UNIVERSITÀ DEGLI STUDI DI SALERNO

CONTEXT AWARE SECURITY ANALYTICS IN COMPUTER VISION

OCCLUSION HANDLING IN VIDEO SURVEILLANCE SYSTEMS

Professors: Michele Nappi, Ignazio Passero

Project coordinator: Carmen Bisogni

Student: Michelangelo Esposito – 0522500982

Academic year 2020/2021

INDEX

INTRODUCTION

Computer Vision (CV) is a scientific field that deals with how computers can gain high-level understanding from digital images or videos; in the last decades, impressive milestones have been reached in various fields: in medicine, segmentation of brain tumour has high clinical relevance for the estimation of the volume and spread of a tumour and skeleton segmentation techniques have been able to provide a fast and reliable 3D observation of fractured bones; in the security industry, CV techniques such as real-time face recognition or object detection, combined with biometry are able to provide an easier control over entire areas. However, despite all of these advancements, the dream of having a computer interpret an image at the same level as a human remains elusive. So, why must an "intelligent" machine resort to physics-based and probabilistic models to disambiguate between different possible solutions, when describing the world and reconstructing its properties is such an effortless task for humans? The problem is based both on the still limited understanding of biological vision and on the complexity of vision perception in a dynamic and nearly infinite varying physical world; simulating an entire brain artificially would obviously be of great help, however the technology isn't quite there yet, so we must resort to a simplified version of the brain by using a much more abstract version of neurons and synapses, in the form of Artificial Neural Networks (ANN).

Focusing on the application of CV in the video surveillance field, different problems arise, such as face recognition, object tracking or behaviour analysis. One particular challenge is the handling of occlusion in a scene: occlusion is the condition when one of the targets in a computer vision system is partially or completely hidden by another object in the current frame; today it remains a problem difficult to handle reliably.

Recent studies [3][4] classify occlusion into three categories:

- **Dynamic (inter-object) occlusion**: the most common type of occlusion, caused by the overlapping with another object, which is closer to the camera.
- **Scene (background) occlusion**: happens because of objects inside a background model that are actually located closer to the camera.
- **Apparent occlusion**: occurs because non-visible regions emerge due to pose/shape changes.

Some attributes are also defined to better understand how an object is being occluded: the extent of an occlusion is defined over which features of the target are visible and which are hidden; the duration of the occlusion defines for how long a section of the observed object is occluded; finally, the complexity of the occlusion is influenced by factors such as distance from the camera, size of the object or orientation.

In the next section I will provide a brief analysis of the state-of-the-art techniques for dealing with occlusion, before shifting the focus towards the developed Machine Learning project, its architecture, and the achieved results.

1. *STATE-OF-THE-ART IN OCCLUSION DETECTION*

In video surveillance systems, not dealing with occlusion make result is several … consequences, depending on the use case: if a tracker is employed in a real time camera system, for example, and occlusion occurs for even a brief amount of time, the tracker may lose is target and drift away into some other points of the scene. Therefore, it is necessary to mitigate the effects that occlusion has on video processing systems; the following techniques and algorithms provide a genera idea on how occlusion handling happens in the real world:

Kalman Filter Assisted Occlusion Handling (KFAOH)

This algorithm [1], focused on car occlusion detection, works through two periods namely tracking period when no occlusion occurs and detection period when occlusion takes place, thus depicting its hybrid nature: The Kanade-Lucas-Tomasi (KLT) feature tracker is responsible of handling the tracking steps of the algorithm, while a Cascaded Object Detector (COD) of weak classifiers, previously trained on a large dataset of cars, governs the detection of occluded objects, with the assistance of the Kalman Filter (KF).

Motion Models

Generally, after occlusion takes place, the subject can be found close to its previous location, thus a uniform local search around the area is capable of effectively finding the target, except when motion is unexpectedly fast. To alleviate this problem, probabilistic Gaussian motion models [3] are utilized. By putting more weights on locations closer to the previous location, this solution poses more bias on the target motion than uniform search, hence, fails easier in the case of fast and abrupt motion. Since the search area is predefined in such schemes, they are unable to handle persistent occlusion.

Occlusion Regions Detection

This approach is based on Bronx's energy function [2]: when an object moves, it creates two instability areas: region U', at the head of the motion vectors, and region U", at the tail of the motion vectors. In the first region there are two groups of instable pixels: the first one contains pixels moving outside of the next frame and the second contains pixels overlapping together in the next frame. Since the is not enough information, these pixels will not be accessed properly and thus the two regions create an occlusion boundary.

2. *FRAMEWORK DESIGN*

The main goal of this project is to build a small Machine Learning framework capable of identifying the presence of occlusion in the frames of a video. More specifically, given a video and an integer n as the input, the goal is to run the frames of the video through a neural network, classify them based on the probability of occlusion occurring in them, and saving the n least occluded ones in a folder.

2.1 DATASET AND DATA PRE-PROCESSING

The dataset used for this project is called GOTCHA-I. This set includes videos from 62 subjects in indoor and outdoor environments, in both cooperative and uncooperative mode; it has been put together with the goal of addressing both security and surveillance related Machine Learning problems. A total of 493 videos have been captured, where subject move in the environment with different behaviours, such as free movement, following a path pr going up a ramp of stairs. There are 10 different environments from which a total of 100 videos has been selected, equally distributed in each category and, when possible, containing the same subjects. It is also important to notice that this dataset wasn't built specifically with occlusion in mind, so the only two types of occlusion that occur in the videos are either caused by part of the subject being cut from the camera view or by small leaves/branches that are captured in front of the subject, for a very limited number of frames, as they walk in the scene.

A few examples of subjects from the GOTCHA-I dataset:



From these raw videos some processing was due, in order to extract usable information to later feed to a neural network in with the purpose of training and evaluating the classifier: first of all, each frame of these videos had to be extracted, and then needed to be labelled according to some criteria. Extracting the frames was a pretty straight-forward process: OpenCV for Python contains built-in methods for video processing and image manipulation, thanks to which the whole process has been easily automated. Then, for the sake of obtaining an initial reference for the labelling of the frames, the Machine Learning library OpenPose was used: this framework allows us to detect a series of key-points on a body in a scene and to save these in a variety of formats.
The following images provide an example of the key-point detection process:

OpenPose was adopted to extract from each video a set of JSON files, one per frame, containing such key-points: in particular, each file contains the list of people in the scene and their respective list of detective key-points, in the form of coordinate-accuracy pairs. The following image provides an example of such files:


```
{
    "version":1.3,
    "people":[
        {
            "person_id":[-1],
            "pose_keypoints_2d":[
                1401.61, 433.07, 0.955282, 1401.59, 444.98, 0.910681,
                1392.66, 444.966, 0.854473, 1380.95, 465.508, 0.892902,
                1380.93, 486.121, 0.93837, 1422.01, 444.965, 0.893203,
                1425.05, 465.575, 0.924373, 1424.93, 489.029, 0.946113,
                1401.66, 486.161, 0.948203, 1398.46, 486.191, 0.985984,
                1398.5, 527.181, 0.930556, 1398.62, 553.783, 0.95154,
                1419.04, 486.129, 0.926191, 1410.36, 524.375, 0.878712,
                1410.35, 553.869, 0.89859, 1401.45, 430.183, 0.966695,
                1404.54, 430.162, 0.895213, 1398.51, 433.027, 0.842837,
                1413.26, 433.012, 0.6742, 1404.39, 568.521, 0.804975,
                1413.23, 568.474, 0.745511, 1413.25, 556.814, 0.786393,
                1398.46, 565.567, 0.761373, 1395.54, 559.773, 0.746628,
                1398.68,553.942,0.73804
            ],
            "face_keypoints_2d":[],
            "hand_left_keypoints_2d":[],
            "hand_right_keypoints_2d":[],
            "pose_keypoints_3d":[],
            "face_keypoints_3d":[],
            "hand_left_keypoints_3d":[],
            "hand_right_keypoints_3d":[]
        }
    ]
}
```


Further hand and face key-points were not detected for computational efficiency sake; a total of more than 38.000 frames was processed this way.

From the obtained JSON files, some extra processing was necessary: first of all, there were several frames where multiple people were present or where, due to an OpenPose error, a skeleton was detected in an "empty" area of the frame (for example in a set of leaves on the ground). Then, each individual frame needed to be labelled according to some criteria, in order to decide whether it was "Good" (minimal or no occlusion of the subject) or "Bad" (significant occlusion of the subject). The first issue was resolved by analysing all of the skeletons detected in one frame and taking into consideration only the one closest to the camera: this was achieved by retrieving the height of the torso from the distance of two key-points in the list: the longer distance was chosen as the

reference subject. Initially, for the labelling step, the extracted images have all been classified according to the same thresholds: a frame is labelled as "Good" (or 1) if it's corresponding JSON file contains at least 14 key-points with an accuracy of at least 70%, otherwise it is labelled as "Bad" (or 0). This seemed like a good compromise between what is visually a good frame of a non-occluded or partially occluded subject, and what OpenPose is capable of detecting; however, two main issues were found: first of all, the number of "Good" and "Bad" frames was really unbalanced, with the second class largely outnumbering the first one; then, by using only this criterion, the first trained network developed a kind of bias towards outdoor subjects in the categories "10 - Outdoor path – cooperative" and "11 – Outdoor path – uncooperative", for which it worked really well, compared to indoor ones, for which the resulting probability scores were extremely low, even when the only occluded portions of the subject were his feet or lower legs. As a result, two different key-point thresholds have been used to classify the frames before the training and testing phases: for outdoor subjects in the categories 10 and 11, the threshold remained the same, however for indoor environments and for subjects in the 4, 5, 6, 8 and 9, the threshold was lowered to a much more achievable 7 key-points detected by OpenPose, with at least 60% accuracy. This also resulted in a more balanced number of good and bad frames.

Finally, prior to the training process, images have been resized to a shape of 150 x 150 pixels, in order to reduce load on the memory resources, both RAM and VRAM.

At this point, the processed dataset was partitioned into three categories, each contained in its own .csv file:

- **Train** (80%): this subset is the one on which the network will be trained and the parameters will be fit.
- **Validation** (10%): the dataset used as a reference for network behaviour during the hyperparameter tuning phase.
- **Test** (10%): once the training process is complete, this data will provide a final unbiased evaluation of the model.

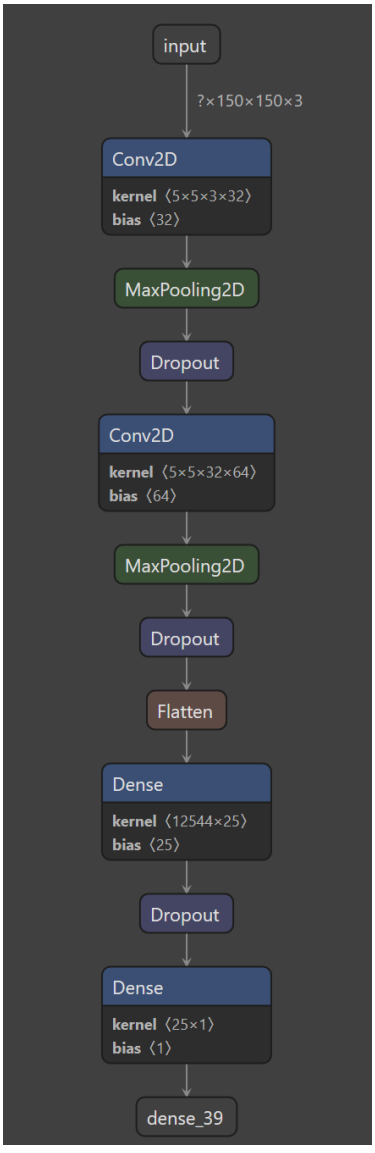
A different partition of the dataset, where the three categories take respectively 60%, 20% and 20% of the total data, has also been explored, however the training process was found to be more inconsistent this way. Additionally, the subjects in the validation and test subsets have been selected in a way that ensures they are completely unknown

to the network, in order to obtain an accurate evaluation, which would otherwise have been biased, in the case of applying a randomized partitioning approach.

2.2 CONVOLUTIONAL NEURAL NETWORK

Considering the nature of the dataset and the spatial relationship between the pixels in the images, the usage of a Convolutional Neural Network seemed like a solid choice to build the architecture of the Machine Learning model on which to train the classifier and build the framework: the ability of CNNs to develop an internal representation of a two dimensional image allows the model to learn position and scale in different varying structures found in the data; this is extremely important when working with images.

The trained network is made up of two convolutional layers, two max pooling layers, one hidden layer and the final output layer; the following image graphically summarizes the network architecture.



Convolution is a linear operation designed for two-dimensional data, it involves a multiplication performed between an array representing the input data and a two-dimensional array of weights, the kernel, which "slides" across the input data performing the multiplication several times in order to detect high-level features. The first convolutional layer is made up of 32 feature filters and a kernel size of 5 x 5, which is a pretty common choice; it takes an image of shape (150, 150, 3) as the input and applies the filters to it, outputting data of shape (146 x 146 x 32); the second convolutional layer is made up of 64 feature filters and a kernel of the same size; it outputs data in a shape of (44 x 44 x 64).

Pooling layers, on the other end, is generally added to the convolutional layer after a non-linear operation, such as ReLU, and is applied to the extracted feature maps to reduce the dimensions of the data; the chosen operation performed is max pooling, where the maximum value for each patch of the feature map is calculated. The result of using a pooling layer and down sampling the previously obtained feature maps is a summarized version of those features. The chosen cluster size is 3 x 3: this way, the first pooling layer receives data in the shape (146 x 146 x 32) , while the second receives data in the shape (44 x 44 x 64). After each max pooling layers, dropout is applied to prevent overfitting the model on the training data. This ensures a reduction in the probability of overfitting the model on the training data.

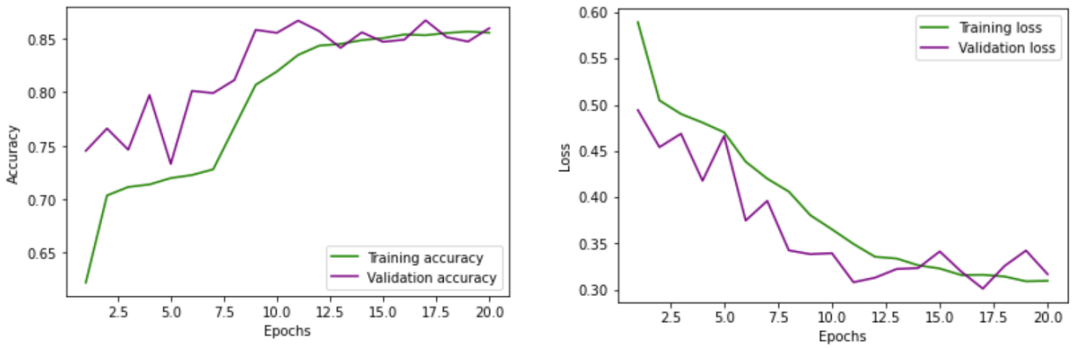
At this point the model is flattened, in order to feed the data to a traditional Artificial Neural Network, made up of one hidden layer with 25 units and the output layer which returns the probability score. The chosen activation function for most of the layers is the Rectified Linear Unit function, by which, given x, $f(x) = max\{0, x\}$; this function was chosen for its ability to make the network converge quickly.

After passing through the fully connected layer, since this is a binary classification problem that requires a probability score, the final layer uses the Sigmoid activation function to generate said score for the frame in the range [0, 1]: this score indicates the presence of occlusion in the input image, with a lower score indicating a higher occlusion in the scene. Finally, the model has been compiled using the "adam" optimizer with a learning rate of 0.001, as an alternative to classical stochastic gradient descent, and the Binary Cross-Entropy loss function; the dataset has then been fit on the built network for 20 epochs, using a batch size of 128.

Before arriving to the final network architecture discussed above, different configurations were considered during the hyperparameter tuning process. First of all, some experiments have been made with the number of convolutional/pooling layers and their number of filters, size of the kernel and size of the pooling cluster, before settling for the chosen values; using a higher number of units in the hidden layer was often cause of overfitting, while 25 seemed like a more reasonable choice. At last, different activation functions and optimizers have been tested, but due to the shallowness of the network, none of them resulted in a much noticeable difference.

2.2.1 PERFORMANCE EVALUATION

The following two graphs highlight the accuracy and loss results obtained during the training process:



As shown, the green line indicates train values, while the purple one indicates validation ones. Some spikes are clearly visible in some regions of the validation accuracy and loss, with a particularly noticeable one occurring around the 5th epoch: these can be attributed to the fact that, at each epoch, the validation accuracy and loss are usually computed on a smaller batch of the validation set, and since it is unseen data, more noise is expected in the resulting graph. In the end, however, once the 20th epoch is reached, the two lines manage to converge to an acceptable margin.

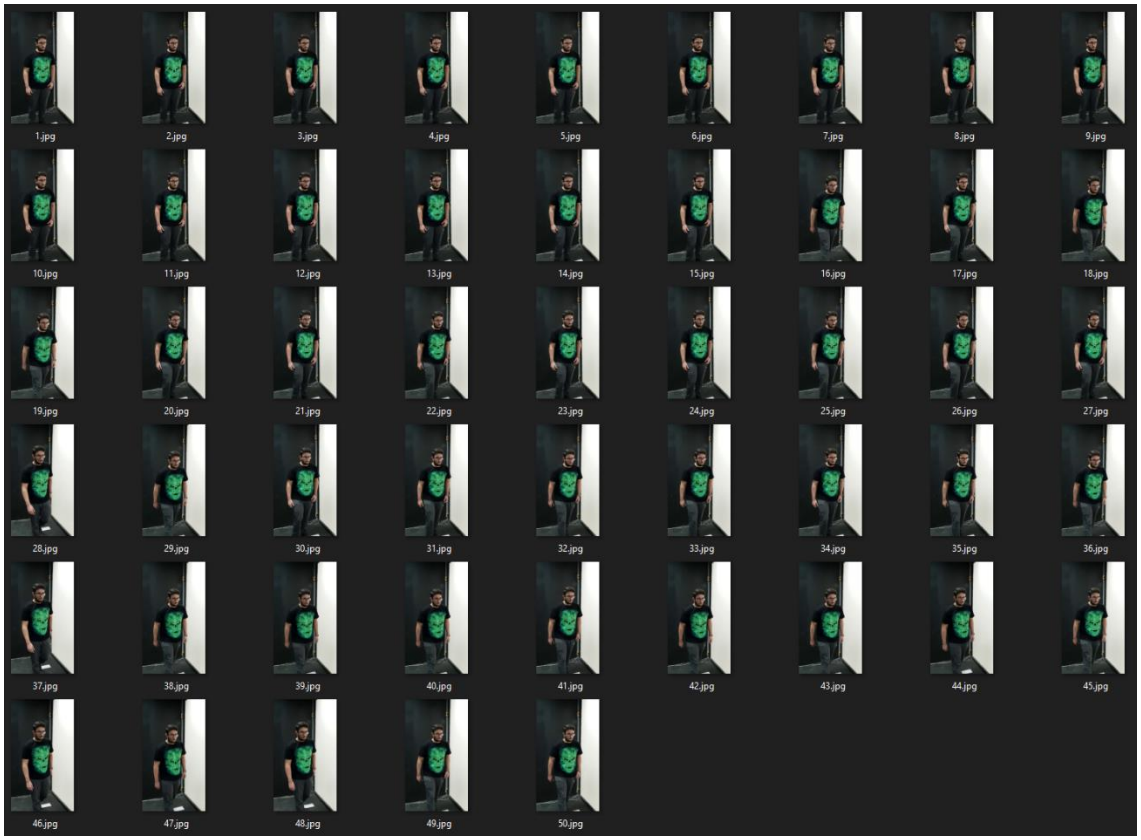
Regarding test performance, the overall accuracy and loss scores are respectively 0.81 and 0.4. So, when does the model behave poorly? First of all, let's consider the two indoor environments with flash light from the dataset: when videos from these categories were initially processed by OpenPose, the number of key-points detected was extremely low, thus resulting in the majority of these frames being labelled as "Bad"; the same result emerged from the processing of subjects wearing dark clothes, for which

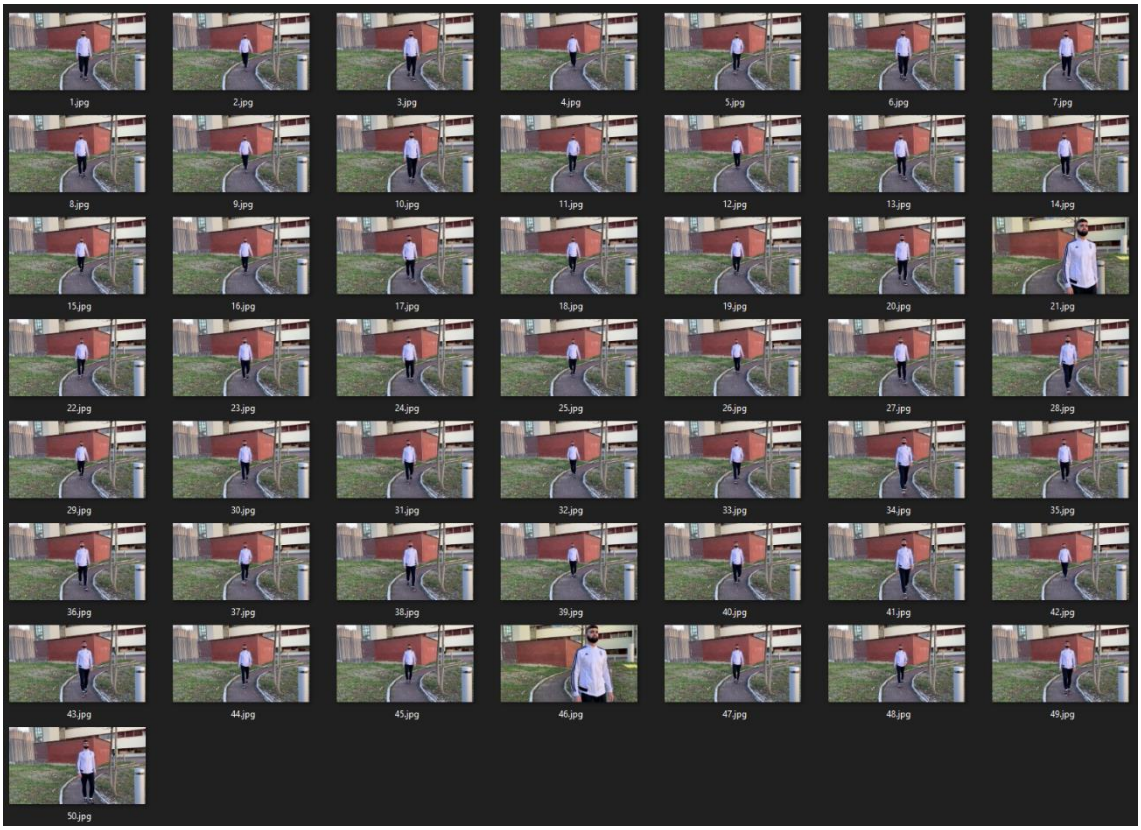
(always in the indoor environments with a dark background) the number of detected key-points on the region of the body covered by the dark clothes was, again, very low. The trained model, as a result, doesn't behave optimally in poor lighting conditions and when dealing with darker clothes on the subjects.

On the opposite side, when considering videos taken in an outdoor environment, such as the ones from the categories "10 - Outdoor path – cooperative" and "11 – Outdoor path – uncooperative", where the video is shot from a larger distance and with natural lighting conditions, the model behaves extremely well.

Conclusively, this suggests that the usage of OpenPose has had an impact on the final result, even though this does not really result in a bad, noticeable outcome, except in some of the areas mentioned above.

In most situations though, the model still behaves correctly as shown by the next results, in which the best 50 frames of two videos, not from the initial dataset obviously, have been extracted using the model.




1.jpg
2.jpg
3.jpg
4.jpg
5.jpg
6.jpg
7.jpg
8.jpg
9.jpg
10.jpg
11.jpg
12.jpg
13.jpg
14.jpg
15.jpg
16.jpg
17.jpg
18.jpg
19.jpg
20.jpg
21.jpg
22.jpg
23.jpg
24.jpg
25.jpg
26.jpg
27.jpg
28.jpg
29.jpg
30.jpg
31.jpg
32.jpg
33.jpg
34.jpg
35.jpg
36.jpg
37.jpg
38.jpg
39.jpg
40.jpg
41.jpg
42.jpg
43.jpg
44.jpg
45.jpg
46.jpg
47.jpg
48.jpg
49.jpg
50.jpg

REFERENCES

[1]: Joshi, Rakesh & Singh, Adithya & Joshi, Mayank & Mathur, Sanjay. (2019). A Low Cost and Computationally Efficient Approach for Occlusion Handling in Video Surveillance Systems. International Journal of Interactive Multimedia and Artificial Intelligence.

[2]: Synh Viet-Uyen Ha, Tuan-Anh Vu, Ha Manh Tran. (2018). An Extended Occlusion Detection Approach for Video Processing. School of Computer Science and Engineering, International University, Vietnam National University, Ho Chi Minh City, Vietnam.

[3]: Kourosh Meshgi, Shin Ishii. (2015). The State-of-the-Art in Handling Occlusions for Visual Object Tracking

[4]: R. Vezzani, C. Grana, R. Cucchiara. (2011). Probabilistic people tracking with appearance models and occlusion classification: The ad-hoc system.