# LAB 4.1 b Histogram-based object tracking individual work

APPLIED VIDEO SEQUENCES ANALYSIS

Zsófia Sára Budai
budai9902@gmail.com

## I. INTRODUCTION

This lab report describes an implementation for a single-object tracker, which focuses on generating and matching candidates for each bounding box. The implementation utilizes C++ and the OpenCV library on Ubuntu with Eclipse IDE. For tracking, the model and candidates are compared by computing the normalized cross-correlation coefficient (TM_CCOEFF_NORMED) as a distance. The candidate with the minimum TM_CCOEFF_NORMED distance to the object model is selected as the prediction. The goal of this implementation is to generate candidates and match them with a pre-defined model for accurate tracking of a single object.

## II. METHOD

To generate candidates for this project, a grid approach inspired by the methodology proposed in the [1] source paper was employed. The main idea is to create multiple candidates around the location of the previous prediction, following a squared grid structure. In the initial frame, the ground truth value was used to predict the center of the grid.
Two crucial parameters govern this process. First, the ”gridSize” variable determines the number of layers of candidates surrounding the previous prediction. For instance, a ”gridSize” of 1 generates 9 candidates, while a ”gridSize” of 2 generates 25 candidates. It’s important to note that only candidates within the frame are considered valid in this implementation. If any candidates fall outside the frame, they are excluded from further processing and comparison.
The second essential parameter is determined by the ”step” variable, which specifies the number of steps in a unit of the horizontal and vertical axes. If the ”step” value is 1, the candidate locations do not overlap.
Once the candidates have been generated, the objective is to identify the candidate that is closest to the ground truth. This is achieved by calculating the distance between each bounding box and the ground truth, and then selecting the candidate with the minimum distance. For this the TM_CCOEFF_NORMED method is used from the opencv library to determine the distances. The estimated bounding box, which corresponds to the closest candidate, is highlighted in pink on the output video.

## III. IMPLEMENTATION

### A. Functions, variables

The main responsibility of the main.cpp file is to read a video file as a sequence of images, process each frame, and display the frames with bounding boxes. The ground truth bounding box is marked with a light blue color, while the estimated position bounding box is marked with a pink color. After processing each frame, an output video is saved, including the frames with the bounding boxes.Additionally, the ”main.cpp” file calculates the average processing time and performance. The processing time refers to the time taken to process each frame, while performance refers to the efficiency and effectiveness of the algorithm in accurately estimating the bounding box positions. The average processing time and performance metrics are printed out for analysis and evaluation purposes. The utils files contain essential helper functions that are used to process the candidates and estimations, those are listed below.

- readGroundTruthFile and estimateTrackingPerformance are the two prewritten functions for processing the ground truth data and evaluating the performance of the tracking system
- createCandidates function generates a set of candidate bounding boxes based on a predicted bounding box and specified parameters. It creates candidates by shifting the position of the predicted box in both horizontal and vertical directions and checks if the generated candidate falls within the frame boundaries, adding it to the list of candidates if it satisfies the condition.
- compareCandidates function compares the similarity between an object model region and a set of candidate regions in an input frame. It computes the distance metric between the object model and each candidate using template matching, where a lower distance indicates a higher similarity and returns a list that contains the calculated distances.
- step parameter in the createCandidates function determines the spacing or stride between the candidate bounding boxes in both the horizontal and vertical directions.
- gridSize parameter in the candidate generation process determines the number of levels of candidates around the previous prediction.

### B. Running the code

The submitted files adhere to the format specified for the assignment, comprising the requisite source C++ code and corresponding Makefile. To compile, link, and execute the task 4.1.b, simply navigate to the source folder for the corresponding task and run the "make" command in the terminal. This will generate the necessary built object file, which can be executed by running "./main".

## IV. DATA

The algorithms underwent testing on video sequences, created from jpg files from AVSA_lab4_dataset. Specifically for this first task, the "bolt1" folder was utilized, which includes 352 jpg images in the "img" folder and a corresponding "groundtruth.txt" file that provides ground truth values for the object being tracked. The video used for tracking was captured using a high definition camera and features a scenario where 8 athletes are running on a racing track. The specific task was to track Usain Bolt among the athletes.

## V. RESULTS AND ANALYSIS

| step | numCand | Average processing time | Average tracking performance |
|---|---|---|---|
| 1 | 1 | 0.975424 ms/frame | 0.0196267 |
| 1 | 2 | 2.12289 ms/frame | 0.029323 |
| 2 | 2 | 2.19838 ms/frame | 0.0239914 |
| 4 | 4 | 6.9333 ms/frame | 0.163885 |
| 10 | 5 | 9.68677 ms/frame | 0.624913 |
| 10 | 10 | 20.6652 ms/frame | 0.675786 |
| 20 | 10 | 20.2715 ms/frame | 0.719225 |

TABLE I: Table of some results for different gridSize and step values.

The I table presents average processing times and tracking performances for different values of gridSize and step. Generally, increasing the gridSize and considering more candidates improves accuracy, but it also increases processing time. The objective was to strike a balance between processing time and accuracy by finding an optimal combination of gridSize and step values.

### A. Size of the grid

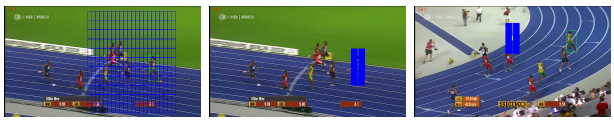

Fig. 1: The step is 2 on the left, 20 in the middle and 30 on the right with numCand = 10 constantly.

The size of the grid area is a crucial aspect of tracking. A grid that is too large can lead to tracking inaccuracies and slower processing, while a grid that is too small can result in drifting, causing the estimated bounding box to gradually deviate from the ground truth. The impact of different grid sizes is illustrated in the 1 figure. In the left image, the grid area is very large, leading to a jumping and inaccurate estimated bounding box with a performance of 0.0361413 and a processing speed of 10.8922 ms/frame. Increasing the step size to 10 improves the accuracy significantly to 0.675786, with a slightly increased processing time of 21.2346 ms/frame. Further increasing the step size to 20 only slightly increases the processing time to 22.2466 ms/frame, while achieving an accuracy of 0.719225. However, setting the step size too large, such as step = 30, results in a drop in accuracy to 0.470917 and the tracking is missed, as evident in the right image.

### B. Number of the candidates

Selecting an appropriate number of candidates is important. Considering too many candidates increases computational time, while using too few candidates leads to drifting in the estimation over time, as depicted in the left image. It is crucial to strike a balance between the number of candidates and accurate tracking of the target. In Figure 2, it is evident that using only 9 candidates resulted in a drifted bounding box from the runner in the left image. However, in the middle and right images, where 81 and 441 candidates were considered respectively, the detection is accurate. Despite the significant increase in processing time from 7.58799 ms/frame for 81 candidates to 27.7612 ms/frame for 441 candidates, the improvement in accuracy was moderate, with the accuracy increasing from 0.466922 to 0.675786.

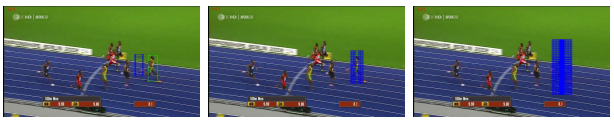

Fig. 2: Number of candidates is 1 on the left, 4 in the middle and 10 on the right with step = 10 constantly.

## VI. CONCLUSIONS

In conclusion, the objective of implementing the generation and matching of candidates for a single-object tracker was successfully achieved. Different grid sizes, numbers of generated candidates, and distance measurement methods were tested, and it was observed that increasing these parameters generally led to better tracking performance but at the expense of increased processing time. Overall, the results demonstrated the effectiveness of the proposed approach in accurately tracking a specific object in a given video sequence.

## VII. TIME LOG

Approximately 9 hours were dedicated to this project.

- Studying: 0.5 hours
- Coding, debugging code: 5 hours
- Testing, screenshots: 0.5 hours
- Report: 3 hours

## REFERENCES

[1] P. Fieguth and D. Terzopoulos. 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*, 1997.