Fast Foreground Object Detection Methods Designed for Ultra High Definition Videos

UHD 映像のための前景物体検出の高速化

February 2017

Axel BEAUGENDRE

ボジャンドル　アクセル

Fast Foreground Object Detection Methods Designed for Ultra High Definition Videos

UHD 映像のための前景物体検出の高速化

February 2017

Waseda University

Graduate School of Fundamental Science and Engineering

Axel BEAUGENDRE

ボジャンドル　アクセル

Abstract

Foreground detection is one of the most popular topics in computer vision in the past decades. The main objective is the extract foreground objects from the background scene. Foreground objects can be defined as all objects of interest such as humans or cars for example in applications such as video surveillance or automatic cars. In the past, multiple categories of methods have been developed like the background subtraction, the temporal difference or the optical flow. The topic remains very challenging even for known methods like the Gaussian Mixture Model (GMM)(C.Stauffer, ICCV 1999) due to issues such as change of illumination, stationary objects, complex background or moving background parts (trees, grass, etc.). The background subtraction method is the most efficient method for fixed scenes. The background model is subtracted from the current frame to extract the foreground objects into a foreground mask. To be used in applications such as tracking system, it is required to extract all connected components into a list of foreground object boundaries through a labelling process. To have an efficient system, it is necessary to detect as few false positives (FP) as possible.

The Ultra High Definition (UHD) is the next standard of video resolution. There are two main resolutions: the 4K UHD (3840×2160) and the 8K UHD (7680×4320). This increase of pixel number compared to standard definition videos comes with three new challenges in addition to the existing ones. First, the computational time for the foreground detection is exploding and can take about 10 days to process a 4K video of 10 seconds long by methods like the Adaptive Gaussian Mixture Model (AGMM) (Zivkovic, PRL, 2006) with an Intel Quadcore i7@2.83GHz CPU (Beaugendre, ITC-CSCC 2016). Such methods designed for SD are clearly unfit for UHD. With a higher resolution, it is easier to detect small movements from the background like moving leaves, but it increases the number of FP. Finally, UHD cameras can capture tiny objects or objects located very far. Their size can represent 0.01% of the image size. Therefore, down-scaling the image before detecting foreground objects is not a suitable approach since small objects would be missed.

This dissertation presents three works to reduce the processing time of UHD videos and more particularly of the foreground detection process with the aim of making it much faster than existing software approaches. They revolve around the principle of using the results from the previous frames to improve the quality and the speed of their process. The Real-Time Refinement Method for Moving Object Detectors (RTRM) is a real-time filtering method, independent from the resolution, which reduces greatly the number of false positives and improves greatly the detection rate. The goal of the Adaptive Block-Propagative Background Subtraction (ABPBGS) is the processing time reduction by focusing on objects to skips unnecessary areas while still being able to detect very small objects. Last, the Mixed Block Background Modelling (MBBM) reduces greatly the computation time of the update of a UHD background model.

Chapter 1 [**Introduction**] first introduces the field and the applications. Then several background subtraction methods will be presented. After that, the differences and issues of SD and UHD will be addressed. Finally, an introduction of our major contributions will conclude the chapter.

Chapter 2 [**Real-Time Refinement Method for Moving Object Detectors**] (RTRM) introduces a resolution free moving object processing layer which reduces the number of false positives and miss detections from the results of existing foreground detection methods. Tracking methods such as the Particle Filters (PF) (Ye, CMSP. 2012) (Tang, ICCS 2010) or the Kalman Filter (Zhao, IEEE TPAMI., 2004) are mainly based on visual appearance (color, contour, etc.) which requires too much resources to track objects in real-time on UHD videos. The RTRM uses the position in the previous frame of the OOI to filter the detected objects and remove unnecessary objects while keeping the interesting ones. Also in the case of occlusions, fusions of objects can happen. The fused object will be split into as many new objects as the number of OOI overlapping it. Then the new objects are moved for a better space occupancy of the fused object. With one detected object per OOI, the matching can be done by a Hungarian algorithm (Gross, ICWN 2003) on the distance between objects and the previous position of the OOI. The RTRM has been tested on our HD videos and one 8K UHD from NHK sources with the CLEAR metrics (Kasturi, IEEE TPAMI. 2009) which defines the N-MOD/TA (Normalized Multiple Object Detection/Tracking Accuracy, best score=1) and the N-MODP (Normalized Multiple Object Detection/Tracking Precision, between 0 and 1).

On the 8K video, the proposed RTRM reduces the FP from 2634 to 0. As a consequence, the RTRM greatly improves all the results compared to the SBGS: -41.2→0.98 for the N-MODA, -42.9→0.92 for the N-MOTA and 0.10→0.50 (+400%) for the N-MO(D/T)P. The RTRM average speed is $4.2 \times 10^{\ 4}$ s and the tracking average speed is $2.7 \times 10^{\ 5}$ s which makes it suitable for real-time applications.

Chapter 3 [**Adaptive Block-Propagative Background Subtraction**] (ABPBGS) presents a novel foreground detection method solution which aims to reduce the number of FP and computational time of UHD sequences by skipping parts which do not contain foreground objects. Current methods such as the Pixel Based Adaptive Segmenter (PBAS) (Hofmann, CVPRW 2012) or the AGMM, process the full frame in any circumstances. The Region-based Moving Object Detection (Berjon, ICCE 2014) uses random blocks in addition to ROI blocks to detect foreground objects. However, in their method, the objects have a size and shape requirement. The proposed ABPBGS is a detection method which starting from a seed block, propagates itself from block to block as long as a part of an object has been detected. The ABPBGS is able to detect a full object from a single block and independently from the shape or size of the object. This proposal skips unnecessary areas and even though it can detect enormous objects, it is especially suitable and efficient if the objects of interest are much smaller than the video resolution. The ABPBGS has been tested on a publicly available custom dataset with five various scenes (different size of objects, background noise, etc.) in 4K and a scene in 8K UHD. Compared to the PBAS, the best state-of-the-art method from Sobral's survey (CVIU 2014), the ABPBGS reduced a lot the processing time per pixel $3.78\times10^{\ 6}\rightarrow2.33\times10^{\ 8}$ s/p (-99.38%). For the 8K sequence, the average memory usage of the ABPBGS is greatly diminished compared to the PBAS 11 GB→450 MB (-96%) by skipping most parts of the image. The pixel-based detection quality metrics show that the ABPBGS reduces the FP and FN with an average precision (between 0 and 1) improved 0.327→0.495 (+51%) as well as a better recall score 0.275→0.384 (+39%). Using the F-Measure, Similarity and SSIM measures (between 0 and 1) the ABPBGS achieves a better average quality score 0.493→0.572 (+16%) compared to the PBAS and 0.482→0.572 (+19%) compared to the AGMM.

Chapter 4 [**Mixed Block Background Modelling for Foreground Detection in UHD videos**] (MBBM) presents an amelioration of the ABPBGS by including a background modelling method solely designed for very big resolutions. Whether recent background modelling methods are block-based (Deng, ICASSP 2008) or pixel-based (Yao, CVPR 2007), they still process all the image and require the use of multiple parameter kernels for each block or even pixel. Their high complexity and memory requirement make them unfit for UHD videos. The novelty of the MBBM lies in the reduction of the background model computation time by updating very small parts (blocks) instead of the full image at once. To keep the homogeneity of the model, two blocks per region are selected by a mixed selection: one through a linear order selection and one by a pseudo random selection. Then the background model parts corresponding to the selected blocks are updated through a Gaussian average (Wren, IEEE TPAMI, 1997). Finally, the MBBM includes the ABPBGS background subtraction method to extract foreground objects. Twelve UHD 4K scenes from 4 categories (small objects, big objects, monitoring, stationary objects/illumination) have been used to test and compare the MBBM with the state-of-the-art methods and the proposed ABPBGS. The MBBM has a better precision rate (between 0 and 1) than the ABPBGS 0.466→0.585 (+26%), the PBAS 0.401→0.585 (+46%) or the AGMM 0.313→0.585 (+87%). The MBBM can handle the various challenges with an average score of 0.597 using the F-Measure, Similarity and SSIM measures (between 0 and 1). The MBBM clearly outperforms the other methods: ABPBGS 0.536(+11%), PBAS 0.548(+9%), AGMM 0.489(+22%). Even though the MBBM updates the background model, its processing time per pixel is lower compared to the ABPBGS $3.37\times10^{\ 8}\rightarrow3.18\times10^{\ 8}$ s/p (-5.6%) or to the PBAS $3.77\times10^{\ 6}\rightarrow3.18\times10^{\ 8}$ s/p (-99.16%). A last tracking test on an 8K video using the MBBM combined with the RTRM resulted in an average total speed of 156 ms/frame (6.46 fps) with a N-MODA of 0.99 and a N-MOTA of 0.98.

Chapter 5 [**Conclusions**] finally concludes this dissertation by summarizing the different proposals and discussing the future work.

Acknowledgements

This thesis is the work of not just one person but of a multitude of people who helped or supported me. I would like to acknowledge all of you here.

First of all, I would like to express my deepest thanks to my supervisor Professor Jiro Katto for his council and help.

I would like to thank Professor Satoshi Goto without who nothing would have been possible. You are the one who gave me the opportunity to come and realize a dream of mine. Your patience, motivation, enthusiasm, knowledge and guidance helped me during all my PhD and your advices, both in research and in life, were greatly appreciated.

I would also like to thank Professor Takeshi Yoshimura for accepting me in his laboratory after the retirement of Professor Goto and for guiding me through the final stage of my Ph.D.

I would also like to thank Professor Shinji Kimura and Professor Nozomu Togawa for their precious advices and insightful comments which helped me greatly for improving my dissertation and for all the guidance through my research.

My thanks to Professor Yves Lepage for his conversation and advices during my stay in Waseda University of IPS.

My next thoughts go to my parents and my family who believed in me and who always supported me through good and very difficult times. Even though we were separated by 14 000 km I could always feel your support, trust and love.

I would like to thank my girlfriend Susana da Cruz Neves who has been the greatest support I could dream of during the writing of this present dissertation. Thanks to you I could focus on my work and your support was an oasis of hope during the difficult times.

A thought for my friend Ella Blanquet who always supported me and helped me many times the past years.

I would also like to thank all the people from Professor Goto's laboratory. To Ms Izumi who has been of a great help during all my PhD. To Professor Daijiang Zhou, Dr. Jinjia Zhou and Professor Xin Jin for their inspiring works and advices. To Dr. Minghui Wang who his kindness, advices and care in my daily life. To Dr. Jiu Xu, Dr Wenxin Yu, Dr. Jiang Ning, Dr Xinwei Xue for working together and giving me advices on the image processing field. Special thanks to Ms Chenyuan Zhang for her kindness, happiness and for her collaboration and assistance in multiple projects. To all of the students who have been in the laboratory for their welcome, for communicating with me and for including me even though I could not speak chinese.

Last, I would like to thank all the people I met and interact with like Mrs Tamaki, Mr Benny Lee

Finally I thank the "Ambiant Soc Global COE Program of Waseda University" of the Ministry of Education, Culture, Sports, Science and Technology, Japan, Core Research for Evolution Science and Technology (CREST) for supporting my research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Video understanding and Foreground Object Detection

Image understanding is a very big part of the computer vision field. It consists of extracting and analyzing the content in images. The task is very complex to reach the level of what a human being is capable of. Indeed, an image is the capture of the world under the form of matrices of numbers and understanding an image is being able to translate back those matrices into clear and understandable information for other applications. An extension of the image understanding is the video understanding which includes the notion of temporality and events. Most of image and video understandings are transposed from the human brain functioning to an artificial process. For example, an image is the digital transposition of what the brain perceives through the eyes. For human beings, moving objects tend to get more attention, objects with distinguishable colors are more easily recognizable or we can sort objects by their shape.

The foreground object detection is the process of extracting in an image all objects which do not belong to the background. The task is easy for a human mind but it is very challenging to emulate it in an artificial process. Background subtraction, optical flow or temporal difference methods are different techniques which can be applied to detect foreground objects (also called moving objects) in a video. They can result in two kinds of output: a pixel-level foreground mask which is a visual state of the foreground or an object-level output with a set of object contours. Those outputs are still very raw but they are nonetheless valuable for their interaction with other systems which usually require to process all or a large area of the image.

1.2 Foreground Detection in Applications

The detection of foreground objects is only a means to the end, the foreground mask or the detected objects are meant to be use in other systems. Indeed, the moving objects can be used as region of interest (ROI) to focus the next processes on a much smaller area. Basically, all applications which require to know what is moving in the field of view is a target application. The most common applications which have a usage of such objects are tracking-based applications such as remote, video-surveillance [1], traffic monitoring [2] or people tracking. More specific detection systems like augmented reality systems, human detection or face detection can also benefit from the foreground detection. For example, Zeljkovic [3] presented a personal access control system based on moving object detection and face recognition. Berjon [4] presented an augmented reality system based on moving object detection. Those are examples of particular use of foreground detection but the main use are the tracking based applications.

Most of the application want to focus their interest on a specific target for identification or tracking [5, 6, 7]. Human detectors or face detectors look for a reduction of field of research. Even though some of the moving objects do not contain a human being or a face, the search window is restricted to the most probable zone.

Another field which is using more and more image and video processing is the medical area. Indeed, advance medical science has been using computer vision to help detect anomalies and search for treatments. An example of that is the work of Chao et al. [8] who used computer vision to track drosophila specimens.

This motivation of using moving objects to reduce the processed area is even bigger when

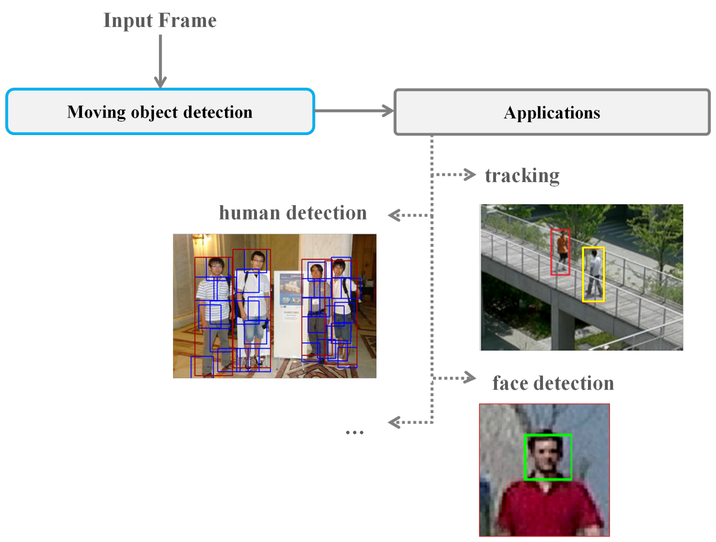
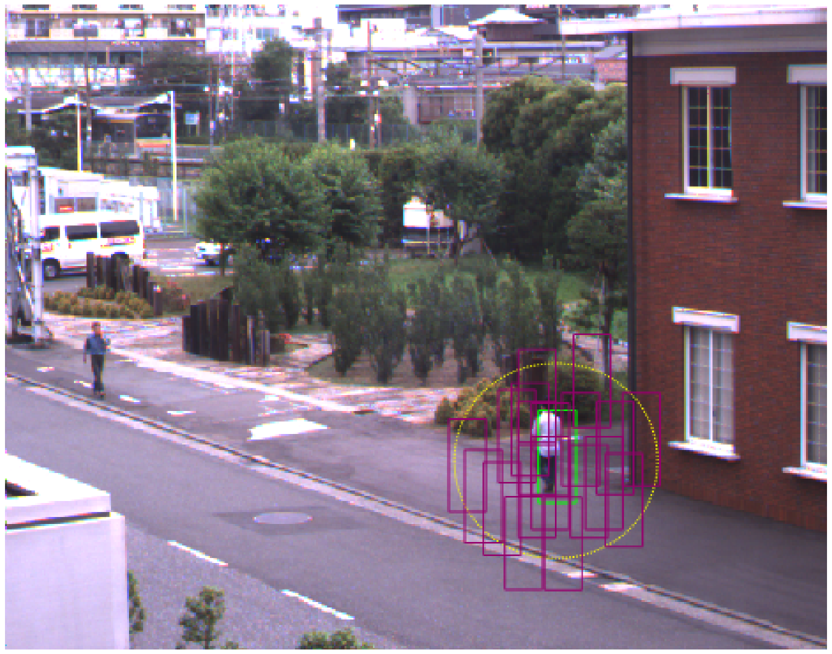


Figure 1.1: Example of applications which can directly use background subtraction results as an input.

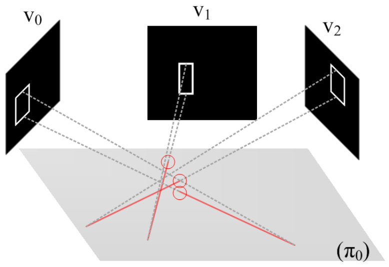
the input source has a high resolution and the detection method complex. Depending on the application compromises have to be made and the level of quality required for the detection is different. A face detection system will be satisfied even if the shape of the objects detected does not perfectly fit the reality. On the contrary, a tracking system or a human detection application would benefit greatly of having the right shape for all moving objects in the scene.

1.3 Background Subtraction

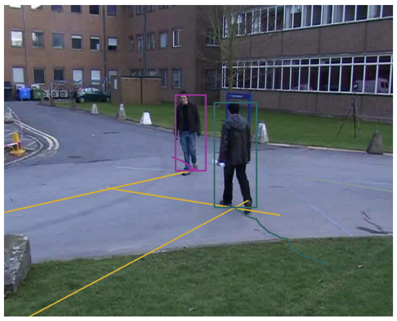
There are multiple approaches to do foreground detection, also called moving object detection. They can be categorized in three categories: background subtraction, temporal difference of frames and optical flow [9, 10, 11]. In this dissertation, we will focus more on the background subtraction methods which are very suitable for static environments. The general principle (1.3) based on the Static Background Subtraction method is very simple: considering a known background image, the foreground is obtained by subtracting it to the current frame. All the differences detected should represent all the objects which are not part of the original scene. However, due to numerous reasons such as optic noise, compression artifacts, illumination



(a) Particle Filter Tracking (dataset from NEC)



(b) Multi-View Projection from detection blobs



(c) Multi-View Tracking (dataset from PETS2009)

Figure 1.2: Two examples of tracking applications which can use moving objects as a feature. The particle filter can use moving objects to measure faster the weight of each particle depending on the distance and the overlapping they have to them. In a multi-view application, it is possible to obtain a 3 dimensional estimation of the position of moving objects thus reducing greatly the number of regions to track in.

differences or simply parts of the background which are moving, it is necessary to further filter the subtracted image.

The first filter is known as the Threshold filter. All pixels which have value above the threshold value T will be set to white (255) and black (0) in the opposite case. The adjacent foreground pixels do now form connected components also called *blobs*. Follows morphologi-

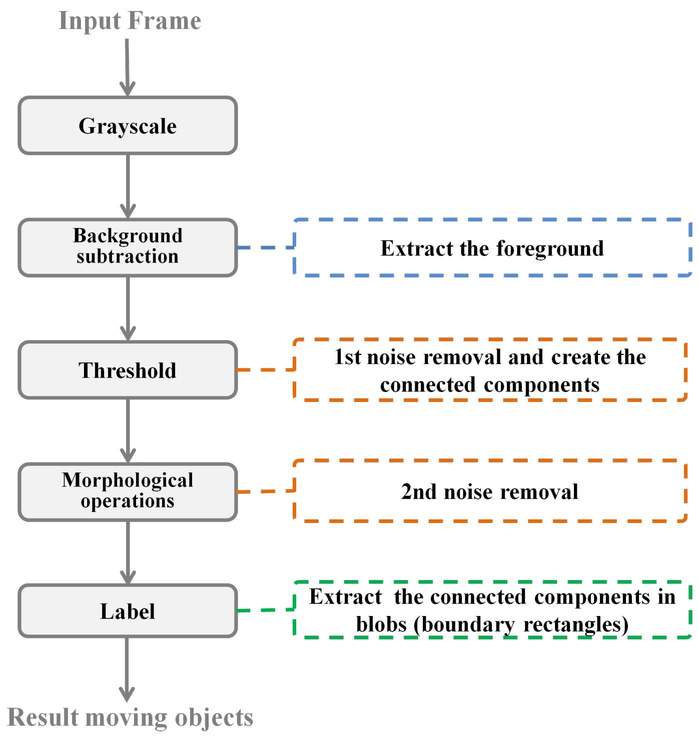


Figure 1.3: Background subtraction process step by step.

cal operations (erosion, dilatation, open, close) to fill small holes in the structures or remove noise. The last operation will label all the blobs. By doing so, all connected components will have their boundaries computed and extracted with a distinctive index label to identify each one of them. This operation is one of the costliest in terms of time consumption depending on the number of blobs detected. The result of such label will result of a set of contours, usually rectangles for their simplicity. Figure 1.5 presents a simple example of a background subtraction outputs.

The static background subtraction method is the simplest and also the fastest due to its simplicity and to the fact that the background model is not updated. All more complex method which would update the background would therefore be slower. In real applications, many factors make the background subtraction hard to do. All the little differences between the current frame and the background will be as many potential noise pieces and as many

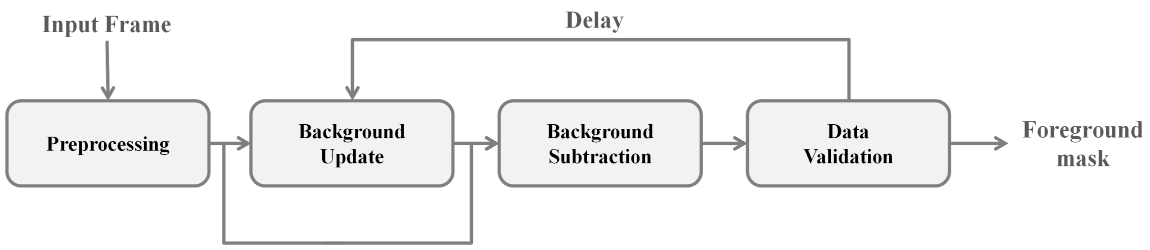
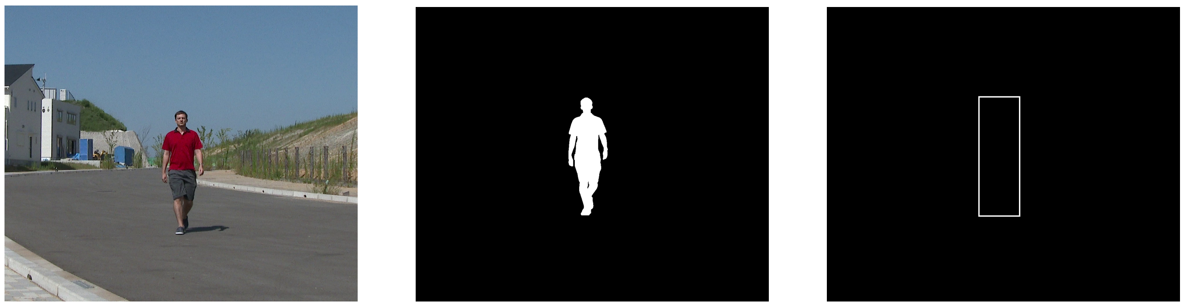


Figure 1.4: Background subtraction flowchart including the background modelling.

fragments. Blob fragments happen when a part of an object is perceived as background thus splitting the object into small parts. A change in the illumination, some tree's leaves or a moving electric cable are only examples of the many sources of incorrect detections. A slightly different approach based on successive frame difference, called Frame Difference, can deal better with tiny changes between two frames, however it may fail if the object stops moving.



(a) Original image (b) Foreground mask (c) Result Blob

Figure 1.5: There are two output results after a background subtraction. In the middle the raw unprocessed foreground mask; on the right the rectangle boundaries (or blob) of the detected object extracted from the foreground mask by a label process.

Strictly speaking, the differences between background subtraction methods do not lie much in the subtraction itself but mainly in the background modelling [12, 13]. Figure 1.4 presents the flowchart of a background subtraction system which includes a recursive background modelling. A model of the background can be computed on the go by calculating the weighted mean of the successive images as in [14]. This way the background is updated frame after frame. The drawback of this approach is that, in regions containing foreground elements, the background will be updated by using both background pixels and the foreground pixels [15].

Some approaches are statistical background modelling methods. This category includes the single Gaussian model and the multiple Gaussian model. The single Gaussian model can be reduced to a subtraction of each frame by a background image and verifies if the difference is bigger or smaller than the threshold. This method can fit for application where only small modifications happen to the background. The multiple Gaussian model also called Mixture of Gaussian (MOG) or Gaussian Mixture Model (GMM) [16] is the extension of the first method by using more than one Gaussian distribution. Its main use is image segmentation so for detection purposes, each component of the model can represent a different aspect of the background such as the colors.

Background subtraction can also be achieved by a lot of other approaches. For example, *fuzzy* theory can be applied to background modelling [17].

Using neural networks [18] is another approach for background modelling but it requires one neural map per pixel which makes it very expensive if the dimensions of the background are high.

Modelling a background can require a lot of resources depending on its complexity and its consumption can become a dramatic issue in the case where the resolution of the input images gets very big.

1.4 From SD to UHD and more

The video hardware for display and capture has greatly improved in the last years and we now see the new generation of displays and camera on the market. This new generation is the Ultra High Definition Television (UHDTV) [19, 20, 21] coming with the capacity to capture [22, 23] and to display [24]. The UHD resolution succeeds to the High Definition (HD) which has succeeded to the Standard Definition (SD) previously. The HD includes resolutions of 1280×720 pixels and 1920×1080. For noun simplifications, we will consider all definitions and resolution below the HD, such as VGA (640×480) or QVGA (320×240) as SD.

The UHD has two resolutions: 3840×2160 pixels called 4K UHDV and 7680×4320 called 8K UHD. As of today, most of the researches on the topic of UHD have been mainly focused on encoding/decoding matters [25, 26] and the rare computer vision works also focus on image quality [27]. The UHD datasets available [28, 29] are the proof of that since the sequences are all about image quality.

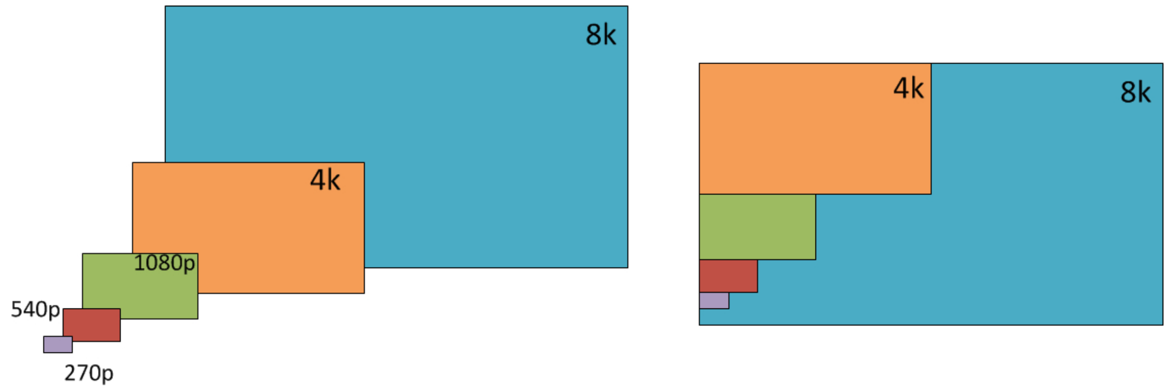


Figure 1.6: Representation of the different resolutions from 270p to 8K UHD

Few UHD dataset are available and most of them are not suitable to foreground detection purposes [28][29]. Indeed, those datasets contain movie-like videos in which objects are rather big and the image quality is exceptional. The level of detail of the videos used for foreground detection is on the contrary usually quite low. Until now, the majority of researches on foreground detection are based on Standard Definition (SD) videos as the various dataset can testify [30, 31, 32, 33]. Images contain a rather limited amount of information defined by the hardware it had been captured from. The resolution and definition are important factors for a better understanding and work better by pair. Figure 1.7 show blocks of the same size but from images of different resolutions. With a very high definition, objects require a lot more pixels to be comprehensible. This is why a small image with a good definition is better than a bigger image with a low definition. For this reason, it seems interesting to decrease the size of an image by down-scaling it to get a higher *important pixel* density.

For the same object-size/image-size proportions, a higher resolution implies a higher pixel density for the objects. Therefore, objects which are made of very few pixels in SD are defined

Table 1.1: Pixel count comparison for the different resolutions from QVGA to 8K UHD.

Resolution Name	Resolution	Pixel count	Ratio to 8K
QVGA	320×240	76800	1/432
270p	480×270	129600	1/256
VGA	640×480	307200	1/108
HD 720p	1280×720	921600	1/36
HD 1080p	1920×1080	2073600	1/16
4K UHD	3840×2160	8294400	1/4
8K UHD	7680×4320	33177600	1

by hundreds or thousands of pixels in UHD. This characteristic opens a whole new kind of videos in which there can be very small objects compared to the video resolution. Figure 1.8 presents an example of the new possibilities of UHD videos. When compared to an image taken from the BMC dataset [31], the ROI seems of normal size and the object of interest appears quite big, but it is in fact just a tiny piece of the original image.

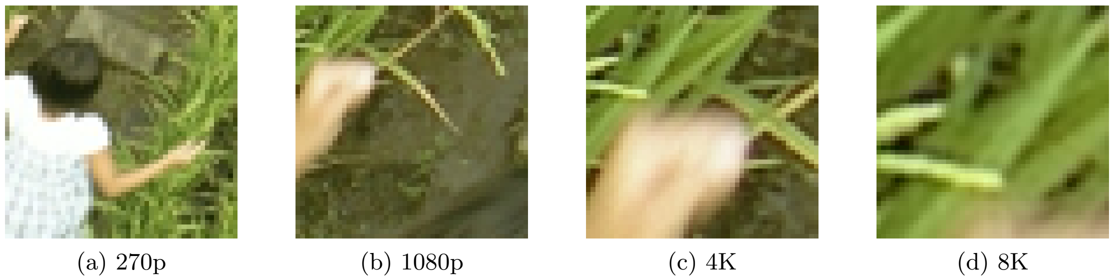
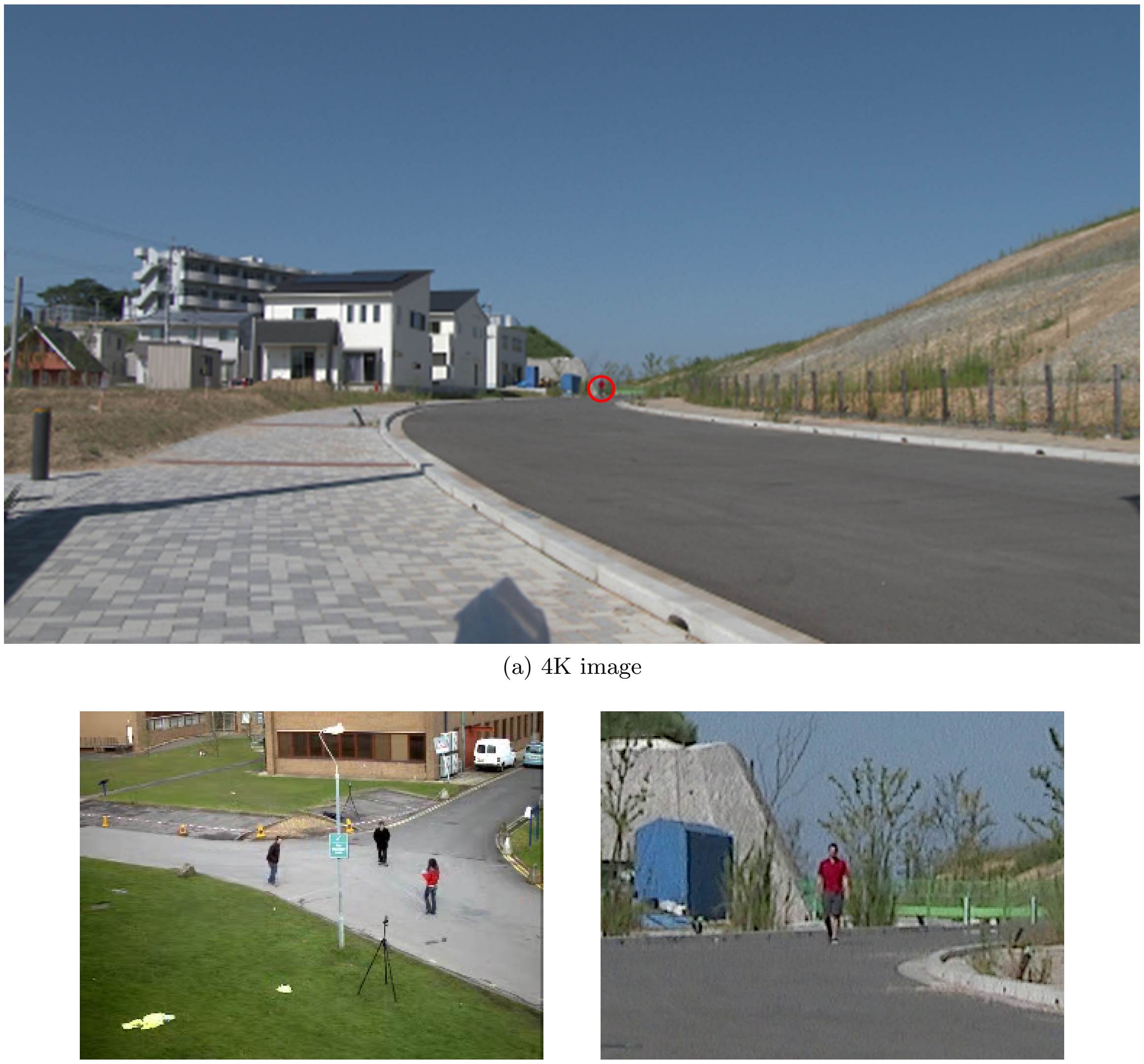

(a) 270p (b) 1080p (c) 4K (d) 8K

Figure 1.7: Blocks of 50×50 pixels from the same image at different scales from 270p to 8K UHD.

Such an increase of the field of view can have many applications such as videos captured by satellites, videos of sparse microscopic life or any kind of videos which would contain very tiny or far objects.

We believe it is important to tackle as soon as possible UHD processing since even bigger resolutions are already on their way like the 250 Mega-pixel (19,580 × 12,600 pixels) CMOS sensor unveiled by Canon (http://www.canon.com/news/2015/sep07e.html).



(a) 4K image

(b) Image from BMC dataset (320×240 px)

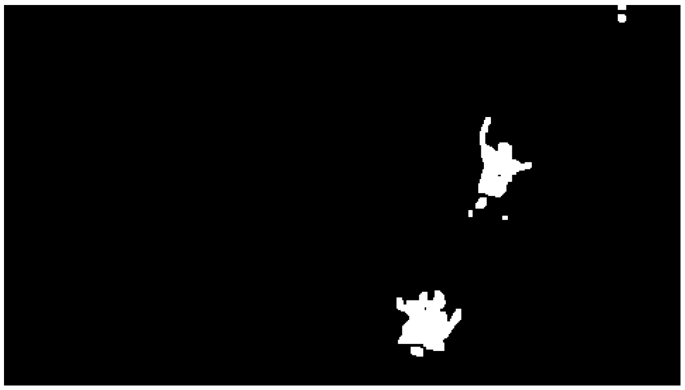
(c) ROI of 4K image (320×240 px)

Figure 1.8: Comparison of object sizes and resolutions. In the 4K image on top (downscaled for this screen), the object is so small compared to the size of the total image that is seems almost invisible. On the bottom, comparison of the image taken from the BMC dataset and the ROI of the 4K image represented at the same scale, side by side.

1.5 Problems

1.5.1 Noise sensibility and False Positives

In UHD videos, all the scene benefits from the increase in definition, from the foreground to the background. In SD videos, the background noise is relatively small and only big movements can be captured/detected. In UHD videos, the movement being bigger, it becomes more easily detectable by foreground detectors thus creating false positives. Figure 1.9 shows the difference between the processing of an 8K UHD video and its down-scale equivalent at 270p. There is almost no false positive in the SD scale but the UHD scale contains a lot of them. We can easily think of morphological operation such as closing and opening to reduce the noise. The task of removing the noise through such operations is manageable in single images, it is not easily achievable automatically in a video where the size of the noise changes every frame.


(a) 270p

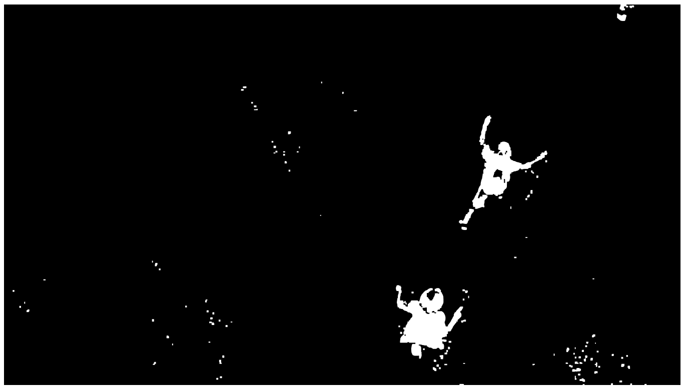

(b) 8K

Figure 1.9: Detection result in 270p (at the top) and in 8K (at the bottom). The background noise does not appear in the 270p scale while it is very perceptible in 8K

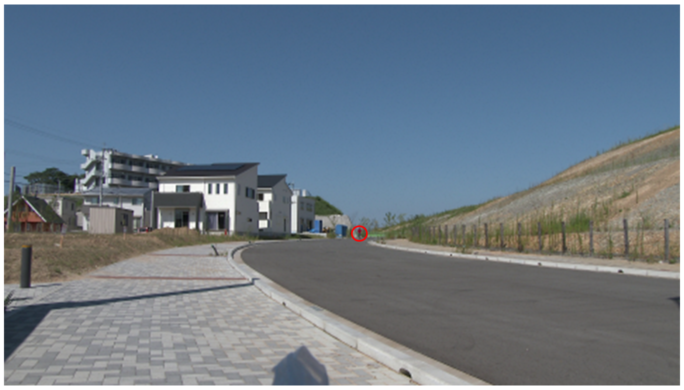
1.5.2 Time and memory consumption

For a real application, it is necessary to be able to process the video in a reasonable amount of time (¿5fps). The problem is that current methods are too slow to process UHD videos. Indeed, they have been designed for SD sources with the use of more and more powerful hardware. Therefore, with UHD and especially with 8K UHD, we multiply by 108 the amount of work to process compared to VGA. State-of-the-art methods such as the Pixel Based Adaptive Segmenter (PBAS) [34] can take more than 14600 s to process a single 8K frame, from the loading of the frame to the extraction of the detected objects by a labelling function. Table 1.2 shows a comparison of the total time estimated to process 12 4K UHD videos of a total length of 50 minutes. The fastest of the state-of-the-art method requires at least 23 weeks while the slowest is estimated at almost 8 years of continuous process. One could argue that 50 minutes of videos is big, but on the contrary, it represents only a fraction of the total length of video captured each day by video surveillance cameras. Therefore, for a method to be a potential solution for UHD computer vision, it is necessary that it does not require more than 10 times the length of the input video. Very recent works [35] [7] started to look for a way to speed up the detection process using spatio-temporal approach works with the use of Graphic Processing Units. Even so, they could only achieve 40fps for a 288×352 pixels video and the memory requirement is also quite high. Hardware optimizations cannot always compensate the slowness of software approaches and the more complex the method, the bigger is the gap that the hardware will have to fill. Also, it is important to realize that the foreground detection is an intermediate step in a bigger system, and the computing power cannot all be allocated on a single task.

Table 1.2: Average speed per frame and estimated time to the 12 4K videos of the dataset based on the *parking2* sequence. The total length of the dataset is about 50 minutes long.

Method	Time/frame (s)	Total Time (s)
PBAS	130.56	38.56 weeks
MultiLayer	3738.43	21.18 years
ZivkovicAGMM	1363.01	7.72 years
AdaptiveMedian	77.99	23.04 weeks
PratiMediod	292.40	1.66 years
WrenGA	1072.27	6.07 years
AdaptiveSelectiveBGLearning	299.83	1.70 years
IndependentMultimodal	4.02	1.19 weeks

The difference between the size of the image and the size of the objects of interest can have a big influence of the detection speed. If the case does not occur much with SD videos, it is a very big issue with UHD videos. Indeed, it is very questionable to process the entire frame if only a small part contains a moving object. Figure 1.10 shows an example of an image in which the object of interest is much smaller than the full frame. Knowing that a frame requires to goes through multiple processes from the color conversion to the label of the detected objects, it is unreasonable to process the full image for few tiny objects. Not only it requires much more time to process but it also requires a much higher amount of memory than necessary.


(a) 4K image

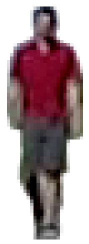

(b) Object of Interest (23×70)

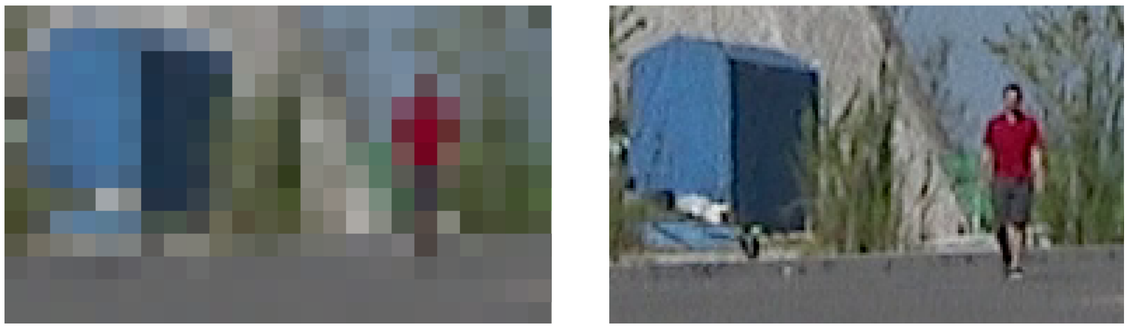
Figure 1.10: Compared to the size of the original 4K image, the object of interest occupies around 0.012% of the total area.

1.5.3 Down-scaling

Last but not least, down-scaling seems to be the way to go, but against preconceptions it is in fact not necessary an asset. We have seen previously that down-scaling was leading to fewer false positives but it can also make small objects of interest very tiny (composed with even less pixels to define them) or even make them disappear (Figure 1.11). There are scenarios which make it necessary to process at the original scale so small objects can be detected thanks to the higher resolution.

1.5.4 Objectives of the solutions

The ideal solutions of foreground detection methods for UHD sequences should focus on three axis: the reduction of false positives; the ability to detect all object of interest, including



(a) ROI from 270p down-scaled version (b) ROI from original image

Figure 1.11: Quality comparison of similar ROI at different scales. The shape of the object of interest is totally unrecognizable in the 270p scale while it is distinctively visible at the original scale (4K).

small objects; and the reduction of the processing time of foreground detection. Reducing the amount of noise will have two effects, improving the detection quality and reducing the time to label the objects. Being able to detect all the objects of interest is another key to better detection results but also correct tracking. Finally, a fast detection is a requirement for all systems which are based on foreground detection. A method which would take minutes or hours to process a single frame has very few real applications.

1.6 Contributions

In order to reach the objectives, the different works presented in this thesis are based on two principles: reusing the results from the previous frames and trying to process just the necessary by focusing on areas which contain the objects of interest. In this dissertation, we present four works related to foreground detection with the specificity to be designed for UHD sequences with two main objectives. The first goal is to increase the quality of the detected foreground objects so they could be used more efficiently by different applications such as tracking or face detection. This is the goal of the first contribution *Real-Time Refinement Method for Moving Object Detectors.* The second is to reduce the time consumption of foreground detectors while processing UHD videos. This is the subject of the other two research proposals *Adaptive Block-Propagative Background Subtraction System for UHD Videos* and

Mixed Block Background Modelling for Foreground Detection in UHD videos. If the main interest is obviously a much greater speed, it cannot be done at the expense of the detection quality.

1.6.1 Real-Time Refinement Method for Moving Object Detectors

The first contribution is the Real-Time Refinement Method for Moving Object Detectors (RTRM), a resolution free moving object filter which refines the moving object detection results obtained after a foreground detection. Its aim is to improve the detection quality through, among other things, the reduction of the number of false positives. The RTRM uses the position in the previous frame of the trackers to filter the detected objects and remove unnecessary objects while keeping the interesting ones. In the case of occlusions, fusions of objects can happen. The fused object will be split into as many new objects as the number of trackers overlapping it. Then the new objects are moved for a better space occupancy of the fused object. With one detected object per tracker, the matching can be done by a Hungarian algorithm on the distance between objects and the previous position of the tracker.

The RTRM has been tested on our HD sequences and one 8K UHD sequence from the NHK Nebuta festival. The results show a great reduction of the background noise, especially for the 8K sequence with 100% less false positives with the RTRM (2634$\rightarrow$0). The experiments also show that the number of objects after the RTRM matches the number of trackers under conditions such as moving background, occlusion between two trackers, or both. As a consequence, the RTRM greatly improves all the results compared to the SBGS: -41.2$\rightarrow$0.98 for the N-MODA, -42.9$\rightarrow$0.92 for the N-MOTA and 0.10$\rightarrow$0.50 (+400%) for the N-MO(D/T)P. The RTRM average speed is 4.2×10^{-4} s and the tracking average speed is 2.7×10^{-5} s which makes it suitable for real-time applications.

1.6.2 Adaptive Block-Propagative Background Subtraction System for UHD Videos

The second contribution is a foreground detector named Adapting Block-Propagative Background Subtraction (ABPBGS). The main idea is to skip all areas in the image in which there are no moving objects. This is particularly interesting for UHD when the objects of interest represent less than 0.013% of the total area. The novelty of this work leads in the detection which will detect and spread along the object as long as it detects a part of it. A block history map guaranties that each block is processed only once. It is a virtual grid containing the position of every block already and currently processed. Moreover, the detection loads and processes only small blocks saving computational time and memory usage. The propagation can also be parallelized to increase even more its speed.

Compared to the best state-of-the-art method, the PBAS, the ABPBGS reduced a lot the processing time per pixel $3.78\times10^{-6}\rightarrow2.33\times10^{-8}$ s/p (-99.38%). For the 8K sequence, the average memory usage of the ABPBGS is greatly diminished compared to the PBAS 11 GB→450 MB (-96%) by skipping most parts of the image. The pixel-based detection quality metrics show that the ABPBGS reduces the FP and FN with an average precision (between 0 and 1) improved 0.327→0.495 (+51%) as well as a better recall score 0.275→0.384 (+39%). Using the F-Measure, Similarity and SSIM measures (between 0 and 1) the ABPBGS achieves a better average quality score 0.493→0.572 (+16%) compared to the PBAS and 0.482→0.572 (+19%) compared to the AGMM.

1.6.3 Mixed Block Background Modelling for Foreground Detection in UHD videos

The last contribution is the Mixed Block Background Modelling (MBBM), a method solely designed for UHD videos. The novelty of the MBBM lies in the reduction of the computing time by updating very small parts (blocks) of the background model instead of the full image at once. To keep the homogeneity of the model, two blocks per region MB are selected by a mixed selection: one through a linear order selection and one by a pseudo random selection. Then the background model parts corresponding to the selected blocks are updated. The MBBM is an amelioration of the ABPBGS in order to deal with various and difficult situations such as changes of illumination or stationary objects.

The tests effectuated on 12 videos show that the MBBM is a major improvement of the ABPBGS. The MBBM has a better precision rate (between 0 and 1) than the ABPBGS 0.466→0.585 (+26%), the PBAS 0.401→0.585 (+46%) or the AGMM 0.313→0.585 (+87%). The MBBM can handle the various challenges with an average score of 0.597 using the F-Measure, Similarity and SSIM measures (between 0 and 1). The MBBM clearly outperforms the other methods: ABPBGS 0.536(+11%), PBAS 0.548(+9%), AGMM 0.489(+22%). Even though the MBBM updates the background model, its processing time per pixel is lower compared to the ABPBGS $3.37\times10^{-8}\rightarrow3.18\times10^{-8}$ s/p (-5.6%) or to the PBAS $3.77\times10^{-6}\rightarrow3.18\times10^{-8}$ s/p (-99.16%). A last tracking test on an 8K video using the MBBM combined with the RTRM resulted in an average total speed of 156 ms/frame (6.46 fps) with a N-MODA of 0.99 and a N-MOTA of 0.98.

Chapter 2

Real-Time Refinement Method for Moving Object Detectors [†]

2.1 Introduction

In the introduction chapter we have seen that foreground object detection is one of the most studied topics in computer vision and that it consists of the extraction of all moving objects or people from the environment.

Until now most of the existing moving object detection methods [9, 10, 11, 16, 36, 37, 38] have been focused on quality rather than speed. This choice can be a real issue if there is a need for real-time applications especially if the input video resolution is high. Simple but fast approaches suffer from their inaccuracy notably during occlusions. This lack of accuracy is the main reason why moving object is not the main feature of most tracking systems. False positive, fusion or fragments are as many reasons why tracking systems prefer to rely on appearance for matching. However, appearance-based features are extremely slow to compute, problem which is gets even worse with the increase of the size of the objects of interest.

The motivation for this present chapter is to improve the quality of the foreground detection so the extracted objects could be used directly for tracking. We propose here a new

[†]This chapter is adapted from the work published in [J5]

top-down refinement method for moving object detection systems. This extra layer between the detection process and the tracking process is able to remove remaining background noise and brings one answer to occlusions by splitting the fused objects using the tracking results from previous iterations as a new source of information. The major novelties of this proposed method are as follow:

1. We propose a fast refinement method (RTRM) adaptable to any existing foreground detection method as long as they produce objects.

2. The use of tracking results to categorize objects from the original detection depending on the number of trackers on the moving objects. No tracker will lead to the removal of those objects, one tracker will keep the object and associate it to the tracker, last two or more trackers will start the occlusion handling process which splits the original blob into multiple new blobs.

3. By having at all time the same number of blob as active trackers, the tracking problem can be reduced to a well known assignment problem. Moreover, the tracking process exclusively uses moving objects, thus freeing the tracking process from any video resolution dependencies and making possible real-time tracking for UHD video sequences.

The rest of the chapter is organized as follow: the rest of section 2.1 further introduces moving object detection methods and tracking systems. The different properties of moving objects will be the subject of section 2.2. Then section 2.3 and 2.4 present our refinement system in details as well as the tracking system associated. In Section 2.5 we present the different experiments and discussions. Finally, section 2.6 concludes this chapter.

2.1.1 Foreground object detection

Early works use the average of gray or color of background pixels to define the background [39] but they could not handle multiple backgrounds. In 1999 the popular Gaussian Mixture Model (GMM) has been proposed by Stauffer and Grimson [16] to address some of the issues.

In 2005, a non-parametric algorithm for background subtraction also known as the codebook method has been introduced by Kim *et al.* [40], and in 2011 Xu [41] proposed a block-based with Oriented-Gradient Feature version of the codebook model. In [42] a fast object detection method based on a likelihood background model has been proposed. More recently, Liu *et al.* [43] introduced a generalized expectation maximization framework that uses bottom-up cues with top-down information for foreground object detection. Some surveys and evaluations of the background subtraction methods are available in [36, 37, 38].

Since those methods only focus on the detection, most of them however, are unable to deal with occlusions between targets when two moving objects are too close and fuse into a single object. In this chapter we chose to use, as main moving object detection, the most basic background subtraction approach which is the fastest but is also supposed to be the least accurate one, in order to show the different assets of our refinement method.

2.1.2 Multi-objects tracking systems

Through the years, powerful tracking systems have been developed such as Kalman Filters or Particle Filters [16, 44, 45]. Their complexity however is very high and they usually require a large number of samples to be efficient [46].

In [47] and [48], the assignment of the detected object positions to predict the position of all targets is done by the Hungarian algorithm, but once again the state estimation is performed by a Kalman filter-based algorithm. In all of those existing solutions, the approaches are based on pixel values and appearance making those methods very dependent of the resolution of the video.

The tracking method we decided to adopt is a pure object-based assignment algorithm. No other information than the moving objects is required and the blob-tracker assignment is based on their proximity. Although this method seems too simplistic it is also a great way to observe the efficiency of the refinement process since the tracking is only based on the accuracy of the refined moving objects.

2.2 Moving Objects

For tracking purposes, extracting moving objects seems to be evident and a necessity. For this reasons moving objects appear among the first researches of computer vision. Knowing where things moved in an image is a good asset in a tracking system, since the areas containing moving objects can be focused on and the rest can be avoided thus saving resources. A moving object is defined by four variables, either a couple of coordinates $pt_1(x_1, y_1)$ $pt_2(x_2, y_2)$ for the corners (top-left and bottom right corners traditionally) or the coordinate of the center $c(x_c, y_c, w, h)$ (or top-left corner) and the width and height of the object.

Aside from the simplicity of its representation, a moving object can locally be the only resource needed to track a target. Indeed, if there is only one moving object at proximity of a targets estimated position then the likelihood of this moving object to match the target is very high. Locally using only moving objects for tracking can then skip any unneeded comparison process and directly associate the blob and the tracker. However, moving objects suffer from two major drawbacks. First of all there is no distinction between background movements and movements from objects. A moving tree and a moving target will be detected without distinction and the size and shape of the blobs cannot be used to sort the noise from the targets. The second drawback is the fusion effect. When two or more moving objects are too close from each other, their bounding boxes are fused into a bigger blob. Because of this effect, occlusions between targets create less blobs than targets and the noise from moving background can also merge with real targets making blobs not only bigger than the targets but the fact that there are not centered on the targets anymore makes them also less accurate.

Because of all the drawbacks, tracking systems still require additional appearance information. Indeed this visual information is related to the size of the objects so an increase of the size of the object would slow down the tracking system. The purpose of this work is to increase the accuracy of the moving object results through an efficient noise removal and occlusion handling process.

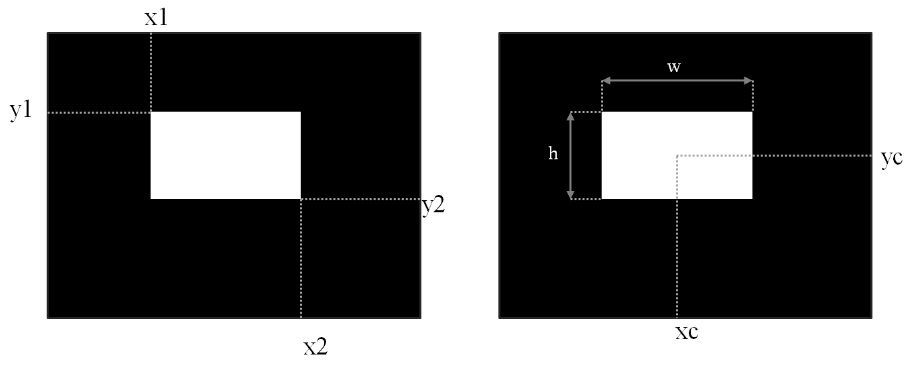


Figure 2.1: Example of a moving object. In both definitions of a moving object, 4 integers are enough to characterize it.

2.3 Object refinement

In the present chapter, we propose a refinement layer between the detection process and the tracking process. This layer has the function to enhance the results of the detection part by making a link with the tracking system. Indeed, since the positions of all targets are known from previous tracking, they can be used directly in the detection process to sort the interesting blobs, which represent one target or more, from the uninteresting ones (or noisy blobs) which are every other moving part. In the moving object detection, the background can bring a lot of noise. Indeed, moving trees for example are a high source of noisy moving objects and it is quite difficult to erase all that noise automatically. The other aim for the RTRM, in addition to the noise removal, is to deal with occlusions which can happen when two or more targets get close and the detection cannot separate them.

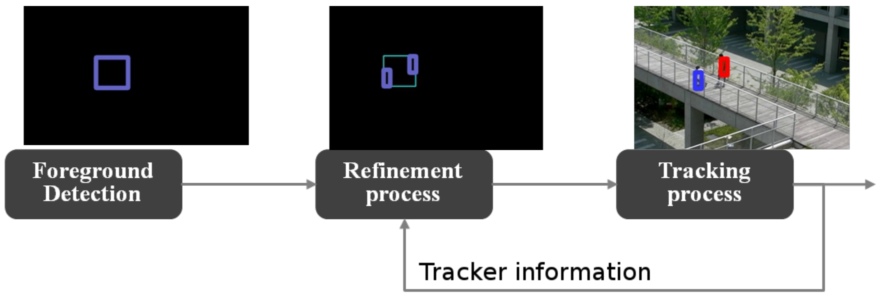


Figure 2.2: General system layout. The RTRM is an additional layer between the detection and the tracking systems thus making it compatible with various detection methods.

The RTRM algorithm is based on the use of previous iterations tracking results to have an estimation about the positions of all targets in the scene. But first and in order to remove and split blobs efficiently, targets must not be fragmented after the original object extraction. Since we want to remove only useless blobs, we do not want to remove blobs which represent a part of a target. By merging all blobs at a distance from each other less than ϵ_m, the number of fragments greatly reduced if not totally. The value of ϵ_m is a constant value which depends on the average size of the objects of interest.

2.3.1 Blob filter and Occlusion Detection

The first task of the object refinement is the filtering of the blobs which correspond to false positives. From a tracking point of view, false positives can be defined as blobs which do not correspond to any of the targets. Therefore, a blob which does not overlap any tracker will be considered as one and need to be removed from the list.

In the system at a frame t, there are N_t active trackers and N_b blobs from the original detection method. Knowing the number of blobs and trackers is not enough to say if the blobs are good or not. Indeed, it can happen that $N_t = N_b$ and have $N_t - 1$ false positive blobs and a unique blob containing the N_t targets. That is a reason why it is important to verify if each blob can correspond to an object of interest or not. Therefore the presence of trackers on the blobs is checked by counting their number C_b inside each blob according to

Algorithm 1. The estimated position $T_{estm}(x, y)$ for a tracker is determined by its position in the last frame and by its speed according to the previous iterations: $T_{estm}(x, y) = \begin{pmatrix} x + \dot{x} \\ y + \dot{y} \end{pmatrix}$. A tracker is on a blobs if $T_i \cap B_j \neq \emptyset$.

When the verification is effectuated for a pair blob-tracker four situations can occur. First, in the case where no tracker is on a blob then this blob is very probably a noise and it is erased. The second scenario is when only one tracker is on a blob. Since no other tracker shares the blob, the tracker can be linked to this blob. The third case appears when the trackers has already been linked to a previous blob. That situation occurs when a tracker is overlapping two or more blobs, therefore a choice is needed and if the current blob is closer to the estimated position of the tracker, then it is linked to the tracker and the previous link is destroyed.

At the end of this first selection, for trackers which are not on any blob we reach the last possible situation. There is still a chance that the target moved very quickly or that the detection has produced smaller blobs. To take that chance, an enlarged search is performed. The enlargement search consists of an artificial increase of the size of the tracker and a search of blobs which could meet this enlarged tracker.

If some trackers are not on any blob this could either mean the target stays still (this can happen depending on the way the original detection is done), or the target has left the scene. In this situation, a new blob is created as a clone of the tracker and the appearance countdown starts. If this countdown reaches 0 the tracker is considered as inactive and removed from the list.

The selection done, only blobs which are susceptible to match targets are kept, all the other blobs are removed. Two possible cases only are possible for each blob (see Algorithm 2): it covers only one tracker or it covers two or more trackers, it represents then the fusion of different targets. In the first situation, the blob is kept and is associated to the tracker on it; in the second situation the algorithm detects an occlusion and the blob needs to be separated into new blobs, one for each tracker on the blob.

Algorithm 1 Blob selection

```
1: **Input:** Trackers $\{T_i, i = 1, \dots, N_t\}$ , Blobs $\{B_j, j = 1, \dots, N_b\}$;
2: $1^{st}$ **Pass : Normal selection**
3: **for all** $T_i$ **do**
4:     **if** $T_i$ is on the image **then**
5:         **for all** $B_j$ **do**
6:             **if** $T_i \cap B_j \neq \emptyset$ **then**
7:                 **if** $T_i$ is already on a blob $B_k$ **then**
8:                     **if** $\Delta(T_i, B_k) < \Delta(T_i, B_j)$ **then**
9:                         Remove $T_i$ from the tracker list of $B_j$ and add it to $B_k$.
10:                    **else if** $T_i$ is not on a blob yet **then**
11:                        Add $T_i$ to the tracker list of $B_j$
12:                    **end if**
13:                **end if**
14:            **end if**
15:        **end for**
16:    **end if**
17: **end for**
18: $2^{nd}$ **Pass : Extend search**
19: **for all** $T_i$ **do**
20:    **if** $T_i$ is not on a blob **then**
21:        **for all** $B_j$ **do**
22:            Get $B_{min}$ as $\Delta(T_i, B_{min}) = min(\Delta(T_i, B_j))$ & $\Delta(T_i, B_{min}) < \epsilon$
23:            Increase search area $A$ for $T_i$
24:            **if** $(T_i \cup A) \cap B_{min} \neq \emptyset$ **then**
25:                Add $T_i$ to the tracker list of $B_{min}$
26:            **end if**
27:        **end for**
28:    **end if**
29: **end for**
```

2.3.2 Keep Blob

In the case where only one tracker has been affected to a blob, the system considers that the blob is locally the only logical match for the tracker. Therefore, the tracker can already be updated with the new characteristics of the blob. The most common situation that can happen is that the tracker's size and the detected blob have similar size. In this situation the tracker T_p is simply updated with the properties of the blob B_p (see Eqn. 2.1).

Algorithm 2 Refinement

```
1: Input: Trackers $\{T_i, i = 1, \ldots, N_t\}$ , Selected Blobs $\{M_l, l = 1, \ldots, N_m; N_m \leq N_t\}$;
2: for all $M_l$ do
3:     if $M_l \ni T_k, T_k \in T$ then
4:         Associate $T_k$ and $M_l$ ; Store $M_l$
5:     else if $M_l \ni T_{k \ldots k'}, T_{k \ldots k'} \in T$; $|T_{k \ldots k'}| = N_{tr}$ then
6:         Split $M_l$ into $N_{tr}$ new blobs
7:     end if
8: end for
```

$$T_p[x, y, w, h] \quad = \quad B_p[x, y, w, h] \tag{2.1}$$

If the size is too different however, this means that the target's shape has changed a lot from the previous frame, or more likely that some part of the background has been fused with a target. In this particular case, each property (x, y, w, h) of the blob is compared to the tracker's respective property and if the distance d between them is such as $d > \gamma$, γ being the respective distance threshold, then for x (respectively for y, w and h):

$$T_p(x) \quad = \quad \begin{cases} B_p(x) & \text{if } d < \gamma \\ T_p(x) & \text{otherwise} \end{cases} \tag{2.2}$$

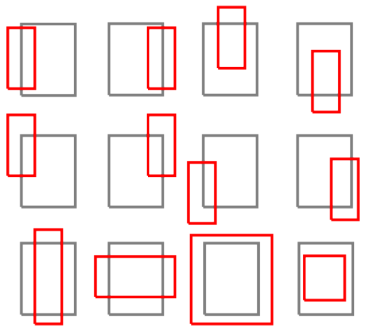
2.3.3 Occlusion handling: splitting of the fused blobs

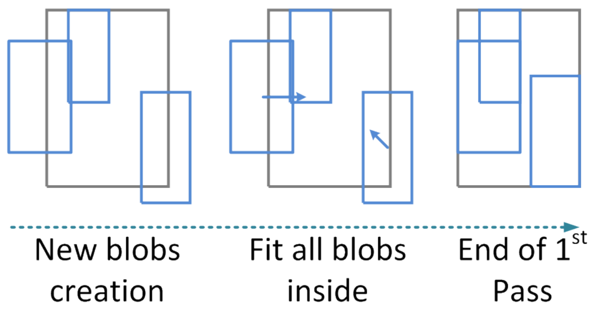
Occlusions of blobs are the main issues for moving object detection systems because the detection of close objects or overlapping objects will result in a unique blob containing multiple targets. The objective of the splitting process is to create the same number of blobs as the number of target inside this unique blob.

Fused blobs are the worst situation since we have to split the fused blob into new blobs to match as much as possible the reality of the situation but with very limited data. The splitting process is divided into two parts: the first pass creates new blobs as copies of the trackers inside the blob M and position them inside it. The second pass focuses on the spatial optimization of the original blob, and moves the new blobs previously created to fit M as much as possible. The splitting process is detailed in Algorithm 3.

First pass

At this stage the original blob M has now to be divided into N_{tr} new blobs, one for each tracker on the blob. The estimated positions of all trackers have been computed previously during the blob selection step. Those estimated positions are the start for the creation of the new blobs. We can create new blobs, exact copies of the trackers in size but with their estimated position for the current frame.


(a) Possible positions of a new blob created (in red) by the first pass of the division process.



(b) First a new blob will be created as a copy of the trackers. Then, each blob which is stepping outside from the object being divided are moved inside it.

Figure 2.3: First pass of the occlusion handling process. New blobs (in red and blue) are created from the trackers which are on the original blob (in grey).

Then, all those new blobs have to fit into the shape of the original blob M to optimize its occupancy. For each border D_b of M, if the distance $\Delta_{i,b}$ from B_i's border to the corresponding border D_b is positive, the blob goes over the border of M and need to be shifted to fit inside M. The shift length is simply equal to the distance to the border $\Delta_{i,b}$, only if the size on the axis of the border of the new blob B is not greater than the size of M. In that particular scenario, the new blob's size is reduced to the size of M and then shifted on the corresponding axis. In the case where the distance is negative, it means the blob is already inside M and

Algorithm 3 Splitting Process

```
1: Input: Trackers $\{T_i, i = 1, \dots, N_{tr}\}$ , Original blob $M$, Borders of $M$ $D_{bottom}, D_{top}, D_{left}, D_{right}$;
2: $1^{st}$ Pass : New blobs creation and prepositioning
3: for all $T_i$ do
4:     Create new blob $B_i$ as copy of $T_i$
5: end for
6: for all $B_j$ do
7:     for all borders $D_{bottom}, D_{top}, D_{left}, D_{right}$ do
8:         Calculate distance $\Delta_{i,b}$ to $D$
9:         if $\Delta_{i,b} \geq 0$ then
10:            if $w_{B_i} > w_M$ ($h_{B_i} > h_M$ respectively for the vertical axis) then
11:                Shift horizontally (resp. vertically) and resize $B_i$ to fit in $M$.
12:                Link $B_i$ to $D$.
13:            else
14:                Shift horizontally (resp. vertically) $B_i$ to fit the border $D$ of $M$.
15:                Link $B_i$ to $D$.
16:            end if
17:        end if
18:    end for
19: end for
20: $2^{nd}$ Pass : Re-positioning
21: if no blob is linked to any border $D$ then
22:     Get $B_k \in B$ such as $\Delta^c_{k,D_{bottom}} = \min(\Delta^c_{i,D_{bottom}})$
23:     Shift and link $B_k$ to $D_{bottom}$ if $\Delta_{k,D_{bottom}} < \varepsilon_v$
24:     Get $B_{k'} \in B \backslash B_k$ such as $\Delta^c_{k',D_{top}} = \min(\Delta^c_{i,D_{top}})$
25:     Shift and link $B_{k'}$ to $D_{top}$ if $\Delta_{k',D_{top}} < \varepsilon_v$
26:     Get $B_{k''} \in B$ such as $\Delta^c_{k'',D_{left}} = \min(\Delta^c_{i,D_{left}})$
27:     Shift and link $B_{k''}$ to $D_{left}$ if $\Delta_{k'',D_{left}} < \varepsilon_h$
28:     Get $B_{k'''} \in B \backslash B_{k''}$ such as $\Delta^c_{k''',D_{right}} = \min(\Delta^c_{i,D_{right}})$
29:     Shift and link $B_{k'''}$ to $D_{right}$ if $\Delta_{k''',D_{right}} < \varepsilon_h$
30: else                                                    ▷ Some blobs are already linked
31:     if $D_{bottom}$ is not linked then
32:         if $D_{top}$ has 2+ links then
33:             Get $B_k \in B$ such as $\Delta_{k,D_{bottom}} = \min(\Delta_{i,D_{bottom}})$
34:             Shift and link $B_k$ to $D_{bottom}$ if $\Delta_{k,D_{bottom}} < \varepsilon_v$
35:         else if $D_{top}$ has 1 link with a blob $B_k$ then
36:             Get $B_{k'} \in B \backslash B_k$ such as $\Delta^c_{k',D_{bottom}} = \min(\Delta^c_{i,D_{bottom}})$
37:             Shift and link $B_{k'}$ to $D_{bottom}$ if $\Delta_{k',D_{bottom}} < \varepsilon_v$
38:         else
39:             Get $B_k \in B$ such as $\Delta^c_{k,D_{bottom}} = \min(\Delta^c_{i,D_{bottom}})$
40:             Shift and link $B_k$ to $D_{bottom}$ if $\Delta_{k,D_{bottom}} < \varepsilon_v$
41:         end if
42:     end if
43:     if $D_{top}$ is not linked then
44:         if $D_{bottom}$ has 2+ links then
45:             Get $B_k \in B$ such as $\Delta_{k,D_{top}} = \min(\Delta_{i,D_{top}})$
46:             Shift and link $B_k$ to $D_{top}$ if $\Delta_{k,D_{bottom}} < \varepsilon_v$
47:         else$D_{bottom}$ has 1 link with a blob $B_k$
48:                              ▷ At this point the is always at least a link for $D_{bottom}$
49:             Get $B_{k'} \in B \backslash B_k$ such as $\Delta^c_{k',D_{top}} = \min(\Delta^c_{i,D_{top}})$
50:             Shift and link $B_{k'}$ to $D_{top}$ if $\Delta_{k',D_{bottom}} < \varepsilon_v$
51:         end if
52:     end if
53:                                       ▷ $D_{left}$ and $D_{right}$ have a similar approach
54: end if
```

does not need to be moved yet.

This first pass done, there are N_{tr} new blobs inside the original blob M and all of them are also occupying the insides of M. For each blob B_i which shares a common border with M, a link l_i^b is created. Those links will be useful in the second step of the process to set priorities in order to re-position the blobs.

Second pass

The purpose of the second pass is to optimize the space occupancy of original blob M with the new blobs recently created from estimated positions of the trackers. This optimization is necessary for the two main situations which can occur during an occlusion: the meeting and the separation. During the meeting, the moving object resulting is getting smaller and smaller and can, in a case of total occlusion, has the shape of a single target but still containing more than one target. The separation is the opposite effect, the blob is getting bigger and bigger until the total separation of the targets. The restriction of this method is that it is designed for relatively short occlusion and for a small number of object fused. Therefore, it is not made for tracking of crowds.

For the meeting scenario, the first pass is perfectly able to deal with it since the new blobs are forced to fit into the fused blob. However, the separation scenario requires a special attention in order to avoid the degeneracy of the system. Indeed, the blobs should fit as much as possible the area given by the original blob, thus they have to be repositioned if necessary. This repositioning algorithm works by borders' couple (bottom-top, left right) and it is based on the distance $\Delta_{i,b}^c$ from the center of the blobs to the borders $b \in D_{bottom}, D_{top}, D_{left}, D_{right}$ and $\Delta_{i,b}$ the distance border to border.

The algorithm is quite simple: if there are no link l_i^b then the blobs can be chosen freely according to the distance of their center $\Delta_{i,b}^c$ to the respective borders. The closest blob is shifted to fit the current border by $\Delta_{i,b}$ if $\Delta_{i,b}^c < \varepsilon_(v,h)$, $\varepsilon_{v,h}$ being a moving threshold depending on the average size of a target. After each shift, a link is created so the blob which just got moved does not move anymore (at least on the same axis).

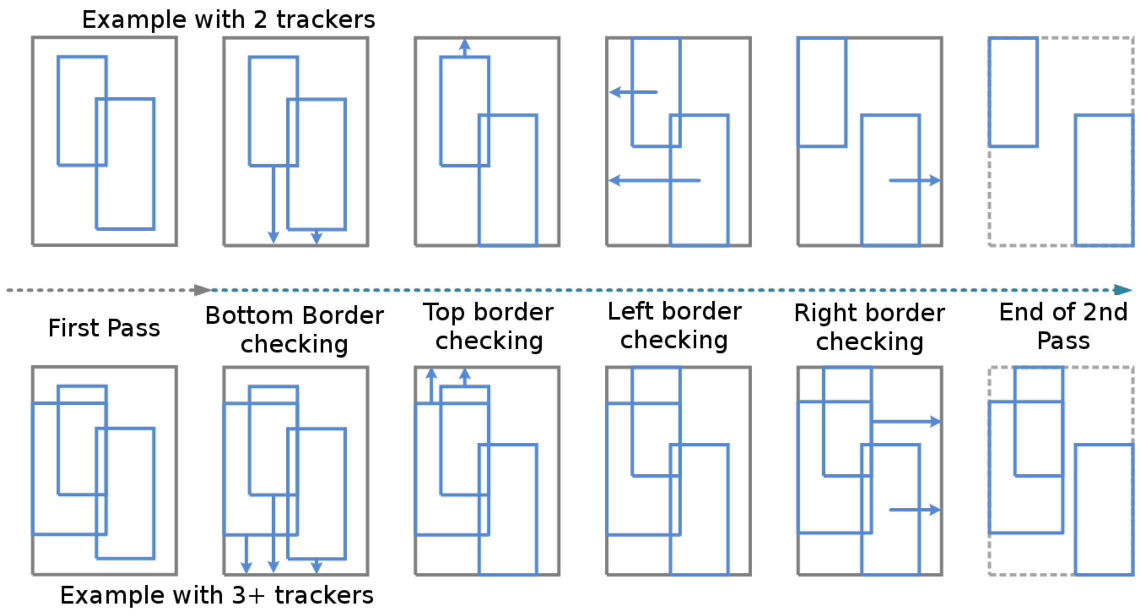


Figure 2.4: Example of the second pass of the occlusion handling process with two and three trackers inside the original blob before division.

In the case where some links have already been made, then the borders, starting from the bottom, are checked for any link as well as the opposite border. The objective is to move only the closest blob to the border at the one condition that this blob B_k is not necessary for the opposite border. Indeed, it would mean that B_k was already the best solution for the opposite border so trying to move it on the opposite border would be contradictory. An opposite border with two or more blobs linked to it can *give* one of its blobs to the border currently processed since it will still have a link. In the case where there is only one link, the linked blob B_k cannot move, so the closest blob has to be chosen from the rest of the remaining blobs.

The original blob M is finally removed at the end of the occlusion handling process since it has been replaced by the new blobs B_i. Figure 2.4 shows two examples of the blobs' states during the splitting process with respectively 2 and 3 trackers fused in a single blob.

2.4 Blob matching

In order to show the efficiency and interests of the RTRM, the tracking system used in this work is a classic assignment problem where blobs and trackers have to be associated by pair according to a matching score. There are in fact two ways to associate blobs and trackers. The first one is directly through the refinement process when a blob is linked to a single tracker. Since the blob is locally the only possible option for the tracker, this last one and the blob are assigned together. The second option for assignment occurs when fused blobs have been divided into new blobs. In this configuration, we use an heuristic assignment algorithm [49]: the pair with the highest score is associated first. Since it is the purpose of this work to have as many blobs as active trackers, the number of elements of all sets is equal ($N_t = N_b = N$; N_t trackers, N_b blobs), simplifying greatly the assignment process. In our case we used as matching scores the invert of the Euclidean distance between the blobs and the trackers as follow:

$$s_p(t,b) = \frac{1}{\delta_p(t,b)} \tag{2.3}$$

with

$$\delta_p(t,b) = \sqrt{(x_t - x_b)^2 + (y_t - y_b)^2} \tag{2.4}$$

The fact that there are a small number of candidates for all trackers, the processing time is far less than any other tracking system. Indeed, unlike complex methods like particle filters which require a large number of samples, usually hundreds or thousands, using a very low number of objects as possible candidates makes the tracking process almost instantaneous. The heuristic algorithm for the blob-tracker pair matching performs in $O(N^2 log(N))$ with $N << 100$, making it way faster than any probabilistic methods and even faster than the optimized Hungarian algorithm which performs in $O(N^3)$.

Once the optimized pairs have been computed, the trackers are updated with the properties of the blob they have been associated to. Thus, their coordinates and size are set to the one of the blob and the trackers speed is updated as well for future iterations following equation 2.1

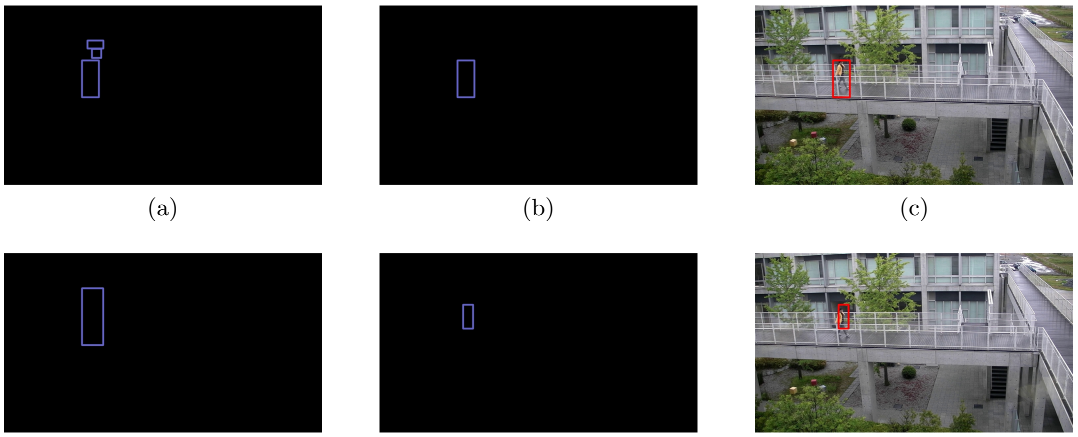
2.5 Experimental Results

The proposed refinement technique has been tested on a custom dataset in order to emphasize its different assets. The parameters for the experiments are as follow: $\varepsilon_v = 1\,\varepsilon_h = 0.2$, the elements' size for the morphological operations of the background subtraction are {17, 13, 11}, {29, 17, 15}, {23, 15, 13}, {150, 100, 75} for respectively the blur, open, close and the merging distance, and the distance threshold $\gamma = 0.3$ The computer used is a Quadcore i7@2.83GHz with 2 Go of RAM.

2.5.1 Noise Removal

The first experiment is a Full HD sequence (1920×1080 pixel) from our database *goto_lab_01 (single)* in which a single target moves in front of moving trees (Figure 2.5). Two detection methods, the static background subtraction (Figure 2.5a) and the codebook method from [41] (Figure 2.5d) have been compared with and without the use of the proposed refinement. We can notice the presence of background noise due to the moving trees. It usually is a problem as we can see in Figure 2.5d where a big blob is resulting from the fusion of the target and the noise. In the background subtraction case, there are many noise blobs which are independent. Those are simply removed by the RTRM as there are no tracker on them. The effect of the moving background is dealt differently by the two detectors, creation of fragments for the background subtraction (Figure 2.5c) and fusion of target and noise for the codebook (Figure 2.5f). Even though the detection methods provide different results, we can observe that the results are very similar in tracking (Figures 2.5c and 2.5f), speed (Figures 2.6c and 2.6d) or number of objects after refinement (Figures 2.6a and 2.6b).

The comparison of the two graphs show that the number of objects after original detection process is not the same through the entire sequence but the proposed RTRM succeeds in keeping the number of objects the same as the number of active trackers proving it is efficient with different detection methods.

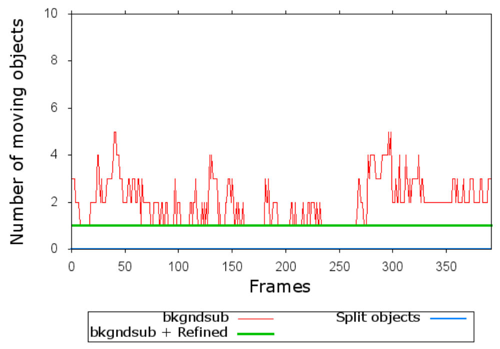

(a) (b) (c)

(d) Original detection (e) After RTRM (f) Tracking result

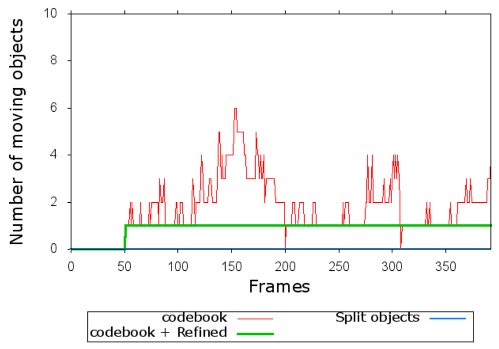
Figure 2.5: Single person tracking with moving background, visual comparison of two different detection algorithms, by a static background subtraction (top row) and by the codebook method (bottom row).

2.5.2 Occlusions

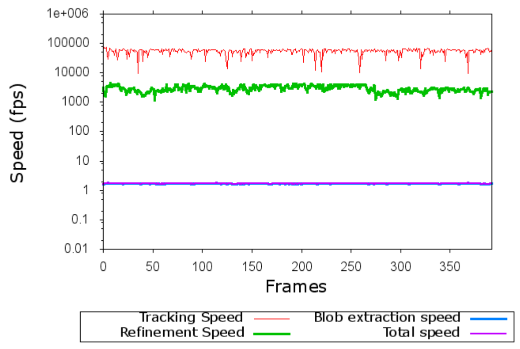
The second experiment is another sequence from our database *goto_lab_02(crossing2)* (Figure 2.7). In this 1280×720 video two people are crossing producing a total occlusion at one point of the sequence. During a total occlusion, there is a lack of information since one target is hidden by the other one. Therefore, the separation can be delicate. When the two targets are getting far from each other, the fused blob's size is increasing. The fact that the occluding handling process forces the blobs to spread as much as possible in the original blob creates the opportunity, when the targets become really separated, to position the trackers in a better place so they can track back once the targets are fully separated. The occlusion handling process splits blobs containing two or more blobs (Figure 2.7b) into new blobs (Figure 2.7e) then the tracking continues based on the speed and position of the objects (Figures 2.7g to 2.7i).



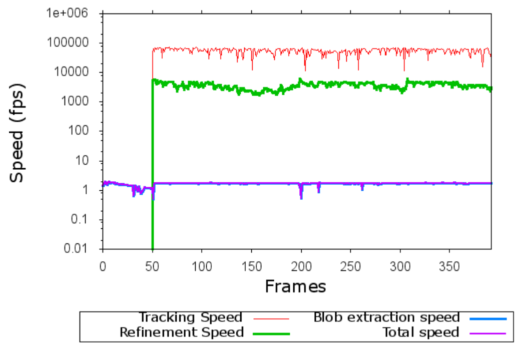
(a) Object count : SBGS



(b) Object count : CB



(c) Speed : SBGS

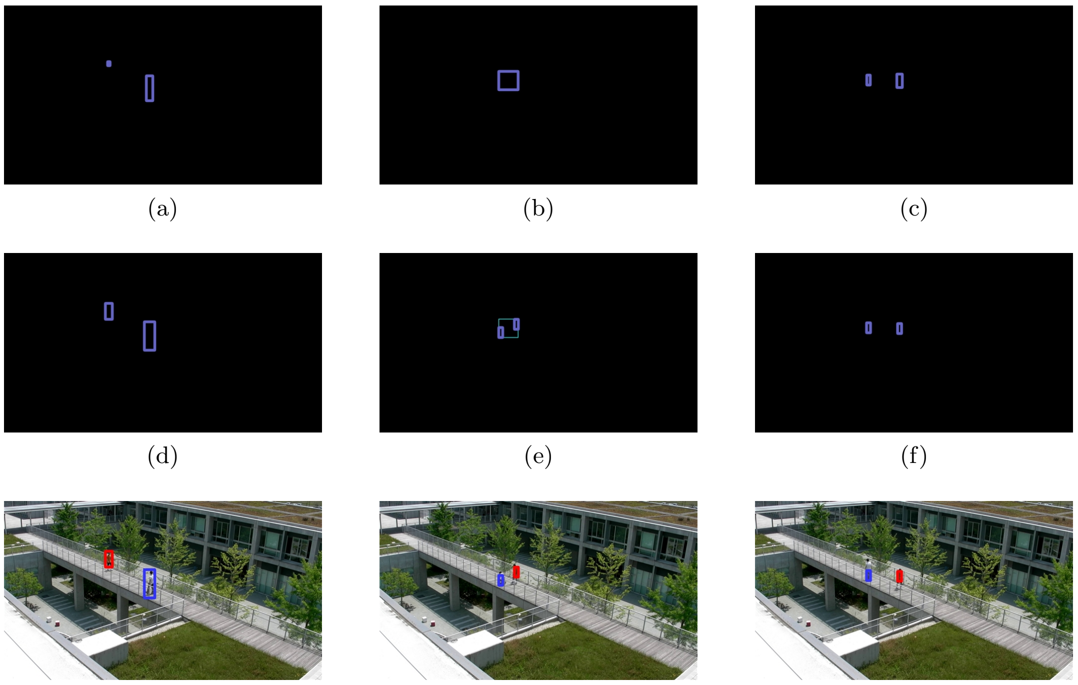


(d) Speed : CB

Figure 2.6: Single person tracking with moving background, comparison of the speed and object count for two different detection algorithms, a static background subtraction and by the codebook method.

2.5.3 Resolution-free system example: UHD sequence

The last experiment presents the advantages of using simple objects instead of pixel-based models to define the objects of interest. We tested our method with Nebuta UHD sequence (frames 13200-13643) from JCT-VC [19]. As presented before in this chapter, working with objects frees us from resolution dependency once the moving object detection is done. There is absolutely no reference to the size of the video during the refinement or the tracking process. Therefore, the refinement and tracking process speed results of the same sequence at different resolutions show an average speed approximately the same. Only the detection process is different because it is necessary at that point to scan all the pixel thus lowering the speed for very high resolutions. If the global system cannot really be called real-time because of the slowness of the detection, these results prove that this method is nonetheless an online



(a) (b) (c)

(d) (e) (f)

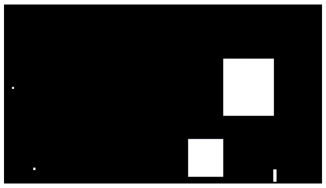
(g) Frame 2168 (h) Frame 2377 (i) Frame 2440

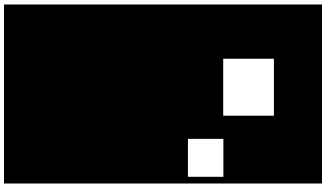
Figure 2.7: Occlusion Handling Process. From top to bottom: raw moving object detection (2.7a, cr4, cr7), after proposed RTRM (2.7d, 2.7e, 2.7f), tracking result (2.7g, 2.7h, 2.7i).

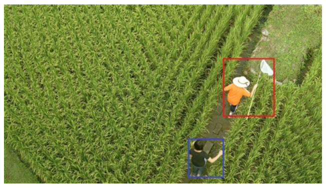
solution which has a lot of potential for more complex UHD tracking systems.

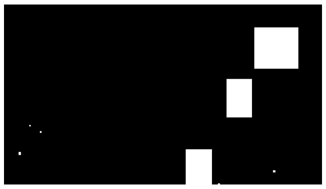
Table 2.1: Tracking performances of the difference sequences tested with the proposed method (w/ proposal) and without (w/o). Best results in bold.

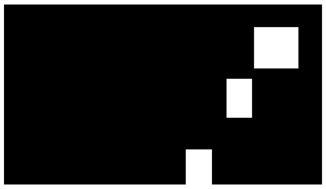
Sequence	RTRM	Occl.	MODP	MOTP	MODA	MOTA	IDS	miss	FP
goto_lab_01	w/o	-	0.35	0.35	-0.10	-0.10	0	0	368
	w/ proposal	0	**0.41**	**0.41**	**0.86**	**0.86**	0	0	4
goto_lab_02	w/o	-	0.08	0.08	-13.9	-16.8	851	0	1655
	w/ proposal	213	**0.35**	**0.35**	**0.94**	**0.91**	60	8	0
field 8K	w/o	-	0.10	0.10	-41.2	-42.9	53	0	2634
	w/ proposal	53	**0.50**	**0.50**	**0.98**	**0.92**	19	10	0
field 4K	w/o	-	0.08	0.08	-32	-33.9	72	0	2080
	w/ proposal	153	**0.38**	**0.38**	**0.98**	**0.93**	13	10	0
field 1080p	w/o	-	0.13	0.13	-13.9	-15.4	43	0	889
	w/ proposal	10	**0.22**	**0.22**	**0.98**	**0.94**	8	10	0

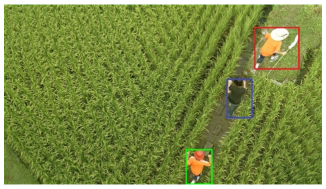

(a)

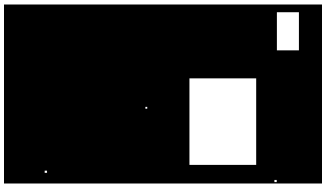

(b)

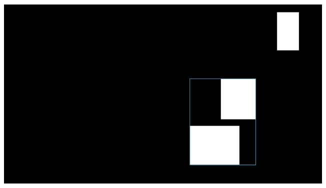

(c)

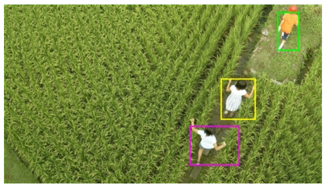

(d)

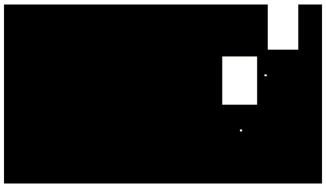

(e)

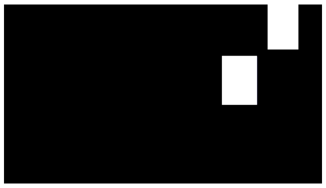

(f)


(g)


(h)


(i)


(j) Org. blobs


(k) Refn.

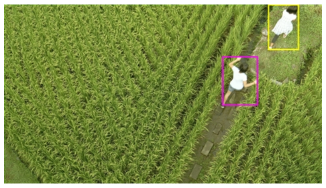
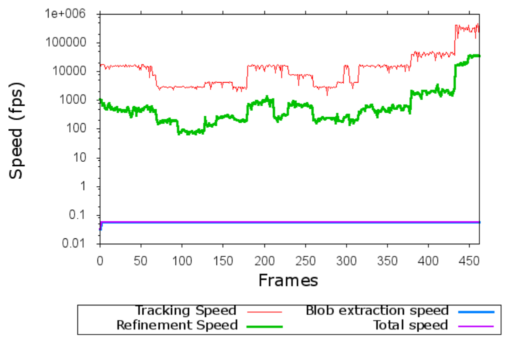

(l) Track.

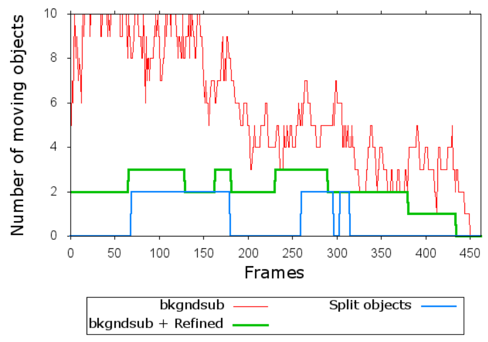
Figure 2.8: Example of the RTRM for an 8K UHD sequence (Nebuta Festival, NHK). From top to bottom: frames 0, 80, 270 and 350. On the left the blobs from the original background subtraction detection method (2.8a, 2.8d, 2.8g, 2.8j), in the middle the blobs after RTRM (2.8b, 2.8e, 2.8h, 2.8k), on the right the tracking result associated (2.8c, 2.8f, 2.8i, 2.8l). We can observe the efficiency of the RTRM which removes the noise left and also splits the fused blob (2.8g) into two new blobs (2.8h).

2.5.4 Tracking performances

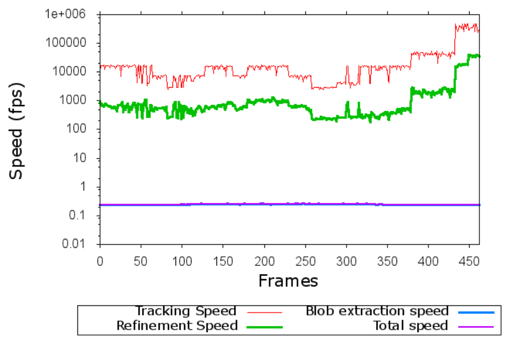
In this part we will show the tracking performances of the RTRM process that we introduced in this chapter. We used the CLEAR metrics [50] to evaluate the performances of this combination of algorithms and tested the sequences with and without the RTRM to observe its efficiency. The **N-MODP** and **N-MOTP** are respectively the *Normalized Moving Object Detection Precision* and *Normalized Moving Object Tracking Precision* while the **N-MODA**



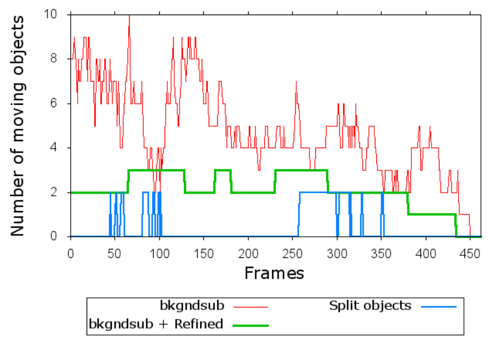
(a)



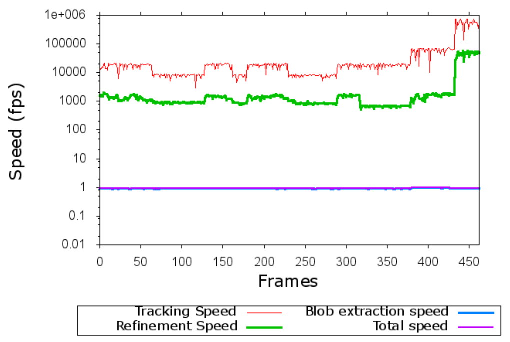
(b)



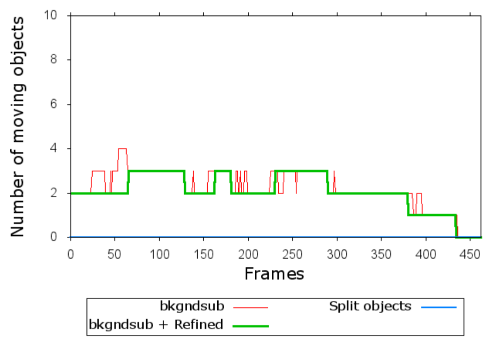
(c)



(d)



(e)



(f)

Figure 2.9: Comparison of speed (top row) and number of objects (bottom row) for three different resolutions of the same sequence (Nebuta Festival, NHK) (from left to right, respectively 8k, 4k and 1080p). Working at object level and not at pixel level after the detection process shows that even though the detection speed (in purple) is different, the RTRM speed (in green) and the tracking speed (in red) are about the same. The difference in number of objects is mainly due to a lot of small noisy blobs from background parts which are not detected in the smallest resolution.

and the **N-MOTA** stand for *Normalized Moving Object Detection Accuracy* and *Normalized Moving Object Tracking Accuracy*. In the following tables, they will simply appear as MODA,

Table 2.2: Tracking performances of two different detection methods with the proposed method and without (w/o). Best results in bold.

Sequence	RTRM	Method	Occl.	MODP	MOTP	MODA	MOTA	IDS	miss	FP
goto_lab_01	w/o	BSG	-	0.35	0.35	-0.10	-0.10	0	0	368
		CB [41]	-	0.60	0.60	-0.08	-0.08	0	0	307
	with proposal	BSG	0	**0.41**	**0.41**	0.86	0.86	0	0	4
		CB [41]	0	0.40	0.40	**0.91**	**0.91**	0	0	2
goto_lab_02	w/o	BSG	-	0.08	0.08	-13.9	-16.8	851	0	1655
		CB [41]	-	0.14	0.14	-15.4	-15.5	682	0	1532
	with proposal	BSG	213	0.35	0.35	**0.94**	**0.91**	60	8	0
		CB [41]	197	**0.43**	**0.43**	0.88	0.87	20	13	0
field 1080p	w/o	BSG	-	0.13	0.13	-13.9	-15.4	43	0	889
		CB [41]	-	0.22	0.22	-13.4	-13.4	67	0	833
	with proposal	BSG	10	0.22	0.22	**0.98**	**0.94**	8	10	0
		CB [41]	6	**0.62**	**0.62**	0.95	0.93	6	10	0

Table 2.3: Average speed performances (in fps). Best results in bold.

Sequence	Size	Occl.	Detect. (fps)	Refin.(fps)	Track.(fps)	Total(fps)
goto_lab_01(single)	1920 × 1080	0	1.72	**2925**	**54480**	1.72
goto_lab_02(crossing2)	1280 × 720	213	3.38	**2041**	**23405**	3.38
field 8K	7680 × 4320	53	0.057	**2223**	**35594**	0.057
field 4K	3840 × 2160	153	0.25	**2319**	**35124**	0.25
field 1080p	1920 × 1080	10	1.90	**2485**	**35974**	1.90

MODP, MOTA, MOTP. The precision is calculated in function of the common area between the mapped object and the ground truth object while the accuracy only takes into account the number of miss detection (*miss*), the number of *false positive* (*FP*) and also the number of ID switches (*IDS*) for the tracking. The number of occlusions (*Occl.*) in the video is also mentioned. Finally, the use or not of the RTRM algorithm is mentioned to compare its direct effectiveness. Precisions and accuracies are rates (for reference, the tracking accuracy is perfect if $MOTA = 1$) and best results are presented in bold.

The first thing we can observe in Table 2.1 is the impossibility to use directly the moving objects as an only feature for tracking as shown by the very poor results. The only sequence where the system could eventually work is *goto_lab_01* which contains only one target and few moving objects detected thus increasing the chances for a correct tracking.

We can continue in that even though the original moving object is really simple and fails

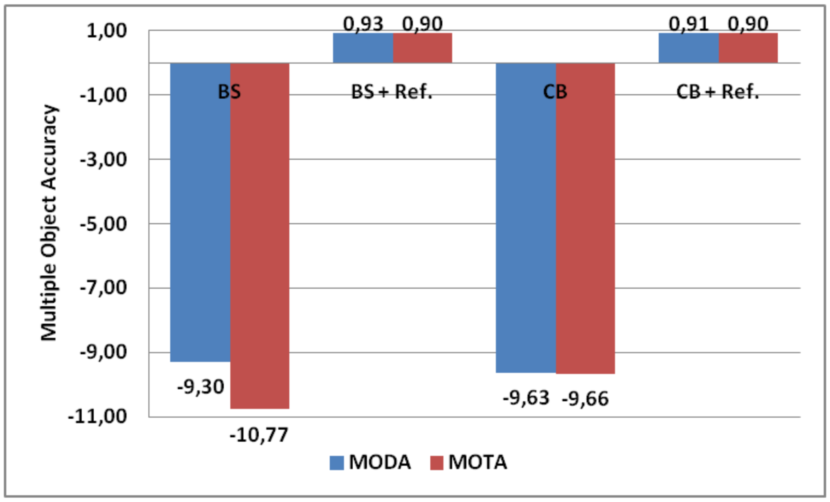


Figure 2.10: Comparison of the detection and tracking accuracy average values for each method with and without the proposal.

by itself, after our RTRM method, the accuracy for all sequences is very high. Not only this proves the efficiency of the removal effect of the RTRM but also that tracking with moving objects only is feasible. Without any background noise, only object representing targets are left so the association ground-truth objects to detected objects is simplified. The MODP and MOTP are identical for a simple reason: the tracking is only based on moving objects therefore there is no distinction between the two object thus making the precision calculation the same. If the precision results are very low, it is mainly because of the low precision of the background subtraction method which tends to create smaller objects than the ground truth as we can see in Figure 2.12.

The Table 2.2 presents the comparison of two different moving object detection methods with and without the RTRM layer in order to show the effectiveness of the layer itself. The two methods compared are the background subtraction and the codebook from [41]. To compare the two methods we limited the tests to the HD sequences since the codebook code did not support higher resolutions. Figure 2.10 shows the average accuracy score with and without the use of the RTRM layer. Figure 2.11 presents the average values for the precision of the SBGS and CB with and without the RTRM. We can observe that the best results are all achieved by the use of the RTRM. The accuracy values for both methods are high and quite similar in the case the RTRM is used. It represents an improvement of 10.23 to 11.67

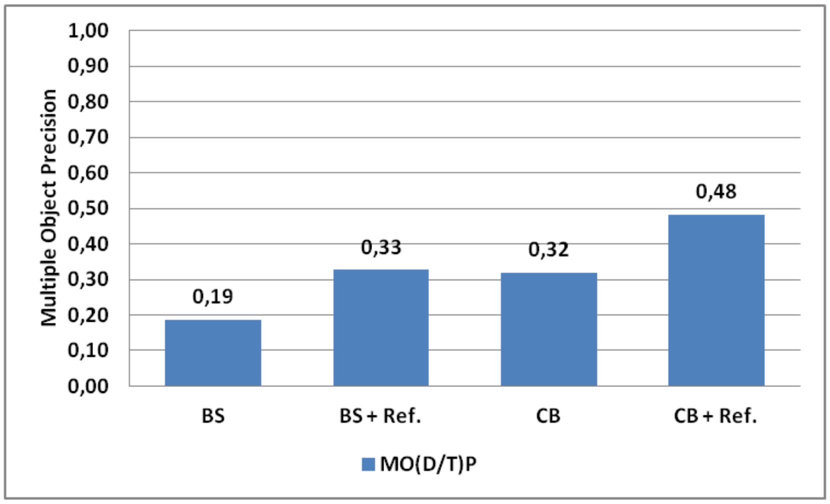


Figure 2.11: Comparison of the precision average values for each method with and without the proposal.

pts for the MODA The codebook is more precise than the background subtraction method.

The Table 2.3 shows the average speed performances of the different parts. The first remarkable thing is that the speeds for the RTRM or the tracking are not only very high but also very similar in every video. The tracking speed is only function of the number of targets as there is the same number of blobs to associate as targets. In the goal to reach real time tracking, a big number of targets would not have a significant impact on the tracking speed. The RTRM speed depends on the number of blobs detected, the number of trackers and the number of occlusions. Having an important number of blobs after the moving object detection is not a problem. For most of them they are noisy blobs which are removed directly during the first checking pass. However, having more targets implies higher chances to have occlusions and thus more fused blobs to split resulting in a drop of the speed depending on the number of objects in the blob to split.

The calculation of the speed has been made so it takes into account the number of objects in each frame thus removing extravagant speeds when there is no object to track in the video. Those very high speeds are reachable because of the use of moving objects only. Indeed, with less than ten objects, instead of hundreds, the tracking is almost instantaneous and the

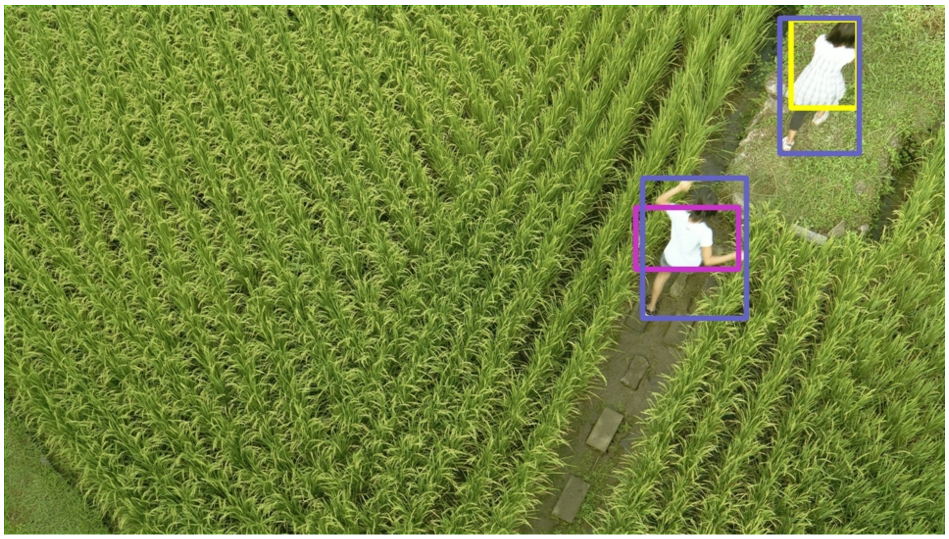


Figure 2.12: The difference of size between the GT in blue and the detected blob (in yellow and purple) leads to a low precision while the accuracy is still very good. This is explained by an inaccurate detection system.

RTRM speed is not bad either. With an average higher than *2000 fps* the RTRM proves its capacity to be adapted to any system without much impact on the speed of the system it is integrated in. Last but not least we can observe that the total speed is related to the detection process which is the bottleneck of this system even though we chose a very simple and quick detection method.

2.6 Conclusion

This chapter has shown that moving objects results can be improved to the point where they are sufficient for an efficient and fast tracking not only in simple situations but also in moderately complex ones like total occlusions and crossings.

Since the raw blobs from state-of-the-art detecting methods are not accurate enough, a refinement process is necessary to sort blobs. For that purpose, we used previous tracking results to bring additional information on the situation. At each frame, the trackers provide the number of target and their shape, position and speed. Knowing this is enough for the RTRM to determine the future of each raw blob. In the case where there is no tracker on them, they are removed, if there is only one tracker then the blob is kept and associated to that tracker. In the last case where multiple trackers are on the same blob, this one is divided into the number of trackers on it and the position of the new blobs is ruled to fit a probable situation depending on their position, speed and size. The tracking system associated is a direct association between blob candidates and trackers depending on the distance score between them. This association is only possible because the RTRM creates at all time as many blobs as there are active trackers on the scene.

We showed from the results that even with the simplest and not really accurate detection and tracking systems, our method was very efficient. The average accuracy goes from -10 to 0.90 and the precision is also increased with a highest value of 0.48 for the codebook using the RTRM compared to 0.32 if we don't use it. The results finally show that the simplicity of the objects allows doing a super fast tracking ($>$ 37000 fps) even in UHD sequences at the price of a refinement process which speed is also much higher than the detection process itself.

We also have seen that the foreground detection was the bottleneck of such system. Indeed, with 0.0057 fps in average for the fastest foreground detection method, it is unthinkable to be applied in any real application. A new and much faster foreground detection method must be developed to process UHD videos.

Chapter 3

Adaptive Block-Propagative Background Subtraction System for UHD Videos [††]

3.1 Introduction

We have seen in the previous chapter that foreground detection was a bottleneck for UHD processing. Without a method fast enough, there cannot be any real application using UHD videos.

There are numerous foreground object detection methods, the most famous and/or recent have been compared in the very recent literature [51, 52]. The panel of methods compared is exhaustive: basic methods based on mean and variance over time [53, 54], statistical methods based on a Gaussian distribution [55], statistical methods using a mixture of Gaussians [16, 56, 57, 58], statistical methods using appearance-based features such as color and textures [59], non-parametric methods [34] or methods based on eigen-values and eigen-vectors [60].

There have been attempts to use region blocks to detect foreground such as in [61] and [62]

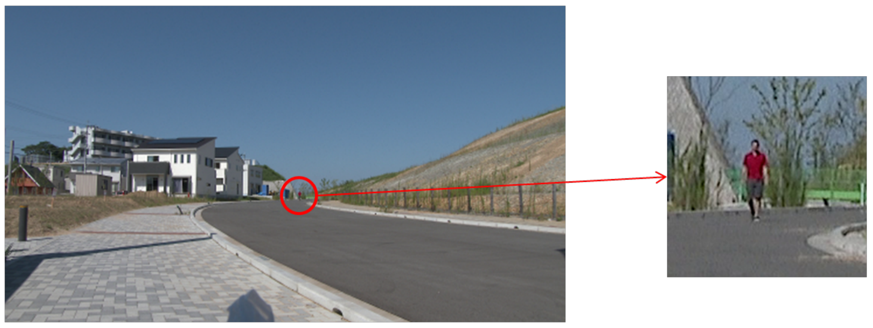
[††]This chapter is adapted from the work published in [J2]

but only in [7] we can see a use of blocks to process a smaller part of the image. They process some blocks at random and increase the number of local blocks around the areas in which foreground has been detected. Those methods have all in common that they all focused on improving the quality of the detection without really taking into account the processing time it would require. For that reason their complexity has increased more and more using more and more memory. This kind of approach was possible because the resolution of the input videos tested was very small so the impact of such complex methods was not very visible. Since the objective is to achieve the detection of foreground objects in UHD videos, such matters have a huge importance.

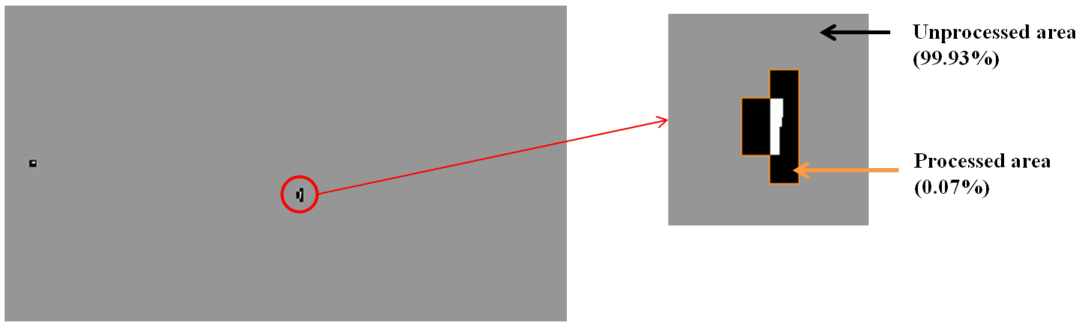
We propose in this chapter an extension of the Block-Propagative Background Subtraction (BPBGS) [63] which is a block per block detection with a fixed size of the blocks. The novelty of the Adaptive Block-Propagative Background Subtraction (ABPBGS) is its ability to detect objects according to their shape and size by dynamically updating the size of the blocks depending on the size of the detected objects. The objective of this method is to reduce the processed area by focusing on the parts where objects have been detected in the previous frame. Starting from a part of an object, the detection will propagate to adjacent blocks as long as a part of the object is detected on the corresponding borders. This way a great part of the computing resources can be saved by avoiding processing uninteresting areas and the detection will follow the shape of the object. This chapter is organized as follow: Section 3.2 will describe the method itself. We will compare our ABPBGS to the state-of-the art methods in section 3.3. Section 3.4.4 presents a direct application of the ABPBGS, section 3.4 and section 3.5 close this chapter.

3.2 Adaptive Block-Propagative Background Subtraction

In outdoor video surveillance scenes, the objects of interest are quite often very small. Indeed, the cameras are set in a way so they can capture a great field of view. With UHD, the field of view can be increased a lot and as a consequence, the objects of interest can appear even smaller compared to the total size of the image. We have seen in the introduction chapter that the down-scale of the image would only result in the loss of the objects, but the process of the whole frame would be a waste of resources especially considering the ratio between the size of the objects and the size of the image. The Adaptive Block Propagative Background Subtraction is our answer to this matter. Figure 3.1 presents a perfect example of this type of situation where the object of interest is much small than the total size of the image. Figure 3.2 presents the flowchart of the ABPBGS.



(a) Original image



(b) Detection result

Figure 3.1: Example of the usefulness of the method. Most of the frame is unprocessed saving more than 99.9% of the full area.

3.2.1 Propagation and Dynamic Block Size

The purpose of the proposed ABPBGS is to reduce the processed area to the strict minimum. Ideally the detection of each foreground object should be restricted to their contour but it is hardly possible without prior knowledge of the current state. If we consider that at a time *t-1* foreground objects have been detected as a consequence their position is known. If we also consider that the movement of the objects of interest between two consecutive frames is limited, it is very likely that at time t a part of the object can be detected at the position where it was detected at time *t-1*. Based on this idea and instead of scanning the whole frame, the proposed ABPBGS processes object after object, block after block. It starts with the detection of a single block: the seed block. This seed block will set the starting position for the rest of the detection for the current object. Then the detection will spread to neighbouring blocks as long as a part of the current object is detected in the current block.

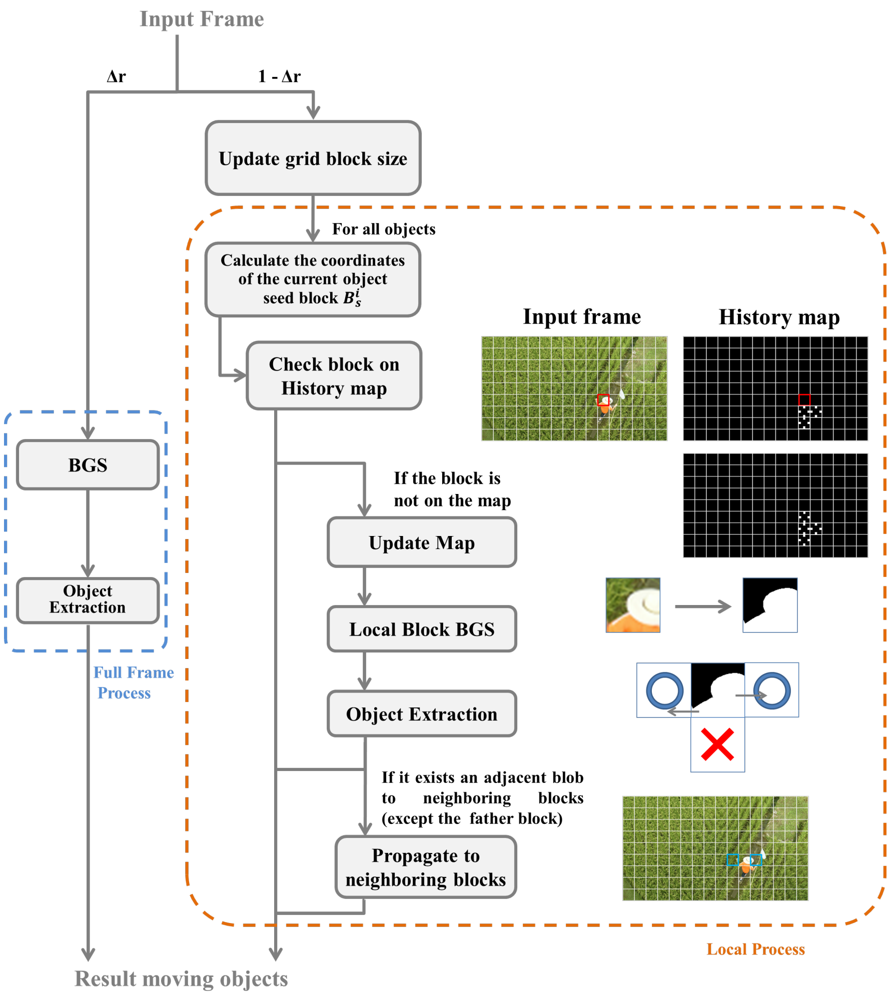

Input Frame
Δr
1 - Δr
Update grid block size
For all objects
Calculate the coordinates of the current object seed block B_S^i
Check block on History map
Input frame
History map
BGS
Object Extraction
Full Frame Process
If the block is not on the map
Update Map
Local Block BGS
Object Extraction
If it exists an adjacent blob to neighboring blocks (except the father block)
Propagate to neighboring blocks
Local Process
Result moving objects

Figure 3.2: Block detection flowchart. The current block iteration stops if the block has already been processed and it does not propagate further if no blob is on a border.

Objects can have various shapes and sizes through time. Therefore, the size of the blocks must be adapted to the detected content. A method such as the BPBGS which has a block size fixed at the beginning of the process is unable to be efficient in all the situations if the objects' dimensions' change. The ABPBGS has evolutive parameters to adapt the block size and morphological kernels to the size of the detected objects. The blocks are based on a temporary block History Map which is built at each frame i to avoid processing twice the same block. The shape and size of objects may change so the number of blocks $\mathbf{W}_i \times \mathbf{H}_i$ composing the map and their size $w_i \times h_i$ must be updated as well. The blocks cannot be too small, in which case the propagation gets slowed down, nor too big otherwise the proposed method loses its purpose of processing as few parts as possible. The width w_i (and respectively the height h_i) is defined by:

$$w_i = min\left(w_{MAX}, max\left(w_{MIN}, \frac{max(O_n^i(w_n))}{N_{parts}}\right)\right), \tag{3.1}$$

where w_{MIN} and w_{MAX} are respectively the minimum and maximum size values for a block, w_n the width of the nth object O_n^i detected at frame $i-1$ and N_{parts} the number of blocks it would take to go through an object horizontally or vertically. $N_{parts} = 3$ or 4 is enough for small objects (a tenth of the image's height or less). For very large objects, the propagation will more efficiently follow their shape by using small blocks rather than by using big blocks. Therefore, the size of the blocks is limited by $w_{MAX} \times h_{MAX}$ which values should be much smaller than the image's dimensions $w_{img} \times h_{img}$. The number of blocks $\mathbf{W}_i \times \mathbf{H}_i$ is determined by the size of the grid blocks:

$$\mathbf{W}_i = \frac{Img(w)}{w_i}, \qquad \mathbf{H}_i = \frac{Img(h)}{h_i}. \tag{3.2}$$

Then the position of the seed block $B_s(x_b, y_b)$ is set. Considering plain objects, the most suitable position for a good spreading is their center $O_n(x_c, y_c)$. The coordinates on the block grid are set by:

$$x_b = \left\lfloor \frac{x_c}{w_i} \right\rfloor \times w_i, \qquad y_b = \left\lfloor \frac{y_c}{h_i} \right\rfloor \times h_i. \tag{3.3}$$

Once the grid is established, the detection of the first seed block can start. It is a simple ROI-based detection but instead of using the region of interest from the previously detected objects, the region $R(x_b, y_b)$ is defined by the current block position and size. From the original frame only the block part is extracted and processed and after the memory for that block is released, keeping the memory usage at a minimum. The process consists of a static frame background subtraction followed by a threshold and morphological operations (open and close). A label of connected components ends the process. Since the block is rather small comparing to the size of the image, the label process is done quickly. When all the blocks necessary to detect the current object are processed, all the blobs extracted are merged to form the object's new boundaries.

The ABPBGS is based on the previous iteration detected objects to reduce the field of research for the current frame. To not miss objects and to avoid having too much discrepancies, a foreground detection using the full frame is required every Δ_r frames. The rest of the time, the ABPBGS loops on itself as shown in Figure 3.2. The value of Δ_r will determine the refreshing period at which a full scan of the frame will check for appearing objects. Its value should be dependent of the frame-rate (*fmr*) of the video. Refreshing too often would be a waste of resources since the state of the scene would not have change much since the last update. In the case of a too low value on the contrary, object could have been miss detected for a long time. Empirical tests have shown that $\Delta_r = 2/3 \times fmr$ was a suitable value for our purpose.

3.2.2 Detection Propagation

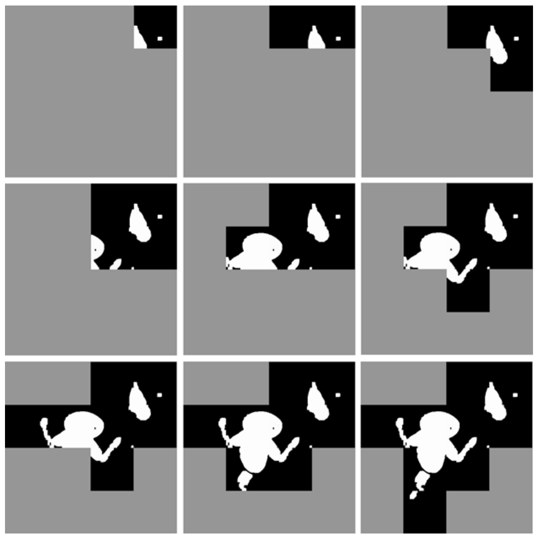
The principle of the propagation is very simple: after the foreground detection in a block, if a foreground object has been detected on a border, the next block corresponding to that border will be the subject of another detection. The process repeat itself until no more part of the current object is detected.

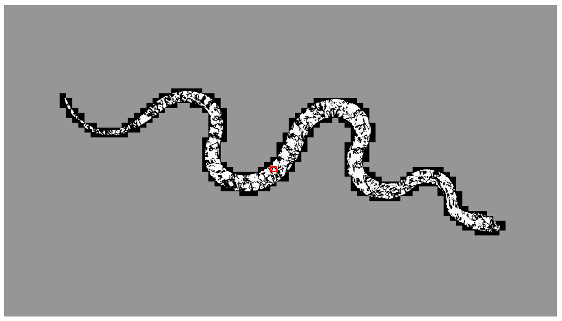
There are three conditions for the propagation to happen. The first condition is that there are connected components adjacent to one or more borders to the current block. The presence of foreground pixels on a border indicates that the object has been truncated and that another part is on the other side of the border in the adjacent block. The second condition is that this next block should not be the *father*-block which would have already propagated itself to the current block. Finally, we must avoid processing any block twice so we must check that next block has not been already processed anytime in the past for this frame. The *history map* is a virtual grid which keeps track of the blocks being or already processed. A simple and graphical way to keep track of already processed blocks and to verify if a block can be process of not, is to set, at the position of the processed block, 4 pixels to a non-zero value, one for each border (top, bottom, left, right). The position of those pixels is set to the middle of each border of the block being memorized/verified so the comparison can be as fast as the pixel access. In the case of a propagation other than based on Γ_4, for example a Γ_8 propagation where the propagation would be done through the corners, the history map would have to be adapted with the indication of the corner propagations.

If one of those propagation conditions is not verified, the detection does not continue further. The detection for the current object stops when the detection no longer propagates itself. Figure 3.3 presents two examples of the propagation, one showing the propagation step by step and the other the final result of the ABPBGS on an object of complex shape. The gray areas are the unprocessed zones. They did not even have been extracted from the current frame for any process. The propagation is neither limited by the size nor the shape of the detected objects and will continue as long as a part of the objects can be detected (Figures 3.3b and 3.3c).

3.3 Experimental Results

In order to test the full potential of our ABPBGS, we captured 5 sequences [64] in 4K@60fps (4K video with a framerate of 60fps) and we added the field sequence from the nebuta festival 3.4. We created the Ground Truth information (GT) for all the sequences. Chapter A presents the dataset with visuals and more details.


(a) Example of block-propagation iterations from the seed block up-left to the final result bottom-right.


(b) Example on a more complex shape, a snake. The red rectangle is the seed block. Unprocessed areas are in gray.

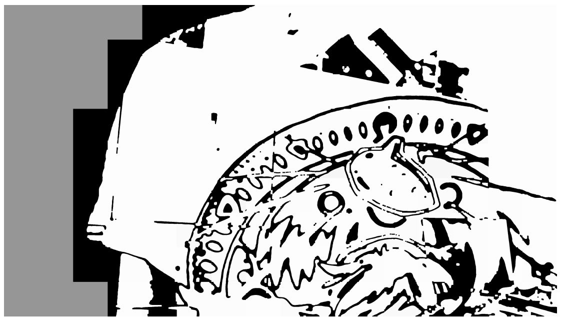
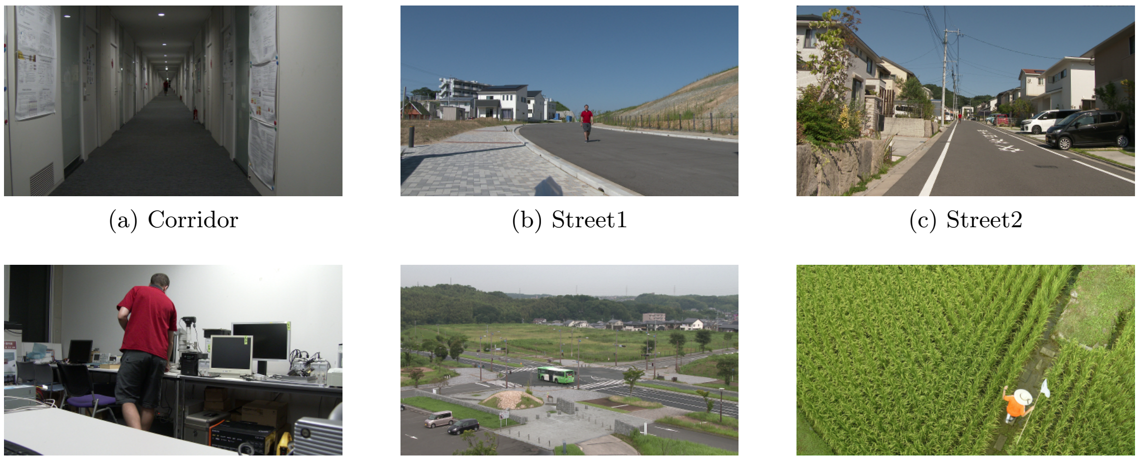

(c) Example on a very large object. There are no limits to the propagation detection. Unprocessed areas are in gray.

Figure 3.3: Three examples of the propagation of the ABPBGS. On top the process step by step; in the middle the ABPBGS used to process a snake which has a complex shape; on the bottom the ABPBGS used to detect a very big object.

The videos contain various situations with objects of different size, from less than 0.1% of the image size to height of the image (*Corridor*, *Street1*, *Street2*). The difference between those videos is the complexity of the background: *Street1* has a clear setting when *Street2* has a complex background and *Corridor* is rather dark. Some videos (*Crossing*, *Field*) contain strong background movements such as moving grass or trees which are a very common issue in video surveillance.

The parameters used are as follow: $\Delta_r = 40$, $T = 30$, $N_{parts} = 3$. We developed in C++ with the OpenCV library (http://opencv.org) and OpenMP 2.0 (http://www.openmp.org), the computer used is a Quadcore i7@2.83GHz with 16 Go of RAM.



(a) Corridor (b) Street1 (c) Street2

(d) Screws (e) Crossing (f) Field

Figure 3.4: 8K UHD and 4K UHD dataset used for the testing.

From the 29 state-of-art algorithms presented in [52] and available in the BGSLibrary [51] for OpenCV, we selected the best 9 according to our experiments. The 9 methods have been selected if they appeared at least one time in the top 5 in at least one sequence. Those methods are: PBAS [34], DPZivkovicAGMMBGS [58], MultiLayerBGS [59], DPAdaptive-MedianBGS [53], AdaptiveSelectiveBGLearning, DPWrenGABGS [55], DPPratiMediodBGS [54], IndependentMultimodalBGS [65], DPEigenbackgroundBGS [60]. Their respective parameters have been set as described in [52].

Although the main purpose of this work is to process UHD video, the comparison could not be done exclusively on 4K UHD scale. Indeed, we have seen in the introduction chapter that the state-of-art methods are very slow to process UHD videos. For example, the PBAS takes an estimated 14706 s to process one frame of the 8K sequence. It represents more than 4 hours to process a single frame and it would require too long to compute the results for all videos for each state-of-the-art method. For this reason we will compare detection quality metrics of the ABPBGS in both 4K and 270p scale to the state-of-art methods running on a 270p down-scaled version. It is important to notice that only the results of the proposed ABPBGS on 4K videos represent the real interest of this work as it has been solely developed for UHD processing.

Table 3.1: Quality comparison of the ABPBGS with state-of-art methods. Best scores are in bold.

Method ID	Size	Recall	Pre.	Sim.	F-Meas.	SSIM	A-Score	Speed(fps)
ABPBGS (Proposal)	4K	0.384	**0.495**	0.317	**0.407**	0.991	**0.572**	5.18
	270p	0.339	0.430	0.358	0.276	**0.993**	0.550	**35.58**
BPBGS	4K	0.376	0.461	**0.367**	0.284	0.974	0.542	5.06
PBAS [34]	270p	0.276	0.327	0.230	0.275	0.974	0.493	2.04
DPZivkovicAGMM [58]	270p	0.566	0.310	0.243	0.308	0.894	0.482	13.93
MultiLayer [59]	270p	0.244	0.333	0.197	0.251	0.982	0.477	2.85
DPAdaptiveMedian [53]	270p	0.348	0.363	0.166	0.234	0.953	0.451	20.56
AdaptiveSelectiveBGLearning	270p	0.399	0.268	0.187	0.241	0.914	0.447	14.02
DPWrenGA [55]	270p	0.510	0.245	0.171	0.234	0.913	0.439	12.35
DPPratiMediod [54]	270p	0.425	0.266	0.164	0.227	0.924	0.438	3.68
IndependentMulti. [65]	270p	0.529	0.186	0.163	0.219	0.932	0.438	2.04
DPEigenBGS [60]	270p	**0.802**	0.146	0.139	0.179	0.746	0.355	3.28

Static quality metrics are used to compare the different methods: the *recall* which represents the correct detection, the *precision* (Pre.) which represents the influence of the background noise and three similarity scores *similarity* (Sim.), *F-Measure* and Structural Similarity (SSIM)[66]. The Peak Signal to Noise Ratio (PSNR) is also mentioned. In addition to the static metrics presented in section B.2 we also compute a global score *A–Score* as:

$$A\text{–}Score = \frac{Similarity + F\text{–}Measure + SSIM}{3}. \tag{3.4}$$

Table 3.1 shows a comparison of the average scores of all the methods. The similarity and F-Measure scores are the mean of the similarity and F-Measure scores for each video. The recall metric shows the ability to detect the objects of interest but does not take into account the number of false positives (noise from moving background or illumination variation). This is why even though the Eigenbackground method achieves recall score of 0.802, it is last on the average ranking because it is unable to deal efficiently with background noise. Indeed, in sequences in which objects are small or those which contain a lot of background movement, the ratio of potential false positives and objects of interest is very high. The precision shows the capacity of a method to be able to handle background noise and since the objects are much smaller than the size of the image, the precision has a much bigger impact on the final results than the recall score. Compared to the best of the state-of-the-art method, the PBAS, our ABPBGS not only performs the best on the precision rate 0.327→0.495 (+51%) but also on recall rate with 0.275→0.384 (+39%). We can finally observe that the ABPBGS obtains the best results compare to all other methods with an score of *A–Score* 0.572 at 4K. In comparison that is 16% to 61% better than the state-of-art methods: 0.493→0.572 (+16%) compared to the PBAS (ranked 1st among the state-of-the-art methods), 0.482→0.572 (+19%) compared to the ZivkovicAGMM (ranked 2nd) and 0.477→0.572 (+20%) compared to the MultiLayer (randked 3rd).

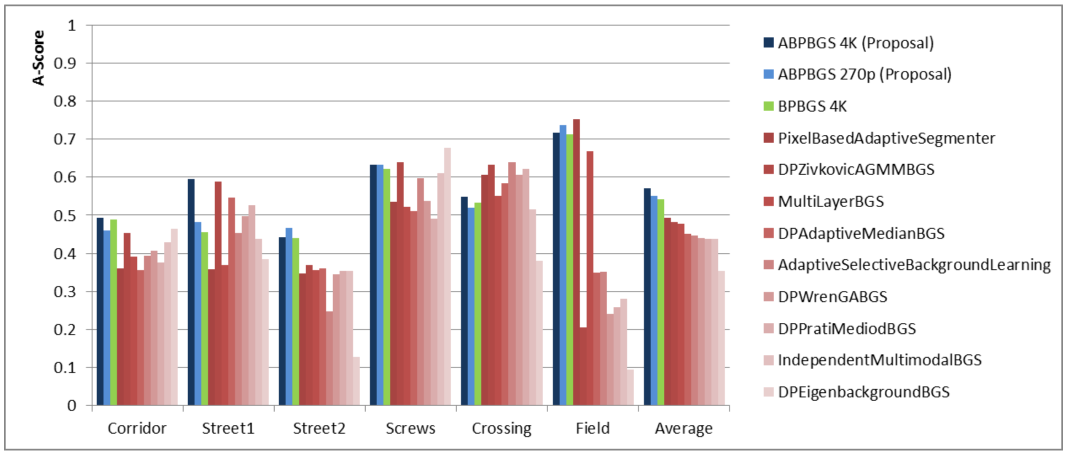
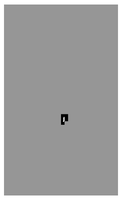


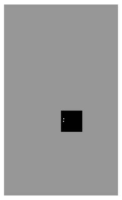
Figure 3.5: Bar graph of the A-Score. The graph shows the result of the ABPBGS processing 4K images in blue compared to the state-of-the-art methods processing 270p (in nuances of red).

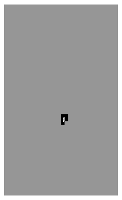
The reason for this gap in scores is very visible in Figure 3.5 which compares the *A–Score* of the ABPBGS_4K to the other methods for each sequence. In the sequences where the object of interest is very small most of the time, the proposed ABPBGS shows better quality results because it is among the few method to be able to detect the small objects but also to keep the number of false positives very low as we can see in Figures 3.12, 3.13 and 3.14. In scene containing moving background parts such as grass or leaves from trees (Figures 3.13 and 3.17), the propagation aspect of the ABPBGS shows anti-false positive properties. Indeed, even though some background part get detected, since those are only temporary movements, few frames later the propagation will propagate less and less on those areas thus detecting less and less false positives. In Figure 3.17, we can observe three groups of behaviors among the different methods: those which can deal with the background noise (ABPBGS, PBAS and MultiLayer), the method which is completely overwhelmed (DPEigenBackground) and finally those which are not able to erase most of the noise. If the PBAS seems more efficient to erase the noise, the shape obtained by the detection from the ABPBGS fits better the real state of the ground truth. Also, even if some noise is detected in one frame, the propagation process is more likely to detect less and less noise than adding more and more. Most of the noise detected in the ABPBGS come from the surroundings of the object of interest (Figures 3.13 and 3.17).

Figure 3.6 shows the differences between the BPBGS and the ABPBGS on the *street1* sequence. This sequence shows the main interest of adapting the block size to the detected content. Indeed, we can observe that, with the same block size, the BPBGS is not as efficient when the size of the content varies. Small blocks work better with small objects and big blocks with big objects. The ABPBGS which main interest is to be able to adapt its block size to the content shows that it is much faster that the BPBGS 16x9 in detecting tiny objects 0.083s→0.066s (-26%) (Figures 3.6c) and 0.25s→0.18s (-28%) (Figures 3.6g). On another side, the results of the BPBGS 96x54 show that constantly using small blocks can be an asset for tiny objects but becomes a drawback on larger objects as the number of blocks to process gets higher and the cost of the propagation becomes an issue. In that case, using bigger blocks is more efficient: 1.16s→0.82s (-29%). Small blocks may also be too small to

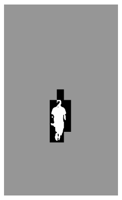
(a) Frm. 2330

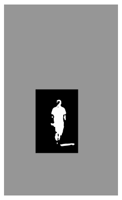

(b) 96x54 0.066s

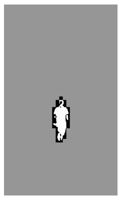

(c) 0.083s


(d) 0.066s

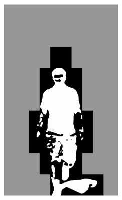
(e) Frm. 3999

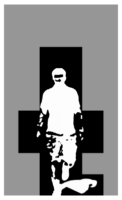

(f) 48x18 0.18s

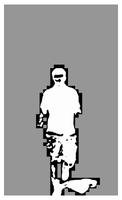

(g) 0.25s

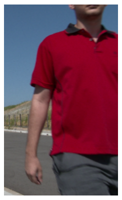

(h) 0.15s

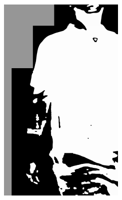
(i) Frm. 4281

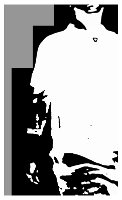

(j) 32x9 0.46s


(k) 0.47s


(l) 0.50s


(m) Frm. 4392


(n) 16x9 0.82s


(o) 0.82s

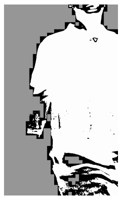

(p) 1.16s

Figure 3.6: Average speed and processed areas comparison of some frames from Street1 sequence between ABPBGS and BPBGS. From left to right: cropped original image, ABPBGS, BPBGS 16x9, BPBGS 96x54. Fastest times in green, slowest times in red. Grey areas are unprocessed.

reach all fragments thus missing some parts like the missing arms in Figures 3.6l and 3.6p. The last and probably biggest issue of using small blocks for the detection of big objects is the number of propagation required. Not only each propagation is done in a fixed amount of time but also each propagation is another call on the stack of the program. With the multiplication of the blocks to propagate to there is a risk to exceed the capacity of the stack resulting in an stack overflow and a crash of the program. On average, the ABPBGS quality score is an improvement of the BPBGS 0.542$\rightarrow$0.572 (+6%).

The speed calculations in Table 3.1 and Figure 3.7 include the label of the connected components. We have made this choice for two reasons. First we consider that the foreground detection is complete only after the extraction of the moving objects and not just the foreground mask. Second, including the label in the speed calculation is a way to penalize methods which create very fragmented objects. Table 3.1 presents the computational speeds measured in frames per second (fps) and Figure 3.7 relates both the speed and the average score with the processing time per pixel and the A-Score. Using the processing time per pixel allow us to compare all the methods on the same ground even though they do not process at the same scale the videos. Indeed, with an average speed of 5.18 fps, the proposed ABPBGS is about 154% better than the PBAS (2.04 fps) if we consider the speed in frames per seconds. However, considering the processing time per pixel, the ABPBGS is, in fact, much faster than the PBAS $3.78\times10^{-6}\rightarrow2.33\times10^{-8}$ px/s (-16343%). Moreover, with 13.98 fps, the ZivkovicAGMM seems much faster than our proposed ABPBGS but taken back to the processing time per pixel it is in fact much slower $5.54\times10^{-7}\rightarrow2.33\times10^{-8}$ px/s(-2309%). Figure 3.7also shows that proposed ABPBGS performs better than the non-adaptive version the BPBGS. The ABPBGS is indeed about 37% faster ($3.71\times10^{-8}\rightarrow2.33\times10^{-8}$ px/s). The 270p version of the ABPBGS performs generally faster than any other method, but the processing time per pixel actually shows that it requires more time per pixel than the 4K scale $2.33\times10^{-8}\rightarrow2.17\times10^{-7}$ (+831%). This shows that the ABPBGS is best to process ultra high resolution rather than SD videos. This is explained by the propagation time which is noticeable when processing SD videos and becomes less and less observable when processing very big images.

Figure 3.8 compares the memory consumption depending on the scale used. The ABPBGS processes block by block as if they were independent images and in addition to that only

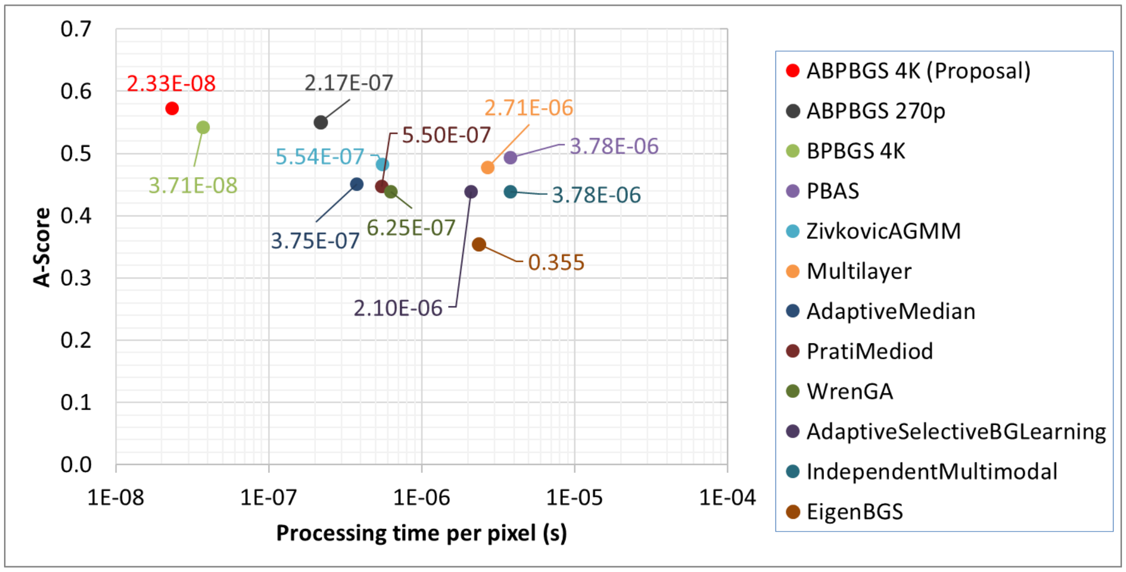


Figure 3.7: Comparison of the average processing time per pixel and A-Score.

the blocks surrounding the object of interest are processed. The state-of-the-art methods, however, compute the full frame and require to store and compute a lot more data. We can observe that starting from 1080p, the ABPBGS is very interesting to save memory but at 8K, the difference is enormous with a ratio of 24. Indeed, the memory usage for the ABPBGS is about 450 MB and the memory consumption for the PBAS is about 11 GB.

3.4 Discussion

3.4.1 Full frame refresh

The first point of discussion is about the refreshing. The principle asset and interest is that the detection is spread from a point of an object until it is fully detected. There are two consequences to this. The refreshing which is done by processing the full frame, is a necessary step. Indeed, without any refresh of the detection on other parts of the image, then we would not detect any object which would have appeared at another place than the ones close to existing objects. An approach similar to [7] in which some regions are check randomly, would be interesting. The size of the region would have to be set correctly and it might also be an

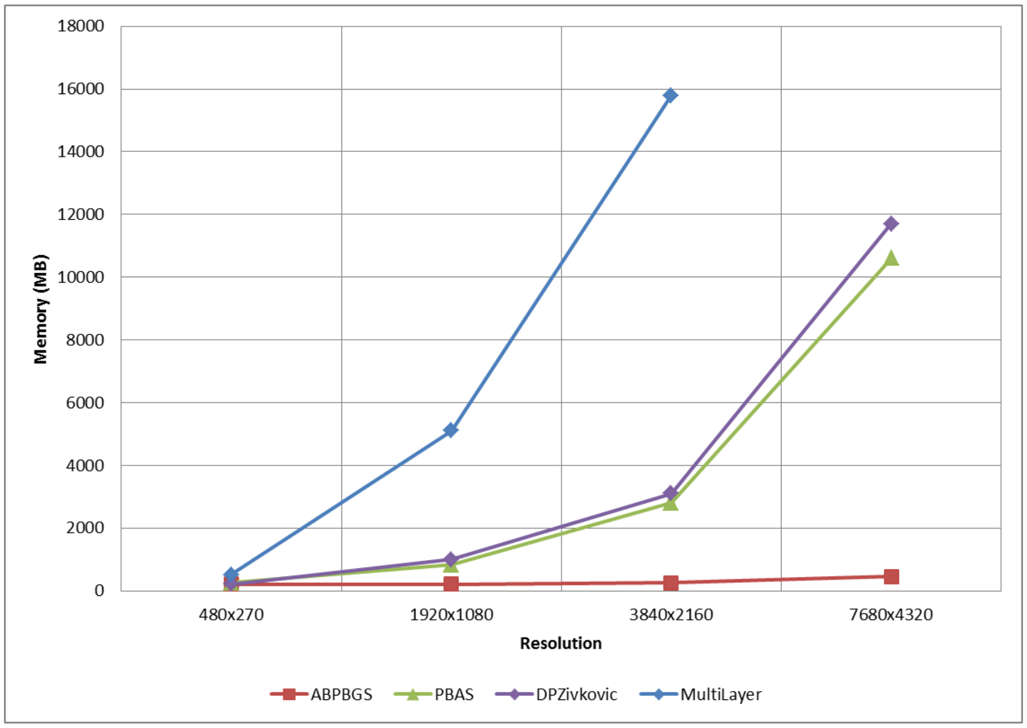
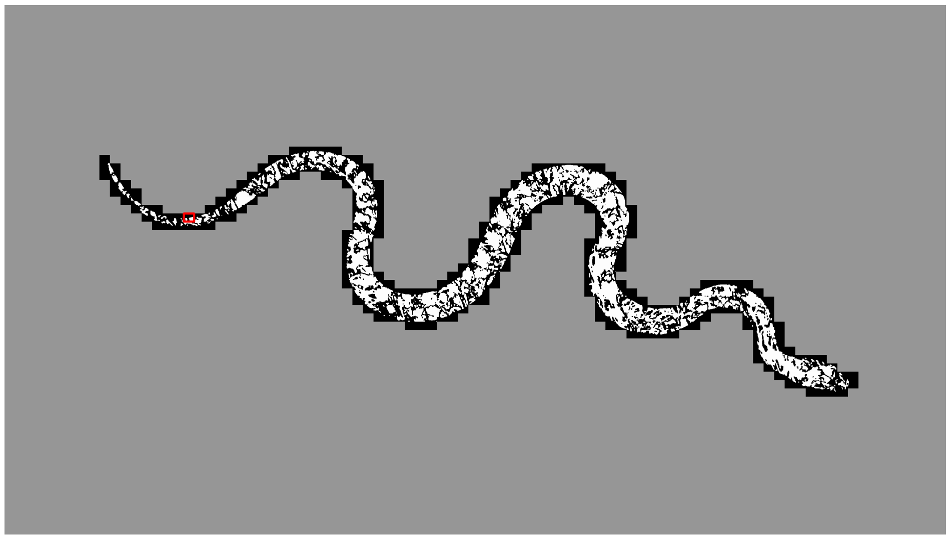


Figure 3.8: Comparison of the memory consumption depending on the resolution used, from 270p to 8K UHD. Except for the ABPBGS (in red), the results for resolutions above 1080p are estimations of the consumption from the 20-50 first frames since the algorithms were unreasonably slow to process the full sequence.

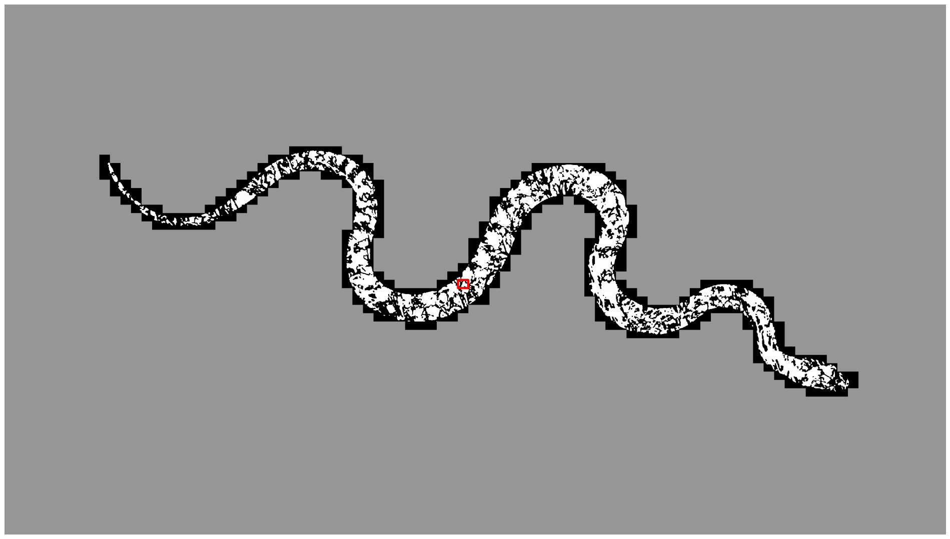
issue if too many of those are required to detect small objects.

3.4.2 Background update

Another point of improvement is the update of the background. The update of the background is very time consuming and goes directly against the purpose of not processing uninteresting areas. Indeed, in methods which adapt the background, the pixels detected as background are used for the update. All those pixels belong to areas the ABPBGS tries to avoid processing. There is a possibility of updating the background which lies in the refreshing step. Since the refresh process requires to deal with the full frame, a background update could be effectuated. However, this kind of approach is not much different from the existing methods and the main drawback is the increase of the memory usage and processing time. In the next chapter, a



(a) Seed on the tail of the snake: 83 ms.



(b) Seed in the middle of the snake: 49 ms.

Figure 3.9: Speed comparison depending on the position of the seed block (in red). On the top the seed has been set at an extremity of the snake while on the bottom the seed has been implanted in the middle. Unprocessed areas are in gray.

background modelling method adapted for UHD processing will be presented.

3.4.3 Position of the seed and adapted content

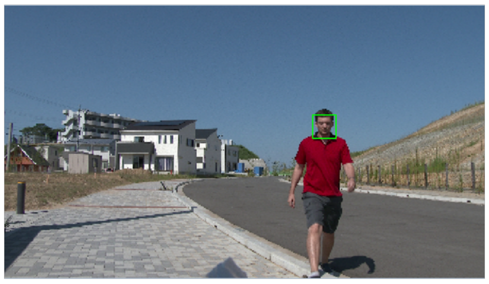
The next point is the position of the seed. The ABPBGS propagates itself via independent block detections. For this reason, the position of the seed can have a great influence on the detection speed as seen in Figure 3.9. When the seed is placed in the center of the object, the detection speed is much faster than if the seed is placed at an extremity. For objects which occupy the center of their boundary box, it is not an issue but it can be for objects for which the center of the boundary box like the snake. The propagation can be optimized if the seed is positioned at a place where a part of the object is sure to be detected and where the propagation time would be minimal.

The last point is the size and number of objects. The main purpose of this work is to skip the processing of all unnecessary area. However, if the objects cover all the image then this proposal would be inefficient since no part could be skipped. Therefore, videos which content is exclusively objects taking all the image are not suitable for a propagative method like the ABPBGS

3.4.4 Face detection application

In Chapter 2 we have seen that a moving object detector could be coupled with a tracking system. Here we will present another direct application with a face detection. The first experiment is very simple. We detect faces in the sequence Street1 at two different scales, one at 270p, the other at 4K. In the video, the man starts from far and he is coming closer and closer. Therefore, it is possible to calculate the minimum size in which a face can be detected by checking when a face is detected for the first time.

Figure 3.10 shows that in 270p the first frame in which a face has been detected is the 4266th frame on a total of 4439. When processed at 4K, a face can be detected on frame 3016. When we compare the two images, we can clearly see that the objects' size is beyond comparison. This experiment is another example of the use of UHD videos to detect very


(a) (270p) frame #4266/4439

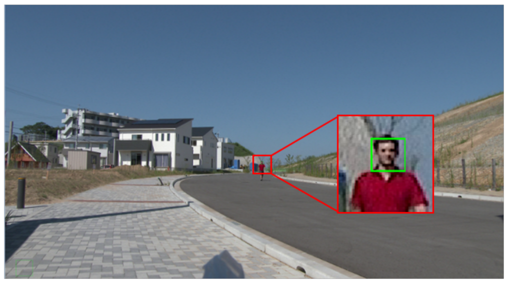

(b) (4K) frame #3016/4439

Figure 3.10: First frames in which a face has been detected using two different resolutions on the Street1 4K sequence. On the left a down-sampled video to 270p, on the right the original 4K. By using the original 4K image, a face could be detected 1250 frames earlier than by using the 270p resolution.

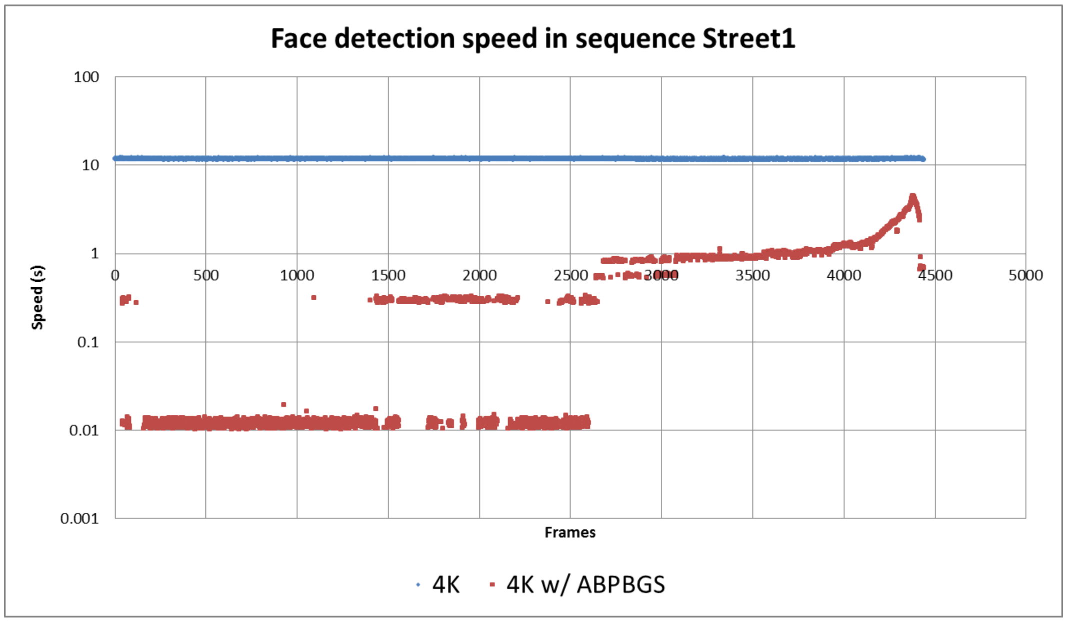


Figure 3.11: Evolution of the speed of the face detection process on a 4K UHD sequence with the use of the ABPBGS beforehand (red) and without (blue).

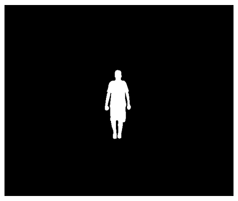
small objects. Figure 3.11 presents a comparison of the face detection speed with and without using the ABPBGS. Without, the face detection performs at a constant speed (10s/frame), proof that it processes the full frame. By using the ABPBGS, the face detection is about 23 times faster on average.

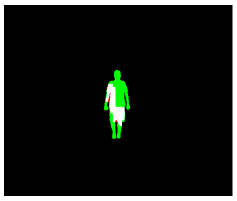
3.5 Conclusion

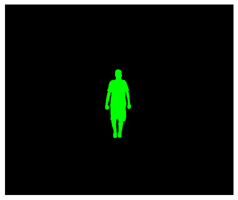
We presented in this chapter a novel foreground detection methods which follows the shape of the objects to process just the necessary area. This Adaptive Block-Propagative Background Subtraction or ABPBGS has shown the best results with a global score of 0.572 for an average processing time per pixel of 2.33×10^{-8} (about 5.18 fps in average for 4K videos). Moreover, the memory usage is very low, with only 450 MB used to process an 8K UHD sequence while the best state-of-the-art methods require 11 GB.

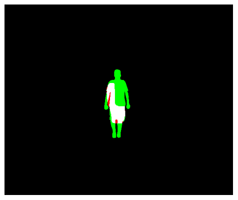
Further improvement is possible and necessary by improving the background modelling so it could adapt to variation of illumination. Another axis of improvement is to develop an alternative to the full frame process to detect new objects.

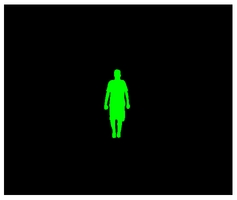
(a) ROI

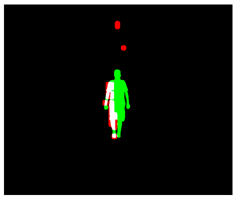

(b) Ground Truth

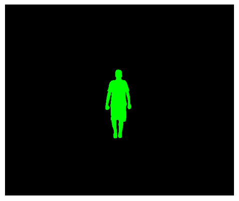

(c) ABPBGS 4K

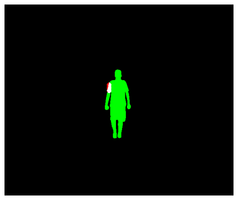

(d) ABPBGS 270p

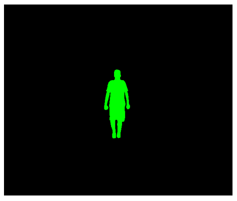

(e) BPBGS 4K

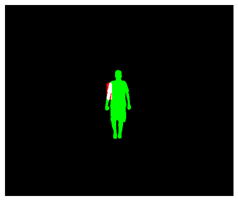

(f) PBAS 270p

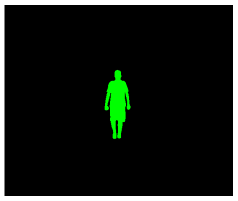

(g) DPZivkovicAGMM-BGS 270p

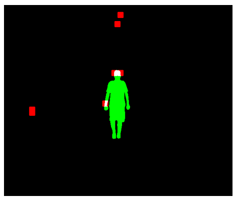

(h) MultiLayerBGS 270p


(i) DPAdaptiveMedianBGS 270p


(j) DPPratiMediod 270p


(k) DPWren 270p


(l) AdaptiveSelectiveBGLearn. 270p


(m) Independent-Multim. 270p

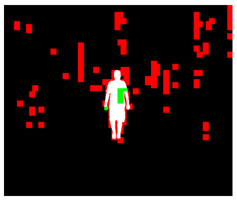
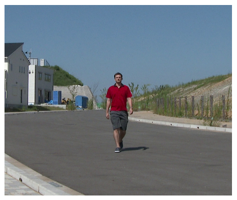
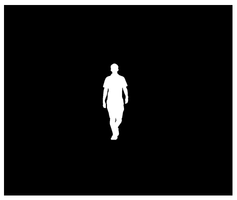
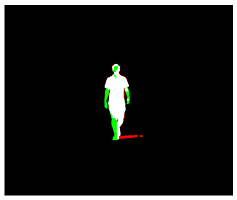
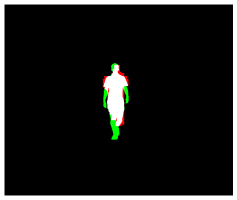

(n) EigenBGS 270p

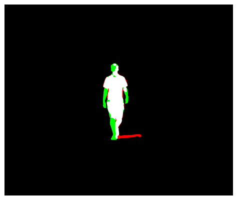
Figure 3.12: Foreground mask obtained from the different BGS algorithms for the Corridor sequence. Color map: TP-white, TN-black, FP-red, FN-green.

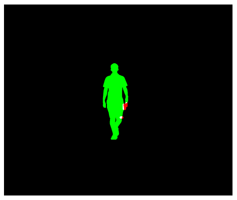

(a) ROI

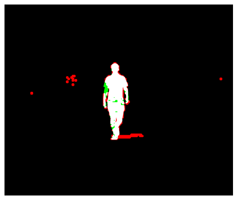

(b) Ground Truth

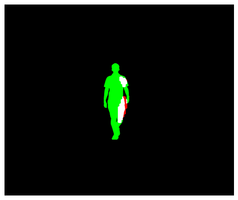

(c) ABPBGS 4K

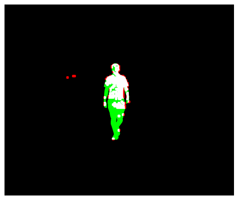

(d) ABPBGS 270p

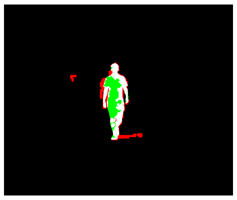

(e) BPBGS 4K

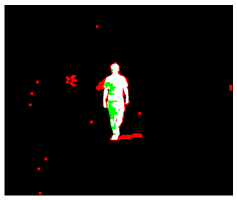

(f) PBAS 270p

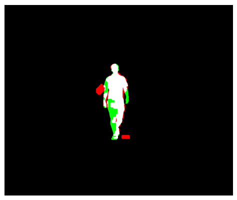

(g) DPZivkovicAGMM-BGS 270p

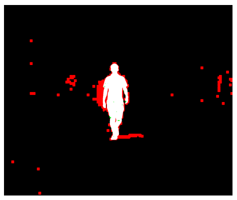

(h) MultiLayerBGS 270p


(i) DPAdaptiveMedianBGS 270p


(j) DPPratiMediod 270p


(k) DPWren 270p


(l) AdaptiveSelectiveBGLearn. 270p


(m) Independent-Multim. 270p

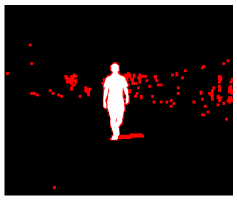
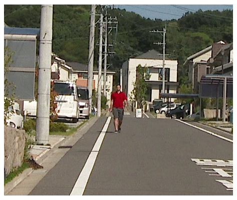
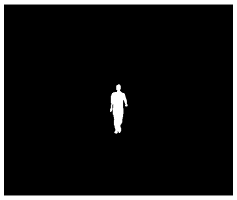
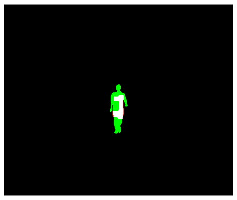
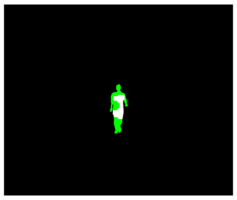

(n) EigenBGS 270p

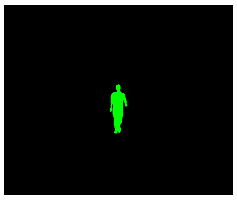
Figure 3.13: Foreground mask obtained from the different BGS algorithms for the Street1 sequence. Color map: TP-white, TN-black, FP-red, FN-green.

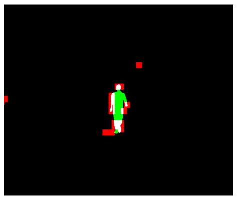

(a) ROI

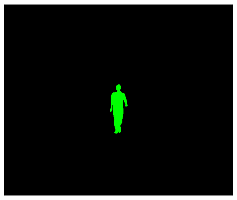

(b) Ground Truth

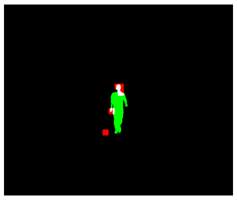

(c) ABPBGS 4K

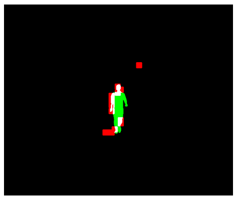

(d) BPBGS 4K

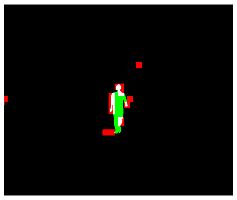

(e) PBAS 270p

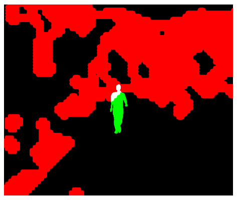

(f) DPZivkovicAGMM-BGS 270p

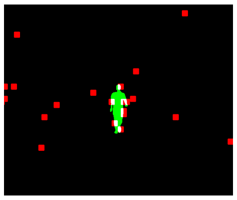

(g) MultiLayerBGS 270p


(h) DPAdaptiveMedianBGS 270p


(i) DPPratiMediod 270p


(j) DPWren 270p


(k) AdaptiveSelectiveBackgroundLearning 270p


(l) IndependentMultim. 270p

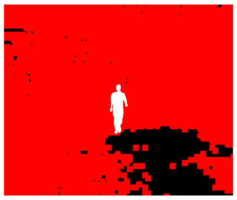
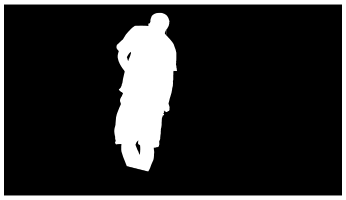
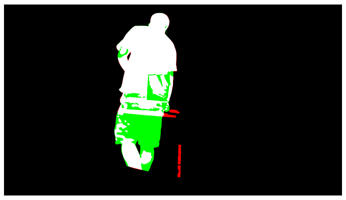
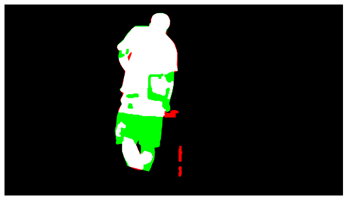

(m) EigenBGS 270p

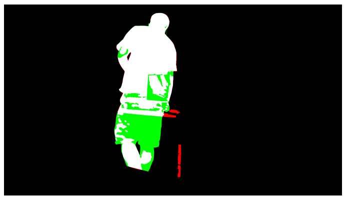
Figure 3.14: Foreground mask obtained from the different BGS algorithms for the Street2 sequence. Color map: TP-white, TN-black, FP-red, FN-green.

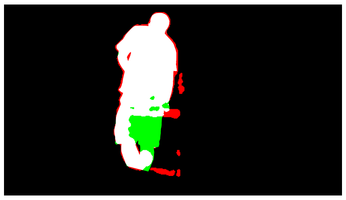
(a) Original

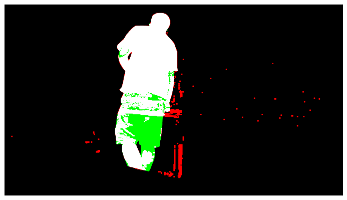

(b) Ground Truth

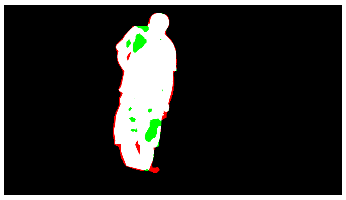

(c) ABPBGS 4K

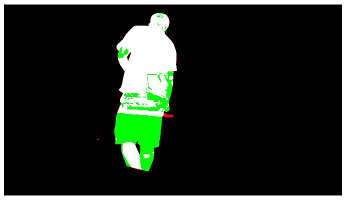

(d) ABPBGS 270p

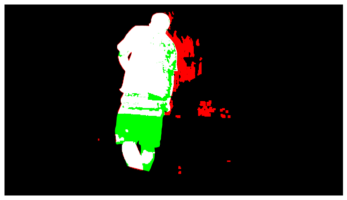

(e) BBPBGS 4K

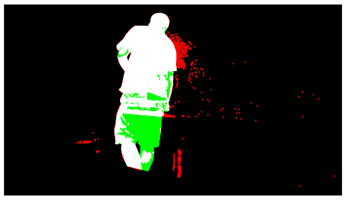

(f) PBAS 270p

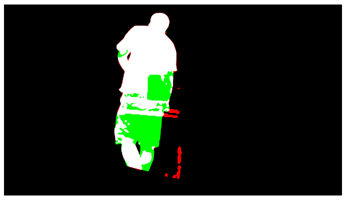

(g) DPZivkovicAGMMBGS 270p

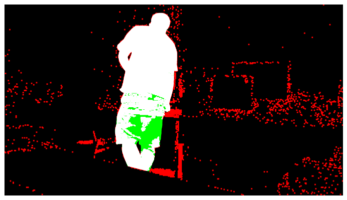

(h) MultiLayerBGS 270p


(i) DPAdaptiveMedianBGS 270p


(j) DPPratiMediod 270p


(k) DPWren 270p


(l) AdaptiveSelectiveBackgroundLearning 270p


(m) IndependentMultim. 270p

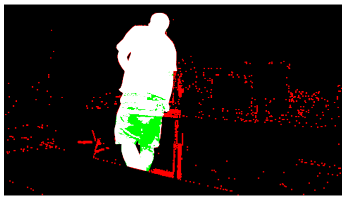
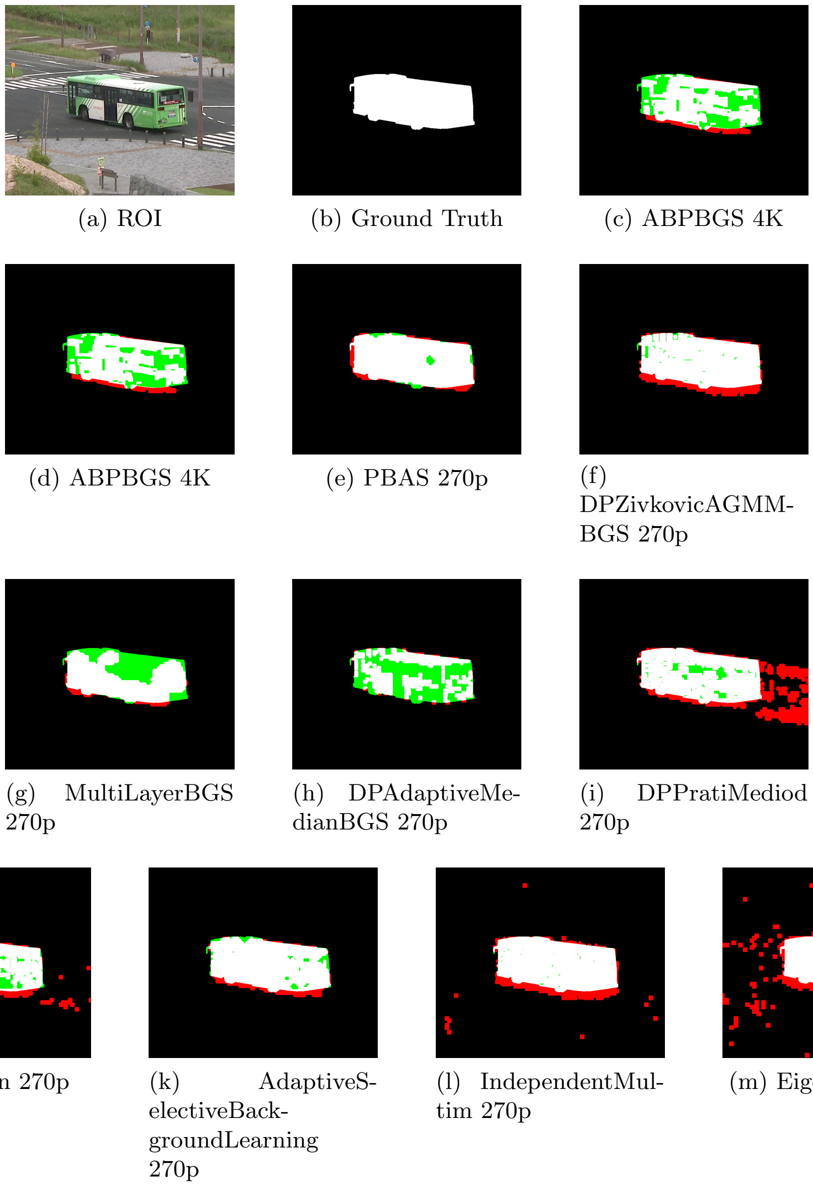

(n) EigenBGS 270p

Figure 3.15: Foreground mask obtained from the different BGS algorithms for the Screws sequence. Color map: TP-white, TN-black, FP-red, FN-green.


(a) ROI

(b) Ground Truth

(c) ABPBGS 4K

(d) ABPBGS 4K

(e) PBAS 270p

(f) DPZivkovicAGMM-BGS 270p

(g) MultiLayerBGS 270p

(h) DPAdaptiveMedianBGS 270p

(i) DPPratiMediod 270p

(j) DPWren 270p

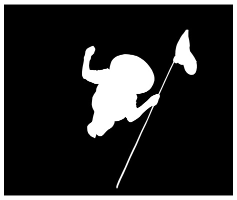
(k) AdaptiveSelectiveBackgroundLearning 270p

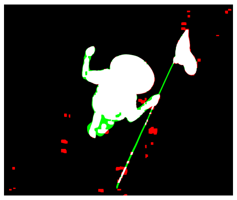
(l) IndependentMultim 270p

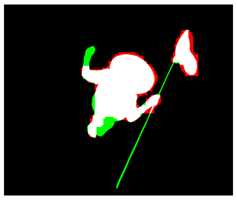
(m) EigenBGS 270p

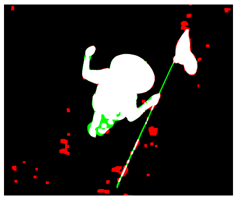
Figure 3.16: Foreground mask obtained from the different BGS algorithms for the Crossing sequence. Color map: TP-white, TN-black, FP-red, FN-green.

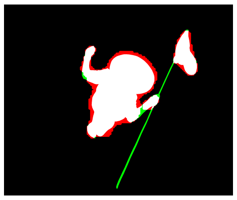
(a) ROI

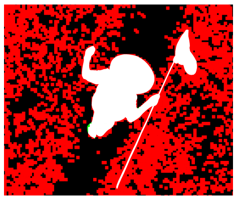

(b) Ground Truth

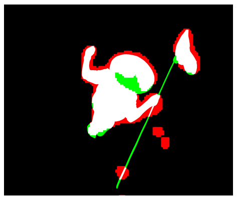

(c) ABPBGS 8K

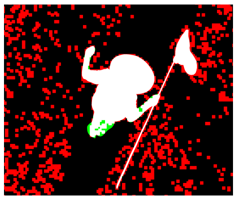

(d) ABPBGS 270p

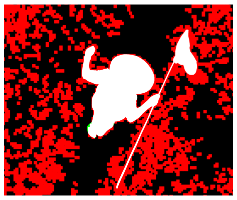

(e) BPBGS 8K

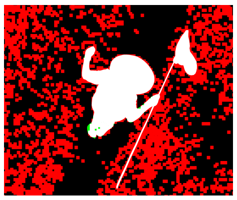

(f) PBAS 270p

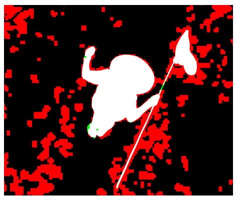

(g) DPZivkovicAGMM-BGS 270p

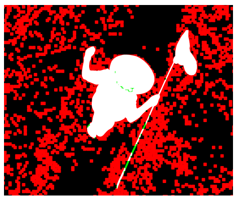

(h) MultiLayerBGS 270p


(i) DPAdaptiveMedianBGS 270p


(j) DPPratiMediod 270p


(k) DPWren 270p


(l) AdaptiveSelectiveBackgroundLearning 270p


(m) Independent-Multim. 270p

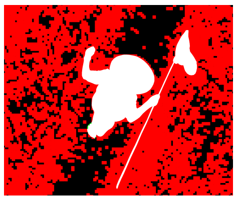

(n) EigenBGS 270p

Figure 3.17: Foreground mask obtained from the different BGS algorithms for the Field sequence. Color map: TP-white, TN-black, FP-red, FN-green.

Chapter 4

Mixed Block Background Modelling for Foreground Detection in UHD videos [†††]

4.1 Introduction

As a following to the conclusion of the previous chapter, modelling the background is the next necessary focus in order to handle more various situations. A foreground detector system has to be able to deal with modifications of the background scene such as long staying objects and variations of illumination. Adapting the background model to those modifications can require a lot of resources in time and memory. Indeed, the modelling can be done by taking into account multiple previous frames and/or by using a cluster of parameters for the whole frame, for each block of pixels or even for each pixel. State-of-the-art methods have been designed for SD videos as most of the focus was on the improvement of the detection quality rather than on the detection speed. As a result, very efficient but complex methods have been created and the average requirements in memory and time has also increased. With UHD videos,

[†††] This chapter is adapted from the work published in [J1]

those requirements become the bottleneck of the state-of-the-art methods [34][59][58] and it is now necessary to employ algorithms which take the processing speed into account. Indeed, those methods can take months to process a single 10s video sequence at 4K UHD [67] which makes them unfit for UHD foreground detection. We have seen in the previous chapter that block-based methods designed for UHD videos with the Adaptive Block Propagative Background Subtraction (ABPBGS) [J2] but that method lacks a background modelling function. Among the existing background modelling methods figure statistical methods using one Gaussian [55] or multiple Gaussians [16], [56], [57], [58]. More recently spatio-temporal methods based on the mean and variance [53], [54] as well as non-parametric methods [34] or even based on colors and textures [59] have been developed. All those methods as well as block-based methods [68] [69] have in common the fact that all the blocks and pixels are processed every frame. The second similarity among them is that they require a cluster of parameters for each pixel or each block. Updating those parameters at each frame makes the computational time even larger.

The objective of this work is to quickly detect foreground objects from a 4K UHD input video. To achieve this goal, we propose to improve the existing ABPBGS by adding a new background modelling which novelty lies in two points:

- Using small blocks as Region of Interest (ROI) to reduce the total amount of image processed at each frame by updating a very small number of blocks instead of the whole image. This will allow the update parts to process in real time.
- A homogeneous update by using a partition in Mega-Blocks of the background model and a mixed linear and pseudo random selection of blocks to avoid updating the same parts in consecutive frames. This will ensure that the update is well spread on the whole image and that there are very limited differences between adjacent blocks.

The mandatory condition for this method to be effective is to have reduced differences between consecutive frames. Therefore, the input videos must have a framerate high enough ($\geq$60 fps) and the speed of the light variation must be must slower than the speed of the moving objects of interest.

This chapter is organized as follow: Section 4.2 presents the proposed method in details. Section 4.4 show the different results by comparing the proposed method to the state-of-the-art methods. Finally, section 4.5 closes this chapter.

4.2 System overview

The main principle of the MBBM is to update little by little the background model as it would require too much resources to process every pixels in the frame at once. To reduce the temporal differences among blocks as much as possible, only sequences with a high framerate ($\geq$60 fps) in which the light variations happen gradually are considered. In order to obtain a uniformed model, we consider Mega-Blocks (MB) and Sub-Blocks (SB) defined as in Figure 4.1. The image of size $width$ and $height$ contains a total $nb\_mb\_w \times nb\_mb\_h$ MB, themselves containing $mb\_size\_w \times mb\_size\_h$ SB (Figure 4.1). The size of those SB is defined by the following equation:

$$sb\_size\_w = \frac{width}{nb\_mb\_w \times mb\_size\_w}, \qquad sb\_size\_h = \frac{height}{nb\_mb\_w \times mb\_size\_w}. \tag{4.1}$$

The number of SB defines the updating speed as two blocks will be updated at each time t. Unlike other works using blocks [68], [69], the proposed work does not process all the blocks but only two for each Mega-Block, one by linear order selection and one by pseudo random selection.

For each mega-block, a sub-block is selected at random by a uniformed random id generator. The id S^M is made from the sub block coordinates inside its mega-block as $S^M \in ([0; mb\_size\_w[; [0; mb\_size\_h[)$. The randomness is necessary in order to not have systematically a temporal difference in the update between two adjacent blocks. Indeed, if the update was done from top left block to bottom right, then there would always be
Even though the selection is random we want to make sure that every block is processed

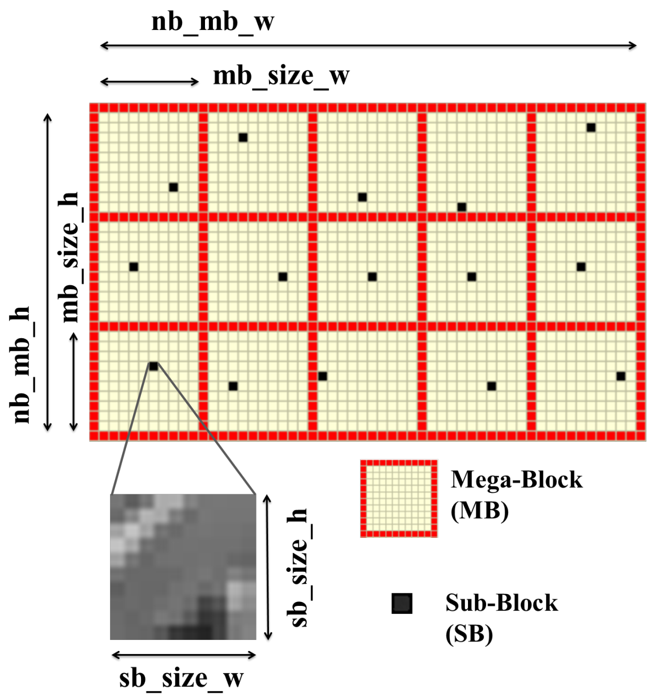


Figure 4.1: The Mega-Blocks (in red) contain several Sub-Blocks themselves containing pixels. Each time the background model is updated, one random Sub-Block (in black) per Mega-Block is chosen to update a part of the background model.

at least once in a while. To decrease the chances of a block which has already been picked recently, we use a *preview map* which is an greyscale image of size $nb\_sb\_w \times nb\_sb\_h$ defined as:

$$nb\_sb\_w = nb\_mb\_w \times mb\_size\_w, \qquad nb\_sb\_h = nb\_mb\_h \times mb\_size\_h. \tag{4.2}$$

Each pixel of the map represents one of the sub-blocks. If the pixel value of the selected SB is lower than a threshold value τ, the block can be updated. If the value is higher, then a new block id is generated until we find a correct block. Once the chosen SB have been processed, the value of their corresponding pixel on the preview map is set to 255 and all the other pixels on the map have their value lowered by α. This forces a quicker rotation in the choice of the parts to update.

Linear block selection

First of all, one SB per MB is selected through a linear order. Indeed, a linear order brings some stability to the background modelling which lacked in [70] as it ensures that after a maximum of T_u frames, all blocks from the model would have been updated at least once. Figure 4.2 and Algorithm 4 present the process in details. From a frame t to the next frame $t+1$, the position index of the current SB $id^{SB}_{w,h}$ is saved. Since the process of all Mega Block follow the same process, we can store only a single position index for all SB. Once the selection of the SB is done, the corresponding parts of the current frame and the background model are loaded and the region of the background corresponding is then updated. The update equation processes each pixel μ_t of the background model region at time t considering the corresponding pixel in the current frame I^i_t and the learning rate ρ:

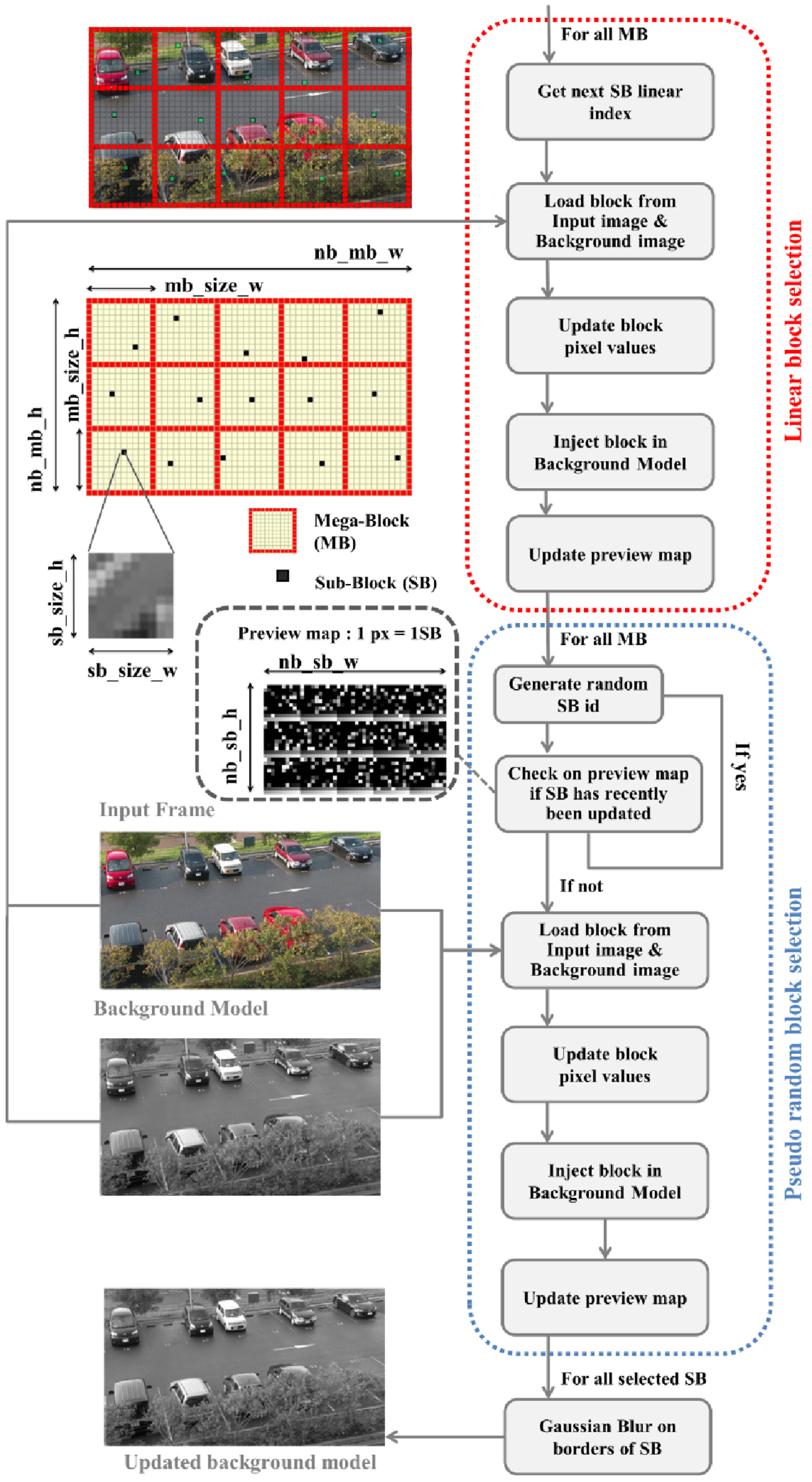


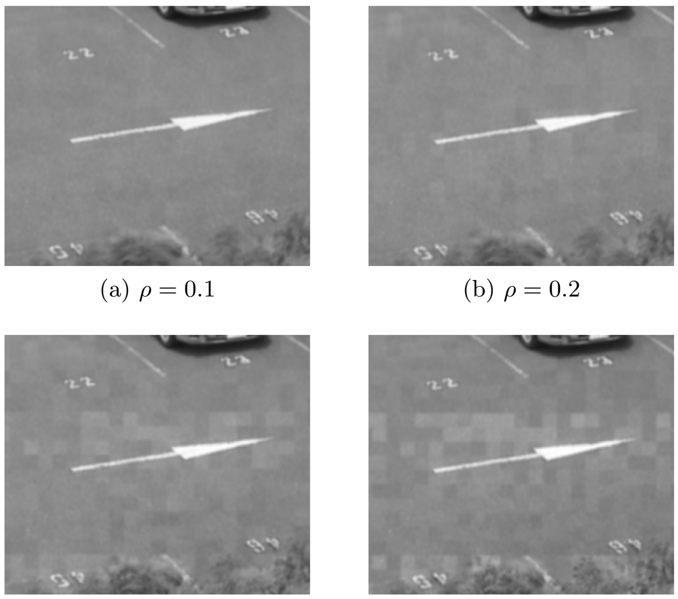
Figure 4.2: Mixed order block selection and background update.

Algorithm 4 Mixed Block Background Modelling process

```
1: Input: Input image $I_t$, preview_map image P, Background image B, MegaBlocks MB; SubBlocks SB; index Current LinearSubBlock $id_{SB}(x, y)$
2: Linear Block Selection:
3: for all $MB$ do
4:     Get the next index of SB to process $id_{SB}(x, y)$
5:     Load the corresponding SB from the input image and the background image
6:     Update the values of block pixels : $\mu_t^i = \rho I_t^i + (1 - \rho)\mu_{t-1}^i$
7:     for all Selected $SB$ do
8:         $P(id_x, id_y) = 255$
9:     end for
10: end for
11: Pseudo Random Block Selection:
12: for all $MB$ do
13:     repeat
14:         Generate random SB index $id(x, y)$
15:     until pixel value $P(id_x, id_y) > \tau$
16:     Load the corresponding SB from the input image and the background image
17:     Update the values of block pixels : $\mu_t^i = \rho I_t^i + (1 - \rho)\mu_{t-1}^i$
18: end for
19: for all Selected $SB$ do
20:     Gaussian blur $r = 2$ on the border pixels of the SB in the background model B
21:     $P(id_x, id_y) = 255$
22: end for
23: for all pixels from the preview_map P(x,y) do
24:     $P(x, y) = P(x, y) - \alpha$
25: end for
```

$$\mu_t^i = \rho I_t^i + (1 - \rho)\mu_{t-1}^i. \tag{4.3}$$

The value of ρ corresponds to the learning rate usually indicated in other literature and the common value used is $\rho = 0.1$. In the case of a low learning rate ($\rho \leq 0.1$), the new elements of the scene are incorporate slowly as all the blocks get updated. In the case a higher learning rate (>0.1) is used, temporal differences between blocks start to appear (Figure 4.3). Those differences are to be avoided at all cost since they break the homogeneity of the background model. That is the reason why there is a condition on the speed of the light variation. Indeed, in the cases of a sudden light modification a higher should be required but it would assuredly



(a) $\rho = 0.1$ (b) $\rho = 0.2$

(c) $\rho = 0.5$ (d) $\rho = 0.7$

Figure 4.3: Influence of the learning parameter ρ on the background modeling. The higher the rate is, the more temporal differences between adjacent blocks can be visible.

break the homogeneity of the model. The number of blocks per MB T_u also influences the speed of the update as well as the number *nb_sb_selec* of selected SB per MB modify the number of pixels to update at each frame. In other words, the higher T_u is the longer it will take to update the model but if more SB are selected per MB, then more parts of the background get updated. It becomes a trade-off between the total area processed at each frame and the update frequency. If too many blocks are processed, then the situation will be similar to a full background modelling.

Random block selection

Processing blocks exclusively through a linear order has a latent drawback. Indeed, the last block updated would always be right next to a block which has not been updated for T_u frames thus creating a moving temporal band. By adding a pseudo random selection after the linear selection, a second block per Mega Block can be updated as well in order to maintain a homogeneous background model (Figure 4.2, Algorithm 4).

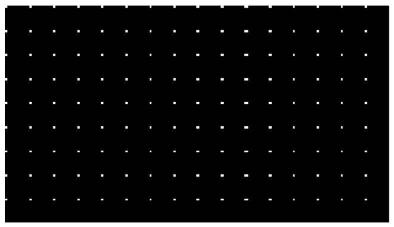
A pre-selection is done by a uniformed random generation of block ID $S_{id}(x, y)$ for each MB. The ID $S_{id}(x, y)$ is generated from the sub block coordinates inside its mega-block as:

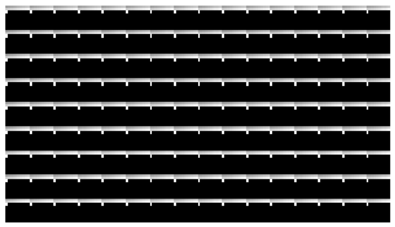
$$S_{id}(x, y) \in ([0; mb\_size\_w[; [0; mb\_size\_h[) \,. \tag{4.4}$$

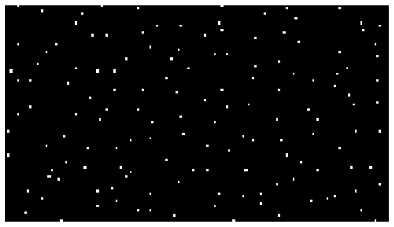
Then, an eligibility verification on the pre-selected block is done by checking on the *preview map* the pixel corresponding to the generated block id. The block is eligible (selectable for an update) only if the pixel has a value below the threshold τ which defines the minimum time before a block can be re-updated. In the case the block is not eligible, another block ID is generated and checked until an eligible block is found. The *preview map* is an observable tool to easily check which block have been updated recently. It consists of a $nb\_sb\_w \times nb\_sb\_h$ grey scale image which pixels correspond to each one of the Sub-Blocks. A pixel value is set to 255 right after that the corresponding block has been updated. At the end of the modelling, all pixels of the *preview map* get their value decreased by a fixed value α to simulate the time passing. The higher α is, the faster blocks will be considered as *old*.

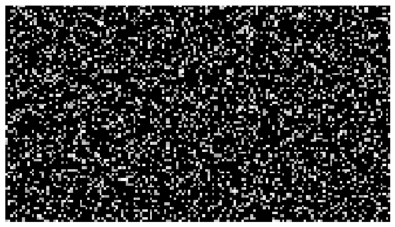
Figure 4.4 shows the different behaviors of the selection order on the *preview map* between strictly linear, strictly pseudo random and the proposed mixed order. By itself, the random selection suffers from the drawback of pure randomness, there would be no guaranty that all blocks get updated in a limited number of frames. It would thus create temporal differences between adjacent blocks. Using both linear and random order preserves the homogeneity of the background model.

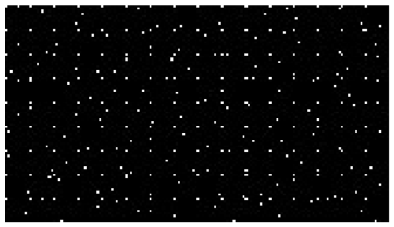
Finally, the selected blocks are updated and in order to reduce possible border effects from the block updating, a Gaussian blur is effectuated on the border between the selected blocks and their direct neighbours.


(a) Linear Order, t = 0


(b) Linear Order, t = 20


(c) Random Order, t = 0


(d) Random Order, t = 20


(e) Mixed Order, t = 0

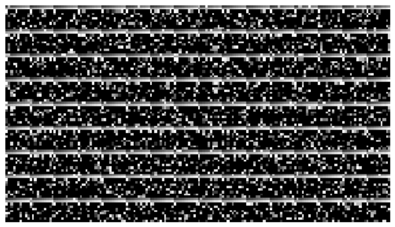

(f) Mixed Order, t = 20

Figure 4.4: Representation of the preview map at frame 0 and frame 20 with 16×9 MB and 10×10 SB/MB. The intensity of the white pixels represent the time when the selected blocks have been updated.

4.3 Background Subtraction

The background subtraction part is done by the Adaptive Block Propagative Background Subtraction (ABPBGS) seen in the previous chapter. The background model is then updated after each object detection with the MBBM. We have seen in the previous chapter how efficient the ABPBGS was to not only deal with various size objects but also to handle background noise in 4K UHD videos. is a very efficient background subtraction method for ultra high definition videos since it processes only parts where objects of interest are located independently of their size or shape.

Even though both MBBM and the ABPBGS divide the image in blocks to process a smaller part of the frame, their respective block numbers and size are not related in any way. The two process are totally independent from each other. The principal reason for this is that the blocks of the ABPBGS can change their size and shape through time depending on the foreground object detected while the number and dimensions of the Mega-Blocks and Sub-Blocks of the MBBM is fixed from the beginning. It can for example lead to a full frame detection which is the situation we want to avoid as much as possible.

4.4 Experimental results

Our proposed MBBM has been tested with our custom 4K dataset [64] composed of 12 4K videos of various length, from 10 seconds to 40 minutes with a framerate of 60 fps. The total dataset represents a total of about 50 minutes and 178631 frames. The dataset is organized in 4 categories: *Small Objects* (street1, street2, corridor, laser), *Big Objects* (screws, circle, 80percent), *Monitoring* (crossing, field) and *Illumination* (illu, parking, parking2). The *Small Objects* and *Big Objects* focus on respectively very tiny object (down to 0.01% of the image) and very big objects (up to the full size of the image). The *Monitoring* contains various situations as we could expect from surveillance videos and finally *Illumination* puts the accent on videos containing big modifications of the illumination and also contains stationary objects. Most of the comparisons have been done with methods presented in [52] and available in the BGSLibrary [51]. We selected 8 methods : PBAS [34], ZivkovicAGMM [58], MultiLayerBGS [59], DPAdaptiveMedian [53], AdaptiveSelectiveBGLearning, DPWrenGA [55], DPPratiMediod [54] and the IndependentMultimodal [65].
The PC used is a Quadcore i7@2.83GHz with 16 GB of RAM and the parameters of our methods are $n\_mb\_w = 16$, $n\_mb\_h = 9$, $mb\_size\_w = 15$, $mb\_size\_h = 15$, $\rho = 0.1$, $\alpha = 10$. Results from [67] showed that the background modelling part could be done at an average speed of 76.9 fps which means that the total speed of the detection would depend mainly on the foreground detection algorithm.

4.4.1 Quality comparisons

The goal of this present work is to process UHD videos and it has been solely designed for that scale. Therefore, it is necessary to present the true results of our proposed method on the 4K scale. Processing all the videos at the original 4K scale for each one of the state-of-art method would have required years of non-stop process making a direct comparison at the 4K scale impractical [67]. Therefore, the quality metrics for those state-of-the-art methods are based on their performances on 270p versions of the dataset. Despite the fact that the proposed method is for UHD videos only, we present the results of both 4K and 270p for the proposed MBBM.

Table 4.1 gathers all the quality comparisons sorted by category of video: *Small Objects*, *Big Objects*, *Monitoring*, *Illumination* and finally *General* which is the mean of the scores for all 12 videos. The comparisons are based on the Recall, the Precision, the F-Measure, the Similarity, the SSIM (Structural SIMilarity) introduced in [66] and finally the A-Score as defined in [J2]. The Recall, Precision, F-Measure and Similarity are based on the count of the number of foreground pixels classified as foreground also called True Positive (TP), the number of background pixels classified as background or True Negative (TN), the number of False Positive (FP) which are background pixels classified as foreground and the number of foreground pixels classified as background or False Negative (FN). The recall represents the capability to detect correctly parts of the objects of interest as foreground. However, the calculation ($Recall = TP/(TP + FN)$) cannot be trusted alone as it does not take into account the number of false positives. Same with the Precision ($= TP/(TP + FP)$) which indicates if a method is able to avoid detecting background noises from background movement or modification of illumination. The F-Measure ($= 2\dot{(}Rec.\dot{P}rec.)/(Rec. + Prec.)$) and the Similarity ($= TP/(TP + FN + FP)$) are measures which take into account both aspects. Finally, the A-Score represents the general ability to correctly detect objects.

Small objects category

For small objects, the proposed MBBM takes a clear lead compared to the other methods. Indeed, it is vital to avoid detecting background noises to be able to distinguish small objects. With a precision of 0.613, the MBBM is better than the ABPBGS (+23%) and much higher than the other state-of-the-art methods (+61% to +402%). Its F-Measure score is also the best with 0.441 which is between +46% to +521% better than the usual state-of-the-art methods which show here that they are clearly not suitable to detect very tiny objects and at the same time the limitation of using very small definition for foreground detection. The A-Score of the MBBM for the *Small Objects* category is about 0.589 which is between +4% to +64% better than all the other methods. The Precision of the 270p scale of the MBBM shows the main interest of using the original scale instead of downscaling. With 0.283, the precision of the 270p scale drops by -54% compared to the 4K scale.

Big objects category

The category *Big Objects* is here only to show that the method can deal with all sorts of content, from tiny objects to objects taking all the image. As expected, the objects being very big, state-of-the-art methods have less trouble to detect them. Therefore, the scores are much closer. With an A-Score of 0.590, the MBBM is about +4% to +21% better than the rest.

Monitoring category

The *Monitoring* category contains videos with multiple objects and some background movements. For this type of video, we can observe that there seems to be two groups of methods as half of them have A-Scores above 0.6 and the rest is below 0.5. With a score of 0.652, the MBBM is the second best method after the PBAS (0.679). What makes the difference between the two groups is their ability to avoid detecting the background movement. Indeed, the top half methods have precision measures about the double compared to the bottom

Table 4.1: Quality comparison of the different methods. Best scores are in bold.

Cate.	Method ID	Size	Recall	Pre.	F-Meas.	Sim.	SSIM	A-Score	fps
Small Objects	MBBM (Proposal)	4K	0.385	**0.613**	**0.441**	**0.329**	**0.997**	**0.589**	4.02
	MBBM	270p	0.325	0.283	0.279	0.207	0.967	0.484	**55.52**
	ABPBGS [J2]	4K	0.371	0.498	0.401	0.299	0.995	0.565	5.23
	BPBGS [63]	4K	0.341	0.491	0.372	0.280	0.970	0.541	5.31
	PBAS [34]	270p	0.223	0.122	0.133	0.096	0.984	0.404	1.80
	MultiLayer [59]	270p	0.070	0.125	0.071	0.050	0.993	0.371	2.71
	ZivkovicAGMM [58]	270p	**0.571**	0.303	0.332	0.253	0.979	0.522	13.30
	AdaptiveMedian [53]	270p	0.315	0.374	0.250	0.179	0.985	0.471	17.26
	PratiMediod [54]	270p	0.407	0.289	0.266	0.192	0.986	0.481	4.42
	WrenGA [55]	270p	0.538	0.227	0.249	0.175	0.981	0.468	11.90
	AdaptiveSelectiveBGLearning	270p	0.186	0.123	0.084	0.063	0.928	0.358	17.77
	IndependentMultimodal [65]	270p	0.385	0.103	0.129	0.089	0.984	0.401	3.16
Big Objects	MBBM (Proposal)	4K	0.483	0.552	**0.506**	0.412	**0.853**	**0.590**	2.30
	MBBM	270p	0.495	0.524	0.498	0.409	0.821	0.576	**20.47**
	ABPBGS [J2]	4K	0.448	0.522	0.472	0.387	0.778	0.546	2.78
	BPBGS [63]	4K	0.443	0.515	0.466	0.382	0.777	0.542	2.39
	PBAS [34]	270p	0.473	0.544	0.460	0.387	0.798	0.548	1.68
	MultiLayer [59]	270p	0.308	0.568	0.367	0.284	0.822	0.491	2.61
	ZivkovicAGMM [58]	270p	0.540	0.520	0.499	**0.416**	0.793	0.569	6.40
	AdaptiveMedian [53]	270p	0.327	**0.593**	0.395	0.296	0.842	0.511	9.97
	PratiMediod [54]	270p	0.403	0.456	0.380	0.289	0.796	0.488	2.54
	WrenGA [55]	270p	0.471	0.455	0.421	0.333	0.803	0.519	5.10
	AdaptiveSelectiveBGLearning	270p	0.530	0.501	0.489	0.399	0.800	0.563	16.58
	IndependentMultimodal [65]	270p	**0.557**	0.416	0.421	0.332	0.747	0.500	0.44
Monitoring	MBBM (Proposal)	4K	0.518	0.562	0.529	0.434	0.992	0.652	4.51
	MBBM	270p	0.462	0.548	0.492	0.408	0.981	0.627	**73.15**
	ABPBGS [J2]	4K	0.466	0.606	0.515	0.423	**0.995**	0.644	4.59
	BPBGS [63]	4K	0.406	**0.629**	0.485	0.389	0.993	0.622	3.35
	PBAS [34]	270p	0.575	0.584	**0.568**	**0.483**	0.985	**0.679**	2.04
	MultiLayer [59]	270p	0.472	0.520	0.472	0.372	0.987	0.610	3.12
	ZivkovicAGMM [58]	270p	**0.682**	0.241	0.280	0.225	0.755	0.420	6.42
	AdaptiveMedian [53]	270p	0.518	0.313	0.279	0.204	0.915	0.466	14.69
	PratiMediod [54]	270p	0.659	0.240	0.276	0.217	0.828	0.441	2.46
	WrenGA [55]	270p	0.672	0.217	0.263	0.202	0.803	0.423	6.18
	AdaptiveSelectiveBGLearning	270p	0.620	0.304	0.320	0.252	0.915	0.495	8.56
	IndependentMultimodal [65]	270p	0.660	0.139	0.197	0.135	0.862	0.398	2.03
Illumination	MBBM (Proposal)	4K	0.398	0.595	0.430	0.317	0.959	0.536	4.48
	MBBM	270p	0.474	0.445	0.379	0.284	0.866	0.510	**34.09**
	ABPBGS [J2]	4K	0.348	0.276	0.215	0.146	0.891	0.417	1.13
	BPBGS [63]	4K	0.372	0.259	0.175	0.115	0.804	0.365	1.18
	PBAS [34]	270p	0.622	0.510	0.543	0.450	0.960	0.651	2.74
	MultiLayer [59]	270p	0.566	**0.620**	**0.571**	**0.489**	**0.967**	**0.676**	3.67
	ZivkovicAGMM [58]	270p	0.563	0.165	0.189	0.123	0.917	0.410	2.73
	AdaptiveMedian [53]	270p	0.429	0.229	0.224	0.143	0.951	0.439	16.56
	PratiMediod [54]	270p	0.578	0.179	0.170	0.105	0.941	0.405	3.64
	WrenGA [55]	270p	0.634	0.118	0.146	0.091	0.893	0.377	1.83
	AdaptiveSelectiveBGLearning	270p	0.481	0.227	0.248	0.174	0.890	0.474	15.53
	IndependentMultimodal [65]	270p	**0.702**	0.085	0.115	0.077	0.820	0.337	0.65
General	MBBM (Proposal)	4K	0.435	**0.585**	**0.469**	**0.364**	**0.950**	**0.597**	3.78
	MBBM	270p	0.429	0.431	0.396	0.311	0.907	0.538	**44.34**
	ABPBGS [J2]	4K	0.400	0.466	0.391	0.304	0.915	0.536	3.57
	BPBGS [63]	4K	0.379	0.466	0.375	0.290	0.906	0.529	2.35
	PBAS [34]	270p	0.444	0.401	0.390	0.322	0.932	0.548	2.05
	MultiLayer [59]	270p	0.320	0.425	0.337	0.272	0.943	0.517	2.99
	ZivkovicAGMM [58]	270p	**0.580**	0.313	0.330	0.257	0.880	0.489	7.79
	AdaptiveMedian [53]	270p	0.380	0.382	0.285	0.203	0.929	0.472	14.83
	PratiMediod [54]	270p	0.491	0.295	0.272	0.199	0.901	0.457	3.43
	WrenGA [55]	270p	0.568	0.255	0.269	0.198	0.885	0.451	6.73
	AdaptiveSelectiveBGLearning	270p	0.459	0.276	0.278	0.213	0.893	0.461	15.85
	IndependentMultimodal [65]	270p	0.553	0.183	0.210	0.155	0.863	0.409	1.66

methods.

Illumination category

The *Illumination* set of videos presents cases where the changes of light have great influence on the appearance of the background and where stationary objects require to be incorporated to the background model. Only methods which are able to handle light variations and the appearance or disappearance of stationary objects show decent scores. With a score of 0.536, the MBBM is clearly among them as only the PBAS (0.651) and the MultiLayer (0.676) present better results. This is expected as they both are which have been designed for that purpose. We can also observe with this category the real difference with the ABPBGS as the MBBM is able to handle the variation of light when the ABPBGS is not capable of doing it. The precision of the MBBM is much higher than the ABPBGS 0.276→0.595 (+116%) as well as its average score 0.417→0.536 (+29%). Unsurprisingly, the 270p scale of the MBBM performs better than the ABPBGS and the BPBGS as it can deal with the stationary objects and the slow light variation but it does not match the precision of the 4K MBBM (-25%) due to the fact that with bigger blocks (compared to the image size) it is easier for the detection propagation to catch background noise.

Conclusion

Finally, the *General* part of the table presents the scores for the full dataset. With an A-Score of 0.597, the MBBM is +9% to +46% better than the state-of-the-art methods. The remarkable part is its precision score. Indeed, with a precision of 0.585, the MBBM is +26% better than the ABPBGS and +46% better than the PBAS (0.401→0.585). This result shows that with 4K there are contents which are undetectable by some state-of-the-art methods. Even though those methods are very accurate in general for medium size or big objects, they become unfit for tiny objects as they confuse them with small background noise. Also, the MBBM has the best F-Measure score and is about +20% better than both ABPBGS and PBAS (respectively 0.391→0.469 and 0.390→0.469). These results show that the MBBM is

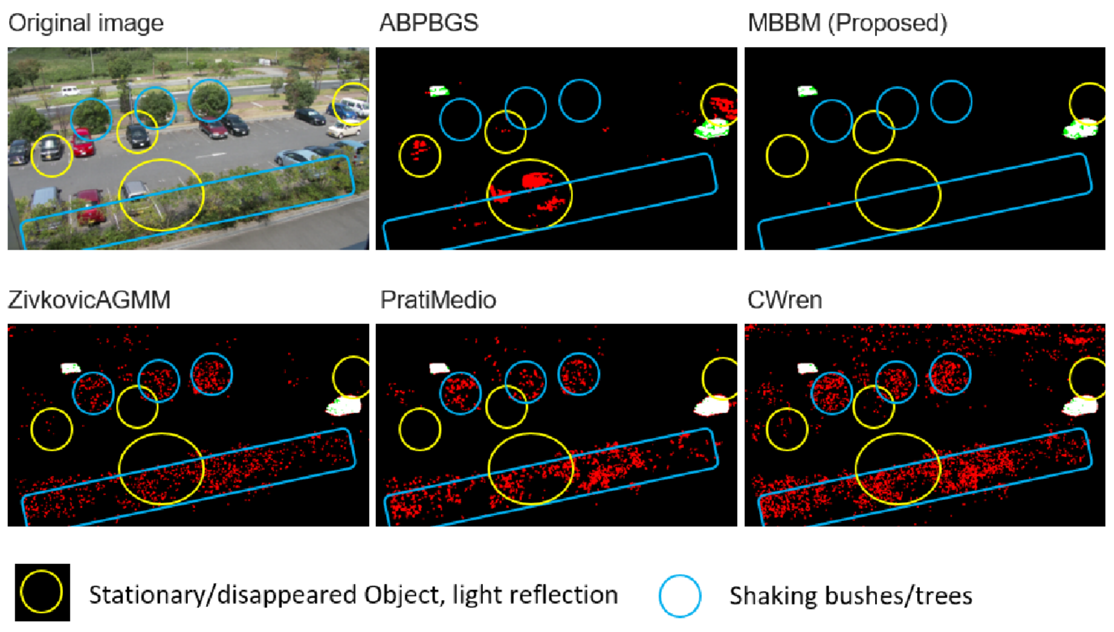


Figure 4.5: Comparison of the foreground masks of the *parking* sequence. Up: original image, ABPBGS, proposed MBBM; Bottom: ZivkovicAGMM, PratiMedio, CWren. In yellow circle figure the areas where stationary objects have appeared/left and where the light changed as well. In the blue circles figure areas where bushes and trees are moving with the wind. Red: FP; Green: FN; White; TP; Black: TN.

able to handle various cases from the detection of tiny objects to big object taking the whole image and under more difficult but rather common situation such as illumination variation stationary objects. Even though the MultiLayer and especially the PBAS are the most efficient methods on some categories of videos, they show their weakness when the content is not adapted to downscaling. The MBBM performs better on average than them because it is able to deal with more various situations.

4.4.2 Visual comparisons

Figures 4.9, 4.10, 4.11 and 4.12 present a sample of the foreground detection results of the different methods for the four categories.

On videos which focus on small objects (Figure 4.9), methods like the proposed MBBM or the ABPBGS show their superiority over the other methods. Methods such as the PBAS or the MultiLayer which have show some of the best results in average are completely lost when they have to detect tiny objects. Indeed, their strong ability to remove background noise becomes a drawback in such situations because the objects of interest get removed along with the false positives. On the contrary, methods able to detect the objects are usually unable to do with precision as a lot of background noise get detected as well.

The two groups of results for the *Monitoring* category is clearly shown in Figure 4.11. Indeed, we can observe that in half of the methods, a huge amount of false positives is detected, the moving grass being wrongly detected as foreground. The other half which includes the MBBM are able to reduce the amount of false positive. Moreover, the contours of the objects of interest are better defined with the MBBM on 4K than it is with methods using the 270p scale.

On the *Big Object* category, all methods perform very similarly as expected. Two methods though, MultiLayer and AdaptiveMedian, distinguish themselves by their ability to remove the shadows (Figure 4.10). Very big objects are not a problem for a propagative method like the MBBM but it is nonetheless not advisable to use such methods on videos where objects take all the frame. Indeed, the main interest of those methods is to reduce the processing time by processing as few parts of the image as possible so if all the image requires to be processed such methods are not appropriate.

The *parking* sequence, illustrated by Figure 4.12, shows the whole purpose of the present method compared to the ABPBGS. Figure 4.5 shows in more detail the differences between the MBBM, the ABPBGS and some of the state-of-the-art methods. With the ability of handling both stationary objects and modification of the illumination, the proposed MBBM does not wrongly detect the stationary cars nor the reflection of the light on the windshields (yellow circles). Notice that the state-of-the-art methods can also correctly adapt the background on those areas since almost no false positive appear in the yellow circle at the position of the light reflection and stationary cars. However, those methods suffer from different issues which are the shaking bushes and the leaves from the trees (blue circle and rectangle). Therefore, the main difference between the state-of-the-art methods and the proposed MBBM is not related to illumination handling but with temporary background noise which those methods are unable to process correctly. Methods with strong background noise removal like the PBAS and the MultiLayer show here their real strength.

In the *Illumination* category, the *illu* sequence shows the limitations of the proposed method. It is probably the most difficult case we can face. Indeed, the object of interest comes at the same speed as the light changes which makes it very difficult to separate the object from the background. In normal situations, the speed of the object is much higher than the variation of light. This case shows that the background model cannot be updated correctly when consecutive frames are very different to each other. The temporal difference between blocks disturb the detection and the system fails to detect correctly the foreground from the background. Almost no method can handle this sequence correctly. Even the MultiLayer which has almost no FP show that very few parts of the object can be detected correctly, most of it being considered as missed detection. The only method which is actually showing decent results is the PratiMedio which is efficient to handle low intensity shadows.

4.4.3 Speed and memory comparisons

Table 4.2 presents a comparison of the time per frame required to process the full dataset between the MBBM and an estimation of the detection time for the full 12 4K video dataset. The estimation has been made from the average time necessary to process at least the first 20 frames of the *parking_short* sequence with the exception of the MBBM which is based on the full process of the videos. We can observe that methods which have not been designed with the purpose of processing UHD videos have estimated time way too big for practical use. Indeed, the fastest method, the Independent Multimodal requires about 1.19 weeks to process the whole dataset. The other methods take from 23.04 weeks for the Adaptive Median to 21.18 years for the MultiLayer. Results from [67] showed that the background modelling part could be done at an average speed of 76.9 fps which means that the total speed of the detection would depend mainly on the foreground detection algorithm. However, we can observe

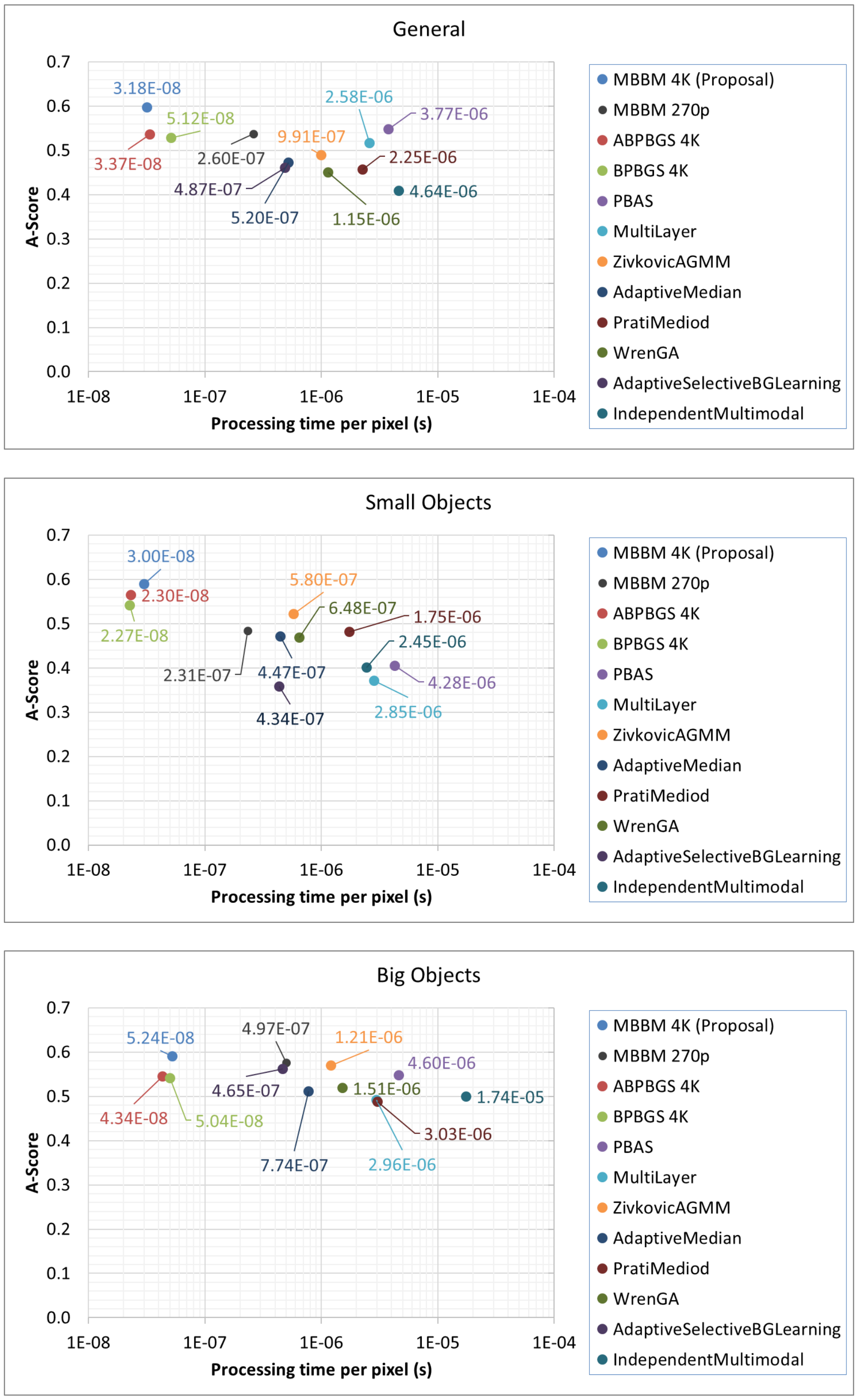
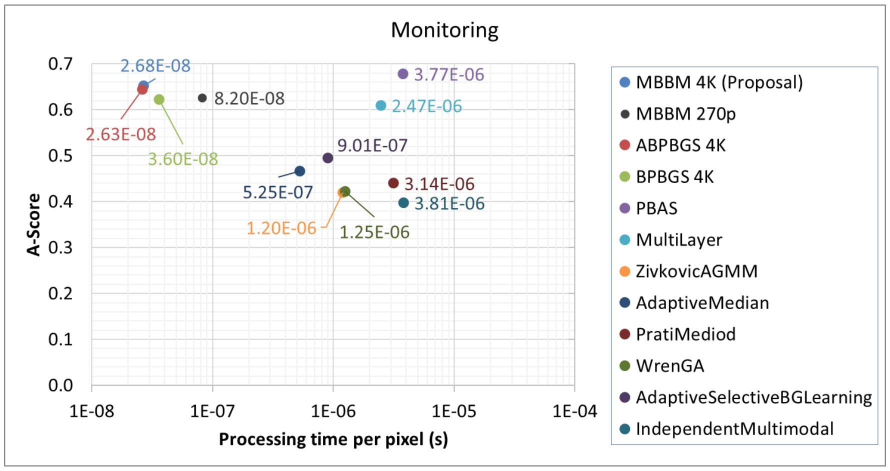


Figure 4.6: Comparison of the processing time per pixel and the average global quality score of the proposed MBBM to the state-of-the-art methods on each category of video.



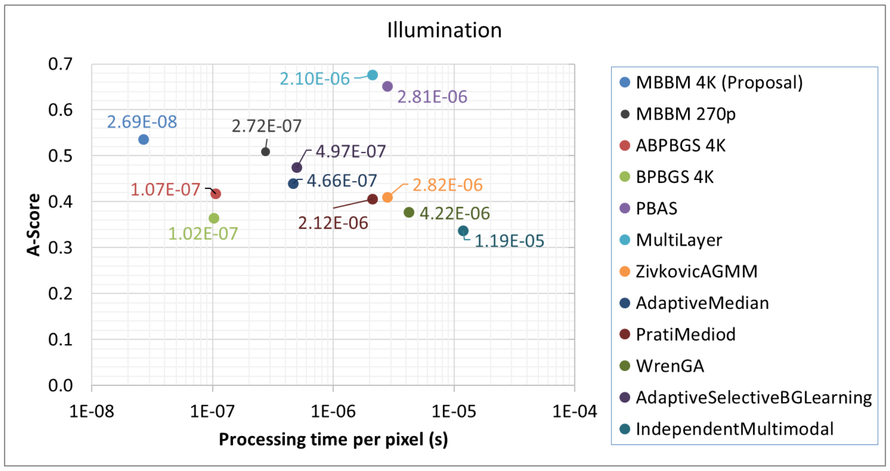


Figure 4.5: Comparison of the processing time per pixel and the average global quality score of the proposed MBBM to the state-of-the-art methods on each category of video.

here the huge improvement of the MBBM compared to the ABPBGS which would use a full frame processing as it is about 9 times faster (2.23days $\rightarrow$ 5.81 hours). Last, the MBBM is the only method which is less than 10 times slower than the total length of the dataset. Above that level, we can consider that a method is not suitable for UHD processing.

Table 4.2: Average speed per frame and estimated time to the 12 4K videos of the dataset based on the *parking2* sequence. The normal length of the dataset is about 50 min. long. Only methods which include a background modelling are compared.

Method	Time/frame (s)	Total Time (s)
MBBM	0.12	5.81 hours
PBAS	130.56	38.56 weeks
MultiLayer	3738.43	21.18 years
ZivkovicAGMM	1363.01	7.72 years
AdaptiveMedian	77.99	23.04 weeks
PratiMediod	292.40	1.66 years
WrenGA	1072.27	6.07 years
AdaptiveSelectiveBGLearning	299.83	1.70 years
IndependentMultimodal	4.02	1.19 weeks

Figure 4.6 shows the A-Score and the processing time per pixel (PTPP) for each category of video and the average for all the dataset. The PTPP is used to compare algorithms' speed on the same base even if they processed at different scales. In general, the BPBGS-based methods perform much faster than the other state-of-the-art methods. The proposed MBBM has a PTPP of 3.18×10^{-8} s/p. If it is just faster than the ABPBGS 3.37×10^{-8} s/p (-6%), it is about a hundred times faster than the best state-of-art method, the PBAS (3.77×10^{-6} s/p).

The *Small Objects* category shows higher speed in general but lower A-Scores. The fact that some objects are missed leads to less objects to extract and thus to a faster detection.

The *Big Objects* category shows more spread speed but higher scores. Indeed, it is easier to detect big objects but they can generate a lot of fragments which can possibly slow the detection.

In the *Monitoring* category two groups exist if we only consider the score. However, if we also consider the speed, another emerge with the PBAS and the MultiLayer which perform very well but quite slower than the other methods.

This group is even more visible with the *Illumination* set of videos. Indeed, they appear way above the other methods on a quality level. The MBBM also distinguishes itself from the others with a processing time per pixel of 2.69×10^{-8} s/p. It is much faster than the ABPBGS $1.07\times10^{-7}\rightarrow2.69\times10^{-8}$ s/p (-74%). Once again, the number of detected objects has a great influence on the speed of a method. In that case, the ABPBGS suffers from the light variation thus detects much more false positives which lowers its detection speed.
Besides from the *Small Objects* category, the 270p scale of the MBBM logically follows the same behavior as the 4K MBBM. The 4K MBBM is remains faster due to the fact that is can use tiny blocks (compared to scale) to reduce the process areas.

In figure 4.6 we compared the memory consumption of the MBBM, the ABPBGS and the three best methods: PBAS, DPZivkovicAGMM and MultiLayer. The state-of-the-art methods are not able to process the videos at their original scale so the tests have been done on the first 50 frames. The second thing to mention is that the MultiLayer memory requirements for the 8K scale is so big that our 16 GB were not enough, the program crashed because of a lack of memory. For the other state-of-the-art methods, we can observe that the memory consumption is increasing exponentially with the proportional increase of the dimensions of the video. In comparison, the MBBM's memory requirement is very low with a maximum of about 450 MB for the 8K resolution. Moreover, since the MBBM uses very small temporary images corresponding to the blocks, it does not add more memory consumption to what the ABPBGS requires.

4.4.4 Example of 8K tracking

This work has been the result of a long work with the objective to reduce the computational time of a 8K UHD tracking system so we could get closer to real-time processing than with state-of-the-art methods. Therefore, we present here a last UHD test with the use of the MBBM, which evolved from the ABPBGS, and the RTRM together in order to track objects in the field 8K video. The RTRM and the MBBM have in common that they improve their

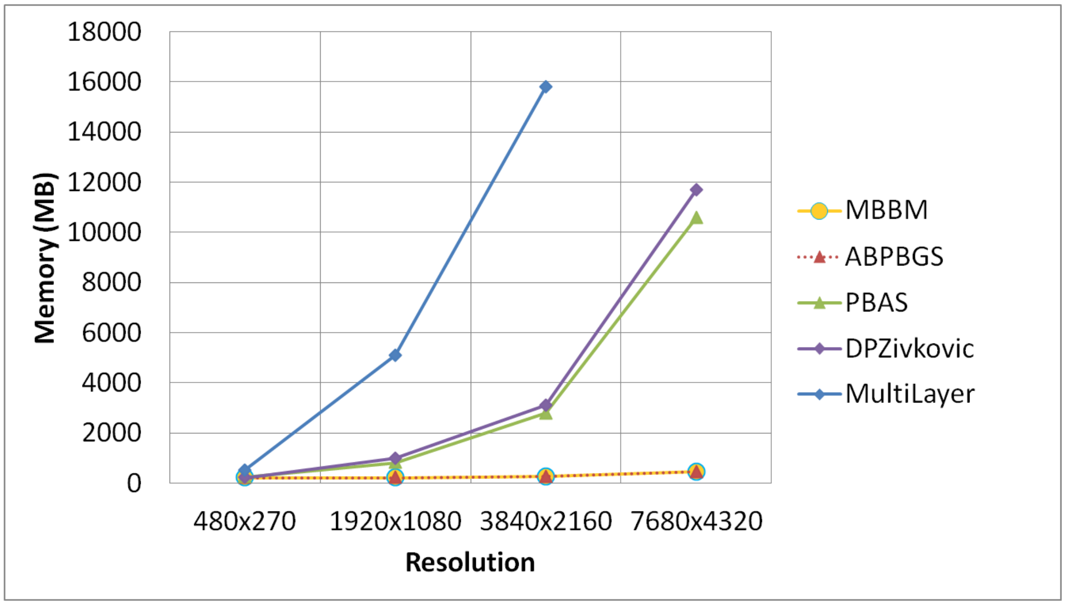


Figure 4.6: Comparison of the memory consumption depending on the resolution used, from 270p to 8K UHD. Except for the ABPBGS based methods (in orange and red), the results for resolutions above 1080p are estimations of the consumption from the 20-50 first frames since the algorithms were unreasonably slow to process the full sequence.

results by using the information from the previous frame. Ideally, there are one seed for each object to detect and they are located on a part of each object. By using the tracking results of the RTRM as seed point for the MBBM, we get as close to the ideal case as we can get. Indeed the tracking results are supposed to represent the position and shape of the target. In this last experiment, we combined both RTRM and MBBM in such a way. In the Table 4.3 we can observe that the MBBM achieves with the RTRM much better results than the BGS since it improves the precision to reach a N-MOP score of 0.50→0.70 (+40%). It means that the detected objects have a very similar size compared to the ground truth objects. Figure 4.8a shows the evolution of object count for the sequence. We can see in red that the original method does not create much extra blobs. The accuracy is also better with almost perfect scores N-MODA of 0.99 and N-MOTA of 0.98.

Figure 4.7 presents an example of the use of the RTRM with the MBBM on the 8K Field sequence. The raw detection shows the unprocessed areas in grey as well as the object boundaries. Unlike other methods, the boundary is quite large because a fusion of objects is

Table 4.3: Comparison of tracking performances of the MBBM and the BGS, both used with the RTRM. Best results in bold.

Sequence	Method	Occl.	MODP	MOTP	MODA	MOTA	IDS	miss	FP
field 8K	BSG+RTRM	10	0.50	0.50	0.98	0.92	19	10	0
	MBBM+RTRM	41	**0.70**	**0.70**	**0.99**	**0.99**	9	1	0

effectuated automatically during the process of the MBBM. The refinement of the fused blob gives two correct blobs. The size and shape is not accurate for all frames because the RTRM forces the dimensions to not change too much between two frames.

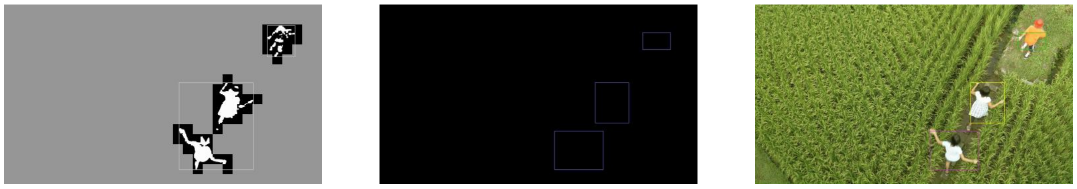
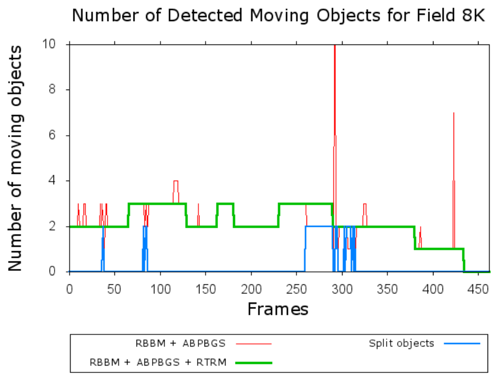
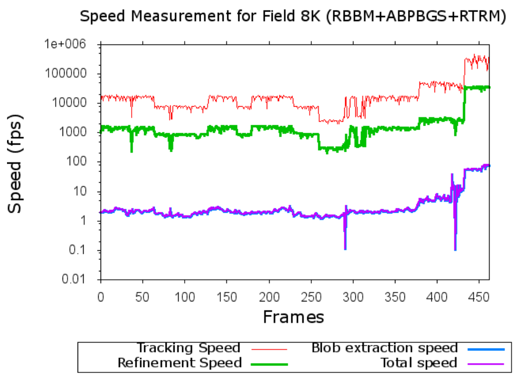

(a) Raw detection (b) After RTRM (c) Tracking track

Figure 4.7: Example of the RTRM for an 8K UHD sequence (Nebuta Festival, NHK) by the MBBM foreground detection method.



(a) Object count



(b) Speed

Figure 4.8: Object count and Speed for the MBBM

The most important improvement is the reduction of the processing time required to detect and track objects in the 8K UHD video. Table 4.4 shows a comparison of the BGS and the MBBM for that sequence. The refinement and tracking speed are very similar since they are mostly dependent of the number of occlusions. What is very different is the total speed of about 6.5 fps for the MBBM. It represents an improvement of about 11232% compared to the BGS method. It is also a significant progress than the MBBM average speed for 4K scale seen in Table 4.1 due to the RTRM which insures that all the seeds belong to an object of interest and not to background noise. Less propagation leads to less false positives which leads to a faster label process and thus to an even faster detection. All the works presented in the previous chapters have led to not only better quality results but also a processing speed which is quite close to real time. The processing of 8K UHD scenes which was very improbable in Chapter 2 due to a very slow detection part is now possible by using a foreground detection system designed for such enormous resolution.

Table 4.4: Comparison of the average speed performances for the BGS and the MBBM on the 8K sequence (in fps).

Sequence	Size	Method	Detect. (fps)	Refin.(fps)	Track.(fps)	Total(fps)
field 8K	7680×4320	BGS+RTRM	0.057	2223	35594	0.057
		MBBM+RTRM	6.459	2486	34942	6.459

4.5 Conclusion

This chapter presented the Mixed Block Background Modelling method which is a fast background modelling method designed for UHD videos. Instead of updating the full background model at once, the model is updated through time block per block. This spatiotemporal approach selects blocks by a linear and a pseudo-random order. The linear order makes sure that every block will be updated and the pseudo-random order allows more blocks to be updated in order to keep the homogeneity of the background model. The method is combined with a foreground detection designed for UHD videos such as the Adaptive Block-Propagative Background Subtraction method. Experimental results show that despite the relative simplicity of the proposed MBBM the foreground detection can compete with efficient state-of-the-art methods with an average A-Score of 0.597. Moreover, the processing time per pixel of the MBBM is the lowest of all compared methods with an average of 3.18×10^{-8} s/p. An improvement of the MBBM is possible but it would require trades-off. It would likely complexify it and slow it down. Finally, by processing few tiny blocks, the MBBM does not require more memory than the ABPBGS with an average 450 MB for an 8K video.

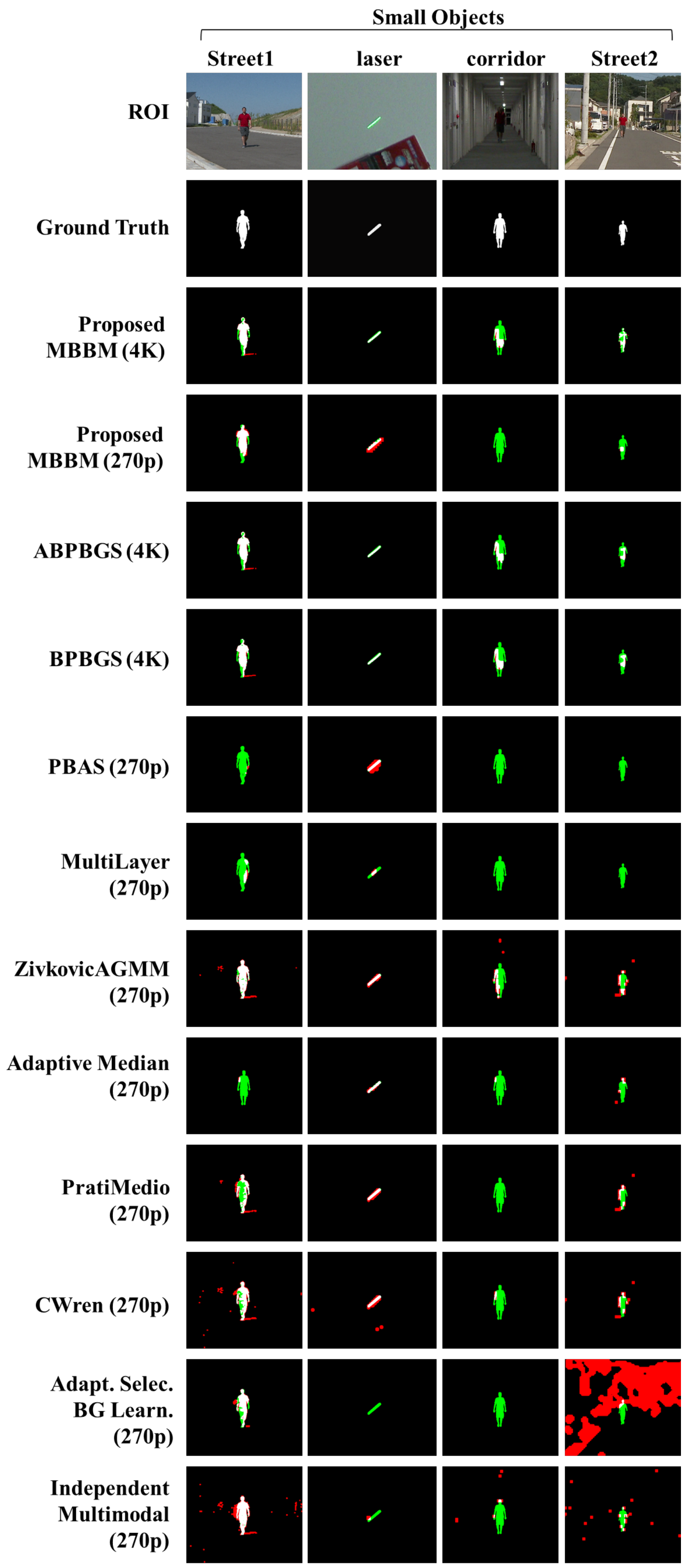


Figure 4.9: Visual comparisons of the proposed method to the state-of-the-art methods on videos for the *Small Objects* category. From top to bottom: Original 4K image/ROI, ground truth image, proposed MBBM, ABPBGS, BPBGS, PBAS, MultiLayer, ZivkovicAGMM, AdaptiveMedian, PratiMediod, WrenGA, AdaptiveSelectiveBGLearning, IndependentMultimodal. Color legend: Black-TN, White-TP, Red-FP, Green-FN.

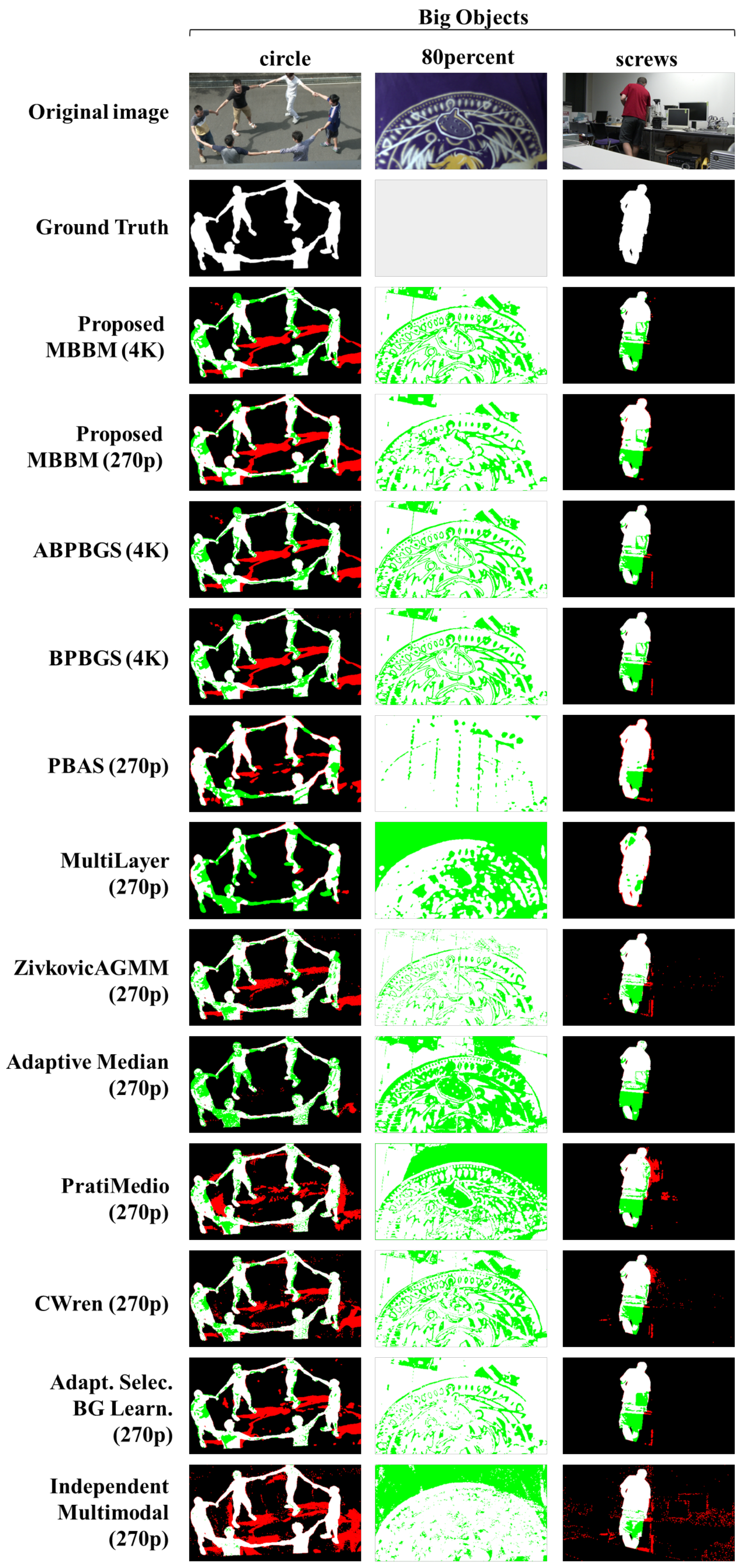


Figure 4.10: Visual comparisons of the proposed method to the state-of-the-art methods on videos for the *Big Objects* category. From top to bottom: Original 4K image/ROI, ground truth image, proposed MBBM, ABPBGS, BPBGS, PBAS, MultiLayer, ZivkovicAGMM, AdaptiveMedian, PratiMediod, WrenGA, AdaptiveSelectiveBGLearning, IndependentMultimodal. Color legend: Black-TN, White-TP, Red-FP, Green-FN.

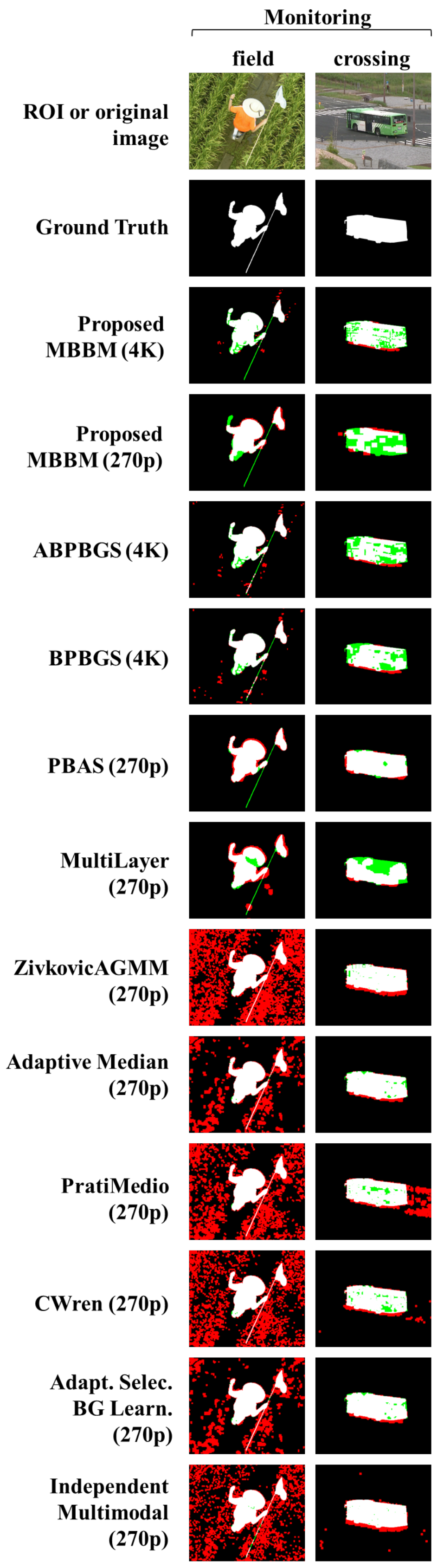


Figure 4.11: Visual comparisons of the proposed method to the state-of-the-art methods on videos for the *Monitoring* category. From top to bottom: Original 4K image/ROI, ground truth image, proposed MBBM, ABPBGS, BPBGS, PBAS, MultiLayer, ZivkovicAGMM, AdaptiveMedian, PratiMediod, WrenGA, AdaptiveSelectiveBGLearning, IndependentMultimodal. Color legend: Black-TN, White-TP, Red-FP, Green-FN.

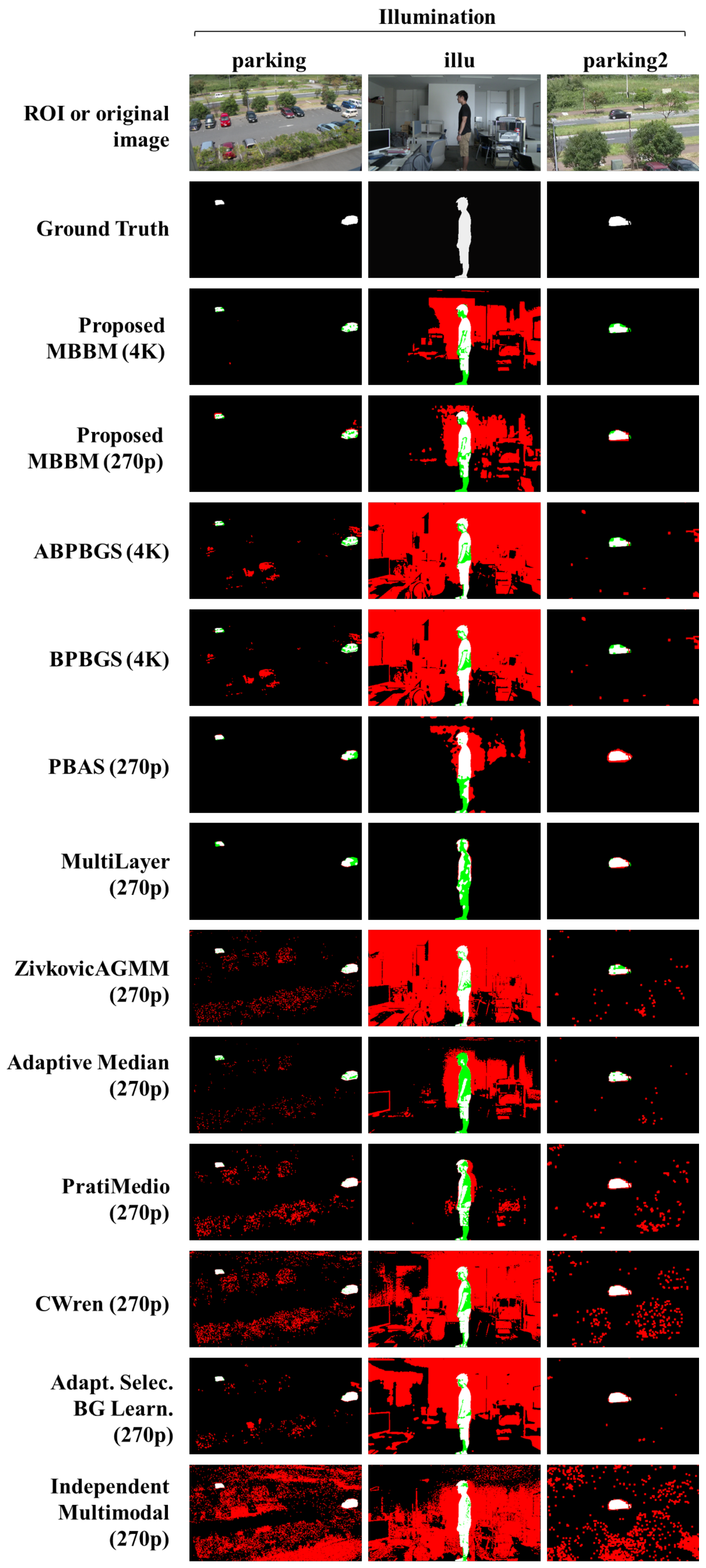


Figure 4.12: Visual comparisons of the proposed method to the state-of-the-art methods on videos for the *Illumination* category. From top to bottom: Original 4K image/ROI, ground truth image, proposed MBBM, ABPBGS, BPBGS, PBAS, MultiLayer, ZivkovicAGMM, AdaptiveMedian, PratiMediod, WrenGA, AdaptiveSelectiveBGLearning, IndependentMultimodal. Color legend: Black-TN, White-TP, Red-FP, Green-FN.

Chapter 5

Conclusion

5.1 Summary of the different works

The first contribution was the Real-Time Refinement Method for Moving Object Detectors (RTRM), a resolution free moving object filter which refines the moving object detection results obtained after a foreground detection. Its aim was to improve the detection quality through, among other things, the reduction of the number of false positives. The RTRM used the position in the previous frame of the trackers to filter the detected objects and remove unnecessary objects while keeping the interesting ones. In the case of occlusions, fusions of objects can happen. The fused object would be split into as many new objects as the number of trackers overlapping it. Then the new objects were moved for a better space occupancy of the fused object. With one detected object per tracker, the matching could be done by a Hungarian algorithm on the distance between objects and the previous position of the tracker.

The RTRM had been tested on our HD sequences and one 8K UHD sequence from the NHK Nebuta festival. The results showed a great reduction of the background noise, especially for the 8K sequence with 100% less false positives with the RTRM (2634$\rightarrow$0). The experiments also showed that the number of objects after the RTRM matches the number of trackers under conditions such as moving background, occlusion between two trackers, or both. As a consequence, the RTRM greatly improved all the results compared to the SBGS:

-41.2→0.98 for the N-MODA, -42.9→0.92 for the N-MOTA and 0.10→0.50 (+400%) for the N-MO(D/T)P. The RTRM average speed was 4.2×10^{-4} s and the tracking average speed was 2.7×10^{-5} s which made it suitable for real-time applications.

The second contribution was a foreground detector named Adapting Block-Propagative Background Subtraction (ABPBGS). The main idea was to skip all areas in the image in which there are no moving object. This is particularly interesting for UHD when the objects of interest can represent less than 0.012% of the total area. The novelty of this work leads in the detection which will detect and spread along the object as long as it detects a part of it. A block history map guaranties that each block is processed only once. It is a virtual grid containing the position of every block already and currently processed. Moreover, the detection loads and processes only small blocks saving computational time and memory usage. The spreading could also be parallelized to increase even more its speed.

Compared to the best state-of-the-art method, the PBAS, the ABPBGS reduced a lot the processing time per pixel $3.78 \times 10^{-6} \rightarrow 2.33 \times 10^{-8}$ s/p (-99.38%). For the 8K sequence, the average memory usage of the ABPBGS was greatly diminished compared to the PBAS 11 GB→450 MB (-96%) by skipping most parts of the image. The pixel-based detection quality metrics showed that the ABPBGS reduced the FP and FN with an average precision (between 0 and 1) improved 0.327→0.495 (+51%) as well as a better recall score 0.275→0.384 (+39%). Using the F-Measure, Similarity and SSIM measures (between 0 and 1) the ABPBGS achieved a better average quality score 0.493→0.572 (+16%) compared to the PBAS and 0.482→0.572 (+19%) compared to the AGMM.

The last contribution was the Mixed Block Background Modelling (MBBM), a method solely designed for UHD videos. The novelty of the MBBM lied in the reduction of the computing time by updating very small parts (blocks) of the background model instead of the full image at once. To keep the homogeneity of the model, two blocks per region MB were selected by a mixed selection: one through a linear order selection and one by a pseudo random selection. Then the background model parts corresponding to the selected blocks were updated. The MBBM was an amelioration of the ABPBGS in order to deal with various and difficult situations such as changes of illumination or stationary objects.

The tests effectuated on 12 videos showed that the MBBM was a major improvement of the ABPBGS. The MBBM had a better precision rate (between 0 and 1) than the ABPBGS 0.466→0.585 (+26%), the PBAS 0.401→0.585 (+46%) or the AGMM 0.313→0.585 (+87%). The MBBM could handle the various challenges with an average score of 0.597 using the F-Measure, Similarity and SSIM measures (between 0 and 1). The MBBM clearly outperformed the other methods: ABPBGS 0.536(+11%), PBAS 0.548(+9%), AGMM 0.489(+22%). Even though the MBBM updated the background model, its processing time per pixel was lower compared to the ABPBGS $3.37\times10^{-8}\rightarrow3.18\times10^{-8}$ s/p (-5.6%) or to the PBAS 3.77×10^{-6} $\rightarrow3.18\times10^{-8}$ s/p (-99.16%). A last tracking test on an 8K video using the MBBM combined with the RTRM resulted in an average total speed of 156 ms/frame (6.46 fps) with a N-MODA of 0.99 and a N-MOTA of 0.98.

5.2 Discussion and Future

The objective of this thesis was to improve the foreground detection systems for Ultra High Definition videos by making them much faster than existing methods while keeping a good quality in the results. Image understanding on UHD was unprecedented and it is just beginning of this particular topic. There is still a lot to discover about the properties of UHD videos and the different and new applications they can offer. Our work on the subject was an introduction on the possibilities both for a different use of UHD sequences and for the image processing of them. We presented solutions designed for UHD video which are far more efficient than the existing methods. The accomplishment of those works is a complete system which runs at 6.46 fps for 8K UHD while achieving better results and lowering the resources consumption by 24 compared to the state-of-the-arts detectors was the fruit of all of that. This speed is an incredible improvement, especially if we consider that it had been done only by using a software approach, meaning that even greater speeds, possibly real-time, could eventually be achieved with specific hardware optimization and design.

We sincerely believe that the next step to really make the research advance on this topic would be to create a complete 4K and 8K dataset. Those datasets could include various scenarios and scenes all completed with ground truth information, foreground masks and boundary boxes. An even more complete dataset could also have data for tracking purposes or face detection in order to become the main reference in the area. Concerning the different methods proposed, the first improvement should come from the background model which is vital for harsher situations such as shadows or reflections. Then the ABPBGS could also be improved with a better block seed selection and a replacement for the full frame detection. Improving such system to more complex is very tempting but any improvement must be done without jeopardizing the computational speed nor the memory required.

Bibliography

[1] T.P. Chen, H. Haussecker, A. Bovyrin, R. Belenov, K. Rodyushkin, A. Kuranov, and V. Eruhimov, "Computer vision workload analysis: Case study of video surveillance systems.," 2005.

[2] S.-C. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," p. 881892, 2004.

[3] V. Zeljkovic, D. Zhang, V. Valev, Z. Zhang, S. Zhu, and J. Li, "Personal access control system using moving object detection and face recognition," in *High Performance Computing Simulation (HPCS)*, 2014, pp. 662–669.

[4] D. Berjon, C. Cuevas, F. Moran, and N. Garcia, "Moving object detection strategy for augmented-reality applications in a gpgpu by using cuda," in *Consumer Electronics (ICCE) 2012 IEEE International Conference on*, Jan 2012, pp. 319–320.

[5] S.L. Dockstader and A.M. Tekalp, "Tracking multiple objects in the presence of articulated and occluded motion," in *Proc. Workshop Human Motion*, 2000, pp. 88–95.

[6] S. Ye, Y. Zhao, F. Zheng, and Z. Song, "A multi-features based particle filtering algorithm for robust and efficient object tracking," in *Multimedia and Signal Processing*, 2012, vol. 346, pp. 8–15.

[7] D. Berjon, C. Cuevas, F. Moran, and N. Garcia, "Region-based moving object detection using spatially conditioned nonparametric models in a gpu," in *International Conference on Consumer Electronics (ICCE)*. IEEE, 2014, pp. 359–360.

[8] R. Chao, G. Maca-Vzquez, E. Zalama, J. Gmez-Garca-Bermejo, and J-R Pern, "Automated tracking of drosophila specimens," *Sensors*, vol. 15, no. 8, pp. 19369–19392, 2015.

[9] K. Pauwels and M. M. Van Hulle, "Optic flow from unstable sequences through local velocity constancy maximization," in *Image and Vision Computing (The 17th British Machine Vision Conference (BMVC 2006))*, 2009, vol. 27, pp. 579–587.

[10] P.M. Jodoin and M. Mignotte, "Optical-flow based on an edge-avoidance procedure," *Computer Vision and Image Understanding*, vol. 113, no. 4, pp. 511–531, 2009.

[11] H. Meuel, M. Munderloh, M. Reso, and J. Ostermann, "Optical flow cluster filtering for roi coding," in *Picture Coding Symposium (PCS)*, Dec 2013, pp. 129–132.

[12] T. Bouwmans and F. El Bafand B. Vachon, "Background modeling using mixture of gaussians for foreground detection a survey.," pp. 219–237, 2008.

[13] Y. Benezeth, P.M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Review and evaluation of commonly-implemented background subtraction algorithms," pp. 1–4.

[14] Y-K Lai and C.-C.J. Kuo, "Perceptual image compression with wavelet transform," in *Circuits and Systems, 1998. ISCAS '98. Proceedings of the 1998 IEEE International Symposium on*, May 1998, vol. 4, pp. 29–32.

[15] M. Piccardi, "Background subtraction techniques: a review," in *IEEE International Conference on Systems, Man and Cybernetics*, Oct 2004, vol. 4, pp. 3099–3104.

[16] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," IEEE International Conference on Computer Vision and Pattern Recognition, 1999, vol. 2.

[17] F. El Baf, T. Bouwmans, and B. Vachon, "A fuzzy approach for background subtraction," in *15th IEEE International Conference on Image Processing (ICIP).*, Oct 2008, pp. 2648–2651.

[18] L. Maddalena and A. Petrosino, "A self-organizing approach to background subtraction for visual surveillance applications," *IEEE Transactions on Image Processing*, vol. 17, no. 7, pp. 1168–1177, 2008.

[19] Y. Shishikui, Y. Fujita, and K. Kubota, "Super hi-vision - the star of the show!," in *EBU Technical review*, 2009.

[20] M. Sugawara and K. Masaoka, "Uhdtv image format for better visual experience," in *Proceedings of the IEEE*. IEEE, 2012, vol. 101, pp. 8–17.

[21] E. Nakasu, "Super hi-vision on the horizon: A future tv system that conveys an enhanced sense of reality and presence," *Consumer Electronics Magazine, IEEE*, vol. 1, no. 2, pp. 2162–2248, 2012.

[22] H. Shimamoto, T. Yamashita, and H. Maruyama M. Kubota, "Advanced camera technologies for broadcasting," *Micro, IEEE*, vol. 31, no. 6, pp. 51–57, 2011.

[23] T. Yamashita and K. Mitani, "8k extremely-high-resolution camera systems," in *Proceedings of the IEEE*. IEEE, 2012, vol. 101, pp. 74–88.

[24] T. Ito, "Future television - super hi-vision and beyond," in *IEEE Asian Solid-State Circuits Conference*. IEEE, 2010.

[25] Dajiang Zhou, Jinjia Zhou, Gang He, and Satoshi Goto, "A 1.59 gpixel/s motion estimation processir with -211 to +211 search range for uhdtv video encoder," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 4, Apr. 2014.

[26] Chen-Han Tsai, Chi-Sun Tang, and Liang-Gee Chen, "A flexible fully harwired cabac encoder for uhdtv h.264/avc high profile video," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 4, pp. 1329–1337, Nov. 2012.

[27] H. Lee, "Fine grain creation for uhdtv," in *International Conference on Consumer Electronics (ICCE)*, 2013, pp. 1–4.

[28] J. Le Feuvre, J-M. Thiesse, M. Parmentier, M. Raulet, and C. Daguet, "Ultra high definition hevc dash data set," in *ACM Multimedia Systems Conference (MMSys)*, March 2014.

[29] Li Song, Xun Tang, Wei Zhang, Xiaokang Yang, and Pingjian Xia, "The sjtu 4k video sequence dataset," in *Quality of Multimedia Experience (QoMEX), 2013 Fifth International Workshop on*, July 2013, pp. 34–35.

[30] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: principles and practice of background maintenance," in *The Proceedings of the Seventh IEEE International Conference on Computer Vision (ICCV)*, 1999, vol. 1, pp. 255–261.

[31] A. Vacavant, T. Chateau, A. Wilhelm, and L. Lequivre, "A benchmark dataset for foreground/background extraction," in *ACCV Workshop: Background Models Challenge*, November 2012, pp. 291–300.

[32] EC Funded CAVIAR project/IST 2001 37540, "http://homepages.inf.ed.ac.uk/rbf/caviar/," .

[33] X. Wang, X. Ma, and E. Grimson, "Unsupervised activity perception in crowded and complicated scenes using hierarchical bayesian models," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2009, vol. 31, pp. 539–555.

[34] M. Hofmann, P. Tiefenbacher, and G. Rigoll, "Background segmentation with feedback: The pixel-based adaptive segmenter," in *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2012, pp. 38–43.

[35] D. Berjon, C. Cuevas, F. Moran, and N. Garcia, "Gpu-based implementation of an optimized nonparametric background modeling for real-time moving object detection," *Consumer Electronics, IEEE Transactions on*, vol. 59, no. 2, pp. 361–369, May 2013.

[36] Y. Benezeth, P. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Review and evaluation of commonly-implemented background subtraction algorithms," in *Proc. IEEE Int. Conf. Pattern Recognit.*, 2008, pp. 1–4.

[37] D. Parks and S. Fels, "Evaluation of background subtraction algorithms with post-processing," in *Proc. IEEE Int. Conf. Adv. Video Signal Based Surveill.*, 2008, pp. 192–199.

[38] T. Bouwmans, F.E., and B. Vachon, "Statistical background modeling for foreground detection: A survey," *Handbook of Pattern Recognition and Computer Vision*, vol. 4, pp. 181–199, 2010.

[39] T. Horprasert, D. Hardwood, and L. Davis, "A statistical approach for real-time robust background subtraction and shadow detection," in *IEEE Frame-Rate Appl. Workshop*, 1999, pp. 1–19.

[40] K. Kim, T.H. Chalidabhonse, D. Hardwood, and L. Davis, "Real-time foreground segmentation using codebook model," *Elsevier Real-Time Timaging*, vol. 11, pp. 167–256, June 2005.

[41] J. Xu, N. Jiang, and S.Goto, "Block-based codebook model with oriented-gradient feature for real-time foreground detection," MMSP, 2011.

[42] H. Ikeda and E. Ishidera, "An object detection method based on a likelihood background model," FIT2006, 2006, vol. 3, pp. 175–176.

[43] Z. Liu, K. Huang, and T. Tan, "Foreground object detection using top-down information based on em framework," *IEEE Trans. Img. Proc.*, vol. 21, pp. 4204–4217, Sep. 2012.

[44] B. Ristic, S. Arulampalam, and N. Gordon, "Beyond the kalman filter : Particle filters for tracking applications," in *Artech House Radar Library*, 2004.

[45] A. Elgammal, D. Hardwood, and L.Davis, "Nonparametric model for background subtraction," ECCV, 2000, pp. 751–767.

[46] R. Hess and A. Fern, "Discriminatively traned particle filters for complex multi-object tracking," in *CVPR*, 2009.

[47] Q. Wan and Y. Wang, "Multiple moving objects tracking under complex scenes," in *6th World Congress on Intelligent Control and Automation*, 2006.

[48] F. Lutteke, X. Zhang, and J. Franke, "Implementation of the hungarian method for object tracking on a camera monitored transportation system," in *Robotik*, 2012, pp. 1–6.

[49] J. Gross, H. Karl, F. Fitzek, and A. Wolisz, "Comparison of heuristic and optimal subcarrier assignment algorithms," in *International Conference on Wireless Networks*, 2003.

[50] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang, "Framework for performance evaluation of face, text and vehicle detection and tracking in video: Data, metrics and protocol," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 319–336, Feb. 2009.

[51] Sobral A., "Bgslibrary: An opencv c++ background subtraction library," in *IX Workshop de Viso Computacional (WVC'2013)*, Rio de Janeiro Brazil, Jun 2013.

[52] Andrews Sobral and Antoine Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," *Computer Vision and Image Understanding*, vol. 122, no. 0, pp. 4 – 21, 2014.

[53] N.J. B. McFarlane and C.P. Schofield, "Segmentation and tracking of piglets in images," *Mach. Vis. Appl.*, vol. 8, no. 3, pp. 187–193, 1995.

[54] S. Calderara, R. Melli, A. Prati, and R. Cucchiara, "Reliable background suppression for complex scenes," in *ACM International Workshop on Video Surveillance and Sensor Networks*, 2006, pp. 211–214.

[55] C. Wren, A.Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: real-time tracking of the human body," *IEEE Trans. Pattern Ana. Mach. Intell.*, vol. 19, no. 7, pp. 780–785, 1997.

[56] P. Kaetrakulpong and R.Bowden, "An improved adaptive background mixture model for realtime tracking with shadow detection," in *European Workshop on Advance Video Based Surveillance Systems (AVBS)*, 2001.

[57] Z.Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *ICPR*, 2004.

[58] Z.Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773–780, 2006.

[59] J. Yao and M. Odobez, "Multi-layer background subtraction based on color and texture," in *IEEE Computer Vision Recognition Conference (CVPR)*, 2007.

[60] N.M. Oliver, B. Rosario, and A.P. Pentland, "A bayesian computer vision system for modeling human interactions," *IEEE Trans. Pattern Ana. Mach. Intell.*, vol. 22, no. 8, pp. 831–843, 2000.

[61] W. Tsai, , M. Sheu, and C. Lin, "Block-based major color method for foreground object detection on embedded soc platforms," *Embedded Systems Letters, IEEE*, vol. 4, no. 2, pp. 49–52, 2012.

[62] V. Reddy, C. Sanderson, and B.C. Lovell, "Improved foreground detection via block-based classifier cascade with probabilistic decision integration," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 1, pp. 1051–8215, 2013.

[63] A. Beaugendre and S. Goto, "Block-propagative background subtraction system for uhdtv videos," *IPSJ Transaction on Computer Vision and Application*, 2015.

[64] Waseda GOTO Lab's HD and 4K UHDTV Dataset, "http://www.f.waseda.jp/goto/html/uhdtv_dataset.html," .

[65] D. Bloisi and L. Iocchi, "Independent multimodal background subtraction," in *Proceedings of the Third International Conference on Computational Modeling of Objects Presented in Images: Fundamentals Methods and Applications*, Rome, Italy, September 2012, pp. 39–44.

[66] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli, “image quality assessment: from error visibility to structural similarity,” *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.

[67] A. Beaugendre, S. Goto, and T. Yoshimura, “Fast uhd background modelling with mixed order block updates,” in *International Technical Conference on Circuits/Systems, Computers and Communications ITC-CSCC*, 2016.

[68] Xiaoyu Deng, Jiajun Bu, Zhi Yang, Chun Chen, and Yi Liu, “A block-based background model for video surveillance,” in *IEEE International Conference on Acoustics Speech and Signal Processing*, 2008, pp. 1013–1016.

[69] D. Russell and Shaogang Gong, “A highly efficient block-based dynamic background model,” *11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 417–422, 2005.

[70] A. Beaugendre, S. Goto, and T. Yoshimura, “Random block background modelling for foreground detection in uhd videos,” in *IPSJ SIG-CVIM: Computer Vision and Image Media 200th*, 2016.

List of Publications

Journal Papers

[J1] A. Beaugendre, S. Goto, and T. Yoshimura, "Real-time uhd background modelling with mixed selection block updates," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E100-A, no. 2, pp. 1–11, 2017.

[J2] A. Beaugendre, S. Goto, and T. Yoshimura, "Adaptive block-propagative background subtraction method for uhdtv foreground detection," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E98-A, no. 11, pp. 2307–2314, 2015.

[J3] A. Beaugendre and S. Goto, "Block-propagative background subtraction system for uhdtv videos," *IPSJ Transaction on Computer Vision and Application*, vol. 7, pp. 31–34, 2015.

[J4] A. Beaugendre and S. Goto, "Multi-scale foreground detection system for 8k uhdtv videos," *IIEEJ Transactions on Image Electronics and Visual Computing*, vol. 2, no. 2, pp. 174–182, 2014.

[J5] A. Beaugendre and S. Goto, "Real-time refinement method for foreground objects detectors using super fast resolution-free tracking system," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E97-A, no. 2, pp. 520–529, 2014.

[J6] J. Xu, N. Jiang, H. Sun, A. Beaugendre, and S. Goto, “Real-time human detection based on multi-scale bidirectional local template patterns,” *IIEEJ Transactions on Image Electronics and Visual Computing*, vol. 1, no. 1, pp. 28–37, 2013.

International Conference Papers

[C1] A. Beaugendre, S. Goto, and T. Yoshimura, “Near real-time tracking system for 8k uhd videos,” in *International Technical Conference on Circuits/Systems, Computers and Communications ITC-CSCC*, 2016.

[C2] A. Beaugendre, S. Goto, and T. Yoshimura, “Fast uhd background modelling with mixed order block updates,” in *International Technical Conference on Circuits/Systems, Computers and Communications ITC-CSCC*, 2016.

[C3] A. Beaugendre, S. Goto, and T. Yoshimura, “Random block background modelling for foreground detection in uhd videos,” in *IPSJ SIG-CVIM: Computer Vision and Image Media 200th*, 2016.

[C4] A. Beaugendre, C. Zhang, J. Xu, , and S. Goto, “Enhanced moving object detection using tracking system for video surveillance purposes,” in *Visual Communication and Image Processing (VCIP)*, 2012.

[C5] C. Zhang, J. Xu, A. Beaugendre, and S. Goto, “A klt-based approach for occlusion handling in human tracking,” in *29th Picture Coding Symposium (PCS)*, 2012, pp. 337–340.

[C6] C. Zhang, A. Beaugendre, J. Xu, X. Xue, and S. Goto, “A novel klt-based scale feature applied for scale adaptation in crowded scenes,” in *Asian Conference on Computer Vision (ACCV)*, 2012.

[C7] J. Xu, A. Beaugendre, and S. Goto, "Real-time human tracking by detection based on hog and particle filter," in *International Conference on Computer Sciences and Convergence Information Technology (ICCIT)*. IEEE, 2011, pp. 193–198.

[C8] A. Beaugendre, H. Miyano, E. Ishidera, and S. Goto, "Human tracking system for automatic video surveillance with particle filters," in *Asia Pacific Conference on Circuits and Systems (APCCAS)*. IEEE, 2010, pp. 152–155.

Appendix A

Dataset

A.1 Introduction

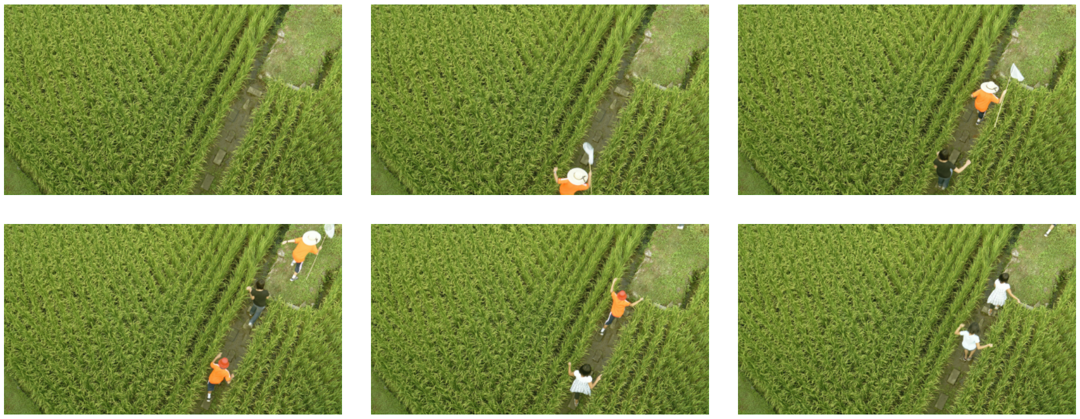
In this appendix chapter we present the different video sequences used in this thesis. First of all, the 8K UHD video is part of the NHK Super Hi-Vision presentation videos. The 4K UHD videos have been manually captured with a SONY 4K Handycam FDR-AX1. Finally, there are 2 HD videos (1080p and 720p) also manually captured and a multi-view sequence from the PETS 2009 dataset.

Each video present various difficulties from moving background parts, occlusions, small size of objects of interest, etc. All the ground truth data (boundary rectangles and/or foreground masks) have been extracted manually.

A.2 8K Videos

A.2.1 Field

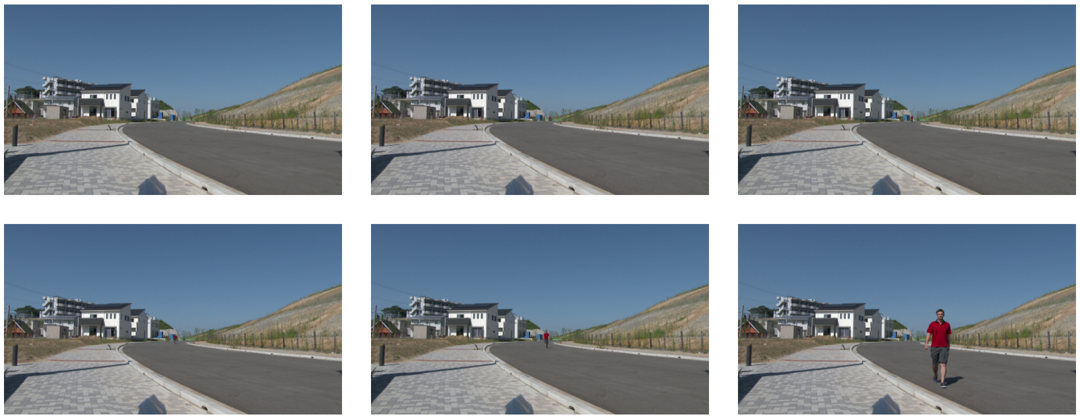
Name	Field
Source	NHK [19]
Size	65.2 GB
Length	00:12
Resolution	7680 × 4320
Number of frames	704
Description of content	Five children are running through the stem field from bottom to top of the scene.
Difficulties	The stems of the field move a lot with the wind. It makes it difficult to distinguish the movement from the object of interest (children) and the many movements of the background parts (field).



A.3 4K Videos

A.3.1 Street1

Name	Street1
Source	Custom video
Size	1.3 GB
Length	01:14
Resolution	3840 × 2160
Number of frames	4440
framerate	60 fps
Description of content	One man is coming from afar towards the camera until is passes it.
Difficulties	The main difficulty is the very small size of the object of interest (the man) at the beginning of the video. It represents less than 0.01% of the size of the image.



A.3.2 Street2

Name	Street2
Source	Custom video
Size	1.7 GB
Length	01:37
Resolution	3840×2160
Number of frames	5799
framerate	60 fps
Description of content	One man is coming from afar towards the camera until is passes it.
Difficulties	The background is more complex than the previous video. There are some slight illumination changes and wind effects.

A.3.3 Corridor

Name	Corridor
Source	Custom video
Size	2.8 GB
Length	02:47
Resolution	3840 × 2160
Number of frames	9746
framerate	60 fps
Description of content	One man is passing the camera from the back to the far end of the corridor. He enters a room to reappear moments later. Then, he walk towards to camera and passes it.
Difficulties	The lack of luminosity makes it difficult sometimes to distinguish the object of interest (the man) clearly. The second difficulty is the small size of the man when he reaches the far end of the corridor. Last, the ceiling lamps change the illumination when the man is getting under them.

A.3.4 Screws

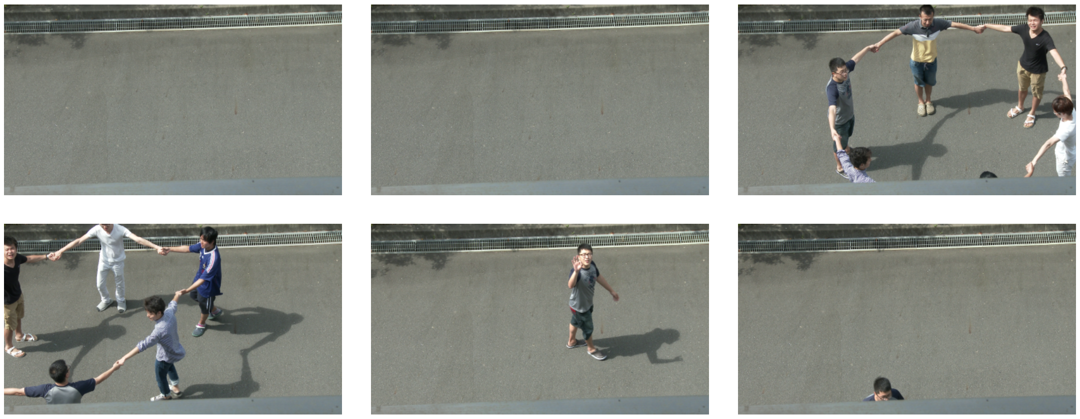
Name	Screws
Source	Custom video
Size	518 MB
Length	00:30
Resolution	3840×2160
Number of frames	1781
framerate	60 fps
Description of content	One man is getting in the room and disposes screws at diverse places then leaves.
Difficulties	The main difficulty is to detect the small screws which are visible only at full scale. The second is the part occlusion with the chair which splits the object of interest (the man) in multiple parts.

A.3.5 Crossing

Name	Crossing
Source	Custom video
Size	1.05 GB
Length	01:04
Resolution	3840×2160
Number of frames	3844
framerate	60 fps
Description of content	It is a standard street surveillance video in which cars and bus are circulating.
Difficulties	The main difficulty is the small size of the objects and the occlusions with the background (trees and posts).

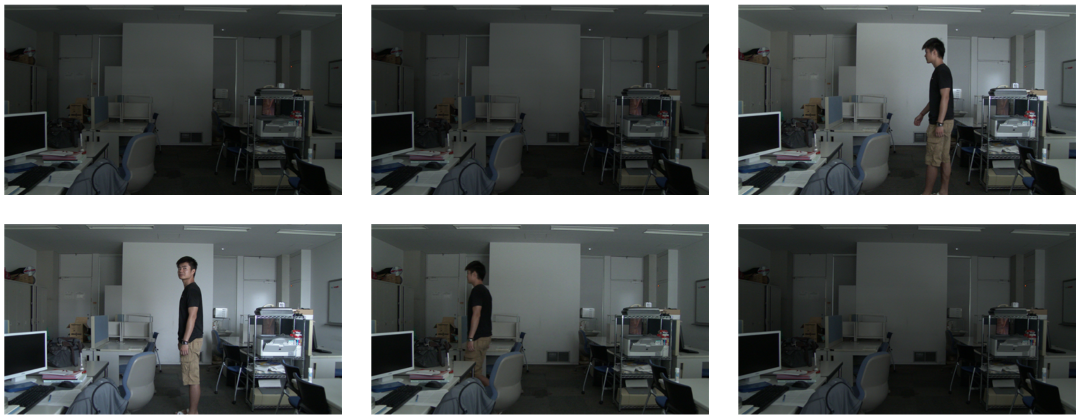
A.3.6 Circle

Name	Circle
Source	Custom video
Size	875 MB
Length	00:53
Resolution	3840 × 2160
Number of frames	3171
framerate	60 fps
Description of content	A group of men is meeting and creating a human circle. Then they move altogether and finish by splitting in different directions.
Difficulties	The strong shadows produced by the group is the main problem here.



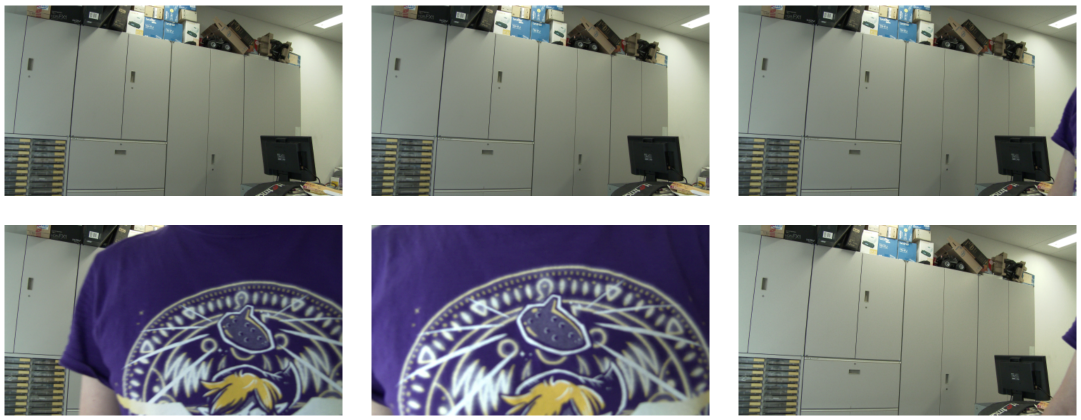
A.3.7 Illu

Name	Corridor
Source	Custom video
Size	263 MB
Length	00:15
Resolution	3840 × 2160
Number of frames	886
framerate	60 fps
Description of content	One man is coming from the right, stops in the center for some seconds and then leaves on the left.
Difficulties	The main obstacle in this video is the sudden illumination change which happens twice, from dark to bright and then to dark again.



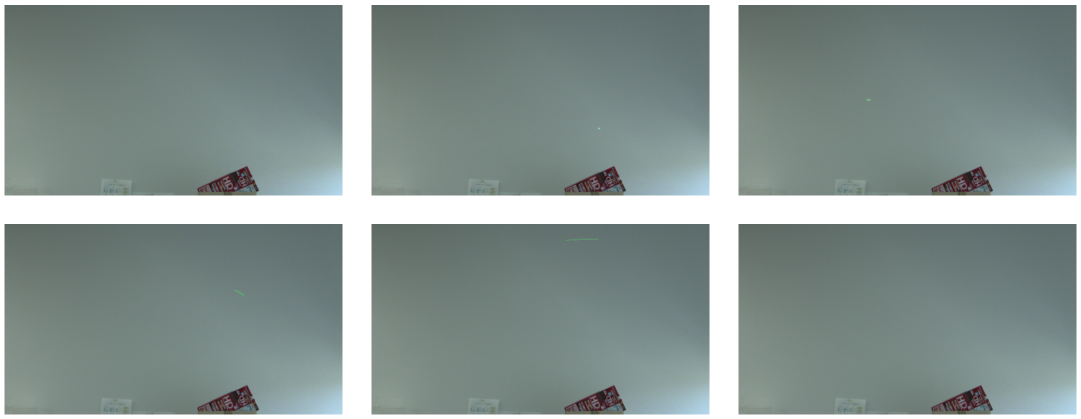
A.3.8 80percent

Name	80percent
Source	Custom video
Size	178 MB
Length	00:10
Resolution	3840 × 2160
Number of frames	610
framerate	60 fps
Description of content	A man comes, masks the whole screen and then leaves.
Difficulties	No particular difficulty except that the object covers the scene entirely with his body.



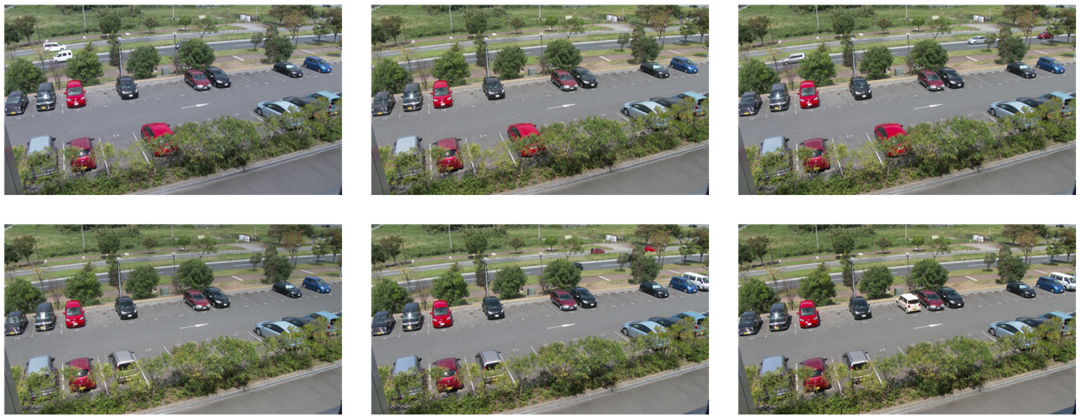
A.3.9 Laser

Name	Laser
Source	Custom video
Size	246 MB
Length	00:14
Resolution	3840 × 2160
Number of frames	855
framerate	60 fps
Description of content	A green laser pointer is projected on a wall and moves quite quickly.
Difficulties	The laser's size if very tiny which makes it difficult to see. Also the laser moves quickly changing its shape a lot.



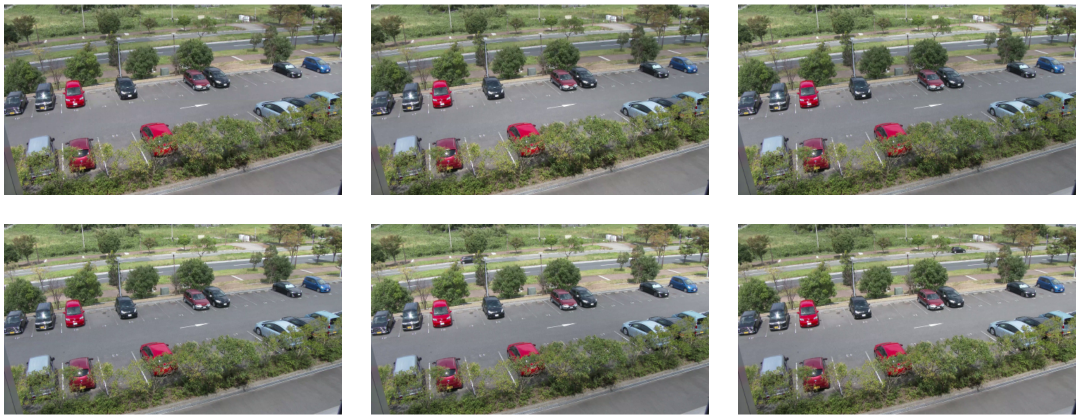
A.3.10 Parking

Name	Parking
Source	Custom video
Size	43.2 GB
Length	40:37
Resolution	3840×2160
Number of frames	146 189
framerate	60 fps
Description of content	A standard parking surveillance video in which cars come and go away.
Difficulties	The illumination changes a lot and very often but smoothly. Many cars cross the scene and some park or leave. The fact that it is a very long video car also be an issue for very slow methods and/or those which require a lot of resources.



A.3.11 Parking2

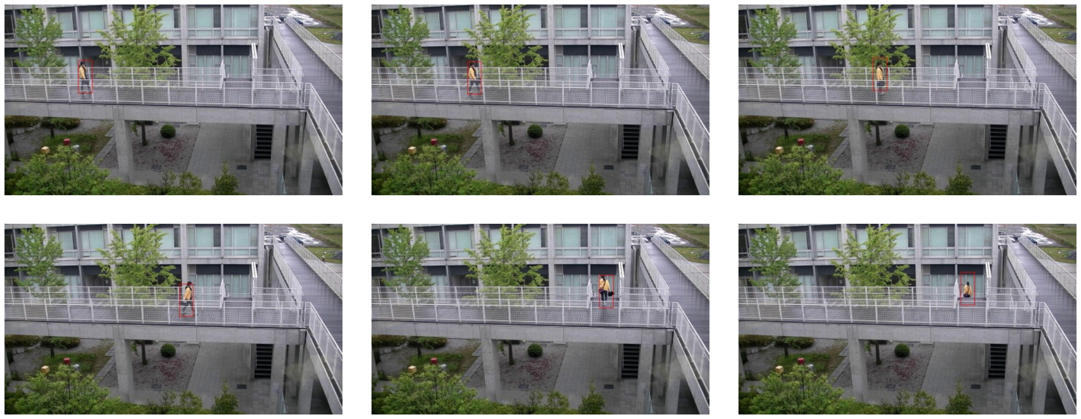
Name	Parking2
Source	Custom video
Size	62.5 MB
Length	0:10
Resolution	3840 × 2160
Number of frames	606
framerate	60 fps
Description of content	A standard parking surveillance video in which cars come and go away.
Difficulties	Similarly to *Parking*, in this sequence, the illumination changes slowly while cars pass through. The video is very short on purpose to easily test other methods.



A.4 Other videos

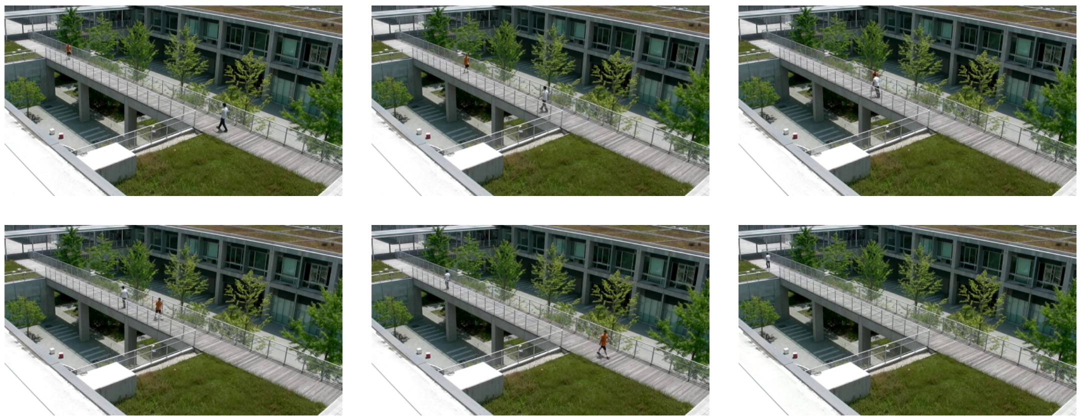
A.4.1 Single

Name	goto_lab_01
Source	Custom video
Size	2.8 GB
Length	02:47
Resolution	1920 × 1080
Number of frames	9746
framerate	30 fps
Description of content	A girl si passing in front a tree during a windy day.
Difficulties	The leaves in the background are moving a lot because of the wind making it difficult to detect only the object of interest.



A.4.2 Crossing2

Name	goto_lab_02
Source	Custom video
Size	2.8 GB
Length	02:47
Resolution	1280×720
Number of frames	9746
framerate	30 fps
Description of content	Two men are crossing on a plateform. When they cross, one of them falls ,gets up and leaves.
Difficulties	There is a total occlusion between the two objects of interest.



Appendix B

Quality Metrics

The foreground detectors can provide two kinds of results: a foreground mask and a list of blobs. The foreground mask is a pixel level output and it shows all pixels which have been set to 255 when detected as foreground and 0 when detected as background. The list of blobs is an object level output which contains all the boundaries of the connected components detected. The foreground mask is used to compare the visual quality but there is not notion of object, the content is indifferent. The blobs on the contrary bring the notion of objects, more complex entities which have a position and a shape. For each one of those type of results appropriate metrics are required.

B.1 Object Metrics

The evaluation of the accuracy and precision of the foreground objects and of the trackers can be done by using the CLEAR metrics [50]. The CLEAR metrics consist of a total of four scores: a pair of accuracy and precision scores, one pair for each detection and tracking. The accuracy assesses the existence or not and the precision the rightness, if the object/tracker has a similar shape and size as its ground truth model.

Before anything else, the system output objects (foreground object or tracker) must be associated one-to-one with ground truth objects. With N objects and M ground truth objects, the association should result in a $N \times M$ matrix. However since we require a one-to-one correspondence, we add fake output objects if $N < M$ or we add fake ground truth objects if $N > M$. Having more output objects than ground truth objects means that there are more detected objects than there are in the reality, we are in a case where there are *false positives.* The contrary means that some ground truth objects are *missing* in the results.

The accuracy score for the detection, also called the Multiple Object Detection Accuracy (MODA), is based on the count of misses m_t and false positives fp_t for each frame *t*. The normalized score is defined as:

$$N\text{–}MODA = 1 - \frac{\sum_{t=1}^{N_{frames}} (c_m(m_t) + c_f(fp_t))}{\sum_{t=1}^{N_{frames}} N_G^{(t)}}, \tag{B.1}$$

with $N_G^{(t)}$ being the number of ground truth objects in the tth frames. The criticalness of the number of misses and false positives can be customized through the cost functions $c_m()$ and $c_f()$ (both = 1). <u>It is very important to notice that if the sum of misses and false positives is equal or bigger than the total number of ground truth objects ($m + fp \geq N_G$, the accuracy score can become null or negative.</u>
The tracking accuracy, Multiple Object Tracking Accuracy, follows a similar formula but it takes an additional parameter into account, *ids* the number of ID mismatches. Each fusion or split will be counted as an ID switch. The cost functions include $c_s = log_{10}$ weighted function to penalize the ID switches.

$$N\text{–}MOTA = 1 - \frac{\sum_{t=1}^{N_{frames}} (c_m(m_t) + c_f(fp_t)) + c_s(ids_t)}{\sum_{t=1}^{N_{frames}} N_G^{(t)}}. \tag{B.2}$$

The Moving Object Precision scores, MODP for the detected objects and MOTP for the tracked objects, follow the same formula:

$$N\text{–}MODP = \frac{\sum_{t=1}^{N_{frames}} \sum_{i=1}^{N_m^t apped} \frac{|G_i^{(t)} \bigcap D_i^{(t)}|}{|G_i^{(t)} \bigcup D_i^{(t)}|}}{\sum_{t=1}^{N_{frames}} N_{mapped}^{(t)}}. \tag{B.3}$$

The precision is based on the calculation of the overlapping areas between the Ground Truth objects $G^{(t)}$ and the system output objects $D^{(t)}$ for the frame t. N_{mapped}^t is the number of object pairs mapped. If no object has been mapped, the precision will be forced to zero for the frame.

B.2 Pixel Metrics

The evaluation for the foreground mask is based on static quality metrics. Usually the foreground mask is a black and white image, with the pixels detected as background in black and the pixels detected as foreground in white. The majority of them require to count the number of foreground pixels classified as foreground also called True Positive (TP), the number of background pixels classified as background or True Negative (TN), the number of False Positive (FP) which are background pixels classified as foreground and the number of foreground pixels classified as background or False Negative (FN). To measure the static quality metrics we computed different metrics: the Recall or detection rate which focuses on missed detection or false negative

$$\text{Recall} \quad = \frac{TP}{TP + FN}, \tag{B.4}$$

the positive prediction or Precision (Pre.)

$$\text{Precision} \quad = \frac{TP}{TP + FP} \tag{B.5}$$

which takes into account the background noise and incorrect detection or false positive. A perfect recall score does not necessary mean that the detection is perfect, an foreground mask with only pixels detected as foreground will obtain the perfect recall score. The same situation happens with the precision and an mask in which all the pixels set as background. Therefore, FN and FP as well as recall and precision measures should not be consider separately.
The Similarity measure considers both incorrect and miss detections equally:

$$\text{Similarity} \quad = \frac{TP}{TP + FN + FP}. \tag{B.6}$$

We measured another metric which uses the pixels metrics, the F-score which is the weighted harmonic mean of Precision and Recall is defined by

$$\text{F–Measure} \quad = \frac{2 * Precision * Recall}{Precision + Recall}. \tag{B.7}$$

Additionally we compute the perceptual measure SSIM (Structural SIMilarity) introduced in [66]:

$$\text{SSIM(S,G)} \quad = \frac{1}{n}\sum_{i=1}^{n} \frac{(2\mu_{S_i}\mu_{G_i} + c_1)(2cov_{S_iG_i} + c_2)}{(\mu_{S_i}^2 + \mu_{G_i}^2 + c_1)(\sigma_{S_i}^2 + \sigma_{G_i}^2 + c_2)}, \tag{B.8}$$

in which σ_{S_i}, σ_{G_i} are the standard deviations, μ_{S_i}, μ_{G_i} the means and $cov_{S_iG_i}$ the covariance. The different values used in the literature are $c_1 = (0.01 \times L)^2$ and $c_2 = (0.03 \times L)^2$ in which L is the dimension size and $L = 255$ for gray-scale images. Finally, the Peak Signal-Noise Ratio (PSNR) is calculated by:

$$\text{PSNR} \quad = \frac{1}{n}\sum_{i=1}^{n} 10\log_{10} \frac{m}{\sum_{j=1}^{m} ||S_i(j) - G_i(j)||^2}. \tag{B.9}$$