Video Surveillance for Road Traffic Monitoring

Facundo Ferrín, Agustina Pose, Noa Mor

***Abstract*— Video analysis refers to the use of computer vision techniques to analyze the content of videos and extract useful information from them. Among its applications is the analysis of medical videos to segment moving organs, such as the heart; surveillance and road control; robotic guidance for navigation; sports analysis, and many others.**

In this project we implemented several video analysis techniques to solve the problem of multi camera multi tracking. It consists in tracking multiple objects at the same time using more than one camera. As dataset we use the videos provided by the AI City Challenge and present three different approaches to solve the problem, as well as the metrics of one of them. The code can be found in this repository.

I. Introduction

This project was organized in the context of the AI City Challenge, which encourages research and development into techniques that rely on transfer learning, focusing on Intelligent Transportation System (ITS) problems.

Our goal was to implement object tracking algorithms for single and multiple cameras. With this goal, we used the data provided by the AI City Challenge.

A. AI City Challenge

Immense opportunity exists to make transportation systems smarter, based on sensor data from traffic, signaling systems, infrastructure, and transit. Unfortunately, progress has been limited for several reasons — among them, poor data quality, missing data labels, and the lack of high-quality models that can convert the data into actionable insights There is also a need for platforms that can handle analysis from the edge to the cloud, which will accelerate the development and deployment of these models. [1]

The AI City Challenge Workshop is meant to help address these challenges by encouraging research and development into techniques that rely less on supervised approaches and more on transfer learning, unsupervised and semi-supervised approaches that go beyond bounding boxes. It focuses on Intelligent Transportation System (ITS) problems, such as:

- City-scale multi-camera vehicle tracking
- City-scale multi-camera vehicle re-identification
- Traffic anomaly detection – Leveraging unsupervised learning to detect anomalies such as lane violation, illegal U-turns, wrong-direction driving, etc.

The challenge solicits original contributions in these and related areas where computer vision and specifically deep learning have shown promise in achieving large-scale practical deployment that will help make cities smarter.

B. Multiple object tracking with single camera

Object tracking is a complex task, composed by different subsequent steps:

1) Detecting multiple objects in each frame separately (this is achieved by object detection algorithms). The output from this step is a list of bounding boxes per frame, indicating the position of the detected objects.
2) Connecting the positions of the same object along multiple frames. In this way, we identify the same object along the whole video and create a ”track”.
3) In case of multiple cameras, the last step is to relate the same objects detected in different videos.

For the first part of our project, we needed to develop tools that would allow us to track multiple objects from a single camera detection.

As an input, we used the detections obtained from two object detection networks: *YOLO3* and *Mask RCNN*. In this case, the networks were not implemented by us: we just used the list of bounding boxes that were provided in the Challenge website.

In order to track objects from a single camera, we explored two methods: Maximum intersection over union and Kalman filtering.

C. Multiple object tracking with multiple cameras

Once we were able to track multiple objects from a single camera, as a second step we needed to relate the same object detected from different cameras. As an input for this task we used the detections acquired in the previous section.

For this goal, we explored two different kind of approaches:

1) Computing similarity functions.
2) Computing the homographies that related the different cameras present in the same sequence.

The methods comprising the first approach obtain information from frame features. We need to read each frame and its corresponding bounding boxes, and extract information from the portion of the image contained in those boxes. A positive characteristic, is that these methods allow visual direct feedback. On the other hand, there are some major drawbacks: these methods are more computationally expensive, and they are also sensitive to differences in illumination or camera pose / orientation.

We explored mainly two different methods for computing similarity functions: histogram comparison and matching SIFT keypoints.

II. Method

This project is divided in two main tasks: tracking of multiple objects with a single camera, and also with multiple cameras. In our case, we needed to respect the order between these two tasks, since we used the output from the first one as input for the second one.

The py-motmetrics library was used to calculate metrics. This library provides a Python implementation of metrics for benchmarking multiple object trackers (MOT). Among these metrics is IDF_1, which measures the ratio of correctly identified detections over the average number of ground-truth and computed detections and is calculated on the basis of the formula 1:

$$IDF_1 = \frac{2 \cdot IDTP}{2 \cdot IDTP + IDFP + IDFN} \tag{1}$$

where:

- IDTP: true positive matches after global min-cost matching.
- IDFP: false positive matches after global min-cost matching.
- IDFN: false negatives matches after global min-cost matching.

A. Single camera tracking

For the first task, we implemented two different tracking methods: Maximum intersection over union and Kalman filtering. For both cases, we also tried to improve the obtained results by adding post processing steps. As an input, we used the defections obtained from two object detection networks: *YOLO3* and *Mask RCNN*, which were already provided by the AI City Challenge website.

1) Maximum Intersection over union: We calculated all IOUs of all possible pairs between successive frames. The pairs were sorted in descending order according to the IOU value with a threshold of $IOU > 0$. First we linked those with the highest score, and mark them as the same track. Bounding boxes in the current frame that didn't match any of the boxes in the previous frame are marked as a new track with a new track id. Bounding boxes from the previous frame that didn't match any of the boxes in the current frame are considered as a track that terminated.

2) Kalman filtering: Kalman Filter addresses the problem of extracting the useful signal from noisy measurement variables. R. E. Kalman (the author of this method) adopted the idea of state space representation and incorporated it into statistical estimation theory for the development of this filtering technique. Based on the statistical characteristics of the system noise and measurement noise, the measurement variables are used as input signal, and the estimation variables that we need to know are the output of the filter. The whole filtering process is composed of a prediction equation and an update equation, which also serve to portray the entire system: [2]

Prediction equation:

$$X(n) = F \cdot X(n-1) + V_q(n-1) \tag{2}$$

Update equation:

$$Y(n) = H \cdot X(n) + V_p(n) \tag{3}$$

where $X(n)$ and $Y(n)$ are the estimated state variable and measurement variable, respectively. F is the state transition matrix and H is the measurement matrix. $V_q(n)$ and $V_p(n)$ represent the system noise and measurement noise, respectively.

The optimal estimation can be achieved by Kalman filtering only if the following three assumptions are met simultaneously: the state model and measurement model are both linear; the model is designed well to fit the actual system; the noise is additive Gaussian noise. In actual or real systems, the second and third assumptions are quite difficult to be met. The designed model will deviate, more or less, from the actual system. Additionally the noise environment is more complicated in real applications.

In order to track objects with a Kalman filter, we got inspired in the tutorial [3] by Amaia Salvador. The algorithm uses the Hungarian algorithm for box association based on Intersection over Union, and uses Kalman filtering for tracking. The output of this implementation are a set of refined bounding boxes and a track id per object. The tracker finds these matches, and adjust the bounding box coordinates based on tracking information.

3) Post Processing: To improve our results we created 3 different post processing possibilities that take into account the track characteristics:

1) Continuous motion: connect tracks that has similar motion. We created a synthetic track for 10 frames after the last frame of each track, under the assumption of linear motion (orientation and zoom) and constant velocity. Velocity, orientation and zoom are calculated as an average of the last 5 frames of each track. After creating those synthetic tracks we check again for matches with "Maximum IOU" II-A.1
2) Track length: we eliminate tracks with less than 10 frames.
3) Motion variation: eliminating static tracks. After calculating the standard deviation of the motion of the center of the bounding boxes. Tracks that exceed 15 pixels std were removed. The threshold was set after an analysis of the ground truth in Sequence 3.

B. Multiple camera tracking

For the second task, we needed to match in different cameras the tracks that were already identified from the previous step.

First of all, in order to accomplish this we needed to convert the frames from each camera to the same temporal reference. This would allow us to compare corresponding frames from different cameras. In order to do this we used the temporal calibration provided by the AI City challenge.

Once we were able to compare corresponding frames from different cameras, we implemented three methods in order to match the tracks between different cameras. We worked with algorithms that computed similarity functions such as histogram comparison and SIFT matching; and we also computed the homography that related the different cameras.

1) Histogram comparison: Our first attempt for tracking objects along multiple cameras consisted in carrying out a histogram comparison.

Once we had adapted the frames to the same temporal reference, we selected the corresponding frames in each camera. We transformed the frames to HSV color space and, for each detected object in each camera, we computed the histogram of the image patch contained in the bounding box. We only considered the first channel of the image in order to measure Hue values.

Once we had the color histogram of the object in one camera, we compared it with the histograms of all the detected objects in the second camera. In order to match boxes, we maximized the Histogram intersection defined as:

$$\sum_{I} min(H_1(I), H_2(I)) \tag{4}$$

The matching candidate was the object that maximized the histogram intersection. If this value was bigger than a threshold, then it was computed as a correct match.

Since there is no ground truth in our detections, a video is taken as a starting point, whose ID tracks make the initial reference. When comparing a second video against this one, in case of finding correspondences, the track IDs of this second video are updated to match the video. Once this video has been analyzed, videos can be taken successively and the same procedure can be performed. In this way, all the ID tracks will be synchronized with the tracks of the video taken as reference.

2) SIFT matching: For the second method we implemented SIFT to get the matches between two videos. The idea is to take each frame from these videos, obtain the bounding boxes of each detection and calculate the correspondence between them. Since we have the track ID of each object in each video, if any correspondence is found with SIFT, the track IDs are reassigned so that they match each other.

For matching, we use OpenCV's BFMatcher implementation, which uses brute force to find the best matches between two sets of descriptors. For each detection in a frame the keypoints and descriptors associated to that object are computed. These descriptors are added to a list that corresponds to that particular frame and, along with a similar list from another frame, are used as inputs for BFMatcher. This function compares two sets of descriptors associated with two different objects and calculates the L2 distance. Based on this distance, correspondences are sorted and those pairs that differ the least from each other are selected. As an additional step, the number of correspondences between the frames is limited to the minimum number of objects in the frames and they are also selected in such a way as to obtain a univocal correspondence between both frames.

As in the previous case, a video is taken as a starting point, whose ID tracks make the initial reference. When comparing a second video against this one, in case of finding correspondences, the track IDs of this second video are updated to match the video. Once this video has been analyzed, videos can be taken successively and the same procedure can be performed. In this way, all the ID tracks will be synchronized with the tracks of the video taken as reference.

Method	Detection Network	Post Processing	c001	c002	c003	c004	c005
Kalman	RCNN	NO	0.159	0.220	0.186	0.273	0.086
Kalman	RCNN	YES	0.368	0.269	0.292	0.481	0.113
Kalman	YOLO3	YES	0.333	0.299	0.334	0.461	0.10
Max - IOU	YOLO3	YES	0.367	0.347	0.367	0.497	0.102

TABLE I

IDF1 VALUES FOR SEQUENCE 1

Method	Detection Network	Post Processing	c010	c011	c012	c013	c014	c015
Kalman	YOLO3	NO	0.255	0.11	0.33	0.486	0.328	0.361
Kalman	YOLO3	YES	0.6	0.454	0.69	0.525	0.348	0.937
Kalman	RCNN	NO	0.197	0.041	0.27	0.248	0.298	0.166
Kalman	RCNN	YES	0.552	0.349	0.70	0.481	0.35	0.96
Max - IOU	YOLO3	NO	0.273	0.142	0.34	0.632	0.428	0.395
Max - IOU	YOLO3	YES	0.672	0.345	0.811	0.689	0.464	1

TABLE II

IDF1 VALUES FOR SEQUENCE 3

3) Homography computation: Our last approach in order to relate tracks from different cameras, was computing the planar homography between each camera. A positive aspect of this method is that we don't need the actual frames in order to compute the matches: we can only use the list of detections per camera.

As a first attempt, we used the Homographies that were already provided by the AI City Challenge. The main idea of the method is described as follows:

1) Project the bounding box centers from the image plane into LLA plane (latitude, longitude, altitude). (whereas altitude is neglected, this homography relates camera plane to a common surface plane)
2) Compute each track's trajectory as a 3D vector (x, y, t).
3) Since temporal reference may be inaccurate, create a temporal sliding window.
4) Within this time window (set to 10 seconds), trajectories of tracks from different cameras were compared using an euclidean similarity matrix. (LS)

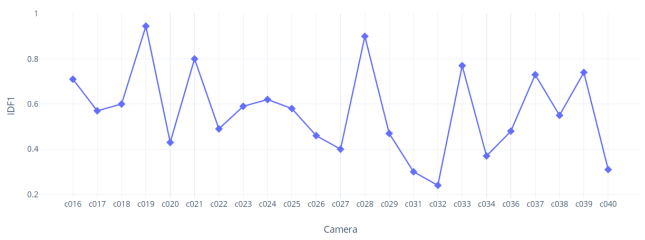
As a second attempt, we tried to calculate ourselves the homography between cameras. For this goal, we obtained each sequence's background (Week 2 algorithm) and computed matching keypoints between the different backgrounds. With this data we calculated the Fundamental matrix relating different cameras.

III. RESULTS

A. Single camera tracking

As a first step, we implemented the single camera tracking methods described in section II. The metric we used to evaluate the quality of the results was IDF1, as explained in the Section II. We evaluated our methods on Sequences 1, 3 and 4. The results for the different implementations are detailed in Tables I, II and Figure III-A.

In all cases, we obtained better results by applying post processing methods (discarding detections with short track lengths, and detections of static objects). Moreover, we



(a)

Fig. 1. IDF1 of MAX IOU with Post Processing (Tracking in Single Camera). Results displayed for Sequence 4

Metric	IDF1	IDP	Precision	Recall
Seq 1	0.09	0.09	0.10	0.08
Seq 3	0.10	0.13	0.20	0.31
Average	0.115	0.15	0.205	0.27

TABLE III

RESULTS FOR MULTIPLE CAMERA TRACKING WITH HISTOGRAM COMPARISON

assume we could reach better results by also merging similar tracks as a post-processing step, before eliminating tracks.

Regarding the comparison between the different sequences, we observe that we obtained higher IDF1 on Sequence 3 than on Sequence 1. We think this is probably due to the fact that Sequence 1 has many more cars, mixed trajectories and occlusions than Sequence 3.

In Sequence 4 we had inconsistent results, we attribute this to the large number of cameras, points of view and traffic lights (traffic lights can contribute to wrong labelling a track as a static object).

Analyzing the different object detection methods, we were surprised to see that, after post processing, we obtained even higher results with YOLO3 than with Mask - RCNN. Since YOLO3 is known as a high speed net with lower accuracy, we were amazed by how good the detections from this net were.

Comparing the implemented tracking methods, we didn't observe a systematic difference between the performance of Kalman filtering and Max - IOU. After applying post processing methods, both algorithms were equally good.

B. Multiple camera tracking

As described in Section II, for multiple camera tracking we implemented three different methods: histogram comparison, SIFT matching and Homography computation.

1) Histogram comparison: In the context of computing similarity functions, first of all we implemented the histogram comparison described in Section II. The results for the different sequences are displayed in Table III.

First of all, we see that the obtained IDF1 is lower than expected. We think the low quality of the results is related to the following issues:

- Our major drawback was that the temporal calibration provided by the challenge was not accurate. This led to a poor temporal correlation between frames from different cameras. Since our method compared frame by frame, this temporal shift had an important negative impact on our results.
- Pose and orientation of the cameras are really different. This leads to failures in color matching.
- In the case of having multiple objects, color can be a poor feature to compare different objects: if more than one car has the same color, this method will fail.
- Color matching is not robust to differences in illumination of specular reflections.

(a) Camera 01 (b) Camera 06

Fig. 2. Same vehicle at same time stamp from two different cameras

Comparing the results from both sequences, we see that we had a similar performance in both cases. In order to understand the reason, we need to remember that the multi camera tracking method we implemented uses as input the tracking obtained as an output from the first part of the project (single camera tracking).

Given that single tracking performed better on Sequence 3 than on Sequence 1, our input data for this task was more accurate in the case of Sequence 3 than on Sequence 1. On the other hand, the temporal calibration provided by the challenge was better on Sequence 1 than on Sequence 3. The combination of these two factors leads to similar results on both sequences.

2) SIFT matching: It is necessary to note that this method has the disadvantage of referring to the large changes of scale and points of view of the objects detected by the different cameras. Even though SIFT is considered as a scale invariant descriptor, depending on the location of the cameras, the area of a detection can be between 3 and 5 times larger than the detection from another camera, in addition to making captures from diametrically opposed points, so it is possible that the results are not very accurate. As an example, Figure III-B.2 shows the same car from two different cameras at the same time.

Nevertheless, this problem can be remedied in part by carefully choosing the cameras that are used at each step of the process. If the cameras are in similar positions and observing similar regions, there is a high probability that the matches will correspond to each other. We selected the order of the cameras according to this criteria.

As in the case of the histogram comparison, our major drawback was that the temporal calibration provided by the challenge was not accurate. This led to a poor temporal correlation between frames from different cameras. Since our method compared frame by frame, this temporal shift had an important negative impact on our results.

Also, the differences in camera pose and orientation were big enough to make our algorithm fail. While in histogram comparison we managed to obtain results, we didn't succeed in the case of keypoint matching.

From the comparison between SIFT and histogram matching, we conclude that color is a more robust feature than image keypoints against differences in camera pose and orientation.

Another disadvantage of SIFT and histogram matching is that they are not online, since a tracking algorithm is necessary prior to the matching step. However, if the tracking can be obtained in real time, these algorithms could be modified to make comparisons between all cameras frame by frame.

3) Homography computation: Our first attempt was to match different tracks by computing the homographies provided by the AI City Challenge website. Unfortunately, these calibrations were not accurate and we experienced an important reprojection error. Finally, we could not obtain any results with this method.

Since the calibration was not accurate, we tried to calculate ourselves the homography between cameras, as explained in the Section II. Once again, this attempt failed due to big differences in camera's pose and orientation: we could not find any matching keypoints between the backgrounds computed for the different cameras.

IV. Conclusions

A. Single camera tracking

- In all cases, we obtained better results by applying post processing methods (discarding detections with short track lengths, and detections of static objects). Moreover, we assume we could reach better results by also merging similar tracks as a post-processing step, before eliminating tracks.
- Each one of the post processing steps have clear advantages and disadvantages. Removing short tracks before merging tracks with similar motion can result in a lose of True detections. The problem of discontinuous tracking has high feasibility due to occlusions or inaccuracy in the detection, therefore it is more reasonable to first merge tracks and only then eliminate them. The results in this report are without the merging post processing step. We believe that adding this step can improve our results.
- Single camera tracking had very good results for different detections (YOLO and RCNN) and algorithms (Kalman filtering and Max IoU). All cases were improved with post processing.
- We obtained higher IDF1 on Sequence 3 than on Sequence 1. We think this is probably due to the fact that Sequence 1 has many more cars, mixed trajectories and occlusions than Sequence 3.
- We expected low results with YOLO, but after post processing we got even better results than with Mask - RCNN.
- We didn't observe a systematic difference between the performance of Kalman filtering and Max - IOU. After applying post processing methods, both algorithms were equally good.

B. Multiple camera tracking

Multiple camera matching gave worse results than expected. Main problems were:

- Frames were difficult to synchronize with the provided calibration.
- Pose and orientation of the cameras are really different. This leads to failures in feature matching (color in the case of histograms, or key points in the case of SIFT).
- The most promising idea was using a Homography relating the different cameras. Nevertheless, the projective error was too big so we couldn't obtain good results.

V. Further work

The results we managed to achieve were not sufficient in our opinion. In order to to improve the tracking algorithms we proposed, we would like to examine several directions that could potentially overcome the gap of the discontinuity of the time stamp and the inaccuracy of the homographies by:

- Using the entire track features in one camera as a cluster, and creating an adaptive cluster with each new bounding box assigned. The cluster can be composed of SIFT descriptors, color histograms or any other feature.
- Solve the time calibration drift using a sliding window over the temporal axis.

References

[1] A. C. Challenge, "Ai city challenge website." [Online]. Available: https://www.aicitychallenge.org/

[2] X. W. Xi Chen and J. Xuan, "Tracking multiple moving objects using unscented kalman filtering techniques."

[3] A. Salvador, "Tutorial for kalman filtering." [Online]. Available: https://github.com/telecombcn-dl/2017-persontyle/blob/master/sessions/tracking/tracking_kalman.ipynb