Histogram-based Object Tracking

Jingwen Yang and Peter Szabo

I. INTRODUCTION

The laboratory report represents different histogram based tracking approaches. Its is written in C++ using the OpenCV library in eclipse. The color based tracker and gradient based tracker and their fusion were implemented based on the approach described in [1]. This report were split into 3 part, in each part we describe the methods itself, detail the implementation and the data, point out the challenges and discuss the effect of different parameters and their impact.

II. SECTION 1: TASK 4.1 AND 4.2 COLOR-BASED TRACKER

A. METHOD

The first task was the implementation of a color-based tracker, described in [1]. The approach is based on color histogram.

The target model is set on the first image (using the given ground truth), it is represented as a normalized color histogram, of a given color space. In our work we could set the different color spaces: gray-level, R, G, B from RGB and H,S from HSV.

The target localization is the following step. It starts from the position of the object in the previous frame, and scans its neighborhood, with a window, and looks for the most similar histogram in its neighborhood. This can be described as finding the maximum on a similarity map, which is created based on some similarity function. In our case this will be the Bhattacharyya coefficient. It describes the overlap of 2 distribution.

The same process is repeated for each frame.

B. METHODOLOGY

The project is divided into 2 class: main and utils. $Main.cpp$: is the basic execution of the algorithm, calls functions of the $utils.cpp$, in which the algorithms and functions are coded. Also $ShowManyImages.cpp$ is responsible for the visualization.

1) Detailed description:: Each frame will be converted into the desired, single channel image. For the first frame, we read the previously provided ground truth and obtain, the accurate position and details of the model. The target model, will be stored in a rectangle, that will store its x and y position (the upper left coordinate of the blob), and the width and the height. During the later frames, the same width and height will be used for histogram calculation for every candidate as well.

Based on these details a window will be created, by cutting this rectangle from the frame. $returnHistogramAroundRegion$ function calculates its normalized color histogram.

At each frame the candidates will be searched in a selected window around the position of the object in the previous frame. We can specify the size of the neighbourhood we wanna search, and how many pixel shift we expect between each window center. At each position we save the positional information of the candidates (stored in $candidate_rectangle$ container), and their normalized color histogram (stored in $candidates$ vector).

Finally the comparison between the candidates and the target is executed in $selectBestCandidate$ function, which will return the candidate with the smallest Bhattacharyya distance (its histogram has the highest overlap).

This object will be saved (for next iterations), and visualized. In visualization, at each frame, the given frame, the best candidate, the target model and a histogram image will be shown. In the histogram image, we can observe, both the best candidate(green) and the target(red) histogram.

2) Running the code: The running of the code can be done via the attached make file. The different a parameters can be set via the arguments of the makefile. Arguments:

- **colorspace:** which color-space we want to use in our algorithm. R: red in RGB, G: green in RGB, B: blue in RGB, H: Hue in HSV, S: Saturation in HSV, d: (default) stands for gray-scale color channel.
- **bins of histogram** expects a number for the size of the histogram, which in this task was always 16 (as the description specified)
- **size of neighborhood** half of the width of the window we are looking for candidates (around or object)
- **stride** how many pixels to be examined in the window, in case 1, every pixel, 2 every second etc.

Example command:

```
$ ./main R 16 4 1
```

Which will choose the Red channel with a histogram of 16 bins, 4 neighbors with one stride, which will result 64 candidates. The testing should be done on $sphere$ and car data-set, and they were hard-coded into the code.

The path of the dataset should be configured manually in the code.

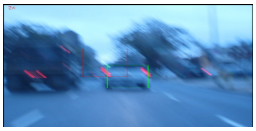
In addition, out program saves the video sequence into a file, displays the frames, and saves the score of the $estimateTrackingPerformance$ function.

C. DATA AND CHALLENGES

Two data sets were used for testing: $car1$ and $sphere$.

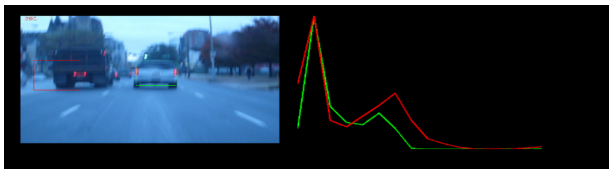
Sequence	Frames	Challenges
$car1$	400	complex background,big motion,low resolution
$sphere$	201	rapid movement, transparent structure

TABLE I: The details and challenges of the data sets



(a) Rapid movement of camera

(b) Different object scale



(c) Candidate matched object with similar histogram (red: target hist, green: best candidate)

Fig. 1: Possible tracking problems. At image (a) we can see the rapid movement of the camera, (b) visualizes re-scaling (green is the ground truth, red is the model), (c) shows when the target histogram is not distinctive, and the tracker finds other candidates within the candidates with similar histogram

In the table above a few on the biggest challenges were collected. In the following paragraph they will be extended and detailed. Also more general errors will be covered. One of the most significant problem in tracking is the correct definition of the target model. It should be representative, distinctive and stable. In this section, the color space was to only parameter for us to choose for the model, the width, height and the histogram (based on the ground truth initial position) were given and we were not required to update it. Later on we will see its impact.

Lack of uniqueness could potentially pose the risk of finding multiple, false objects having more similar histogram as the ground truth, as we can see in **car** data set on Figure 1 (c). In addition, a further challenge of car data set is the rapid movement of the whole camera. This results a sudden change of the object placement, which could result the loss of tracking. This also lowers the resolution on those frames. This can be seen on Figure 1 (a).

In the $sphere$ data set, one of the biggest challenge is the transparency of the object. The object changes its own look, based on its background, which makes it difficult to track using a fixed target model definition.

In addition, we could observe the change of object, not only in terms of color, but also shape (as object could show us different side to us, we don't have that in these data sets), or scale (observable in both data sets). This can be seen on Figure 1 (b), where we see, that the we used the initial width and height for creating the box, but the as the ground truth shows (in green), the object changed its scale. Finally the inclusion of background could also pose a tracking challenge. This can be magnified with the object being re-scaled (for example to be smaller, and include more background).

	Candidates	Average score		
		car1	sphere	avg
Gray	64	0.356277	0.319773	0.338025
	144	0.380754	0.357864	0.369309
	1296	0.52644	0.522306	**0.524373**
	2304	0.39411	0.532236	0.463173
Hue	64	-	0.157347	-
	144	-	0.210404	-
Saturation	64	0.258973	0.139701	0.1993
	144	0.31681	0.183556	0.250183
R	64	0.033585	0.34679	0.19019
	144	0.137893	0.326057	0.231975
G	64	0.345275	0.278108	0.311691
	144	0.361272	0.328317	0.344794
B	64	0.24022	0.342306	0.291263
	144	0.297903	0.26883	0.283366

TABLE II: Results of color based tracking with different number of candidates and different color channel

Color Channel	Stride	No Candidates	AverageScore		
			car1	sphere	average
Gray	2	64	0.384237	0.380613	0.382425
		144	0.434651	0.502412	0.4685315
	3	64	0.479897	0.497764	0.48883
		144	0.532391	0.517662	0.5250265
Green	2	64	0.392338	0.347938	0.370138
		144	0.447046	0.49981	0.473428
	3	64	0.346662	0.448822	0.397742
		144	0.433989	0.538612	0.4863

TABLE III: Results of color based tracking, and the effect of different stride

D. DISCUSSION

After we defined the possible problems, we started to explore the effect of different number of candidates and different color spaces. In the section we will discuss our results of seeking for the best parameter setup.

We set the **number of bins** to be equal to **16** throughout the whole task, and the modified the color channel and the number of parameters.

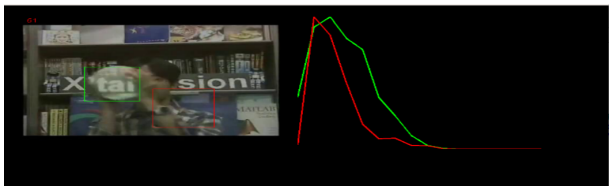
Table II shows the result of different parameter setups, with stride = 1. At the beginning our we experimenting with candidate numbers around 100 (based on the task specification).

For **car** data set, as we can observe on Table II, within these candidate range, we can see, that gray-scale and gray color channel gave the best results. We took a more detailed look on the **gray color space**. Observing the scene, we can see that due to the light conditions, the car has a gray color, which explains the superior accuracy of the gray image scale. That indicates that this color space is able to a good histogram representation of the object. In addition, Table II also shows that the between that more than 100 candidates gave better results, since they seem to be more robust to the rapid camera movements. In contrary with smaller examined region (with **64** candidates), if the camera takes a bigger movement, we need more time to get back to the object. Also, with a more reliable target model, we could allow the algorithm to discover a higher number of candidates, with smaller risk of finding a false positive result. Which is not true in the case of a less reliable object model.

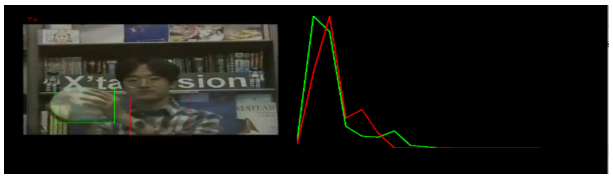
(a) Finding neighboring cars with similar histogram

(b) False detection of lamp, due to its red color

Fig. 2: Problems with red channel on car1 data set with more than 100 candidates



(a) Saturation channel tracking



(b) Hue channel tracking

(c) Hue channel tracking

(d) Hue channel model

Fig. 3: Results of tracking in Hue and Saturation channels in sphere database, with less then 100 candidate object

To model this scenario examined a less accurate color space: the **red color channel** on car1 data set. The red color channel only modeled the red light of the car. But as we can see on the images, the red brake light is not always visible (if its not lighting its less visible red), and its only small portion of the whole car. As we can see on Figure 2 it loses the object easily, and finds other histograms with more similar histogram (the red light of an another car).

In the case of the **sphere data set**, as we already mentioned the biggest challenge is the transparency of the object. We experimented with the **hue** (utilizing their photo metric invariance property), assuming that a ball will be brighter before a brighter background. Unfortunately as we can observe on Figure 6 both H and S failed to track the changes of the background, and we regularly lost the track of the real object, both with high number or low number of candidates.

As Table II, in this case also the gray scale and R color space proved to provide the highest accuracy. The models can be seen on 4 Figure.

But as we can see on Table II, the best model, that generalized the best, is the gray channel model. Moreover

(a) The model of red color channel

(b) The model of blue color channel

(c) The model of green color channel

Fig. 4: Models of red, blue, green color spaces on $sphere$ data set

we also experienced with increasing the **stride**. The results can be seen on Table 6. As we can see, by letting the model the explore a bigger neighbourhood, we managed to achieve higher accuracy. This can be explained by the fact, that in both data sets we can observe rapid movements in the movements of the object.

And it is worth to note, that even though it increased the accuracy of multiple color channel tracking, the gray scale remained the best. It supports the idea, that the among all the color features the gray scale is the most general.

That was the motivation to inspect the effect of essentially higher number of candidates in case of gray scale channel. That can be seen in Table II 3rd and 4th row. And as the results imply that was the best a parameters setup we managed to achieve: **with gray scale image channel, 1 stride and 1296 candidates!**. On downside that due to this search the algorithm runs slower due to the higher computational costs, but in the scenario regardless of our weak hardware (intel i5) it still run real time.

An interesting observation is, that even though we see, that as we increased the size of the searching area, we got better results, there is limit for that. We can see, that at 2304 candidates we dont get better results anymore (but slightly worse).

III. SECTION 2:TASK 4.3 AND 4.4 GRADIENT-BASED TRACKER

A. METHOD

The second task was the implementation of a gradient-based tracker[1]. By computing the normalized sum of the gradient magnitude around the perimeter of the bounding box. As gradient features we are using the HOG descriptor.

$$\phi_g(s) = \frac{1}{N_\sigma} \sum_{i=1}^{N_\sigma} \mid g_s(i) \mid \tag{1}$$

where $g_s(i)$ is the intensity gradient at perimeter pixel i of the bounding box at locaition s, and N_σ is the number of pixels on the perimeter of an bounding box with size σ.

The target model is set on the first frame (using the given groundtruth),which is represented as a normalized gradient histogram.
The candidate generation is the following step. It starts from the state of the object in the previous frame(the location of its bounding box), and do exhaustive search inside its neighborhood, with a window, looking for the most similar gradient histogram generated by candidates. This can be described as finding the maximum on a similarity map, which is created based on L2 distance. The candidate generation will be applied on each frame.

B. Implementation

The project is divided into one main class and a main file: the class file $utils.cpp$, the headfile of the class $util.hpp$ and the main script $LAB4\_TASK1$ which show cases the results and sets up the experiment. 1. The following subsections point out the specifics of the implementation, and the last subsections explains how to run the project.

1) utils: There are 7 functions created to help with this task.

- **readGroundTruthFile**: given function, reads the path of ground truth data and returns the ground truth bounding box of each sequence.
- **estimateTrackingPerformance**:given function, reads the bounding box of groundtruth and estimated object state, and compare them to give a score according to their overlapped area over union area.
- **returnGradientHistogramAroundRegion**:created function, reads the image cropped by a bounding box and the number of bins expected for the gradient histogram. It returns the HOG descriptor of cropped area in Mat format.
- **setHistogramColor**: created function, reads the current frame and expected color channel, returns the correspondent one channel image, which is always gray scale in this task. We keep this function in case we want to compare the gradient histogram tracker's performance in differnt color channel.
- **calculateCandidates**: created function, reads the gray scale current frame, the empty vector for saving estimated candidates Mat, the empty vector for saving estimated candidates bounding boxes, the previous State of object, then the three parameters $size\_of\_neighbourhood$: size of searching neighbor for candidate generation, $step\_size$ stride for candidate generation, $hist\_Size$: the number of bins of histogram. It basically just save the estimated candidates' information.
- **histogramImage**: created function, reads the model histogram, candidate histogram and the number of bins of the histogram. It visualizes the histograms on the screen.
- **selectBestCandidate**: created function, reads the model histogram, the vector of generated candidates, and the distance vector of candidates. It returns the most likely candidate with smallest distance comparing to the target model histogram.

2) Running the code: In the main file $LAB4\_TASK1.CPP$, the parameters could be set in a similar way as we do in the previous task. Since the number of candidates is tricky to implement in the code, instead we use the radius of searching area. So the number of candidates N can be easily calculated as

$$N = (\frac{2 * size\_of\_neighbourhood}{step\_size})^2 \tag{2}$$

On the test sequence $bolt1$, the default parameter settings below will generate 100 candidates for each frame. with stride 1 or stride 2, both works pretty well for the first 150 frames.

TABLE IV: Parameters for test data

Parameter	setting 1	setting 2
$hist\_size$	9	9
$size\_of\_neighbourhood$	10	20
$step\_size$	2	1

C. DATA

Two sequences will be used to evaluate the code implementation in this task, $ball2$ and $basketball$.

TABLE V: Sequences for evaluating Gradient-based Tracking

Sequence	frames	challenges
$ball2$	40	complex background,big motion,low resolution
$basketball$	499	similar candidate,significant motion

D. METHODOLOGY

The algorithm has 3 parameters as below:

TABLE VI: Sequences for evaluating Gradient-based Tracking

Parameter	Variable in code
Number of bins in histogram model	$hist\_size$
Searching radius for candidate generation	$size\_of\_neighbourhood$
Stride for candidate generation	$step\_size$

To analyze each parameter's function in gradient-based tracking, we need to tune each of them one by one. We keep the other parameters fixed while tuning one parameter, so that we can separate the individual influence of each of them.
There are two standards to evaluate the tracking performance, one is visually observe if the red bounding box(estimations) is following and covering the green bounding box(groundtruth). For this one we mainly focus on the following possible cases:

The other one is quantitatively calculating the average score(Overlapped Area of the bounding boxes of groundtruth and estimations divided by Union Area of them) of the sequence. For this one, first we try to discover the role of number of candidates, the number of bin is fixed to default

TABLE VII: Possible matching for evaluating Gradient-based Tracking

Case	Fact
True Positive	Object matched, histogram matched
True Negative	Object mismatched, histogram mismatched
False Positive	Object mismatched, histogram matched
False Negative	Object matched, histogram mismatched

value$hist_size = 9$. Since we believe the stride plays an essential role in candidate searching process, we also tune the stride a little bit to see how it helps. With different combinations of searching radius $size\_of\_neighbourhood$ and stride $step\_size$, we gradually change the number of candidates. After finding an relatively ideal setting for number of candidates, we keep the ideal combination of $size\_of\_neighbourhood$ and $step\_size$ and tune the other parameter number of bins in the histogram model $hist\_size$.

E. RESULTS & DISCUSSION

1) 'ball2': By visual evaluation, the performance of gradient-based tracking is regarded bad in this sequence. Only the first few frames is tracking the object in a proper manner. It totally lost the tracking soon after around 10 frames and the following estimations are far away from the ground truth.

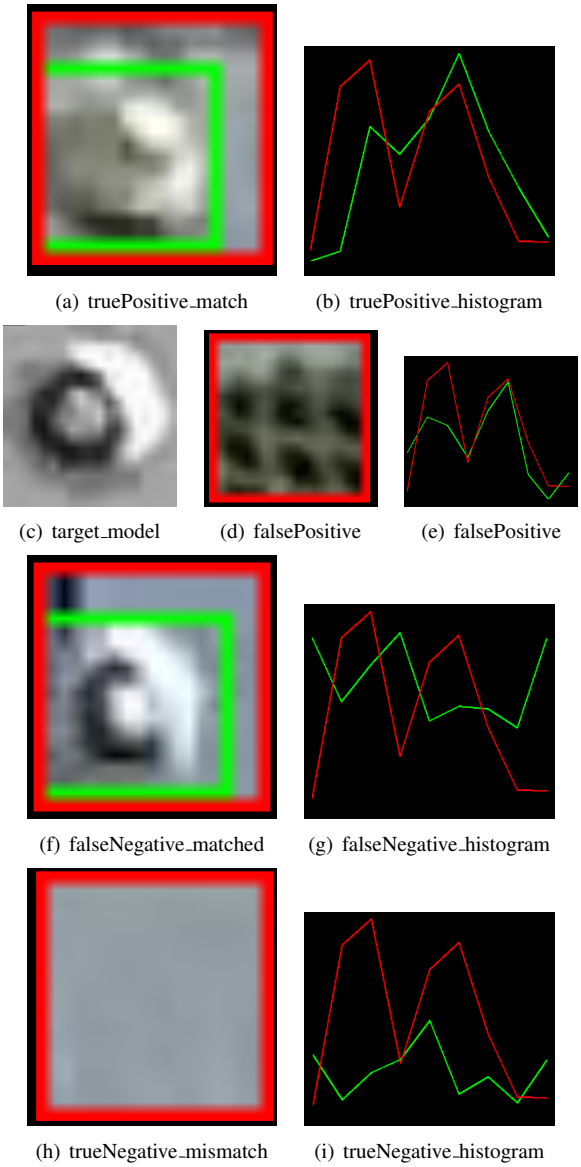
As figure 5 shows, we call see the drawbacks of a simple model. At very beginning when the tracker is still working properly, we can have the result of subfigure a&b with matched estimation and matched histogram. But even the estimation is good, sometimes the histogram differs a lot from the model histogram as shown in subfigure f&g, which can be improved by updating the target model. Soon after a few frames, we lost the track of the ball and the tracker started to estimate in the middle of the sky, which we have neither matched estimation nor matched histogram, as shown in subfigure h&i. Then the background became complicated, the tracker got confused the background with target model. So the estimation is not correct but histogram matched.For this problem we should develop more robust target model, as shown in subfigure c&d&e.

With the quantitative methodology presented above, we get the result table for the first sequence as below:

From table VIII we can tell the best performances are generated by 1936 candidates with stride 1, 1600 candidates with stride 1, 324 candidates with stride 2. 256 candidates with stride 3. We should try to avoid huge number of candidates in case we got big model. So the combinations of **324 candidates with stride 2**, **256 candidates with stride 3** seem to be reasonable. Besides, if computational cost is feasible, we can also use the combination of **1600 candidates with stride 1**.

Because of the significant movement of ball and the tiny size of itself, it is difficult to locate the next state of the ball. In this case we should try to enlarge the searching area.

From table IX we can see the best performance most of the time belongs to 5 bins histogram model. It has better



(a) truePositive_match (b) truePositive_histogram

(c) target_model (d) falsePositive (e) falsePositive

(f) falseNegative_matched (g) falseNegative_histogram

(h) trueNegative_mismatch (i) trueNegative_histogram

Fig. 5: Results of ball2 With 9 bins and 256 candidates

scores than 9 bins maybe because of the low quality of the image.

The interesting thing is, even the number of bins decrease to 3, the average score is not getting too bad but still acceptable. But once it reduces to 2, the whole model just crack. So we believe gradient-based histogram model has a least demand in terms of number of bins for depicting objects.

In total, the highest score produced by gradient-based histogram tracking on sequence 'ball2' is around 0.20, which is not ideal at all. It is probably led by the sudden great motion of the ball when it hits the goal. Since that moment the tracker lost tracking completely. Also, the background is complex and have rich gradient information which can not be handled by such a simple model constructed by gradient histogram. Overall, the best result is obtained by searching around 1000 candidates with 5bins histogram model.

TABLE VIII: Results for sequence *ball2*, tuning Number of Candidates

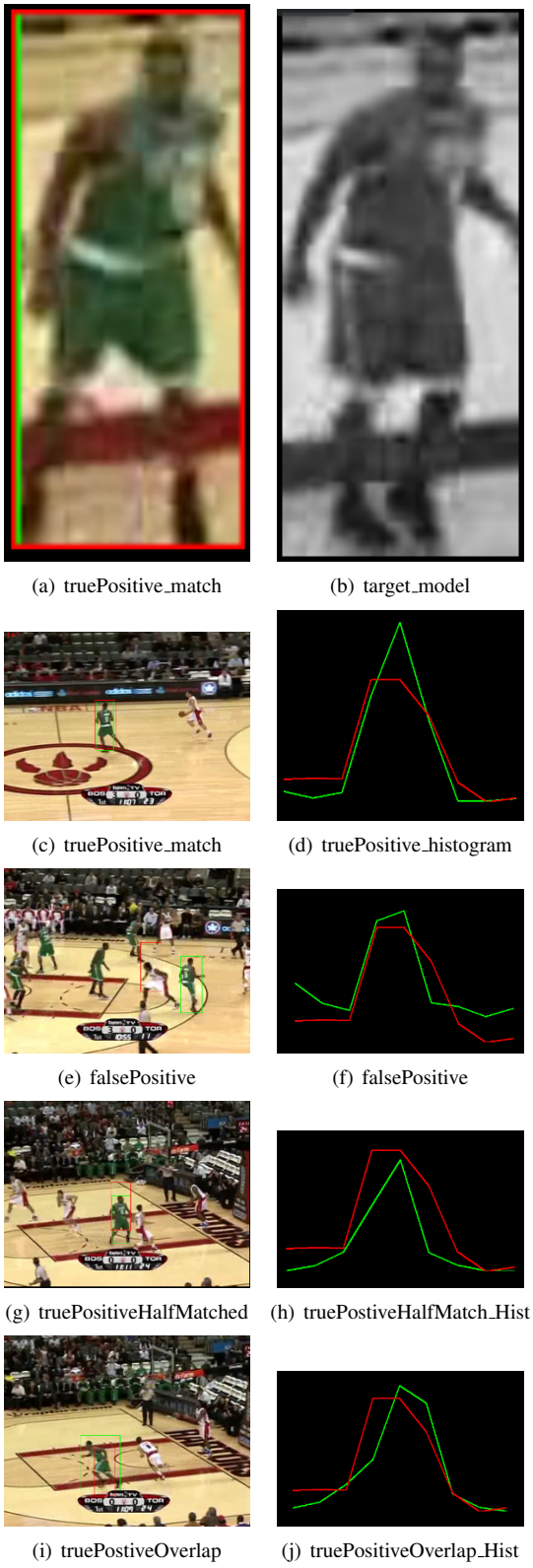
NumberofBins	NumberofCandidates&Stride	AverageScore
9	1936—1	**0.1786**
9	1600—1	**0.1791**
9	1024—1	0.1536
9	900—1	0.1462
9	400—1	0.0576
9	625—2	0.1595
9	400—2	0.1117
9	324—2	**0.1760**
9	225—2	0.1601
9	100—2	0.0522
9	256—3	0.1635
9	196—3	0.1122
9	144—3	0.1556
9	100—3	0.0944
9	49—3	0.0477
9	100—4	0.1153
9	64—4	0.1582
9	25—4	0.0105
9	64—5	0.1608
9	36—5	0.0654
9	400—5	0.1117

TABLE IX: Results for sequence *ball2*, tuning Number of Bins

NumberofBins	NumberofCandidates&Stride	AverageScore
11	1600—1	0.1067
10	1600—1	0.1776
9	1600—1	0.1786
7	1600—1	0.1110
5	1600—1	**0.1942**
3	1600—1	0.1901
2	1600—1	0.0479
11	1024—1	0.1568
9	1024—1	0.1791
7	1024—1	0.1745
5	1024—1	**0.2008**
3	1024—1	0.1901
9	324—2	0.1536
9	324—2	0.1718
5	324—2	**0.1994**
3	324—2	0.1866
2	324—2	0.0506
9	625—2	0.1536
5	625—2	**0.1993**
3	625—2	0.1602
2	625—2	0.0198

2) 'basketball': By visual evaluation, the performance of gradient-based tracking is regarded good in this sequence. With a good parameter setting, only for last few frames the tracker lost its target object. The first around 480 out of 500 frames have decent estimations which are quite close to the ground truth. We can obviously see the improvement of gradient-based tracker in person tracking than ball tracking.

As figure 5 shows, we call see the candidate histogram most of the time matches very well with the target model. This apparently could be a potential problem when there are similar candidates showing around since they all have exactly the same gradient histograms, shown by subfigure c&d&e&f. Apart the problem of mismatching, the matches could also have problems in accuracy and precision. One of the cases is imprecise estimation of the location As shown in subfigure



(a) truePositive_match (b) target_model

(c) truePositive_match (d) truePositive_histogram

(e) falsePositive (f) falsePositive

(g) truePositiveHalfMatched (h) truePostiveHalfMatch_Hist

(i) truePostiveOverlap (j) truePositiveOverlap_Hist

Fig. 6: Results of ball2 With 9 bins and 256 candidates

g&h, there is a great translation between the ground truth and estimation. But the histograms match well. Another case is the imprecise size of estimation. As the shown in subfigure i&j, the estimation is way smaller than the ground truth but the histograms match well. For this problem we should develop more progressive updated target model to fit the change of object state.

With the quantitative methodology presented above, we get the result table for the first sequence as below:

TABLE X: Results for sequence $basketball$, tuning Number of Candidates

NumberofBins	NumberofCandidates—Stride	AverageScore
9	1296—1	**0.6148**
9	900—1	0.6146
9	576—1	0.0912
9	400—1	0.0910
9	484—2	**0.6178**
9	400—2	0.6153
9	324—2	0.6147
9	256—2	0.6158
9	144—2	0.0905

From table X we can tell the good performances can be achieved by minimum 900 candidates with stride 1, minimum 256 candidates with stride 2. When the accuracy get 61%, the increase of number of candidates doesn't help to improve the performance any more.

Because of the similar appearance in terms of gradient histogram model for basketball players in this video, when the candidates get too close to each other, we lost the track of the player we want. In this case, we should try to merge with other identical information to make sure we are tracking the same person.

TABLE XI: Results for sequence $basketball$, tuning Number of Bins

NumberofBins	NumberofCandidates&Stride	AverageScore
15	900& 1	0.4561
9	900& 1	0.6146
7	900& 1	**0.6203**
5	900& 1	0.0947
12	225& 2	0.0864
9	256& 2	0.6158
7	225& 2	**0.6192**
5	225& 2	0.6158

From table XI we can see the best score is achieved by 7 bins histogram model. It has slightly better scores than 9 bins. So we can draw the conclusion that around 7-9 bins should be the ideal number for person gradient histogram model. The further increasing in number of bins will cause decrease in tracking performance.

The interesting thing is, with 5 bins, the combination of 225 candidates with stride 2 works pretty well without noticeable decrease in the performance, while 900 candidates with stride equals to one, greatly decreases the performance. We believe that the searching area does matter in this case. Both of them have the same searching area which is a 30*30 window. Therefore, we can draw the conclusion that the stride does matter in tracking problem. Further decrease in number of bins will cause greatly decrease in tracking performance.

In total, the highest score produced by gradient-based histogram tracking on sequence 'basketball' is around 0.62, which is pretty decent. The gradient-based histogram is suitable for people tracking, which might because the person is significantly different from the background. Overall, the best result is obtained by searching around 900 candidates with 7 bins histogram model, and 225 candidates with 7 bins histogram model if considering the computation cost.

IV. SECTION 3:TASK 4.5 AND 4.6 COLOR& GRADIENT-BASED TRACKER

A. METHOD

The third task is the implementation of a combined color&gradient-based tracker[1]. By summing up the normalized distances of gradient histogram and the color histogram for each bounding box.

$$\phi_M(s) = \phi_g(s) + \phi_c(s) \tag{3}$$

where $\phi_M(s)$ is the mixture score of color score and gradient score(1). Keep in mind that since the color histogram comparison we are using the Battacharyya distance while the gradient histogram comparison we are using the L2 norm. Therefore, make sure we normalize them before adding up.

Same as previous tasks, the target model is set on the first frame (using the given groundtruth), which is represented as a normalized color/gradient histogram. The candidate generation is the same as previous task, saving the histograms for each neighbor within the searching area. But candidate selection is modified to fit in both color feature and gradient feature. The distances we got from color histogram and gradient histogram are fused for each potential candidate, and the smallest one is selected inside the searching area. The estimation process will be applied on each frame.

B. Implementation

The project is divided into one main class and a main file: the class file $utils.cpp$, the headfile of the class $util.hpp$ and the main script $LAB4\_TASK1$ which show cases the results and sets up the experiment. 1. The following subsections point out the specifics of the implementation, and the last subsections explains how to run the project.

1) utils: There are 7 functions created to help with this task.

- **readGroundTruthFile**: given function, reads the path of ground truth data and returns the ground truth bounding box of each sequence.
- **estimateTrackingPerformance**:given function, reads the bounding box of groundtruth and estimated object

state, and compare them to give a score according to their overlapped area over union area.

- **returnColorHistogramAroundRegion**:created function, reads the image cropped by a bounding box and the number of bins expected for the color histogram. It returns the color histogram of cropped area in Mat format.
- **returnGradientHistogramAroundRegion**:created function, reads the image cropped by a bounding box and the number of bins expected for the gradient histogram. It returns the HOG descriptor of cropped area in Mat format.
- **setHistogramColor**: created function, reads the current frame and expected color channel, returns the correspondent one channel image, which is among H,S,R,G,B,GRAY for color histogram and remains GRAY for gradient histogram.
- **calculateCandidates**: created function, reads $singleChannelFrame$: the gray scale current frame , $candidates$: the empty vector for saving estimated candidates Mat, $candidate\_rectangle$: the empty vector for saving estimated candidates bounding boxes, $currentState$: the previous State of object, then the three tuning parameters $size\_of\_neighbourhood$: size of searching neighbor for candidate generation, $step\_size$: stride for candidate generation, $hist\_Size$: the number of bins of histogram, and $color\_based$ bool variable indicating if it's for color-based model or gradient-based model. It saves the estimated candidates' information.
- **histogramImage**: created function, reads the model histogram, candidate histogram and the number of bins of the histogram. It visualizes the histograms on the screen.
- **CalculateDIstances**: created function, reads $basicModelHistogram$: the model histogram, $candidates$: the vector of generated candidates, $distances$: the distance vector of candidates, and $color\_based$ the bool variable indicating it's calling for color-based model or gradient-based model. It calculates and saves the distances for all the candidates comparing to the target model histogram.
- **selectCombinedCandidate**: created function, reads $distances\_color$ & $distances\_gradient$ the distances of all the candidates for two models. It returns the most likely candidate with smallest combiend distance comparing to the target model histogram.

2) Running the code: In the main file $LAB4\_TASK1.CPP$, the parameters to play around are as follows:

The parameters are set as table 10 because in task 4.1 it is suggested that keep the number of bins at 16, and in task 4.4 it is concluded that the number of bins for person tracking is the best at 7. For the number of candidates we are generating 225 candidates with stride 2, searching radius of 15 pixels here, which can have decent performance for the first around 130 frames. The bool parameters are for choosing the model.

TABLE XII: Parameters for tuning

Parameter	Test Setting	Place in MainScript
$color\_features$	Gray	line 75
$color\_hist\_size$	16	line 76
$gradient_hist\_size$	7	line 78
$size\_of\_neighbourhood$	15	line 81
$step\_size$	2	line 82
$only_color$	false	line 84
$only_gradient$	false	line 85

The number of candidates can be calculated as equation 44.

$$N = (\frac{2 * size\_of\_neighbourhood}{step\_size})^2 \tag{4}$$

C. DATA

Three sequences will be used to evaluate the code implementation in this task, $ball2$ and $basketball$.

TABLE XIII: Sequences for evaluating Gradient-based Tracking

Sequence	frames	challenges
bag	196	significant change in shape
$ball$	601	fast motion
$road$	400	Obstruction

D. METHODOLOGY

The parameters we are tuning here are in the tableXII. First we play around the $color\_features$ and $color\_hist\_size$ for color-based histogram to find the best settings. Then we fixed them and tune $gradient_hist\_size$, $size\_of\_neighbourhood$ and $step\_size$ to improve performance for gradient-based tracker. Try to find the best parameter settings for the combined model, then compare with single feature model and analyze the results.
There are two standards to evaluate the tracking performance, one is visually observe if the red bounding box(estimations) is following and covering the green bounding box(ground truth).
The other one is quantitatively calculating the average score(Overlapped Area of the bounding boxes of groundtruth and estimations divided by Union Area of them) of the sequence.

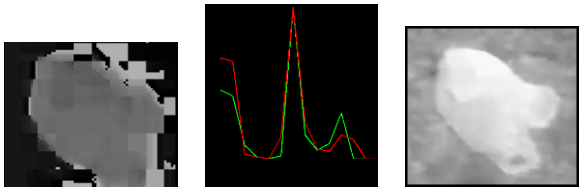
E. RESULTS & DISCUSSION

1) 'Bag': By visual evaluation, the performance of color&gradient-based tracking is regarded not satisfying in this sequence. Only with an outstanding parameter setting can we track the object in high proportion of frames. With the quantitative methodology presented in methodology, we get the result table as below:

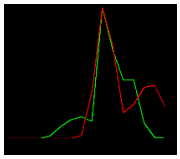
As we can see from tableXIV, the R channel does a superb job in bag sequence while the H channel also works relatively good. As we could observe during our previous experiments, the Gray scale image has a decent performance. As shown in figure7, the H channel model has very strong contrast, which

TABLE XIV: *Bag*, tuning color features

CHANNEL	NumberofCandidates	AverageScore
H	225	**0.3584**
S	225	0.0549
R	324	0.3642
R	225	**0.3717**
R	100	0.3558
G	225	0.2927
B	225	0.2610
GRAY	225	**0.3526**



(a) H channel (b) H channel Histogram (c) R channel



(d) R channel Histogram

Fig. 7: H and R Channel for Bag

makes it easier to separate the white bag from the dark floor. The candidate's histogram of H channel fits quite well with the target model histogram. The best number of candidates will be around 225.

TABLE XV: Tuning Gradient Features,stride 2

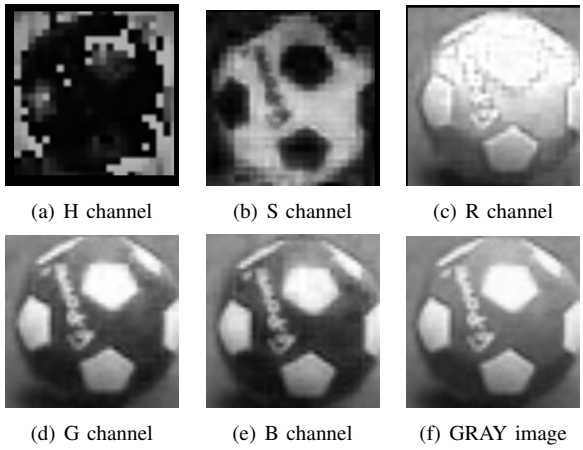
NumberofBins	NumberofCandidates	AverageScore
9	100,GRAY	0.2794
7	225,GRAY	0.3526
5	225,GRAY	0.3016
9	400,R	0.3638
9	225,R	0.3490
7	225,R	**0.3717**
5	400,R	**0.3757**
5	225,R	0.3447

From tableXV we can tell that The best performance is achieved by around 400 candidates with 5 or 7 bins as gradient features. It is because of the big movement of the bag caused by the wind.

TABLE XVI: Results for sequence *ball*, tuning Gradient-based features& number of candidates

MODEL	AverageScore
COLOR	0.3881
GRADIENT	0.3757
COMBINED	0.0319

For model comparing, as the tableXVI, we can see the performance of single feature color-based tracker is even better than the combined model. It is possibly because of



(a) H channel (b) S channel (c) R channel

(d) G channel (e) B channel (f) GRAY image

Fig. 8: Different Color Channels for Combined Tracker

the round shape and blurry boundary makes gradient-based tracker function improperly. Thus the information regarding gradient is not helpful at all, which even decreases the judgement of the combined tracker. The difference between color-based tracker and color&gradient-based tracker is not obvious.

2) 'ball': Since in task 4.2 the gray scale proved the provide the best result, for this sequence we start from gray color feature. Based on visual evaluation, the combined model shows promising result for sequence 'ball'. On most of frames it could be able to track the object successfully and not much bias in terms of the overlapping with ground truth.

The color channel models are shown in figure 68.

With the quantitative methodology, we can obtain results as in table 12.

TABLE XVII: Tuning Color-based features with 7bins gradient histogram, 225 candidates, stride 2

CHANNEL, NumberofBins_color	AverageScore
H,16	0.2222
S,16	0.2664
R,16	0.4067
G,16	**0.6073**
B,16	0.4490
GRAY,16	**0.6076**
GRAY,25	0.1065
GRAY,12	0.1453

In the following paragraph we describe, why G channel and gray scale image works better than the others. The contrast information is kept from original image. The R channel is too similar to the sofa in the background, which will make the color tracker unable to distinguish. H and S channel have too much dark pixels which may also be confused with the dark area in background.

Therefore for this sequence we are going to fix the channel to gray level image. Then we tune the number of bins. Our experiments and experience shows, that the 16 bins model provides accurate results. The increase of bins doesn't contribute to improve the performance while the

decrease of bins greatly decreases the performance of the combined tracker. As the histogram of Figure 8 shows, the color histogram has a special pattern, the pixels mainly aggregated in small value range and evenly distributed for the rest of the spectrum, which makes the resolution of histogram unimportant. Thus, in the later tuning of other parameters, the parameters for color-based tracker will be fixed to 'gray' and 16bins.

TABLE XVIII: Results for sequence *ball*, tuning Gradient-based features& number of candidates

NumberofBins	NumberofCandidates,Stride	AverageScore
3	225,2	0.6084
5	225,2	0.6014
7	100,2	0.2657
7	225,2	0.6076
7	400,2	0.6012
9	225,2	0.6040
11	225,2	0.6065
7	1600,1	0.6088
7	400,1	0.6076
7	256,1	0.2751

With stride 2 and 225 candidates, apparently the number of bins for gradient histogram does not have a significant impact on the performance of combined tracker. We fix the number of bins to 7, which has the best score under the same settings. The number of candidates turns out to be more than 200. With 100 candidates the combined tracker performs way worse compared to 225 candidates. With stride 1, the number of candidates have to be at least around 400. Further increase will lead to a better performance. Therefore, in this case we can conclude, the **number of candidates around 225** with **stride 2** and **7 bins** in the gradient histogram works the best.

With the best parameters we found, now we are comparing among three models. As shown in table 16, the combined model is definitely better than single feature model. As visual evaluation, it also can be observed that the color model works good at the beginning, once there is similar color in the background it lost the tracking while the gradient-based tracker doesn't work well the whole time.

TABLE XIX: Results for sequence *ball*, tuning Gradient-based features& number of candidates

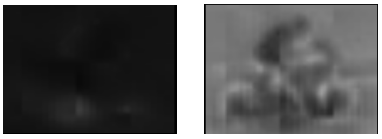
MODEL	AverageScore
COLOR	0.1928
GRADIENT	0.0848
COMBINED	0.6076

3) 'road': By visual evaluation, the performance of color&gradient-based tracking is regarded decent in this sequence. With a good parameter setting, the object is always been tracked in every frame. With the quantitative methodology presented in methodology, we get the result table as below:

From tableXX we can see that, the H and S channel doesn't work for this sequence. The target models of H and S channel are shown in figure9. We suppose it is because the H and S channel does not have enough contrast information. At the mean time, the R,G,B,GRAY channel all works similarly good for this sequence. With the comparison between 100 candidates and 225 candidates we can obviously draw the conclusion that 225 candidates works better. So we are going to stick to color features with GRAY or G channel and 225 candidates.

TABLE XX: *road*, tuning color features

CHANNEL	NumberofCandidates	AverageScore
H	225	0.2215
S	225	0.2803
R	225	0.5153
R	100	0.5014
G	225	**0.5194**
G	100	0.4847
B	225	0.5143
B	100	0.4312
GRAY	225	**0.5187**
GRAY	100	0.4992



(a) H channel (b) S channel

Fig. 9: Different Color Channels

TABLE XXI: Tuning Gradient Features,stride 2

NumberofBins	NumberofCandidates	AverageScore
5	225,GRAY	0.5038
7	225,GRAY	**0.5194**
9	225,GRAY	0.5016
10	225,GRAY	0.2846
7	225,channelG	**0.5187**
9	225,channelG	0.4843

From Table XXI we can tell that the number of bins for gradient model should be around 5-9.

TABLE XXII: Results for sequence *ball*, tuning Gradient-based features& number of candidates

MODEL	AverageScore
COLOR	0.2950
GRADIENT	0.4878
COMBINED	0.5194

For model comparing, as the tableXXII shows, the color tracker could not handle the tracking task alone, while the gradient tracker is not bad, only a little bit worse than the combined model. We suppose it is because the motor rider has very clear shape comparing to the background while the color histogram will be influenced by the obstruction of trees due to the angle of filming.

V. TIME LOG

Wayyy tooo much time

VI. REFERENCES

[1] 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, doi: 10.1109/CVPR.1998.698614.