data: $\varsigma_c$ - Skewness value from core distances
Data: $K_c$ - Kurtosis value from core distances
Data: $\varsigma_d$ - Skewness value from intra-cluster distances
Data: $K_d$ - Kurtosis value from intra-cluster distances
Result: $validity = 0$
if $\varsigma_d > 0$ & $K_d > 0$ then
    $validity \leftarrow validity + 2$
else if $\varsigma_d < 0$ & $K_d > 0$ then
    $validity \leftarrow validity + 1$
else
    $validity \leftarrow validity - 1$
end

if $\varsigma_c > 0$ & $K_c > 0$ then
    $validity \leftarrow validity + 2$
else if $\varsigma_c < 0$ & $K_c > 0$ then
    $validity \leftarrow validity + 1$
else
    $validity \leftarrow validity - 1$
end
```

3.5 Summary

In this chapter, we have discussed the single object tracker algorithms that have been implemented in the OpenCV library. We justified the selection of BOOSTING as the preferred SOT for this thesis by comparing the execution time and drift resistance.

While GOTURN showed the ability to adjust to scale of the target but was very unstable which suggest that it requires more training. TLD's detector operating on the whole image resulted in target loss when dealing with multiple target instances.

In Section 3.2, we review the current literature on multi object tracking. While we did not make any experimental comparisons, we did discuss the key deficiencies in the literature. The main challenge common to all is the initialisation process. Either the user has to provide multiple ROIs or they have to train the algorithm off-line, both of which are impractical for the requirements of this thesis. For our requirements, we develop our own MOT built around HDBSCAN in Chapter 5.

We also looked at SURF local feature detector and descriptor. We reviewed how the algorithm works and how the descriptor matching works. We also explained the inadequacies of the current feature matching algorithms and suggested the use of density-based clustering as a way of matching multiple feature instances. The HDBSCAN algorithm which we use in this thesis for was reviewed as well as explaining the details that allow our implementation to be fast by employing parallelism and memory requirements. In addition, we explained how to analyse the clusters detected to provide discreet values on validity while also providing a measure of similarity within the clusters.

In the next chapter, we discuss how we use HDBSCAN to detect matching SURF features in a video frame. We use the cluster quality and validity approaches developed in this chapter to justify why our multiple feature instance matching works better than the current approaches. We explore the different types of results we can obtain and show the steps we can take to get the best possible results.

Chapter 4

Multiple Local Feature Instance Detection

In Chapter 3, we discussed various background literature relevant to the work in this thesis. We discussed and compared the various SOT algorithms implemented in OpenCV. We have explored the literature on multiple object tracking and the current inadequacies. We also reviewed the current local feature matching approaches and explained how they are inadequate for our purpose. Finally, we discussed density-based clustering detection algorithm HDBSCAN as a potential candidate for solving the current local feature matching algorithms shortfalls.

The approach in this thesis is based on the hypothesis that the small variations in matching features would form clusters in the descriptors space which can be detected using HDBSCAN. When working with multiple object instances, we want to do a one-to-many matching which automatically rules out brute-force matcher. FLANN is not suitable either because of the parameter k, which cannot be determined beforehand. We also cannot use Lowe's Ratio Test [34] to test the quality of the matches as it was designed to evaluate one-to-one feature matches.

In this chapter, we explore the matching of multiple features as a density-based clustering problem. By hypotheses 1 and 2, outlined in Chapter 3, we show that not only do the local features form clusters within the descriptor dataset, Lowe's ratio test fails to measure the quality of matches. We validate the cluster quality by calculating core-distance and intra-cluster distance confidences as percentages of the maximum variations within the clusters. We show how the validity calculations in Section 3.4.5 is used to inform the proper choice of *minPts*.

4.1 Multiple Features as Density-Based Clusters

On a frame-by-frame basis, our approach detects clusters in each frame's SURF features. The quantity of these features varies from frame to frame, and from video to video. The number of SURF local features in our dataset are shown in Tables 4.1 (LODV), 4.2 (MODV) and 4.3 (HODV). An interesting observation is that some of the LODVs have higher number of features than some MODVs and HODV. This is because the number of points in the frames is not only dependent on the number of objects of interest in the video. Instead, they rely on the global variations in the video frames. In videos with complex backgrounds, a lot of the local features do come from the background as opposed to the objects of interest. The size of the objects also has an effect; the larger the object in the frame, the more the number of features on the object.

The fact that the larger objects have more features is illustrated in Figure 4.1 where the first frame of the LODV *voc-18-bd-11* had the highest number of features, 329, due to complexity of the background. In comparison, the MODV video *voc-18-bd-16* had 157 features, and the HODV video *voc-18-bd-14* had 308 features. In addition, the two latter video had scale and vanishing point problems which meant that the objects closer to the camera had more features that those further away. The size of the birds in *voc-18-bd-11* also contributes to the high number of features in the frame.

In our approach, we ran HDBSCAN on the frame descriptors, $\vec{D}$, with $minPts = 3$ and for each frame. We analysed each of the clusters and the overall clustering validity using the equations 3.8 - 3.14 and Algorithm 3 discussed in Subsection 3.4.5. We used $minPts = 3$ because using two is similar to attempting a one-to-one matching of features. On a frame by frame basis, we discovered that the best clustering results are achieved when the validity is 4 since lower values of validity often have noise features.

Table 4.1: The number of SURF features detected for LODVs.

Video	**Min**	**Max**	μ	σ
voc-18-bd-4	197	321	270	33.71
voc-18-bd-5	136	239	192	39.458
voc-18-bd-6	242	348	292	27.177
voc-18-bd-7	156	258	218	24.031
voc-18-bd-8	343	525	426	56.345
voc-18-bd-9	198	425	314	54.602
voc-18-bd-11	328	501	400	35.893
voc-18-bd-20	75	139	106	18.957

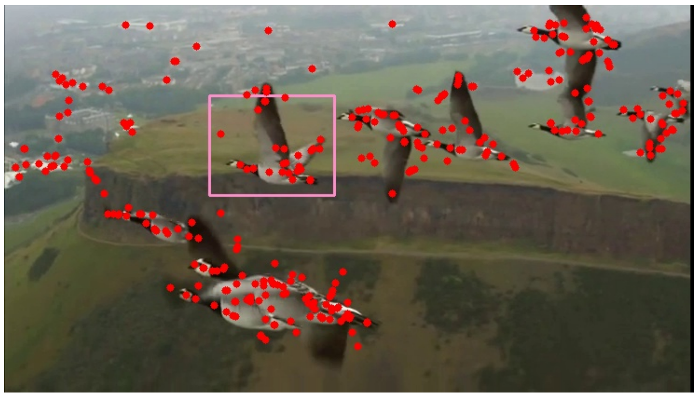
Table 4.2: The number of SURF features detected for MODVs.

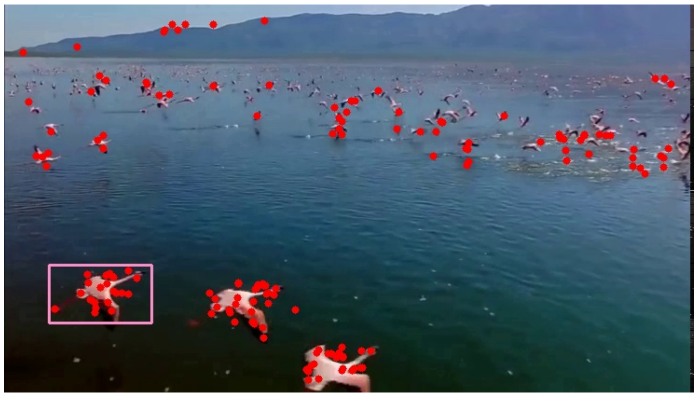
Video	**Min**	**Max**	μ	σ
voc-18-bd-2	47	265	118	58.939
voc-18-bd-10	207	457	358	66.743
voc-18-bd-16	96	200	144	26.016
voc-18-bd-19	851	1046	950	35.348

We also found that the ratios between the minimum and maximum distances within the clusters can fail Lowe's Ratio test even though the features selected are visually very similar.

Table 4.3: The number of SURF features detected for HODVs.

Video	**Min**	**Max**	μ	σ
voc-18-bd-1	2949	3451	3272	99.198
voc-18-bd-3	394	607	509	43.243
voc-18-bd-12	1305	2209	1918	248.344
voc-18-bd-13	130	299	199	39.743
voc-18-bd-14	176	360	290	44.828
voc-18-bd-15	317	324	271	22.972
voc-18-bd-17	228	304	256	14.41
voc-18-bd-18	1786	2363	2142	158.16
voc-18-bl-1	3722	6662	6410	466.954
voc-18-bl-2	5961	6041	6000	23.615
voc-18-bl-3	822	3260	2035	844.449
voc-18-bl-4	1613	1699	1659	23.002


(a) Frame 1 of *voc-18-bd-11* which had 329 features.


(b) Frame 1 of *voc-18-bd-16* which had 157 features.

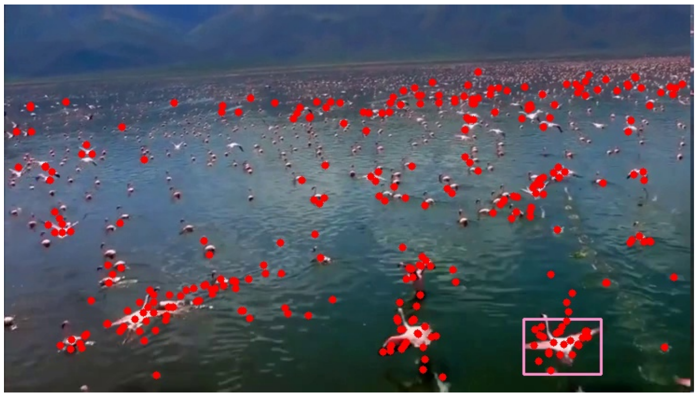

(c) Frame 1 of *voc-18-bd-14* which had 308 features.

Figure 4.1: Differences in the number of features in LODVs, MODVs and HODVs.

4.1.1 Lowe's Ratio in Clusters

In Table 4.4, we show the two of the features from cluster 100 in frame 1 of *voc-18-bd-1* where $minPts = 3$ was used. The features of reference are shown in bold in row 1 and the table data is arranged in ascending order by the distance from the reference features. One interesting observation in the table is that there is a noise feature that is closer in distance to feature 2495 that one of the features in cluster 100. This is because the shape of a cluster can lead to noise features, or indeed features from other clusters being closer to one feature that other features in the same cluster.

In terms of Lowe's Ratio test, we use the distance of the closest feature to the reference feature. In the first set of results, we use feature 2503, and in the second set of results we use 2670. We then calculate the ratio using the first noise feature, then using the first non noise feature that belongs to a different cluster and finally using the furthest feature in the same cluster.

When the reference point is 2496, the first noise feature is 2574 in row 4 and the distance ratio is 0.93 which means the test would fail. With the fist non-noise feature belonging to a different cluster, we use feature 2980 in row 6 and the ratio is 0.75. The furthest feature in cluster 100 from feature 2495 is 2670 in row 5 and the ratio is 0.83. The results show that the match of feature 2495 to 2503 is not valid when using the first noise feature closest to 2495. The match was valid when using feature 2980. The ratio from the furthest feature in the same cluster cannot be used because the ratio will be affected by the density of the cluster and the size of the area occupied by the cluster within the feature space.

The distance ratios in the second feature shows that the features within clusters 100 all fail Lowe's ratio test for the match between features 2666 and 2670 while all the other features pass. Considering that all the features within a cluster are a match to each other, the ratio of the match from feature 2666 to feature 2503 with the first noise feature 2574 is 0.79 which shows that the match is valid. As such, we conclude that the use of Lowe's Ratio is not adequate for evaluating many-to-many matches as the results may fail or pass depending on which feature is used as reference in the cluster.

Table 4.4: This table shows the closest features to two features from cluster 100 in frame 1 of voc-18-bd-1. The cluster had 4 features.

#	**Cluster**	**Feature**	**Distance**	**Lowe's Ratio**		**Cluster**	**Feature**	**Distance**	**Lowe's Ratio**
1	*100*	*2495*	*0*	*0*		*100*	*2666*	*0*	*0*
2	100	2503	0.198	1		100	2670	0.185	1
3	100	2666	0.212	0.94		100	2495	0.212	0.88
4	0	2574	0.213	0.93		100	2503	0.225	0.82
5	100	2670	0.238	0.83		0	2574	0.284	0.72
6	101	2980	0.265	0.75		101	3106	0.295	0.71
7	0	2903	0.267	0.74		0	3258	0.305	0.62
8	101	2838	0.273	0.73		0	3242	0.319	0.61
9	0	3258	0.278	0.71		101	2838	0.319	0.6
10	101	2901	0.281	0.7		0	2835	0.321	0.59
11	101	3106	0.283	0.7		101	2821	0.325	0.58
12	101	2821	0.283	0.7		101	2901	0.333	0.57
13	0	3286	0.297	0.67		0	3247	0.334	0.57
14	101	2657	0.298	0.66		101	2980	0.336	0.57
15	0	2933	0.303	0.65		101	2657	0.342	0.56
16	0	3242	0.313	0.63		265	2626	0.347	0.56
17	0	3322	0.313	0.63		0	3355	0.348	0.56
18	0	3355	0.317	0.62		285	2852	0.352	0.55

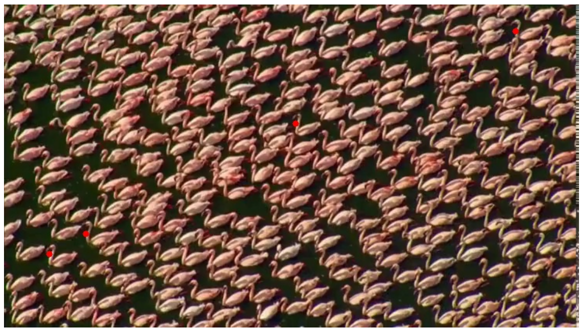
The results of Table 4.4 also show the reason cluster 101 features are close to the features of cluster 100. This similarity can be seen in Figure 4.2 where the first two sub-figures, represent clusters 100 (Figure 4.2a) and 101 (Figure 4.2b) respectively. Both clusters identify the same areas on the necks of the birds which highlights the potential of cluster over-segmentation. However, the other two closest clusters show significantly different features as can be seen in Figures 4.2c and 4.2d.

4.1.2 Cluster Feature Similarity

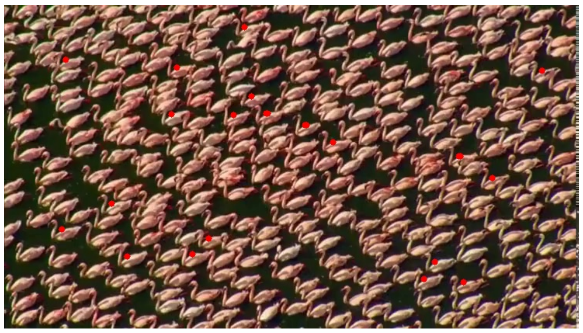
In this section, we show the credibility of the many-to-many matching approach used in this thesis. We show the results of frame 4 of *voc-18-bd-20* with $minPts = 4$. The table in this section shows, for each cluster, the minimum and maximum distances between the cluster features. We then calculate the *min:max* ratios. However, these ratios in this section are used to highlight the variance in the density of the clusters.

In Table 4.5, cluster 0, which represent the noise features in the descriptor set, has the lowest ratio in both core and intra-cluster distances. The similarity confidences have also been calculated as 0% which makes logical sense since the noise cluster represents no real match between the features within it. In most cases, the cluster also has more features than in other clusters. It should be noted that this might not always be the case. In a situation where the continuity of the non-cluster within the dataset space has more points than the noise, the span of the cluster might surpass that of the noise cluster.

As shown in Table 4.5, cluster 0, which represent the noise features in the descriptor set, has the lowest ratio in both core and intra-cluster distances. The similarity confidences have also been calculated as 0% which makes logical sense since the noise cluster represents no real match between the features within it. In most cases, the cluster also has more features than in other clusters. It should be noted that this might not always be the case. In a situation where the continuity of the non-cluster within the dataset space has more points than the noise, the span of the cluster might surpass that of the noise cluster.


(a) Cluster 100.

(b) Cluster 101.


(c) Cluster 265.

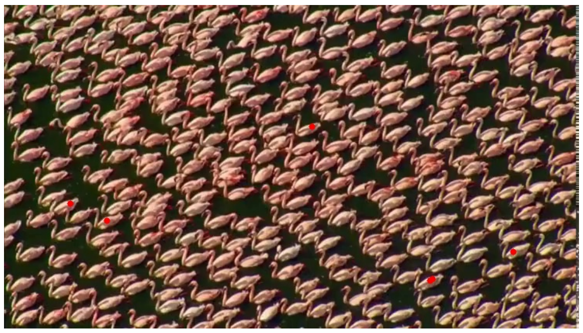

(d) Cluster 285.

Figure 4.2: Visualisation of some of the clusters from Table 4.4.

SURF feature descriptors have 64 dimension, which makes visualising the similarities of the clusters in a graph like manner impossible. In this thesis, we visualise the similarities by drawing the features in the frames. In Figure 4.3, we show some of the clusters from Table 4.5. As expected, cluster 0 points do not show any discernible similarity.

The other clusters show varying degrees of visual similarities with cluster 6 in Figure 4.3b showing the most easily identifiable similarity within the points. An interesting observation of cluster 6 is that while the points are not on the birds, they do identify the area around the neck and the outstretched wing. This suggests that the features in the surrounding of the objects of interest can still be used to identify the objects. This is contrary to common approaches where object identification using local features relies on the features *on* the objects such as in [22].

Another example of how well HDBSCAN performs in recognising similar descriptors is shown in Figure 4.4. The figure is from the first frame of *voc-18-bd-1* where $minPts = 3$ was used and validity was 4. In that frame, HDBSCAN detected 146 clusters from 3375 descriptors with the number of points for non-noise clusters ranging from 3 to 61. The noise cluster had the largest number of points at 2463 and both confidences at 0%. Cluster 3 (Figure 4.4b) had intra-cluster confidence of 97%. It also had $min : max$ ratio of 0.859 for core distances and core distance confidence of 54%. What is most interesting about the cluster is that the points are perfectly aligned with the heads of the birds.

Clusters 157 (Figure 4.4c) and 294 (Figure 4.4d) show how robust the clusters can be in handling a lot of features. In cluster 157, there are 50 features with intra-cluster distance ratio of 0.128 and core distance ratio of 0.536. The similarity can be best summarised by the 84% inter cluster distance confidence. Visually, cluster 157 has most of the points in the 'V' shape made by the neck and the back of the birds, while some of the points are on the 'V' shape made by head of one bird in occlusion with the belly of another bird. Cluster 294 is the largest non-noise cluster with 61 features. These features are located on the backs of the birds with intra-cluster ratio of 0.231 and core distance ratio of 0.596. The visual similarity of the points is confirmed by the intra-cluster confidence of 91% even though the core confidence is at 33%.

Table 4.5: Cluster data from *voc-18-bd-20* frame 4 with *minPts* = 4.

		Core Distances				**Intra-Cluster Distances**			
Cluster	**Num. of Points**	**Min**	**Max**	**Min:Max Ratio**	**Confidence (%)**	**Min**	**Max**	**Min:Max Ratio**	**Confidence (%)**
0	53	0.298	0.744	0.401	0	0.063	1.322	0.047	0
2	4	0.206	0.436	0.473	15.134	0.094	0.444	0.211	77.244
5	5	0.160	0.301	0.532	24.665	0.117	0.332	0.353	86.405
6	5	0.215	0.364	0.591	32.125	0.213	0.477	0.447	89.273
9	3	0.337	0.345	0.978	59.006	0.184	0.345	0.535	91.035
13	7	0.285	0.324	0.878	54.297	0.064	0.644	0.1	52.04
14	5	0.267	0.342	0.779	48.529	0.187	0.568	0.329	85.401
17	4	0.334	0.378	0.884	54.61	0.329	0.44	0.746	93.572
19	3	0.357	0.362	0.985	59.263	0.357	0.546	0.654	92.662
20	7	0.266	0.340	0.781	48.646	0.048	0.479	0.101	52.254
21	18	0.176	0.337	0.521	23.064	0.115	0.81	0.142	66.297
24	4	0.131	0.161	0.814	50.742	0.08	0.171	0.465	89.691
25	3	0.166	0.186	0.891	54.971	0.07	0.186	0.378	87.313

During our experiments, we observed that for the same value of *minPts*, the validity may vary from one frame to the next. These variations are inherited from the slight variations in the frames as the video progresses. These variation in turn affect the SURF features being detected such that while in one frame proper clusters were detected, in the next, some features may disappear while new ones are discovered. This, in turn, affects the distribution of the features in the descriptor space which can lead to failure in detecting the clusters.

We also observed that when moving between two frames where one has the $validity =$ 4 and one with lower validity, especially when the lower validity is below two, two things occur. The first is that there is either big in the number of clusters or a change in the number of features in the noise cluster as shown in Table 4.6. In *voc-18-bd-6*, frame 1 had $validity = 0$ and frame 2 had $validiy = 4$. In frame 1, we detected four non-noise clusters (See Table 4.7) but in frame 2, 18 non-noise clusters were detected. The noise cluster in frame 1 had 22 out of 246 features (See Figure 4.5a) and the noise cluster in frame 2 had 149 out of 242 features (See Figure 4.5b). It is worth noting that this behaviour does not always happen if the validity is between two and four.

The first thing to note about the change in the number of noise features is that the

Table 4.6: The first 8 frames of *voc-18-bd-6* showing the variations in number of clusters and the validities from frame to frame.

Frame	**Feature Size**	**Num of Clusters**	**Validity**	**Size of Cluster 0**
1	246	4	0	22
2	242	19	4	149
3	251	23	4	150
4	254	16	2	154
5	254	16	2	112
6	254	16	4	161
7	258	3	0	26
8	264	20	4	16

Table 4.7: The clusters detected in frame 1 of *voc-18-bd-6*. The clustering results had validity = 0.

		Core Distances				**Intra-Cluster Distances**			
Cluster	**Number of Points**	**Min**	**Max**	**Min:Max Ratio**	**Confidence (%)**	**Min**	**Max**	**Min:Max Ratio**	**Confidence (%)**
0	22	0.402	0.592	0.68	49.847	0.364	1.362	0.268	88.126
3	210	0.155	0.456	0.341	0	0.043	1.359	0.032	0
7	5	0.313	0.395	0.791	56.894	0.107	0.569	0.188	83.076
10	3	0.279	0.32	0.87	60.792	0.161	0.32	0.502	93.676
11	6	0.243	0.331	0.734	53.551	0.242	0.452	0.536	94.07

features in the noise clusters are still very noisy. As shown in Figure 4.5a, there is no visible similarity between the features even though the intra-cluster confidence in Table 4.7 is 88%. The second thing to notice is that with the reduction of both the number of clusters and the number of noise features, one of the non-noise clusters has the largest number of features. In the case of frame 1, it was cluster 3 with 210 features. It also results in the cluster having the smallest $min : max$ ratios as such also having 0% confidence. It is worth noting that the other clusters the results do have features that are visually similar. By the experimental observation of other video results we concluded that the only results that should be accepted are the ones where validity is 4.

In Tables 4.8, 4.9 and 4.10 we show the overall validity statistics for the videos in our dataset. In terms of overall video clustering validities, we noted that the best results had $\mu \geq 3$ which signifies that a lot of the frame had validities of 4 and $\sigma \leq 1$ which signifies that there is little variation between the validities from frame to frame. As the table shows, keeping $minPts$ at 3 resulted in a lot of videos that we outside the desired values of validity. Of the 24 videos, only 8 videos had an average validity greater than 3. Only three of the videos had acceptable mean and standard deviation. Empirical evidence shows that there is no correlation between the validity values and the video categories.

Table 4.8: The clustering validity statistics for LODVs for $minPts = 3$.

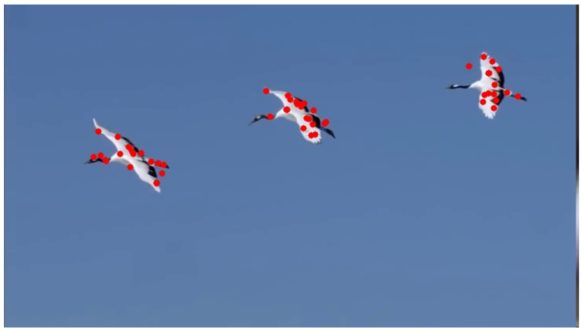
Video	**Min**	**Max**	μ	σ
voc-18-bd-4	2	4	3.69	0.727
voc-18-bd-5	0	4	2.98	1.395
voc-18-bd-6	-1	4	2.62	1.831
voc-18-bd-7	0	4	3.66	0.819
voc-18-bd-8	-1	4	1.98	1.908
voc-18-bd-9	-1	4	3.02	1.456
voc-18-bd-11	-1	4	2.99	1.663
voc-18-bd-20	-1	4	1.4	1.54

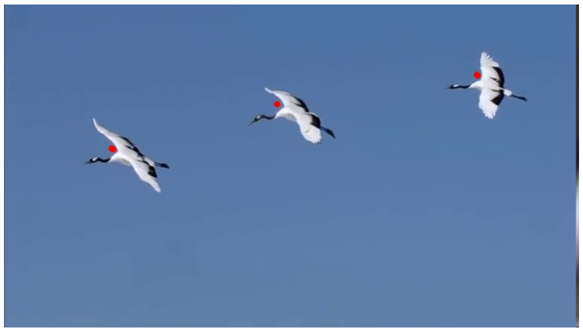
Table 4.9: The clustering validity statistics for MODVs for $minPts = 3$.

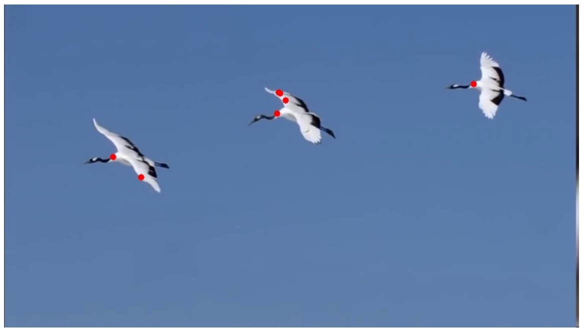
Video	**Min**	**Max**	μ	σ
voc-18-bd-2	-2	4	-0.352	1.266
voc-18-bd-10	0	4	3.22	1.107
voc-18-bd-16	0	4	2.57	1.317
voc-18-bd-19	-2	4	3.31	1.6

Table 4.10: The clustering validity statistics for HODVs for $minPts = 3$.

Video	**Min**	**Max**	μ	σ
voc-18-bd-1	-1	4	2.78	1.889
voc-18-bd-3	-1	4	1.89	1.94
voc-18-bd-12	0	4	3.42	1.394
voc-18-bd-13	-2	4	0.43	1.356
voc-18-bd-14	-2	4	2.89	1.681
voc-18-bd-15	-2	4	0.63	1.484
voc-18-bd-17	-2	4	0.81	1.6
voc-18-bd-18	0	4	2.13	2.066
voc-18-bl-1	-1	2	-0.58	0.644
voc-18-bl-2	0	4	1.83	2.003
voc-18-bl-3	0	4	3.65	1.137
voc-18-bl-4	0	4	3.84	0.799


(a) The noise cluster.


(b) Cluster 6 with 5 points.


(c) Cluster 13 with 7 points.

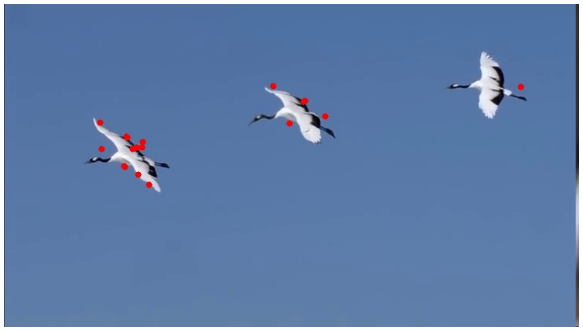
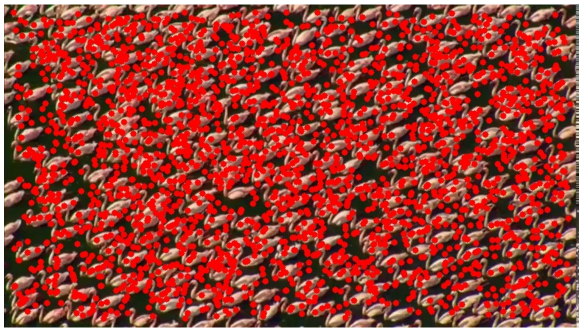
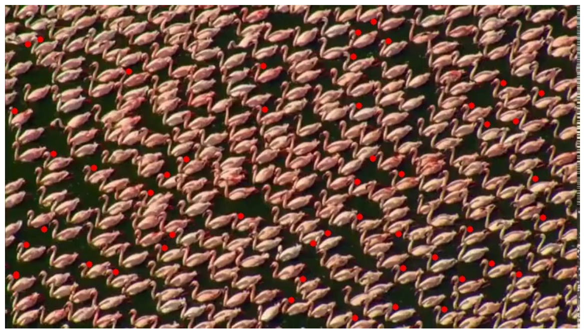

(d) Cluster 21 with 18 points.

Figure 4.3: Visualisation of some of the clusters from Table 4.5.

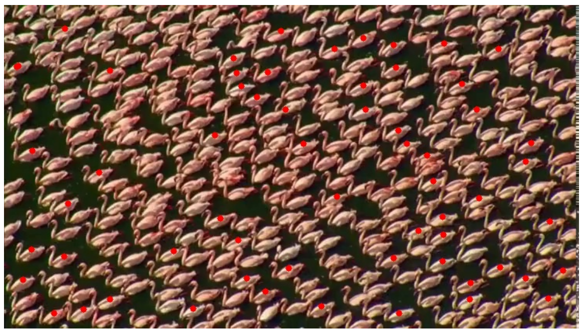


(a) The noise cluster.

(b) Cluster 3 with 3 features.

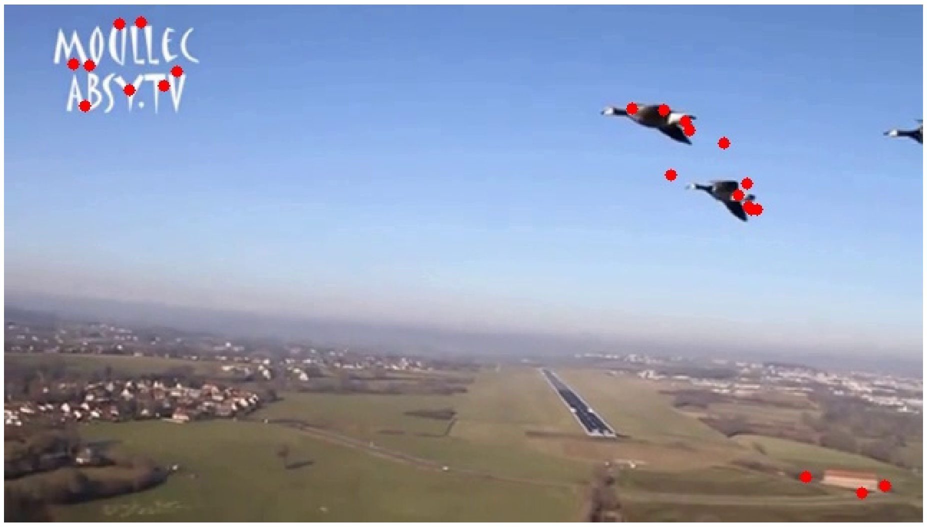


(c) Cluster 157 has 50 features.

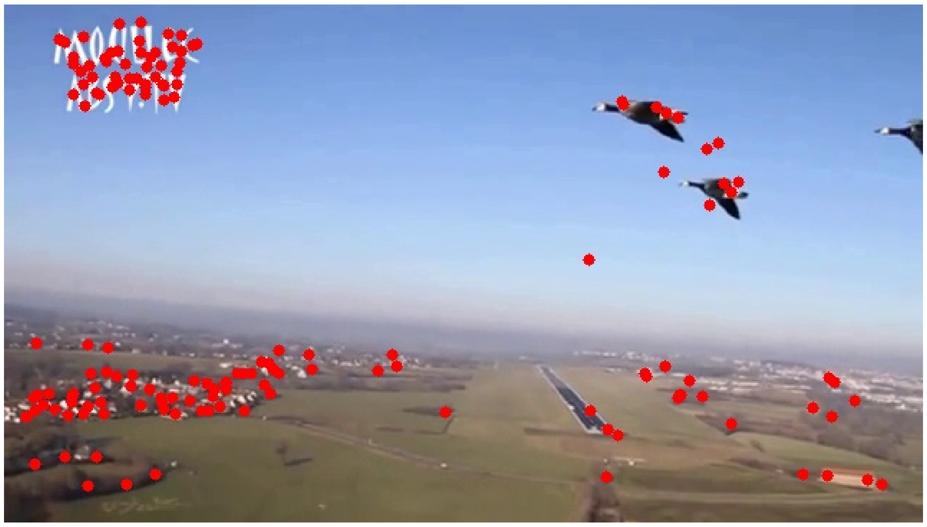


(d) Cluster 294 with 61 points.

Figure 4.4: Visualisation of some of the clusters from the first frame of *voc-18-bd-1* HODV.



(a) The noise cluster in frame 1 with *validity* = 0.



(b) The noise cluster in frame 2 with *validity* = 4.

Figure 4.5: Visualisation of some of the noise clusters between frames 1 and 2 of *voc-18-bd-6* LODV.

4.2 Varying minPts

Empirically, we found that if $minPts = 3$ did not produce $validity = 4$, we can vary $minPts$ between 3 and 7 inclusively to increase the chance of detecting proper clusters. If none of the results produced $validity = 4$, the result with the highest validity was selected. If multiple values have the same validity, then the validity with the highest number of clusters is selected, otherwise the results of the smallest $minPts$ was used. It is worth noting that this approach while it increased the time for processing each frame, our HDBSCAN implementation reduced the time needed for subsequent values of $minPts$ by only calculating distance values only for $minPts = 3$.

The reason changing the value of $minPts$ may improve the validity is due to the way HDBSCAN discovers the clusters. Given a dataset, core-distances are calculated from the distance matrix of the data. These core distance are based on the $minPts$ value and are used in constructing the MST from which the clusters are detected by finding the sub-trees with the best stabilities. Changing the value of $minPts$ changes the core-distances which propagates the changes to the MST and subsequently, the sub-tree stabilities. As such some values of $minPts$ may find more stable clusters than others.

We show in Tables 4.11, 4.12 and 4.13 that this approach either improved the average or reduced the standard deviation of the validity for all videos. In fact for all but one of the videos, both values moved closer to the acceptable values. While previously we had 8, we now had 12 videos whose average validity was greater than 3, seven of which had standard deviation less than 1. Only in *voc-18-bl-4* video was there no change observed when varying $minPts$ because none of the other values of $minPts$ produced greater validities or more number of clusters than $minPts = 3$. This is because for the video, with varying $minPts$, only two of the 65 frames selected $minPts$ other than 3 as explained in Subsection 3.4.4.

The improvement from testing varying values of $minPts$ is shown in Table 4.14. With $minPts = 3$, the validity was 0 and there were only 3 clusters detected in frame 17 and 4 detected in frame 18. By changing the value of $minPts$ to 4, we were able to detect clustering results with validity of 4 for both frames with 78 clusters in frame

Table 4.11: The clustering validity statistics for LODVs with varying *minPts* values.

Video	**Min**	**Max**	μ	σ
voc-18-bd-4	2	4	3.78	0.625
voc-18-bd-5	0	4	3.13	1.28
voc-18-bd-6	-1	4	2.65	1.783
voc-18-bd-7	2	4	3.77	0.642
voc-18-bd-8	-1	4	1.99	1.887
voc-18-bd-9	0	4	3.05	1.421
voc-18-bd-11	-1	4	3.01	1.63
voc-18-bd-20	-1	4	1.5	1.6

Table 4.12: The clustering validity statistics for MODVs for varying *minPts* values.

Video	**Min**	**Max**	μ	σ
voc-18-bd-2	-2	4	-0.141	1.138
voc-18-bd-10	2	4	3.59	0.813
voc-18-bd-16	0	4	2.57	1.317
voc-18-bd-19	0	4	3.78	0.821

Table 4.13: The clustering validity statistics for HODVs for varying *minPts* values.

Video	**Min**	**Max**	μ	σ
voc-18-bd-1	0	4	3.74	0.992
voc-18-bd-3	-1	4	1.99	1.844
voc-18-bd-12	0	4	3.84	0.727
voc-18-bd-13	-2	4	0.42	1.286
voc-18-bd-14	0	4	2.82	1.632
voc-18-bd-15	-1	4	0.82	1.519
voc-18-bd-17	-1	4	0.89	1.559
voc-18-bd-18	0	4	2.93	1.668
voc-18-bl-1	-1	2	-0.22	0.583
voc-18-bl-2	-1	4	3.18	1.633
voc-18-bl-3	0	4	3.83	0.792
voc-18-bl-4	0	4	3.84	0.799

Table 4.14: The effects of varying *minPts* in frames 17 and 18 in *voc-18-bl2*.

Frame No.	**minPts**	**Validity**	**Clusters**
17	3	0	3
	4	4	78
18	3	0	4
	4	4	75

17 and 75 in frame 18.

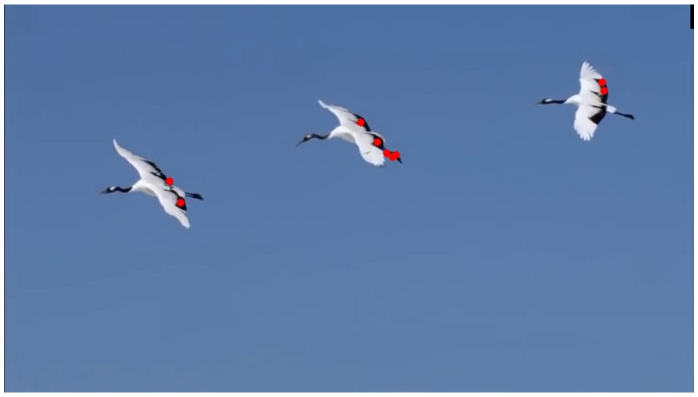
Empirically, we discovered that varying *minPts* between $[minPts, ..., minPts + 5]$ is sufficient. If the validity does not improve within that range, we discovered that using higher values does not result in better validities. We also found out that starting at $minPts = 3$ works for all classes of videos as discussed in Chapter 2. This meant that when varying *minPts*, we used values from $[3, ..., 8]$ inclusively.

4.3 Cluster Over-Segmentation with minPts = 2

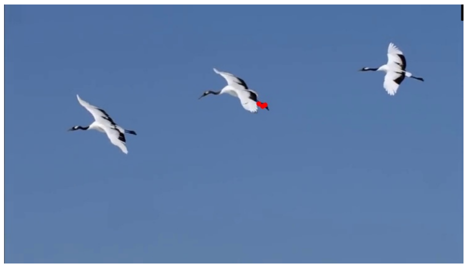
While varying *minPts* has some effect in increasing the amount of results with higher validities, we still have a lot of results with low validities. We, therefore, dropped the value of *minPts* to 2. Essentially, we attempted 1-to-1 descriptor matching which had the effect of breaking up the clusters detected by higher values of *minPts*. The overall results of this approach was that all but four the videos in our dataset had mean validities less than 4 and validity standard deviations greater than 0. However, even for those four videos, the lowest mean validity was 3.69 and the highest standard deviation was 0.394.

An obvious effect of breaking up clusters is the increase in the number of clusters detected. This effect was common across all videos and their frames. An example of this is can be seen in Figure 4.6. With $minPts = 3$, 14 clusters were discovered while 39 clusters were discovered when using $minPts = 2$. In the figure $minPts = 3$ produced cluster 12 (Figure 4.6a) with 12 features. This cluster was then split up when using $minPts = 2$ into 5 sub-clusters with the 12 features split up among them.

It is worth noting that the break up of clusters does not always happen. As seen

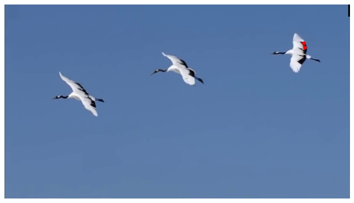


(a) Cluster 12 with 12 features for *voc-18-bd-20* when using $minPts = 3$ on frame 1.

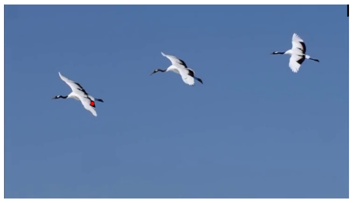


(b) Cluster 23 with 4 features for *voc-18-bd-20* when using $minPts = 2$ on frame 1.

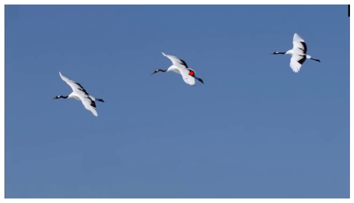
(c) Cluster 27 with 2 features for *voc-18-bd-20* when using $minPts = 2$ on frame 1.



(d) Cluster 45 with 2 features for *voc-18-bd-20* when using $minPts = 2$ on frame 1.



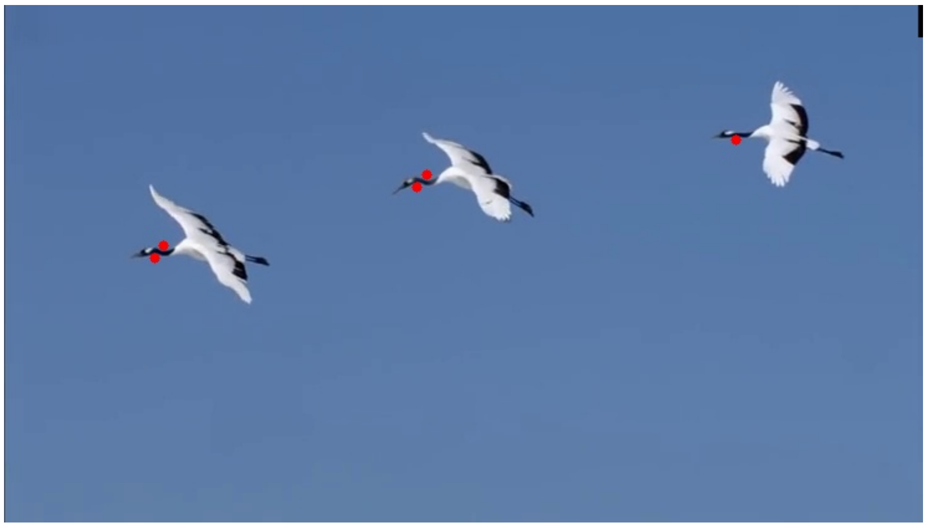
(e) Cluster 70 with 2 features for *voc-18-bd-20* when using $minPts = 2$ on frame 1.



(f) Cluster 71 with 2 features for *voc-18-bd-20* when using $minPts = 2$ on frame 1.

Figure 4.6: The splitting of cluster 12 from $minPts = 3$ into 5 clusters with $minPts = 2$.

in Figure 4.7, cluster 5 was discovered when using $minPts = 3$ (Figure 4.7a). The same cluster with the same features was discovered when using $minPts = 2$ (Figure 4.7b). This ability of the cluster to resist breaking up even when using $minPts = 2$ is a testament of how stable the cluster is within the descriptor space.


(a) Cluster 5 for *voc-18-bd-20* when using $minPts = 3$.

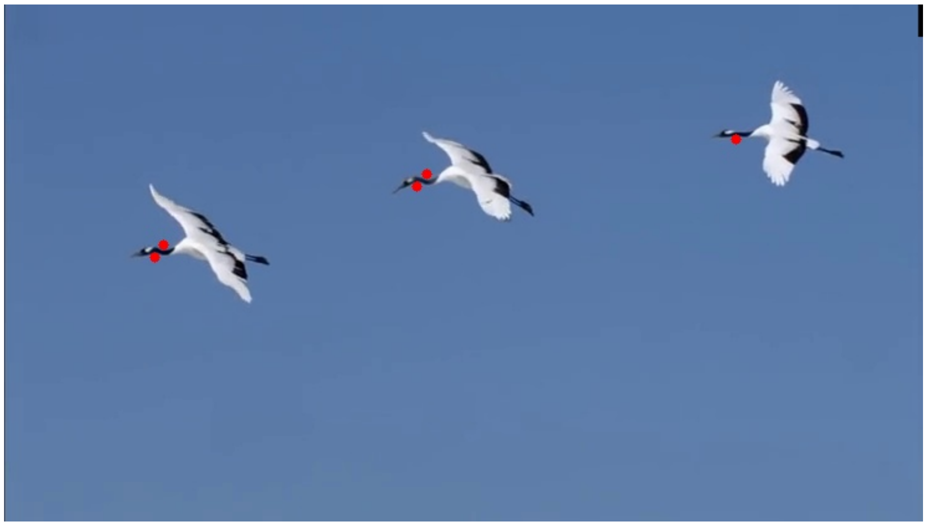
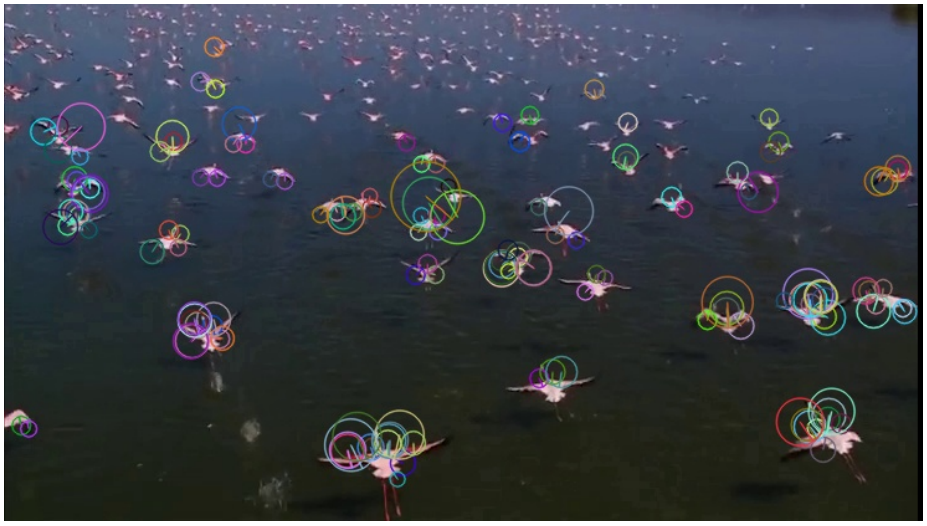

(b) Cluster 5 for *voc-18-bd-20* when using $minPts = 2$.

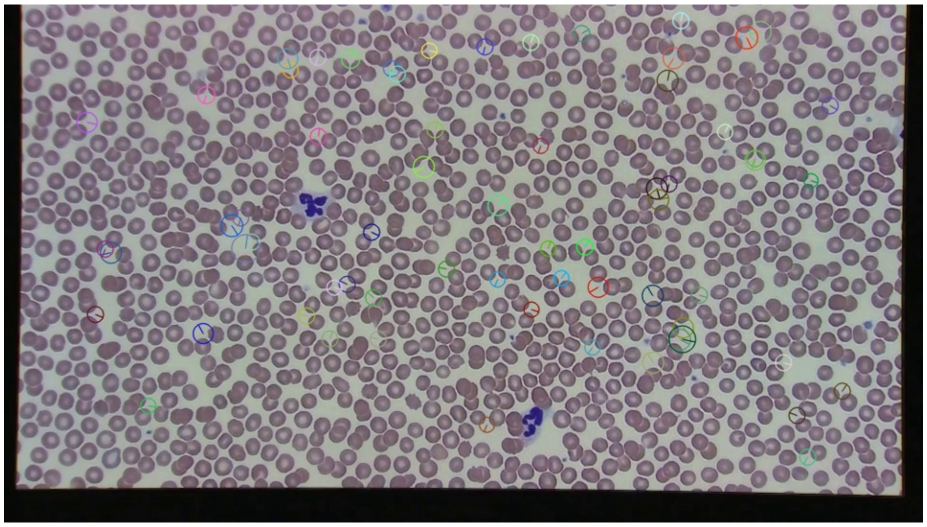
Figure 4.7: The same cluster discovered in frame 1 of *voc-18-bd-20* with $minPts = [2, 3]$.

4.4 Rotational Invariance

One of the key properties of SURF features is 'rotational invariance', which allows the same feature to be identified even if it is rotated. In our experiments, we encountered this in objects that show some symmetry to their appearance such as blood cells and flying birds. In Figure 4.8a the feature are located on both sides of the neck on two of the birds and therefore are mirror images of each other as can be seen in the angles. Figure 4.8b highlights the same rotational invariance but with features that are not directly on the blood cells.



(a) Cluster 5 for *voc-18-bd-20* when using $minPts = 3$ showing rotational invariance in SURF features.



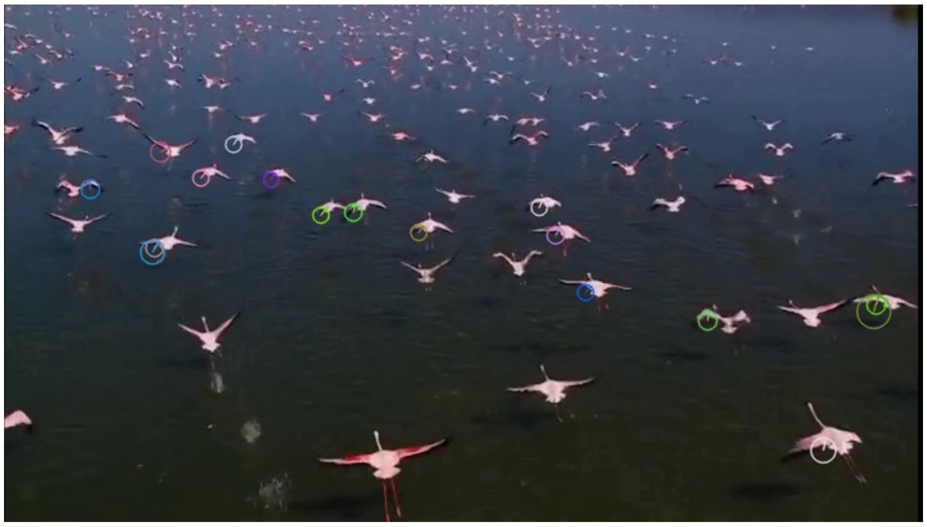
(b) Rotational invariance in blood cells.

Figure 4.8: Rotational invariance in SURF.

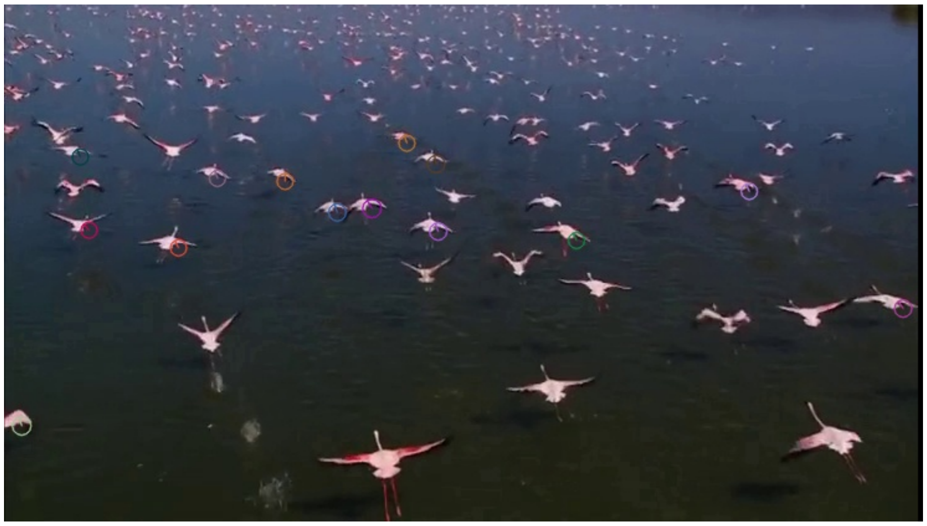
As will be demonstrated in the next chapter, ignoring angle of rotation when localising the objects has adverse effect on our localisation algorithm. One way of addressing this would be to rotate the locations detected by the difference between the sample object features and their matches. In this thesis, we instead incorporate the feature angles in to the dataset that HDBSCAN processes. OpenCV gives the keypoint angles in degrees, so we convert them to radians and the normalise them between $[0, 1]$. We then add each normalised angle to the end of the corresponding descriptor vector. This creates a new N by *65* matrix of augmented descriptor dataset such that $\vec{\mathcal{D}} = \{\vec{d_1}|\alpha_1, \ldots, \vec{d_N}|\alpha_N\}$, where α is the normalised angle in radians and $\vec{d}|\alpha$ represents the concatenation of the 64 dimension descriptor vector and the angle.

The effect of this augmentation of descriptors is that the clusters we discovered in the descriptor space are also similar in their angles. Figure 4.9, shows the results of augmenting the descriptor vectors. The cluster in Figure 4.8a was split into multiple clusters using $minPts = 3$, two of which are shown in Figure 4.9. In Figure 4.9a, one of the features is on the left side of the birds while others are on the right side. However, the angle of the feature is 'similar' to the other features. The effect of this will also be discussed in Subsection 6.2.3.

Another example is the breaking up of the cluster in Figure 4.7, however, in this case, using $minPts = 3$ resulted in only one viable cluster of three features while the other two points became noise because together they do not meet the requirements that a cluster should have at lease three points. If $minPts = 2$ is used, both the other two points showed up as another cluster of features with similar angles.



(a) Cluster 25 which resulted from the breaking up of cluster 19 in Fig. 4.8a.



(b) Cluster 38 which resulted from the breaking up of cluster 19 in Fig. 4.8a.

Figure 4.9: Results of augmenting descriptors with angles in *voc-18-bd-15* frame 17.

4.5 Summary

In this chapter, we have shown how we apply HDBSCAN to SURF feature descriptors to achieve multiple feature matching which is not possible with available matching methods. We showed that there is no proper way of using Lowe's ratio test to determine the validity of the clusters detected by HDBSCAN. We have used cluster similarity and clustering results validity methods to validate the descriptor clusters and showed the high degree of similarity in the points. The similarity was also verified by drawing the features on the frames which confirmed that visually, the features are on similar patches of the frame.

We have also discussed the cases where HDBSCAN failed to locate proper clusters within the descriptor dataset. By analysing the different validities in different frames within the video, we concluded that the acceptable clustering results are when $validity = 4$ as this signifies that the non-noise clusters have minimal noise features. We showed significant properties of failed clustering results by showing the effects of such failure on the noise cluster and the overall number of clusters. Since varying $minPts$ does not always yield $validity = 4$, we also explored cluster over-segmentation by using $minPts = 2$. We showed that this results in high average validities and low validity standard deviations. We also noted that not all clusters split up when using $minPts = 2$.

In the next chapter, we explain how we learn the object of interest features and use them to select the clusters that represent our objects. We then show how we extract the object locations from clusters, thus locating our objects of interest for counting. We show the extent to which the clusters can be used to accurately identify object location. We also present different parameters that can be used to fine-tune the counting estimation of our approach and compare them with the ground truth that was discussed in Chapter 2.

Chapter 5

Colour Model Tracking

In the Chapter 3 (Section 3.2), we introduced the concept of multi-object tracking and the main challenges of some state-of-the art MOTs. In this chapter, we introduce a different approach to tracking multiple objects which we call Learn-Select-Track. The algorithm is designed to have online training where user interaction is independent of the number of objects to be tracked while having the ability to track hundreds of objects at the same time. Although the algorithm detail in this chapter is a multi object tracker, in this thesis, we use colours as features to be tracked in the videos. The colours are then used in Section 6.5 to find and remove false positive locations from the localisation results.

The training stage is made up of learning and selection processes. The learning process discovers the best possible colour separation for the frame and the selection process relies on the user to select the colours of interest for tracking. In the *learning* stage, the algorithm analyses the colours in the video frame to find the best colour separation as clusters. In the *selection* stage, the user is given the detected colours to choose the ones they are interested in. The user interaction is therefore dependent on the number of colours detected in the video rather than the number of objects to be tracked.

We separate the colours in the frame by employing a density based clustering algorithm. We find the colours by using HDBSCAN to discover the frame's colour clusters. Due to computational intensity of the algorithm, it is impractical to use every pixel in the frame. We therefore use a local feature detection algorithm to detect feature points on the image and use the colours at those locations as input to the clustering algorithm. The tracking stage combines the colours selected from the previous frame with the colours in the current frame. HDBSCAN is used again to find the clusters in the new combined data and the previous selection is used to find the similar colours in the current frame.

The rest of this chapter is arranged as follows: in Section 5.1, we explain the detail of our MOT. We discuss the two aspects of the algorithm; the training and the tracking processes. In section 5.2, we show the results for both the training and the tracking processes. We show the successes and failures as well as the execution times for each process to highlight the online nature of the MOT.

5.1 Learn-Select-Track

This section discusses the design of the Learn-Select-Track algorithm. We first discuss the training process the algorithm goes through to find best colour separation for the first frame. The colours selected on this frame, along with the *minPts* are then used to track and update the colour model by the tracking algorithm. Since SURF features are created from the surrounding area, we use a Gaussian smoothing algorithm with a *5 by 5* kernel to remove noise at the feature location.

5.1.1 Training

In the HDBSCAN paper, the authors explained that in the dendogram, the most prominent clusters survive the longest. While their analysis was about using the same value of *minPts*, we find this holds even when varying it. Colour model training aims to detect these clusters by varying the HDBSCAN parameter *minPts* for the dataset made up of the colours at the locations of the local features from 3 to 30 inclusively. We then use the results to detect the best choice of *minPts*.

The training begins by extracting a new colour space dataset, $\vec{C}$, from the local feature dataset, $\vec{\mathcal{D}}$, and the frame, F. The resulting dataset is an *N by 3* matrix where each row contains the BGR values of the pixel. Given the dataset, incrementally varying *minPts* results in four observations of interest to the training algorithm:

- **Observation 1**: Given two values of *minPts*, $minPts_i$ and $minPts_{i-1}$, where $minPts_{i-1} = minPts_i - 1$, the cluster sets resulting from them are such that some of the clusters from $minPts_{i-1}$ merge to form a cluster that appears in the results for $minPts_i$ or become noise.
- **Observation 2**: Given two values of *minPts*, $minPts_1$ and $minPts_{i-1}$, where $minPts_{i-1} = minPts_i - 1$, if one of the clusters has the same number of points as $minPts_{i-1}$, and it is distinct from other clusters, its points will be labelled as noise for $minPts_i$.
- **Observation 3**: Changing the value of *minPts* does not affect the resulting clusters, but rather results in smaller cluster sizes as some of the outlier points become noise.
- **Observation 4**: Changing the value of *minPts* does not affect the resulting clusters in any way.

Ideally, the colour model training algorithm should detect a sequence of *minPts* values where Observation 4 occurs. However, in practice, this scenario is unlikely as clusters are not perfectly defined within the dataset. Instead, the algorithm looks for Observation 3. Within a specified range on *minPts* values, there is always a chance that there will be more than one sequence where Observation 3 occurs. The algorithm gets around this by detecting the longest sequence of *minPts*. The lowest *minPts* value in that sequence is then treated as the optimum value for colour separation.

It is worth noting that ideally, the lowest possible value of *minPts = 3* could be used to avoid repetitive cluster detection. While the similarity of the points within the clusters would be high, there is a high likelihood that there will be a high number of clusters. This provides practicality problems as asking the user to choose between a lot of colour clusters becomes tedious and error prone.

Detecting the clusters that appear for consecutive values of *minPts* can be achieved in two ways. The first approach requires direct comparison of the clusters. Given two sets of clusters for two values of *minPts*, $\vec{C}_l = \{c_1^l, ..., c_n^l\}$ and $\vec{C}_{l-i} = \{c_1^{l-i}, ..., c_m^{l-i}\}$, where l is *minPts*, n and m are the number of clusters and i is an arbitrary value such that $l - i \geq 3$, the points of each cluster in C_{l-1} have to be compared to each point in each cluster in $\vec{C}_l$. This approach develops a $\mathcal{O}(n^2)$ complexity, where n is the number of points in the dataset.

The approach used in this chapter stems from Observation 3. Starting at the first value where this observation appears, the number of cluster does not change. Since most data structures that can be used for managing clusters and their points such as hash tables, maps and dictionaries already keep a record of their size, the complexity of this approach is $\mathcal{O}(1)$. In the worst case scenario, the number of clusters have to be counted each time which results in a $\mathcal{O}(m)$ complexity, where m is the number of clusters. Empirically, it has been found that $m << n$.

Another experimental observation involves the application of the learnt colour model to subsequent frames. The results show that if the object and some of the background colours are different shades of the same colour, subsequent frames can end up with background and object colour clusters merging. While there is nothing wrong with the clusters merging if they have some similarity, the tracker can lose the colour model. Assuming $i = 1$, when the scenario in Observation 1 occurs, the cluster that results from clusters in $\vec{C}_{i-1}$ merging has more points than the sum of the merging clusters. This is because in order to merge the clusters, some of the points that were noise in $\vec{C}_{i-1}$ are brought in to form part of the new cluster in $\vec{C}_i$. The inclusion of these points into the new cluster reduces the similarity of points in the new cluster.

The final step in the training algorithm is offering the user a list of detected colour clusters for them to choose from. In this thesis, we provided the choice by drawing dots at the locations of the cluster features which helps the user visualise the colours on the frame. While the training algorithm can select a default *minPts* value, we realise that the value may not be optimal, as such, we offer the user the ability to explore other values of *minPts* from which they can potentially see the best colour clusters.

5.1.2 Track and Update

The track and update algorithm requires the learnt colour model $\vec{S}_{i-1}$ and the *minPts*, s_l, at which the model was learnt. In addition, it requires the colour datasets $\vec{\mathcal{D}}_{i-1}$ and $\vec{\mathcal{D}}_i$. A new dataset $\vec{\mathcal{D}} = \vec{\mathcal{D}}_{i-1} \cup \vec{\mathcal{D}}_i$ such that it has length $r = p + q$ where p and q are lengths of $\vec{\mathcal{D}}_{i-1}$ and $\vec{\mathcal{D}}_i$ respectively. With this arrangement, the first p points of $\vec{\mathcal{D}}$ belong to $\vec{\mathcal{D}}_{i-1}$.

HDBSCAN is then applied to $\vec{\mathcal{D}}$ with $minPts = 2 * s_l$ and the data labels, $\vec{L}$ are extracted for $\vec{\mathcal{D}}$. $\vec{L}$ is then split into two such that $\vec{L} = \vec{L}_{i-1} \cup \vec{L}_i$, where $\vec{L}_{i-1}$ has p labels for the dataset $\vec{\mathcal{D}}_{i-1}$ and $\vec{L}_i$ has q labels for dataset $\vec{\mathcal{D}}_i$. We then need to find the new labels for the selected points in $\vec{S}_{i-1}$. It is worth noting that if any of the points now has a noise label, they are ignored. Using the new $\vec{S}_{i-1}$ labels, we then find all the points in $\vec{\mathcal{D}}_i$ that have the same labels using $\vec{L}_i$. The new selected model $\vec{S}_i$ will then be used as input to the track and update for the frame F_{i+1}.

With this approach using two frames, the method in this thesis is therefore only concerned with frame-by-frame tracking. We also used HDBSCAN on D with $minPts = 2 * s_l$ so that we can get clusters that span $\vec{L}_{i-1}$ and $\vec{L}_i$. If we only used $minPts = s_l$, the resulting clusters could be such that each of $\vec{L}_{i-1}$ and $\vec{L}_i$ have their own independent clusters which would make updating the colour model impossible. In theory, if we wanted to increase temporal awareness of the track and update algorithm to y frames, the colour model $\vec{S}_{i-1}$ would not need to change, but the combined dataset would have to change such that $\vec{\mathcal{D}} = \vec{\mathcal{D}}_{i-y-1} \cup \vec{\mathcal{D}}_{i-y-2} \cup \ldots \cup \vec{\mathcal{D}}_i$.

The MOT in this chapter was tested on the dataset described in Chapter 2. While it was designed for object counting, the videos contain a variety of scenes with birds (*voc-18-bd-*{*1-20*}) and blood cells (*voc-18-bl-*{*1-4*}) which provide a good platform to test our approach. The dataset is also not annotated and does not have predefined benchmarks for common object tracking evaluation.

The training algorithm was assessed on the number of clusters the selected *minPts* produced. Ideally, the number of clusters considered to be acceptable was set to be between 2 and 10 not counting the noise cluster. The track and update algorithm was assessed on the number of frames it took before the colour model was lost. For both the training and tracking algorithms, the times were also recorded. For training, we measured the time it took for the algorithm to analyse the colours and select the best *minPts* for detecting colour separation. For tracking, we measured the time it took for the algorithm to combine the colours in the current and previous frame, detect clusters and select the colour model for the current frame.

5.2 Colour Model Training and Tracking Results

The algorithms developed in the previous section were tested separately. The training algorithm was developed to select the best colour separation in the video frame by finding the best value of *minPts*. The algorithm then requires the user to choose from the colours detected by HDBSCAN for the selected value of *minpts*. The selected *minPts*, the number of clusters and the number of clusters chosen were recorded in Table 5.1.

The tracking algorithm was used to test the *minPts* selection from the training algorithm. We looked at how long it took in terms of the video frames before the tracker lost the colour model. The effects of user choice of colours on the tracking algorithm were also tested. We also tested the times it took to learn the colour model as well as how long it took to track the features from frame to frame.

5.2.1 Colour Model Training

When evaluating the training results, we first looked at the change in the number of clusters as *minPts* is varied from 3 to 30. The important output from this part of the algorithm is the optimum *minPts* value and the selected colour model which are both used for tracking the colours from one frame to the next. The quality of the choice for these two is evaluated by looking at the tracking results.

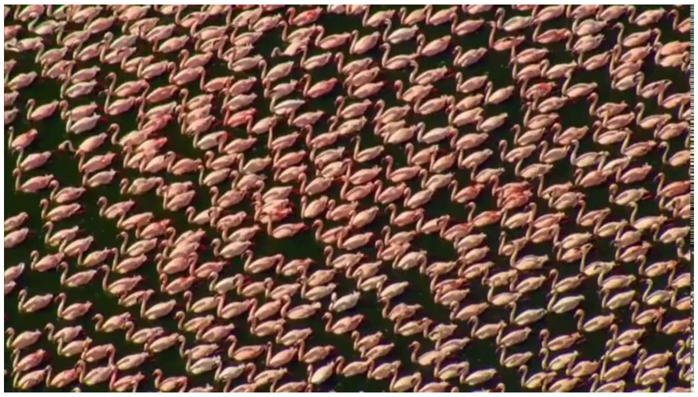
The overall trend from the videos we tested the training algorithm on show an exponential decay in the number of clusters for increasing value of *minPts*. In most cases the higher values of *minPts* results in the same clusters being detected with slight variations in the number of points in each of clusters as well as the number of clusters found.

Table 5.1: The training results with VOC-18 dataset. The ‘Points’ column shows the number of colour points detected in the first frame and ‘Time’ column shows the amount of time it took to analyse the colours by the training algorithm.

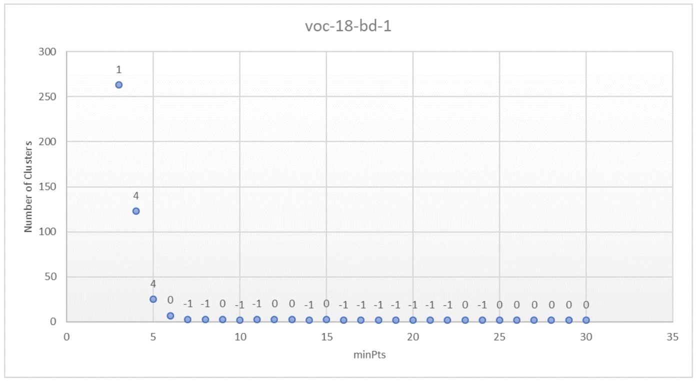
Video	**Points**	**minPts**	**Clusters**	**Time (s)**
voc-18-bd-1	3375	6	7	17.43
voc-18-bd-2	265	7	3	0.3196
voc-18-bd-3	415	4	11	0.355
voc-18-bd-4	198	6	5	0.2354
voc-18-bd-5	141	6	3	0.2049
voc-18-bd-6	246	6	4	0.2161
voc-18-bd-7	201	8	2	0.1312
voc-18-bd-8	379	3	24	0.2773
voc-18-bd-9	222	7	5	0.1869
voc-18-bd-10	219	15	4	0.1571
voc-18-bd-11	329	5	11	0.2711
voc-18-bd-12	1934	10	3	6.1631
voc-18-bd-13	185	6	2	0.1395
voc-18-bd-14	308	3	17	0.3915
voc-18-bd-15	300	4	3	0.3416
voc-18-bd-16	157	6	3	0.0673
voc-18-bd-17	304	3	24	0.1575
voc-18-bd-18	1826	3	151	4.6714
voc-18-bd-19	853	6	4	1.0212
voc-18-bd-20	129	14	2	0.0621
voc-18-bl-1	6631	13	2	37.4817
voc-18-bl-2	6010	24	5	31.601
voc-18-bl-3	3256	15	4	10.1651
voc-18-bl-4	1639	8	3	3.8409

Figure 5.1 shows the training results from frame 1 of *voc-18-bd-1* video. In Figure 5.1b, the trend can be clearly seen as the number of clusters falls from 263 with $minPts = 3$ to 2 between $minPts = \{10, ..., 30\}$. The dominance of these two clusters signifies that the video frame has two dominant colours which can be seen in Figure 5.1a showing flamingoes sitting on water and Figure 5.1c which shows how the colours of the frame were clustered. While only 2 clusters were detected for the majority of the *minPts* values, there was fluctuation between the *minPts* sequences that produced 2 and 3 clusters. As per the design of the algorithm both sequences were rejected. The final selected value resulted in 7 clusters which (Figure 5.1c) is still low enough for the choice of colours to be simple.

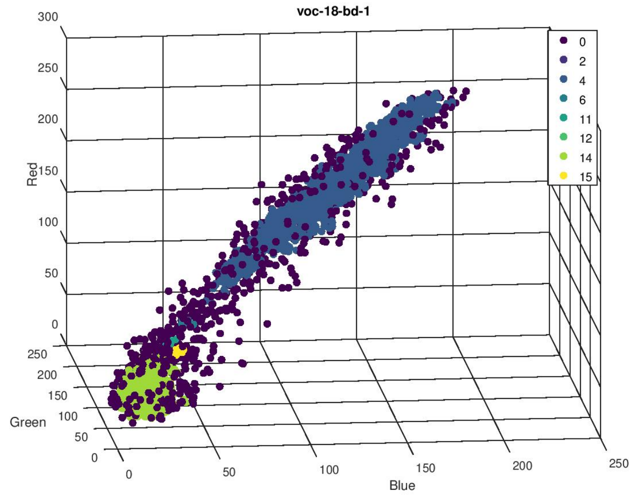
The strictness of the algorithm over the continuity of clusters between varying values of *minPts* can cause the selection of a value that has a high number of clusters as shown in Figure 5.2. In the test video *voc-18-bd-18* (Figure 5.2a), the number of clusters being discovered was irregular. This resulted in $minPts = 3$ selection, which had 151 clusters (See Figures 5.2b and 5.2c). This presents a challenge for the part of the training algorithm that requires a person to select the colours of interest because there too many clusters at $minPts = 3$.



(a) The training frame from *voc-18-bd-1* video.



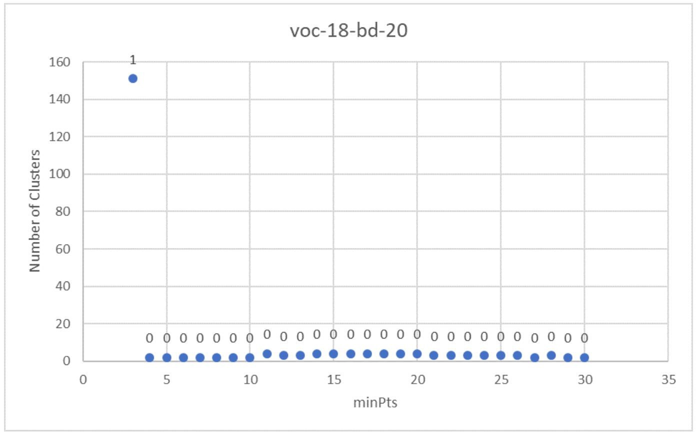
(b) The training results for *voc-18-bd-1* video.



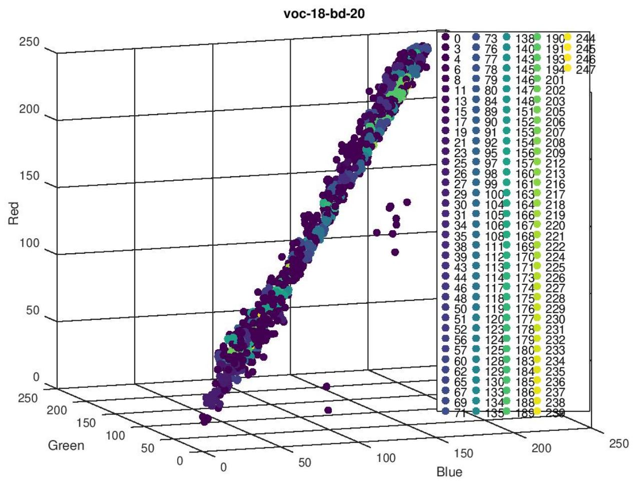
(c) A scatter plot of *voc-18-bd-1* colours and the results of the clusters at $minPts = 6$.

Figure 5.1: Training, and tracking the colour model of *voc-18-bd-1*.

(a) The training frame from *voc-18-bd-18* video.



(b) The training results for *voc-18-bd-18* video.



(c) 3D scatter graph of *voc-18-bd-18* video's first frame.

Figure 5.2: Training, and tracking the colour model of *voc-18-bd-18*.

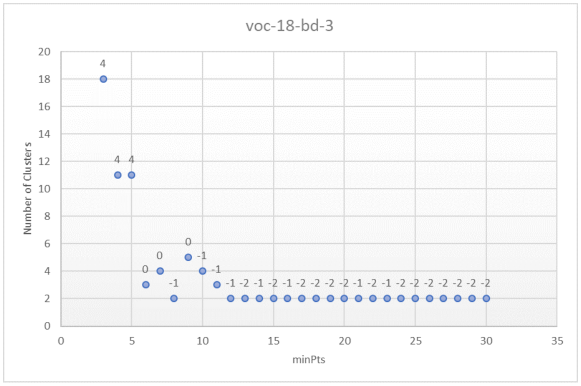
The outcome from the training algorithm is confirmed by Figure 5.2c. The figure shows that there are no visually clear separation between the colours. Instead, the colours are distributed in the frame with gradual increase such that they are in a straight line. In terms of the detected clusters, this gradual distribution of colours in the frame can be seen in Figure 5.2b where there is big change in the number of clusters from 151 to 3 between *minPts* values of 3 and 4. The subsequent values produce a fluctuating 2 and 3 number of clusters.

Another video where our algorithm had a difficult time separating the colours is *voc-18-bd-3* whose results are shown in Figure 5.3. In the video, there are birds flying over water with the sun causing a glare over the frame such that there is a gradual change in the pixel values (See Figures 5.3a and 5.3b). Furthermore, some of the birds' colours are affected by the glare making it difficult for clear separation of colours.

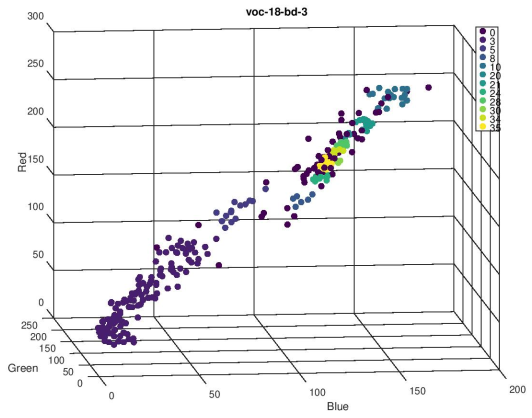
The 3D scatter (Figure 5.3c) shows that a good argument can be made for 2 groups of colours in the frame. However, the glare caused too much variance in the colours affected. This did not only affect the validity values ($minPts = \{12, ..., 30\}$), but also affected the smooth separation of the colours ($minPts = \{6, ..., 12\}$). In this case, cluster 4 was the one chosen to represent the colours of the objects of interest. The cluster however, does not represent all the birds, nor does it represent their true colours. The reason for selecting it is because it is consistent throughout the video.

In Figure 5.4, we show the choices offered to the user for selection in video *voc-18-bd-10* when $minPts = 11$ was selected for optimum cluster detection. The results for *voc-18* dataset in Table 5.1 show that for most of the videos, the training algorithm managed to select values of *minPts* with the number of clusters less than 10. We found that this number of clusters is manageable for selection of object colours. In two of the videos, *voc-18-bd-19* and *voc-18-bl-1*, the provided clusters did not offer a good selection of colours as such there were no colours learnt.

(a) The training frame from *voc-18-bd-3* video.

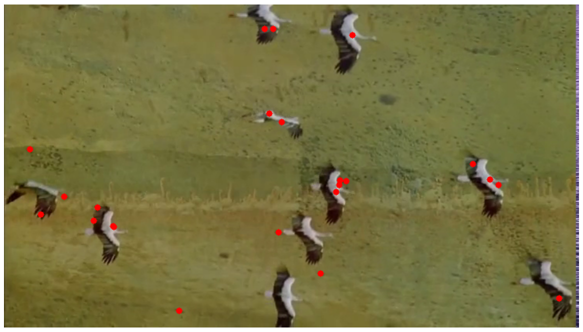


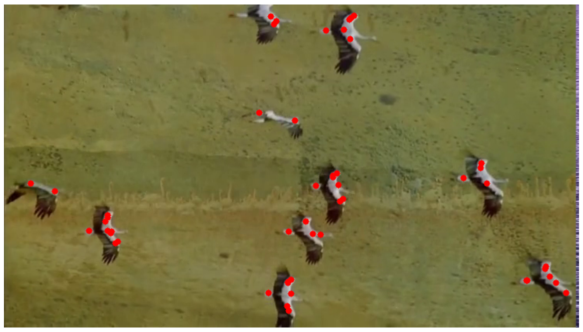
(b) The training results for *voc-18-bd-3* video. For this video, the *minPts* selected was 4 which had 11 clusters.

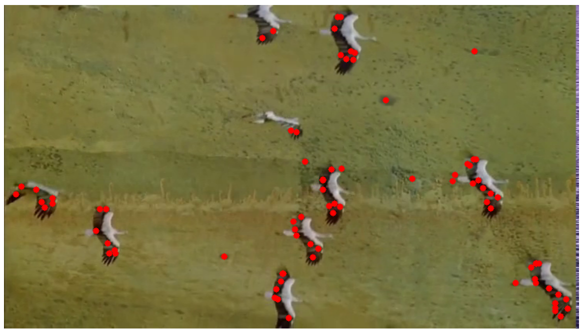


(c) 3D scatter plot of the first frame of *voc-18-bd-3* video.

Figure 5.3: Training, and tracking the colour model of *voc-18-bd-3*.


(a) The noise colour cluster.


(b) Cluster 3


(c) Cluster 4

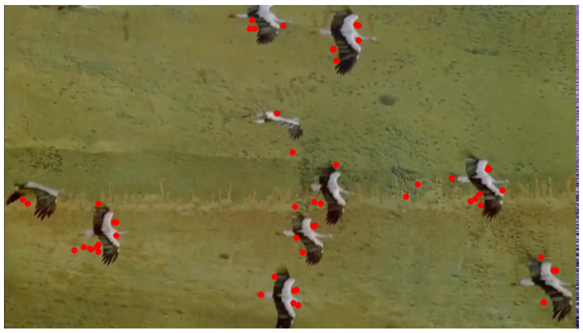

(d) Cluster 5.

Figure 5.4: Options for colour selection *voc-18-bd-10* video. With selected $minPts = 11$, 3 non-noise clusters were detected and offered as choices for the colour model.

5.2.2 Frame by Frame Colour Tracking

The tracking results are shown in Table 5.2. The table does not have results for videos *voc-18-bd-18* and *voc-18-bl-1* because the training algorithm did not successfully learn the colour model. The *"Tracker Lost at"* column shows the first frame at which the tracker loses the selected colour model. In analysing the result from the tracker, we considered two scenarios; instant failure (colour model lost in less than 10 frames), and successful tracking.

In the 22 videos where the colour model was successfully learned, 7 of them fell in the instant failure group. In those videos, the results in the frame when the colour model was lost showed either a significant decrease in the number of clusters detected or a significant increase in the number of points in the selected colour model (See Table 5.3). In terms of the number of clusters, the effect is that the points that were rejected in the previous frame end up being labelled with the colour model points which causes the tracker to lose the model. This also leads to an increase in the number of points. In cases where the number of clusters did not change, there was a significant change to the structure of the clusters. But since we were not doing deep cluster analysis the tracking algorithm did not recognise the loss of the model.

In videos such as *voc-18-bd-12* (Figure 5.5) where there is a distinct difference of colours between the difference in colours of interest and the background, our algorithm successfully tracked the chosen colours from beginning to end of the video. This was observed in 10 of the videos which are characterised by high contrast between the

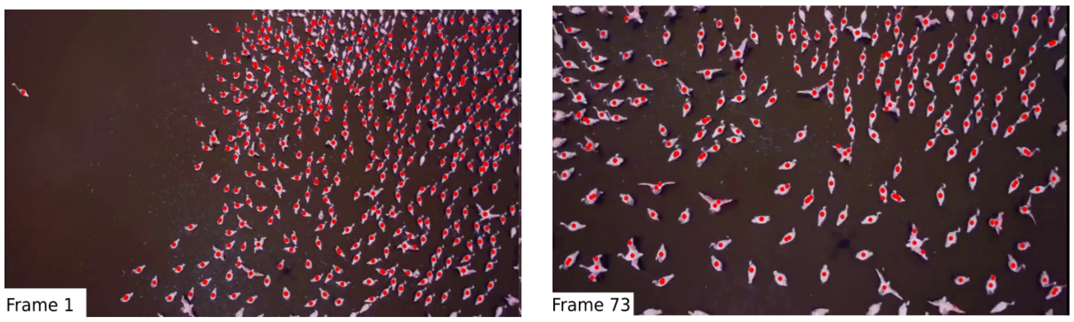
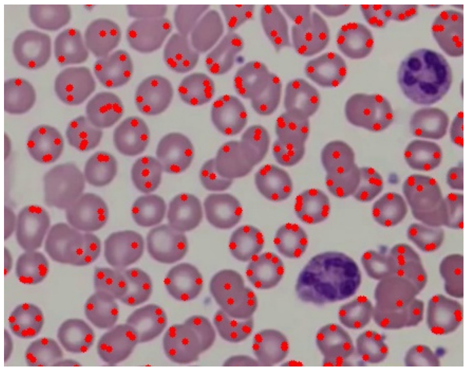


Figure 5.5: The tracker results for *voc-18-bd-12* video showing successful colour model tracking from the first to the last frame.

objects and the background. The exception is *voc-18-bd-19* where the building had a lot of colours similar to the birds and as such the building colours were part of the colour model from the beginning. The rest of the videos in the dataset showed varying degrees of successful tracking.

The effect of user selection of the colours during the algorithm training can be seen in *voc-18-bl-2* video. Figures 5.6a and 5.6b depict clusters 5 and 6 that the training algorithm offered as some of the possible choices for the colour model. Cluster 5 points (Figure 5.6a) are on the darker side of the blood cells while cluster 6 points (Figure 5.6b) are in the middle of the cells. However, the middle of the cells have colours similar to the background. Selecting cluster 6 as part of the tracked colours resulted in the introduction of noise in the 34th frame. If cluster 5 is the only one selected, the tracking algorithm successfully tracked the colours in all the frames.


(a) *voc-18-bl-2* video cluster 5. The points are on the edge of the cells where the colours are more distinct.

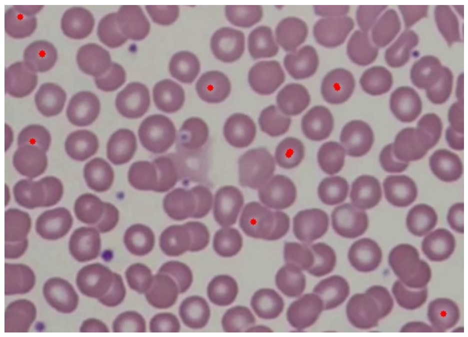

(b) *voc-18-bl-2* video cluster 6. The points are in the middle of the blood cells which is a lot similar in colour to the background.

Figure 5.6: The problem posed by object colours that are similar to the background colours.

Table 5.2: The results of tracking the colours based on the selected colours.

Video	**Length**	**Tracker Lost at:**
voc-18-bd-1	77	77
voc-18-bd-2	71	51
voc-18-bd-3	103	103
voc-18-bd-4	65	4
voc-18-bd-5	56	56
voc-18-bd-6	37	10
voc-18-bd-7	87	2
voc-18-bd-8	85	2
voc-18-bd-9	143	10
voc-18-bd-10	121	121
voc-18-bd-11	155	2
voc-18-bd-12	73	73
voc-18-bd-13	99	22
voc-18-bd-14	110	5
voc-18-bd-15	115	115
voc-18-bd-16	75	3
voc-18-bd-17	85	72
voc-18-bd-19	117	117
voc-18-bd-20	93	76
voc-18-bl-2	94	94
voc-18-bl-3	80	6
voc-18-bl-4	49	11

Table 5.3: The table showing the videos where the tracker lost the model instantly.

Video	**Clusters (I-1)**	**Clusters (I)**	**Points (I-1)**	**Points (I)**
voc-18-bd-6	7	4	10	67
voc-18-bd-7	3	3	9	67
voc-18-bd-8	25	4	36	190
voc-18-bd-11	12	3	29	215
voc-18-bd-14	7	2	16	30
voc-18-bd-16	3	3	24	42
voc-18-bl-3	5	2	1291	2318

5.2.3 Training and Tracking Times

The bulk of HDBSCAN's time complexity is dominated by the distance calculation. The distance matrix is symmetric with 0 on the principal diagonal such that $d_{i,j} = d_{j,i}$ and $d_{i,j} = 0$ if $i = j$. This property of the distance matrix allows for significant reduction in the memory requirements by only storing the top half of the matrix. If the length of the dataset is denoted by N , then the new distance matrix as a vector will have length $N_d = N(N-1)/2$. Our implementation takes advantage of this property to speed up distance calculations and reduce required memory. However, the overall complexity of the algorithm is still $\mathcal{O}(n^2)$. The colour model training times can be seen in Table 5.1 while the complexity has been highlighted in Figure 5.7.

Table 5.4 shows the average number of point per frame for each of the videos in the dataset and the average time it took for each frame. The data in the table has also been arranged in ascending order of the average number of points. It is worth noting that the number of points per frame may vary widely as they rely on the number of objects in the video. It is also worth noting that for tracking, we use points from two frames which means per frame, the algorithm is processing around twice the number of points. The tracking algorithm still relies on HDBSCAN which mean it is also a $\mathcal{O}(n^2)$ as can be seen in Figure 5.8.

Table 5.4: The average number of features being tracked per frame and the average time taken to process each frame arranged in ascending order.

Video	**Average Points Per Video**	**Average Duration Per Frame (s)**
voc-18-bd-2	47	0.06234171
voc-18-bd-20	106	0.03249336
voc-18-bd-16	144	0.00843991
voc-18-bd-9	172	0.01150438
voc-18-bd-5	193	0.05855851
voc-18-bd-13	199	0.05959434
voc-18-bd-7	218	0.03284515
voc-18-bd-17	256	0.03664438
voc-18-bd-15	270	0.03493461
voc-18-bd-4	271	0.06568578
voc-18-bd-14	290	0.05997002
voc-18-bd-6	293	0.15951065
voc-18-bd-9	315	0.06294197
voc-18-bd-10	359	0.07135481
voc-18-bd-11	401	0.18854269
voc-18-bd-8	426	0.09738525
voc-18-bd-3	509	0.10776893
voc-18-bd-19	951	0.15017254
voc-18-bl-4	1659	0.91024479
voc-18-bl-3	1912	1.03131769
voc-18-bd-12	1918	1.47564444
voc-18-bl-18	1950	0.83814666
voc-18-bd-1	3271	3.08022

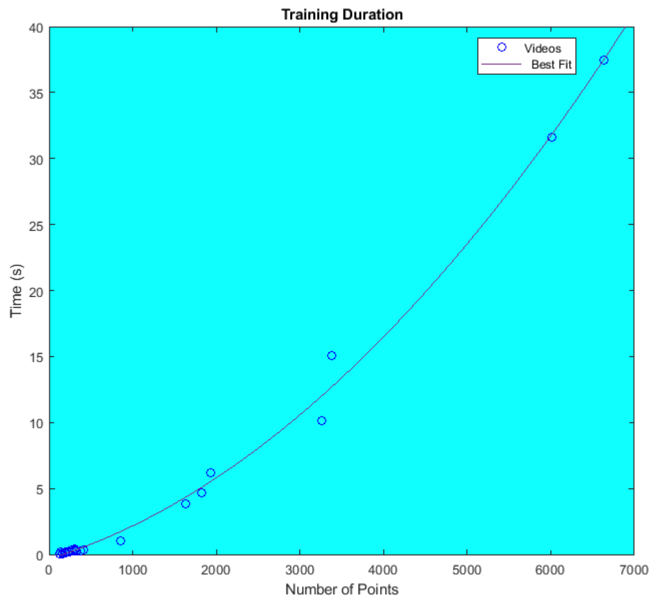


Figure 5.7: The graph of number of points vs time for learning the object colour model on the first frame.

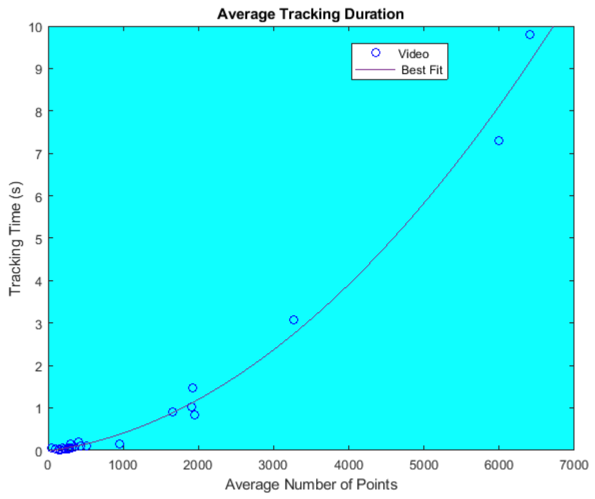


Figure 5.8: The graph of average number of point per frame vs the tracking duration.

5.3 Summary

In this chapter we have presented an approach for tracking colours of interest in multiple locations in videos without the need for the user to directly input the locations. The approach was tested on videos with different properties such as cluttered scenes, distinct background/foreground and similar background/foreground. We tested the approach on 24 videos with varying degrees of success. While the chapter uses colours for tracking, this approach can work with other features that are matched using euclidean distance.

Out of the 24 videos, the training algorithm was able to select *minPts* values where the number of clusters was 10 or less in 20 of the videos. In the 6 videos where this observation did not hold, the results showed a fluctuating number of clusters and validity values for all values of *minPts*. Future improvement on the training algorithm could include an in-depth analysis of the clusters for varying values of *minPts* to detect the points that cause the validity of the results to reduce. The points can then be manually removed from the results in order to obtain more stable clusters.

The most important feature of the training algorithm in this chapter is the online training capability that reduces the user input. It also overcomes the weaknesses of deep learning techniques by reducing the amount of work, time and computational requirements. While deep leaning approaches rely on user labelled data and supervised offline training, our approach reduces the user interaction to selection of detected feature clusters. This reduces the training time from hours and days to less than a minute even when processing thousands of features. The training data requirements are reduced to just a single frame.

The tracking algorithm showed varying degrees of success, but it is more about the weakness of colours as a unique feature. In videos that had distinct foreground/background colours, the algorithm successfully tracked the colours for all frames in the video. In cluttered scenes of similar foreground/background videos, the tracking algorithm was able to track the objects for a while before background colours started appearing in the results. The latter of the two types of videos showed more susceptibility to erroneous results.

In the next chapter, we show how to extract the objects of interest locations from the clusters detected in Chapter 4. The localisation algorithm described in the chapter also gives some false positive location. In Section 6.5, we show how to use the colour model from this chapter to detect and remove the false positive locations.

Chapter 6

Object Locations and Count Estimations

In Chapter 3, we introduced SOT and tested the algorithms implemented in OpenCV. We showed that different SOT algorithms had varying degrees of success on our dataset with BOOSTING tracker performing the best. In Chapter 4, we demonstrated that given SURF features in a frame, we can use HDBSCAN to detect clusters of descriptors which represents multiple feature instances matches. We showed that these would fail Lowe's ratio test which was designed for evaluating one-to-one matches. We also demonstrated that the validity evaluation method described in Subsection 3.4.5 can be used successfully to determine whether the clustering results were good or not.

In this chapter, we use the user-given ROI to learn the object of interest's features. We select the non-noise clusters that intersect with the ROI to detect and localise the object instances in each cluster. In this thesis, a cluster C intersects with an ROI r, if there is at least one feature whose spatial location is in side r. We then combine locations from each cluster into frame-wide locations which we use to represent the object count estimations. Additionally, we show how we detect occlusions as well as objects whose clusters do not intersect with the initial ROI.

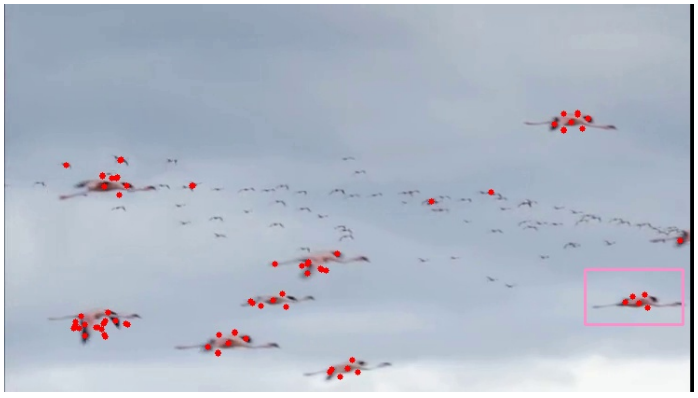
This chapter is organised as follows: in Section 6.1, we discuss how to learn the object of interest features, track the the object from frame to frame and re-initialise the tracker when the object going off the frame. In Section 6.2, we demonstrate how to find other object using the ROI and the intersecting clusters. Rotational variances are handled in 6.2.3 and in 6.4, we explain how to find objects whose clusters do not intersect with the original ROI.

6.1 Object of Interest Feature Learning and Tracking

In Chapter 3, we explored the SOTs in OpenCV and found that BOOSTING tracker provides the most stable tracking abilities and as such was used in this thesis. In this section, we explain how we use the SOT to provide a template initial object for our object counting approach. We employ the SOT in our approach so that the user interaction is limited to selecting the ROI for the object of interest only once. As such, regardless of how many objects are in the frame, the user interaction is always $\mathcal{O}(1)$.

Given a set of frame keypoints, $\mathcal{K}$, we use the ROI, r, to select $\hat{\mathcal{K}} \subset \mathcal{K}$, where for each $k \in \hat{\mathcal{K}}$, the feature location p is inside r. We detect the clusters in the descriptor dataset $\mathcal{D}$ and extract a set of clusters $\mathcal{C}$ as described in chapter 4. For each $k \in \hat{\mathcal{K}}$, we extract ROI clusters $\hat{\mathcal{C}} \subset \mathcal{C}$ and ROI cluster labels $\hat{\mathcal{L}} \subset \mathcal{L}$ such that for all $C_{\hat{j}} \in \hat{\mathcal{C}}$, $l_{\hat{j}} \in \hat{\mathcal{L}} \neq 0$. Essentially, we are using this approach to learn the good features of our object which we then use in the next section to localise object of interest instances in the frame.

In our video dataset, we compiled videos with a lot of motion, both in terms of the camera and the object themselves. As such, our tracking has to handle the chance that the target object will drift off the frame. In this thesis, we address this issue by using one of the objects detected in the previous frame. We set a ten pixel boundary around the edge of the frame. If the initial object breaches the boundary, the object is abandoned and a new object location is selected from the previous frame's detected locations to re-initialise the tracker. We first order the previous frame's located object in descending order of their matching criteria which will be explained in Section 6.2. We then select the object that best matches the frame's initial object, the result of


(a) The target object about to breach the ten pixel barrier.

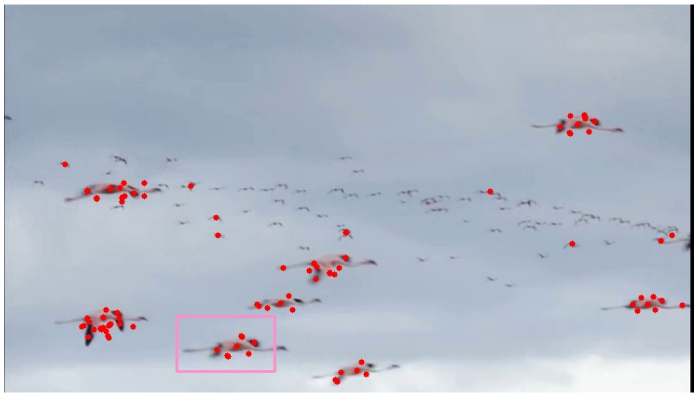
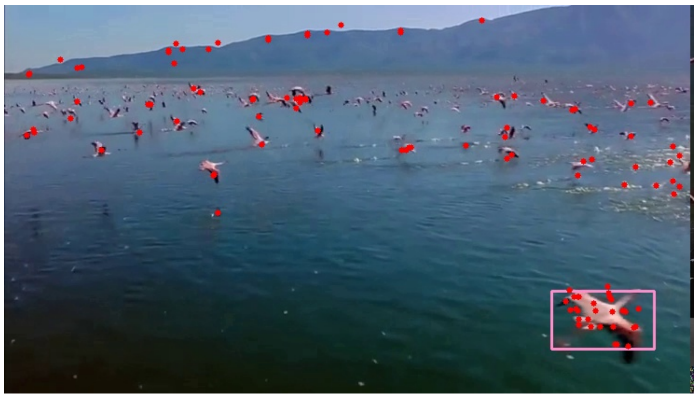

(b) New target object selected from the previous frame.

Figure 6.1: Re-initialisation of the tracker when the target breaches the ten pixel barrier.

which is shown in Figure 6.1.

The re-initialisation of the tracker approach did not work well in some of our videos. Most notably, *voc-18-bd-16*, which has one of the most extreme cases of scale and vanishing point in our dataset. In fact, it is one of the two videos that have all the challenges as outlined in Chapter 2. In the video, the initial object is very close to the camera while a lot more birds are further away. As will be explained in Section 6.2, the scale problem caused the re-initialisation algorithm to completely lose the initial object as can be seen in Figure 6.2. In the figure, the initial object in frame 23 is about to move off the frame, however, the scale of the other objects and other challenges


(a) The target object about to breach the ten pixel barrier.

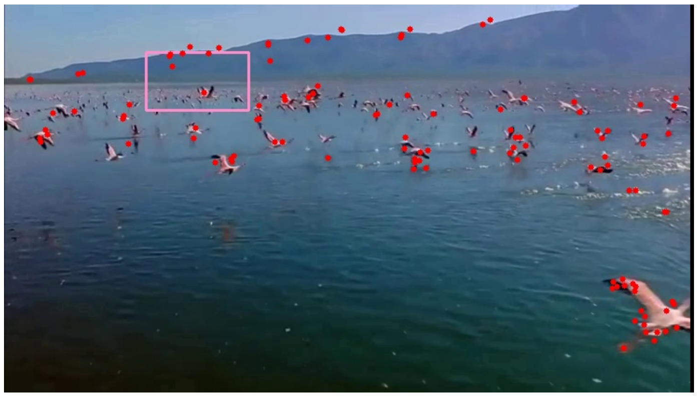

(b) Re-initialisation failure to re-initialise the object in the next frame.

Figure 6.2: The effect of scale in re-initialising the tracker when the object moves off the frame.

caused the localisation algorithm (Section 6.2) to fail as such the re-initialisation algorithm selected the area of the frame that contained the object but with a rather large ROI that far exceeded the size of the object.

The feature learning approach used in this thesis is comparatively better than the offline training methods discussed in Section 1.1 [10, 18, 33]. The advantage with our learning approach is that we do not need to know what the object looks like beforehand. In fact, the feature learning does no computationally intensive work as it just looks for features within the initial ROI. The computationally intensive process of identifying non-noise ROI features is done as part of the matching algorithm defined in Chapter 4. This means that at most, the learning algorithm takes as long as the combination of matching (see Section 4.1) and localisation (see Section 6.2) tasks.

6.2 Object Localisation

Homography [90] is a projective mapping from one plane to another. Given two matched points p_1 and p_2, homography provides a mapping such that $p_1 = H * p_2$, where H is a *3 by 3* matrix as shown in Equation 6.1. In feature matching and object localisation, a minimum of four points are needed to solve for H [91]. In this thesis, we push the limit of matched features to just one per object. As such, we cannot use homography to localise the object instances in the frame.

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \tag{6.1}$$

In this section, we discuss how we extract the location of objects from the descriptor clusters. In addition, we explain how our localisation handles the different challenges introduced in Chapter 2. We describe how different dataset challenges detailed in Chapter 2 affect the localisation algorithm and possible mitigation techniques employed.

6.2.1 Cluster Objects Localisation

In this thesis, we take a different approach to object localisation. Utilising SURF features' precision, we use Algorithm 4 to find the location of object of interest instance from just one match. Given a non-noise cluster $C_{\hat{\jmath}} \in \hat{\mathcal{C}}$ that intersects with the ROI r, we find the intersecting feature location p_t, and for each of the other features locations p_l, we calculate the difference between the two locations and use it to shift r to the new location. In cases where there is more than one feature in a cluster within the r, we set p_t to be whichever ROI feature that p_l is closest to the centre of r. We also calculate a matching score by using moments to compare the template object bounded by r with the new object bounded by r_l.

Algorithm 4: Detecting the new object location using the locations of a sample feature and a cluster feature.

```
Data: $r$ - region of interest
Data: $p_t$ - location of the sample feature
Data: $p_l$ - location of the point in the same cluster with $p_t$
Result: $r_l$ - Detected object ROI
Result: $m_l$ - Matching score based on moments comparisons.
$p_s \leftarrow p_t - p_l$
$r_s \leftarrow shift(r, p_s)$
$\hat{r_s} \leftarrow r_s * 2$
$r_l \leftarrow templateMatch(\hat{r_s}, r)$
$m_l \leftarrow momentsCompare(r_l, r)$
```

As accurate as SURF can be, the new ROI r_s is almost always slightly off the located object instance (See Figure 6.3a). In our approach, we compensate for this by using localised template matching [92]. We first increase the width and height of the r_s by half before performing the template match with r. Then we locate the best matching coordinates and re-centre r_s at the coordinates as depicted in Figure 6.3b obtaining the final object location r_l. While this method does manages to correct the locations of the new ROIs, it also suffers from occlusion influence. In Figure 6.3b, the top right location has been shifted off the bird that actually influenced the feature due to occlusion. This approach to localisation also shows how we are able to successfully detect the object instances while using just one feature per instance instead of four as required when using homography.

The localisation algorithm relies solely on the similarity of the features in the cluster. In some cases, the clusters contain features that are similar but are visually on different parts of the objects. Such clusters then cause the localisation algorithm to create some false positive locations as shown in Figure 6.4. In Figure 6.4a, the initial ROI feature is on the neck of the bird, as are some of the features in the clusters. However, some of the features in the cluster are on the bellies of the birds which leads to false positives locations.

It is worth noting that in this instance, the frame clustering results had validity of 4. The cluster had intra-cluster distance ratio of 0.277 and confidence of 98.61%. In

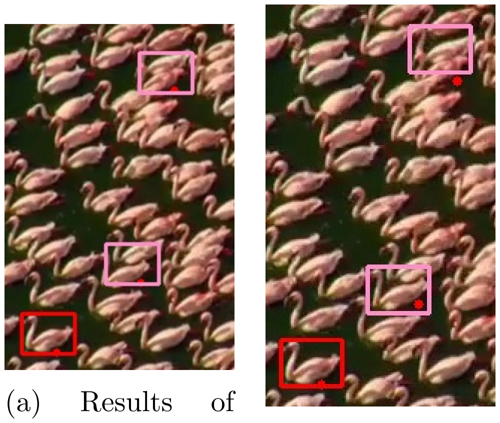
Table 6.1: Selected clusters in frame 10 of *vo-18-bd-1*.

		Core Distances				**Intra-Cluster Distances**			
Cluster	**Number of Points**	**Min**	**Max**	**Ratio**	**Confidence %**	**Min**	**Max**	**Ratio**	**Confidence %**
37	5	0.242	0.273	0.888	60.3	0.188	0.336	0.561	99.31
55	12	0.22	0.263	0.84	58.06	0.149	0.537	0.277	98.61
89	17	0.161	0.249	0.645	45.42	0.155	0.502	0.309	98.76
169	36	0.137	0.23	0.596	40.91	0.119	0.459	0.259	98.52
170	22	0.132	0.229	0.579	39.18	0.121	0.435	0.278	98.62
238	53	0.129	0.216	0.595	40.82	0.072	0.518	0.138	97.22

comparison, clusters 169 and 238 had distance ratios of 0.259 and 0.138 respectively (See Table 6.1), and as shown in Figures 6.4b and 6.4c, localisation had more accurate locations. This is a further confirmation that distance ratios cannot be used to filter out clusters that might cause false positives.

6.2.2 Scale Challenges

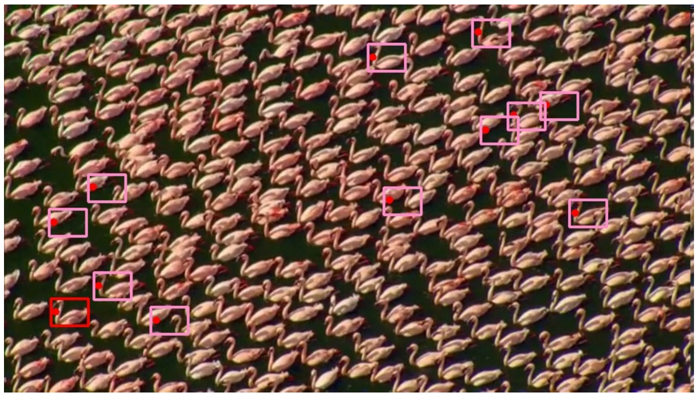
SURF features detect scaled features by using image pyramids and applying box filters directly on the integral image [24]. Rather than iteratively reducing the image



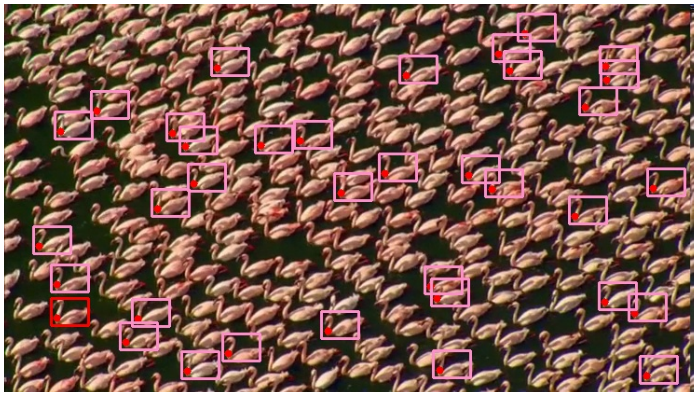
(a) Results of shifting without compensating for slight variations in locations of features.

(b) Variations in feature location compensated by localised template matching.

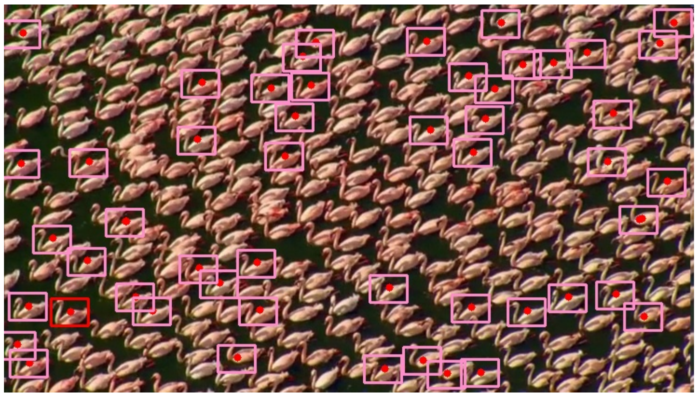
Figure 6.3: Comparison of raw shifting and shifting with template matching.



(a) False positive locations due to a cluster 55 with visually different features.



(b) Correct locations detected from cluster 169.



(c) Correct locations detected from cluster 238.

Figure 6.4: Locations of birds in *voc-18-bd-1* for different clusters. The first sub-figure shows the false positive locations while the other two show accurate localisations.

size, SURF analyses the scale space by up-scaling the filter size. This approach increases computational efficiency compared to SIFT [34]. The scale at which the feature is detected is also used to determine the size of the area around the keypoint from which the descriptor is calculated. This means that for features that are on the objects, the size of the keypoint can signify the size of the object as can be seen in Figure 6.5. The feature locations are in the centre of the objects. Since the objects have the most influence on the descriptor, the size of the keypoint area reflects the size of the objects and suggests that the size of the keypoint area can be used to scale the ROI size.

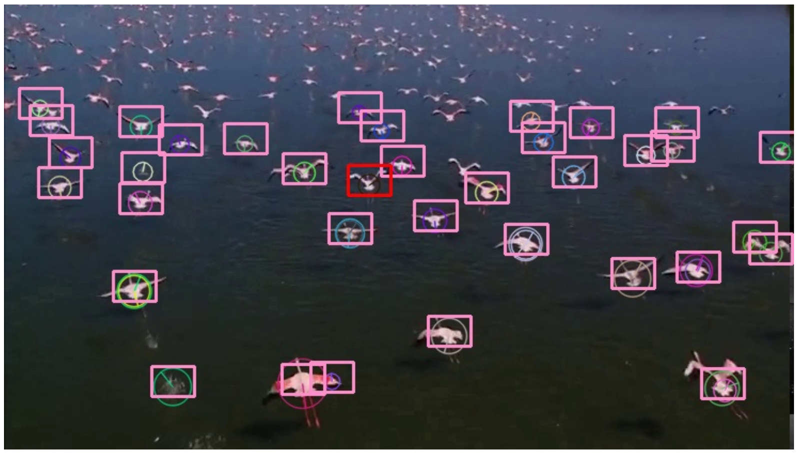
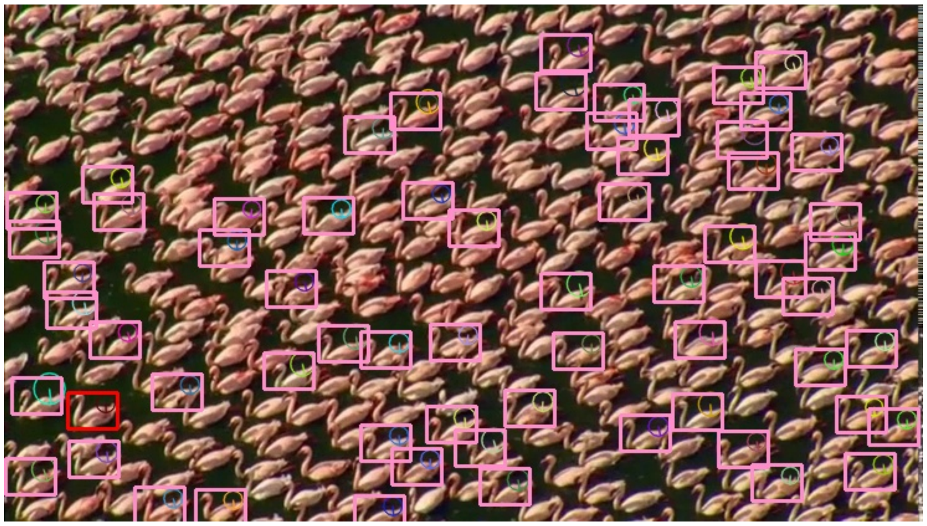


Figure 6.5: Cluster features drawn to show scale invariance of the SURF features.

In this thesis, we do not scale the located object ROI because we discovered that features that are not on the objects or are heavily influenced by the object surroundings can have widely different scales even if the objects are of relatively same size as shown in Figure 6.6. In Figure 6.6a, the features are located off the objects, but have been used successfully by our localisation method to locate the objects. However, they have different sizes and since the objects are of the same size, adjusting the ROIs for scale would not properly locate the objects. The same problem would be encountered in Figure 6.6b where even though the features are located on the objects, there is still a lot of influence from the surrounding which led to significant differences in feature sizes.

An extreme case of localisation failure due to scale challenges can be seen in Figure


(a) Features on the object surrounding with different feature sizes.

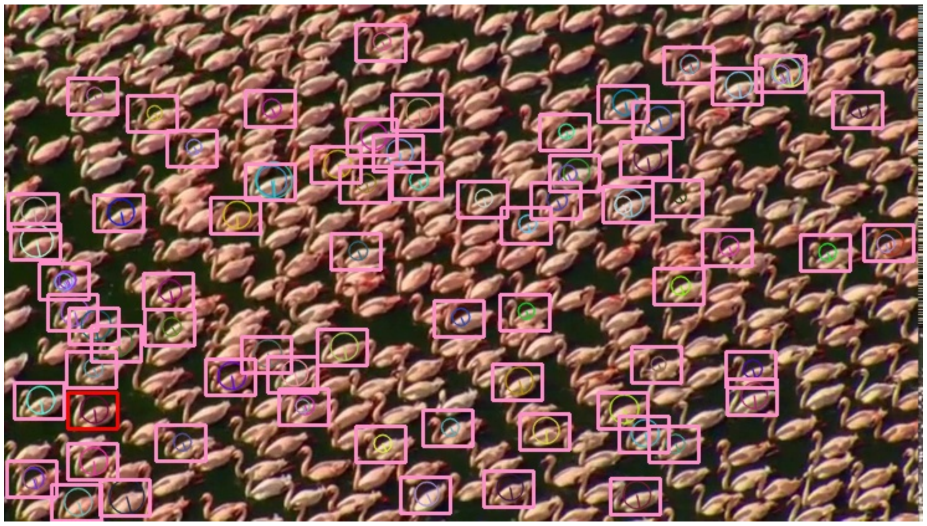

(b) Features on the object but being influenced by the object surrounding.

Figure 6.6: Evidence of feature scale not representing object size.

6.7. This is the last frame with accurate initial object that was mentioned in Section 6.1 and the value of $minPts = 3$ was used. Due to the extreme scale challenge between the initial object and the other objects in the frame, the localisation algorithm failed to properly locate any other object. The results of such localisation failure for each cluster can be seen in Figure 6.7a which shows all the cluster locations combined.

As we show in Table 6.2, the intra-cluster distance ratios in the selected clusters were

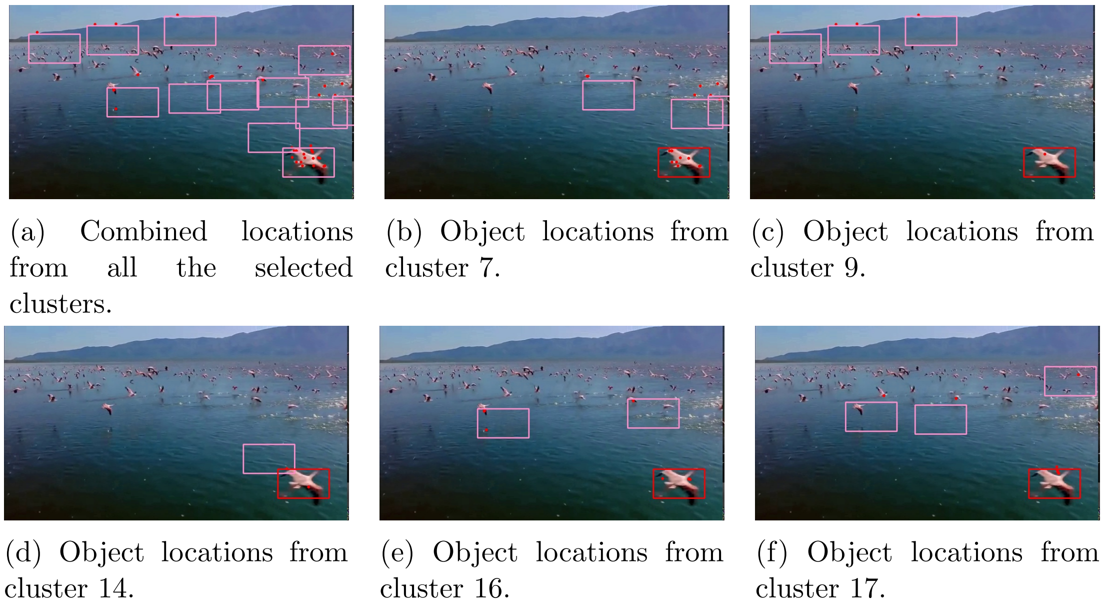

(a) Combined locations from all the selected clusters.
(b) Object locations from cluster 7.
(c) Object locations from cluster 9.
(d) Object locations from cluster 14.
(e) Object locations from cluster 16.
(f) Object locations from cluster 17.

Figure 6.7: Failure of localisation algorithm to find the other objects due to extreme scale challenges in *voc-18-bd-16*.

very low, with cluster 7 being the lowest with the ratio of 0.05 and cluster 16 being the highest with a ratio of 0.27. The extreme case in this video can also be observed when detecting clusters using $minPts = 2$ as shown in Table 6.3. In this case, the largest ratio from using $minPts = 2$ where the number of features was greater than 2 is cluster 10. In fact, apart from cluster 10, all the other clusters with the number of features greater than 2 are all below the ratio of 0.5. The clusters that had only two features were composed of features on the same initial object.

Table 6.2: Distance values of the selected clusters in *voc-18-bd-16* frame 23 with $minPts = 3$.

Cluster	**Num. of Features**	**Min Distance**	**Max Distance**	**Ratio**
7	17	0.0375719	0.796797	0.0471537
9	5	0.0608773	0.335271	0.181576
14	3	0.0401456	0.323144	0.124234
16	8	0.156832	0.59152	0.265133
17	8	0.0811655	0.686787	0.118181

Table 6.3: Distance values of the selected clusters in *voc-18-bd-16* frame 23 with $minPts = 2$.

Cluster	**Num. of Features**	**Min Distance**	**Max Distance**	**Ratio**
5	2	0.33764	0.33764	1
10	4	0.405366	0.619812	0.654013
13	5	0.0608773	0.335271	0.181576
25	4	0.0375719	0.484922	0.0774802
26	3	0.0401456	0.323144	0.124234
29	2	0.314023	0.314023	1
33	2	0.377196	0.377196	1
37	6	0.0811655	0.521528	0.15563
38	5	0.283829	0.589078	0.481819
48	3	0.156832	0.316224	0.495952

6.2.3 Rotational Invariance

In Section 4.4, we showed that without taking into consideration SURF features' rotational invariance, we get clusters where the angles may widely vary. We also showed how we can augment the features with the angle such that HDBSCAN will detect clusters of features with similar angles. In this section we show the effect of localising the objects with and without the angles of rotation.

In Figure 6.8a, we localised the objects of interest from Figure 4.8a. As evident from the figure, there are a lot of false positive locations. In comparison, Figures 6.8b and 6.8c show that with augmented descriptors clusters reduce the number of false positives. The feature that was identified in Figure 4.9a can be seen in Figure 6.8b having resulted in a false positive location. In this thesis, we use rotational variance as one of the parameters we can use to fine tune the results of our counting approach. In line with the aims in this thesis of reducing technical understanding of the data, the use of this parameter only requires the visual inspection of the object to determine if the objects in the video have symmetry or not.

6.2.4 Occlusion and Partial Visibility

One of the challenges posed by our dataset as discussed in Chapter 2 is occlusion and partial visibility. Our localisation algorithm relies on precision feature description by SURF and assumes that every feature match can be used alone to identify the location of the object. We have shown that this is true in the subsections above. As such, partially visible objects can be properly identified as long as there is a matched feature on or around the visible part of the objects as shown in the bottom left of Figure 6.8c.

Detection of partially visible objects relies on whether there were SURF features detected in and around the partially visible objects and whether the features formed clusters with any feature inside the ROI. In order to allow for the creation of the oriented quadratic grid around the detected feature, SURF leaves a boundary at the edges of the frame (Figure 6.10a). As such, a partially visible object has to extend inside the boundary in order for any features to be detected on it. Figure 6.10b shows the detection of partially visible objects where the partially visible birds were inside the boundary and features were detected on their tails. The features then formed a cluster with the feature on the tail of the bird inside the initial ROI (the red rectangle). Our localisation algorithm takes into consideration the possibility of partial visibility which leads to the detected object ROI exceeding the boundaries of the frame and as such was able to properly locate the birds.

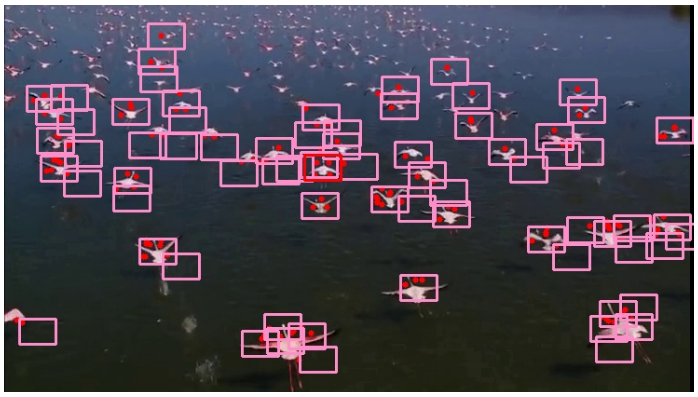
In terms of occlusion, our localisation algorithm has two weaknesses. The first weakness is the localised template matching. In Figure 6.3a, the cluster provided 3 features which show features located below the flamingoes. One of the flamingoes which has a feature is partially occluding the one above it. However the occlusion is small so the local template matching part identifies the top flamingo instead of the bottom one which is the one that created the feature.

The second weakness is because of the threshold enforced by our algorithm that two ROIs identify the same object if they overlap by more than 50% of the area. This restriction makes our localisation algorithm to be very sensitive to partial occlusion due to irregular shape of some of the objects, the ROI area may not properly match the shape of the object. In Figure 6.9a, the localisation algorithm missed the bottom object while it located both objects in Figure 6.9b. However, empirical evidence suggests that dropping the overlap below 50%, objects may be counted multiple times.

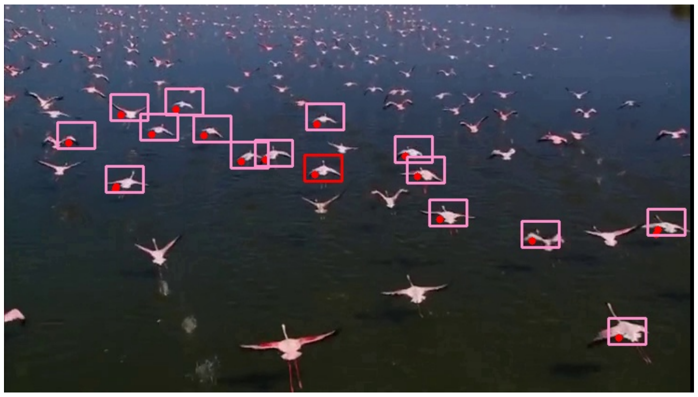
6.2.5 Complex Background

Our approach to locate objects relies only on the clusters we detected without any further processing to determine whether the matches are accurate. As such, any matched features from the complex background will be treated as any *actual* object features. Such outcomes can be seen in Figure 6.11 where some of the features in the clusters come from the background and as such have resulted in false positive locations for that cluster which led to the background feature localisations being included in the final frame object count estimation.

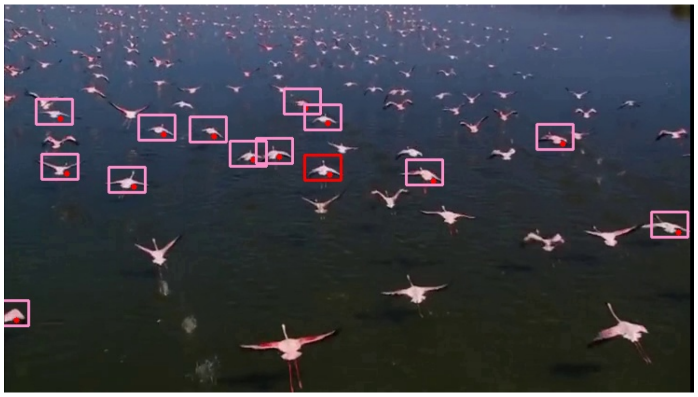
The presence of background features in a cluster does not necessarily mean that the cluster is noisy or that the clusters are invalid. However, experimental results show that the clusters often have low intra-cluster distance ratios. However, this cannot be conclusively used to filter out the false positive as the distance ratio can vary widely. As an example, in Figures 6.11a and 6.11b, the distance ratio was 0.153 and 0.017 respectively. In both cases, there were a lot of false positive locations due o presence of background features in the cluster. In comparison, Figure 6.11b had a comparable distance ratio of 0.172 but with more visibly similar features and very good localisation results.



(a) False positive locations created by not considering rotational invariance of SURF features.

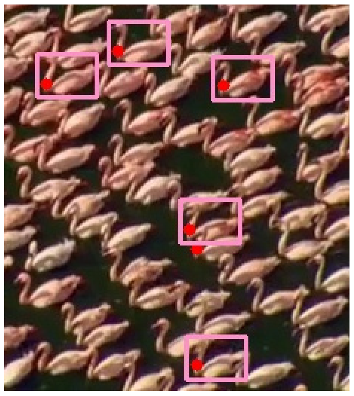


(b) Object locations detected from clusters with rotational variance consideration.

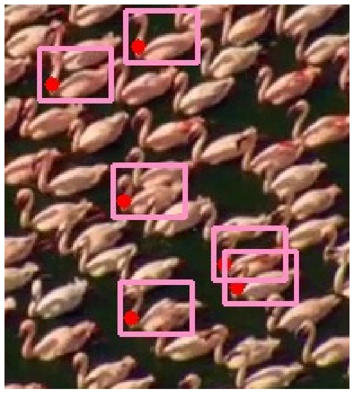


(c) Object locations detected from clusters with rotational variance consideration.

Figure 6.8: Comparison of object localisation with and without rotational consideration.

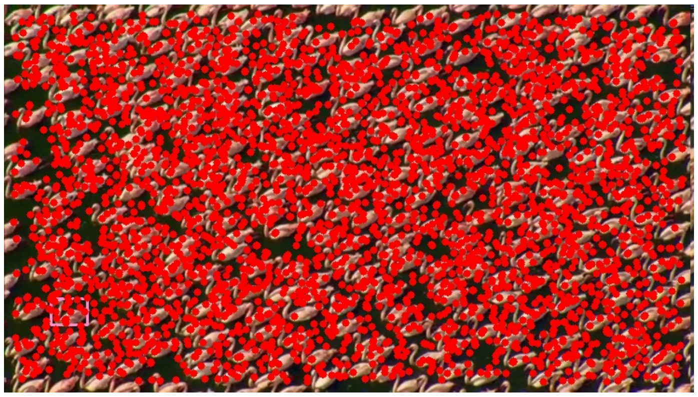


(a) Partial occlusion missed by the localisation algorithm in Frame 2.

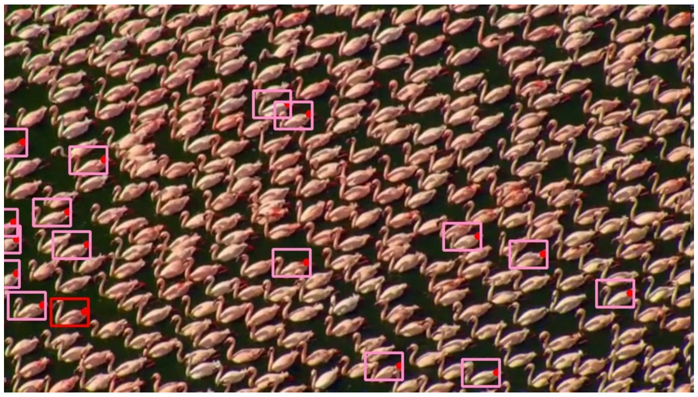


(b) Partial occlusion recognised by the localisation algorithm in Frame 7.

Figure 6.9: Susceptibility of the localisation algorithm to partial occlusion.

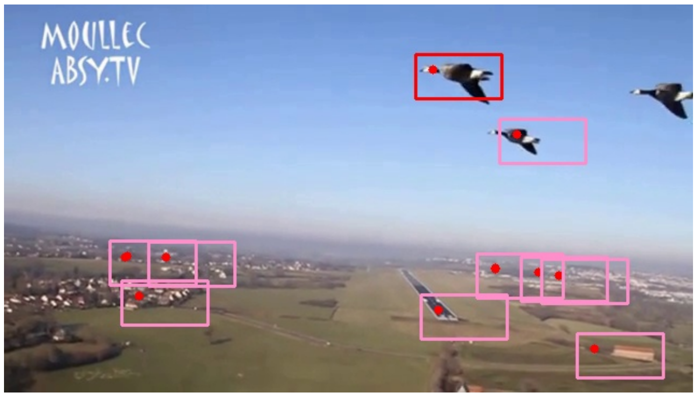


(a) All the SURF features detected in Frame 10 of *voc-18-bd-1*.

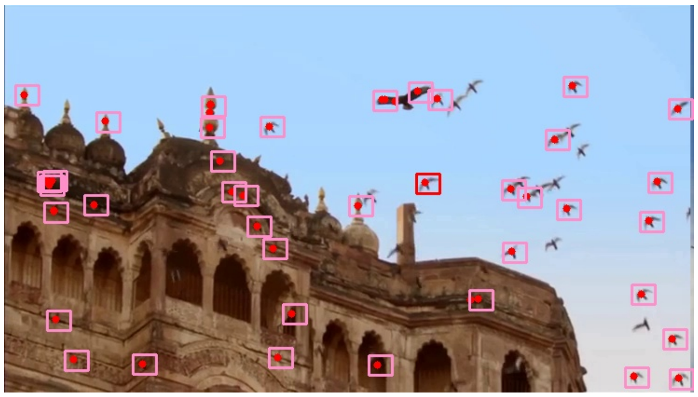


(b) Detection of partially visible objects.

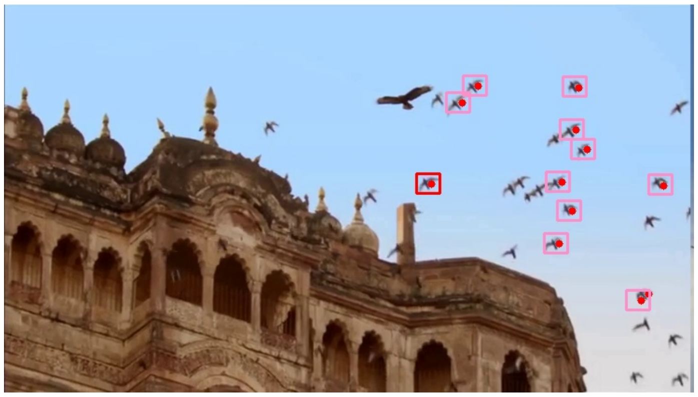
Figure 6.10: Susceptibility of the localisation algorithm to partial visibility.



(a) Background features in *voc-18-bd-6* affecting object localisations with intra-cluster distance ratio of 0.153.



(b) Background features in cluster 56 of *voc-18-bd-19* affecting object localisations with intra-cluster distance ratio of 0.017.

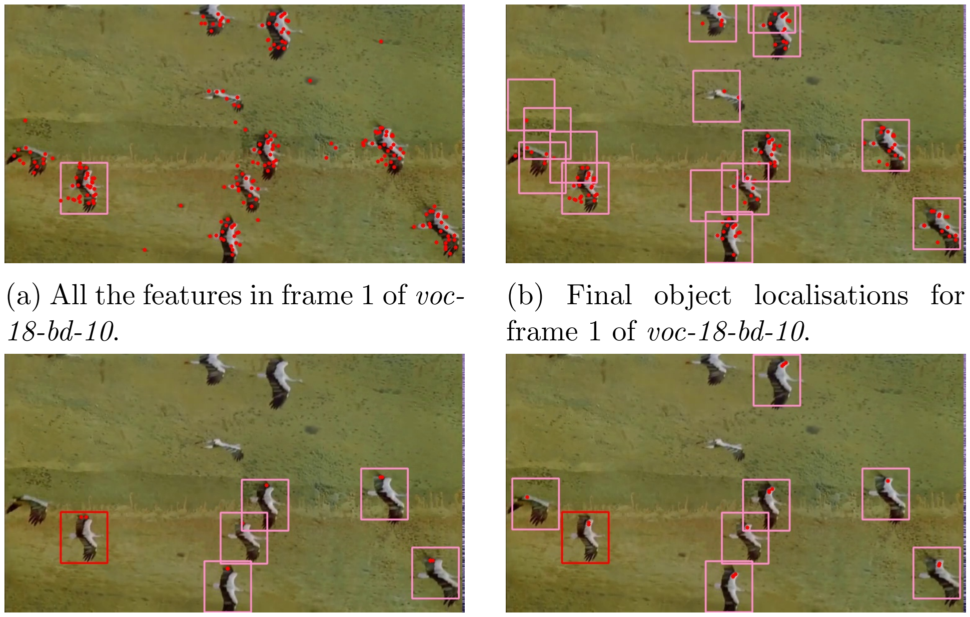


(c) Background features in cluster 78 *voc-18-bd-19* affecting object localisations with intra-cluster distance ratio of 0.172 which is comparable to Figure 6.11a.

Figure 6.11: False positives from background features.

6.3 Frame Count Estimation

We maintain a list of detected object instances for each intersecting cluster, $C_{\hat{j}}$, $R_{\hat{j}} = \{r^1_{\hat{j}}, \ldots, r^x_{\hat{j}}\}$, where r^x is the location of new object instances and x is the number of features in the cluster. For each new discovered location, r_l, we first check if it already exists in the list. We do this by checking if there is an ROI $r^x \in R_{\hat{j}}$ such that $r^l \cap r^x > 0.5$. The effect of this intersection limit can be seen in Figure 6.12d where cluster 37 had 16 features but the localisation algorithm found eight objects because some of the intersecting ROIs have been combined.



(a) All the features in frame 1 of *voc-18-bd-10*.

(b) Final object localisations for frame 1 of *voc-18-bd-10*.

(c) Localisation in cluster 12 in Frame 1 of *voc-18-bd-10*.

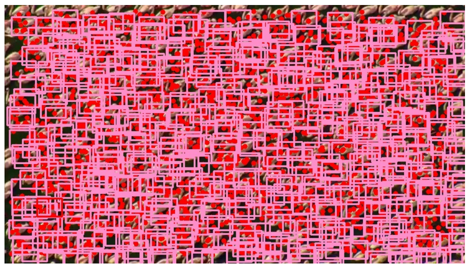
(d) Localisation in cluster 37 in Frame 1 of *voc-18-bd-10*.

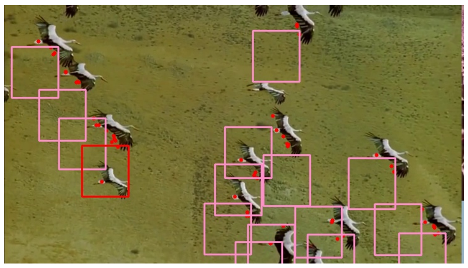
Figure 6.12: Final object localisations and multiple clusters identifying the same objects.

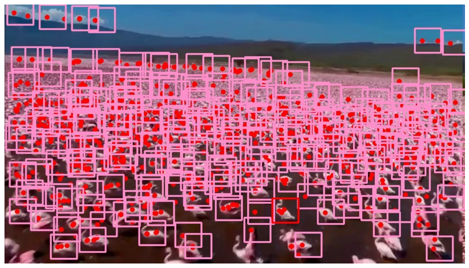
With the method described above, we obtain $\hat{R} = \{R_1, \ldots, R_j\}$ for each frame, where each $R_{\hat{j}} \in \hat{R}$ corresponds to locations detected from cluster $C_{\hat{j}} \in \hat{\mathcal{C}}$. We then combine all $R_{\hat{j}}$ to obtain $\mathcal{R} = \{\hat{r}_1, \ldots, \hat{r}_z\}$, where z is the number of objects detected within the frame. The combination of the locations from individual clusters uses the same approach to add a new location to cluster locations since features from different clusters may identify the same object instance.

In Figure 6.12, we show the results of our approach with frame 1 of *voc-18-bd-10* using varying *minPts* approach described in Section 4.2. Figure 6.12a shows all the SURF keypoints in frame 1, Figure 6.12b shows the final localisation results while 6.12c and 6.12d show two of the clusters in $\hat{\mathcal{C}}$. In this frame, $minPts = 3$ produced the best results with $validity = 2$. The final object count estimation was 15 against the ground truth of 10 due to some false positive detections which will be discussed in detail, along with some mitigation techniques, in the next sections and in Chapter 7.

Low validities, as discussed in Chapter 4, introduce a lot of noise features into non-noise clusters. This means the localisation algorithm then suffers from the noise in the clusters, resulting in a lot of false positive locations found. In Figure 6.13, we show some examples of these noisy clusters and how they lead to false positive locations. The approach to mitigating this problem will be discussed in Section 6.5.


(a) The effect of noise in cluster 3 of frame 3 from *voc-18-bd-1*.


(b) The effect of noise in cluster 4 of frame 54 from *voc-18-bd-10*.


(c) The effect of noise in cluster 32 of frame 1 from *voc-18-bd-18*.

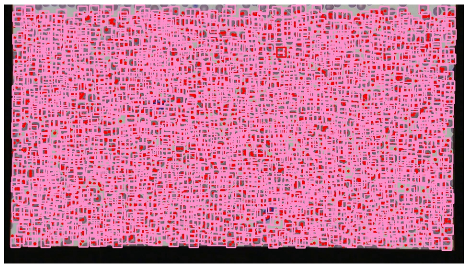

(d) The effect of noise in cluster 5 of frame 1 from *voc-18-bl-1*.

Figure 6.13: Example of noisy clusters in creating false positive locations. In these images, the noise can be noticed because some features are on the object while others are on the background where the colours are noticeably different.

6.4 Iterative Cluster Daisy-Chain

In Section 6.1, we showed how we learn the object of interest features using the ROI provided by the user. In 6.2, we showed how to filter out noise features from the ROI features and use the valid features to locate the other objects in the frame. However, this approach assumes that the single initial object instance provides all the possible features of the object of interest. Empirically, we found that some valid object feature clusters may not intersect with r which can lead to some object instances being missed by the localisation algorithm.

In Figure 6.14, cluster 127 intersected with the initial ROI and the object instances were properly identified. However, cluster 10 did not intersect with the initial ROI and as such was left out of the object count even though it clearly identifies some of the birds in the frame. In this example, the clusters were detected using $minPts = 3$ with no rotational variance. The problem of missing objects become more pronounced when cluster over-segmentation (See Section 4.3) is used. Using $minPts = 3$ for frame 1 shown above, only seven clusters were found to intersect the initial ROI giving 107 object count estimation while nine were found when using $minPts = 2$ resulting in 88 object identified. The ground truth for the frame was 273 objects.

In this thesis, we use cluster *daisy-chaining* to find the clusters that do not intersect with the initial ROI. Given $\mathcal{R}^0$ as the list of object instance locations detected from using clusters that intersected with the initial ROI, we iterate through the $\mathcal{R}^0$, and for each location, we find all clusters that intersect with the instance ROI that were not used in the original ROI. We then use the approach described in Subsection 6.2.1, to find the new object instances location $\mathcal{R}^1$. Since a cluster may intersect more than one ROI, we use the ROI with the best template match score. This iterative cluster daisy-chaining can be used as often as necessary to get satisfactory results. In this thesis, we use I to represent the number of iterations such that $\mathcal{R}^I$ is the final object locations after I iterations.

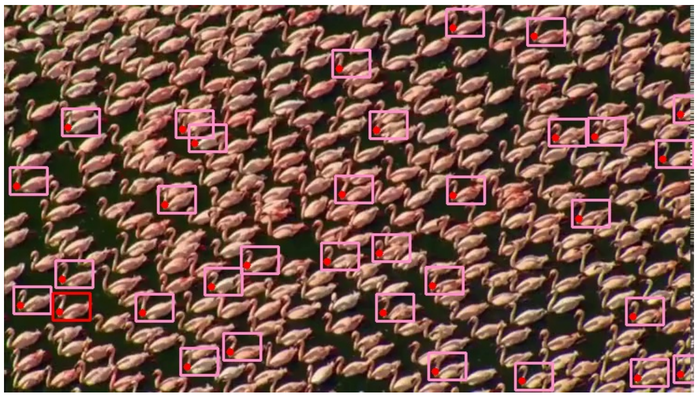
The results of one such cluster can be seen in Figure 6.14c where the new object locations were detected from cluster 10 intersecting one of the object locations detected by cluster 127 which was the one that intersected with the initial ROI. In this instance, the overall object count estimation from daisy chaining the clusters rose to 263 when using $minPts = 3$ and to 286 when using $minPts = 2$ with just one iteration of cluster daisy-chaining. The results of using various iteration values can be seen in Table 6.4. The table shows that there is a limit to the number of objects detected which is bounded by the number of clusters detected. It is worth noting that empirically, we found that HODVs tend to require cluster daisy-chaining the most while LODVs require it the least.

Table 6.4: Object count estimations using various iteration values for $minPts = 2$ and 3 on frame 1 of *voc-18-bd1*.

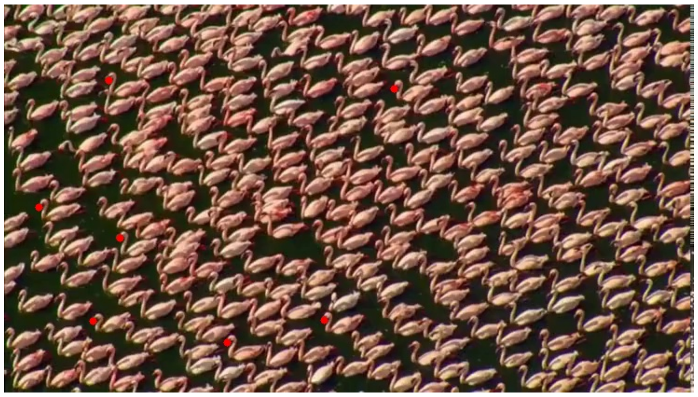
	$minPts = 2$ $clusters = 706$		$minPts = 3$ $clusters = 145$	
Iterations	**Selected Clusters**	**Estimation**	**Selected Clusters**	**Estimation**
0	9	88	7	107
1	380	286	113	260
2	654	329	141	281
3	665	331	143	283
4	665	331	143	283

In this thesis, we make an assumption that if a cluster intersects with an ROI, then it can be used to find other object locations without any further validations. This often works well when $I = 0$ but can introduce false positive locations when using iterative cluster daisy-chaining. In Table 6.4, we notice that for $minPts = 3$, we ended up selecting 143 clusters out of 145 clusters detected in the frame. The result is that some of the clusters do not properly represent the objects of interest or when the wrong location is selected as the basis for the intersecting clusters.

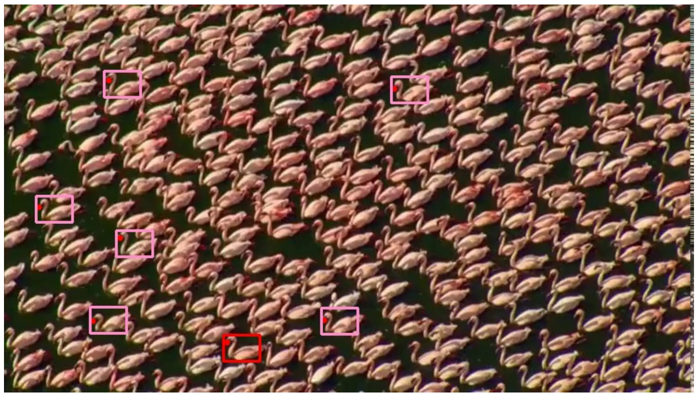
In Figure 6.15, we show the introduction of false positives due to daisy-chaining two clusters in frame 1 of *voc-18-bd-1* for $I = 3$. In Figure 6.15a, the failure is due to the features being on the surrounding of the objects. Such features are often influenced by the more than one object as well as the surrounding. The features also often have different scales which, as discussed in Subsection 6.2.2, leads to false positive locations. In Figure 6.15b, the daisy-chaining failure is due to the wrong ROI being selected as a template for cluster 220 which led the new object locations to be false-positives.



(a) Locations from cluster 127 in frame 1 of *voc-18-bd-1*.

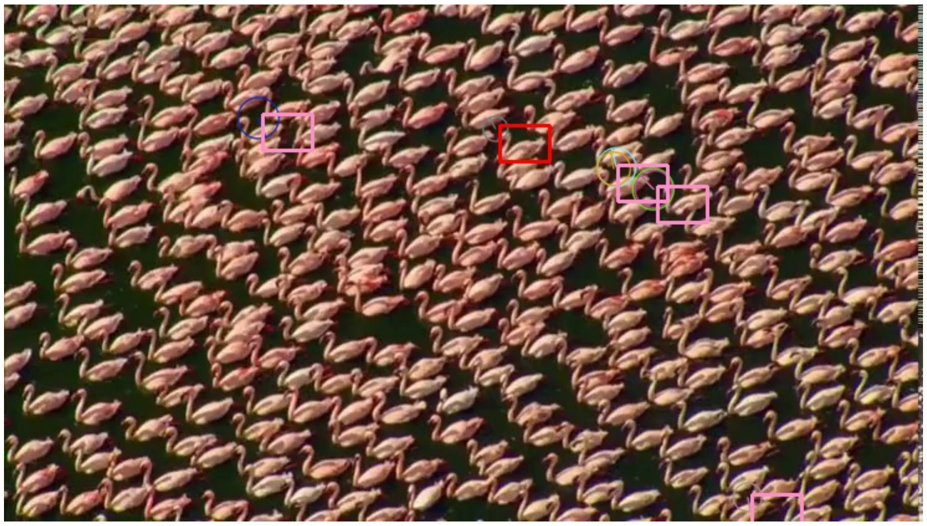


(b) An example of a good cluster that has been skipped because it does not intersect the initial ROI.



(c) New objects detected by using cluster daisy-chaining approach.

Figure 6.14: Iterative cluster daisy-chaining to discover clusters that do not intersect with the initial ROI in order to reduce false negatives.


(a) Daisy chaining failure due to off-object features.

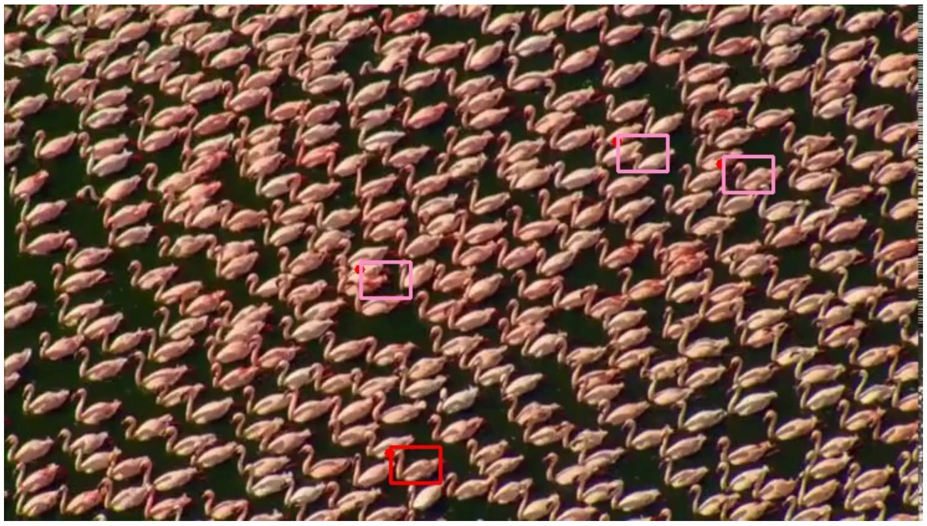

(b) Daisy-chain failure due to selection of wrong intersection ROI.

Figure 6.15: Iterative cluster daisy-chaining introducing false positives.

6.5 False Positive Locations Detection and Elimination

In this section, we outline ways of reducing the false positive object locations. At the centre of our approach is the colour model training and tracking described in Chapter 5. We discuss two ways in which we can limit the presence of false positive locations in the final object count estimation. In the first subsection, we approach this problem by detecting the clusters in the frame's colour descriptors. In the second subsection, we select the locations from using full frame descriptors which contain the colour model features.

6.5.1 Frame Object Count using Colour Model Clusters

The first approach to elimination of false positive locations is the detection of the clusters within the colour model descriptors. Given the colour model keypoints, $\mathcal{K}^{cm}$, we create a new descriptor set, $\mathcal{D}^{cm}$, from the descriptors corresponding to the keypoints. We then process the descriptors in the same way as explained in Chapter 4 and extract the location of the objects using using the process described in Sections 6.2 and 6.3.

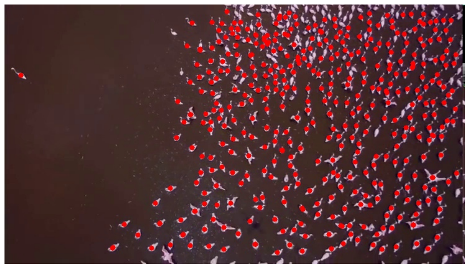
In videos where the colour model was successfully tracked, the number of features is much less than the frame features. The effect of this reduction is that descriptor clusters that do not belong to the colour model are removed as well as some of the noise clusters. This leads to a descriptor dataset that may not have well defined clusters which makes it difficult to detect the clusters. In terms of clustering results validities, the effect is lower validity values as well as low confidences for visually good clusters.

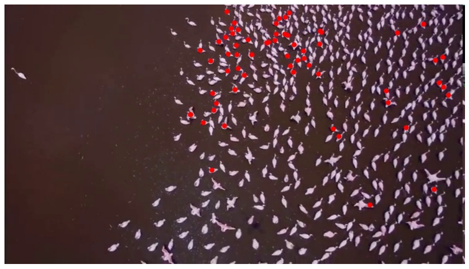
An extreme case of this is shown in Figure 6.16 and Table 6.5 for *voc-18-bd-12*. The best colour model was selected at $minPts = 15$, and the descriptors were processed using varying $minPts$ values (See section 4.2) with no cluster over-segmentation (See section 4.3) and no descriptor augmentation (See section 4.4). In frame 1, the total number of features was 1934 compared with 432 for the colour model and the results selected were for $minPts = 3$ with $validity = 0$ compared to the same $minPts$ and

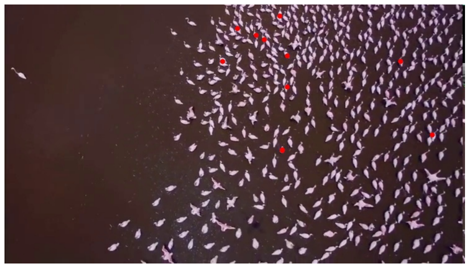
Table 6.5: The clusters detected

		Core Distances				**Intra-Cluster Distances**			
Cluster	**Number of Points**	**Min**	**Max**	**Ratio**	**Confidence**	**Min**	**Max**	**Ratio**	**Confidence**
0	62	0.278	0.392	0.711	54.77	0.261	0.889	0.293	74.16
3	359	0.089	0.277	0.322	0	0.06	0.788	0.076	0
2	11	0.21	0.272	0.77	58.26	0.038	0.423	0.089	14.66

validity = 4 in full frame descriptors. Only two non-noise clusters were detected for the colour model features compared with 64 for the frame descriptors.


(a) The colour model features.


(b) The colour model noise cluster.


(c) Cluster 2 in the colour model features.

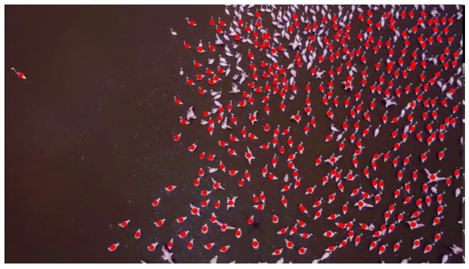

(d) Cluster 3 in the colour model features.

Figure 6.16: Detecting clusters in the colour model features in frame 1 of *voc-18-bd-12*.

A key note from such results is the visual similarity of the features in the clusters (even the noise cluster). This similarity and the nature of the clusters led to the non-noise cluster 3 being calculated to have 0% confidence using both core distances and intra-cluster distances. The low confidences are a consequence of the cluster being so big that it has the largest maximum core and intra-cluster distances, even larger than the noise cluster. Essentially, the larger the cluster, the higher the possibility of it occupying a larger potion of the descriptor space leading to a very low *min:max* ratio.

A comparable cluster when detecting clusters in the full frame descriptors was 132 shown in Figure 6.17c which had 147 features compared with 359 features of colour model cluster 2. The extra features from colour model cluster were either noise, or in other clusters when detecting when using all frame descriptors. In frame descriptor clustering, cluster 132 had core and intra-cluster distance ratios of 0.429 and 0.093 respectively while in colour model descriptors, cluster 3 had 0.322 and 0.076 core and intra-cluster distance ratios respectively. However, in full frame descriptors clusters, the noise cluster was much larger and had occupied a larger descriptor space. This led to core and intra-cluster confidences of 18% and 92% respectively.

We show comparable results from frame descriptors in 6.17. For clarity, we removed the non colour model features from the noise cluster (Figure 6.17a). There is a large number of features that were noise in the frame descriptors that have now been detected in a non-noise cluster (Figure 6.16d) while some are still detected as noise. Figure 6.17b shows cluster 89 from frame descriptor clusters which contains some of the colour model features that were detected in cluster 2 (Figure 6.16c) of the colour model descriptors.

The good thing about detecting clusters within the colour model features is that since we already have high confidence that the features represents our objects, or at the very least, areas similar in colour to our objects, we can assume that localisation will results in fewer false positives. A confirmation of this is shown in Figure 6.18 where the frame localisation results for the colour model results shown in Figure 6.16 and Table 6.5. In all the results, we used varying *minPts* with no angle augmentation and no over-segmentation.

In Figures 6.18a and 6.18b, we show the final localisation results for frame 1 with both frame descriptors and colour model descriptors cluster detections. In 6.18a, the objects were properly identified while in 6.18b, the clustering validity was 0 and the noisy clusters created a lot of locations where the bounding boxes were partially off the actual objects. The good thing about using the colour model is that even when the clusters are noisy, the localisation does not stray too far off the actual objects.

Further evidence of the usefulness of colour model in reducing false locations even when handling low validity clusters is shown in Figures 6.18d. The birds in *voc-18-bd-1* were large enough that they had a lot of features on them while in *voc-18-bd-12*, only one feature was on the objects. Therefore, the localisation with noisy clusters for *voc-18-bd-1* had skewed results while in *voc-18-bd-12*, the localisation was still accurate. In fact, we could have used the colour model features as it is without detecting clusters.

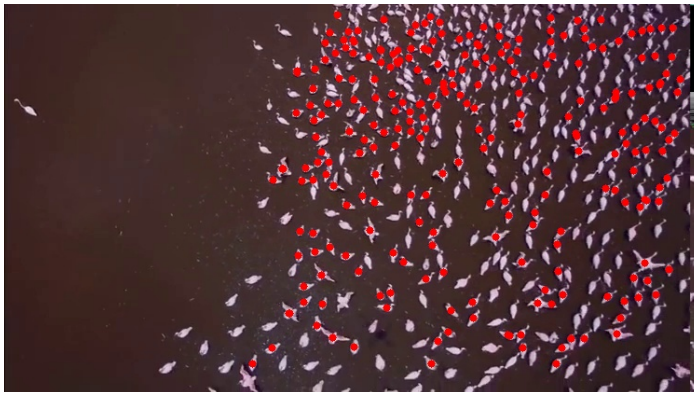
6.5.2 Frame Descriptor Locations Combined with Colour Model Locations

Given the colour model keypoints, $\mathcal{K}^{cm}$, the object locations from frame descriptors, $\mathcal{R}_d$, and the object locations from the colour model clusters, $\mathcal{R}_{cm}$, we combine locations from $\mathcal{R}_d$ and $\mathcal{R}_{cm}$ using Algorithm 5. If any of the locations in $\mathcal{R}_d$ have at least one colour model feature inside, we consider it to be a true object and append it to $\mathcal{R}_c$. All the locations detected from colour model descriptor clustering are considered to be true locations and as such, they are just added to $\mathcal{R}_c$. When adding locations to $\mathcal{R}_c$, we use the same approach described in Section 6.2 to detect duplicate locations.

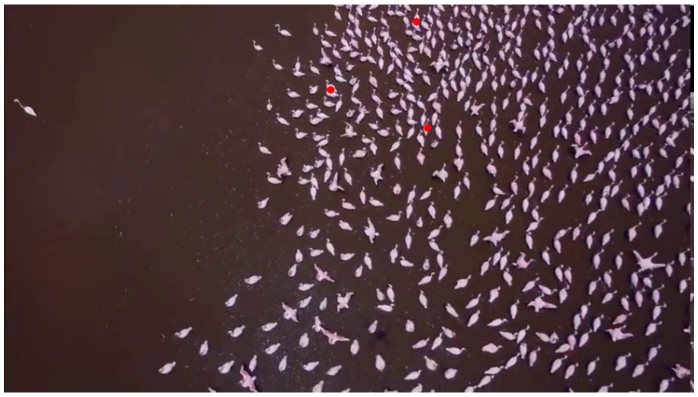
Algorithm 5: Combining the locations from frame descriptor clusters and colour model descriptor clusters.

```
Data: $\mathcal{K}^{cm}$ - Colour model keypoints
Data: $\mathcal{R}_{cm}$ - Object locations from colour model clusters
Data: $\mathcal{R}_d$ - Object locations from frame descriptor clusters
Result: $\mathcal{R}_c$ - Combined locations
for $r \in \mathcal{R}_d$ do
    for $k \in \mathcal{K}^{cm}$ do
        if $r$ contains $k$ then
            $append(\mathcal{R}_c, r)$
        end
    end
end
for $r \in \mathcal{R}_{cm}$ do
    $append(\mathcal{R}_c, r)$
end
```

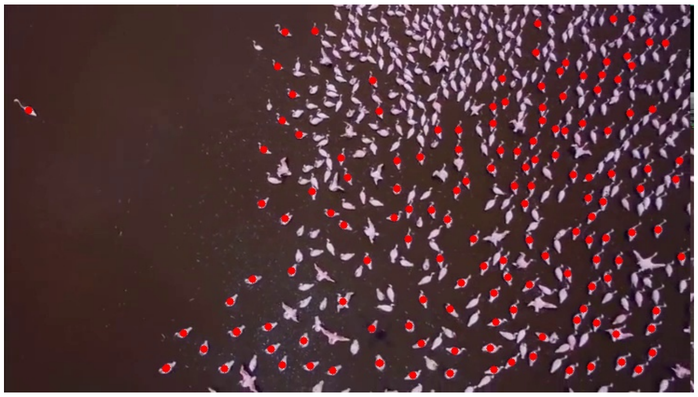
With this approach, false positive locations from frame descriptor clusters were detected and removed. The effect can be seen in Figure 6.19 where the frame descriptor clusters (Figure 6.19a) detected 22 birds while colour model clusters (Figure 6.19b) detected 11 birds. The combination of the two results detected 12 objects. In this case, the ground truth was 10 and the descriptors were augmented with angles, *minPts* was set to 2 and one iteration of cluster daisy-chaining was used.



(a) Cluster 0 from the full descriptor cluster with non colour model features removed.

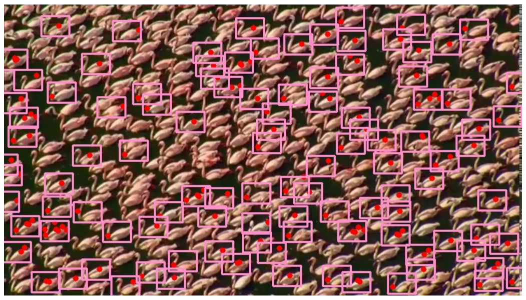


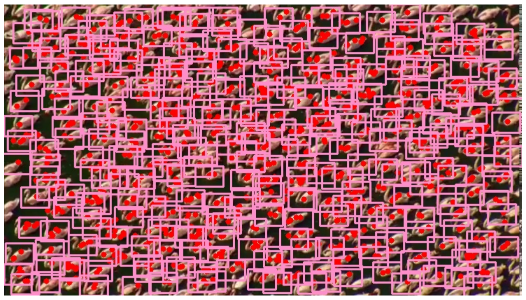
(b) Cluster 89 from frame 1 descriptors of *voc-18-bd-12* comparable with cluster 2 in Figure 6.16c.

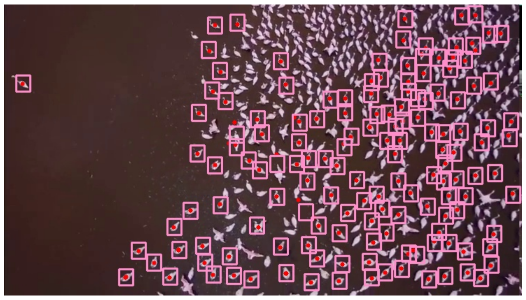


(c) Cluster 132 from frame 1 descriptors of *voc-18-bd-12* comparable with cluster 3 in Figure 6.16d.

Figure 6.17: The clusters from the frame descriptors.


(a) *voc-18-bd-1* frame 1 descriptor clusters localisation where *validity* = 4.


(b) *voc-18-bd-1* frame 1 colour model clusters localisation where *validity* = 0.


(c) *voc-18-bd-12* frame 1 descriptor clusters localisation where *validity* = 4.

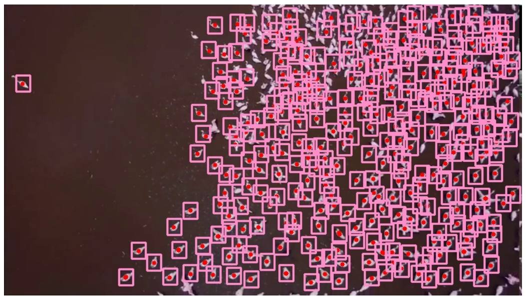
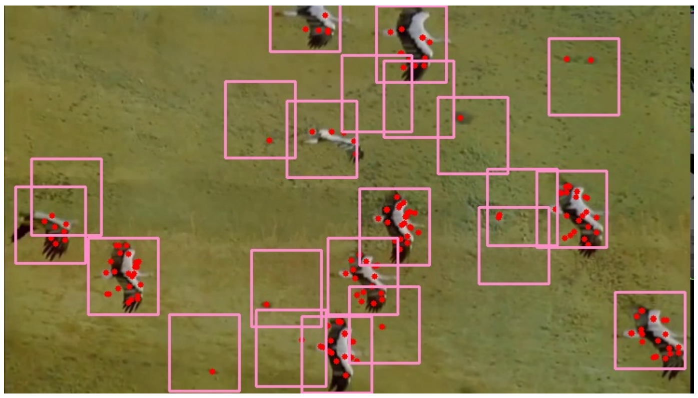
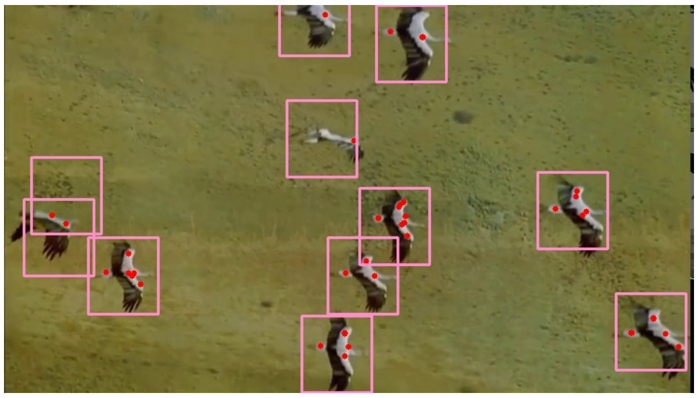

(d) *voc-18-bd-12* frame 1 colour model clusters localisation where *validity* = 0.

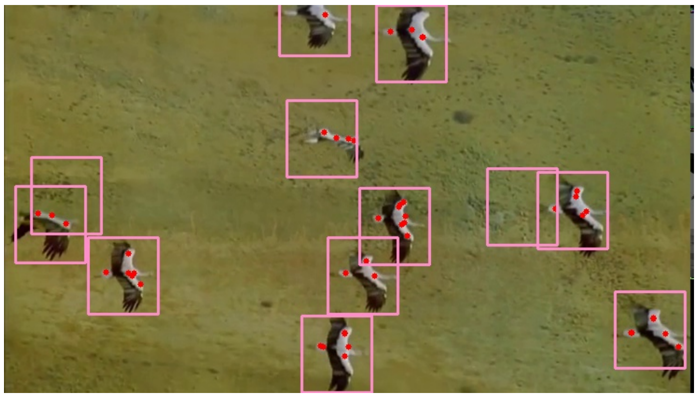
Figure 6.18: Comparison between frame descriptor localisation and colour model localisation. In this figure we used varying minPts and no rotation or cluster over-segmentation.



(a) Locations from frame descriptors with a lot of false positive locations.



(b) Precise locations detected with the colour mode.



(c) Combined results with much less false positives than descriptor $\mathcal{R}_d$.

Figure 6.19: Combining the locations detected using frame descriptors and colour model descriptors to get rid of the false positive locations.

6.6 Summary

In this chapter, we showed how we use the descriptor clusters presented in Chapter 4 to detect object of interest instances. We started by showing how we select the clusters that belong to the object of interest by using the user provided ROI and selecting non-noise clusters that intersect with it. We then used BOOSTING SOT to track the initial object of interest from frame to frame and repeat the cluster selection process. We showed how we handle the object drifting off the frame as well as the problems created by scale challenges in re-initialising the tracker.

In Section 6.2, we showed how we extract the object locations from the selected clusters. We explained the approach by discussing it in terms of the challenges our dataset posed. We explained why we do not use homography in an attempt to find the objects of interest even when only one feature per object has been detected. We noted that challenges such as scale cannot be solved by using only one feature because the scale at which a feature was detected may not reflect the size of the object. Rotational variance was handled by augmenting the descriptors with the feature angles which proved effective in countering false positive locations due to feature rotations. Occlusion was identified as one challenge where our occlusion detection approach worked in some cases and not in others. A possible improvement could be to increase the intersection of the ROIs which also risks counting the same object multiple times.

We combined the cluster locations in Section 6.3 to show how we extract the object count estimation for each frame. We showed the effect of noisy clusters in introducing false positive location into the final count. The noisy clusters have been shown to have very low intra-cluster $min : max$ distance ratio. In future improvements, this attribute can be used to threshold the cluster selection process at the risk of missing the good features within the noisy clusters. The plus side of this noisy cluster problem is that is affects mostly LODVs which do not really need a computer based counting approach since humans can count easily at those low numbers.

We identified that for HODVs, counting using only clusters that intersect with the initial ROI is often not enough to find all the objects in the frame. We used cluster daisy-chaining to find the clusters that identify the objects of interest but do not intersect the initial ROI. These clusters were found to intersect one or more of the objects located through their cluster intersecting with the initial ROI. We showed that while this approach improves the object count estimation, it also introduces false positives object locations. We also showed that this approach only works if the objects have multiple features that form non-noise clusters. During iterative daisy-chaining, we select the ROI to use by finding the best matching ROI based on the calculated matching score using moments.

Finally, the problem of false positives is addressed by using the colour model explained in Chapter 5. We use the colour model first by detecting the clusters within the colour model descriptors an then extract the object count estimation the same way as in Section 6.3. We also use the colour model to filter out the false positive locations. The filtered object locations are also combined with the locations detected using only the colour model. Finally the object count is also estimated by detecting the clusters in the frame descriptors but then only using the clusters whose features are also in the colour model. We showed that this use of colour model only works for videos where the colour model tracking was successful.

In the next chapter, we set out to show the overall counting estimations for the videos in our dataset. We show the results using different parameters and explain the failures and successes for the videos. We explain the failures of our approach for the some of the videos and explain the reasons for such failures. We also make a comparison between our count estimation approach and the current object counting literature.

Chapter 7

Video Object Count Estimations

In the previous chapter, we demonstrated how we use the descriptor clusters to locate the object of interest in a video frame based on the initial ROI provided by the user. We showed how we combine the detected object locations from each cluster to get the locations of the objects within a frame. We also presented a way in which we can detect and eliminate the false positive locations using the colour model that was described in Chapter 5. In this chapter we analyse how our object counting approach performed against our dataset from Chapter 2 and make comparisons to other approaches discussed in Chapter 1.

We show the best results from each of the three object count estimations methods described in Chapter 6; frame descriptor clustering, colour model descriptor clustering and a combination of the two. For each of the methods we use for counting, there are four parameters that can control the final count estimation. These are shown in the column "Parameters" of the tables in this chapter. The format of the parameters is *minPts, I, O, R*, where *minPts* is the value of *minPts* used for HDBSCAN, I shows the number of iterations for cluster daisy chaining, O denotes whether cluster over-segmentation was allowed (See Section 4.3) and R denotes augmentation of descriptors using feature angles (See Section 4.4). The *minPts* and I parameters are numbers while O and R are boolean values. For example, if we used $minPts = 3$, with one iteration of cluster daisy chaining, no cluster over-segmentation and augmented descriptors, the “Parameters” column will have $3, 1, N, Y$. It should be noted that for $minPts \geq 3$, we use the technique described in 4.2 to find the best clustering results. If we used $minPts = 2$, then over-segmentation is enabled hence the parameter ‘O’ will always be set to ‘Y’.

We also show the average time per frame taken to achieve the results in the tables as well as the *min, max*, *mean* (μ) and *standard deviation* (σ). The value of σ is used first to show how stable the estimations were from frame-to-frame and secondly as an uncertainty measurement such that the other values in the table are $min \pm \sigma$, $max \pm \sigma$ and $\mu \pm \sigma$. Since we were testing our approach on videos that have ground truths, we selected the counting estimations that were as close as possible to the ground truth values. We first look for the closest σ and μ and then look for the best *min* and *max* such that there is an overlap between the values in our estimation to the corresponding values from the ground truth taking into consideration the uncertainty represented by the σ value. The sample results for each of our videos are shown in Appendix B.

7.1 Video Count Estimations

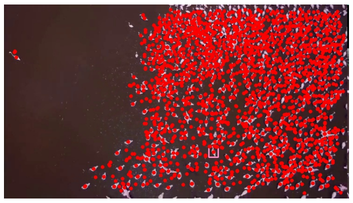
In this section, we present the results of estimating the object count in the our video dataset. The main restriction on our methods is that the SOT as discussed in Section 6.1, should consistently provide a valid initial object. As such, the results shown in this section are for videos for which this restriction held. As discovered during the experiments, in some videos such as *voc-18-bd-4* and *voc-18-bl-3*, the SOT had trouble tracking the initial objects. However, we do include their results for those frames where the initial object was successfully tracked. For *voc-18-bd-4*, that means the first 40 frames and for *voc-18-bl-3*, the first 26 frames.

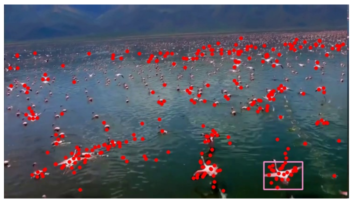
Most notably, our approach could not reliably estimate the number of objects in *voc-18-bd-14* and *voc-18-bd-16*. These two videos have all the five challenges in our dataset as shown in Table 2.4. As we explained in Section 6.2, our counting approach had the most difficulty with scaled objects. Since scale may be gradual like in *voc-18-bd-14* (See Figure 6.5), our algorithm had varying successes based on how extreme the scale differences are between the initial object and other objects in the frame. Another challenge with scale is intra-frame scale changes of the initial object. In this thesis, we used BOOSTING tracker to find the initial object in each frame. The tracker does not compensate well for object size changes from frame to frame, as such, videos like *voc-18-bd-7* end up with a tracker bounding box that is much larger than the object it is tracking.

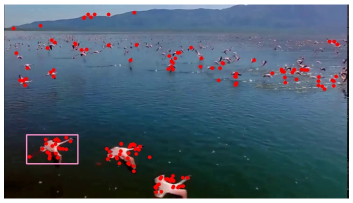
In fact the weakness of our approach in handling scale is so profound that the results in Tables 7.1 - 7.3 highlight this as well. The videos that have scale challenges have the widest differences from the ground truth, most notable in the *min, max* and σ values even when the μ value is close to the ground truth value. This wide differences are due, in large part, to the localisation algorithm's inability to compensate for size. We find that when the initial object is larger than the detected object instance, the bounding box may end up covering up more than one object. When the initial object is smaller than the detected object instance, the localisation algorithm can then provide more than one bounding box for different parts of the same object.

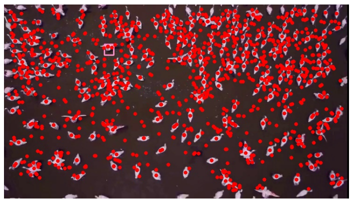
At its core, our approach relies on successful detection of SURF features on the objects of interest. Scale challenge creates a situation where the objects further from the camera may be too small for detection of SURF features which is the case in some of our videos. The restriction from SURF features does not only come into play when there is scale problems, but also when the object are just far from the camera. In *voc-18-bd-12* (Figures 7.1a and 7.1d), there is no scale challenge, but the birds are so far from the camera that some of the birds have no features especially in the first few frames. However, in this video, the camera does come close as the video progresses which leads to the all birds having features on them. For both *voc-18-bd-14* (Figures 7.1b and 7.1e) and *voc-18-bd-16* (Figures 7.1c and 7.1f) a majority of the birds did not have SURF features on them.

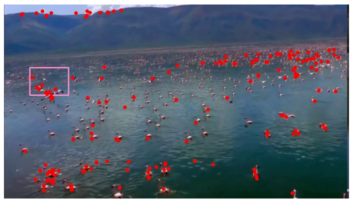
Selection of the best object count estimations for our approach relies on multiple test on each video. We first set a *minPts* value, angle augmentation (R) and cluster over-segmentation (O). We then run for various iterations (I) starting at 0 and increasing until we start getting the values that do not vary too much between iterations. We then change one of the *minPts*, R and O parameters and repeat the process. In


(a) All the features in frame 1 of *voc-18-bd-12*.


(b) All the features in frame 1 of *voc-18-bd-14*.


(c) All the features in frame 1 of *voc-18-bd-16*.


(d) All the features on the last frame of *voc-18-bd-12*.


(e) All the features on the last frame of *voc-18-bd-14*.

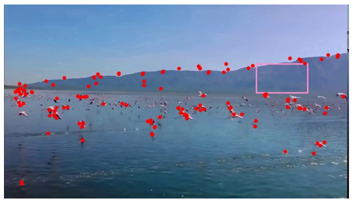

(f) All the features on the last frame of *voc-18-bd-16*.

Figure 7.1: The presence of SURF features dictate how many objects our approach can detect. In *voc-18-bd-14* and *voc-18-bd-16*, the tracker even lost the initial object due to localisation failure.

videos where there is some symmetry in the features such as *voc-18-bd-20* and the blood videos, parameter R provided the most stable results in both localisation and count estimations.

7.1.1 Frame Descriptors

In Table 7.1, we show that all LODVs required the use of $minPts = 2$ and $I = 0$ to obtain stable results with frame descriptor clustering. This is because at low object count, our videos had large objects with enough features that a one-to-one matching provided enough matches to locate all the objects in the frame. Therefore, we do not need to use iterative daisy chaining to locate all the other objects. The need for augmentation of descriptors with feature angles was dependent on the objects shapes. The same $I = 0$ is also evident for MODV (See Table 7.2) where $minPts$ value varied from video to video. However, since we have results from only three of the four MODVs in our dataset, we cannot draw definite conclusion on the use of particular parameters.

Table 7.1: LODV counting estimations using the frame descriptors.

Video	**Time Per Frame (s)**	**Parameters**	**Min**	**Max**	μ	σ
voc-18-bd-4	0.014	2,0,Y,N	1	11	5.79	2.55
voc-18-bd-5	0.023	2,0,Y,Y	2	11	6.84	3.23
voc-18-bd-6	0.022	2,0,Y,Y	3	12	6.37	2.18
voc-18-bd-7	0.024	2,0,Y,Y	3	14	8.11	2.36
voc-18-bd-8	0.027	2,0,Y,N	4	24	12.25	4.1
voc-18-bd-9	0.027	2,0,Y,Y	4	18	8.83	2.77
voc-18-bd-11	0.069	2,0,Y,N	3	23	12.43	3.95
voc-18-bd-20	0.032	2,0,Y,Y	3	9	5.29	1.31

Table 7.2: MODV counting estimations using the frame descriptors.

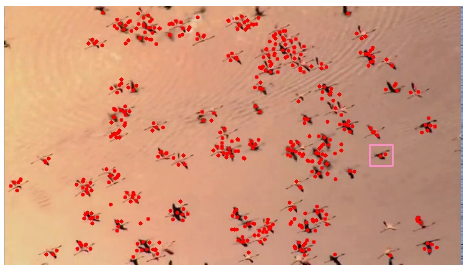
Video	**Time Per Frame (s)**	**Parameters**	**Min**	**Max**	μ	σ
voc-18-bd-2	0.026	2,1,Y,Y	2	29	12.11	5.61
voc-18-bd-10	0.055	2,0,Y,N	7	26	14.44	3.26
voc-18-bd-19	0.086	3,0,Y,Y	0	54	22.3	8.58

In Table 7.3, we show that in videos where objects are present in high density, there is need to use iterative cluster daisy-chaining to achieve higher and stable accuracy results. In our dataset, we required $I = 1$ whenever the stable results were obtained with $minPts = 3$ but had to use higher iteration values whenever $minPts = 2$ was used, except in the case of *voc-18-bd-18*. The need for iterative cluster daisy-chaining in HODVs for $minPts = 2$ is explained in Sections 4.3 and 6.4.

In some of the HODVs shown in the table, the minimum object count detected was 0. This comes from frames where the localisation algorithm failed to detect any clusters that intersect the initial ROI. In *voc-18-bd-3* frame 38, 114 non noise clusters were detected, however, the two features inside the initial ROI (Figure 7.2a) were detected as noise (See Figure 7.2b). With such outcome, there is no amount of iterative cluster daisy-chaining that can detect the other objects in the frame.

Table 7.3: HODV counting estimations using the frame descriptors.

Video	**Time Per Frame (s)**	**Parameters**	**Min**	**Max**	μ	σ
voc-18-bd-1	1.716	3,1,Y,N	259	408	291.65	27.97
voc-18-bd-3	0.094	2,4,Y,Y	0	123	92.3	18.98
voc-18-bd-12	0.474	3,1,Y,Y	93	459	325.45	80.85
voc-18-bd-13	0.03	3,1,Y,Y	0	110	31.55	18.32
voc-18-bd-15	0.035	2,3,Y,Y	0	76	48.89	13.13
voc-18-bd-17	0.047	3,1,N,Y	13	96	51.91	14.15
voc-18-bd-18	0.41	2,1,Y,Y	7	166	77.71	30.64
voc-18-bl-1	4.34	2,5,Y,Y	0	850	448.27	316.28
voc-18-bl-2	5.354	2,2,Y,Y	250	504	394.45	5.354
voc-18-bl-3	1.461	2,2,Y,Y	100	300	199.12	39.7
voc-18-bl-4	0.884	2,2,Y,Y	109	168	141.41	12.22


(a) All the features in frame 38 of *voc-18-bd-3*.

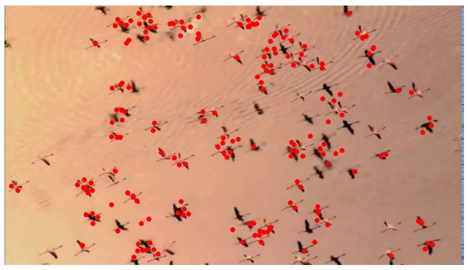

(b) The two features inside the ROI being part of the noise cluster.

Figure 7.2: An example of how our approach can fail to detect any objects due to ROI features being labelled as noise.

7.1.2 Colour Model Descriptors

In Chapter 5, we described an MOT for tracking colours of interest in a video. In Section 6.5, we showed how we can use a frame's colour model to reduce false positive locations by detecting clusters in the colour model descriptors. In this subsection we show the results of using this approach to estimate the object count in our dataset. This approach relies on the MOT's ability to learn and track the object of interest's colour model successfully throughout the video.

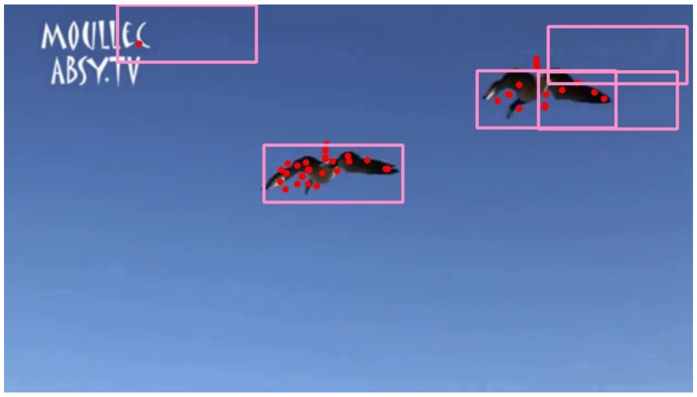
In 5.2.1, we showed that two of the videos in our dataset, *voc-18-bd-18* and *voc-18-bl-1*, our MOT failed to learn the colour model and as such we do not show the results those videos in this subsection. In Table 5.2, we showed that in some videos, the tracker lost the colour model before the end of the video. This means that we were selective about on which videos we used the colour model for count estimations. We selected the videos on which the colour model features were on all the objects for the majority of the video frames. We also used the videos for which the colour model was not lost before halfway through the video. We show this by including the "Num. of Frames" column in the results tables.

By the criteria above, only two videos from the LODV and MODV sets were used for colour model count estimations (See Tables 7.4 and 7.6). From the HODV set, we used three videos as shown in Table 7.7. The HODV video *voc-18-bd-3* was excluded because even though the colour model was successfully tracked for all the frames, it did not cover all the birds while *voc-18-bd-12* was included because for most of the frames in the video, the colour model was detected on all of them.

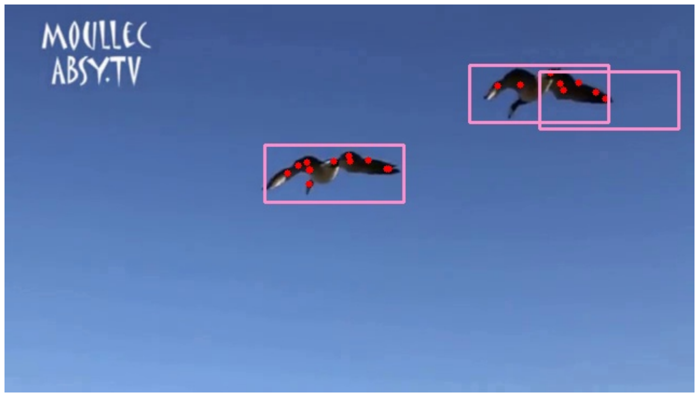
Empirical evidence shows that the colour model descriptor clustering method are more accurate with LODVs and MODVs. The removal of surrounding features changes the demography of the dataset, which can result in a different set of clusters from the full frame descriptor and also reduces the chance that a surrounding feature cluster could intersect the initial ROI and cause false positives. As such, the number of selected features and selected clusters from colour model descriptors is often less than from frame descriptors. An example can be seen in Table 7.1, where *voc-18-bd-5* had higher values than in Table 7.4. In that video's frame 6, 14 clusters were used to obtain count estimations of 5 for frame descriptors while 6 clusters were used to obtain 3 objects with colour model descriptors against a ground truth of 2 as shown in Figure 7.3.

Table 7.4: LODV counting estimations using the colour model descriptors.

Video	**Num of Frames**	**Time Per Frame (s)**	**Parameters**	**Min**	**Max**	μ	σ
voc-18-bd-5	56	0.01	2,0,Y,Y	2	8	3.71	1.34
voc-18-bd-20	76	0.0117	2,0,Y,Y	0	8	3.5	1.63



(a) Detected object locations in frame 6 of *voc-18-bd-5* using frame descriptors.



(b) Detected object locations in frame 6 of *voc-18-bd-5* using colour model descriptors.

Figure 7.3: Difference between colour model and frame descriptor clustering results. In both results, augmented descriptor clustering was done with $minPts = 2$ and no cluster daisy chaining.

The accuracy of colour model descriptor clustering can be seen in Table 7.5. The results in the table show the mean of the count estimations that is very close to the ground truth mean. In fact, taking into account the standard deviation as uncertainty in the truth count, the mean count estimations and their standard deviations are all within acceptable range. In the *min* column, the first two rows have values much lower that the ground truth and in the *max*, the other rows had values much higher than the ground truth. However, a close look at the frame by frame count estimation showed this low and high values to be on a few number of frames. The low values in the first 2 rows were improved by using a single iterative cluster daisy-chaining which also resulted in high *max* values. In both *2,1,Y,N* and *2,1,Y,Y*, the mean improve to be much closer to the ground truth while the standard deviations increased with *2,1,Y,Y* having lower standard deviation than *2,1,Y,N*.

From the results in Table 7.5, we selected *2,1,Y,Y* as the best parameters to obtain the best count estimations for *voc-18-bd-10* as shown in Table 7.6. It is should be noted that while the birds had two colours (black and white), we selected white as the colour model because we noted during experiments that if we selected black as well, we ended up with more false positives because some of the background features were selected as well. With cluster over-segmentations and descriptor augmentation activated, we could account for rotation while making sure than we only got valid clustering results.

The results for using the colour model to estimate the object numbers were much more unreliable in HODVs. Removing the ability to identify object from surrounding features proved very critical in HODVs. As a results, the colour model results showed a lot of low *min* values and high standard deviations values as shown in Table 7.7. In *voc-18-bd-1* and *voc-18-bd-12*, the *min* is 0 compared with the ground truth of 258 and 211 respectively. While the standard deviation for LODV and MODV decreased for the same parameters when using colour model descriptors, it increased for HODVs. This shows that the inconsistency of the count estimations between frames increased.

Table 7.5: Colour model clustering estimation results for *voc-18-bd-10* whose ground truth is $min = 9, max = 18, \mu = 13.82, \sigma = 2.96$ showing the close similarities between the results for various parameters.

Parameters	**Min**	**Max**	μ	σ
2,0,Y,N	3	18	9.93	3.16
2,0,Y,Y	3	19	10.68	3.26
2,1,Y,N	6	30	13.3	4.62
2,1,Y,Y	6	25	13.54	3.81
3,0,N,N	5	24	11.33	3.42
3,0,Y,N	6	25	13.11	4.26
3,0,N,Y	5	28	13.74	4.31
3,0,Y,Y	6	28	13.59	4.01

Table 7.6: MODV counting estimations using the colour model descriptors.

Video	**Num of Frames**	**Time Per Frame (s)**	**Parameters**	**Min**	**Max**	μ	σ
voc-18-bd-2	51	0.008	2,1,Y,Y	0	19	5.62	4.57
voc-18-bd-10	121	0.039	3,0,Y,Y	6	28	13.59	4.01

7.1.3 Combining Frame Descriptors Estimations with Colour Model Descriptor Estimations

In this subsection, we use the videos where the colour model was successfully tracked and count estimations extracted from the descriptors as described above. The process of combining the object locations from the colour model and frame descriptors has been explained in detail in Subsection 6.5.2. For each of the selected videos, we follow

Table 7.7: HODV counting estimations using the colour model descriptors.

Video	**Num of Frames**	**Time Per Frame (s)**	**Parameters**	**Min**	**Max**	μ	σ
voc-18-bd-1	77	0.383	2,4,Y,N	0	269	228.94	31.56
voc-18-bd-12	73	0.127	3,1,N,N	0	374	229.56	116.01
voc-18-bl-2	94	1.313	2,3,Y,Y	334	437	390.99	21.78

the same process of selecting the best results as explained above. In terms of time complexity, the combined object count estimation does not use HDBSCAN or detect clusters. Instead, it relies on the locations detected from the colour model and frame descriptors and as such the amount of time has to include times for both as well as the time it takes to combine them. In this thesis, we used the same parameters for frame and colour model descriptors for combination count estimations.

Generally, the effect of combining the two approaches this way is that the two previous approaches complement each other. Frame descriptor clustering results also used the features in the surrounding of the objects to locate them while colour model only use the features within the objects of interest but is prone to false positives especially when iterative cluster-daisy chaining is used. Colour model descriptor clustering tends to provide locations that are more accurately on the object and have low false positive locations. In fact, even the localisations that are not perfectly on the objects, they still for the most part include a big part of the objects.

The first noticeable improvement to combining the two leads to a higher *min* value which moves closer to the ground truth *min*. While on their own, frame and colour model descriptors may at one point or another detect very few objects as shown in Tables 7.1 - 7.7. The evidence of this improvement is shown in Tables 7.8 - 7.10. The other improvement is that for the same parameters, the frame to frame results become more consistent and as such, the mean and the standard deviation value tend to move towards the ground truth values signalling a more stable frame to frame count estimation.

Table 7.8: LODVs counting estimations using the combined frame and colour model descriptor locations.

Video	**Num of Frames**	**Time Per Frame (s)**	**Parameters**	**Min**	**Max**	μ	σ
voc-18-bd-5	56	0.035	2,0,Y,Y	2	9	4.71	1.89
voc-18-bd-20	76	0.046	2,0,Y,Y	3	7	4.43	1.19

Table 7.9: MODVs counting estimations using the combined frame and colour model descriptor locations.

Video	**Num of Frames**	**Time Per Frame (s)**	**Parameters**	**Min**	**Max**	μ	σ
voc-18-bd-2	51	0.065	2,1,Y,Y	4	22	11.67	4.36
voc-18-bd-10	121	0.08	2,0,Y,Y	7	20	13.29	2.77

Table 7.10: HODVs counting estimations using the combined frame and colour model descriptor locations.

Video	**Num of Frames**	**Time Per Frame (s)**	**Parameters**	**Min**	**Max**	μ	σ
voc-18-bd-1	77	1.499	3,1,Y,N	236	310	267.95	15.54
voc-18-bd-12	73	0.604	3,1,N,N	131	390	293.14	66.31
voc-18-bl-2	94	6.685	2,3,Y,Y	312	490	408.79	40.05

7.1.4 VOC-18 Dataset Best Results

In Tables 7.11 - 7.13, we show the best count estimation results for all the videos in our dataset. We show each video's ground truth, the method and the parameters used to find the best count estimations. In videos where the colour model tracking was unsuccessful, the best results are the same results shown in Tables 7.1 - 7.3. In videos where the colour model was successfully tracked, the results from combined frame and colour model descriptor locations provide the best estimations. Even though we selected the best results from colour model for *voc-18-bd-5*, the results from Table 7.10 show that for the same parameters, the results are still very close. We determined that the main reason colour model results are not the best for the other videos is because removing the all the other features leaves a descriptor space with fewer clusters and features that are most likely to be similar and hence making it difficult to find clusters. The observed results often have higher standard deviations that the other two methods which shows the lack of consistency in the frame to frame results.

The most important parameters for our approach proved to be the feature angle descriptors augmentation and *minPts*. From our experiment, we observed that if the objects of interest have any form of symmetry, then augmentation would mostly give the best results. In Tables 7.11 - 7.13, it can be seen that of the 24 videos in our dataset, only 4 have best results where no augmentation was used. The objects in those video showed little or no symmetry in their appearance. The most important factor to consider when selecting the value of *minPts* is the size of the objects followed by the number of objects in the video.

In our results LODVs all required $minPts = 2$ while the MODVs and HODVs required different values depending mostly on the size of the objects. In Tables 7.12 and 7.13, all videos except for *voc-18-bd-2* where $minPts = 3$ was used for best results had small objects. The use of $minPts = 2$ for other videos results in over-segmented clusters which have high similarities as discussed in Section 4.3. However, since the objects have multiple features directly influenced by their presence, iterative cluster daisy chaining compensates for the over-segmentation allowing us to find the objects whose clusters do not intersect the initial ROI.

Our decisions on which is the best counting estimation results in this thesis were essentially guided by the ground truths from the dataset. However, our approach was designed with the knowledge that it will not be always possible to know the ground truth, especially for HODVs. Experimentally, we observed that the best approach is to gradually increase the iterations for cluster daisy-chaining. The effect of this is that as more iterations are used, the min, max, μ and σ values become more stable as can be shown in Figure 7.4 where we show the results when using frame descriptors for *voc-18-bd-1* and *voc-18-bl-3*. In the figure, as iterations increase, there is less difference between the values in subsequent iterations. We argue that even though the values may be higher than the ground truth (due to false positives), this stabilisation gives a very good estimation of the number of objects. Even though *voc-18-bd-1* shown in had a successfully tracked colour model, we show the results for using frame descriptors in Figure 7.4a to highlight that our approach still works well without the colour model.

We note though, that this approach of increasing iterations to find the best results does not work well for LODVs and and MODVs especially when the colour model tracking was not successful. This is because at low and medium object densities, the presence of false positives has more effect on the final count values. For LODVs, we noticed that using $minPts = 2$ and $I = 0$ almost always gives good results as can be seen in Table 7.11. MODVs often require iterative cluster daisy chaining when $minPts = 2$ but none when using higher values in order to compensate for over-segmented clusters.

Table 7.11: LODV best counting estimations.

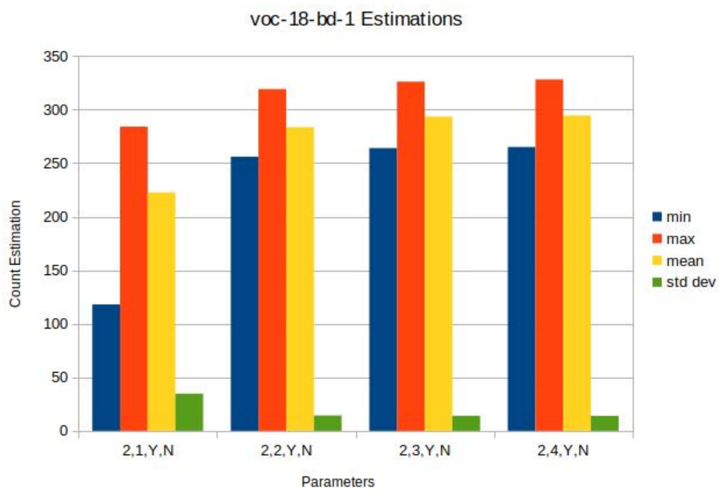
	Ground Truth				**Best Count Estimations**					
Video	**Min**	**Max**	μ	σ	**Method**	**Parameters**	**Min**	**Max**	μ	σ
voc-18-bd-4	4	6	5.31	2.55	Frame	2,0,Y,N	1	11	5.79	2.55
voc-18-bd-5	2	4	3.09	0.96	Colour	2,0,Y,Y	2	8	3.71	1.34
voc-18-bd-6	3	5	3.89	0.91	Frame	2,0,Y,Y	3	12	6.37	2.18
voc-18-bd-7	3	6	4.83	2.13	Frame	2,0,Y,Y	3	14	8.11	2.36
voc-18-bd-8	7	9	7.98	0.49	Frame	2,0,Y,N	4	24	12.24	4.1
voc-18-bd-9	3	7	5.54	1.08	Frame	2,0,Y,Y	4	18	8.83	2.77
voc-18-bd-11	4	8	5.68	1.21	Frame	2,0,Y,Y	3	23	12.43	3.95
voc-18-bd-20	3	3	3	0	Combined	2,0,Y,Y	3	7	4.43	1.19

Table 7.12: MODV best counting estimations.

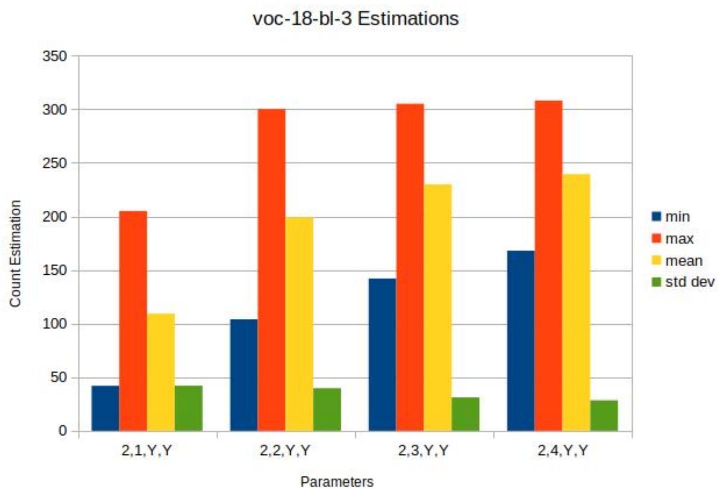
	Ground Truth				**Best Count Estimations**					
Video	**Min**	**Max**	μ	σ	**Method**	**Parameters**	**Min**	**Max**	μ	σ
voc-18-bd-2	4	23	11.32	5.43	Combined	3,0,N,Y	4	22	11.67	4.36
voc-18-bd-10	9	18	13.82	2.96	Combined	2,1,Y,Y	7	20	13.39	2.77
voc-18-bd-19	25	37	33.31	3.05	Frame	3,0,Y,Y	0	54	22.3	8.58

Table 7.13: HODV best counting estimations.

	Ground Truth				**Best Count Estimations**					
Video	**Min**	**Max**	μ	σ	**Method**	**Parameters**	**Min**	**Max**	μ	σ
voc-18-bd-1	258	277	266.82	17.77	Combined	3,1,Y,N	236	310	267.95	15.54
voc-18-bd-3	84	106	92.39	6.69	Frame	2,4,Y,Y	0	123	92.3	18.98
voc-18-bd-12	211	322	296.34	29.63	Combined	3,0,N,N	131	390	293.14	66.31
voc-18-bd-13	47	62	51.83	3.89	Frame	3,1,Y,Y	0	110	31.55	18.32
voc-18-bd-15	33	62	45.54	7.23	Frame	2,3,Y,Y	0	76	48.89	13.13
voc-18-bd-17	106	125	116.45	4.47	Frame	3,1,Y,Y	13	96	51.91	14.15
voc-18-bd-18	87	106	97.57	3.81	Frame	2,1,Y,Y	7	166	77.71	30.64
voc-18-bl-1	698	722	698.13	3.36	Frame	2,5,Y,Y	0	850	448.27	316.28
voc-18-bl-2	460	460	460	0	Combined	2,2,Y,Y	312	490	406.79	40.05
voc-18-bl-3	176	186	184.31	2.05	Frame	2,2,Y,Y	100	300	199.12	39.7
voc-18-bl-4	115	130	122.88	5.13	Frame	2,2,Y,Y	109	168	141.41	12.22



(a) The results for *voc-18-bd-1* with increasing iterations on frame descriptors.



(b) The results for *voc-18-bl-3* with increasing iterations on frame descriptors.

Figure 7.4: Stabilisation of HODV results as more iterations are used.

7.2 Analysis and Comparison to Counting Literature

In the previous section, we discussed the performances of our three algorithms on the videos in our dataset. We showed the performances using the minimum, maximum, mean and standard deviation. We also showed the average time in seconds of how long it takes for our approaches to process one frame. In this section, we compare our approaches to the approaches we discussed in 1.1. While we do not compare them on a dataset basis, we compare the approaches in terms of how they process the data.

The object counting approach detailed in this thesis essentially falls under the 'Counting by Object Detection' discussed in Subsection 1.1.2. Our approach is most similar to the approach in [22]. Both approaches apply density based clustering to SURF features in order to locate the objects and extract the object count estimations although in [22], they use DBSCAN while in this thesis we use HDBSCAN. However, it is worth remembering that HDBSCAN is an extension to DBSCAN and works by finding all possible DBSCAN solution to get around the density parameter ε and be able to detect clusters of varying density.

The similarity between the two, however, ends there. The use of DBSCAN in [22] means that the clusters detected will be of single density. The key difference between the two is that while in [22], the clusters are detected in spatial space, our object detection works in descriptor space. The authors look for spatial clusters in the sample object image and then match those clusters to spatial clusters in the scene image. This means that the objects in both the sample and scence must be large and have enough details to have a lot of features that can form spatial clusters. This approach to detecting the objects means that there is a significant size limitation in terms of the size of objects that can be detected and counted. The other weakness in this algorithm assumes that when objects are large enough the features can form spatial clusters. In videos such as *voc-18-bd-20*, this is true. This fails to hold for videos such as *voc-18-bd-1* where the features do not clearly form spatial clusters as shown in Figure 6.10a.

Our approach matches the SURF features in the descriptor space which is by far the most important part. It also means that we can go as small as a single feature per object as shown in Figures 6.18c and 6.18d. Our approach, therefore, allows us to count objects at much higher number of objects. From their results, the authors of [22] show no more than 5 objects to be counted, which in our approach, would fall under the LODVs. The abundance of features in [22] means that the authors can use homography and perspective transforms to localise the objects which can account for scaled objects. In our approach, we use a single feature to localise the objects and have therefore lost the ability to compensate for size differences between sample and target objects.

The feature learning approaches used in this thesis provide a significant edge against other approaches reviewed in Chapter 1. The approach described in [11] has similar intentions of counting high object densities as the approach in this thesis. The approach uses image density maps which operates at the pixel level while our approach works at the feature level, which is essentially groups of pixel. As such, in [11], they can count much smaller objects as evidenced by their use of the datasets from [12]. Our approach however, is limited by the requirement for detection of at least one feature per object which is why we could not make a direct comparison on the results between our approach and [11].

Our approach has a much faster training approach than in [12]. In our approach, we learn the features of the object of interest on the fly. By using the initial ROI and selecting the features inside it, we simply take the good and the noise features and separate them during the feature matching phase described in Chapter 4. The approach in [12] requires the generation of ground truth density maps from some of the images in the dataset which makes it considerably slower than our approach. This slow training approach is also a problem with deep learning based methods such as the one described in [17].

Trajectory clustering approaches such as [29] and [30] have one key weakness; *the object of interest MUST be moving.* A lot of the approaches in trajectory clustering are therefore used for static camera videos. As discussed in Chapter 2, a lot of the videos in our dataset are from moving cameras. Even with motion compensation approaches such as described in [93], the objects themselves must be moving otherwise the motion compensation will discount the objects of interest as background. The other weakness is that trajectory clustering approaches end up losing a lot of visual information about the objects of interest, the approaches can be used in videos where there is only one object class. As such, or approach provide a more versatile way of counting objects.

7.3 Summary

In this chapter, we explored the performance of the different approaches we used to estimate the object numbers in our video datasets. We measured the performances using the *min, max, mean* (μ), *standard deviation* (σ) and the average time for each video while also providing the information on the number of frames used for the colour model and combined methods. We first discussed the count estimations using the frame descriptors, then discussed using colour model and finally the combination of the two. We provided the evidence under each subsection for the three groups of the videos in our dataset. We also showed and explained the best results for each video as well as the method and parameters that produced those results and discussed ways of selecting the parameters for new videos.

Using frame descriptors in LODVs proved more unstable. The results are often greater than their ground truth counterparts, especially when starting with $minPts = 3$. Empirically, we found out that it is far much better to use $minPts = 2$ without any iterative cluster daisy-chaining. With MODVs, the results for frame descriptors require closer inspection to find the best results. In Table 7.6, one video required $minPts = 3$ for best results while the other required $minpts = 3$ with over-segmentation.

Using the colour model clusters works very well with LODVs where the objects are large enough to have a lot of features within their outline. This is the reason the videos *voc-18-bd-*{*5, 20*} were selected from our dataset. The multiple features within the objects allows for a colour model that has clusters within it. In *voc-18-bd-12*, the birds are very small and as such have only one feature on them and this creates an almost uniform distribution within the colour model. The colour model also works well for HODVs, but tends to under-perform with HODVs. Often, when handling MODVs and HODVs, it is common for the colour model to fail in detecting any other object in the frame.

In videos where the colour model was successfully learned and tracked, the combination of frame and colour model descriptor clustering provided the best results for any parameter combination. The colour model reduces the false positive from the frame descriptor locations while frame descriptors results bring locations that may have been missed by using just the colour model. This type of combination means that this approach is better for HODVs and HODVs. While both *voc-18-bd-5* and *voc-18-bd-20* had successful colour mode tracking, the best results for *voc-18-bd-5* were from colour model clustering while *voc-18-bd-20* had the best results from the combined approach. In contrast, all the MODVs and HODVs where the colour model was successfully tracked had combination results better than both the frame and colour model descriptor.

Chapter 8

Conclusion and Future Work

In this thesis, we have detailed an approach to object counting that relies on matching multiple SURF features using HDBSCAN. We applied our approach to a variety of challenging videos from a dataset we compiled for the task. We have demonstrated an approach of restricting the user interaction to constant time complexity even for high object density videos. We have shown that given a set of SURF features from a video frame with multiple object instances, we can detect clusters in the feature descriptors and used the clusters with a novel approach to localise the objects of interest in the frame. We also showed how we use an MOT developed in this thesis to detect and remove false positive locations. Finally we presented the best results of our counting approach and highlighted the parameters needed to achieve the results.

Chapter 2: We presented a new video dataset designed for video object counting. We discussed the short-comings of available dataset and why they are unsuitable for the object counting as envisioned in this thesis. The videos in our dataset are split into three groups and the videos have different lengths. The first group is the low object density videos where the number of object is less than 10 and has 8 videos. This group is essentially made up of videos where a human can easily count the number of object on the fly. The second group is the medium object density video where the number of objects is between 10 and 50 and has 4 videos. While a person might have a difficult time actually counting the objects, they can make a very accurate estimation. The final group is the high object density videos where the number of objects in the video is greater than 50 making it very difficult if not impossible for a person to make an accurate prediction.

In the chapter we presented the ground truth for comparison with the object counting approach presented in this thesis. We developed the approach using a human-computer hybrid approach. We take advantage of people's ability to accurately identify ambiguous objects and the computer's ability to accurately count well defined objects. In our approach, for each video frame, the human has a task of identifying the objects to be counted and tagging them with a white marker before threshholding all the colours thus creating a binary image of the frame. The binary images are then processed using a connected components algorithm to count the number of white dots in the image thus getting the ground truth on the number of objects in the frame.

While the dataset works well for the task in this thesis, there is still a lot of room for improvement. Currently, the dataset contains two sets of classes; 20 bird videos and 4 blood videos. Future improvement on that would be to include more object classes. The current benchmark in the dataset is object count ground truth on a frame by frame basis. The improvement on this is to provide the number of distinct objects in the video. This would require tracking the objects as they move in and out of the video frames while keeping track of the ones that are still in the frame. Other benchmarks common to tracking, identification and segmentation can also be added to the dataset to allow it to be used with other forms of computer vision tasks.

Chapter 3: We set up the background on most of the techniques used in out object counting approach. We first discuss the single object tracker as implemented in the OpenCV library. We compare the performances of the different algorithms in terms of drift and accuracy. We found that the BOOSTING tracker performed better than the other algorithms tested. We presented the results of the tests to show the reason for our SOT of choice.

In Section 3.2, we review the multi object tracker algorithms. We explained our need for an MOT in terms of removing the false positive locations from out counting approach. We did not directly make comparisons of the algorithms, instead we highlighted the big weaknesses and explained why they are unsuitable for the task explained in Chapters 5 and 6.

In Section 3.3, we gave an overview of local features and reviewed the literature on SURF features. We highlighted the shortfalls in the current matching algorithms. We explained that brute force matcher is concerned with one-on-one feature matching compared with one-to-many approach required for the approach in this thesis. FLANN allows for matching multiple features, but it is concerned with *k-nearest neighbours*. This requires the user to know how many features have to match, which we argue is not possible to know beforehand. We also discussed how the current metric of measuring match quality, the Lowe Ratio Test, does not work when matching multiple features since it only measures match quality between 2 features. Lastly, we present Hypothesis 1 to highlight the clustering nature of multiple featues that are similar and Hypothesis 2 to highlight further why Lowe's Ration Test will not work with multiple feature matching.

The algorithms behind the multi matching approach used in this thesis are explored in Section 3.4. We explore the density-based clustering literature in terms of DBSCAN and HDBSCAN. In this thesis, we use HDBSCAN which did not have C or C++ implementations. We therefore implemented it using C/C++ with Java and Python bindings. We also developed a way of analysing the clusters detected and provide a bound confidence value for each cluster that current clustering methods lack as well as validity values for overall clustering results.

Chapter 4: The problems with current local feature matching are addressed in this chapter. We show how to achieve multiple feature matching by detecting the clusters within the descriptor space of the SURF features. We showed that we can use the *minPts* parameter to explore the descriptor dataset to find the best clustering results. We discovered that the validity of the clustering results may vary from one frame to the next.

In Section 4.2, we explored the use of varying *minPts* values to find the best clustering results based on the validity values or the number of clusters detected. The experiments showed that this approach is very useful at increasing the number of clustering results with high validities. This approach was adopted for the rest of the thesis with the *minPts* values determined experimentally to be between 3 and 7 inclusively. We also determined that with these *minPts* values, the mean validities for the frames was above 3.0 while the standard deviation was acceptable if it was below 1,0.

In an effort to get more clustering results that have higher mean validity and low standard deviation, we explore the use of $minPts = 2$ in Section 4.3. This essentially allows us to attempt a one-to-one matching of features. This proved very effective at increasing the mean and decreasing the standard deviation of validities. The other effect of this approach is the break-up of clusters that would have had high confidences with higher values of *minPts*. which has the effect of returning lot of clusters with just two features. We did however find out that in some cases, the clusters do not break up if they are stable enough.

The rotationally invariant nature of SURF features was addressed in Section 4.4. We proposed augmentation of the descriptor set with the angles detected by SURF for each feature. We showed that with this augmentation, HDBSCAN clusters within the results tend to have features with similar angles. The augmentation of the descriptors with the angles did not always improve the validities of the clusters.

In this thesis, we matched feature descriptors within the frame which has the disadvantage of losing the temporal information available in videos. A proposed improvement to the current approach is to match the descriptors both in the spatial and temporal domains. This approach would bring more stability to the descriptor clusters by increasing the number of similar features within the dataset.

Chapter 5: In this chapter, we developed a multi-object tracker. This novel approach to multi object tracking allows for tracking multiple objects without having prior knowledge about the number of objects in the video. While we did not use the approach to track actual objects in this thesis, we showed that our approach can learn the features needed to track multiple objects. The results showed that even when dealing with thousands of possible features to track, the fast implementation of HDBSCAN helps the approach to train much faster. The provision of colour clusters allows the user to provide supervision to the learning algorithm while not suffering the effects of the large number of features in the video.

In our approach, we used colours at the detected SURF feature location as features to be tracked using two frames at a time. This approach was shown to have mixed results in tracking colour models and is highly reliant on the features detected by SURF. Several improvements can be added to our tracking features to improve the robustness of the tracking algorithm:

- Use more than two frames for tracking. While this will increase the tracking time by increasing the number of features being processed, there is a potential benefit of more stable tracking results. This increased processing times may not be a problem when dealing with LODVs. In *voc-18-bd-5*, there was on average 141 frames (See Subsection 5.2.1) while *voc-18-bd-1* had an average of 3375 features.
- Use feature descriptors instead of colours. Arguably, descriptors offer more precision in similarity than colours and are more resilient to noisy frames.
- Explore other features and image manipulations as features to track. Features such as HoG and optical flow provide more information on structure and movement of objects which may help increase the robustness of our multi-tracker.

Chapter 6: We used the descriptor clusters we detected in Chapter 4 to locate the objects in frame. We first explained the object of interest feature learning that identifies the SURF features located inside an initial ROI. We extracted all clusters that intersect the initial ROI and used the points of intersection to find the locations of the other objects. We explained the reasons we could not use homography and showed that our method of shifting and localised template matching works well for locating objects identified through the features in the clusters.

We highlighted the weakness of our localisation algorithm against scale challenges. We showed that the scale of the feature does not necessarily translate to the scale of the object. Currently, the search area for localised template matching is set to be twice the width and height of the sample area. Future improvements can use the feature area size to determine the size to the target area. A constraint to this approach is that this would only have to apply where the target feature has a larger size since template matching cannot work when the template is smaller than the target.

The other weakness for our algorithm is the detection of false positives. Our current approach is to rely on the cluster matches to give good locations. However, we encountered situations where the localisation algorithm provided false positives. We countered rotational false positives by augmenting the descriptors with feature angles. We countered low validity clustering results by varying *minPts* and cluster over-segmentation. Finally, we used the colour model to minimise false positive in two ways; first, by detecting the clusters within the colour model descriptors and then combining the frame descriptor clustering locations with the colour model descriptor clustering locations. Future improvement to our localisation algorithm could include:

- Use the moments to fine tune the localisations. This can be used instead of template matching although it might not work well for scale.
- Use histograms to better detect the differences in sizes of the template object and the detected locations which can help reduce scale localisation problems.
- Research ways to better detect an object being counted multiple times.
- Research ways to better detect occlusions that lead to one of the object being discarded.

Chapter 7: In this chapter, we showed the the performance of our counting algorithms on our dataset videos. We showed the best results under each of the count estimation methods we developed in this thesis. We showed the best results for each video based on the three counting methods. We discovered that in videos where the colour model was successfully tracked, the combination method provides the best count estimations. We explained how to use our methods to count the objects in HODVs where the ground truth by showing that iterative cluster daisy-chaining stabilises the results which provide very good indications of the number of objects in the videos.

In Section 7.2, we discuss the merits and deficiencies of our approach. Essentially, compared with ‘Object Density Estimation’ which work at the pixel level, our approach performs best in terms of the feature learning but does not handle small objects very well. Against ‘Trajectory Clustering’, our method performs better by being not being restricted to the moving objects. Compared with the ‘Object Detection’, our approach does not perform as accurately as deep learning based approaches but makes up for by learning the object features online while still being able to count generic objects unlike shape-based approaches such as hough transform for blood cell counting.

The approach detailed in this thesis works on a frame-by-frame basis. As such the information of the object detected in the previous frame is never used to fine tune the localisation in the current frame localisation. This is noticeable when looking at the counting results in which in one frame the objects were properly located while in the next they were not. We also had situations where an object in the middle is missed in one frame but detected in the next. Currently we have no ways of validating its appearance in the new frame.

Future improvements to the approach will use location information from previous frames. The use of such information also provides a way of achieving long-term object tracking which would be a significant step towards extraction of aggregated video object count. The aggregated count would be verified through the aggregated ground truth suggested above. The location information from the previous frames would also help reduce frame-to-frame object count estimation variations.

Bibliography

[1] "Birdlife international." `http://www.birdlife.org/`, 2018. (Date last accessed 8-May-2018).

[2] "Convention on international trade in endangered species of wild fauna and flora." `https://www.cites.org/`, 2018. (Date last accessed 8-May-2018).

[3] F. P. T., S. Paul, and P. R. A., "Using super-high resolution satellite imagery to census threatened albatrosses," *Ibis*, vol. 159, no. 3, pp. 481–490.

[4] B. Botswana, "Birdlife botswana." `http://www.birdlifebotswana.org.bw/`. (Date last accessed 8-May-2018).

[5] B. Botswana, "Bird population monitoring in botswana." `http://www.birdlifebotswana.org.bw/doc/bpm_bird_population_monitoring_programme_for_botswana.pdf`. (Date last accessed 8-May-2018).

[6] J. G. A. Barbedo, "Using digital image processing for counting whiteflies on soybean leaves," *Journal of Asia-Pacific Entomology*, vol. 17, no. 4, pp. 685–694, 2014.

[7] W. K. Poon, C. J. Wong, K. Abdullah, E. S. Lim, and C. K. Teo, "Quantification of migratory raptors using digital single reflex camera," in *Humanities, Science and Engineering (CHUSER), 2011 IEEE Colloquium on*, pp. 791–796, Dec 2011.

[8] "The vulture crisis." https://www.birdlife.org/worldwide/vulture-crisis, 2018. (Date last accessed 8-May-2018).

[9] B. International, "Africa's vultures are sliding towards extinction warns birdlife." http://www.birdlife.org/worldwide/news/africa%E2%80%99s-vultures-are-sliding-towards-extinction-warns-birdlife. (Date last accessed 8-May-2018).

[10] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, NIPS'10, pp. 1324–1332, Curran Associates Inc., 2010.

[11] Y. Wang, Y. Zou, and W. Wang, "Manifold-based visual object counting," *IEEE Transactions on Image Processing*, vol. 27, pp. 3248–3263, July 2018.

[12] Z. Ma, L. Yu, and A. B. Chan, "Small instance detection by integer programming on object density maps," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3689–3697, June 2015.

[13] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–7, June 2008.

[14] K. Chen, C. C. Loy, S. Gong, and T. Xiang, "Feature mining for localised crowd counting," in *In BMVC*.

[15] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, "Multi-source multi-scale counting in extremely dense crowd images," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2547–2554, June 2013.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[17] S. Amirgholipour, X. He, W. Jia, D. Wang, and M. Zeibots, "A-ccnn: Adaptive ccnn for density estimation and crowd counting," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 948–952, Oct 2018.

[18] D. Oñoro-Rubio and R. J. López-Sastre, "Towards perspective-free object counting with deep learning," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 615–629, Springer International Publishing, 2016.

[19] P. Hu and D. Ramanan, "Finding tiny faces," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[20] J. Fabic, I. Turla, J. Capacillo, L. David, and P. Naval, "Fish population estimation and species classification from underwater video sequences using blob counting and shape analysis," in *Underwater Technology Symposium (UT), 2013 IEEE International*, pp. 1–6, March 2013.

[21] O. Barinova, V. Lempitsky, and P. Kholi, "On detection of multiple object instances using hough transforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 1773–1784, Sept 2012.

[22] N. K. Verma, T. Sharma, R. K. Sevakula, and A. Salour, "Vision based object counting using speeded up robust features for inventory control," in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 709–714, Dec 2016.

[23] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," *Data Min. Knowl. Discov.*, vol. 2, pp. 169–194, June 1998.

[24] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008. Similarity Matching in Computer Vision and Multimedia.

[25] M. R. Hsieh, Y. L. Lin, and W. H. Hsu, "Drone-based object counting by spatially regularized regional proposal network," in *2017 IEEE International Con-*

ference on Computer Vision (ICCV), pp. 4165–4173, Oct 2017.

[26] T. Stahl, S. L. Pintea, and J. C. van Gemert, "Divide and count: Generic object counting by image divisions," *IEEE Transactions on Image Processing*, pp. 1–1, 2018.

[27] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[28] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results." http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[29] V. Rabaud and S. Belongie, "Counting crowded moving objects," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, pp. 705–711, June 2006.

[30] G. Antonini and J. P. Thiran, "Counting pedestrians in video sequences using trajectory clustering," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, pp. 1008–1020, Aug 2006.

[31] T. Zhao and R. Nevatia, "Bayesian human segmentation in crowded situations," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2, pp. II–459, June 2003.

[32] S. M. Mazalan, N. H. Mahmood, and M. A. A. Razak, "Automated red blood cells counting in peripheral blood smear image using circular hough transform," in *2013 1st International Conference on Artificial Intelligence, Modelling and Simulation*, pp. 320–324, Dec 2013.

[33] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I–511–I–518 vol.1, 2001.

[34] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[35] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander, “Hierarchical density estimates for data clustering, visualization, and outlier detection,” *ACM Trans. Knowl. Discov. Data*, vol. 10, pp. 5:1–5:51, July 2015.

[36] Yale, “The extended yale face database b (cropped),” 2001.

[37] L. Wolf, T. Hassner, and I. Maoz, “Face recognition in unconstrained videos with matched background similarity,” in *CVPR 2011*, pp. 529–534, June 2011.

[38] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, “300 faces in-the-wild challenge: The first facial landmark localization challenge,” in *2013 IEEE International Conference on Computer Vision Workshops*, pp. 397–403, Dec 2013.

[39] G. Griffin, A. Holub, and P. Perona, “Caltech-256 object category dataset,” Tech. Rep. 7694, California Institute of Technology, 2007.

[40] F. Galasso, N. S. Nagaraja, T. J. Cárdenas, T. Brox, and B. Schiele, “A unified video segmentation benchmark: Annotation, metrics and analysis,” in *2013 IEEE International Conference on Computer Vision*, pp. 3527–3534, Dec 2013.

[41] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbelaez, A. Sorkine-Hornung, and L. V. Gool, “The 2017 DAVIS challenge on video object segmentation,” *CoRR*, vol. abs/1704.00675, 2017.

[42] Y. Wu, J. Lim, and M. H. Yang, “Object tracking benchmark,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 1834–1848, Sept 2015.

[43] A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, and K. Schindler, “MOT16: A benchmark for multi-object tracking,” *CoRR*, vol. abs/1603.00831, 2016.

[44] M. Müller, A. Bibi, S. Giancola, S. Al-Subaihi, and B. Ghanem, “Trackingnet: A large-scale dataset and benchmark for object tracking in the wild,” *arXiv preprint arXiv:1803.10794*, 2018.

[45] J. Lou, M. Zhou, Q. Li, C. Yuan, and H. Liu, "An automatic red blood cell counting method based on spectral images," in *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1391–1396, Oct 2016.

[46] B. Venkatalakshmi and K. Thilagavathi, "Automatic red blood cell counting using hough transform," in *2013 IEEE Conference on Information Communication Technologies*, pp. 267–271, April 2013.

[47] D. Ryan, S. Denman, C. Fookes, and S. S., "Crowd counting using multiple local features," in *2009 Digital Image Computing: Techniques and Applications*, pp. 81–88, Dec 2009.

[48] S. Yoshinaga, A. Shimada, and R. ichiro Taniguchi, "Real-time people counting using blob descriptor," *Procedia - Social and Behavioral Sciences*, vol. 2, no. 1, pp. 143–152, 2010. The 1st International Conference on Security Camera Network, Privacy Protection and Community Safety 2009.

[49] F. Bashir and F. Porikli, "Performance Evaluation of Object Detection and Tracking Systems," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS2006)*, June 2006.

[50] S. Bera, "Partially occluded object detection and counting," in *Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT)*, pp. 1–6, Feb 2015.

[51] "Lightning memory-mapped database manager (lmdb)." `http://www.lmdb.tech/doc/`, 2015. (Date last accessed 18-Jan-2019).

[52] OpenCV, "Open source computer vision library." `https://github.com/opencv/opencv`, 2015. (Date last accessed 8-May-2018).

[53] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, pp. 1150–1157 vol.2, 1999.

[54] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pp. 331–340, INSTICC Press, 2009.

[55] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," pp. 226–231, AAAI Press, 1996.

[56] S. V. Kothiya and K. B. Mistree, "A review on real time object tracking in video sequences," in *Electrical, Electronics, Signals, Communication and Optimization (EESCO), 2015 International Conference on*, pp. 1–4, Jan 2015.

[57] Y. Wu, J. Lim, and M. H. Yang, "Online object tracking: A benchmark," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 2411–2418, June 2013.

[58] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," pp. 6–11, 01 2006.

[59] R. McConnell, "Method of and apparatus for pattern recognition," 1982. US Patent Application US06/927,832; Current Assignee Research Foundation of State University of New York.

[60] T. Ojala, M. Pietikainen, and D. Harwood, "Performance evaluation of texture measures with classification based on kullback discrimination of distributions," in *Proceedings of 12th International Conference on Pattern Recognition*, vol. 1, pp. 582–585 vol.1, Oct 1994.

[61] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial Intelligence*, vol. 89, no. 1, pp. 31 – 71, 1997.

[62] B. Babenko, M. H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 983–990, June 2009.

[63] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, pp. 583–596, March 2015.

[64] A. Lukežič, T. Vojíř, L. Čehovin Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter tracker with channel and spatial reliability," *International Journal of Computer Vision*, Jan 2018.

[65] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," in *2010 20th International Conference on Pattern Recognition*, pp. 2756–2759, Aug 2010.

[66] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2544–2550, June 2010.

[67] D. S. Bolme, B. A. Draper, and J. R. Beveridge, "Average of synthetic exact filters," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2105–2112, June 2009.

[68] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 1409–1422, July 2012.

[69] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), p. 749–765, Springer International Publishing, 2016.

[70] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[71] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *Proceedings. International Conference on Image Processing*, vol. 1, pp. I–900–I–903 vol.1, 2002.

[72] Y. Zhang, Y. Tang, B. Fang, and Z. Shang, "Fast multi-object tracking using convolutional neural networks with tracklets updating," in *2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, pp. 313–317, Dec 2017.

[73] L. Chen, H. Ai, C. Shang, Z. Zhuang, and B. Bai, "Online multi-object tracking with convolutional neural networks," in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 645–649, Sept 2017.

[74] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, July 2017.

[75] L. Wancun, T. Wenyan, Z. Liguo, Z. Xiaolin, and L. Jiafu, "Multi-scale behavior learning for multi-object tracking," in *2017 First International Conference on Electronics Instrumentation Information Systems (EIIS)*, pp. 1–5, June 2017.

[76] C. Wu, H. Sun, H. Wang, F. Kun, G. Xu, W. Zhang, and X. Sun, "Online multi-object tracking via combining discriminative correlation filters with making decision," *IEEE Access*, pp. 1–1, 2018.

[77] L. Lan, X. Wang, S. Zhang, D. Tao, W. Gao, and T. S. Huang, "Interacting tracklets for multi-object tracking," *IEEE Transactions on Image Processing*, vol. 27, pp. 4585–4597, Sept 2018.

[78] E. Türetken, X. Wang, C. J. Becker, C. Haubold, and P. Fua, "Network flow integer programming to track elliptical cells in time-lapse sequences," *IEEE Transactions on Medical Imaging*, vol. 36, pp. 942–951, April 2017.

[79] X. Wang, B. Fan, S. Chang, Z. Wang, X. Liu, D. Tao, and T. S. Huang, "Greedy batch-based minimum-cost flows for tracking multiple objects," *IEEE Transactions on Image Processing*, vol. 26, pp. 4765–4776, Oct 2017.

[80] A. Maksai, X. Wang, F. Fleuret, and P. Fua, "Non-markovian globally consistent multi-object tracking," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2563–2573, Oct 2017.

[81] M. Hassaballah, A. Ali, and H. Alshazly, *Image Features Detection, Description and Matching*, vol. 630, pp. 11–45. 02 2016.

[82] M. El-gayar, H. Soliman, and N. meky, "A comparative study of image low level feature extraction algorithms," *Egyptian Informatics Journal*, vol. 14, no. 2, pp. 175–181, 2013.

[83] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1615–1630, 2005.

[84] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, 2014.

[85] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," 1967.

[86] Q. Ye, W. Gao, and W. Zeng, "Color image segmentation using density-based clustering," in *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, vol. 2, pp. II–401–4 vol.2, July 2003.

[87] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," *SIGMOD Rec.*, vol. 28, pp. 49–60, June 1999.

[88] L. Dagum and R. Menon, "Openmp: An industry-standard api for shared-memory programming," *IEEE Comput. Sci. Eng.*, vol. 5, pp. 46–55, jan 1998.

[89] H. Du, "Cluster evaluation, validation and interpretation," in *Data Mining Techniques and Application: An Introduction*, ch. Basic Techniques for Cluster Detection, pp. 100 – 105, Course Technology Cengage Learning, 1st ed., 2010.

[90] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* New York, NY, USA: Cambridge University Press, 2 ed., 2003.

[91] A. Kaehler and G. Bradski, “Homography,” in *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*, ch. Camera Models and Calibration, pp. 660 – 665, O’Reilly, 2016.

[92] A. Kaehler and G. Bradski, “Template matching,” in *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*, ch. Histograms and Templates, pp. 397 – 403, O’Reilly, 2016.

[93] J. Kim, J. Shin, and D. Kim, “Fh-adaptive motion compensation for moving object detection in an active camera,” in *2009 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pp. 532–537, Dec 2009.

Appendices

Appendix A

Code Samples

A.1 HDBSCAN Code Snippets

```
uint trianglular(uint n) {
  return (n * n + n) / 2;
}
```

Listing A.1: Function for encoding and decoding the indices into the optimised distance array.

```
double distance_get(distance* dis, uint row, uint col) {
  uint idx;
  if (row < col) {
    idx = (dis->rows * row + col) - trianglular(row + 1);

  } else if (row == col) {
    return 0;
  } else {
    idx = (dis->rows * col + row) - trianglular(col + 1);
  }
  return dis->distances[idx];
}
```

Listing A.2: Function for retrieving the distance values from the optimised distance array.

```
void distance_get_core_distances(distance *dis){

  double sortedDistance[dis->rows];
  #pragma omp parallel for private(sortedDistance)
  for (uint i = 0; i < dis->rows; i++) {
    for (uint j = 0; j < dis->rows; j++) {
      sortedDistance[j] = distance_get(dis, i, j);
    }
    qsort(sortedDistance, dis->rows, sizeof(double), cmpdouble);
    dis->coreDistances[i] = sortedDistance[dis->numNeighbors];
  }
}
```

Listing A.3: Getting the core distances from the optimised array.

```

void distance_compute(distance* dis, void* dataset, int rows, int cols,
    int numNeighbors){
  dis->numNeighbors = numNeighbors;
  setDimenstions(dis, rows, cols);

  #pragma omp parallel for
  for (uint i = 0; i < dis->rows; i++) {
    for (uint j = i; j < dis->rows; j++) {
      double sum, diff = 0.0;
      uint offset1;
      sum = 0;

      for (uint k = 0; ((k < dis->cols) && (i != j)); k++) {
        diff = get_diff(dis, dataset, i, j, k);
        sum += (diff * diff);
      }

      sum = sqrt(sum);

      int c;
      if (j > i) {
        // Calculate the linearised upper triangular matrix offset
        offset1 = i * dis->rows + j;
        c = offset1 - triangular(i + 1);

        dis->distances[c] = sum;
      } else if (i == j) {
        c = -1;
      } else {
        offset1 = j * dis->rows + i;
        c = offset1 - triangular(j + 1);
      }
    }
  }
  distance_get_core_distances(dis);
}
```

Listing A.4: Function for calculating euclidean distances given an array of vectors. The distances are stored in an array optimised for memory. The function uses OpenMP to parallelise the ditance calculations.

A.2 Video Dataset Code Snippets

```
cap = cv2.VideoCapture(vidFile)

frameCount = 0
while True:
  ret, frame = cap.read()

  if cv2.waitKey(1) & 0xFF == ord('q') or not ret:
    break

  frameCount += 1
  cv2.imshow('frame', frame)
  frameName = outFolder + "/" + str(frameCount) + " " + "frame.jpg"
  cv2.imwrite(frameName, frame)
```

Listing A.5: Extracting and saving video frames

```
def extractGroundTruth(t_count):
  files = [f for f in os.listdir(options['i']) if os.path.isfile(os.path
    .join(options['i'], f))]
  for f in files:
    f_name = os.path.join(options['i'], f)
    dna = scipy.misc.imread(f_name)
    dnaf = ndimage.gaussian_filter(dna, 3)
    T = 25
    labeled, nr_objects = ndimage.label(dnaf > T)
    fnum = f.strip().split()[0]
    t_count[fnum] = str(nr_objects)
```

Listing A.6: Extracting ground truth from binary images

```
def createDB(fname, t_count):
  lmdb_env = lmdb.open(fname, map_size=int(1e9))
  for key, value in t_count.items():
    with lmdb_env.begin(write=True) as lmdb_txn:
      lmdb_txn.put(key.encode('utf-8'), value.encode('utf-8'))
```

Listing A.7: Creating an LMDB database for the ground truth

```
def readDB(fname):
  dc = {}
  lmdb_env = lmdb.open(fname)
  lmdb_txn = lmdb_env.begin()
  lmdb_cursor = lmdb_txn.cursor()
  for key, value in lmdb_cursor:
    dc[int(key.decode("utf-8"))] = int(value.decode("utf-8"))
  return dc
```

Listing A.8: Reading the ground truth from the LMDB database

Appendix B

Best Counting Estimation Results

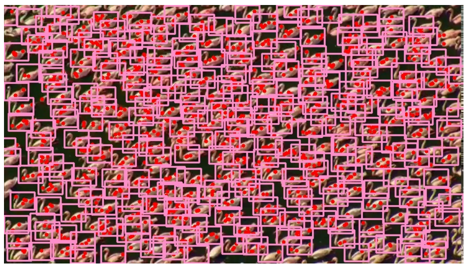
This appendix shows the best results from our approaches as discussed in Chapter 7. For each of the videos, we show results for the first five frames in the form of a table showing the number of features in each frame, the number of features used to estimate the count, the estimation and the ground truth and the time it took to process each video and other useful test values. In the cases where the best results were obtained using the colour model - descriptor combination (See Subsection 6.5.2), only the frame number, count estimation, ground truth, accuracy and duration are shown. We also show the figures of the first four frames with the final localisation results for each video.

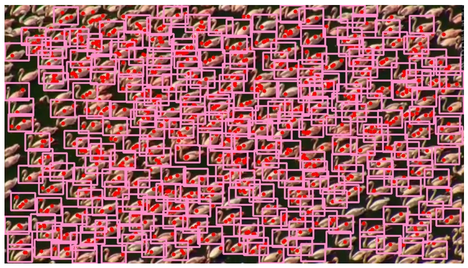
B.1 VOC-18-BD-1

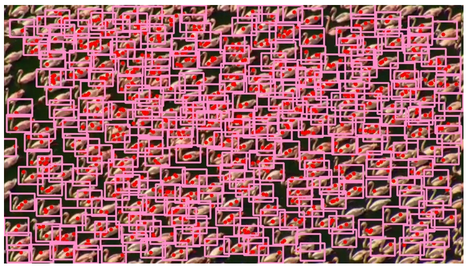
Ground Truth : $min = 258, max = 277, \mu = 266.82, \sigma = 17.17$

Table B.1: Count estimation for *voc-18-bd-1* using combined frame and colour model descriptor results and parameters $minPts = 3, I = 1, O = Y, R = N$.

Frame #	**Count Estimation**	**Ground Truth**	**Accuracy**	**Duration (s)**
1	287	273	105.128	1.529
2	264	272	97.0588	1.014
3	288	275	104.727	3.129
4	252	274	91.9708	1.626
5	272	272	100	1.129


(a) Frame 1.


(b) Frame 2.


(c) Frame 3.

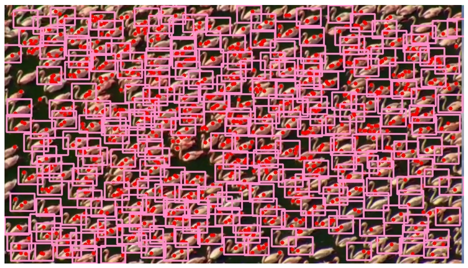

(d) Frame 4.

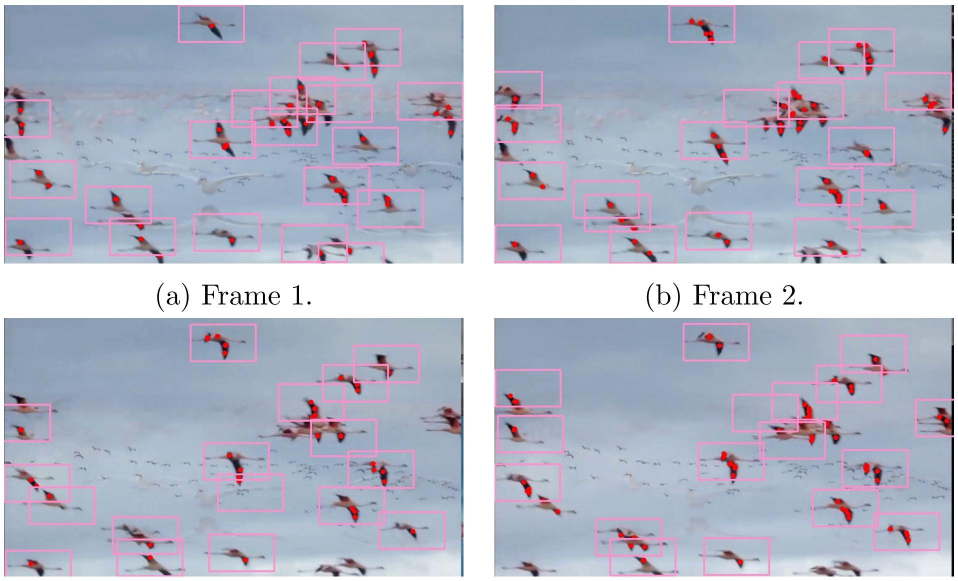
Figure B.1: Combined object localisations for the first 4 frames of *voc-18-bd-1*.

B.2 VOC-18-BD-2

Ground Truth: $min = 4, max = 23, \mu = 11.32, \sigma = 5.43$

Table B.2: Count estimation for *voc-18-bd-2* using combined frame and colour model descriptor results and parameters $minPts = 3, I = 0, O = N, R = Y$.

Frame #	**Count Estimation**	**Ground Truth**	**Accuracy**	**Duration (s)**
1	21	23	91.3	0.1022
2	22	23	95.65	0.1328
3	13	22	59.09	0.0719
4	18	21	85.71	0.1056
5	16	20	80	0.0994


(a) Frame 1. (b) Frame 2.

(c) Frame 3. (d) Frame 4.

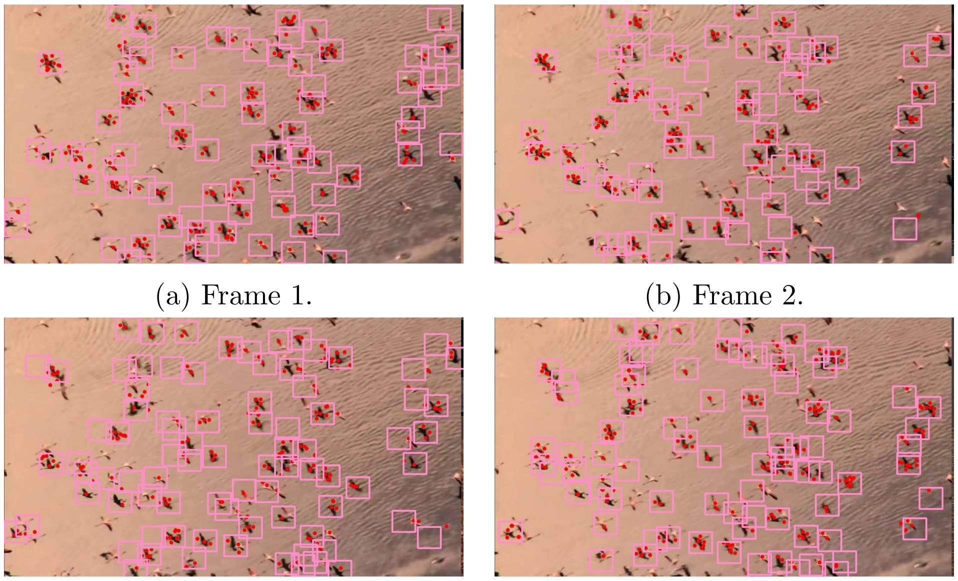
Figure B.2: Combined object localisations for the first 4 frames of *voc-18-bd-2*.

B.3 VOC-18-BD-3

Ground Truth: $min = 84, max = 106, \mu = 92.39, \sigma = 6.69$

Table B.3: Count estimation for *voc-18-bd-3* using frame descriptors and parameters $minPts = 2, I = 4, O = Y, R = Y$.

Frame #	**Feature Size**	**Selected Features**	**# Clusters**	**Count Estimation**	**Ground Truth**	**minPts**	**Validity**	**Accuracy**	**Duration (s)**
1	415	250	81	77	92	2	4	83.69	0.0839
2	431	232	77	72	92	2	4	78.26	0.0595
3	394	238	78	84	91	2	4	92.3	0.0639
4	407	269	81	82	90	2	4	91.11	0.0751
5	430	252	79	85	90	2	4	94.44	0.0707


(a) Frame 1. (b) Frame 2.

(c) Frame 3. (d) Frame 4.

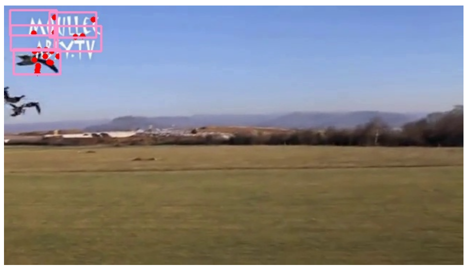
Figure B.3: Object localisations for the first 4 frames of *voc-18-bd-3*.

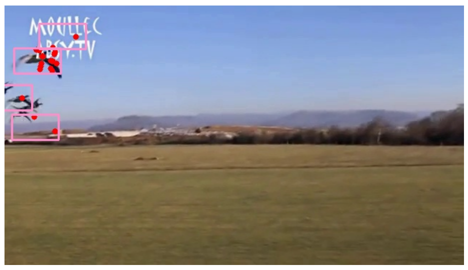
B.4 VOC-18-BD-4

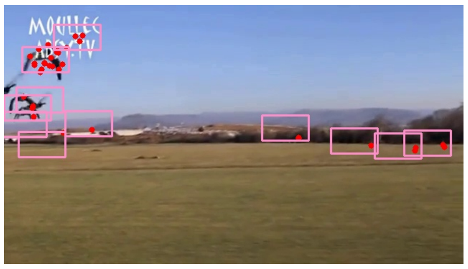
Ground Truth: $min = 4, max = 6, \mu = 5.31, \sigma = 2.55$

Table B.4: Count estimation for *voc-18-bd-4* using frame descriptors and parameters $minPts = 2, I = 0, O = Y, R = N$.

Frame #	**Feature Size**	**Selected Features**	**# Clusters**	**Count Estimation**	**Ground Truth**	**minPts**	**Validity**	**Accuracy**	**Duration (s)**
1	198	19	9	5	4	2	4	125	0.0194
2	210	21	9	4	4	2	4	100	0.0091
3	206	35	13	11	4	2	4	275	0.0173
4	208	20	9	3	4	2	4	75	0.012
5	200	29	9	9	4	2	4	225	0.0116


(a) Frame 1.


(b) Frame 2.


(c) Frame 3.

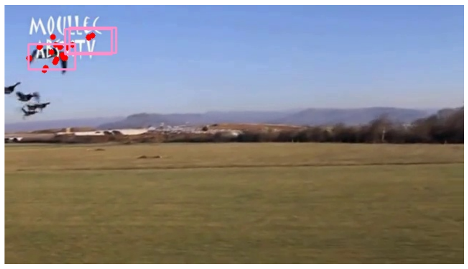

(d) Frame 4.

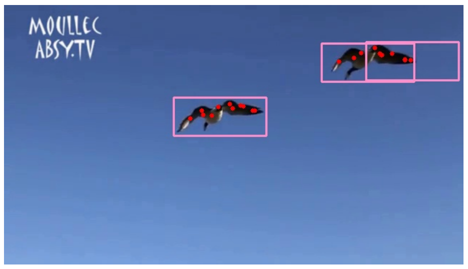
Figure B.4: Object localisations for the first 4 frames of *voc-18-bd-4*.

B.5 VOC-18-BD-5

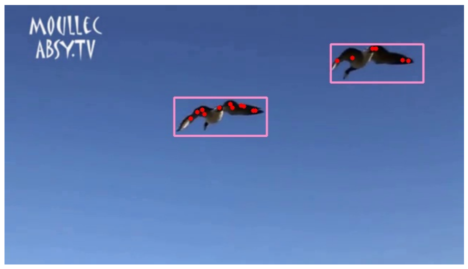
Ground Truth: $min = 2, max = 4, \mu = 3.09, \sigma = 0.96$

Table B.5: Count estimation for *voc-18-bd-5* using colour model descriptors and parameters $minPts = 2, I = 0, O = Y, R = Y$.

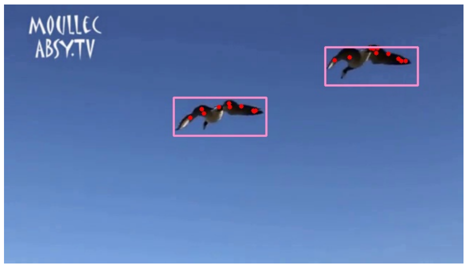
Frame #	Feature Size	Selected Features	# Clusters	Count Estimation	Ground Truth	minPts	Validity	Accuracy	Duration (s)
1	23	22	8	3	2	2	0	150	0.0113
2	22	17	5	2	2	2	0	100	0.0089
3	26	20	8	2	2	2	4	100	0.0101
4	24	19	9	3	2	2	4	150	0.0072
5	22	19	6	3	2	2	0	150	0.0118



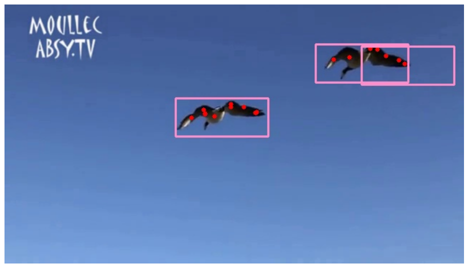
(a) Frame 1.



(b) Frame 2.



(c) Frame 3.



(d) Frame 4.

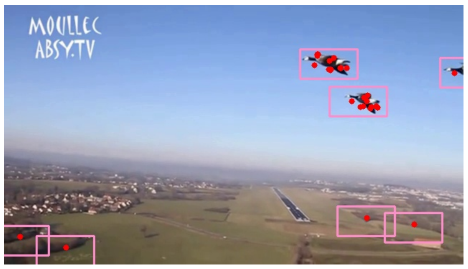
Figure B.5: Object localisations for the first 4 frames of *voc-18-bd-5*.

B.6 VOC-18-BD-6

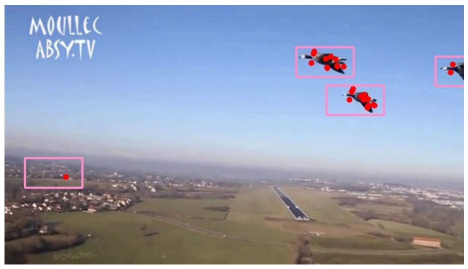
Ground Truth: $min = 3, max = 5, \mu = 3.89, \sigma = 0.91$

Table B.6: Count estimation for *voc-18-bd-6* using frame descriptors and parameters $minPts = 2, I = 0, O = Y, R = Y$.

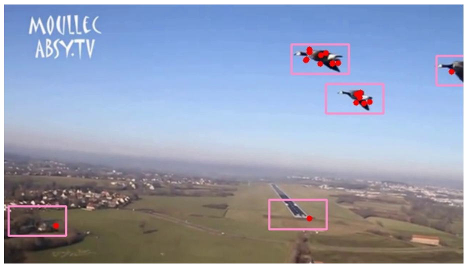
Frame #	**Feature Size**	**Selected Features**	**# Clusters**	**Count Estimation**	**Ground Truth**	**minPts**	**Validity**	**Accuracy**	**Duration (s)**
1	246	28	10	7	3	2	4	233.333	0.0188
2	242	26	10	4	3	2	4	133.333	0.0143
3	251	22	9	5	3	2	4	166.667	0.0134
4	254	20	10	3	3	2	4	100	0.0147
5	254	27	12	4	3	2	4	133.333	0.0227



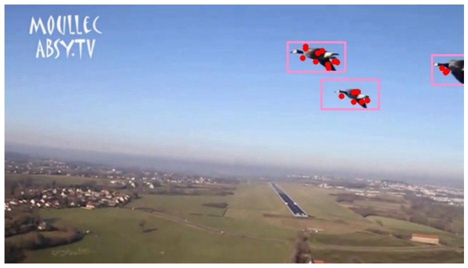
(a) Frame 1.



(b) Frame 2.



(c) Frame 3.



(d) Frame 4.

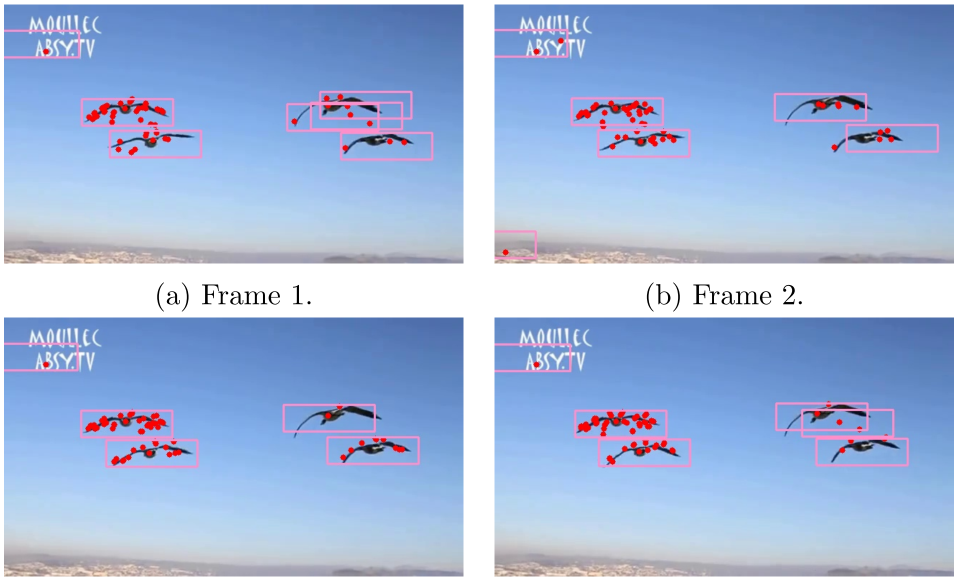
Figure B.6: Object localisations for the first 4 frames of *voc-18-bd-6*.

B.7 VOC-18-BD-7

Ground Truth: $min = 3, max = 6, \mu = 4.83, \sigma = 2.13$

Table B.7: Count estimation for *voc-18-bd-7* using frame descriptors and parameters $minPts = 2, I = 0, O = Y, R = Y$.

Frame #	**Feature Size**	**Selected Features**	**# Clusters**	**Count Estimation**	**Ground Truth**	**minPts**	**Validity**	**Accuracy**	**Duration (s)**
1	201	58	22	7	4	2	4	175	0.0333
2	197	60	22	6	4	2	4	150	0.0227
3	212	70	19	5	4	2	4	125	0.0239
4	221	70	25	6	4	2	4	150	0.0225
5	213	59	20	4	4	2	4	100	0.0207



(a) Frame 1. (b) Frame 2.

(c) Frame 3. (d) Frame 4.

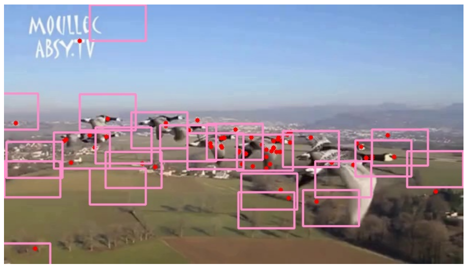
Figure B.7: Object localisations for the first 4 frames of *voc-18-bd-7*.

B.8 VOC-18-BD-8

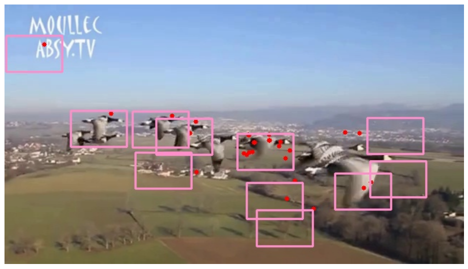
Ground Truth: $min = 7, max = 9, \mu = 7.98, \sigma = 0.49$

Table B.8: Count estimation for *voc-18-bd-8* using frame descriptors and parameters $minPts = 2, I = 0, O = Y, R = N$.

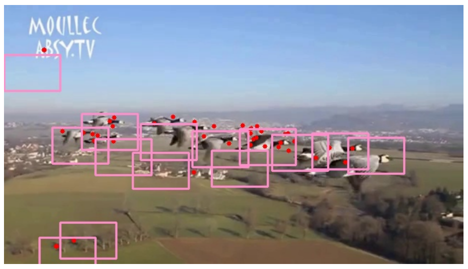
Frame #	**Feature Size**	**Selected Features**	**# Clusters**	**Count Estimation**	**Ground Truth**	**minPts**	**Validity**	**Accuracy**	**Duration (s)**
1	379	66	22	23	9	2	4	255.556	0.0378
2	380	28	12	11	9	2	4	122.222	0.0201
3	355	43	14	14	9	2	4	155.556	0.0235
4	375	22	11	9	9	2	4	100	0.0175
5	374	33	13	11	9	2	4	122.222	0.0188



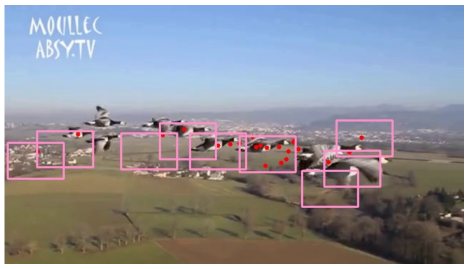
(a) Frame 1.



(b) Frame 2.



(c) Frame 3.



(d) Frame 4.

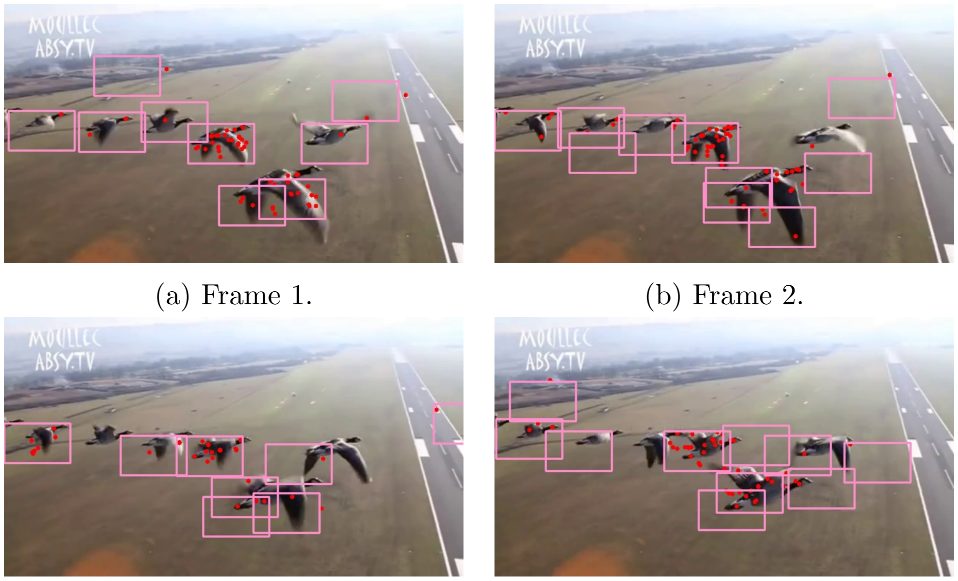
Figure B.8: Object localisations for the first 4 frames of *voc-18-bd-8*.

B.9 VOC-18-BD-9

Ground Truth: $min = 3, max = 7, \mu = 5.54, \sigma = 1.08$

Table B.9: Count estimation for *voc-18-bd-9* using frame descriptors and parameters $minPts = 2, I = 0, O = Y, R = Y$.

Frame #	**Feature Size**	**Selected Features**	**# Clusters**	**Count Estimation**	**Ground Truth**	**minPts**	**Validity**	**Accuracy**	**Duration (s)**
1	222	56	22	9	6	2	4	150	0.0349
2	240	62	21	10	6	2	4	166.667	0.0263
3	247	33	12	8	6	2	4	133.333	0.0204
4	229	50	19	10	6	2	4	166.667	0.0244
5	208	40	12	10	6	2	4	166.667	0.0215



(a) Frame 1. (b) Frame 2.

(c) Frame 3. (d) Frame 4.

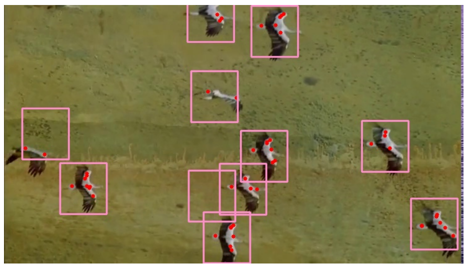
Figure B.9: Object localisations for the first 4 frames of *voc-18-bd-6*.

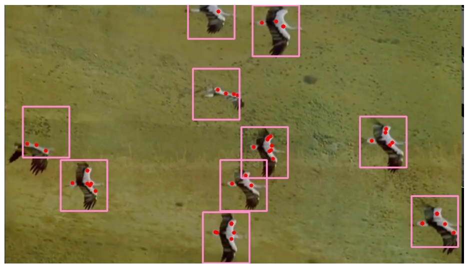
B.10 VOC-18-BD-10

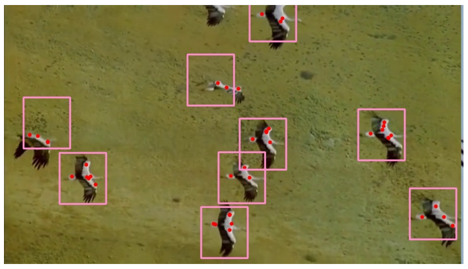
Ground Truth: $min = 9, max = 18, \mu = 13.82, \sigma = 2.96$

Table B.10: Count estimation for *voc-18-bd-10* using combined frame and colour model descriptor results and parameters $minPts = 2, I = 0, O = Y, R = Y$.

Frame #	**Count Estimation**	**Ground Truth**	**Accuracy**	**Duration (s)**
1	11	10	110	0.0757
2	10	10	100	0.0686
3	9	10	90	0.0989
4	10	9	111.11	0.0786
5	10	9	111.1	0.0915


(a) Frame 1.


(b) Frame 2.


(c) Frame 3.

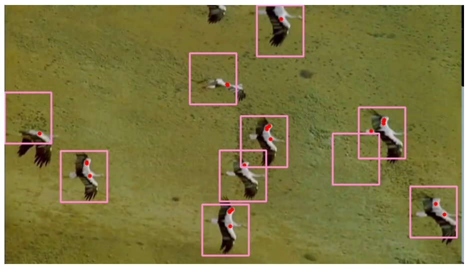

(d) Frame 4.

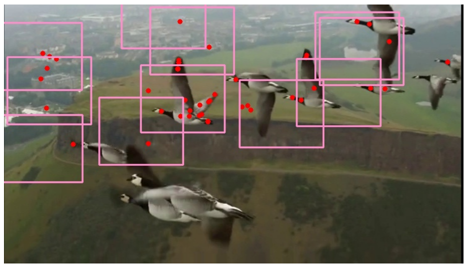
Figure B.10: Combined object localisations for the first 4 frames of *voc-18-bd-10*.

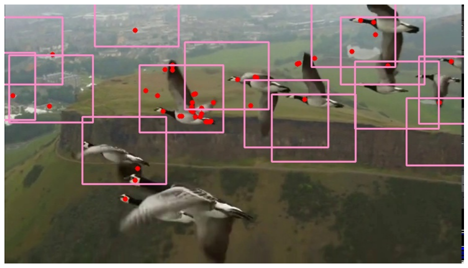
B.11 VOC-18-BD-11

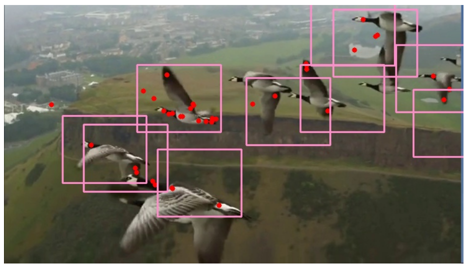
Ground Truth: $min = 4, max = 8, \mu = 5.68, \sigma = 1.21$

Table B.11: Count estimation for *voc-18-bd-11* using frame descriptors and parameters $minPts = 2, I = 0, O = Y, R = N$.

Frame #	**Feature Size**	**Selected Features**	**# Clusters**	**Count Estimation**	**Ground Truth**	**minPts**	**Validity**	**Accuracy**	**Duration (s)**
1	329	45	15	11	8	2	4	137.5	0.0559
2	356	50	17	14	8	2	4	175	0.0508
3	351	40	15	10	8	2	4	125	0.051
4	337	28	10	11	8	2	4	137.5	0.0409
5	330	22	9	9	8	2	4	112.5	0.0361


(a) Frame 1.


(b) Frame 2.


(c) Frame 3.

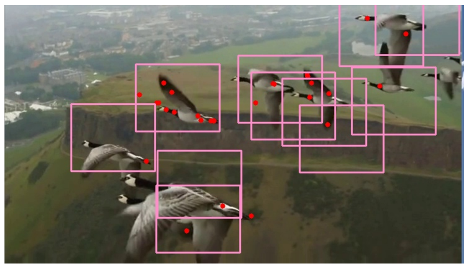

(d) Frame 4.

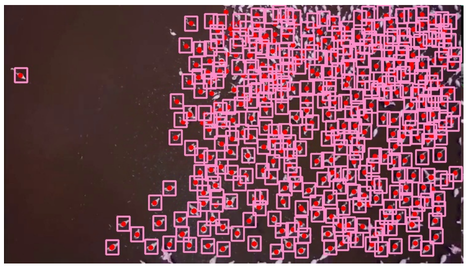
Figure B.11: Object localisations for the first 4 frames of *voc-18-bd-11*.

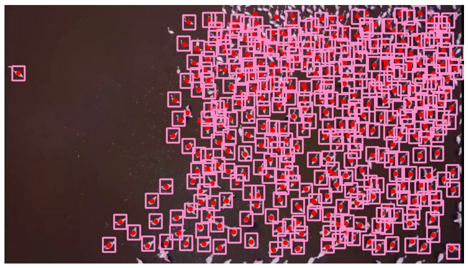
B.12 VOC-18-BD-12

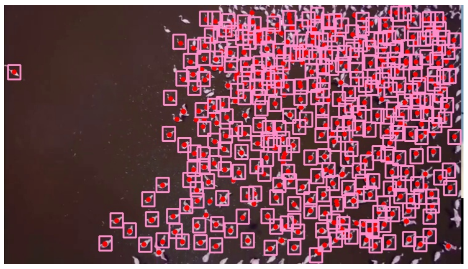
Ground Truth: $min = 211, max = 322, \mu = 296.34, \sigma = 29.63$

Table B.12: Count estimation for *voc-18-bd-12* using combined frame and colour model descriptor results and parameters $minPts = 3, I = 1, O = N, R = N$.

Frame #	**Count Estimation**	**Ground Truth**	**Accuracy**	**Duration (s)**
1	341	287	118.815	0.4756
2	342	286	119.58	0.7537
3	306	288	106.25	0.5158
4	292	288	101.389	1.0002
5	314	289	108.651	0.4381


(a) Frame 1.


(b) Frame 2.


(c) Frame 3.

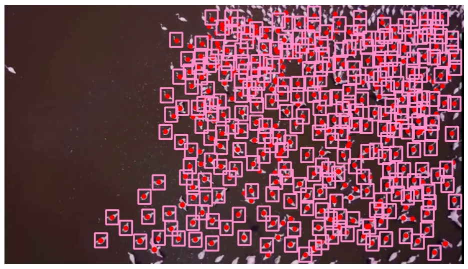

(d) Frame 4.

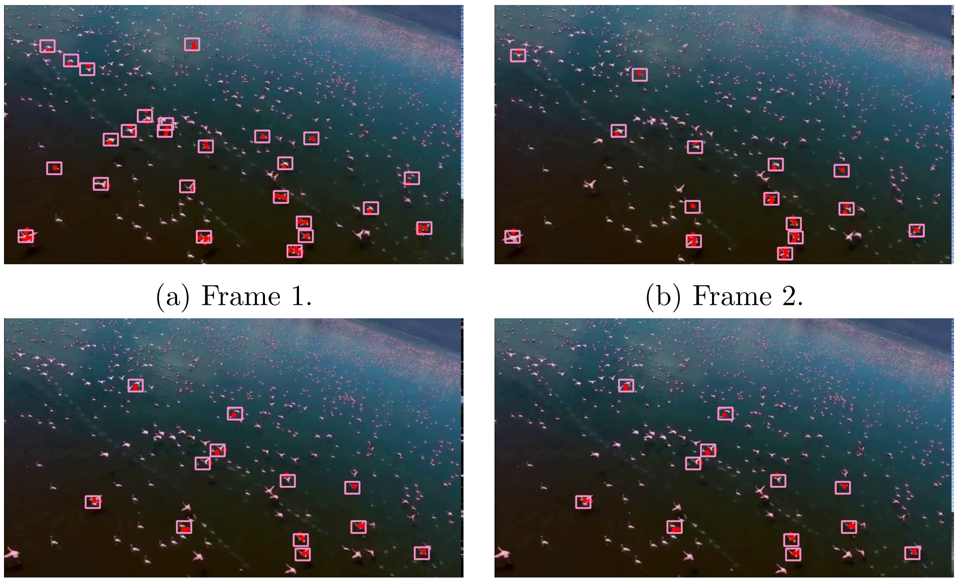
Figure B.12: Combined object localisations for the first 4 frames of *voc-18-bd-12*.

B.13 VOC-18-BD-13

Ground Truth: $min = 47, max = 62, \mu = 51.83, \sigma = 3.89$

Table B.13: Count estimation for *voc-18-bd-13* using frame descriptors and parameters $minPts = 3, I = 1, O = Y, R = Y$.

Frame #	**Feature Size**	**Selected Features**	**# Clusters**	**Count Estimation**	**Ground Truth**	**minPts**	**Validity**	**Accuracy**	**Duration (s)**
1	185	48	13	25	53	2	4	47.17	0.0318
2	188	28	10	15	49	2	4	30.61	0.0254
3	205	21	8	12	49	2	4	24.49	0.0278
4	205	21	8	12	49	2	4	24.49	0.0303
5	210	84	10	51	49	3	4	104.08	0.0168


(a) Frame 1. (b) Frame 2.

(c) Frame 3. (d) Frame 4.

Figure B.13: Object localisations for the first 4 frames of *voc-18-bd-13*.

B.14 VOC-18-BD-15

Ground Truth: $min = 33, max = 62, \mu = 45.54, \sigma = 7.23$

Table B.14: Count estimation for *voc-18-bd-15* using frame descriptors and parameters $minPts = 2, I = 3, O = Y, R = Y$.

Frame #	**Feature Size**	**Selected Features**	**# Clusters**	**Count Estimation**	**Ground Truth**	**minPts**	**Validity**	**Accuracy**	**Duration (s)**
1	300	163	63	46	60	2	4	76.67	0.0347
2	296	183	66	51	62	2	4	82.26	0.0376
3	291	183	61	50	61	2	4	81.97	0.0352
4	268	163	52	52	60	2	4	86.67	0.0369
5	288	201	63	65	58	2	4	112.07	0.0431

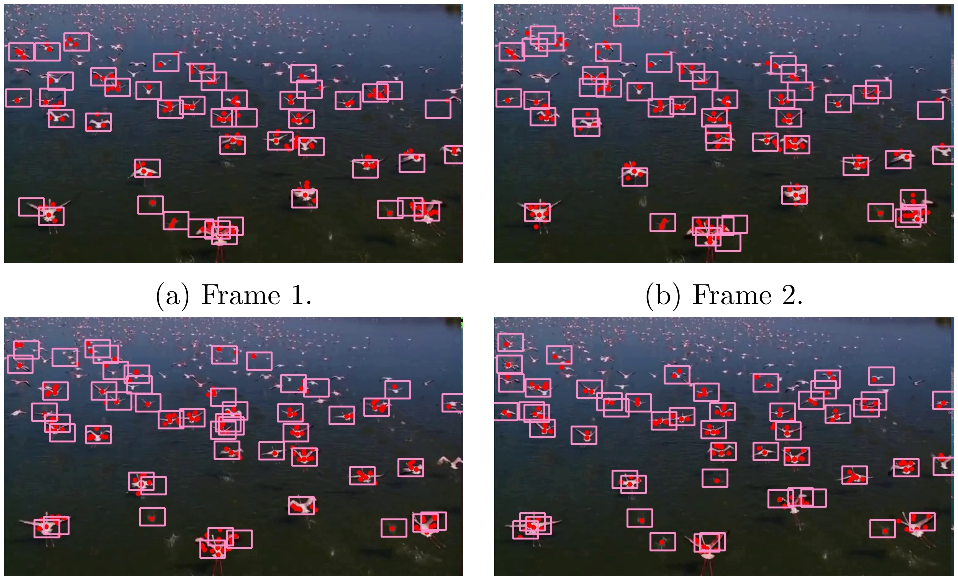

(a) Frame 1. (b) Frame 2.
(c) Frame 3. (d) Frame 4.

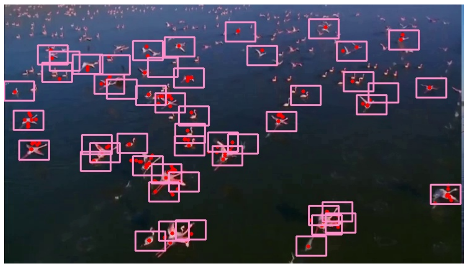
Figure B.14: Object localisations for the first 4 frames of *voc-18-bd-15*.

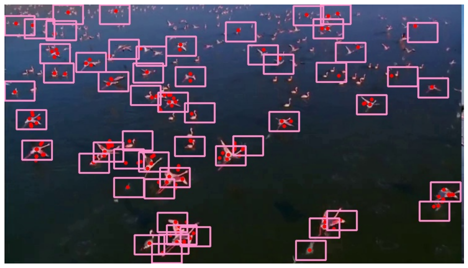
B.15 VOC-18-BD-17

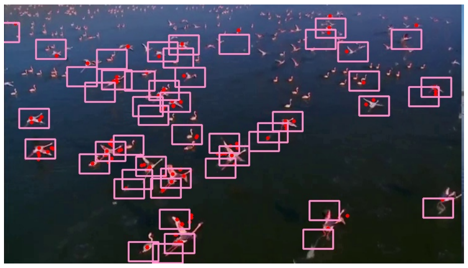
Ground Truth: $min = 106, max = 125, \mu = 116.45, \sigma = 4.47$

Table B.15: Count estimation for *voc-18-bd-17* using frame descriptors and parameters $minPts = 3, I = 1, O = Y, R = Y$.

Frame #	**Feature Size**	**Selected Features**	**# Clusters**	**Count Estimation**	**Ground Truth**	**minPts**	**Validity**	**Accuracy**	**Duration (s)**
1	304	145	23	50	109	3	4	45.87	0.0434
2	293	153	18	55	110	3	4	50	0.0437
3	281	108	15	48	113	3	4	42.48	0.0328
4	269	136	24	40	115	3	4	34.78	0.0365
5	270	129	17	43	114	3	2	37.72	0.0499


(a) Frame 1.


(b) Frame 2.


(c) Frame 3.

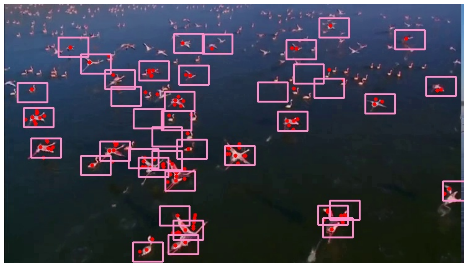

(d) Frame 4.

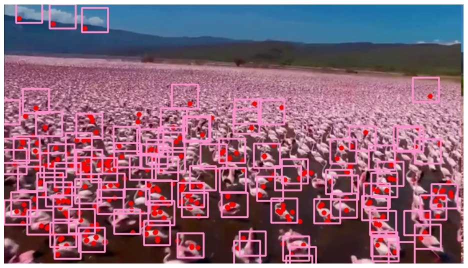
Figure B.15: Object localisations for the first 4 frames of *voc-18-bd-17*.

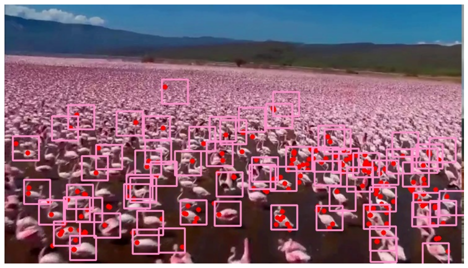
B.16 VOC-18-BD-18

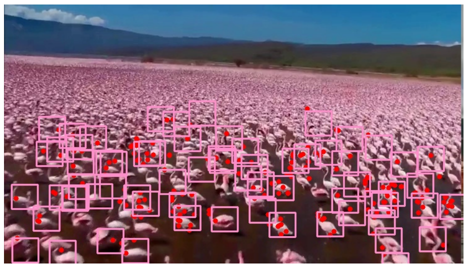
Ground Truth: $min = 87, max = 106, \mu = 97.57, \sigma = 3.81$

Table B.16: Count estimation for *voc-18-bd-18* using frame descriptors and parameters $minPts = 2, I = 1, O = Y, R = Y$.

Frame #	**Feature Size**	**Selected Features**	**# Clusters**	**Count Estimation**	**Ground Truth**	**minPts**	**Validity**	**Accuracy**	**Duration (s)**
1	1826	330	108	92	87	2	4	105.75	0.331
2	1809	207	69	66	94	2	4	70.21	0.301
3	1786	194	62	67	95	2	4	70.53	0.288
4	1812	247	81	77	96	2	4	80.21	0.301
5	1850	173	62	52	97	2	4	53.61	0.312


(a) Frame 1.


(b) Frame 2.


(c) Frame 3.

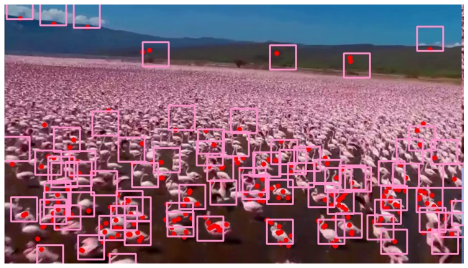

(d) Frame 4.

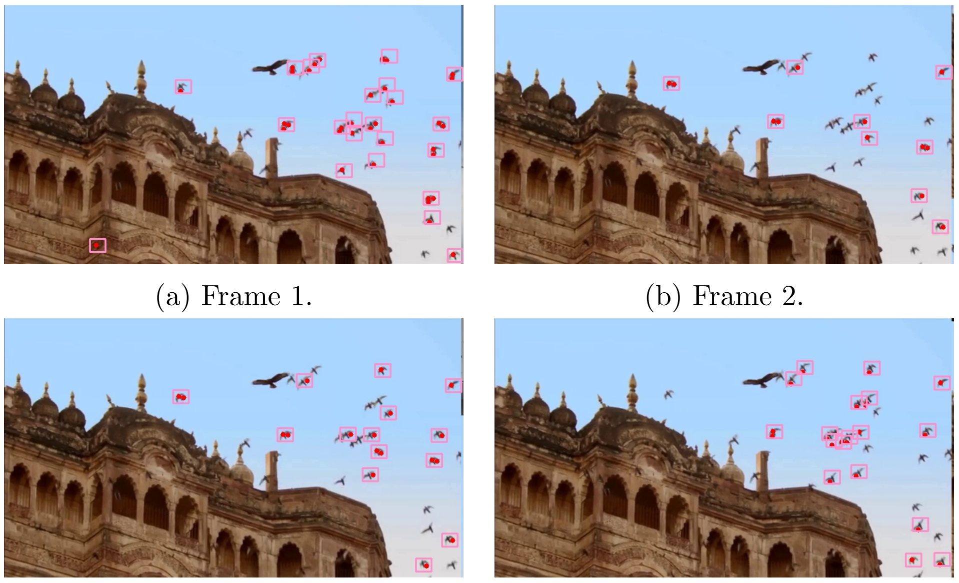
Figure B.16: Object localisations for the first 4 frames of *voc-18-bd-18*.

B.17 VOC-18-BD-19

Ground Truth: $min = 25, max = 37, \mu = 33.31, \sigma = 3.05$

Table B.17: Count estimation for *voc-18-bd-19* using frame descriptors and parameters $minPts = 3, I = 0, O = Y, R = Y$.

Frame #	Feature Size	Selected Features	# Clusters	Count Estimation	Ground Truth	minPts	Validity	Accuracy	Duration (s)
1	853	35	4	23	26	3	4	88.4615	0.0653
2	851	12	4	9	25	3	4	36	0.0604
3	867	18	4	14	25	3	4	56	0.0544
4	862	20	4	18	25	3	4	72	0.0636
5	865	27	4	22	25	3	4	88	0.0627


(a) Frame 1. (b) Frame 2.

(c) Frame 3. (d) Frame 4.

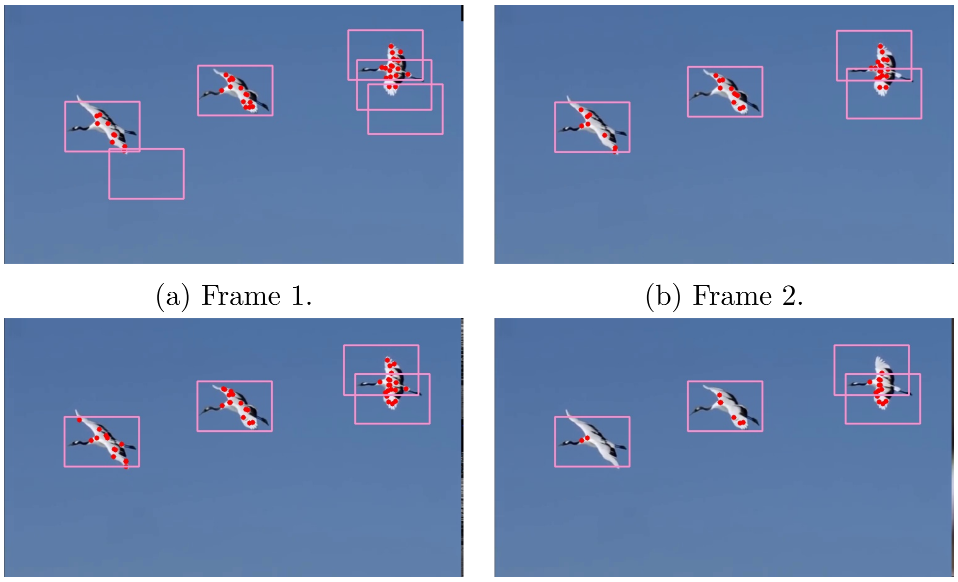
Figure B.17: Object localisations for the first 4 frames of *voc-18-bd-19*.

B.18 VOC-18-BD-20

Ground Truth: $min = 3, max = 3, \mu = 3, \sigma = 0$

Table B.18: Count estimation for *voc-18-bd-20* using combined frame and colour model descriptor results and parameters $minPts = 2, I = 0, O = Y, R = Y$.

Frame #	**Count Estimation**	**Ground Truth**	**Accuracy**	**Duration (s)**
1	6	3	200	0.0645
2	4	3	133.333	0.0501
3	4	3	133.333	0.0543
4	4	3	133.333	0.0458
5	4	3	133.333	0.0495


(a) Frame 1. (b) Frame 2.

(c) Frame 3. (d) Frame 4.

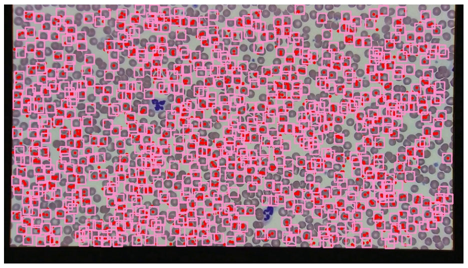
Figure B.18: Object localisations for the first 4 frames of *voc-18-bd-20*.

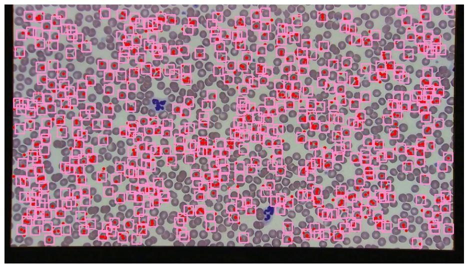
B.19 VOC-18-BL-1

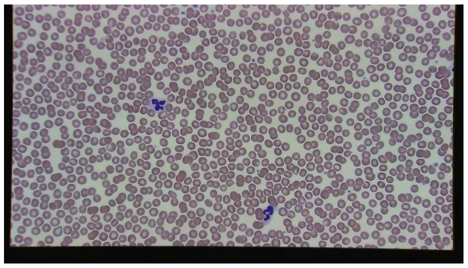
Ground Truth: $min = 698, max = 722, \mu = 698.13, \sigma = 3.36$

Table B.19: Count estimation for *voc-18-bl-1* using frame descriptors and parameters $minPts = 2, I = 5, O = Y, R = Y$.

Frame #	**Feature Size**	**Selected Features**	**# Clusters**	**Count Estimation**	**Ground Truth**	**minPts**	**Validity**	**Accuracy**	**Duration (s)**
1	6631	2052	725	726	722	2	4	100.55	4.412
2	6635	1208	423	494	710	2	4	69.58	4.247
3	6662	0	2	0	706	2	4	0	4.456
4	6655	3025	1095	965	704	2	4	137.07	6.572
5	6661	2	3	2	702	2	4	0.28	7.629


(a) Frame 1.


(b) Frame 2.


(c) Frame 3.

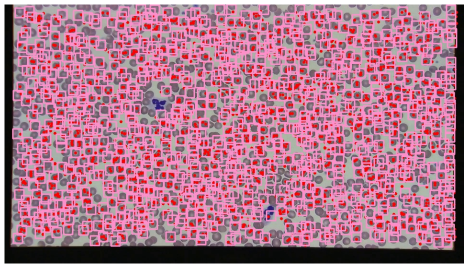

(d) Frame 4.

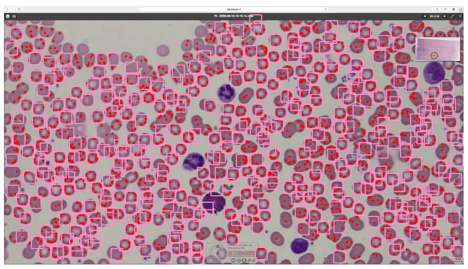
Figure B.19: Object localisations for the first 4 frames of *voc-18-bl-1*.

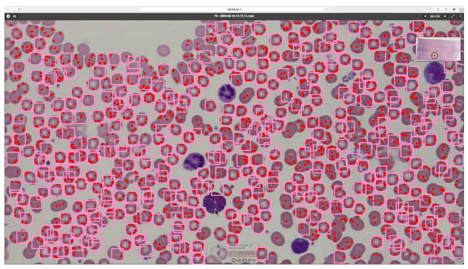
B.20 VOC-18-BL-2

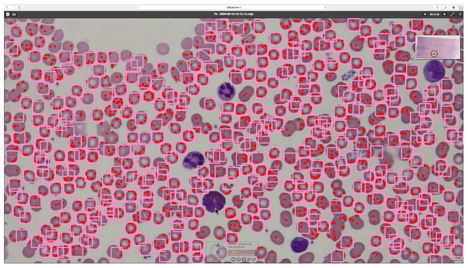
Ground Truth: $min = 460, max = 460, \mu = 460, \sigma = 0$

Table B.20: Count estimation for *voc-18-bd-20* using combined frame and colour model descriptor results and parameters $minPts = 2, I = 2, O = Y, R = Y$.

Frame #	**Count Estimation**	**Ground Truth**	**Accuracy**	**Duration (s)**
1	391	460	85	4.936
2	373	460	81.09	5.371
3	390	460	84.78	7.207
4	428	460	93.04	5.515
5	355	460	77.17	5.101


(a) Frame 1.


(b) Frame 2.


(c) Frame 3.

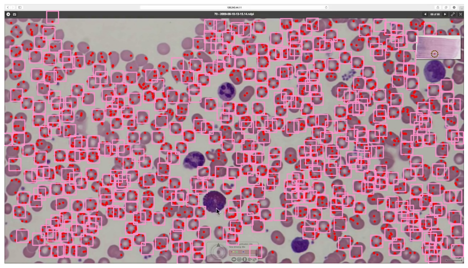

(d) Frame 4.

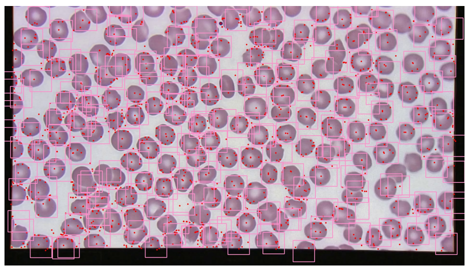
Figure B.20: Object localisations for the first 4 frames of *voc-18-bl-2*.

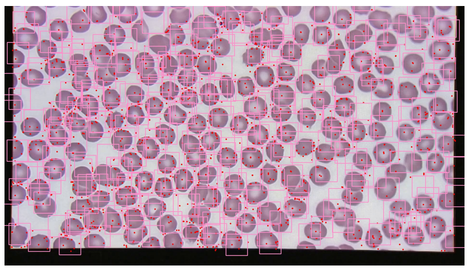
B.21 VOC-18-BL-3

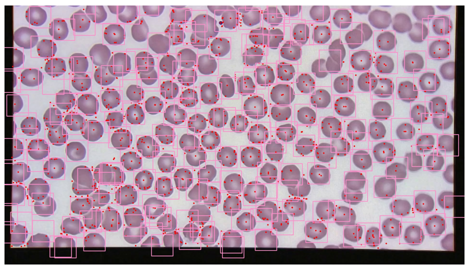
Ground Truth: $min = 176, max = 186, \mu = 184.31, \sigma = 2.05$

Table B.21: Count estimation for *voc-18-bl-1* using frame descriptors and parameters $minPts = 2, I = 2, O = Y, R = Y$ for the first 26 frames where the initial object was successfully tracked.

Frame #	**Feature Size**	**Selected Features**	**# Clusters**	**Count Estimation**	**Ground Truth**	**minPts**	**Validity**	**Accuracy**	**Duration (s)**
1	3256	1942	699	231	179	2	4	129.05	2.336
2	3194	1968	676	300	180	2	4	166.67	2.001
3	3252	1652	574	204	181	2	4	112.71	1.828
4	3224	1828	657	204	182	2	4	112.09	1.919
5	3233	1872	673	246	183	2	4	134.43	2.554


(a) Frame 1.


(b) Frame 2.


(c) Frame 3.

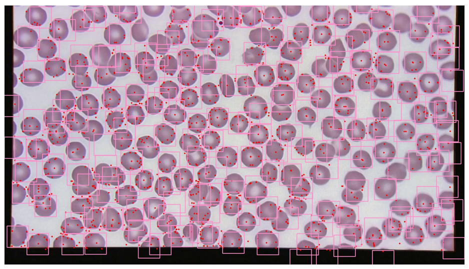

(d) Frame 4.

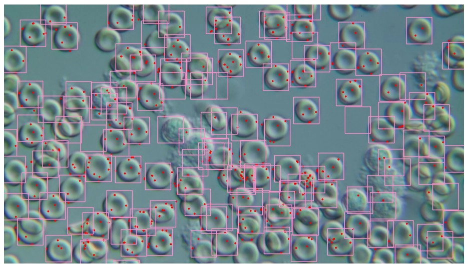
Figure B.21: Object localisations for the first 4 frames of *voc-18-bl-3*.

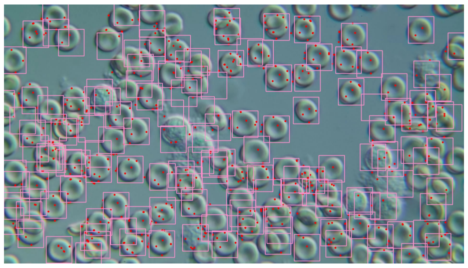
B.22 VOC-18-BL-4

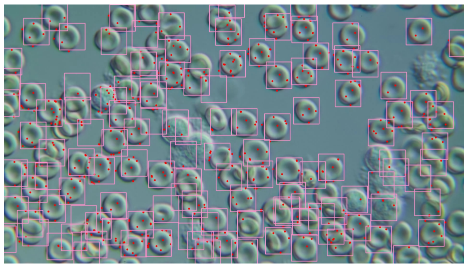
Ground Truth: $min = 115, max = 130, \mu = 122.88, \sigma = 5.13$

Table B.22: Count estimation for *voc-18-bl-1* using frame descriptors and parameters $minPts = 2, I = 2, O = Y, R = Y$.

Frame #	**Feature Size**	**Selected Features**	**# Clusters**	**Count Estimation**	**Ground Truth**	**minPts**	**Validity**	**Accuracy**	**Duration (s)**
1	1639	630	221	135	117	2	4	115.39	0.634
2	1624	776	277	153	120	2	4	127.5	0.689
3	1613	676	242	130	121	2	4	107.44	0.632
4	1650	822	289	139	122	2	4	113.93	0.699
5	1647	726	256	127	122	2	4	104.09	0.708


(a) Frame 1.


(b) Frame 2.


(c) Frame 3.

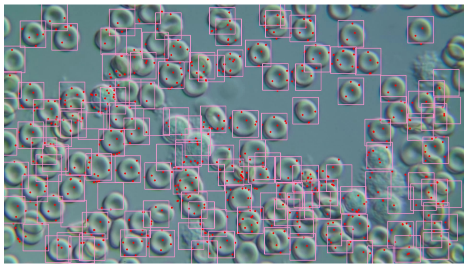

(d) Frame 4.

Figure B.22: Object localisations for the first 4 frames of *voc-18-bl-4*.