Histogram-based object tracking

Mohammad Mohaiminul Islam and Zahid Hassan Tushar

I. Introduction

Object tracking is a very challenging task in video-surveillance systems. Among multiple approaches and methods, this report is solely focused on tracking a single object based on histograms. In this assignment we explored color-based and gradient-based tracking at first. Generally both models have their strengths and weaknesses depending on the application scenario. Eventually we combined those methods to get the best of both worlds. We tested our models on the dataset provided by the lab and attempted to overcome challenges associated with each video sequence. Further we evaluated the performance of each tracking algorithm individually and combined.

II. Data

To develop and test our tracking algorithm, the lab provided us with a dataset. The dataset is composed of nine video sequences drawn from the VOT challenge dataset [3]. We used the *"bolt1"* video sequence to test our both tracking modules. The challenge in this video is that there exists another object with the same color information and also the area of the object varies from frame to frame. Luckily normalized histograms are capable of handling these issues. After development of the tracking modules, we used the *"sphere"* and *"car1"* sequence to evaluate color-based tracking module while gradient-based tracking was evaluated using *"basketball"* and *"ball2"* sequences. We evaluated the fused module using the *"bag"*, *"ball"* and *"road"* video sequences. All the video sequences offer a range of tracking challenges ranging from deformation, change in color information, partial occlusion and sudden motion.

III. Implementation and Discussion

Our object tracking module can be divided into two main subtasks: Candidate extraction and Feature representation and analysis. Candidate extraction is an open issue. For our case we have explored the research work from [1] and adapted it into our module. Feature extraction and analysis is carried with the idea from [2] and incorporated in our work.

A. Candidate Extraction

Candidate extraction means generating probable patches to match our target object model. It is very crucial in the performance of the tracking algorithm. An optimal candidate generation method increases the probability of tracking while a bad candidate method can lead to tracking false targets. Hence we used the idea from [1] to define our candidate extraction method. In our module, we defined each candidate with a fixed area of height and width which is slightly greater than the target module. It increases the robustness of our candidate against a slight variation. We generated candidates using the grid search approach. For the whole process we focused only on the object center. From the target model, we extract its central location. We used this center location as the center of our search grid. Based on the number of candidate parameters, we define a squared area keeping the center at the target object's center. We considered the stride parameter to jump from one candidate's center to another candidate's center position. In case of increase in the stride, the search area would increase proportionately. The higher search area happens to lead to a better performance for sure, but it also means increased computational cost. Therefore, there is a trade-off between the number of candidates we generate and the response time we need.

B. Feature Representation and Analysis

Object modeling is done using different kinds of features. For our work, we used color-based histograms and gradient-based histograms. In the first section, we implemented a color-based model, a gradient-based model for the second section and finally, in the third section we ran a fusion of those two models and analyzed the tracking performance for different challenging situations. Each section is composed of three sub-sections: methods, methodology, and discussion. In method part, we addressed the theoretical study while in the methodology we inscribed the implementation details. The discussion section is dedicated in analysis of the results.

1) Section –I : Color-based Tracking

a) Method

When the object model does not contain much background information, the color information of the model tends to remain fixed for the whole duration of the video sequence. Therefore we can use the color information from the RGB and HSV color space and grayscale value as features. The method starts with separating the individual color channels. We pick one from the pool of six options: grayscale value, R channel, G channel, B channel, H channel, and S channel. Then we model our object with a histogram generated using that particular feature. After that the histogram is normalized to make the model robust against variation in illumination. However, the H channel and S channel are inherently immune to illumination change. Once we get the candidates, we generate histograms for them as well using the same feature. We need to keep in mind that the object histogram and the candidate histogram must have the same number of bins. Otherwise the similarity function will fail. To compare the similarity between a candidate and the target model we used the *Bhattacharyya coefficient*. The object model is given in (1) where $\vec{q}$ represents the normalized histogram with N_b bins. In (2) we can see the candidate model, where y denotes the y^{th} candidate with the same number of bins as object

model in the histogram. In both models *u* denotes the u^{th} bin of the histogram. Finally in (3) the *F(y)* function expresses the *Bhattacharyya coefficient* for the y^{th} candidate.

$$\vec{q} = \frac{1}{\sum q_u}\left(q_1, q_2, \ldots\ldots, q_{N_b}\right) \quad (1)$$

$$\vec{p}(y) = \frac{1}{\sum p_u(y)}\left(p_1(y), p_2(y), \ldots\ldots, p_{N_b}(y)\right) \quad (2)$$

$$F(y) = \sum_{u=1}^{N_b} \sqrt{p_u(y) \cdot q_u} \quad (3)$$

The candidate with the highest coefficient score is the most likely candidate. We take the center coordinate of this candidate, set it as the center coordinate of the search area for our next frame and continue from the candidate extraction method again until the last frame in the sequence is processed. One thing to note, from the first frame using the ground truth, the target model is initialized, after that it is not changed during the whole tracking procedure.

b) Methodology

We start our module by specifying our target model. We use the ground truth information from the 1st frame to define our model. In the consecutive frame, the location of the target model will vary and change from frame to frame but the model information, that is the histogram representing the target model, remains unchanged during the whole video sequence. We use a candidate extraction method named *generateCandidates*() to generate candidates for our tracking. We have to set the number of candidates to be generated as a parameter. This class gives us a rectangle for each candidate. This rectangle is used to crop the desired color information from our current frame. Also the method requires some parameters as stride, top left corner coordinate of the target, and the current frame.

Next, we set parameters to choose from the range of color features we will use to define our model and candidates. The patch goes through a color conversion based on the color feature we want to explore. We extract the desired color information and create a histogram based on that information. This computation is done using one of the methods; *calculateChannelH(), calculateChannelS(), calculateChannelR(), calculateChannelG(), calculateChannelB()* and *calculateChannelGray().* Obviously the number of bins used to create the histogram, has to be set beforehand and remains invariant during the whole analysis. Finally, we used the Bhattacharyya distance function [5] from the OpenCV library as our similarity function. Hence instead of taking the candidate with the maximum score from the similarity function, we take the one with the minimum score since minimum score means the lowest distance between the candidate and the target model. This is implemented in the *computeDistance()* method. The whole algorithm is composed of a single method called *track()* which is needed to be called to run this color-based tracking. All the methods are included within this one. We use *estimateTrackingPerformance()* to estimate the performance of our tracking algorithm.

c) Discussion

To test our color based tracking model we are provided some real video sequences from VOT challenge [3]. First we will try to find the key challenge in each video sequence and then we apply our model and compare between different color channels based on the tracker that we have implemented. Finally we tune the hyper-parameters to get the best performance.

At first after developing the color module we have tested our model on the test sequence *bolt1*. For all the color channels our model was able track successfully upto 150 frames. But for G and H color channels it is able to track upto 250 frames successfully then it loses track for some frames and picks it up again. Following Table. I shows the comparison of tracking performance of different color channels. From scores we can see that the performance for H channels is highest whereas the G channels stood in the second position. But both G and H channels are significantly higher than other color channels in terms of performance. But if we consider both processing time and performance, our G channel based tracking provides an optimal result.

TABLE I. COMPARISON OF COLOR CHANNELS PERFORMANCE ON VIDEO SEQUENCE "BOLT1"

Channels	Parameters	Bounding Box	Average Processing Time per frame (millisecond/frame)	Score
B	#Candidate = 81 #Bin = 16 Stride = 1	Height = 60 Width = 30	1.272	0.165
G			1.301	0.302
R			1.336	0.172
H			3.240	0.499
S			3.311	0.198
Gray			1.460	0.150

Now we will use our model on the real data. Our first real test sequence is *"sphere"* on which we can see that one guy is holding a transparent sphere in his hand and moving it around. The challenges of this sequence is that the color information of the sphere changes frame to frame. This is because the sphere is transparent and it is having the color information according to its background. Also sometimes the hand of the person is at the front side instead of left and right as we can see in the Fig. 1. This change of color information can harm our model as we are using a static model extracted from the 1st frame. Another challenge is that the relative size of the sphere changes also due to the viewing position of the camera.

In the next video sequence *"car1"*, we are trying to track a running car. In this video sequence we believe the main challenge is camera movement, where for the entire video sequence due to the rapid and arbitrary movement tracking process gets difficult. For this rapid and arbitrary movement position of the object between consecutive frames are not consistent as a result our model sometimes fails to track efficiently.

Fig. 1. Frame # 194 from the video sequence 'sphere'

A brief comparison on both video sequences has been shown on Table. II and Table. III. We have taken two top performed color channels from the results of Table. I and also the worst performed color channels in order to understand and have a more robust view of our different flavor of our model performing on real world data. We have explored how these three different color channel based tracking are doing with the conjunction of variable candidate numbers.

TABLE II. COMPARISON OF COLOR MODULE PERFORMANCE ON VIDEO SEQUENCE "SPHERE"

Channels	Parameters	Average Processing Time per frame (millisecond/frame)	Score
G	#candidate = 49	2.682	0.484
	#candidate = 100	5.621	0.575
	#candidate = 400	15.946	0.590
H	#candidate = 49	7.039	0.251
	#candidate = 100	12.091	0.154
	#candidate = 400	33.998	0.124
Gray	#candidate = 49	2.880	0.483
	#candidate = 100	5.730	0.531
	#candidate = 400	16.430	0.552

**Note: For all the instance in the above table following parameters are same patch height =110 & width=110 , #bin=16 & stride=3.

In the case of the video sequence *"bolt1"* even though H channel based tracking was performing best among others but now Table. I shows that this is not the case for 'sphere' video sequence. In fact H color channels based tracking is performing the worst on this particular video sequence. The same is also true for the video sequence *"car1"*. On both of the video sequence G channel based tracking performed best. So we can conclude from here that there is no one color channel that works best in every scenario, we have to try out different options to find the suitable one for our particular case.

In terms of number of candidates usually all the color channels performance increased with the number of candidates for the video sequence *"sphere"*. But from the Table. III we can observe, in case of the *'car1'* video sequence if we increase the number of candidates from 49 to 100 the performance increases but if we further increase it to 400 performance decreases drastically for all the color channels. The reason for this, could be that this video sequence has a lot of camera movement noise so if the search area is too large then the model actually drifts away from the initial detection. Once the detection is far away it is unable to regain the track due to the fact that it is using detection of the previous frame as the searching seed point for the next frame. So again while choosing the number of candidates varies on each application scenario.

TABLE III. COMPARISON OF COLOR MODULE PERFORMANCE ON VIDEO SEQUENCE "CAR1"

Channels	Parameters	Average Processing Time per frame (millisecond/frame)	Score
G	#candidate = 49	2.503	0.477
	#candidate = 100	5.171	0.431
	#candidate = 400	12.611	0.161
H	#candidate = 49	6.459	0.069
	#candidate = 100	13.051	0.045
	#candidate = 400	41.431	0.127
Gray	#candidate = 49	2.393	0.453
	#candidate = 100	5.058	0.524
	#candidate = 400	12.933	0.148

**Note: For all the instance in the above table following parameters are same patch height =110 & width=110 , #bin=16 & stride=3.

Also in every case with the number of candidate processing time per frame has increased significantly, our data shows that for every 100 candidates the processing time has increased by a factor of approximately 1.6 on average. This suggests that even though increasing the number of candidates might increase the performance but it also increases the computational cost. We need to balance the trade-off between the performance gain and computational cost where we have some constraints on computation time/cost.

2) Section – II: Gradient-based Tracking

a) Method

In the second method, we used HOG, or Histogram of Oriented Gradients, as our feature. It is a feature descriptor that is often used to extract features from image data [4]. HOG divides the image into some sections and computes the gradient and orientation of the image patch or template. This information is then placed into histograms. HOG focuses on the structure or shape of the object.

Once we get the histograms describing the target model and the candidates, we can compute the similarity between them. Then we measure the L2 distance between the target histogram and the candidate histogram. The candidate with the lowest score is the most likely candidate for that frame.

b) Methodology

To switch between color-based and gradient-based methods, we define a parameter called feature. At first we need to set this parameter to run this mode of the module. As mentioned in the earlier section, all the methods are identical between color-based and gradient-based tracking, except for the feature

extraction part and the similarity function. Here we are using the HOG descriptor to model our target object and candidates. OpenCV library has a built-in implementation of HOG descriptor [6] which we have used to generate the feature from the object and candidate information. We implement the *calculateGradient()* method which takes input the model data and number of bins of the histogram as parameters and computes a histogram using that information. A similar method *computePatchGradient()* is implemented to compute the histograms of the individual patches as well. After that, we designed a method named *compareGradient()* to compare each candidate patch histogram with the target model histogram. This function generates the optimal candidate. Finally, we use *estimateTrackingPerformance()* to estimate the performance of our tracking algorithm.

c) Discussion

In this section we have been asked to implement and explore the gradient-based tracking and its features. To test our implementation we begin with the test video sequence *"bolt1"* on which it expected that our model would be able to track successfully upto 150 frames. But in reality our gradient module can track the entire video sequence except some frames near after 350. We have used the following parameters and obtained the result mentioned on Table. IV.

TABLE IV. GRADIENT MODULE PERFORMANCE ON VIDEO SEQUENCE "BOLT1"

Parameters	Average Processing Time per frame (millisecond/frame)	Score
Nbins = 9 Stride = 2 Candidates = 100 Height = 65 & Width = 65	22.218	0.442

In this module we have picked two real sequences two further evaluate our model and its characteristics. The first video sequence is *"basketball"* where we are supposed to track one of the basketball players. The challenges in this video sequence are sudden motion, occlusion of the player by other players for a brief period of time and players having the same appearance. According to our observation among them considering gradient features sudden motion is the prime challenge. To cope with this we have adjusted the stride size of the patch that we extracted. Also detailed performance reports are listed in Table. V.

First we compare in terms of candidates for this particular video sequence and we can see that an increase of the number of candidates does not improve the result and if we increase that by a large margin the performance starts to decrease. Also another point to note here that as we increase the number of candidates average processing time for each frame increases eventually slowing down our model. Another problem is that as shown in Fig. 2 we increase the candidate number we are essentially increasing the search area for our candidate, now that we are experimenting with 250 candidates after 418 frame our model starts to detect another nearby player whose appearance is similar to our target model. So we should be careful when we are defining the search area so that we don't include unnecessary objects that might harm the performance of our model.

Fig. 2: Estimated Bounding Box Shifted to Wrong Target Model (similar appearance) due to Large Candidate Search Area.

In terms of the number of bins we can see that from Table. V & VI if we reduce the number of bins the performance remains almost the same. If we increase the number of bins, performance decreases in each case of both of our test video sequences. In most cases 9 bins provides an optimal result for all the sequences. Also number of bins has little to no effect on the average processing time per frame.

TABLE V. COMPARISON OF GRADIENT MODULE PERFORMANCE ON VIDEO SEQUENCE "BASKETBALL"

#Cabdidates	#Bins	Average Processing Time per frame (millisecond/frame)	Score
49	5	15.441	0.652
	9	15.455	0.645
	15	14.939	0.145
100	5	31.031	0.635
	9	32.277	0.643
	15	33.751	0.099
400	5	51.160	0.630
	9	51.611	0.500
	15	55.353	0.463

**Note : For all the instances in the above table following parameters are the same patch height =125 & width=55 & stride=3.

TABLE VI. COMPARISON OF GRADIENT MODULE PERFORMANCE ON VIDEO SEQUENCE "BALL2"

#Cabdidates	#Bins	Average Processing Time per frame (millisecond/frame)	Score
49	5	11.964	0.075
	9	13.256	0.080
	15	13.521	0.057
100	5	26.623	0.097
	9	24.814	0.290
	15	25.051	0.079
400	5	48.821	0.201
	9	50.793	0.200
	15	59.085	0.071

**Note : For all the instances in the above table following parameters are the same patch height =19 & width=19 & stride=4.

Next we have applied our model on video sequence *"Ball2"* where we had to track a small fast moving ball. The main challenge of this fast moving ball and relative size of the ball.

We can observe from Table. VI that for this video sequence we had to search a relatively large area to even get some tracking. In case of 49 candidates we have very poor tracking performance and as the number increased to double the performance went 3 times higher than previous one. Further increment of the candidate number reduces the overall performance.

3) Section – III: Fusion of color-based and gradient-based tracking

a) Method

In this section, we fuse the methods described above, which a combination of color-based is and gradient-based tracking. We separately compute the similarity function for patches using both methods and then we fuse those results together to find an optimal candidate. Since both methods have strengths and weaknesses based on the scenario they are working on, a combination of them results in an outstanding performance. The fusion happens in two stages, we normalize the color-based and gradient-based scores using the formula in (4) and (5) respectively and then we combine those scores and find the optimal candidate using (6).

$$F_c'(y) = \frac{F_c(y) - \min(F_c)}{\max(F_c) - \min(F_c)} \quad (4)$$

$$F_g'(y) = \frac{F_g(y) - \min(F_g)}{\max(F_g) - \min(F_g)} \quad (5)$$

$$s = \underset{y}{\operatorname{argmin}}\{F'_c(y) + F'_g(y)\} \quad (6)$$

Where, F_c and F_g represents the distance scores from the color-based and gradient-based module respectively. $F_c'(y)$ and $F_g'(y)$ stands for normalized scores for the normalized score for the y^{th} candidate.

b) Methodology

We can use this tracking algorithm upon calling the *track()* method. It takes target model data, candidate data, number of bins and the mode (e.g. 1 for colour-based tracking). It generates the corresponding histograms for each candidate. We call this function twice to compute histograms for both types of tracking methods. Upon receiving the histograms, we call the *computeFusedScore()* method with the colour-based patch and gradient-based patch histograms along with their model histograms. This function first normalizes the scores separately and then computes the minimum distance score providing the desired candidate. Finally, we use *estimateTrackingPerformance()* to estimate the performance of our tracking algorithm.

c) Discussion

In this part, we have compared the 3 different approaches that we developed in this assignment. Here we have discussed how color-based, gradient-based and fusion of both approaches performed on different 3 real video sequences. Our initial assumption was that the fusion based approach would outperform other methods but in reality our result shows that this is not true for all the scenarios.

The first video sequence is *"bag"* where we are supposed to track a polyethylene bag. One of the problems with this video sequence is that the polyethylene bag is a highly deformable object. The object shape is constantly changing as it is blowing with the wind and as we are working with a fixed target model from the 1st frame this creates difficulty in tracking.

Regarding the performance we can see from Table. VII that, on this video sequence both colour based tracking and fusion-based tracking performed the same. On the other hand, gradient-based tracking has very poor performance. For the gradient-based model we have also experimented with parameters to improve the performance yet the performance remained fairly the same. The reason could be that the gradient-based tracker fails to cope with arbitrary motion direction of the bag. Though both Color-based and Fusion based model performance is the same but if we take a look at their average processing time per frame then there is a huge difference. Color module is more than 7 times faster than fusion model.

TABLE VII. COMPARISON BETWEEN DIFFERENT TRACKING MODULE

Video Sequence	Module	Parameter Setting	Average Processing Time per frame (millisecond /frame)	Score
"bag"	Color	#cand=225,#bin=16, stride=1,channel=G, height=110, width=110	9.190	0.316
	Gradient	#cand=225,#bin=16, stride=1,height=110, width=110	57.081	0.150
	Fusion	#cand=225,#bin=16, nbin=9, stride=1, channel=G, height=110, width=110	64.571	0.315
"ball"	Color	#cand=225,#bin=16, stride=1, channel=G	5.170	0.635
	Gradient	#cand=225,#bin=9, stride=1,height=60, width=60	56.144	0.088
	Fusion	#cand=225,#bin=16, nbin=9,stride=1, height=60,width=60	58.289	0.183
"road"	Color	#cand=225,#bin=16, channel=R, stride=1, height=60,width=60	2.939	0.332
	Gradient	#cand=225,#bin=9, stride=1 height=60,width=60	47.927	0.427
	Fusion	#cand=225,#bin=16, nbin=9,stride=1, height=60,width=60	40.482	0.031
	**Fusion with un-normalized feature score	#cand=225,#bin=16, nbin=9,stride=1, height=60,width=60	47.541	0.516

*** As we were experimenting with different things we noticed this unusual behavior thus we thought it would be worth mentioning in the report.

In the next video sequence we have tried to track a ball. Apparently in this video sequence we have not been able to notice any major difficulty. Even though except the color module all of the module provides a very poor performance in this video sequence. Color module was able to track the ball with a decent precision of over 63%. Table. VII shows the details of performance of each model with respective parameter settings.

Finally we have compared different approaches on video sequence *"road"* where the task is to track a biker. Some of the challenges associated with video sequence is that it has several brief occlusion in the racing track which specially creates problems with the color based tracker. It can be seen visually that the color based model gets little confused whenever there is an occlusion. Also the relative size of the object (biker) becomes very different when the biker is taking a big turn as a result creates a big difference with our target model. Due to this very fact all of our tracker loses track for the last few frames.

In this video sequence regarding performance Table. VII shows a complete picture of a different approach. Our gradient based model provides a decent result. Also visually the gradient based tracker is stable and smooth on this video sequence. On the other hand, the Color based model provides little less performance than the gradient based module. Also visually tracking of this module is a bit shaky and tends to be unstable. As both the models are doing a decent job our initial assumption was that combining these two would yield even better performance but our experimental result suggests otherwise. Fusion based approach performed very poorly. One of the unusual behaviors we noticed when we were experimenting is that an un-normalized feature score provides a very good result as we have listed in the last row of Table. VII.

IV. Conclusions

In conclusion, the performance of an object tracking algorithm depends on many factors including the application scenario and the optimal set of parameters running the algorithm. In our work, we investigated two different types of tracking approaches: color-based tracking and gradient-based tracking. We analyzed the algorithms separately in different scenarios and generated the optimal set of parameters. Finally, we fused those models in hope of getting a better response in tracking. While in some cases Fusion approach was working better than others but most of the time this was not the case. Thus there is no, one silver bullet for all the different scenarios. One needs to select a model and adjust parameters according to specific cases.

VI. Time Log

- **Background study** 3 hours: Understanding the framework provided by the lab and setting up the project environment.
- **Color-based tracking** 8 hours: Reading the research papers, designing the algorithm and finding the efficient program flow, Dealing with data types and familiarizing with the OpenCV library.
- **Gradient-based tracking** 4 hours: Understanding the HOG descriptor and using it in the program. It took less time to design from the previous tracking algorithm as the program flow is similar.
- **Fusion of color-based and gradient-based tracking** 3 hours: Understanding the fusion technique from the research paper and adapting it to our algorithm.
- **Evaluation** 10 hours: Experimenting with different combinations of parameters to find the optimal set of parameters best suited for the application scenario.
- **Report** 8 hours: writing, taking screenshots, proof-reading, and editing.

References

[1] Fieguth, P., & Terzopoulos, D. (1997, June). Color-based tracking of heads and other mobile objects at video frame rates. In Proceedings of IEEE computer society conference on computer vision and pattern recognition (pp. 21-27). IEEE.

[2] S. Birchfield, "Elliptical head tracking using intensity gradients and color histograms," Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231), Santa Barbara, CA, 1998, pp. 232-237.

[3] https://www.votchallenge.net

[4] https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/

[5] https://docs.opencv.org/3.4.4/d8/dc8/tutorial_histogram_comparison.html

[6] https://docs.opencv.org/3.4.4/d5/d33/structcv_1_1HOGDescriptor.html