Object Tracking & Application

Kanvaly FADIGA[1] **and Yelan MO** [2]

[1]Ecole polytechnique; [2]Ecole polytechnique

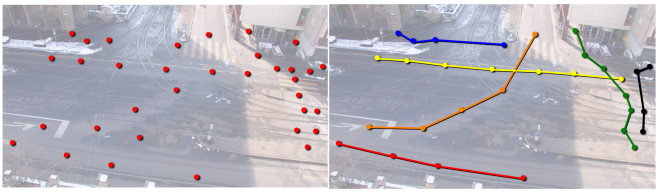
Along this semester we've studied a lot of computer vision technics,. So for the course project, we chose to work on object tracking, which is an important computer vision problem with numerous real-world applications, including human-computer interaction, autonomous vehicles, robotics, motion-based recognition, video indexing, surveillance and security. Our goal is: firstly, build a multiple object tracker by different methods of detection and tracking. Then choose an application domain. We chose to work on surveillance and security. We segment region of interest in a video scene and track motion, position and time spend of any new objects there. Then after a certain amount of time, we start identification (if it is a human or a car) in an asynchronous way and launch alarm given the result of identification.

Object Tracking | Tracking Algorithms | Surveillance | Security

Overview & Road Map

In the tracking part, our goal is to find an object in the current frame given that we have tracked the object successfully in all (or nearly all) previous frames. Since we have tracked the object up until the current frame, we know how it has been moving. In other words, we know the parameters of the **motion model**. But we have more information that justify the motion of the object. We know how the object looks in each of the previous frames. In other words, we can build an **appearance model** that encodes what the object looks like. In few word : *The motion model predicts the approximate location of the object. The appearance model fine tunes this estimate to provide a more accurate estimate based on appearance.*. These two models are often corrected by detection.

Multiple object tracking add one more step: data association, whose purpose is to assign to each detection a tracker.



(a) Object Detection **(b)** Data Association

Fig. 1. Multiple object tracking steps [(7)]

The problem can be taken as online tracking, where only current and previous frames are available or offline/batch tracking, when all frames of video sequence are available. In our case we will only consider online methods.

A multi-object tracker is simply a collection of single object trackers. these single tracker receive whether or not a new detection from the data association model to update their setting. So we divided our task into 3 main steps. the first one will be to create a single object tracker (Appearance and motion). Then add one layer on top which is data assigment solver using hungarian algorithm. Finaly we will use data record on track for different applications such as surveillance.

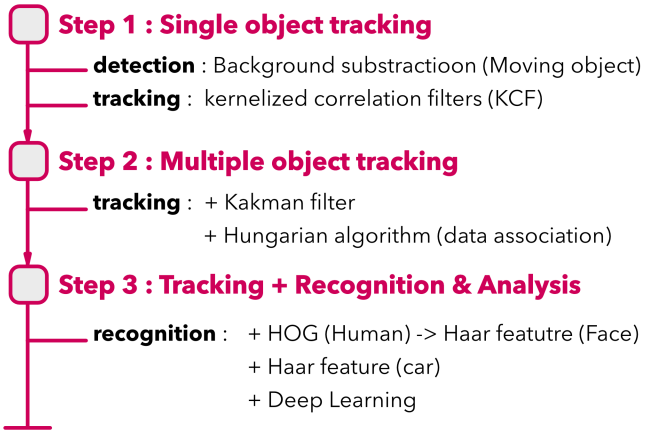


Fig. 2. Roadmap of the project

1. Step 1: Single object tracking

Detection. First step in the process of object tracking is to identify objects of interest in the video sequence and to cluster pixels of these objects. Since moving objects are typically the primary source of information, most methods focus on the detection of such objects. (8) give a detailed explanation of various methods. He present 3 main methods: **background subtraction, optical flow, frame differencing.**. With regard of computational time and accuracy, we choose background subtraction.

As the name suggests, background subtraction calculates the foreground mask performing a subtraction between the current frame and a background model, containing the static part of the scene or, more in general, everything that can be considered as background given the characteristics of the observed scene. Background modeling consists of two main steps: Background Initialization and Background Update. In the first step, an initial model of the background is computed, while in the second step that model is updated in order to adapt to possible changes in the scene.

There are many methods of background subtraction, so we evaluate them and choose the most robust[Figure 3]. The best one in our test was GSOC Background Subtraction. it was implemented during GSOC and was not originated from any paper.

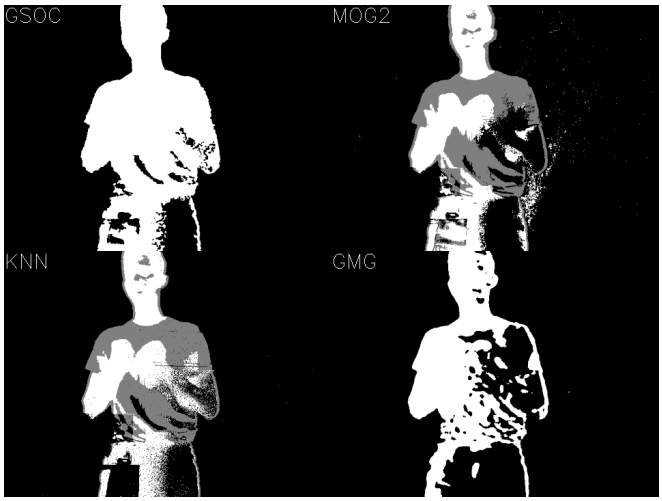


Fig. 3. Compairision of Background substraction model of opencv. (top-left) GSOC, (top-right) MOG2, (bottom-left) KNN and (bottom-right) GMG.

Tracking. OpenCV comes with a new tracking API that contains implementations of many single object tracking algorithms. There are 8 different trackers available in OpenCV 4—BOOSTING, MIL, KCF, TLD, MEDIANFLOW, GOTURN, MOSSE and CSRT.

The most robust after our test was CRST [(5)]. even if it is lower fps(25) we see that it is very robust. moreover, he updates himself the tracking bounding box. As contrary, we have MOSSE which is very fast and give also good result but suffer from occlusion problem.

So once we have done that we can now move to the next level and try to tracks more than one object at the time.

2. Step 2 : Multiple object tracking

Detection. We did not change the detection methods for this part. Meanwhile we develop deep learning approache but they are very slow, detect by category and not only moving object.

Tracking. To associate detection and tracks, we need to estimate similarity or affinity between them. We can use several features to compute affinity scroll. This visual similarity, motion similarity and interaction between objects and objects within the scene. The simplest form of affinity is overlap between detection in the current frame and in the previous frame.To associate detection and tracks, we need to estimate similarity or affinity between them. We can use several features to compute affinity scroll. This visual similarity, motion similarity and interaction between objects and objects within the scene. The simplest form of affinity is overlap between detection in the current frame and in the previous frame. this simple approach has been demonstrated in the recent high-speed tracking by the detection result [(4)].

IOU Tracker. The idea of the algorithm is very simple. For each track, we select detection with the highest IOU. If this IOU is larger than the threshold we, add it to the track and remove it from the list of unassociated detections. If the best overlap is still lower than the threshold, then finish the track. If the track is too short or had no higher confidence directions we consider the track false positive and then remove it from the final list of tracks. This is all. Then we update every single tracker with its associated data. But the problem with this approach is that if the detector fails a frame and misses the object, then tracking of this object is stopped.

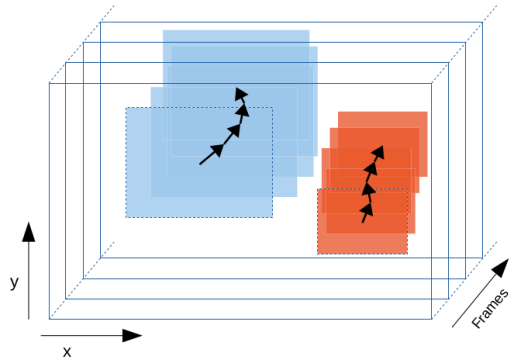


Fig. 4. Basic principle of the IOU Tracker: with high accuracy detections at high frame rates, tracking can be done by simply associating detections by their spatial overlap between time steps. [(4)]

SORT Tracker. The problem of IOU tracker can be alleviated if it can use predicted object position instead of a missed detection in this frame. For example, we can use kalman filter to predict objects based on its positions in previous frames. Then we can associate detections in current frame these predictions from previous frames. If an object is not detected in the current frame, you won't finish the track immediately but continue to track these predictions for several frames. We hope that eventually the object will be detected. And if predictions are good enough, we will successfully associate this detection with predicted position and assume tracking. Because predictions are less precise than detection, we should replace simple grid based with old method. It's the optimal Hungarian algorithm for the assignment problem, as it was done in **simple online in real time tracking (SORT)** [(3)]. It's the one we used.

3. Step 3 : Recognition & Analysis

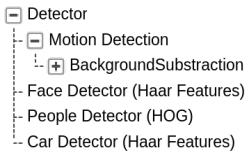
Now that we have a good tracking pipeline, we want to go further. We consider each bounding box as a region of interest. if a box is still in the segmented area of the image after some amount of time we start giving him more attention. asynchoniously, we launch human, face and car detector for each of these bounding boxes. This analysis can be done in c++ or in python (wrapping) for some deep learning cases.

Application. - Restricted zone surveillance - Parking surveillance

4. Implementation details

Our code splits into 3 main folders : **Detection, Tracking and classification.**. For all of them we write in addition our core program, we write the examples to allow us to do some experiment. We will details all of them below.

Fig. 5. Our detectors



Detector. We code 4 detectors as it is written to the left. All these detector herited from the same class **DetectorBase**. Written in this way way it was very easy to add a new detector and change them as we wish. The background substraction methods we talk about before is use on motion detector. Face Detector and Car detector use a pretrain model of cascade classifier.

```
class DetectorBase
{
protected:
    int minObjSize;
    regions_t detected_regions;

public:
    DetectorBase(int minSize);
    virtual ~DetectorBase(void);

    virtual bool Initialize(const config_t& conf)=0;
    virtual void Detect(UMat& frame)=0;
    const regions_t GetDetection() const;
}
```

Listing 1. DetectorBase class

Tracking. We first code tracker for single object by trying opencv 8 API trackers. Then, gap between us and extension to multiple object was data association methods and kalman filter.

Kalman Filter. As the paper (3) recommend us to use Kalman filter that make prediction on the bounding box state. We start with a mouse tracker with help of (1) code which use OpenCV Kalman linear filter. the extention is target modelled as: $[u, v, s, r, \dot{u}, \dot{v}, \dot{r}]$. where u and v represent the horizontal and vertical pixel location of the centre of the target, while the scale s and r represent the scale (area) and the aspect ratio of the target's bounding box respectively. we 6 state variables, 4 measurements $[u, v, s, r]$ given by detection. So have to configuration OpenCV Linear Kalman.

Hungarian Algorithm. (Ma) implementation of the algorithm in C++ is a reference. the algorithm is not diffucult conceptualy. The assignment cost matrix we use was intersection-over-union (IOU) distance (called also jaccard distance) between each detection and all predicted bounding boxesfrom the existing targets.

Recognition & Analysis. Because we want as best as we can do to be close to real time we choose to run rocognition and analysis process in a parallel thread. This thread is a while loop that contain a global queue list than received frequently data to analyse. this data is the ID of a track and a shoot of region of interest. When he finish analysis, he save in a *map* the category of the object. This way at every frame each track in alert mode check if his data have been identify. if so he show up a text idicating who it is.Here is where deep learning was super usefull. instead of applying several different detector, we use Yolo pretrain model thanks to OpenCV DNN (9). In addition we use Homography to map the scene into a 2D map to have a top view overview.

The next step is to associate all this in a well written C++ code. It is at this moment the massive code of (10) served us as good model of code organisation. He try to make a kind of huge code with every imaginable possibility together but that's not what we want. We have already made our test and we knew what we want. But this requires organisation, to have easy communication among classes.

After writting our code and putting every part together we observe that the computational time was high. it was very slow. first we though it was because of CSRT as tracker (still slow down with the solution). Meanwhile we see that many people where using UMat instead of Mat so we decide to try it. surprisly everything become fast. we saw on stackoverflow that: *the actual UMat data can be located in a regular system memory, dedicated video memory, or shared memory. that enables the same APIs to be implemented using CPU or OpenCL code, without a requirement to call OpenCL accelerated version explicitly.*

5. Results

(a) GSOC Background model at begining **(b)** GSOC Background model after 10s

Fig. 6. GSOC Background Modelling

. After the implementation, we test them and see what the cons and pros of using them. First the Motion detection using background substraction. We use OpenCV implementation of GSOC, MOG2, KNN and GMG [Fig 3]. GSOC has we said gives the best results, he has less noise and also he is less sensible to camera movements pertubation. However, sometime he need 10s to remove some noise due to initialisation. Then, the People detector using HOG gives a lot of false positive detection. Car detector using Haar features gives the same kind of result. Face Detector give good prediction but came only with pretrain model only for front face. We try also deep learning methods like fast-rcnn, and yolo. They are very good at classification and detection for a little range of object. Howerver, they are really slow and would restrict us to track object this classifier only knew. We will use their strength for other application.

. We tested the capabilities of the detector and the tracker on this video with a lot of occlusion [Figure 7]. The detector work pretty well. CSRT Tracker does better with occlusions than MOSSE.

. Our program at the begining ask you to select a area on the image where you want to track object[Figure 8]. then you can additionnaly draw a polygon of 4 points (red area) to get a homographic projection of tracked point into it.

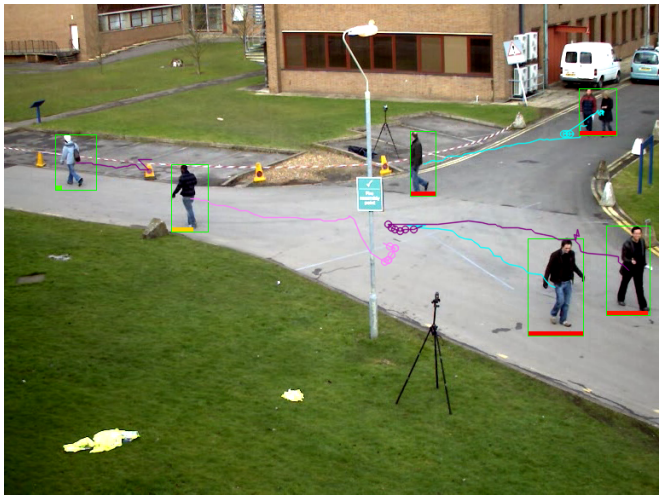


Fig. 7. Detection + Tracking. tracked regions are those green bounding box. box trajectory are record by tracker. each box show bar loadings representating the elapsed time compared to the maximum time.

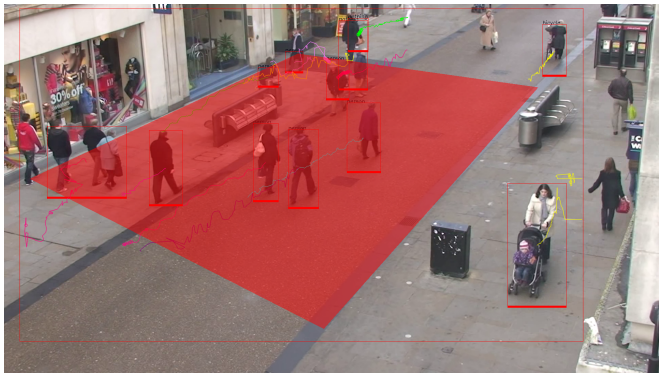


Fig. 8. Scene with homography area selected

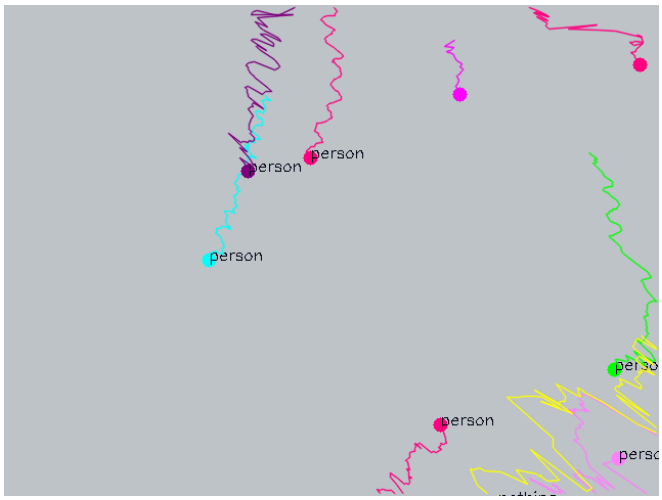


Fig. 9. Projection of each tracked point in homographic area into 2D plane

Next Step. Our detector work only for moving object. If an object stay in the same place for a long time we lost trace of him (kalman stop after certain time of non detection). So to takle this we were thinking about combiniting detection with deep learning by combining him with motion detection. We try also saving regions where alarm are launch and analyse them in python to apply more elaborated deep learning model. Work on them and load using opencv dnn. we were working on car platenumber recognition and head pose estimation (2).

6. Conclusion

This project was a very good way for us to practice on a subject on which there is still a lot of attention in computer vision. We deepened our knowledge on several topics that we had discussed in class and discovered new ones that were even more interesting. It was also a good way to improve our programming skills in C++. We are proud of ourselves because we achieved almost all of our goals and had time to explore other areas of application of our program. The notions acquired in this course and with this project represents for us a good base of skills to face the future project where computer vision will intervene.

References

[1] Armon and Roy (2011). Simple kalman filter for tracking, http://www.morethantechnical.com/2011/06/17/simple-kalman-filter-for-tracking-using-opencv-2-2-w-code.

[2] Benfold, B. and Reid, I. (2008). Colour invariant head pose classification in low resolution video.

[3] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. *CoRR*, abs/1602.00763.

[4] Bochinski, E., Eiselein, V., and Sikora, T. (2017). High-speed tracking-by-detection without using image information. pages 1–6.

[5] Lukezic, A., Vojír, T., Cehovin, L., Matas, J., and Kristan, M. (2016). Discriminative correlation filter with channel and spatial reliability. *CoRR*, abs/1611.08461.

[Ma] Ma, C. Hungarian algorithm repo, https://github.com/mcximing/hungarian-algorithm-cpp.

[7] Manen, S., Timofte, R., Dai, D., and Van Gool, L. (2016). Leveraging single for multi-target tracking using a novel trajectory overlap affinity measure. pages 1–9.

[8] Parekh, H. S., Thakore, D. G., and Jaliya, U. K. (2014). A survey on object detection and tracking methods.

[9] Smorodov, A. Multitarget (multiple objects) tracker, https://docs.opencv.org/master/da/d9d/tutorial$_dnn_yolo.html$.

[10] Smorodov, A. Multitarget (multiple objects) tracker, https://github.com/smorodov/multitarget-tracker.