LAB-4 Histogram-based Object Tracking

Kush Gupta and Sergio Avello Largo

I. INTRODUCTION

This lab report contains routines for histogram creation [1] in different color spaces like RGB, HSV and gray scale, and computation of gradient features [2] like histogram of gradients (HOG), for the region of interest and a model in a given video sequence. Later, an algorithm is developed to track an object by selecting a candidate with the smallest Battacharyya distance between the model and the candidates. A part of the main functionality (histogram creation in different color spaces) for this lab is adopted from "Elliptical Head Tracking Using Intensity Gradients and Color Histograms" [3].

The code is written in C++ using the OpenCV library and designed on Ubuntu using Eclipse IDE. All the tasks are performed on the provided video sequences, which are captured from a still camera. Those videos belongs to AVSA Lab-4 dataset. Firstly, a specific color channel (for example R in RGB or H in HSV) in a particular color space or the gradient feature (HOG) is selected. An object model is initialized (using the first frame and ground truth), and for each frame, the candidate regions (the region of interest) are created around the predicton of the previous frame. Later, using the histogram functionality provided by OpenCV, a normalized histogram is created for the object model and the candidates in the region of interest for the selected color channel or gradient feature. For tracking, the model and the candidates are compared using the Battacharyya distance. Finally, the candidate having the minimum Battacharyya distance with the object model is selected as the prediction.

II. METHOD AND IMPLEMENTATION

A. Color space and channel selection

The first step in the implementation for the object tracking based on color is to choose a suitable color feature (for example R channel in RGB or B channel in RGB or H channel in HSV space). Selection of a suitable color space is necessary in order to analyze, create histogram and track the object of interest. Generally, H and S channels from the HSV feature space are robust to photo-metric changes. Hence, H and S features are widely used in such tracking algorithms. For this implementation we can choose a color space (RGB, HSV or gray scale) by assigning a value to the 'feature' variable in the code (line 53). Assigning a value to 'feature', as "RGB" implies RGB color space is selected and "HSV" implies HSV color space is selected. After selecting the color space, we have to choose a particular color channel in that color space for creation of the histograms. We can do this by assigning different values, to the variable 'clr_channel' in the code (line 54). It is possible to assign a value of "R" for red channel, "G" for green channel and "B" for blue channel in the RGB color space. Also, we can choose "H" for hue and "S" for saturation in the HSV color space. For analysis in gray scale, 'GR' value can be assigned to the 'feature' variable.

B. Histogram Creation

After choosing a suitable color channel in a particular color space, we are almost ready with the creation of histograms for the object model and for the candidates in the region of interest. A histogram is a graphical representation that organizes a group of data points into a particular range known as bins. The histogram condenses a data vector into an easily interpreted visual by taking many data points and grouping them into bins [1]. One of the most important factors while creating a histogram is the number of bins. In the implementation, the number of bins, can be modified by 'histSize' variable in the code (line 55). Histograms were created using calcHist() [1] OpenCV function, and normalized using normalize() opencv function.

C. Histogram of Gradients

The histogram of oriented gradients (HOG) is a gradient feature descriptor widely used in various gradient-based tracking algorithms. This technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, however the difference is that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy [7]. In this implementation, the HOG descriptors are computed using the functions from the HOGdescriptor class [2] of OpenCV. An object is created in order to utilize the functions of the HOGdescriptor class. The patch (region of interest) is resized to 64 x 128 if its size is different than 64 x 128. Finally, The resized patch is given as an input to the compute() function in order to compute the feature vector for the object model and the candidates in the region of interest. In the implementation, to use HOG feature, set the 'feature' variable to 'GR' (line 53) and the HOG_feature to true (line 56) (default value is false). Also, we can control the bin size using histSize variable (line 55).

D. Generation of candidates

The generation of candidates for this project is based on the grid approach proposed in [6]. The basic idea is

to create different candidates around the location of the previous prediction following a squared grid structure. This task is mainly controlled by two parameters: GRID_SIZE that sets the number of 'layers' of candidates around the the previous prediction (GRID_SIZE = 1 creates 9 candidates, GRID_SIZE = 2 creates 25 candidates...) and STEP that sets the distance (in number of pixels) in the horizontal an vertical axis between the location of two consecutive candidates.

III. CODE DESCRIPTION

In the submitted zip file there are 3 folders (T42, T43, T44) with the corresponding sources files for each task. Inside the src folder of each task, the file with the word main in its name acts as the main file for the project, where the different parameters can be modified.

The main file of T42 shows the results of merging task 4.1a and 4.1b without changing any parameter for the video 'bolt1.mp4'. For T43, running the main file shows the obtained results for the video 'car1.mp4' with the best parameter configuration. For T44, after running the main file the results for the video sequence 'basketball.mp4' with the best parameter configuration can be seen.

A. Running the code

In order to compile, link and run the task, it is only needed to run make in the terminal (file inside the src folder) and then run ./(built object).

IV. DATA DESCRIPTION

A. Task 4.2

For task 4.2, the 350 frames video 'bolt' was provided. It was captured from a high definition camera. The video contained a scenario where 8 athletes were running on a racing track. The task is to track Usain Bolt.

B. Task 4.3

For analyzing task 4.3, two video sequences 'car1' with 400 frames and 'sphere' with 201 frames were provided. In the video 'car1.mp4' a car has to be tracked and the video has jitter problem probably because the camera is mounted on a moving object. In the video, 'sphere' a ball has to be tracked.

C. Task 4.4

For performing this task two sequences were used: 'basketball' which has 500 frames and shows a basketball game where the object to track is the basketball player Rajon Rondo and 'ball2' which has 41 frames and shows a football ball entering a goal. The object to track is a ball.

Video	Processing time (ms/frame)	Tracking performance
bolt	0.4679599	0.144531

TABLE I: Quantitative results for Task 4.2

Color features	**Gray**	R	G	B	H	S
sphere Processing time (ms/frame)	**8.85**	9.89	8.87	9.91	15.55	16.51
sphere Tracking performance	**0.53**	0.48	0.55	0.49	0.204	0.371
car1 Processing time(ms/frame)	**5.57**	0.79	2.89	6.21	10.87	10.16
car1 Tracking performance	**0.50**	0.0037	0.082	0.41	0.073	0.36

TABLE II: Quantitative results for different runs to analyze the behaviour of different color features. The parameters GRID_SIZE = 4, histSize = 16, STEP = 3 were constant.

V. RESULTS AND ANALYSIS

A. Task 4.2

Figure 1 shows the visual result obtained for frame 25 after merging tasks 4.1a and 4.1b for the provided video 'bolt' with the original parameter configuration: bin size (histSize) of 32 for Gray color space, GRID_SIZE = 1 and STEP = 5. Until around frame 150, the results were fairly reasonable but from there, they got worse due to the complexity of the scene. The quantitative results can be seen in Table I.

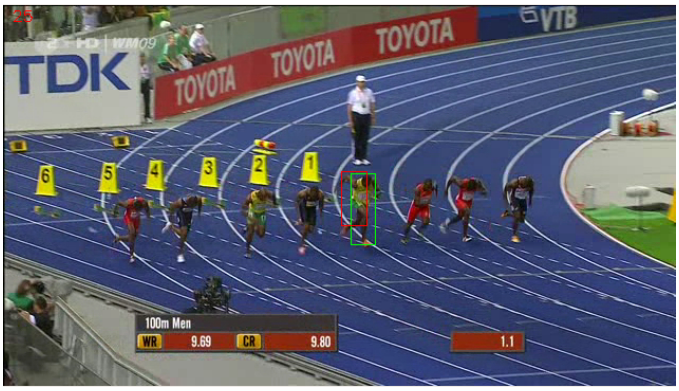

Fig. 1: Visual results for frame 25.

B. Task 4.3

In order to analyze the different color features (e.g. Gray, H, S, R, G, B) for the video sequences, 'sphere.mp4' and 'car1.mp4' different runs were performed on different color features where GRID_SIZE = 4, histSize = 16 bins, STEP = 3 were kept constant. It can be seen on table II that gray color feature gave the best results in terms of processing time (ms/frame) and tracking performance for both the video sequences.

Also, for analyzing the tracker's performance with respect to different number of candidates different runs were carried out for different number of candidates. For all the runs, the gray color feature was used, histSize = 16 and STEP = 2 was kept. The results obtained can be seen in table III. 169

GRID_SIZE (number of candidates)	1 (9)	2 (25)	3 (49)	4 (81)	5 (121)	6 **(169)**
sphere-Processing time (ms/frame)	0.94	2.4	3.55	5.62	8.3	**12.42**
sphere- Tracking performance	0.32	0.33	0.36	0.45	0.49	**0.52**
car1-Processing time(ms/frame)	1.06	2.12	3.89	6.27	9.3	**12.98**
car1- Tracking performance	0.34	0.38	0.39	0.42	0.44	**0.48**

TABLE III: Quantitative results for different runs to analyze the tracker's performance for different number of candidates. The constant parameters were STEP = 2, histSize = 16 and gray feature.

STEP	3	4	5
sphere-Processing time (ms/frame)	12.10	10.6	10.59
sphere- Tracking performance	0.55	0.55	0.56
car1-Processing time(ms/frame)	10.64	10.74	4.92
car1- Tracking performance	0.49	0.22	0.14

TABLE IV: Quantitative results obtained for different step sizes (3, 4, 5) for video sequences 'sphere' and 'car1'.

candidates showed good performance in terms of processing time (ms/frame) and for tracking performance for both video sequences.

Finally, a study to determine an optimal STEP value for the video sequences 'sphere' and 'car1' with the constant parameters (except STEP) mentioned on the previous paragraph was performed. The results can be observed in table IV. STEP = 3 gave good tracking performance in terms of processing time and performance for both sequences.

Hence, the obtained best parameters for the video 'car1.mp4' in the task 4.3, for histSize = 16 were: **feature = 'GR' (GRAY feature), GRID_SIZE = 4 (81 candidates) and STEP = 3** and its corresponding **processing time (ms/frame) is 5.57 and tracking performance is 0.504**. The results for the video sequence 'car1.mp4' can be seen in Figures 2 and 3 and the results for 'sphere.mp4' can be observed in Figures 4 and 5.

C. Task 4.4

In this task a wide study over the gradient-based tracker using the HOG features will be carried out. The results obtained with different number of bins and candidates for the 'basketball' and 'ball2' videos will be discussed and analyzed.

1) basketball: This video is fairly complex, it has many people moving in different directions that appear and disappear. In addition, the camera is not fixed, it moves accordingly with the action in the scene.

First, the performance of the algorithm with different number of bins will be tested. For this first experiment GRID_SIZE will be fixed to 5 and STEP to 1. The numeric results can be seen in Table V.

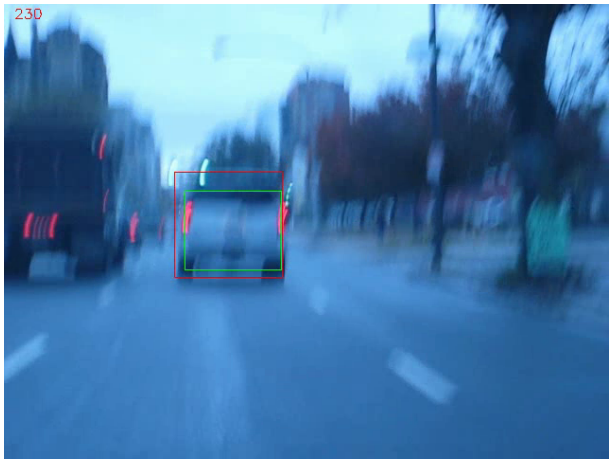

Fig. 2: Frame 230, visual result obtained for sequence 'car1', with parameters feature = GR (gray), STEP = 3, GRID_SIZE = 4 (81 candidates). It can be observed that the tracker is tracking the car accurately.

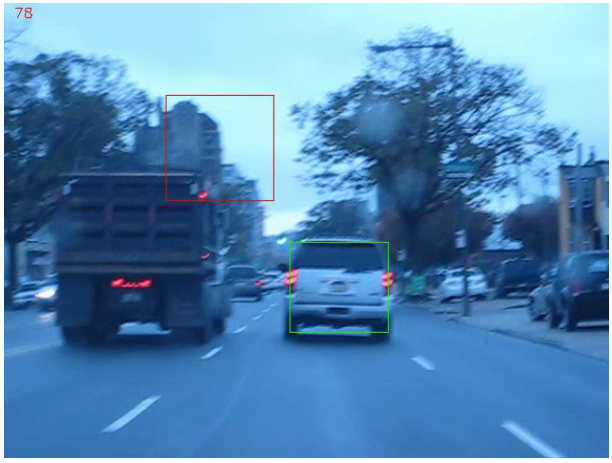

Fig. 3: Frame 78, visual result obtained for sequence 'car1', with parameters: feature = R (red channel), STEP = 3, GRID_SIZE = 4 (81 candidates). It can be observed, that the tracker lost the car.

For studying the behaviour of the algorithm with different number of candidates, we kept fixed STEP = 1 and histSize = 12 (since the best results for the previous experiment were obtained with it). The numeric results are shown in Table VI.

After this analysis, the best results were obtained with the following configuration: **GRID_SIZE = 2 (25 candidates), STEP = 1 and histSize = 12.** However, the results are, in average, really poor. That is mainly due to complexity of the sequence and the problematic caused by using as model the same object (defined in the first frame) along the whole video.

Regarding the visual results, an accurate tracking example can be seen in Figure 6 (with GRID_SIZE = 2 (25 candidates), STEP = 1 and histSize = 12) and Figure 7 shows a bad tracking performance (with GRID_SIZE = 5

Fig. 4: Frame 47, visual result obtained for sequence 'sphere.mp4', with parameters: feature = GR (gray), STEP = 3, GRID_SIZE = 4 (81 candidates). It can be observed, that the tracker is tracking the ball properly.

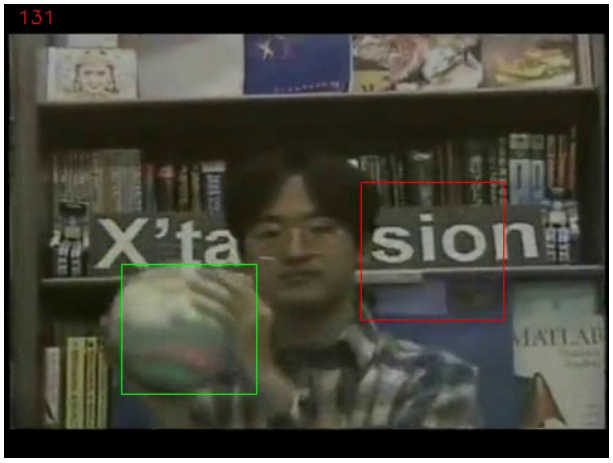

Fig. 5: Frame 131, visual result obtained for sequence 'sphere'.mp4, with parameters: feature = H (H channel), STEP = 3, GRID_SIZE = 4 (81 candidates). It can be observed, that the tracker lost the ball.

(121 candidates), STEP = 1 and histSize = 8).

2) ball2: Compared to 'basketball', 'ball2' is quite simpler, there is a static scene with just one moving object (a small ball entering a goal). The same experiments performed for 'basketball' in the previous subsection will be carried out next. For testing different number of bins GRID_SIZE will be fixed to 5 and STEP to 1. The numeric results can be seen in Table VII.

Since these results were not as good as expected, STEP was set to 3 in order to try to follow the fast movement of the ball and the same experiment was performed (still with GRID_SIZE will = 5). Table VIII summarizes these new results.

Now, the behaviour of the algorithm with different number of candidates will be studied. For that, STEP was set again

histSize	Processing time (ms/frame)	Tracking performance
4	141.581	0.0859733
8	10.539	0.0053475
12	141.481	0.103203
16	139.728	0.0901127
20	149.529	0.0900634
24	143.198	0.0898681
28	147.779	0.0854601
32	142.995	0.0859733

TABLE V: Quantitative results for different number of bins for 'basketball'.

GRID_SIZE (number of candidates)	Processing time (ms/frame)	Tracking performance
1 (9)	11.2842	0.115326
2 (25)	30.7754	0.181933
3 (49)	55.7186	0.0631777
4 (81)	95.1761	0.0854546
5 (121)	138.183	0.103203
6 (169)	203.019	0.0869017

TABLE VI: Quantitative results for different number of candidates for 'basketball'.

to 3 and histSize = 20 (since the best results for the previous experiment were obtained with them). The numeric results are shown in Table IX.

After this analysis, the best results were obtained with the following configuration: **GRID_SIZE = 7 (225 candidates), STEP = 3 and histSize = 20**. They are quite accurate until the last frames of the sequence where the ball is confused with some parts of the net.

Regarding the visual results, an accurate tracking example can be seen in Figure 8 (with GRID_SIZE = 7 (225 candidates), STEP = 3 and histSize = 20) and Figure 9 shows a bad tracking performance (with GRID_SIZE = 3 (49 candidates), STEP = 1 and histSize = 12).

histSize	Processing time (ms/frame)	Tracking performance
4	137.006	0.0416483
8	146.369	0.0416483
12	141.933	0.0483413
16	137	0.0498241
20	141.96	0.0498241
24	141.31	0.0498241
28	144.363	0.0498241
32	140.182	0.0498241

TABLE VII: Quantitative results for different number of bins for 'ball2' with STEP = 1.

histSize	Processing time (ms/frame)	Tracking performance
4	138.462	0.196735
8	135.121	0.185898
12	135.237	0.225949
16	140.558	0.225949
20	143.081	0.270371
24	143.164	0.27687
28	141.936	0.271713
32	146.276	0.27687

TABLE VIII: Quantitative results for different number of bins for 'ball2' with STEP = 3.

Fig. 6: Good tracking result for frame 5 of 'basketball' with GRID_SIZE = 2 (25 candidates), STEP = 1 and histSize = 12.

Fig. 7: Bad tracking result for frame 27 of 'basketball' with GRID_SIZE = 5 (121 candidates), STEP = 1 and histSize = 8.

VI. CONCLUSIONS

In conclusion, the first step of histogram based object tracking is to create normalized histograms of any of the different features (in our case color or gradient) for the object model and for the different candidates. Secondly, in order to track the object, a comparison between the model and the candidates is carried out using the Battacharyya distance. Finally the candidate showing the minimum distance is selected as the prediction of the tracker for the current frame. On the other hand, the major drawback of our histogram based object tracker is that it employs a certainly simple object model that highly affects the final performance of the algorithm. The object model is created in the first frame of the sequence and does not change anymore. Since the approach is quite simple, it was impossible to find a parameter configuration that worked in a reasonable manner for all the provided videos. Therefore we tried to adapt the different parameters (type of feature, GRID_SIZE, STEP, histSize) for each of the sequences.

GRID_SIZE (number of candidates)	Processing time (ms/frame)	Tracking performance
1 (9)	12.9253	0.0385119
2 (25)	30.564	0.0557962
3 (49)	60.1781	0.0695894
4 (81)	95.4733	0.0895221
5 (121)	142.141	0.270371
6 (169)	196.761	0.291857
7 (225)	267.457	0.326813
8 (289)	363.738	0.263494

TABLE IX: Quantitative results for different number of candidates for 'ball2'.

Fig. 8: Good tracking result for frame 19 of 'ball2' with GRID_SIZE = 7 (225 candidates), STEP = 3 and histSize = 20.

VII. TIME LOG

Coding: Around 6 hours for merging the code and debugging the errors.
Testing: Around 12 hours of testing the routine to find the optimal set of hyper parameters for different tasks and then analyzing the behaviour of the algorithm.
Report: Approximately 7 hours to write the report, take the screenshots of relevant frames, adding the results and formatting them. Proof reading the report to eliminate the possible typo errors.

REFERENCES

[1] OpenCV Histogram creation https://docs.opencv.org/3.4.4/d8/dbc/tutorial_histogram_calculation.html
[2] OpenCV HOG descriptor https://docs.opencv.org/3.4.4/d5/d33/structcv_1_1HOGDescriptor.html
[3] 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), 1998, pp. 232-237, doi: 10.1109/CVPR.1998.698614.
[4] OpenCV reference manual http://docs.opencv.org/2.4.13.2/modules/refman.html
[5] OpenCV color conversion https://docs.opencv.org/3.4.4/de/d25/imgproc_color_conversions.html
[6] P. Fieguth and D. Terzopoulos, "Color-Based Tracking of Heads and Other Mobile Objects at Video Frame Rates" Proceedings. 1997 IEEE Conf. on Computer Vision and Pattern Recognition.
[7] https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

Fig. 9: Bad tracking result for frame 17 of 'ball2' with GRID_SIZE = 3 (49 candidates), STEP = 1 and histSize = 12.

VIII. APPENDIX

A. Difficulties during individual work

1) Task 4.1a: The main challenge for task 4.1a was to obtain a normalized histogram for a selected color channel (if color feature was selected) or gradient feature (if gradient feature is selected). Also, visualizing the gradient histogram was a bit challenging task too.

2) Task 4.1b: At first it was a little bit tricky to understand the grid structure approach for creating the candidates and then understand how to compare them with the model.

B. Modifications during code merging

Instead of comparing the model with the candidates using template matching (as in Task 4.1b) we used the Battacharyya distance between the histograms of the model and the candidates.

C. Time log for collaborative Work

1) Coding: Session 1: Six hours

The aim for this session was to merge the code from task 4.1a and 4.1b and start working on task 4.2. Initially, Sergio was acting as driver, and Kush was acting as navigator for the first one and half hours and then we switched the roles every one and half hours. During the first session we were struggling to create a new project in eclipse and to make the build configurations for that. After successfully, creating the new project and build configurations, we were able to merge the two codes and started task 4.2.

Session 2: Four hours

For the second session, our goal was to finish the task 4.2 and begin task 4.3. During this session, Kush was acting as driver, and Sergio was navigator for the first hour and then we switched the roles every hour. During this session, we worked and finished the task 4.2, and found the best parameters for the analysis of the video sequence 'bolt'.

Session 3: Four hours

Our goal for this session was to complete the task 4.3. In the beginning, Sergio was acting as the driver, and Kush was the navigator for the first hour and then we switched the roles every hour. During this session, we worked on the task 4.3 and found out the best parameters for the analysis of the video clip 'car1'.

Session 4: Four hours

We planned to finish the task 4.4 in this session. Kush acted as the driver, and Sergio was the navigator for the first one and half hours and then we switched the roles every hour. During this session, we worked on the task 4.4 to determine the best parameters for the analysis of the video clip 'basketball'.

2) Report: Session 1: Three and a half hours

During this session, Kush acted as the driver and Sergio was the navigator for the first hour and then we switched the roles every hour. We wrote the first half of the report (introduction, method and implementation, code description and data description) during the session and proof read it in the end.

Session 2: Three and a half hours

For this session, we aimed to finish the remaining parts of the report. During this session Sergio acted as the driver and Kush was the navigator for the first hour and then we switched the roles every hour. During this session we worked on the remaining parts like (result and analysis, conclusion, time-log and the appendix) of the report and proof read the whole report in the end.

D. Overall impression

Overall, the proposed methodology for collaborative work was interesting. Using the methodology we experienced that our partner can point out the mistakes that we are committing while writing the code or while writing the report. Also, debugging the errors together was quite efficient. However, the major drawback of this strategy was to find a suitable time window between the partners to work together. In our opinion, working together online via Teams was not a very efficient solution.