Accepted Manuscript

Learn-select-track: An approach to multi-object tracking

Onalenna J. Makhura, John C. Woods

PII: S0923-5965(18)30969-X
DOI: https://doi.org/10.1016/j.image.2019.02.009
Reference: IMAGE 15511

To appear in: *Signal Processing: Image Communication*

Received date : 15 October 2018
Revised date : 20 February 2019
Accepted date : 20 February 2019

Please cite this article as: O.J. Makhura and J.C. Woods, Learn-select-track: An approach to multi-object tracking, *Signal Processing: Image Communication* (2019), https://doi.org/10.1016/j.image.2019.02.009

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

ACCEPTED MANUSCRIPT

Learn-Select-Track: An Approach to Multi-Object Tracking

Onalenna J. Makhura[a,1,*], John C. Woods[a]

[a]*Computer Science and Electronic Engineering, University of Essex, Colchester, United Kingdom*

Abstract

Object tracking algorithms rely on user input to learn the object of interest. In multi-object tracking, this can be a challenge when the user has to provide a lot of locations to track. This paper presents a new approach that reduces the need for user input in multi-tracking. The approach uses density based clustering to analyse the colours in one frame and find the best separation of colours. The colours selected from the detection are learned and used in subsequent frames to track the colours through the video. With this training approach, the user interaction is limited to selecting the colours rather than selecting the multiple location to be tracked. The training algorithm also provides online training even when training on thousands of features.

Keywords: Multi-Object Tracking, Object Colours, density-based clustering, low level local features

1. Introduction

Object tracking is an important and active research field. The research focus has evolved over the years from short term (frame by frame) [1, 2] to long term [3, 4] to multi object tracking(MOT) [5, 6, 7]. This surge in research interest has also seen an increase in datasets dedicated to object tracking such as MOT16 [8] and benchmarks such as BMTT 2016 [9].

Tracking algorithms built around single object tracking such as Kernelized Correlation Filter [2] and online boosting [1] rely on the user input as a region of interest(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 objects 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 cascades [10] are one such algorithm that have gained widespread use. Deep learning approaches [6, 11, 12] have also gained traction for multi-object tracking. 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. The offline nature of these approaches is often due to the time it takes to train and the computational power requirements.

In this paper, we introduce a different approach to tracking multiple objects. We have called the approach 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. The training stage is made up of the learning and selection. For this paper, we use colours as features to be tracked in the videos. While colour is not the most distinctive feature to use, in this paper, we use it to obtain multiple locations

*Corresponding author
Email address: ojmakh@essex.ac.uk (Onalenna J. Makhura)

ACCEPTED MANUSCRIPT

for tracking. In essence, our approach works with any feature that uses distance measure for matching.

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. 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 the find 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. The clustering algorithm 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 the paper is structured as follows; Section 2 discusses the research publications related to this paper, specifically, the recent research into deep learning for multi object tracking, the local feature algorithms and the density clustering algorithms. In section 3, we discuss the training and tracking approach in this paper. Section 4 discusses the results from our tests. Finally we give conclusions and possible future improvements.

2. Related Work

Wu et al.[7] leveraged the power of discriminate correlation filters (DCF) [13] 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.[14] propose an MOT approach that exploits interactions between tracklets. They introduce *close* and *distant* tracklet interactions. *Close interaction* imposes physical constraint on the temporally overlapping tracklets and *distant* 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 reach hundreds.

Other researcher have attempted to address the MOT problem of initialisation in a variety of ways. In [15], 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.*[16] 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.*[17] 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.

This problem is nullified when using offline based learning approaches such as Haar-like features[18] and deep learning techniques. Recent publications such as Zhang et al. [6], Wancun et al.[19] and Chen et al. [11] have demonstrated how convolutional neural networks can provide state of the art performances in multi-object tracking. Deep learning platforms such as YOLO [12], Caffe [20] 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.

ACCEPTED MANUSCRIPT

In this paper, we introduce density based clustering as a way of learning the new object but also to provide a way of bypassing the need for the user to provide a lot of data regarding the object to be learned. Density Based Clustering for Applications with Noise (DBSCAN)[21] is a clustering algorithm that was designed for finding clusters in spatial databases. The algorithm 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 inputs. As such, to use this algorithm properly, one needs to have an insight into the dataset.

Hierarchical DBSCAN (HDBSCAN) was developed by Campello et al. [22] to get get around the need for ϵ. It achieves this by finding all possible DBSCAN solutions 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. Since it only requires *minPts* as a parameter, the algorithm is perfect for exploratory data analysis where the densities of the clusters cannot be determined beforehand.

Frame colours are the feature being tracked in this paper. However, the HDBSCAN computational requirements restrict the size of the dataset. The smallest video frame size in the dataset used in this paper is 640x360, which would results in 230400 data points. The memory and CPU times for distance calculations would be huge. In order to avoid using every pixel we use Speeded-Up Robust Features(SURF) [23] to find significant points in the frame.

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 1. For each square, a Haar wavelet response is calculated in the horizontal (d_y) and the vertical (d_x). The 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. For this paper, we are using the RGB values at the location where the descriptors were found.

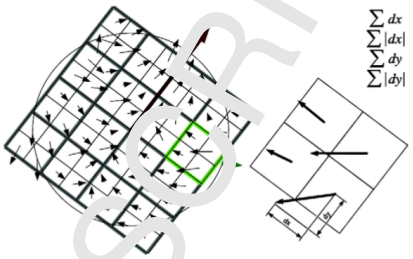

Figure 1: Oriented quadratic grid with 4x4 square sub-regions is laid over the interest point and the $\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|$

3. Learn-Select-Track

This section discusses the design of the Learn-Select-Track algorithm. We first discuss the validity calculation used to determine if a *minPts* value produced good clustering results. Then we 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 $5x5$ kernel to get an average at the feature location.

3.1. Validity Calculations

In one of our previous paper [24], we demonstrated that given a set of local features from a video frame, density based clustering can find clusters that represent very similar features. We applied HDBSCAN to SURF keypoint descriptors to detect which of them were similar. The paper proved that the detection works well even without a specified measure of similarity.

The detected clusters were then analysed statistically for skewness and kurtosis values. These values were calculated for core distances and intra-cluster distances. Using the distances, a percentage based measure of similarity was calculated for each cluster within the results of a particular value of *minPts*.

For this paper, there was a need to find out whether a chosen value of *minPts* produced acceptable results. To find out how good the results are, we use Algorithm 1. With this algorithm, we were able to test the validity of

ACCEPTED MANUSCRIPT

the clustering results to decide whether to use them or try another value of *minPts*.

Algorithm 1: Calculating validity for overall clustering results for a particular *minPts* value.

Data: skew_c - Skewness value from core distances
Data: kurtosis_c - Kurtosis value from core distances
Data: skew_d - Skewness value from intra cluster distances
Data: kurtosis_d - Kurtosis value from intra cluster distances
Result: $validity = 0$
if $skew\_d > 0$ & $kurtosis\_d > 0$ **then**
  $validity \leftarrow validity + 2$
else if $skew\_d < 0$ & $kurtosis\_d > 0$ **then**
  $validity \leftarrow validity + 1$
else
  $validity \leftarrow validity - 1$
end

if $skew\_c > 0$ & $kurtosis\_c > 0$ **then**
  $validity \leftarrow validity + 2$
else if $skew\_c < 0$ & $kurtosis\_c > 0$ **then**
  $validity \leftarrow validity + 1$
else
  $validity \leftarrow validity - 1$
end

3.2. Training

In their paper, [22] explained that in the dendogram, the most prominent clusters survive the longest. Detecting these clusters in colour space gives the best separation of colours in the frame. 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*.

In order to speed up the cluster detection for varying values of *minPts*, our HDBSCAN implementation[1] reuses the calculated distances from the first value to get new core distances which we then use to create a new minimum spanning tree to extract prominent clusters from.

The training begins by extracting a new colour space dataset, C, from the local feature data set, D, and the frame, F. For consistency with OpenCV, the implementation keeps this dataset in a Mat data structure. The resulting dataset is an $n \times m$ matrix where each row contains the BGR values of the pixel. Given the dataset, incrementally varying *minPts* results in three 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. The lowest *minPts* value in that sequence is then treated as the optimum value for colour separation.

[1] https://github.com/ojmakhura/hdbscan.git

ACCEPTED MANUSCRIPT

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*, $C_l = \{c_1^l, ..., c_n^l\}$ and $C_{l-i} = \{c_1^{l-i}, ..., c_m^{l-i}\}$, 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 C_l. This approach develops a $O(n^2)$, where n is the number of points in the dataset.

The approach used in this paper 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 $O(1)$. Worst case scenario, the number of clusters have to be counted each time which results in a $O(m)$ complexity, where m is the number of clusters. Experimentally, 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 cluster-wise 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 C_{l-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 C_{l-1} are brought in to form part of the new cluster in C_l. The inclusion of these points into the new cluster reduces the similarity of points in the new cluster.

3.3. Track and Update

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

HDBSCAN is then applied to D with $minPts = 2 * s_l$ and the data labels, L are extracted for D. L is then split into two such that $L = L_{i-1} \cup L_i$, where L_{i-1} has p labels for the dataset D_{i-1} and L_i has q labels for dataset D_i. We then need to find the new labels for the selected points in S_{i-1}. It is worth noting that if any of the points now has a noise label, the point is ignored. Using the new S_{i-1} labels, we then find all the points in D_i that have the same labels using L_i. The new selected model 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 paper 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 L_{i-1} and L_i. If we only used $minPts = s_l$, the resulting clusters could be such that each of L_{i-1} and 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 S_{i-1} would not need to change, but the combined dataset would have to change such that $D = D_{i-y-1} \cup D_{i-y-2} \cup ... \cup D_i$.

3.4. Testing Approach

The common datasets available were unsuitable for testing in this paper for three reason. The first reason is that they are predominantly about people and general urban settings. Since people have different skin colours and wear different coloured clothes, the datasets are not suitable for testing the approach in this paper. The second reason is that their benchmarks often require the localisation of the objects during tracking. However, the approach described in this paper uses colour from pixels which do not provide any information on the size or shape of the objects. The final reason is the lack of videos with really high number of objects.

ACCEPTED MANUSCRIPT

To test our approach, we used an object counting dataset VOC-18[2]. While it was designed for object counting, the videos contain a variety of scenes with birds (*voc-18-bd-{1-22}*) 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.

In order to use the dataset we devised simple benchmarks for testing the effectiveness of our approach. 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. As simple as these metrics are, they provide a very powerful measure of checking whether the train-select model of training and the track and update approaches work.

4. Results and Discussions

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 detect 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 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 time it took to learn the colour model as well as how long it took to track the features from frame to frame.

4.1. 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 value 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.

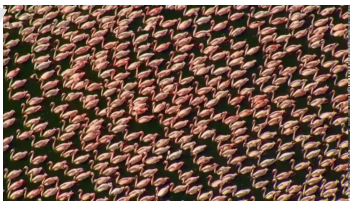

Figure 2: The training frame from *voc-18-bd-1* video.

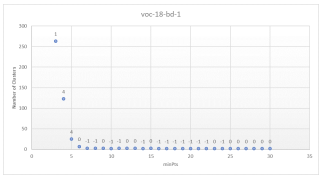

Figure 3: The training results for *voc-18-bd-1* video. The numbers on top of the plot points shows the validity of the results at that *minPts* value.

In Figure 3, 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 2 showing flamingoes sitting on water and Figure 4 which shows how the colours of the frame were clustered.

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

ACCEPTED MANUSCRIPT

Table 1: The training results with VOC-18 dataset. The number of points column shows the number of colour points detected in the first frame and duration column shows the amount of time it took to analyse the colours by the training algorithm.

Video	Points	min--Pts	Clus--ters	Time
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	156	7	2	0.0972
voc-18-bd-10	222	7	5	0.18[illegible]
voc-18-bd-11	219	15	4	0.1571
voc-18-bd-12	329	5	11	0.2711
voc-18-bd-13	1934	10	3	[illegible]
voc-18-bd-14	193	6	4	0.160[illegible]
voc-18-bd-15	185	6	2	0.1395
voc-18-bd-16	308	3	17	0.[illegible]
voc-18-bd-17	300	4	[illegible]	0.3416
voc-18-bd-18	157	6	[illegible]	0.0673
voc-18-bd-19	304	3	24	0.1575
voc-18-bd-20	1826	3	151	4.6714
voc-18-bd-21	853	6	4	1.0212
voc-18-bd-22	129	14	2	0.0621
voc-18-bl-1	6631	13	2	37.4817
voc-18-bl-2	6010	25	5	31.601
voc-18-bl-3	325[illegible]	15	4	10.1651
voc-18-bl-4	1[illegible]9	8	3	3.8409

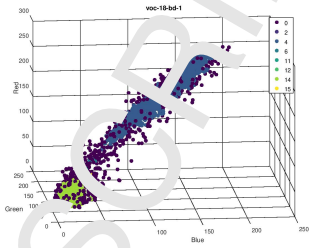

Figure 4: A scatter plot of *voc-18-bd-1* colours and the results of the clusters of the selected value of $minPts = 6$.

While only 2 clusters were detected for the majority of the *minPts* values, there was intertwining 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 4) 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. In the test video *voc-18-bd-20* (Figure 5), the number of clusters being discovered was irregular. This resulted in $minPts = 3$ selection, which had 151 clusters (See Figures 6 and 7). This presents a challenge for the part of the training algorithm that requires a person to select the colours of interest.

Figure 5: The training frame from *voc-18-bd-20* video.

However, the validity of the results showed that all the values of *minPts* produced results with validities greater than 0. In fact, for $minPts = 30$, the two clusters produced, $\{2, 3\}$, had core distance confidences of 73% and 61%, and intra-cluster distance confidences of 70% and 71% respectively. The 3D scatter graph of the colours in

ACCEPTED MANUSCRIPT

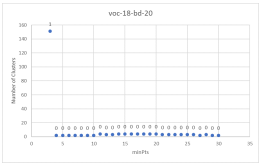

Figure 6: The training results for *voc-18-bd-20* video. For this video, the *minPts* selected was 3 which had 151 clusters.

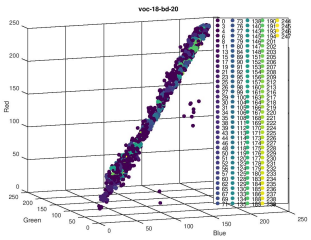

Figure 7: 3D scatter graph of *voc-18-bd-20* video's first frame.

the frame shows the grouping.

The effect of using validity to control the quality of *minPts* can be seen in video *voc-18-bd-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 8 and 9). Furthermore, some of the birds' colours are affected by the glare making it difficult for clear separation of colours.

Figure 8: The training frame from *voc-18-bd-3* video.

The 3D scatter (Figure 10) shows that a good argument can be made for 4 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,

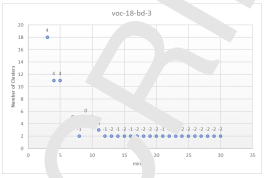

Figure 9: The training results for *voc-18-bd-3* video. For this video, the *minPts* selected was 4 which had 11 clusters.

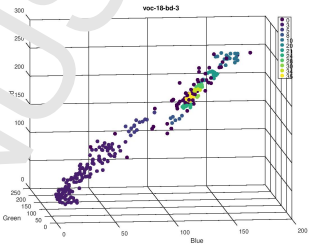

Figure 10: 3D scatter plot of the first frame of *voc-18-bd-3* video.

cluster 4 was the one chosen to represent the colours of the objects of interest.

The overall results shown in Table 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 three of the videos (*voc-18-bd-9, voc-18-bd-20, voc-18-bl-1*), the provided clusters did not offer a good selection of colours as such there were no colours learnt. In *voc-18-bd-9* (Figure 11) and *voc-18-bl-1* the detected clusters did not offer a choice of clear objects of interest colours. In *voc-18-bl-1*, the number of clusters for the chosen *minPts* was just too many for an attempt at choosing the colours.

4.2. Frame by Frame Colour Tracking

The tracking results are shown in Table 2. The table does not have results for videos *voc-18-bd-9, voc-18-bd-20* 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

ACCEPTED MANUSCRIPT

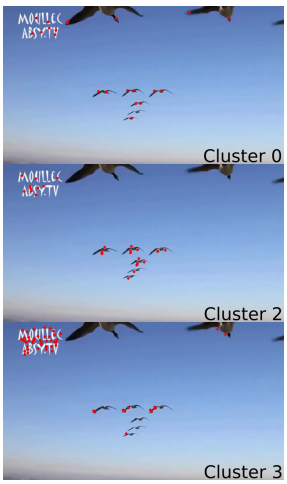

Figure 11: Options for colour selection *voc-18-bd-9_selection* video. With selected *minPts* = 7, the colour points that are actually on the birds have been labelled as the noise.

failure (colour model lost in less than 10 frames), a[illegible] successful tracking.

In the 23 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 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 loose 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-ba-13* (Figure 12) 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. The rest of the videos showed varying degrees of successful tracking.

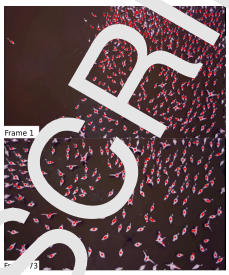

Figure 12: The tracker results for *voc-18-bd-13* video showing the first and the last frame.

The effect of user selection of the colours during the algorithm training can be seen in *voc-18-bl-2* video. Figure 13 and 14 depict clusters 5 and 6 that the training algorithm offered as some of the possible choices for the colour model. Cluster 5 points (Figure 13) are on the darker side of the blood cells while cluster 6 points (Figure 14) 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.

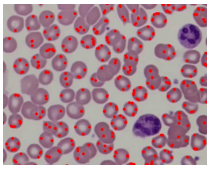

Figure 13: *voc-18-bl-2* video cluster 5. The points are on the edge of the cells where the colours are more distinct.

The background/foreground similarity problem observed in *voc-18-bd-3* video during training was also encountered during tracking. Selecting all clusters that are on the birds results in the tracking algorithm losing the colour model by frame 14. However, if only the clusters on the dark parts of some of the birds are selected, the tracking algorithm did not loose the colour model until the end of the video.

ACCEPTED MANUSCRIPT

Table 2: The results of tracking the colours based on the selected colours.

Video	Length	Tracker Lost at:
voc-18-bd-1	77	67
voc-18-bd-2	71	71
voc-18-bd-3	103	103
voc-18-bd-4	65	41
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-10	143	101
voc-18-bd-11	121	121
voc-18-bd-12	155	2
voc-18-bd-13	73	73
voc-18-bd-14	105	105
voc-18-bd-15	99	22
voc-18-bd-16	110	5
voc-18-bd-17	115	115
voc-18-bd-18	75	3
voc-18-bd-19	85	53
voc-18-bd-21	117	117
voc-18-bd-22	93	38
voc-18-bl-2	94	94
voc-18-bl-3	80	6
voc-18-bl-4	49	4

Table 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-12	12	3	29	215
voc-18-bd-16	7	2	16	30
voc-18-bd-18	5	3	24	42
voc-18-bl-[illegible]	5	2	1291	2318

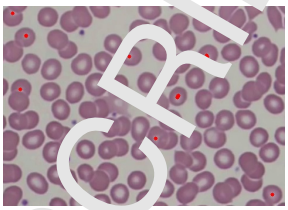

Figure 14: *voc-18-bl-2* video cluster 6. The points are in the middle of the blood cells which have a lot similar in colour to the background.

4.3. Training and Tracking Times

In this section we discuss the training and tracking durations from our approach. The data shown in the figures and the tables is for all the videos even for those where the training or tracking failed. The argument for this is that since the algorithm processes all the points in both failure and success, the information on the time it takes is still worth investigating for this paper. Our implementation takes advantage of the computational resources available by using OpenMP[25] to parallelise some parts and improve speed. We tested the our approach on a Dell XPS laptop with a 7^{th} generation *i7* and 16GB RAM.

The bulk of HDBSCAN's complexity is dominated by the distance calculation.The distance matrix is symmetric with 0 on the principal diagonal such that 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_x , then the new distance matrix as a vector will have length $n_d = n_x(n_x - 1)/2$. 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 $O(n^2)$. The colour model training times can be seen in Table 1 while the complexity has been highlighted in Figure 15.

Table 2 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,

ACCEPTED MANUSCRIPT

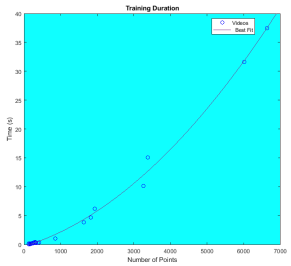

Figure 15: The graph of number of points vs time for learning the object colour model on the first frame.

the algorithm is processing around twice the number of points. The tracking algorithm still relies on HDBSCAN which mean it is also a $O(n^2)$ as can be seen in Figure 16.

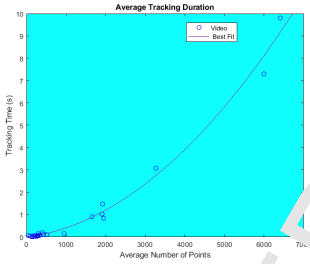

Figure 16: The graph of average number of points per frame vs the tracking duration.

5. Conclusion

In this paper 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 26 videos with varying degrees of success. While the paper uses colours for tracking, this approach can work with other features that are matched using euclidean distance.

Table 4: The average number of points 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_22	106	0.03249336
voc_18_bd_18	144	0.00843991
voc_18_bd_14	152	0.03234725
voc_18_bd_9	172	0.01150438
voc_18_bd_5	193	0.05855851
voc_18_bd_15	199	0.05959434
voc_18_bd_7	218	0.03284515
voc_18_bd_19	256	0.03664438
voc_18_bd_17	270	0.03493461
voc_18_bd_4	271	0.06568578
voc_18_bd_16	290	0.05997002
voc_18_bd_6	293	0.15951065
voc_18_bd_10	315	0.06294197
voc_18_bd_11	359	0.07135481
voc_18_bd_12	401	0.18854269
voc_18_bd_8	426	0.09738525
voc_18_bd_3	509	0.10776893
voc_18_bd_21	951	0.15017254
voc_18_bl_4	1659	0.91024479
voc_18_bl_3	1912	1.03131769
voc_18_bd_13	1918	1.47564444
voc_18_bl_20	1950	0.83814666
voc_18_bd_1	3271	3.08022

ACCEPTED MANUSCRIPT

Out of the 26 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 video 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 and this paper 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 colour 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 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. An improvement to the tracking algorithm could include increasing strictness in accepting tracking results. This can be achieved by thresholding the validity at which results are acceptable.

References

[1] H. Grabner, M. Grabner, H. Bischof, Real-time tracking via on-line boosting (2006) 6–11.

[2] J. F. Henriques, R. Caseiro, P. Martins, J. Batista, High-speed tracking with kernelized correlation filters, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (3) (2015) 583–596. doi:10.1109/TPAMI.2014.2345390.

[3] N. Wang, W. Zhou, H. Li, Reliable re-detection for long-term tracking, IEEE Transactions on Circuits and Systems for Video Technology (2018) 1–1doi:10.1109/TCSVT.2018.2816570.

[4] W. Kang, X. Li, S. L[illegible] G. Liu, Corrected continuous correlation filter for long-term tracking, IEEE Access 6 (2018) 11959–11969. doi:10.1109/ACCESS.2018.2810382.

[5] J. H. Yoon, M.-H. Yang, J. Lim, K.-J. Yoon, Bayesian multi-object tracking using motion context from multiple objects, in: Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on, 2015, pp. 33–40. doi:10.1109/WACV.2015.12.

[6] Y. Zhang, Y. Tang, B. Fang, Z. Shang, Fast multi-object tracking using convolutional neural networks with tracklets updating, in: 2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), 2017, pp. 313–317. doi:10.1109/SPAC.2017.8304296.

[7] C. Wu, H. Sun, H. Wang, F. Kun, G. Xu, W. Zhang, X. Sun, Online multi-object tracking via combining discriminative correlation filters with making decision, IEEE Access (2018) 1–1doi:10.1109/ACCESS.2018.2858853.

[8] A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, K. Schindler, MOT16: A benchmark for multi-object tracking, CoRR abs/1603.00831. arXiv:1603.00831.
URL `http://arxiv.org/abs/1603.00831`

[9] R. B. Girshick, P. F. Felzenszwalb, D. McAllester, Benchmarking multi-target tracking: Motchallenge 2016, https://motchallenge.net/workshops/bmtt2016/.

[10] P. Viola, 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, 2001, pp. I–511–I–518 vol.1. doi:10.1109/CVPR.2001.990517.

ACCEPTED MANUSCRIPT

[11] L. Chen, H. Ai, C. Shang, Z. Zhuang, B. Bai, Online multi-object tracking with convolutional neural networks, in: 2017 IEEE International Conference on Image Processing (ICIP), 2017, pp. 645–649. doi:10.1109/ICIP.2017.8296360.

[12] J. Redmon, A. Farhadi, Yolo9000: Better, faster, stronger, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6517–6525. doi:10.1109/CVPR.2017.690.

[13] D. S. Bolme, J. R. Beveridge, B. A. Draper, Y. M. Lui, Visual object tracking using adaptive correlation filters, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010, pp. 2544–2550. doi:10.1109/CVPR.2010.5539960.

[14] L. Lan, X. Wang, S. Zhang, D. Tao, W. Gao, T. S. Huang, Interacting tracklets for multi-object tracking, IEEE Transactions on Image Processing 27 (9) (2018) 4585–4597. doi:10.1109/TIP.2018.2843129.

[15] E. Türetken, X. Wang, C. J. Becker, C. Haubold, P. Fua, Network flow integer programming to track elliptical cells in time-lapse sequences, IEEE Transactions on Medical Imaging 36 (4) (2017) 942–951. doi:10.1109/TMI.2016.2640859.

[16] X. Wang, B. Fan, S. Chang, Z. Wang, X. Liu, D. Tao, T. S. Huang, Greedy batch-based minimum-cost flows for tracking multiple objects, IEEE Transactions on Image Processing 26 (10) (2017) 4765–4776. doi:10.1109/TIP.2017.2723239.

[17] A. Maksai, X. Wang, F. Fleuret, P. Fua, Non-markovian globally consistent multi-object tracking, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2563–2573. doi:10.1109/ICCV.2017.278.

[18] R. Lienhart, J. Maydt, An extended set of haar-like features for rapid object detection, in: Proceedings. International Conference on Image Processing, Vol. 1, 2002, pp. I–900–I–903 vol.1. doi:10.1109/ICIP.2002.1038171.

[19] L. Wencun, Z. Wenyan, Z. Liguo, Z. Xiaolin, L. Jiafu, Multi-scale behavior learning for multi-object tracking, in: 2017 First International Conference on Electronics Instrumentation Information Systems (EIIS), 2017, pp. 1–5. doi:10.1109/EIIS.2017.8298683.

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

[21] J. Sander, M. Ester, H.-P. Kriegel, X. Xu, Density-based clustering in spatial databases: The algorithm gdbscan and its applications, Data Min. Knowl. Discov. 2 (2) (1998) 169–194. doi:10.1023/A:1009745219419.
URL http://dx.doi.org/10.1023/A:1009745219419

[22] R. J. G. B. Campello, D. Moulavi, A. Zimek, J. Sander, Hierarchical density estimates for data clustering, visualization, and outlier detection, ACM Trans. Knowl. Discov. Data 10 (1) (2015) 5:1–5:51. doi:10.1145/2733381.
URL http://doi.acm.org/10.1145/2733381

[23] H. Bay, A. Ess, T. Tuytelaars, L. V. Gool, Speeded-up robust features (surf), Computer Vision and Image Understanding 110 (3) (2008) 346–359, similarity Matching in Computer Vision and Multimedia. doi:10.1016/j.cviu.2007.09.014.
URL http://www.sciencedirect.com/science/article/pii/S1077314207001555

[24] O. Makhura, J. Woods, Multiple feature instance detection with density based clustering, in: 2018 IEEE International Conference on Consumer Electronics (ICCE) (2018 ICCE), Las Vegas, USA, 2018.

[25] L. Dagum, R. Menon, Openmp: An industry-standard api for shared-memory programming, IEEE Comput. Sci. Eng. 5 (1) (1998) 46–55. doi:10.1109/99.660313.
URL http://dx.doi.org/10.1109/99.660313

ACCEPTED MANUSCRIPT

Highlights

- Density based clustering for learning object features to track.
- Density based clustering for tracking objects in videos.
- Training and tacking without knowing the number of objects.
- Performance of the tacking and training algorithms.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Conflict of Interest

I, Onalenna Junior Makhura, declare that there is no conflict of interest for this paper.

ACCEPTED MANUSCRIPT