Implementing and Evaluating a Tracking-By-Detection Algorithm for a Camera Monitoring System

Khaled Jbaili, Jan Thomanek
Hochschule Mittweida, Technikumplatz 17, 09648 Mittweida

Abstract
In this paper, we designed, implemented, and tested a special surveillance camera system based on a combination of classical image processing algorithms. The system's sub-objective consists of tracking experimental vehicles driving on a defined trajectories (Rail) in real time. Furthermore, it analyzes the scene to collect additional vehicles & rail-related information. The system then uses the gathered data to reach its main objective which confines oneself in independently predicting vehicles collision. Consequently, we propose a hybrid method of detecting and tracking ATLAS-vehicles efficiently. To detect the vehicle at the beginning of the video, periodically every n-frame, and in the case where the tracked vehicle has been lost, we used Histogram Back-Projection. By contrast, Kernelized correlation filter is used to track the detected vehicles. Combining these two methods provides one of the best trade-offs between accuracy and speed even on a single processing core. The proposed method achieves the best performance compared with three different approaches on a custom dataset.

1. Introduction

There is no doubt that Object Detection has always been a key technology behind many applications like video surveillance, image retrieval systems and Advanced Driving Assistance Systems (ADAS). In addition to this, it is considered as a foundation of several tasks, such as instance segmentation, image captioning, and object tracking. To understand object detection problem, it necessary to define it more formally. In the foremost place comes object recognition which is considered as a general term that describes collection of related Computer Vision (CV) tasks and involves identifying objects in digital photographs starting from image classification and ending in object segmentation. Image classification involves predicting the class of one object in an image, whereas Object localization refers to identifying the location of one or more objects in an image. Object detection combines these two tasks in which it localizes and classifies one or more objects in the image. The previous statement can be summarized with the following question: "What Objects are where?"

Going back to object detection where some input data such as a camera image or Lidar point cloud is given. From this one first needs to determine how many objects of interest are present. For instance, if one is trying to detect vehicles and predestines in a traffic situation, then for each detected object, one determines the object type and the object shape.

Object type refers to the class of the object and object shape represents the enclosure of the detected object of interest or detection hypothesis. There are many detection hypotheses used for labeling a detected object. In most cases, the object shape is determined by a bounding box (bbox) in 2D or 3D.

Another research field which is most salient and explored amongst several CV disciplines and plays an important role in this work is Visual Object Tracking (VOT). Object Detection as defined previously involves determining semantic existence of object instances of a specific class in an image or every video frame, e.g., cars, boats, and humans.

In contrast, VOT can be defined as the estimation and prediction of object trajectory in a video frame as it moves from one frame to another based on dynamics. In other words, the use of dynamics, such as object's velocity, direction, location history, and similarity among consecutive frames, is the fundamental difference between just doing detection and tracking. This can be seen in Figure. 1, where a person walks on the pedestrian's path. Due to the nature of human locomotion, the human Body changes shape from Frame to Frame.

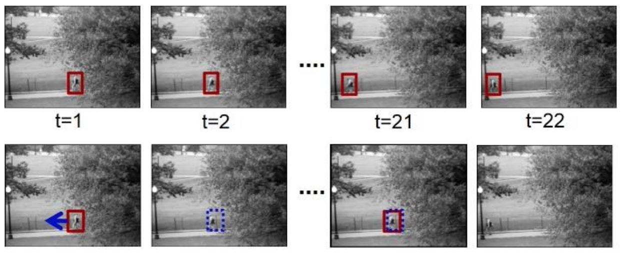


Figure 1: Detection vs tracking. In the first row, a detection occurs in every frame. In the second row a tracking occurs (doted bounding box) in every frame whereas detection occurs every n-frame.

In this study, we propose to utilize a detector alongside a tracker to mitigate its shortcomings. Specifically, The detector acts as a localization initializer and a self-correction mechanism periodically and whenever the tracker loses the target.

The resulting method delivers good segmentation accuracy. It is also lightweight enough to run on single-core central processing unit (CPU) or even edge computing boards. The performance of this method in detecting & tracking ATLAS-vehicles was tested with a dataset we created. Experiment results indicate that the proposed method is suitable for real-time tracking and detection application with very much affordable computation requirements.

1.1 Scope on ATLAS

The project germ: "Asynchroner Transport-, Logistik- und Automatisierungsmodus auf der Schiene" (ATLAS)- [1] represents a demonstration plant (100 m long of total rail-models), on which lightweight individual self-driving vehicles drive. The concept was first launched in 2006 at the University of Applied Science in Mittweida-Germany by Prof. Christian Schultz and is currently being developed. The purpose of this project is not only to introduce a new control principle of the train which in turn is based on distributed control instead of central one, but also its core concept starts from the idea of abandoning the train's manufacture that has been used since the middle of 19th Century as basis structure for train composition along the way until reaching the fully automated level. This leads to a huge reduction in power consumption by acceleration and the breaking distances are shortened, unlike the current train operation where huge safety distances are necessary, and this is why one sees an empty railroad and crowded highway.

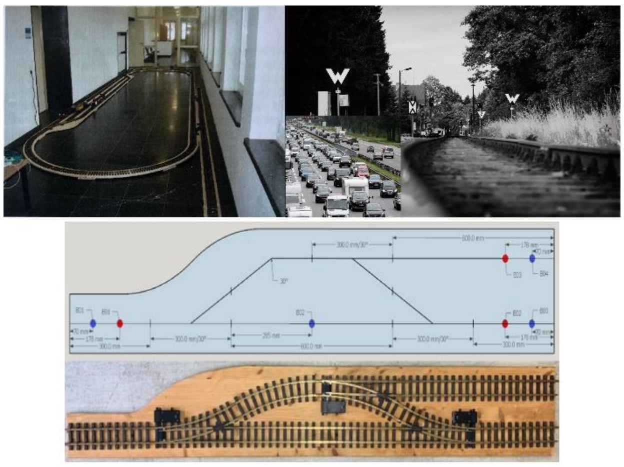


Figure 2: (Top-left) Part of ATLAS's demonstration plant. (top-right) fully occupied highway while empty railroad which expresses the Project's motivations. (Bottom) technical view of an ATLAS submodule. Blue points are position indicator (Balise) and red points represent Infrared-receivers for controlling the switches.

The project also aims to implement and invest those automation driving algorithms on the railroad, since the constraints and demands are much more convenient than the ones on the street. E.g. no permanent steering of the vehicles is required. The main points to be considered are distance control, collision avoidance, controlling the Railroad switches, and cooperative driving and route planning. Unlike the current European Train Control System (ETCS), the vehicles in ATLAS should be able to take full control of the route, accomplish the previously mentioned tasks ideally reverse the philosophy "stupid train, clever route" into "stupid route, clever train"

Therefore, the vehicles in ATLAS are equipped with sensors for sensing the environment. These are, Distance Sensor Module (DSM), Jetson Nano which represents the environmental model and is responsible for route planning as well as communication among the vehicles via vehicle to vehicle wireless link (V2V), a mainboard (M16C) for controlling the engine, and the Infrared (IR)-system.

The Rail-model of ATLAS is provided with IR-senders called Fr. Balisen (milestone). These are position transmitter that broadcast periodically 16-bit code, which represents the absolute position of the vehicle. This means that up to 65536 Balise can be installed. Currently, there are 130 Balisen installed on the Demonstration plant. When a vehicle drives above the Balise, it receives, processes, and shares its position with other neighbor vehicles.

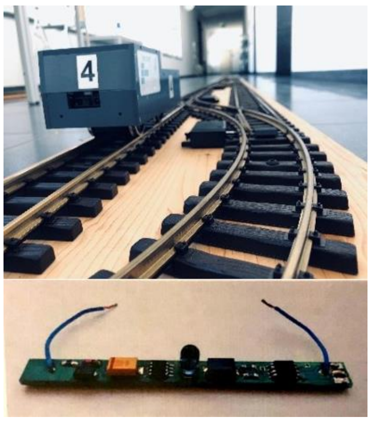


Figure 3: (top) ATLAS-Vehicle model. (bottom) Balise-module.

1.2 Viewpoint

The camera network consists of multiple static cameras (at least two) with relatively non-overlapping views. This network operates independently from ATLAS-system. All cameras are connected through a usb3-hub to the computing unit. The cameras are placed on Tripod and ladder in such a way that a bird's eye view of the scene is obtained. Thus, it is expected that both have relatively the same rotation vector with respect to the world origin. This is very convenient compared to cameras with mutual view and different rotations, since in our case the object of interest (OOI). That is the ATLAS-vehicle that cannot be present in both camera's Field of View (FOV) at the same time. Therefore the problem of associating objects belonging to the same class is improbable.

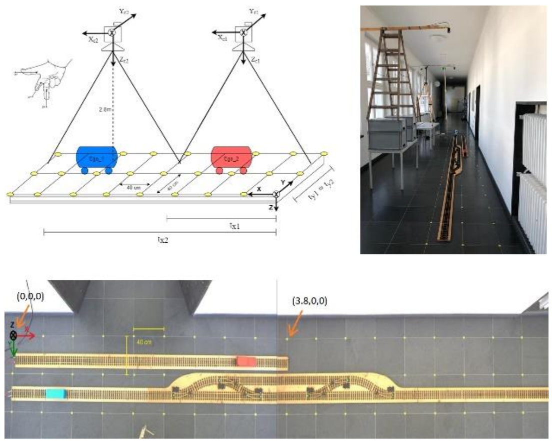


Figure 4: Side and top view of the of camera system and ATLAS respectively. Cameras are placed at 2,8m hight from the scene.

It is important to refer that the world axes directions are set using the convention of the clockwise axes-system (right-hand rule) which makes it aligned with the camera coordinate system. In figure 4, we can see yellow points placed on the world plane in certain positions. These points will be used along with their image points correspondences for estimating the camera pose (Rotation and Translation of both cameras) which results, by incorporating the intrinsics as well as homography between the world and both sensor plane respectively.

2. Overall System Flowchart

The developed camera system can:

- Detect and track any ATLAS-vehicle located in the FOV of any camera.
- Identify the direction of every tracked vehicle.
- Reconstruct the planar-world coordinates of the tracked vehicle.
- Determine the status of each relevant switch with respect to every vehicle being tracked.
- Assemble the independent camera views together to form the bird's eye view mosaic and display the results on a simulation plane.
- Exploit all above information in detecting collision for our scenarios.

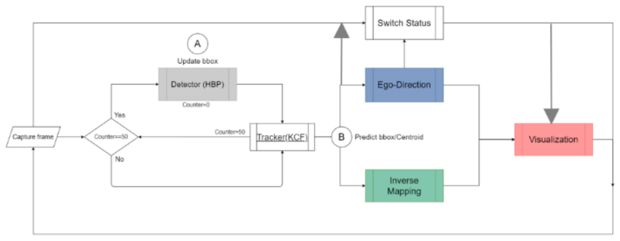


Figure 5: Block diagram of the whole process.

2.1. Proposed Method

In this study, we propose a hybrid method to track ATLAS's self-driving vehicles in real-time. For that reason, the Kernelized Correlation Filter (KCF)- [2], which is particularly fast at tracking, is used along with detection approach, Histogram Back-Projection (HBP)- [3], which in turn uses color information and the geometry of the OOI, namely the Rectangularity, Area and Ratio.

2.1.1. HBP Detector

CV tasks related to object detection expose the answers to the dichotomy: "What/where?". "what" means the ability to identify or classify objects from different object categories in each input image. Algorithms such as histogram intersection, supported vector machine (SVM) and other machine and deep learning- based classification solve this problem. On the other hand, "where" abbreviates the question, where in the target image is the object, one is looking for?

HBP answers this question considering that the image is colored. Given an input target-image which contains the OOI to be localized, the colors which appear in other objects besides the OOI are de-emphasized and they are less likely to distract the search mechanism. On the contrary, the pixel values belonging to the OOI are enhanced. Therefore, each vehicle must have a discriminative color representing its ID.

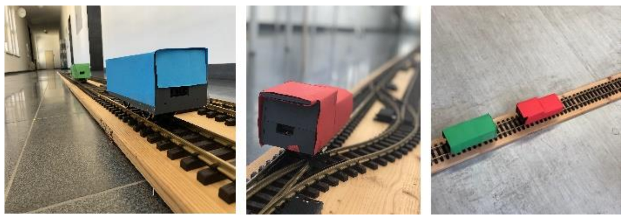


Figure 6: ATLAS-vehicles, each with a different color.

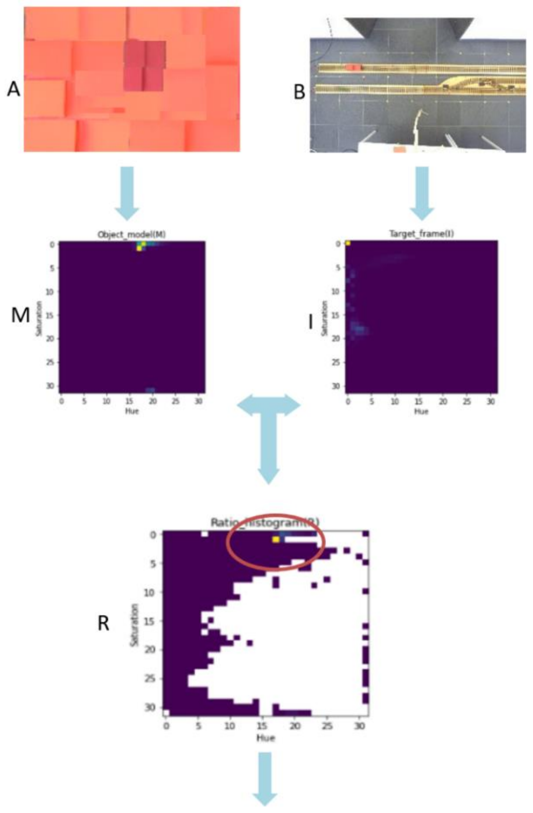
Let M be the calculated 2-dimensional color histogram of the object model (figure 7-A) which we search for, and I the histogram of the target image (figure 7-B). Consequently, a third histogram R, which is the ratio of M divided by I, is computed.

$$\mathrm{R} = \frac{\mathrm{M}}{\mathrm{I}} \tag{1}$$

This histogram contains pixel-values distribution for the object model only. Then the Ratio histogram is back-projected onto the image, that is, the image values are replaced by the values of R that they index. By normalizing pixel values between 0 and 1, each resulted pixel value of the final image represents the probability belonging to the object model, given that there is an object.

$$\mathrm{p(x|obj)} = \frac{\mathrm{p(x \cap obj)}}{\mathrm{p(obj)}} \tag{2}$$

All histograms were created based on HSV-color space using the Hue and Saturation channel



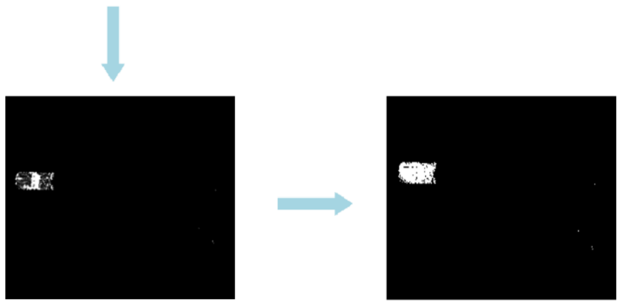


Figure 7: overall visual workflow of the HPB-detection algorithm.

The HBP is then followed by three post-processing stages. These are, refining the obtained mask-image, object representation, and further improving the detection accuracy through shape descriptors that best describes the target object. In the refining process, we worked on a binary image (values range between 0-255) which results from HBP-pipeline. First, a convolution with radius D, which has relatively the same area as the expected area subtended by the OOI, is performed to improve the implicit segmentation.

$$Objmask = Obj\_mask * D \tag{3}$$

Further filtering steps are then performed such as removing bit-wise false negatives (FN) and salt noise in the mask image. These operations which are used for enhancing the detected object are called morphological transformations. E.g., closing and dilation.

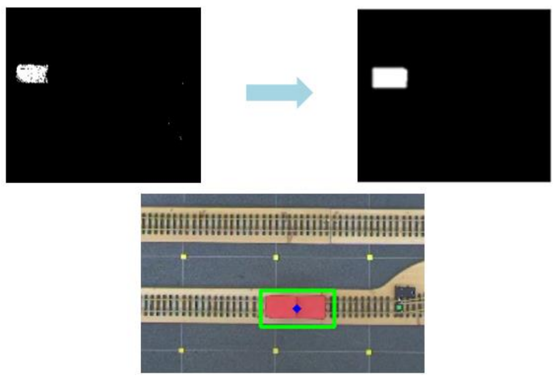


Figure 8: Object representation.

Second, the binary image which contains the enhanced OOI is passed to the feature representation process. In this step, the OOI is represented using primitive shapes. We used Centroid for displaying the OOI on the simulation image. At the same time, a rotated bounding box is implicitly calculated and used for further discriminating the OOI from other false positives (FP) objects that, because of color similarity, remain in the output binary image. The OOI's area, due to the fact that the vehicles are observed from Bird's-eye view, varies in certain range and relatively preserves its shape which in this case a rectangle. Hence, the OOI's rectangularity according to Podczeck's shapes- [4] is obtained through the following equation

$$\text{Rectangularity} = \frac{\text{a}}{\text{wh}} \tag{4}$$

where,

a is the object area, and w and h are the width and height of the smallest bbox containing the object respectively.

2.1.2. KCF Tracker

As the name implies, correlation between two samples is calculated. The correlation value is in the range of X and Y and maximizes when samples are identical. This is the principle of every correlation filter and can be adapted to the tracking problem. Visual tracking methods based on correlation filters are designed to produce the highest response when the filter is applied on the object to be tracked. Unlike all of the tracking-methods such as MOSSE [5], MIL-Track [6] and STRUCK [7] which had one thing in common: a "sparse sampling strategy" that selects p random sub-windows from the Region of Interest (ROI), KCF uses "dense sampling" which takes all sub-windows in the ROI and train a ridge-regression classifier with all samples, and thus it allows a more efficient training. To reduce the computational complexity of the learned filter, and thus to speed up the method, the properties of circulant matrix-[8] are used. Additionally, the calculations are made in the frequency domain using Fast Fourier Transform (FFT) to quickly incorporate information from all sub windows, without iterating over them. KCF also utilizes the kernel trick to improve the accuracy.

let y_i represent a target and $f(z)$ $=\boldsymbol{W}^T z$ be the function that minimizes the squared error over samples X_i and their regression targets y_i. The KCF briefly solves the following ridge regression optimization problem,

$$min_{\boldsymbol{w}} \sum_i (f(X_i) - y_i)^2 + \lambda \left|\left|\boldsymbol{W}\right|\right|^2 \tag{5}$$

Where λ is a regularization parameter that controls overfitting. The solution of the problem in the frequency domain corresponds to

$$\widehat{W} = \frac{\hat{X}^* \odot \hat{y}^*}{\hat{X}^* \odot \hat{X} + \lambda} \tag{6}$$

where $\widehat{\boldsymbol{W}}$ indicates the Fourier transform of $\boldsymbol{W}$, $\hat{X}^*$ is the complex conjugate of $\hat{X}$, and $\odot$ denotes the element-wise product. We can easily obtain $\boldsymbol{W}$ in the spatial domain by using inverse Discrete Fourier Transform (DFT). Since the problem is solved in the frequency domain, the method is extremely fast and thus makes it ideal for practical applications. Consequently, this method is one of the fastest in the literature, robust against to translation and scale variances, and provides high accuracy in general. However, it only searches for the object in a region of interest, and hence it fails when the object is occluded or has left the field of view, as it does not have a self-correction mechanism. This is where the HBP takes its place.

2.1.3 The Proposed Tracking-By-Detection Method

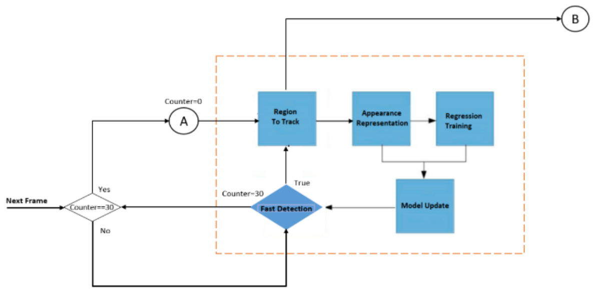


Figure 9: Block Diagram of the Hybrid-KCF Tracker.

As seen from the previous figure, a fast detection occurs implicitly inside the tracker block to generate the new region to track. The detection through HBP on the other hand occurs periodically (e.g., every 30 frame). The tracker can also call the detection earlier when it fails to fast detect the object. This uncovers the fact that the tracker should be accurate enough so that it doesn't invoke the detection very frequently. The periodical occurrence of the HPB-detection is chosen based on multiple experiments.

2.2. Determining Direction

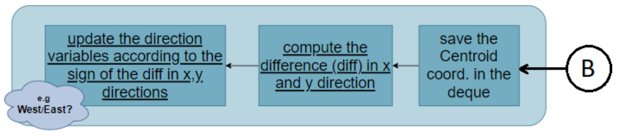

update the direction variables according to the sign of the diff in x,y directions
compute the difference (diff) in x and y direction
save the Centroid coord. in the deque
B
e.g West/East?

Figure 10: Block diagram of direction-determination process. B is the incoming centroid from Tracking Pipeline.

For computing the direction, the movement of the vehicle (if any) is essential. In case of no movement, the direction is not updated and the vehicle's direction is considered as "None". When the vehicle moves, its direction of movement can be obtained by computing dx and dy, namely the deltas (differences) between the x and y coordinates of the current frame and a previous one.

However, using the current and the previous frame is a bit of an unstable solution. Unless the object is moving very quickly, the deltas between the (x, y)-centroid coordinates will be very small. Thus, when computing These Deltas directly to report direction, then the results are extremely noisy, implying that even small changes in trajectory would be considered as a direction change. In fact, these changes are so small that they would be near invisible to the human eye (or at the very least, trivial) and reporting as well as tracking such small movements is not of interest.

Instead, reporting the direction of larger object movements is much more likely of interest — Hence, the deltas between the OOI-centroid coordinates of the current frame and a frame preceding the current with relatively large time-period is computed. Performing this operation helps reduce noise and false reports of direction change.

2.3. Location Reconstruction on Planar World

Any point in the 3D-space is mapped onto a sensor plane through perspective Transformation process (PTP) which can be expressed as 3x4 projection matrix P.

$$P = KR[I| - t]\ ,$$

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}\ ,\ K = \begin{pmatrix} f_x & 0 & x_u \\ 0 & f_y & y_v \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \tag{7}$$

where, K is the camera matrix containing the Intrinsics, R and t are the rotation and translation matrices of the camera with respect to the world origin and both are the Extrinsics.

If the points in the 3D-space lie on a plane, then they and their corresponding points on the image plane are related by a planar-homography, and this is the key idea behind obtaining the planar world coordinates of the vehicle's centroid.

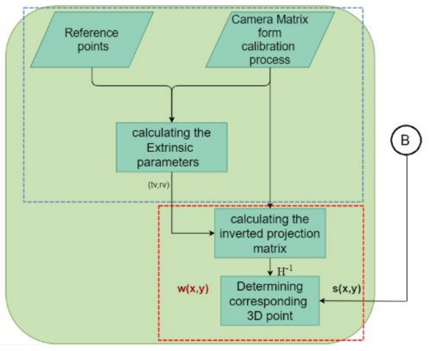

Reference points
Camera Matrix form calibration process
calculating the Extrinsic parameters
(tv,rv)
B
calculating the inverted projection matrix
H⁻¹
w(x,y)
Determining corresponding 3D point
s(x,y)

Figure 11: Block diagram of the location reconstruction's process. (tv, rv) are translation & rotation vector of the camera respectively. H^{-1} denotes the inverse homography. S Is the 2D-vehicle's centroid and W is the corresponding world point with z = 0 (planar world).

The pipeline illustrated in the above figure consists of two main processes:

- Camera Calibration
- Image Plane to World Plane transformation (Planar-Homography)

2.3.1 Camera calibration

This process involves finding the Intrinsics, Extrinsics and distortion coefficient of the cameras. For estimating the Intrinsics of both cameras, we have performed Zahng's Method on Matlab, which uses known calibration points located on a checkerboard and find correspondences between these points when they are in different positions. Regarding the extrinsic, we have created a grid of world points with a constant known distance between them. For the sake of convenience, we have exploited the floor's -tile grid which has 40x40 centimeter square tile. Hence, the intersections of the tiles are the locations of the world points. To find the corresponding 2D-image points effortlessly, we have applied thresholding techniques on the image containing the grid's tiles so that only world points remain visible and then used OpenCV functions to get the 2D-pixel coordinates of the control points. Missing points can be found by manually locating the pixel locations (see figure 12).

	Camera matrix in H.c	
Camera 1	$\begin{pmatrix} 4{,}3807e+03 & 0 & 1{,}0059e+03 \\ 0 & 4{,}3782e+03 & 6{,}9702e+02 \\ 0 & 0 & 1 \end{pmatrix}$	
	Rotation vector	**Translation vector**
	$\begin{pmatrix} -0{,}02823308 \\ 0{,}08623225 \\ 0{,}01563199 \end{pmatrix}$	$\begin{pmatrix} -5{,}47884374 \\ -3{,}08581371 \\ 24{,}15112048 \end{pmatrix}$
Camera 1	$\begin{pmatrix} 4{,}4636+03 & 0 & 9{,}5639e+02 \\ 0 & 4{,}4642e+03 & 6{,}6447+02 \\ 0 & 0 & 1 \end{pmatrix}$	
	$\begin{pmatrix} -0{,}04811656 \\ -0{,}11802668 \\ -0{,}11802668 \end{pmatrix}$	$\begin{pmatrix} -8{,}95359027 \\ -3{,}01504595 \\ 24{,}83908461 \end{pmatrix}$

Table1: Results of calibration parameters for both cameras respectively.

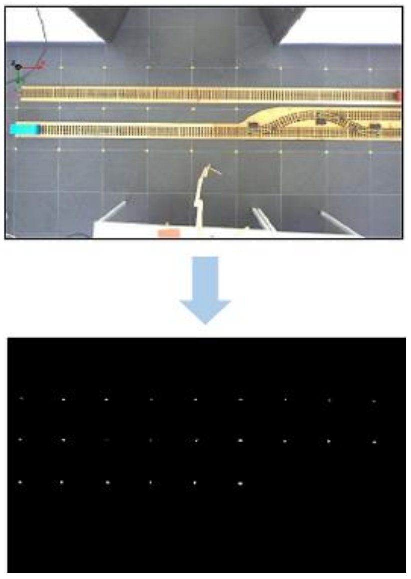


Figure 12: Finding correspondences (white spots on the upper image) of world points (yellow spots on the latter image).

2.3.2 Image Plane to World Plane transformation

Since the world coordinate system can be set anywhere, it can be conveniently positioned on the plane, such that it has zero Z-coordinate (Z = 0). This choice reduces the projection matrix, to

$$s\begin{bmatrix}x\\y\\1\end{bmatrix}=\begin{bmatrix}h_{11}&h_{12}&h_{13}&h_{14}\\h_{21}&h_{22}&h_{23}&h_{24}\\h_{31}&h_{32}&h_{33}&h_{34}\end{bmatrix}\begin{bmatrix}X\\Y\\0\\1\end{bmatrix}$$

$$=\begin{bmatrix}h_{11}&h_{12}&h_{14}\\h_{21}&h_{22}&h_{24}\\h_{31}&h_{32}&h_{34}\end{bmatrix}\begin{bmatrix}X\\Y\\1\end{bmatrix}=\mathrm{H}\begin{bmatrix}X\\Y\\1\end{bmatrix}, \tag{8}$$

where, H is 3x3 planar-homography matrix that represents the projective relationship between the world plane and the corresponding image-plane points. The equality above is up to a scale factor s and as a result the planar-homography matrix has 8 degrees of freedom.

Since this matrix H can be inverted, it follows that if H is completely known, the world coordinates of any image point can therefore be recovered through the following equation:

$$\begin{bmatrix}X/s\\Y/s\\1/s\end{bmatrix}=\mathrm{H}^{-1}\begin{bmatrix}x\\y\\1\end{bmatrix}\quad\Leftrightarrow\quad W=H^{-1}C^{T} \tag{9}$$

where,

W is the reconstructed world coordinate of the detected centroid C. The above equation is expressed in homogenous coordinates.

2.4. Detection of the Switch Status

Determining switch status means, how a switch is connected with respect to each vehicle located on its trajectory and being observed by the system, e.g., Left = L/Right = R connected. To this end, we have exploited two variables:

- the vehicle's relative direction, and
- a Switch-Flag (SF).

The SF is an indicator that returns 1 whenever a switch alters its state in such way so that the white spot placed on the notch is visible for the system (cf. see fig. 13-top). However, detecting such a small area in an image (5-6 pixels) from a relatively far working distance is not straightforward when it comes to the camera specifications, which consist of resolution, sharpness, focal length, sensor dimensions, and angel of view. Meaning to say, they must be examined so that the system can still detect the ROI from the defined working distance and under different lightening conditions. In the research project [9], we have examined all hardware components required for the system and therefore they are out of the scope of this work. The procedure for detecting the white area on the notch is shown in below figure:

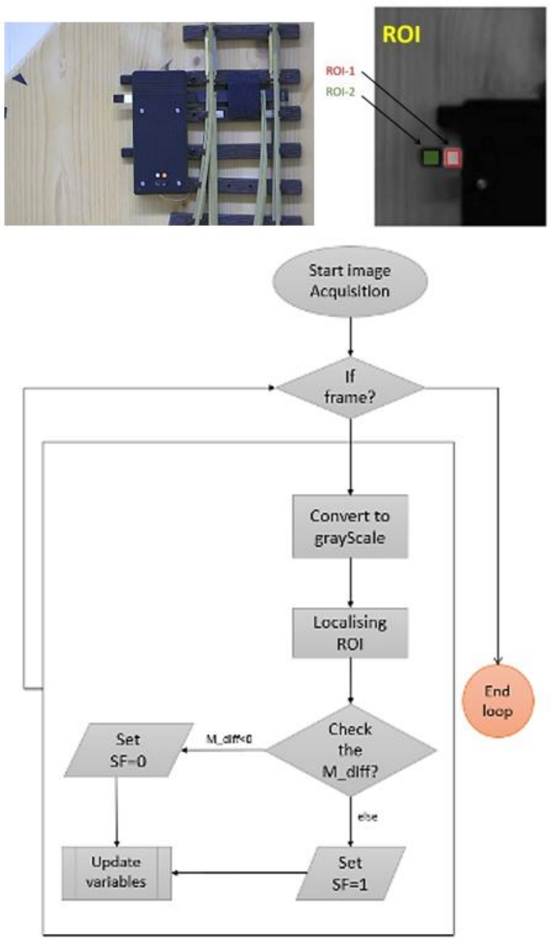


Figure 13: (Top)-ATLAS-Switch Model showing extraction of ROI. (Bottom)-Block Diagram of Detecting white spot on rail switch.

For the generality of the previous procedure, it must be assured that all white spots are placed on the same side for all rail-modules when they are visible! Thus, we could create a Look-Up-Table (LUT) that consists of all possible (considered) directions and SF's state along with the switch status accordingly.

For instance, the blue vehicle shown in figure 14 is heading "North" and the SF of the switch located a head is 1, then according to the LUT, this switch is Left-connected with respect to the blue vehicle.

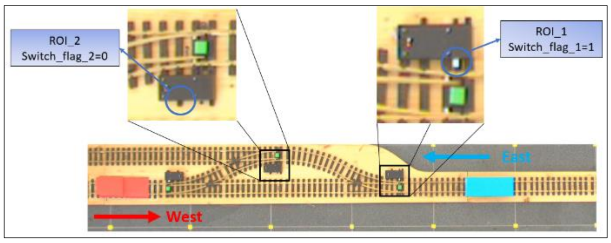


Figure 14: an illustration showing how a combination of SF and vehicle's direction gives an output that indicates the current Status of a certain switch relative to a relevant vehicle.

Vehicle Direction	Switch flag	Status
North	0	R
North	1	L
South	0	L
South	1	R
West	1	L
West	0	R
East	1	R
East	0	L

Table 2: Look Up table.

2.5. Visualisation

This step capsules the work into one uniform view. In other words, all camera's FOVs as well as the extracted scene-related information are transformed onto one view (simulation image). For this purpose, we have used the same concept explained in section 2.3.2.

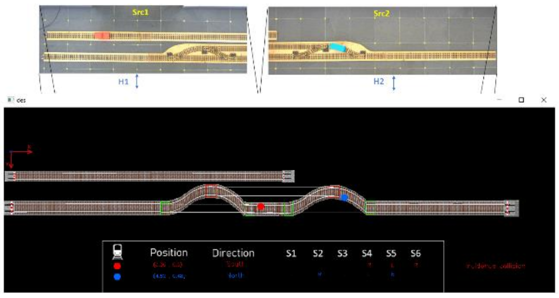


Figure 15: Transformation of Src1 & Src2 onto simulation-plane and thus achieving a central viewing monitor. In the Dst-image, information about the tracked vehicles (colored dots) such as, their position with respect to world origin and directions according to compass direction are shown. S1 till S6 are switch status with respect to every vehicle.

3. System Evaluation

The evaluation process can be split into three subtitles:

- **A.** Evaluation of the adopted Detection-algorithm.
- **B.** Evaluation of the adopted Tracking-algorithm.
- **C.** Evaluation of the reconstructed world coordinates of the vehicles being tracked.

Speaking of dataset, a 20 video sequences with 1280x720 resolution were collected. These videos have attributes that concerns different lightening conditions such as sunny & cloudy weather, with and without artificial room lights, shadows, and illumination variation (IV). The dataset is then split into train and test data. We call it training data because the adopted HBP learns in the sense that it memorizes the training data.

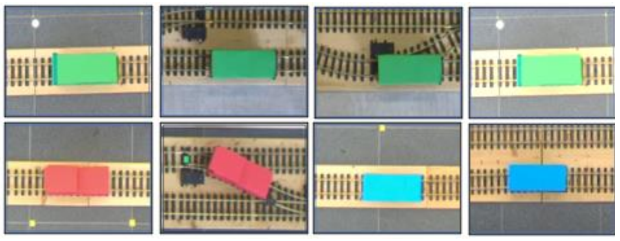


Figure 16: Snapshots from the collected dataset showing different lightening conditions.

A. We have evaluated the HBP against three other approaches, each of which has different methodology and level of complexity, however, all use color component as cue feature representation. These are: Basic Color Range Detector using HSV-color space (CRD), Color Component Ratio (CCR), and Gaussian Mixture Model (GMM). The four approaches are evaluated based on precision, which in this case defined as "how accurate the algorithm can discriminate the OOI", and Speed "number of frames per seconds (FPS) the algorithm can process". Due to unavailability of the optimal ground-truth data, the precision-evaluation is performed through the following method:

Based on the fact that an axis-aligned bounding box or a centroid of the OOI is obtained based on a binary-mask image resulted from an implicit segmentation and that the scene is being observed from Bird's-eye view camera, the OOI would always have a rectangular shape considering a perfect segmentation (under labor-conditions). Hence, a candidate blob that far deviates from rectangle-shape is considered a poor score. In other words, the more the segmented blob deviates from being a rectangle the less precise the detection is.

For making the evaluation-process rational, we have additionally defined a range-threshold for the area of the detected blob. So, for instance, if a detected blob's area falls outside the threshold, it will not be accepted, even though it has a high rectangularity score.

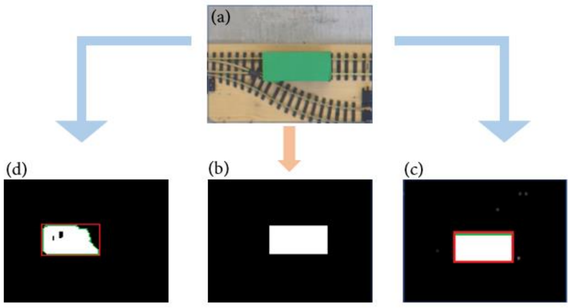


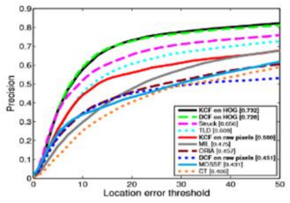
Figure 17: (a)-input image, (b)- the optimal output mask in which the target object has 100% rectangularity and the blob area is in-between reasonable threshold. (c)-output mask showing slight deformation from a rectangular shape. (d)- output mask showing a large deformation.

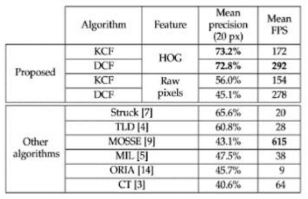
Algorithm	Feature	Mean Rect	Mean FPS
CRD	color component (raw pixels)	0,42	30
CCR		0,75	7
GMM		**0.94**	0.7
HPB (adopted)	color component & shape descriptors	0,79	**34**

Table 3: The table shows Summary of experiment results on 15 Videos. The reported quantities are averaged over all videos. Reported speeds include feature computation and post-processing step.

Looking at the previous table, the HBP outperforms the other candidates regarding speed, despite processing additional features like rectangularity, area, and ratio. On the other hand, it comes in second place after GMM regarding the average rectangularity, which is reasonable, because GMM does not only soft-assign each pixel in the image to the defined clusters with a probability value (weight), but its parameters are also efficiently pre-trained on multiple color distributions of the OOI.

B. For tracking-process, the implemented KCF-tracker is evaluated among other commonly trackers (state-of-the -art at its time). The best way to evaluate trackers is still a debatable subject. However, Babenko et al. [10] argue for the use of precision plots. These plots show, for a range of distance thresholds, the percentage of frames that the tracker is within that distance of the ground truth, and thus are easy to interpret. For obtaining direct results, the outcome from [11] is adopted.





	Algorithm	Feature	Mean precision (20 px)	Mean FPS
Proposed	KCF	HOG	**73.2%**	172
	DCF		**72.8%**	**292**
	KCF	Raw pixels	56.0%	154
	DCF		45.1%	278
Other algorithms	Struck [7]		65.6%	20
	TLD [4]		60.8%	28
	MOSSE [9]		43.1%	**615**
	MIL [5]		47.5%	38
	ORIA [14]		45.7%	9
	CT [3]		40.6%	64

Figure 18: (left)- Precision plot for all 50 sequences. (right)-Results summary over 50 videos dataset [11].

By looking at [11], it is shown that Struck operates on many kinds of features and a growing pool of support vectors. Track-Learn-Detect (TLD) is specifically geared towards redetection, using a set of structural rules with many parameters.

Despite this unevenness, KCF can reach competitive performance when operating on raw pixels as seen in the table of figure 18. Replacing raw pixel information with HOG features allows the KCF) to surpass even TLD and Struck, by a relatively large margin.

As we are looking for a tracker that gives a compromising result between precision and speed, it is shown that KCF is most suitable for the work's use case. In order to realize the difference in performance, namely speed between using pure Detection and the proposed Tracking-By-Detection approach and since the tracking speed is a crucial factor in our use case, the speed of both approaches is therefore plotted on multiple videos from the collected dataset.

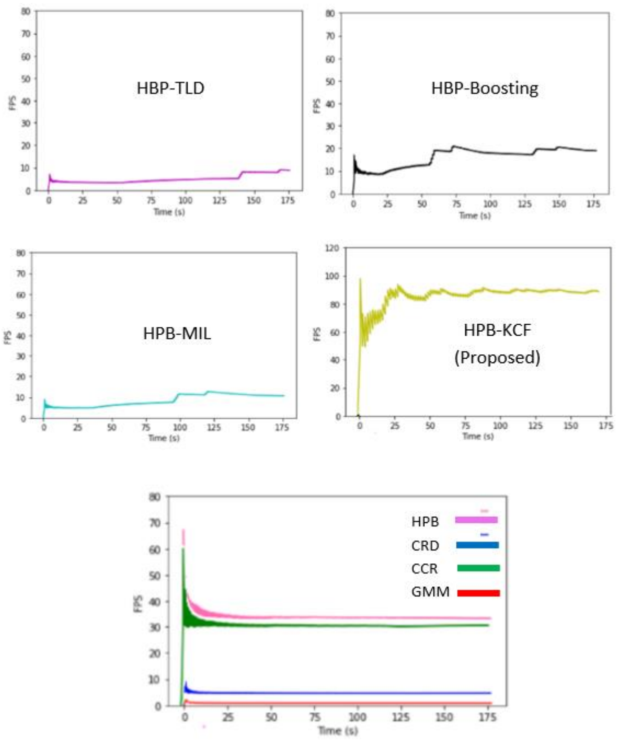


Figure 19: Comparing Hyper-Tracking against pure Detection. The evaluation process is made based on data taken from one camera.

The above figure shows that, integrating KCF in the overall process not only outperforms the pure detection, but also other available trackers by large margin. In fact, an average of 400 fps can be reached by reducing the size of the tracked region, however, it reduces the precision-performance.

C. The last part of the evaluation confines oneself in the following question: How accurate are the world coordinates of the ATLAS-vehicles obtained from the Detect-Track-Reconstruct pipeline with respect to the ground truth-coordinates?

The ground-truth/actual positions were taken using a caliper device. Considering the distance between every two yellow control points in x and y direction as reference and equals to 40cm, direct results can be obtained. On the other hand, an error-deviation in millimeter-range caused from the previous convention is expected.

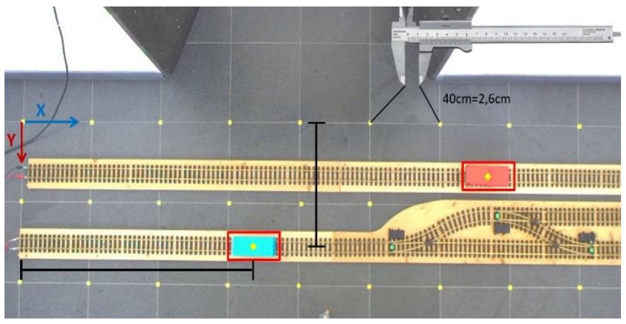


Figure 20: An illustration showing how ground-truth (actual locations) are estimated.

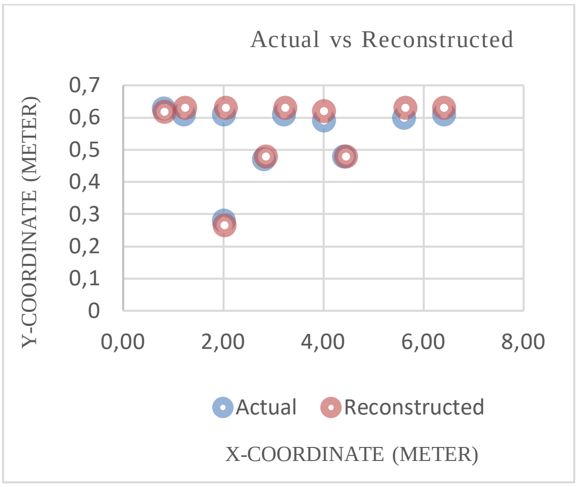


Figure 21: The scatter chart plotting ten different reconstructed coordinates of ATLAS vehicles compared with their observed (actual) ones respectively.

Based on the previous convention, and on the experimental results depicted in figure 20, it is observed that the Euclidean offset between the actual and reconstructed points for both x and y coordinates are very low which implies a very good performance of the process. Mean Squared Error (MSE) was taken as an evaluation metric. The results were (0,00092, 0,000379) meter for x, y world-coordinates respectively.

4. Collision Detection

Detecting a collision is, due to lots of variables that plays an important role by decision making, can range from very easy to very complex. This is essentially based on how the trajectories change. The more the trajectories vary, the more parameters must be taken into consideration.

Collision scenarios can therefore be simplified in two main types:

I. vehicles drive in parallel in the same direction reaching a connecting node (switch)

II. vehicles drive opposite of each other.

In the following, the second case is considered as it is relatively more complex:

1. A collision is potential if two vehicles have opposite directions and at the same time either of their coordinates are equal or fall within a certain small range.

2. A potential collision becomes critical if, in addition to 1st statement, the Euclidean distance between the two vehicles becomes smaller than a defined threshold -distance value. The threshold value is trajectory-specific and chosen rationally based on multiple experiments.

3. A critical collision becomes Incident or Inevitable, if in addition to 1st & 2nd statements, the status of the most-near switches in between two vehicles are different with respect to the vehicles and within certain timestamp do not match.

The timestamp defines the time in which a switch's status is expected to be changed from the current passing vehicle. Since the vehicles control the switches through Infrared-sensor, the timestamp can be translated into a distance which equals to 15cm. This is the maximal distance where a signal can be received via Infrared-link. (For more details, see [1]).

If	II	1	2	3	. . .
Collision is	potential	potential			
	Critical	Critical	Critical		
	Incident	Incident	Incident	Incident	Incident

Table 4: Collision table for collision type I.

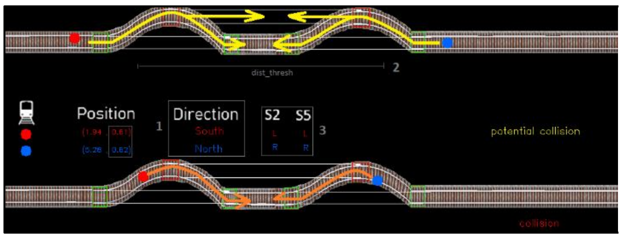


Figure 22: collision scenario. On top of the image, we can see that the system sets the decision as potential because the first condition is True. The system changes the decision to incidence because all three conditions being considered for this trajectory are True.

5. Conclusion & Future Work

In this work, a surveillance system dedicated for observing vehicles driving on the ATLAS-demonstration plant was implemented & tested. By "Observing" it is meant that the system can detect and track trajectory-driven vehicles in real time with the focus on achieving the highest possible detection speed without sacrificing much detection quality. Therefore, a Hyper-Tracking approach was used which combines two different methodologies, HBP for detecting the OOI and KCF for tracking it.

Then, the centroid of the tracked vehicle obtained from the Hyper-Tracker's output was retransformed onto the planar world using planar perspective transformation and used for determining the vehicles direction according to cardinal system. At the same time, the detected-tracked centroids were also used in corporation of other metrics for identifying the status of rail-switches relative to the vehicles.

Then, the above-mentioned information was used on a couple of a collision scenarios. In the first scenario, two vehicles arrive at the critical area at the same time and in the other one, the two vehicles arrive at different time with different velocities.

It is important to mention that the goal is to show that for limited scenarios, the system can recognize collision based on the information extracted throughout the whole process of the system. Nevertheless, the system as in its preliminary stage, cannot be generalized for detecting every possible collision on ATLAS-demonstration plant. This means, by extending the system such as adding a third camera, a new vehicle or even extending the rail-model, the systems capabilities against detection collision must also be considered accordingly.

To make the system more generalized for detecting collision and robust against any extend needed for future implementations, following aspects should be considered on two levels:

1. Computational and Performance Improvement.

    I. In the evaluation chapter, we saw that with integrating KCF, an average speed of 94 FPS was reached (on an Intel® Core™ i5-8250U CPU @ 1.6GHz). By incorporating the second camera, the speed drops down to ~50 FPS. In the current implementation, KCF uses raw pixels for updating its predicted bounding box. However, if we use additional features e.g., HOG as tracking features, we can almost double the current speed (refer to Table in Figure 18). This is important and efficient as it is expected that the speed would drop when more cameras are incorporated. However, this is not the case with, for example HOG features.

    II. Exploiting the technical aesthetics of GPU. The camera system should in the future cover up to 30m long of rail modules which are horizontally aggregated. This requires up to 6 cameras with 2.3 mega pixel. As a result, the computation speed would drastically drop. However, the system does the same exact computations for all data arriving from different cameras using single CPU-core. Instead, these computations can be executed in parallel. In other words, they can be distributed on multiple cores and thus significantly improve the performance. Since GPUs are, by large margin, a lot faster than CPUs when doing parallel programming, they can be deployed instead, and this would further improve the performance

2. Increasing System Complexity

    The system should be extended to consider more metrics that contribute to more efficiently decision-making by collision detection. As an example, can be estimating the speed of each vehicle which helps knowing when a vehicle reaches a collision node.

    In this work, only two collision scenarios were considered. This is not enough to generalize the system for all possible collision scenarios. In other words, the more the scenarios differ, the more the system must be extended, and probably, the more parameters must be considered and tested.

References

[1] Christian Schultz, "ATLAS-Demonstrationsanlage für autonomes Fahren auf der Schiene ", 2018-University of Applied Science, Mittweida-Germany.

[2] Jo˜ao F. Henriques et.al, "High-Speed Tracking with Kernelized Correlation Filters"

[3] "Indexing via color histograms", Swain, Michael J. , Third international conference on computer vision,1990.

[4] polesky shape descriptors.

[5] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 2010, pp. 2544–2550.

[6] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(8):1619–1632, 2011.

[7] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr. Struck: Structured output tracking with kernels. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(10):2096–2109, 2016

[8] Jo˜ao F. Henriques et.al, "Exploiting the Circulant Structure of Tracking-by-detection with Kernels"- Institute of Systems and Robotics, University of Coimbra.

[9] Khaled. Jbaili, "Camera evaluation for system monitoring of ATLAS", Research Project University of Applied Science-Mittweida-Germany 2020.

[10] A.R. Zamir, A. Dehghan, and M. Shah. GMCP-Tracker: global multi-object tracking using generalized minimum clique graphs. In ECCV, 2012.

[11] Jo˜ao F. Henriques et.al, "High-Speed Tracking with Kernelized Correlation Filters"