Università degli Studi di Padova

Department of Information Engineering

Master Thesis in ICT for Internet and Multimedia

Real Time Motion Estimation Algorithm for Temporal Denoising

Supervisor
Giancarlo Calvagno
Università di Padova

Co-supervisor
Stefano Andriani
Arnold & Richter Cine Technik GmbH

Master Candidate
Cesare Caratozzolo

Academic Year 2018/2019

Università degli Studi di Padova

Department of Information Engineering

Master Thesis in ICT for Internet and Multimedia

Real Time Motion Estimation Algorithm for Temporal Denoising

Supervisor
Giancarlo Calvagno
Università di Padova

Co-supervisor
Stefano Andriani
Arnold & Richter Cine Technik GmbH

Master Candidate
Cesare Caratozzolo

Academic Year 2018/2019

Abstract

Motion compensation and estimation are tools to exploit the temporal redundancy in subsequent frames of a video sequence to assist certain video processing applications such as compression, transmission or denoising. Unfortunately, these kind of algorithms could be very computational demanding, so many solutions to reduce their complexity have been proposed.

This thesis introduces a low-complexity, but efficient, motion estimation algorithm, that could be implemented, e.g, in FPGA, in a professional digital camera to apply it on-the-fly while recording a video-sequence. The main aim of the proposed algorithm it to improve the performance of an already existing denoising algorithm, but it could be used for improved compression or image-stabilization algorithms in the future. To meet the real-time constraint, the prediction accuracy is traded for a reduced number of operations that is reflected in a faster computational time, but, at the same time, always delivering reliable and usable temporal estimation.

The algorithm has been tested on different video sequences and the results and the improvements are analysed by comparing the prediction errors and the denoising rates to a ground truth, while some considerations on the limits of the approach are reported.

Contents

1 Introduction

The concepts of motion compensation and motion estimation were first introduced in the scope of video coding, with the objective of exploiting temporal redundancies in the sequences to increase the compression efficiency. The most widely used algorithms fall into the category of *Block Matching Algorithms (BMA)*, intuitively simple as they are purely based on the division of the frames of a sequence into blocks. Each one of these is compared to equally sized blocks in the previous frame, trying to find the best match into a search area by minimizing an error function. The found translation is then saved as a motion vector.

The drawback of the block matching algorithm is that they are usually very computationally demanding. For this reason, a lot of work has been done to reduce their complexity and many algorithms have been proposed, exploring different searching methods and proposing various approaches to the subject.

The objective of this thesis is to continue the work of E. Ballan [1], where a set of suitable parameters for the Full Search Algorithm has been studied with the aim of improving the efficiency of the same temporal denoiser used in this thesis.

Noise is always introduced by the sensors in a camera and it depends on the construction of the sensor, the pixel-pitch, and also the brightness of the recorded scene. This noise could be then reduced or amplified by the color processing chain used to convert an image that is linearly related to the light, into a good-looking image on a display or on a large cinema screen. This conversion usually applies one or more log-shaped curve to enhance the dark scene and compress the highlights. In this situations, the noise level in the dark could become significative and a denoising algorithm is required to make the sequence more pleasant to be viewed. Experience in the cinema industry has proved that this operation must be performed in the temporal direction, because pure spatial algorithms could generate abnormal and visually annoying behaviors of the residual noise left in the image. Motion estimation aids the application of denoising algorithms on the temporal axis by detecting the optical flow of the sequence. Therefore, if an object is moving, denoise operations are applied on the appropriate coordinates between two different frames. Even if the motion estimation could be, theoretically, performed on more than two frames, in this thesis just two are considered due to memory constraint in the camera, where more than one frame buffer is not feasible under the nowadays FPGA technology and power constraints.

It is interesting to note that in cinema applications, a certain amount of noise is also added in post production, as a completely denoised sequence will appear unnatural to the eye. This process is known as *film grain overlay*, and it is used for example when blending CGI elements together as if they were part of the same environment, making the sequence look more realistic by subtly tricking the eye. However, this happens at the final stages of production course, as at first it is important to have a clean image for processing purposes.

The algorithm proposed in this thesis aims at reducing the computational complexity of the FSA, at the cost of a decrease in the accuracy, making it feasible to be implemented in real-time along with a suitable denoising algorithm, as the video sequence is being captured, through FPGA technology (although only the algorithm will be discussed). It is important for the block matching algorithm to use as reference image the already denoised frame, because the removal of artifacts will return a more accurate prediction. The described approach increasingly reduces the level of precision of the prediction process by gradually decreasing the amount of computations performed. The purpose is to perform a rough initial prediction and denoising, while the sequence is being captured, that will aid the more complex algorithms applied during the processing phase that will smoothen the final outcome.

FPGA technology is a trade-off technology well suited for real-time hardware implementation of the image processing applications. They are faster than any software, but thanks to their rewritable nature they are much more flexible and future-proof than the ASIC technology, even if their power consumption is higher than ASIC. Both FPGA and ASIC share the same high development costs and the need to keep all the algorithm in fixed-precision and with a reduced operand set, e.g, divisions and complex exponential or logarithmic operations shall be avoided, or replaced by approximated versions stored in multi-dimensional look-up-tables.

Alternatively, other devices that recently seem to be giving promising results are GPUs, with lots of research spent to improve certain liabilities that typically characterized them, such as the enormous power consumption that made GPUs less desirable for certain devices powered by batteries. Another downside of GPUs is that they run software (although very close to hardware level), which introduces a higher level of latency compared to FPGAs, as instructions are read, and data is retrieved and stored in the memory. This is partly countered by the use of direct memory access, that boosts data management operations.

Among the advantages of GPUs, there is their ability of performing floating point operations, along with their high level of parallelization, which allows them to run software faster than normal CPUs.

The choice between FPGA and GPU technology depends on the application. In this thesis, an algorithm designed for real time applications is described. Although GPUs, as a result of their strong parallelization and direct memory access, allow an almost real time software execution, FPGA technology might still be preferable, as it can be directly integrated on the circuit of the camera, allowing them to sustain the high frame-rate of the camera.

The rest of this thesis is organized as follows: in Chapter 2, a brief theoretical background is presented to introduce the subject, where the concepts of Bayer pattern, Block Matching and Denoising are explained; in Chapter 3, other relevant work on the same subjects is cited, that contributed on different aspects of the issue; in Chapter 4, the algorithms studied in this thesis are explained and examined, while the results obtained are reported in Chapter 5. Finally, in Chapter 6, the obtained outcomes are briefly discussed, with suggestions on the possible future work that can be carried out to further improve the topic.

2
Theoretical Background

Motion estimation has a wide variety of applications in ICT, such as motion compensation for video coding standards and compression for transmission purposes, where temporal correlation between subsequent frames is exploited to reduce the amount of data to be stored [2]; or the removal of noise from video sequences to make them smoother to the viewers eye.

Before elaborating the objective of the thesis, the process of image capturing, the block matching algorithm and the denoising operation are briefly discussed to provide the reader with a theoretical basis about the subject.

2.1 Digital Cameras

To capture an image, the sensor of a digital camera is provided with a pixel array. Each pixel consists of a light-sensitive diode that converts the energy captured from the photons into an electrical signal. There are two main types of sensors, Charge Couple Devices (CCD) and Complementary Metal-Oxide Semiconductor (CMOS). These chips do not have the inherent ability to capture the photon's wavelength (i.e, discriminate between colors), but are able to discern only their number (i.e, the overall brightness level of the captured scene): essentially they are gray-scale sensors. To be

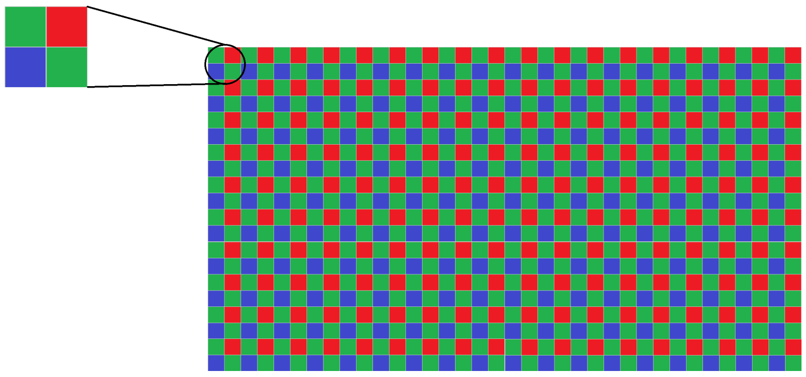

Figure 2.1: The Bayer color filter array.

able to differentiate between colors, two approaches have been developed:

- **3-chip camera**: the incident light is split by using optical prisms and then the three primary colors R, G, and B are captured by three separate sensors provided by color filters.
- **Single-chip camera**: a Color Filter Array (CFA) is applied to a single sensor, allowing each pixel to capture just one color channel. The other two colour components must be calculated via interpolation of the neighbor values.

The 3-chip color method is very expensive, and it is affected by several problems. Since it requires three times the number of sensors, one for each color component, either the camera results very large and heavy, or the sensors are very small, which reduces the amount of captured light (resulting in darker and noisier images) and introduces alignment problems. On the other hand, a single-chip camera is cheaper and smaller. However, even if it requires a computational demanding interpolation algorithm, and the final visual results strongly depend on it, it is usually preferred.

Different color filter arrays have been proposed in the last decades. The most common one is that proposed by Bayer [3], shown in Figure 2.1. As it can be observed, there are twice as many pixels of the green component as of the other two. The reason behind this model is that the human eye is a lot more receptive to green light compared to the other two. Also, most scene illuminations are usually more deficient in blue light compared to green and red. The Bayer pattern is the reference model for this thesis too, as the video sequences provided and the camera used to record them are built using it.

2.2 Block Matching

A lot of research has been done to define a mathematical model for the optical flow [4, 5] and motion estimation, and many algorithms have been developed following different approaches, such as pel-recursive procedures [6] and block matching. Usually, there is no universal algorithm, and the choice depends on the application. However, the method that has shown to provide some of the best results is the intuitive concept of *block matching motion estimation.*

The main assumption behind block matching motion estimation is that the displacement between subsequent frames in a video sequence is small and objects retain a uniform motion [7]. In this approach, a frame is divided into blocks of size $M \times M$. Each block is then compared to blocks of the same size from the previous frame, searching for the displacement that maximizes the correlation [8], represented as a motion vector $[x, y]$, where the coordinates describe the horizontal and vertical displacement of the block from its initial position. This can be described as a minimization problem, where the objective functions are usually the *Mean Absolute Error (MAD)*, defined as

$$MAD = \frac{1}{M \times M} \sum_{i,j} |p_{i,j} - q_{i,j}|, \tag{2.1}$$

and the *Mean Square Error (MSE)*, defined as

$$MSE = \frac{1}{M \times M} \sum_{i,j} (p_{i,j} - q_{i,j})^2, \tag{2.2}$$

where $p_{i,j}$ and $q_{i,j}$ are the pixels of coordinates i, j within the compared blocks.

As the number of comparisons drastically increases with the frame size, usually a search window is defined around the neighborhood of the considered block, and correlation is researched only within that window. If no blocks satisfy the correlation condition given by a threshold, no motion vector is calculated, or it is set to zero.

Several factors affect the number of computations, the complexity, and the accuracy of the algorithm. Firstly, the choice of the block size plays a crucial role. Larger blocks imply more computations for each comparison, as the number of pixels per block increases; it also rises the chances of having objects moving in different directions within the same block. However, this means also that each frame will be divided into

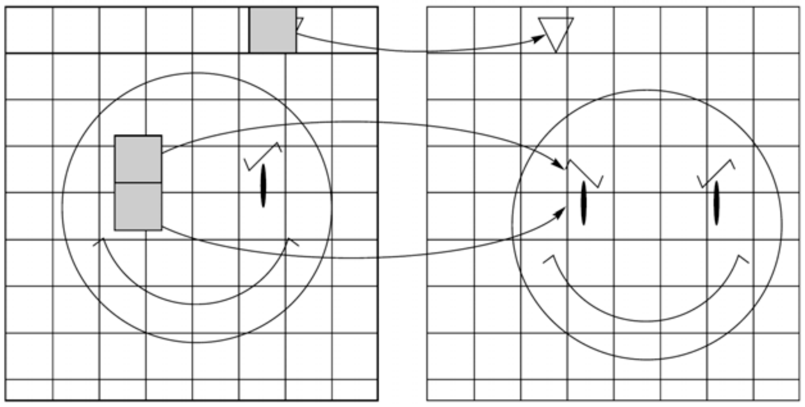

Figure 2.2: Block matching motion estimation. If the displacement is within a defined range, motion vectors are calculated (face), otherwise if it is too big, no estimation is performed (triangle). Figure taken from [2].

fewer blocks, leading to a trade-off between the precision of the prediction and the computational complexity of the algorithm. On the other hand, smaller block sizes are more flexible in detecting small movements, but the precision of their motion estimation could rapidly drop in case of medium-strong noise level.

Considering for example an image of resolution 1920×1080 pixels. A block size of 4×4 divides the image into 129600 blocks. Increasing the size of the block to 8×8 reduces their number to 32400, decreasing it by a factor of 4. The time required to compare two blocks is negligibly affected by their size, as software and hardware implementations usually optimize these types of operations with parallelization. What actually affects the computational time is the number of comparisons that the algorithm must perform, therefore reducing the amount of blocks drastically improves the speed of the matching.

Another factor is given by the size of the search window. A larger one guarantees a broader field where to search matching blocks, increasing the accuracy; on the other hand, this increases the number of comparisons that must be done for each block of the frame, leading to an increment of the computations required. Hence, another precision/complexity trade-off is encountered.

While previous coding standards adopted the use of macroblocks of size 16×16 for the luminance samples (and a color sampling rate of 4:2:0 for the other components), the current H.265 coding standard, also known as *High Efficiency Video Coding (HEVC)* Standard, allows the $L \times L$ size of the luminance component to be chosen as $L = 16, 32$, or 64 samples [9].

2.3 Denoising

Denoising is a very challenging task in image and video processing. An image is affected by noise due to many reasons, for example different sensitivities of the single light capturing sensors, temperature fluctuations and electronic instabilities. Usually, the model used is *Additive White Gaussian Noise (AWGN)* [10], but that is not always the case. For example, *salt and pepper* noise adds sharp artifacts to images, that can be seen as black and white dots spread around, and it cannot be modeled as Gaussian.

To remove noise from an image, a wide variety of filters are adopted with the most elementary ones being low-pass. The *Ideal Low-Pass Filter* is described by the following equations

$$H(u,v) = \begin{cases} 1, \text{if } D(u,v) \leq D_0, \\ 0, \text{if } D(u,v) > D_0, \end{cases}$$

where D_0 is the cut-off frequency and the range $D(u,v)$ is described by

$$D(u,v) = \sqrt{\left(u - \frac{P}{2}\right)^2 + \left(v - \frac{Q}{2}\right)^2}. \tag{2.3}$$

where P and Q describe the size of the window of the filter.

However, low-pass filters do not represent an optimal solution, as the quality of the image resents from their use, with details being blurred out and sharpness lost. A prime example would be the loss of edges, due to the fact that they correspond to strong variations (and therefore high frequencies) of an image. The larger D_0, the more details will be preserved, but less noise will be removed. Moreover, to eliminate salt and pepper noise, statistical solutions such as the Adaptive Median Filter are adopted, which selects the median value within a search window of adaptive size.

To be able to preserve details, many non-linear approaches were proposed. Examples of popular filters include the Bilateral filter and the Non-Local Means (NLM).

The Bilateral filter is described by the equation

$$BF(I)_p = \frac{1}{W_p} \sum_{\mathbf{q} \in S} G_{\sigma_s}(||\mathbf{p} - \mathbf{q}||) G_{\sigma_r}(|I_p - I_q) I_q \tag{2.4}$$

where W_p is a normalization parameter, G_{σ_s} is the space parameter, which describes the size of the considered neighbourhood of pixels, and G_{σ_r} is the range parameter, that describes the minimum size of an edge, and is effectively what allows this filter to preserve edges. The functions $G_\sigma(x)$ refer to the Gaussian

$$G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{x^2}{2\sigma^2}). \tag{2.5}$$

The NLM achieves similar results to the Bilateral by averaging pixels with windows of similar contents, that are not necessarily located on the neighbourhood of the considered pixel.

All the described filters were developed for reducing noise on single images, therefore they operate in the spatial domain, but they are not directly suitable for video processing, as they do not take into account the temporal behavior of the noise. Applying to a video sequence only spatial filtering independently to each frame generates good-looking single frames, but they might behave strangely when shown as sequence.

Temporal denoising filters were developed with the objective of removing visual impurities and variations along the temporal direction, also known as temporal nonstationarities [11]. For example, these can be fast alterations of illumination and movement of an object. Temporal nonstationarities can be reduced when considering groups of slow-varying frames.

Any motion present in these groups can be compensated with motion estimation, further increasing the correlation between the images and aiding the development of the denoising filter, as it can be seen in Figure 2.3, which shows the results of temporal denoising after digitally applying White Gaussian noise to the *Trevor White* video sequence. However, motion compensation does not solve all the problems: prediction can be imperfect, and motion introduces newly covered or uncovered regions. These factors must be taken into account when developing the filter, as simple solutions seemed to affect either stationary or non-stationary regions, introducing blurring effects.

That being said, spatial and temporal filtering, in principle, are very similar. The difference is in the type of signal that is being processed: in spatial filtering, all the pixels belong to the same image, and each one is usually centered around a window of neighbouring pixels. In temporal filtering, the processed signal is a sequence of pixels belonging to different frames taken in temporal order.

Typically, in video processing, hybrid solutions are adopted, to take advantage of both spatial and temporal correlation in the frame sequences [12]. As denoising is a very computationally demanding operation, considering that all pixels of the frame must be analyzed, the length of the group of frames to consider and the size of the filter window must be adequately chosen, based on the application. For real time applications, taking only groups of two frames (the current and the previous) might be an optimal solution to speed up the process. As the green colour is usually the least noisy of the RGB color scheme, it could be exploited to guide the filter along a direction, while denoising the red and blue components. This can be helpful in the detection and preservation of edges too.

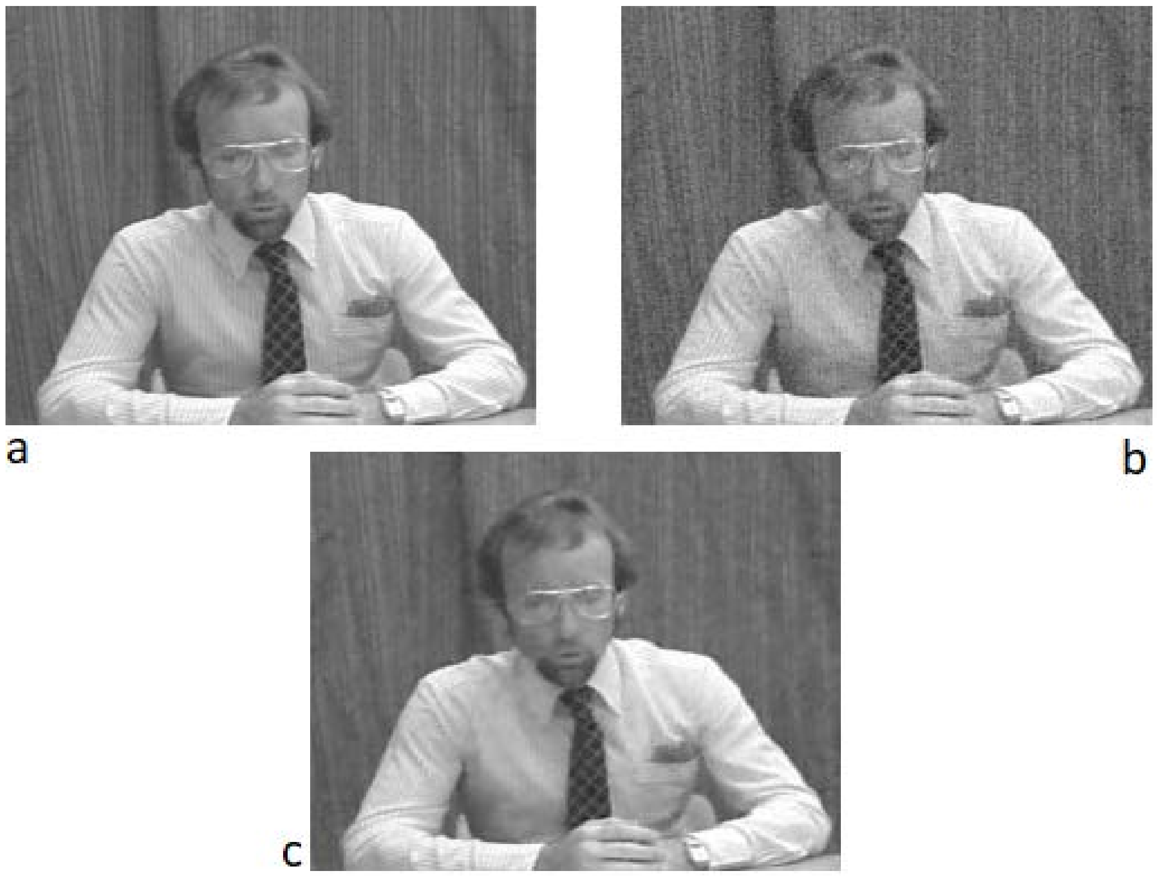

Figure 2.3: a) Original frame from the “Trevor White” image sequence; b) Frame with added White Gaussian noise; c) Denoised frame with a spatio-temporal filter aided with motion compensation. Images taken from [11].

3
Related Works

Lots of research has been done on the motion estimation problem and much effort was put to improve existing algorithms or develop new ones. Many different solutions have been proposed, with multiple implementations, that vary from software to hardware. The aim of the research is usually either to improve the accuracy of the prediction, or to reduce the computations required and to increase the speed, i.e, Fast Block Matching Algorithms (FBMA).

Software Implementations Among the first FBMA developed, we find the Three Step Search (TSS) [13], where a square search window was adopted. The TSS consists on selecting a step size and gradually reducing it, while looking for the global minimum of the objective function. The algorithm has been subsequently improved with a new version known as New Three Step Search (NTSS) [14], which guarantees better performance when the motion on the video is small, as it is based on the assumption that movement in a video sequence is center based. This means that motion vectors are assumed to be small and the prediction centered around the original position of the block.

Another approach is the Diamond Search (DS) algorithm [15], where the square search window is rotated. As the NTSS, it is based on the same assumption, with the addition that motion is mainly oriented horizontally or vertically. It returns reasonable results while reducing the average number of searched locations, by far increasing efficiency.

Further developments were obtained by exploiting the information of motion vectors previously computed. If an object is detected to be moving in a certain direction, there is a high probability that the motion will continue with the same orientation in future frames, as discussed in the Adaptive Rood Pattern Search (ARPS) algorithm [16].

Other suggested approaches include introducing a hexagonal search pattern [17], or dividing the frame in macro and micro-blocks based on the amount of movement detected on each area [18], allowing an in-depth search only where needed, saving a lot of computations while obtaining positive results.

FPGA Implementations As software algorithms usually revolve around iterations, they require a long time to complete all the computations, generally around seconds. Some authors began exploring the possibility of having real time motion estimation by using pipeline architectures and parallel processing. However, due to hardware limitations, precision is usually lost in exchange for speed. In [19], an FPGA-based algorithm with smoothing is proposed, with the intent to improve the accuracy compared to previous designs, while always keeping in mind the constraints of feasibility and velocity. Smoothing is achieved with the introduction of three masks, whose parameters are chosen based on the hardware resources. Overall, as a result of this addition, outcomes seem to be promising.

GPU Impementations Similar to the FPGA case, GPU based approaches aim to improve the speed of the algorithms. Taking advantage of the multi-core structure of a GPU to boost parallelism, in [20] an innovative technique known as *Multilevel Resolution Motion Estimation (MLRME)* is designed for HEVC. The method proposes to generate multi-level resolution frames and apply the full search method to each of them, selecting the *coarsely best MV* (cbMV) as the starting point for the computation of the motion vector of the original frame at maximum resolution.

The high parallelism and acceptable complexity are obtained at the cost of a slight reduction in accuracy. However, the increase in computational speed is quite remarkable, as it can almost meet real-time coding requirements.

4 Proposed Approach

The main objective of this thesis is to study more effective estimates of motion rather than the computationally demanding Full Search Block Matching Algorithm (FSBMA). Motion estimation has proven to be a valid tool to apply denoising algorithms to video sequences with greater precision. The full search approach is conceptually the simplest method to perform motion estimation. Despite its simplicity, it is also very computationally demanding and more appropriate for off-line software applications.

Our target is to investigate whether it is possible to revise the approach with the perspective of applying it into the cameras, that will execute the motion estimation and denoising, while recording the video sequence. FPGA technology has been previously investigated with the perspective of hastening the FSBMA, without losing accuracy [21]. However, as the algorithm shall run in real time, the objective is to develop a specific method that trades precision for speed, while obtaining acceptable results when used to improve a temporal denoiser. Further denoising steps to smoothen the final outcome should be used in a second moment via software techniques.

In the following, a description of three techniques developed to simplify the full search is provided, and the results obtained after testing them on several test video sequences are presented. The first algorithm is a revision of the full search method, while the two others aim at reducing the accuracy of the full search algorithm to boost the computation speed. Furthermore, these three approaches were further hastened with the introduction of cut-offs based on a check on the average error. Every algorithm has been written in *Matlab*.

The algorithm must work on a frame in its RAW format, i.e, as it is captured by the sensor, without any prior processing or interpolation. Therefore, every pixel corresponds to a single colour in the RGB scheme, and in this case they follow a Bayer pattern. As the camera keeps recording while the algorithm is running, motion estimation and denoising must be real-time. The suggested idea is to keep in a fast access buffer memory the immediately previous captured and denoised frame, while all the previous others will already be stored in an external memory.

Furthermore, the size of the blocks is kept constant while the algorithm is operating on a video sequence. In fact, the block size must be selected before the hardware implementation. Different block sizes were tested, both of squared and rectangular shapes, and compared the outcomes to see which one returned the best results.

Finally, as our aim is to assign to each block the motion vector corresponding to the position that minimizes the inaccuracy, in the testing phase the following objective functions were utilized: the *Sum of the Absolute Differences (SAD)*, defined as

$$SAD = \sum_{i,j} |p_{i,j} - q_{i,j}|, \tag{4.1}$$

where $p_{i,j}$ and $q_{i,j}$ are the pixels of coordinates i, j within the blocks of the current two frames that are being compared, and the *Sum of Squared Differences (SSD)*, defined as

$$SSD = \sum_{i,j} (p_{i,j} - q_{i,j})^2. \tag{4.2}$$

These two error functions correspond to the *Mean Absolute Error (MAE)* and to the *Mean Squared Error (MSE)*, but the division by the number of pixels is avoided because it does not affect the final result; on the other hand, it saves on a computationally expensive operation (the division) that requires time when performed on hardware. However, the SSD still requires multiplications, which makes it less desirable for the objective. The results returned by both functions will be compared, expecting that they will be similar.

4.1 Full Search Revisited

The common full search algorithm consists of the comparison of two subsequent frames of a video sequence, where an exhaustive search is performed. The first frame is usually divided into rectangular blocks of $M \times N$ pixels (commonly $M = N$, however, in hardware, line buffering is an expensive operation, therefore it is desirable to have more horizontal than vertical lines). Each block is then compared to a neighbourhood of blocks around its corresponding position and within a predefined search window on the previous frame. The translation corresponding to the smallest difference, given a defined error function, is defined as the motion vector [22].

However, in view of the big dimension of search windows, this process is frequently lengthy. The revised presented approach aims to reduce the amount of inspected pixels and searched locations, and review the concept of search window by taking advantage of the motion vectors calculated in the previous iteration.

Pixel Decimation Pixel decimation consists in only considering a fraction of the pixels of each block for the block matching purposes. This approach allows to reduce the number of operations in each comparison between blocks [23]. As previously stated, frames are provided prior to any processing, and each pixel coincides to a single colour, following the Bayer pattern.

In this procedure, pixel decimation is applied by taking into account only of green pixels. This decision was taken because in the Bayer pattern, green pixels are twice as many as the red and blue ones, and contain double as much information [24]. The green color is also less affected by noise.

A simple method to apply pixel decimation is to multiply element-wise the matrix representing the frame by a Toeplitz matrix of alternating ones and zeros. In this way, red and blue pixels will be set to zero. As there are four possible Bayer patterns, the specific pattern must be specified as parameter for the given sequence.

Revised Search Window The assumption behind the algorithm is that a moving object is likely to continue the movement in the same direction detected in the previous iteration, or with a very small deviation. Additionally, human eyes are not well suited to detect clearly fast movements, therefore the priority is to apply denoising where the image is seen more clearly.

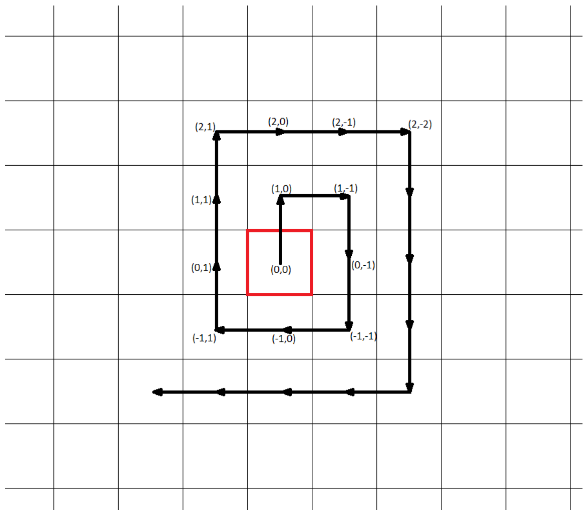

Figure 4.1: Revised concept of the search window for the new block matching algorithm.

To achieve this, we constructed a vector of offsets (or coordinates), that describes a spiral route around a central location, having coordinates $[0, 0]$, as shown in Figure 4.1. These offsets replace the search window, and will be used independently to each block of the frame. During the first iteration of the algorithm, which corresponds to the full search being applied to the first and second frames of the video sequence, each block of the first frame will be overlapped to its corresponding initial coordinates on the second frame. It then slides along the route described by the aforementioned offset vector, and the error function is calculated for each location. The offset corresponding to the lowest error will be taken as motion vector for that block and stored along with the other motion vectors.

Several expedients have been taken to assure the algorithm would run faster in Matlab, for example by keeping in memory some data structures, such as a structure containing the coordinates of every block, which can be calculated at the beginning given the frame and block size (which are kept constant throughout the whole operation).

Once the block matching process is completed, the motion vector list is returned and given as input to the denoising function along with the two subsequent frames that are being compared. The denoising operations will then be applied on the second frame. The refined image returned will be given as input for the following iteration of the block matching algorithm, and used as reference for the detection of motion vectors for the upcoming frame.

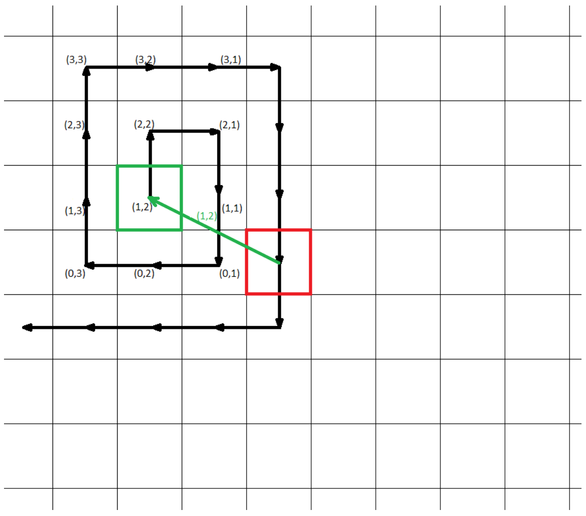

Figure 4.2: Predictive position of the new search window based on the computed motion vector.

In addition, the motion vectors detected in the previous iteration are given as input as well. As stated above, the assumption is that a moving object will keep moving along the direction of motion, or just slightly diverge. Therefore, before carrying out the computation of the error function, every block is translated by the corresponding motion vector calculated in the previous iteration, and the search window is centered around the predicted motion vector. This is accomplished by summing the motion vector to the offsets vector, translating the block to the corresponding coordinates.

As shown in Figure 4.2, if for example at iteration $t-1$ the motion vector calculated for the red block is equal to $[1, 2]$, during iteration t the block will be translated of one pixel vertically and two pixels horizontally, and the research will be performed from that initial position.

Undoubtedly, the longer previous motion vectors are kept track of, the more imprecise the prediction of movement might become. As a consequence, a predefined number of frames (in this case, 24 frames) is fixed, after which the input motion vector will be refreshed, starting the search from the block's original position. Furthermore, a check on the number of denoised pixels on each frame was considered, and if their number is detected to be below a certain threshold (that is set at 75%), an additional refresh of the motion vectors will be carried out.

4.2 Block Subsampling

Subsampling the number of blocks where to apply the block matching algorithm has already proven to return valuable results, as discussed in [25]. The authors note how it is not uncommon to find neighbouring blocks with identical or almost identical motion vectors, as blocks are in most of the cases smaller than the size of the moving object or of the motion field. In their research, they first estimate only a reduced number of the motion vectors by applying the search algorithm to only a fraction of the blocks, and for the remaining blocks the motion vectors are obtained by a suitable interpolation.

Figure 4.3 shows the subsampling pattern used by their algorithm: motion vectors are calculated for blocks *B, C, D, E*. The motion vector of block *A* is very likely to be similar to one of those other motion vectors. Therefore, to assign the motion vector to block A, they test the motion vectors of the adjacent four blocks, and they assign the one which returns the lowest MAD. The estimate will be accurate if block A fully contains an object contained also in any of the other blocks. However, if block A lies within an edge, the estimate might be less accurate.

The approach is inspired by this method, while also taking into account sequential capturing within the camera. When a frame is captured, pixels are usually recorded in a sequential manner horizontally, top to bottom. As the aim is to develop an algorithm that will be able to perform motion estimation almost in real time, the process cannot be delayed until all four adjacent blocks have been computed.

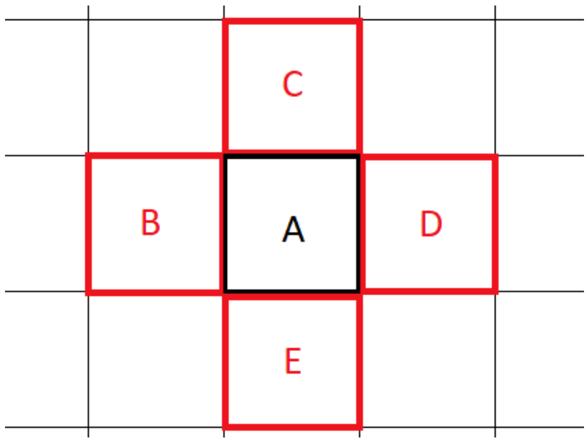

Figure 4.3: Block subsampling as described in [25].

Ideally, the motion estimation process will start already on the top left pixel while the remaining pixels are still being captured and registered in the memory. As a consequence, while the subsampled blocks where motion estimation is performed will eventually compose a checkerboard pattern, the algorithm will take into account, when interpolating with the remaining blocks, only the previous and the above block (as the subsequent ones have yet to be computed).

Hence, given the example shown in Figure 4.3, only blocks B and C will be taken into consideration when interpolating block A.

Similar considerations to those of the original algorithm can be done in this case too. In most cases, adjacent blocks will most likely be part of the same motion field or object, and only a small percentage will lie on edges or be part of a different body. Thus, the number of wrongly assigned motion vectors will most likely be negligible on a successful prediction.

4.3 Additional Subsampling

An additional subsampling technique that is carried out consists in assigning to alternate blocks the motion vectors calculated for the blocks directly adjacent to their left. Therefore, half of the blocks will compute the revisited motion estimation seen in Section 4.1, while the remaining ones will copy the motion vector assigned to the previous block.

Going back to Figure 4.3, this implies that the motion vector assigned to block A will coincide with the motion vector calculated for block B.

Once the assignment process is over, the algorithm proceeds to calculate the error of the remaining blocks relative to the selected motion vector.

4.4 Early Cut-Off

To additionally boost the speed of the algorithm, an early cut-off technique was introduced. When performing the block matching operation, the program will store the minimum error calculated along with assigning the corresponding motion vector to the block. Once all blocks of the frame are assigned, the average error is computed and passed on to the next iteration.

On the following round, when computing the objective function, the algorithm will interrupt the search for a minimum value in the search window in case the current error calculated is strictly less than the average error computed during the previous iteration. The motion vector assigned will correspond to the cut-off position of the block.

The rationale behind this idea is that subsequent frames are naturally similar one to the other, so the average prediction error will be similar too. Moreover, we exploit a natural consequence of the revised concept of search window that was described before: as movements within two subsequent frames tend to be small and uniform (as long as the capture rate is high enough), the predictive nature of the search algorithm, which already centers the search window on the predicted motion vector, can be further advanced by reducing the number of fruitless operations. Motion vectors within two subsequent frames will most likely be similar, and the introduction of early cut-offs takes advantage of this assumption.

It is also interesting to note that the introduction of early cut-offs based on the average error per pixel of the previous estimation should reduce the quality of the prediction only by a negligible amount: even in worst cases where the best match would be subsequent to the cut-off index, the error will still be less than the average of the previous prediction, granting an acceptable quality for the application that the algorithm is aimed to.

5 Results

In this section a variety of results obtained by running the algorithms on the test sequences are presented. The two objective functions SSD and SAD are compared to ascertain the absence of any substantial difference; the performance of the algorithms is analyzed by measuring the motion compensation error (in terms of MSE per pixel) and by reviewing whether the amount of denoised pixels improves when applying motion estimation, compared to the lack of any prediction. A statistical evaluation of the motion vectors that were computed is also presented.

5.1 Video Sequences Description

Firstly, we introduce a brief description of the video sequences used for testing our algorithms. A wide collection of sequences was employed, and they were selected to exhibit different visual characteristics, i.e, pans of outdoor and indoor environments, people walking, objects moving, zoom-in and zoom-out, and brightness changes. Each of these characteristics can influence in a positive or negative manner the motion estimation and the subsequent denoising.

- Akademie: outdoor horizontal pan of the Art Akademy in Munich, with some trees and people walking. 240 frames - 1620 × 2880 px.

- Balls: indoor horizontal panoramic of a toy ball pit room. Illumination is scarce and fading. Useful to study the robustness of the algorithms against noise. 100 frames - 1620 × 2880 px.

- Boys Running: horizontal panoramic of two children running in a dark environment. 70 frames - 1856 × 4448 px.

- Boys Villa: two children walking in an indoor environment, with zooming and panoramic motion. 84 frames - 3096 × 4448 px.

- Eye: green screen close-up on a face showing the right eye blinking, with many light colored hair around. There is a slight horizontal movement of the camera. 8 frames - 1620 × 2880 px.

- Fire Department: still camera pointing on a fire being extinguished by a water jet. Useful for fast and random movement. 72 frames - 2592 × 4608 px.

- Girl: moving side view of the face of a girl with out of focus lights in the background. The sequence is almost still, so very little movement should be detected. 84 frames - 3096 × 4448 px.

- Lake: vertical pan of a snowy tree with a lake in the background. This is the most constant movement sequence. 48 frames - 1620 × 2880 px.

- Snowy Landscape: two lengthy sequences of a snowy environment and a car crossing through it, with pan movement, zoom-in and zoom-out, and an overall uniform light colour due to the snow. Useful to test the robustness of the motion estimation against very fast movements. 428 frames - 3164 × 4608 px.

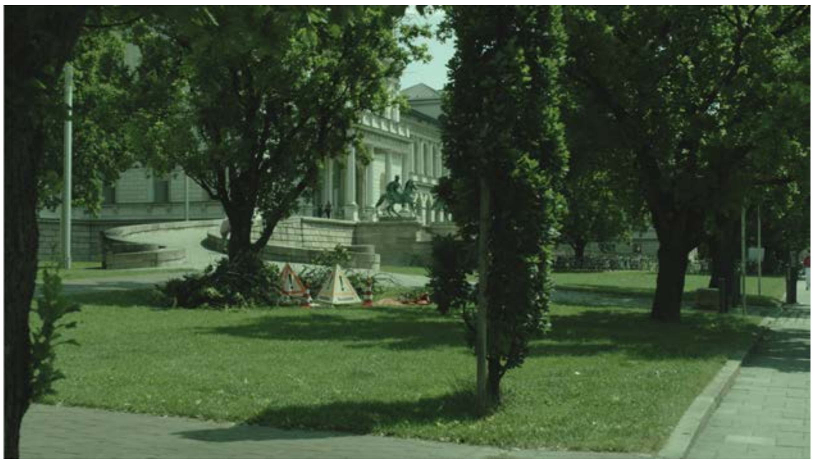

Figure 5.1: Frame of the Akademie sequence.

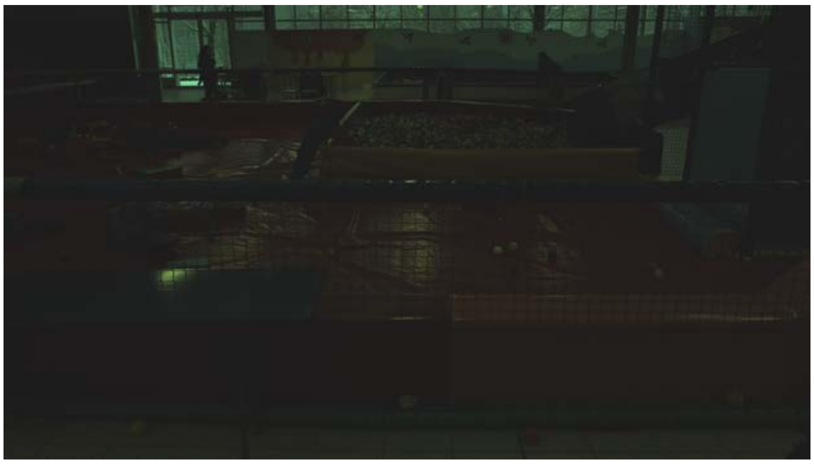

Figure 5.2: Frame of the Balls sequence.

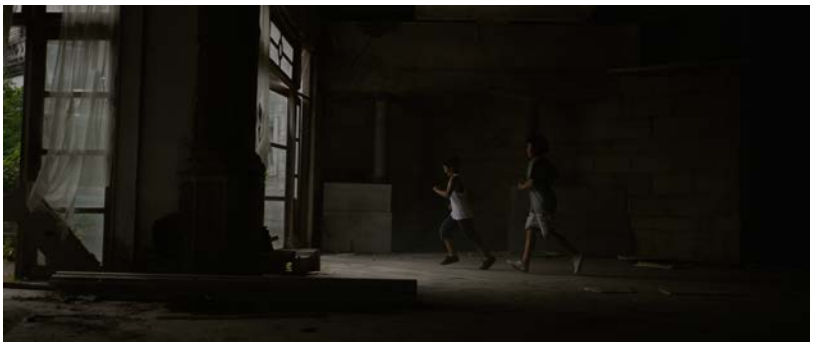

Figure 5.3: Frame of the Boys Running sequence.

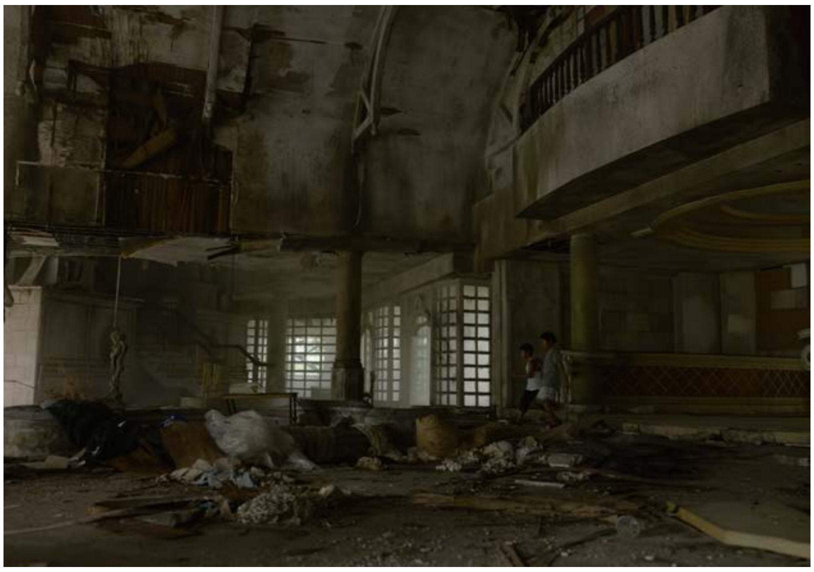
Figure 5.4: Frame of the Boys Villa sequence.

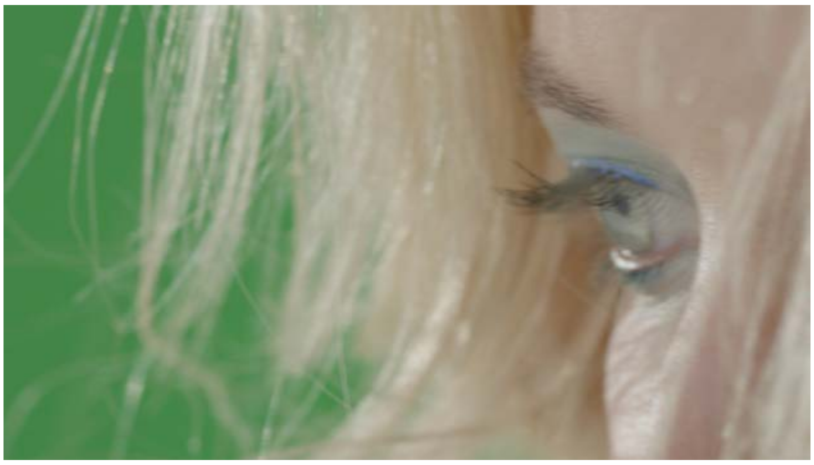
Figure 5.5: Frame of the Eye sequence.

Figure 5.6: Frame of the Fire Department sequence.

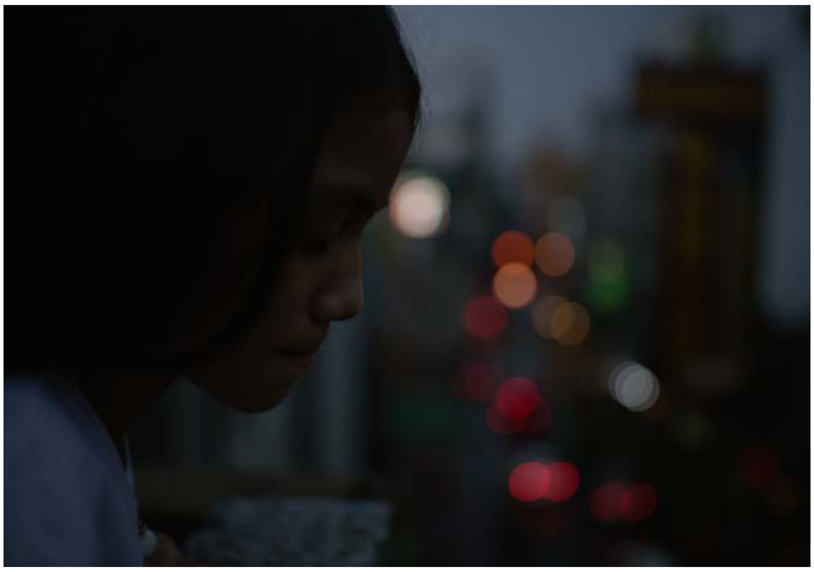

Figure 5.7: Frame of the Girl sequence.

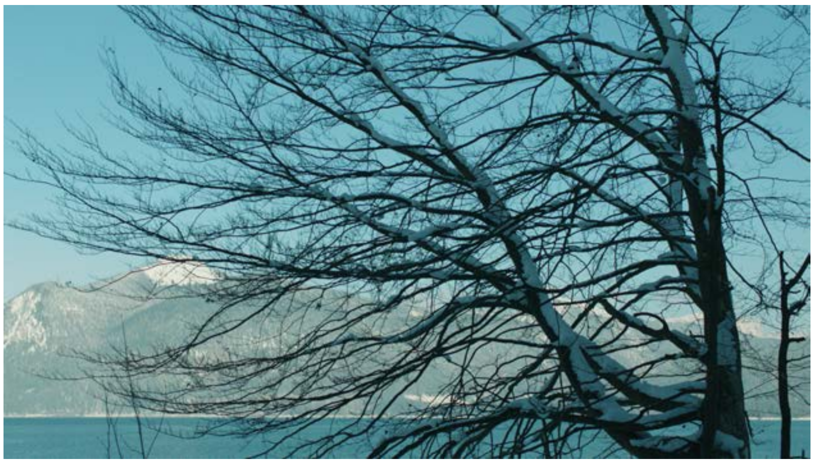
Figure 5.8: Frame of the Lake sequence.

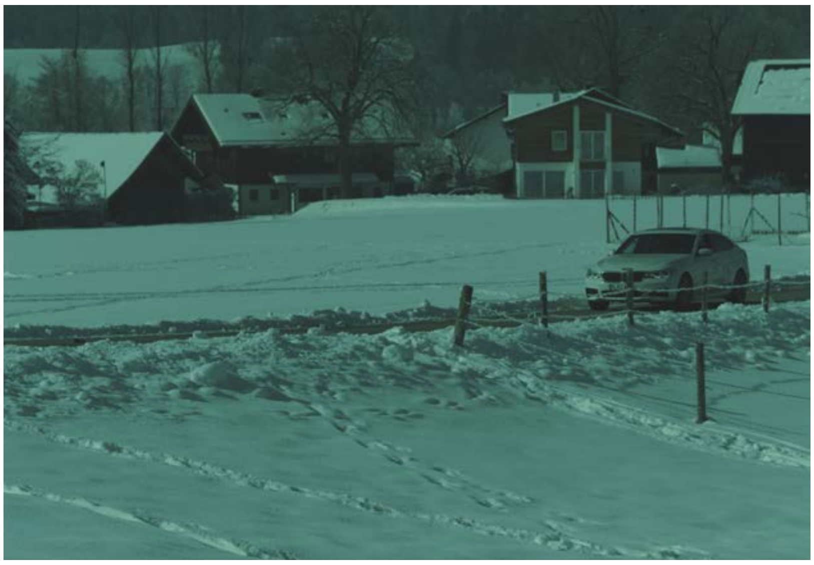
Figure 5.9: Frame of the Snowy Landscape sequence.

5.2 Results

Objective Functions Performance Comparison The first results that were compared were the differences in denoising rates between the two objective functions, in order to verify whether significant differences were present that would result in one being inherently better than the other.

Tables 5.1 to 5.7 report the average difference and the ratio with respect to the size of the frame between the number of denoised pixels obtained by using the SSD and SAD objective functions applied to the different algorithms. Positive results denote a higher denoising rates for the SAD function, while negatives imply a majority for the SSD.

As it can be seen, even though the average number of denoised pixels can vary quite dramatically between sequences (due to the different size of the sequences), the resulting rates are very similar: the difference between the two functions is negligible, with a variation of less than a 0.1% in most cases. As a matter of fact, when applying the algorithm with early cut-offs, there seems to be an increase in denoising rate of up to 6.0% for the SAD function, making it the most viable option.

The increase in the denoising rate can especially be seen on the Boys Villa sequence: the indoor panoramic was filmed with the camera rotating around its axis, giving a zooming effect on objects getting closer. Zooms reduce the capability of the block matching algorithm to detect the movement, increasing the prediction error and, as a consequence, its average. As the SSD is non linear because of the square operation, the cut-off operation has a higher chance of hitting earlier. In fact, it requires on average two iterations less than by using the SAD function. However, it is still preferable to use the SAD, as the increase of the number of iterations is negligible compared to the improvement obtained. Furthermore, this is merely a particular case due to the filming procedure, and generally results are very similar, both in denoising ratios and number of iterations.

Table 5.1: Average difference of the number of denoised pixels and ratio wrt the size of the frame by using the SSD and SAD objective functions on the full search algorithm with no decimation ($[px] \times 10^3$).

	9×9		9×11		11×13		16×16	
	Diff. [px]	**Ratio**	**Diff.** [px]	**Ratio**	**Diff.** [px]	**Ratio**	**Diff.** [px]	**Ratio**
Akademie	9.5	0.002	9.5	0.002	9.5	0.002	8.5	0.001
Balls	-9.0	0.002	-9.0	0.002	-4.5	0.001	-2.0	0.001
Boys Running	-13.0	0.002	-12.0	0.002	-11	0.002	-8.7	0.001
Boys Villa	-23.0	0.002	-19.0	0.001	-14	0.001	-5.8	< 0.001
Eye	-4.2	0.001	-4.1	0.001	-3.4	0.001	-2.5	0.001
Fire Department	21.0	0.002	23.0	0.002	28.0	0.003	34.0	0.003
Girl	-6.7	< 0.001	-7.7	< 0.001	-8.6	0.001	-10	0.001
Lake	-0.7	< 0.001	0.7	< 0.001	2.4	0.001	4.8	0.002
Snowy Landscape	-3.4	< 0.001	-2.9	< 0.001	-1.1	< 0.001	2.9	< 0.001

Table 5.2: Average difference of the number of denoised pixels and ratio wrt the size of the frame by using the SSD and SAD objective functions on the full search algorithm with pixel decimation ($[px] \times 10^3$).

	9×9		9×11		11×13		16×16	
	Diff. [px]	**Ratio**	**Diff.** [px]	**Ratio**	**Diff.** [px]	**Ratio**	**Diff.** [px]	**Ratio**
Akademie	9.2	0.002	9.7	0.002	9.1	0.002	8.4	0.002
Balls	-9	0.002	-8.8	0.002	-7.2	0.002	-3.9	< 0.001
Boys Running	-11	0.002	-11	0.002	-10	0.002	-7.8	0.001
Boys Villa	-22	0.002	-21	0.001	-16	0.001	-6.3	0.001
Eye	-2.9	< 0.001	-2.9	< 0.001	-3.1	0.001	-1.4	< 0.001
Fire Department	9.9	0.001	12	0.001	16	0.002	23	0.002
Girl	-2.6	< 0.001	-4.3	< 0.001	-5.6	< 0.001	-6.7	0.001
Lake	-0.7	< 0.001	0.3	< 0.001	3.3	0.001	5.1	0.001
Snowy Landscape	-6.8	< 0.001	-6.5	< 0.001	-7.5	< 0.001	0.1	< 0.001

Table 5.3: Average difference of the number of denoised pixels and ratio wrt the size of the frame by using the SSD and SAD objective functions on the full search algorithm with pixel decimation and with early cut-offs ($[px] \times 10^3$).

	9×9		9×11		11×13		16×16	
	Diff. [px]	**Ratio**	**Diff.** [px]	**Ratio**	**Diff.** [px]	**Ratio**	**Diff.** [px]	**Ratio**
Akademie	21	0.004	21	0.004	17	0.003	14	0.003
Balls	52	0.011	-51	0.011	-46	0.010	40	0.009
Boys Running	120	0.015	130	0.016	140	0.016	130	0.016
Boys Villa	720	0.052	730	0.053	750	0.054	750	0.054
Eye	45	< 0.010	46	< 0.010	47	0.010	49	< 0.010
Fire Department	94	0.008	92	0.008	86	0.007	77	0.007
Girl	10	0.001	11	0.001	14	< 0.001	16	0.001
Lake	46	0.010	40	0.009	35	0.007	30	0.006
Snowy Landscape	31	0.002	36	0.003	45	0.003	40	0.003

Table 5.4: Average difference of the number of denoised pixels and ratio wrt the size of the frame by using the SSD and SAD objective functions on the block matching algorithm with pixel decimation and block subsampling ($[px] \times 10^3$).

	9×9		9×11		11×13		16×16	
	Diff. [px]	**Ratio**	**Diff.** [px]	**Ratio**	**Diff.** [px]	**Ratio**	**Diff.** [px]	**Ratio**
Akademie	6.2	0.001	6.7	0.001	6.3	0.001	6.2	0.001
Balls	-7.9	0.002	-7.1	0.001	-4.7	0.001	-2.2	< 0.001
Boys Running	-10	0.001	-11	0.002	-8.2	0.001	-7.0	0.001
Boys Villa	-22	0.002	-19	0.001	-15	0.001	-7.4	< 0.001
Eye	-1.9	< 0.001	-2.0	< 0.001	-4.3	0.001	-2.3	< 0.001
Fire Department	37	0.001	5.2	< 0.001	77	0.001	13	0.001
Girl	-0.6	< 0.001	-2.7	< 0.001	-2.5	< 0.001	-3.5	< 0.001
Lake	-1.8	< 0.001	-0.5	< 0.001	2.2	< 0.001	3.6	< 0.001
Snowy Landscape	-6.4	< 0.001	-5.3	< 0.001	-3.4	< 0.001	-0.5	< 0.001

Table 5.5: Average difference of the number of denoised pixels and ratio wrt the size of the frame by using the SSD and SAD objective functions on the block matching algorithm with pixel decimation, block subsampling and with early cut-offs ($[px] \times 10^3$).

	9×9		9×11		11×13		16×16	
	Diff. [px]	**Ratio**	**Diff.** [px]	**Ratio**	**Diff.** [px]	**Ratio**	**Diff.** [px]	**Ratio**
Akademie	23	0.004	21	0.004	17	0.003	13	0.003
Balls	16	0.003	23	0.005	19	0.004	31	0.007
Boys Running	140	0.015	150	0.016	160	0.019	160	0.016
Boys Villa	830	0.060	850	0.062	900	0.067	950	0.069
Eye	59	0.013	64	0.014	68	0.015	75	0.016
Fire Department	75	0.006	74	0.006	74	0.006	69	0.006
Girl	-0.3	< 0.001	0.2	< 0.001	0.9	< 0.001	3.1	0.001
Lake	41	0.009	39	0.008	38	0.008	35	0.008
Snowy Landscape	43	0.003	58	0.004	73	0.005	89	0.006

Table 5.6: Average difference of the number of denoised pixels and ratio wrt the size of the frame by using the SSD and SAD objective functions on the block matching algorithm with pixel decimation and the additional block subsampling ($[px] \times 10^3$).

	9×9		9×11		11×13		16×16	
	Diff. [px]	**Ratio**	**Diff.** [px]	**Ratio**	**Diff.** [px]	**Ratio**	**Diff.** [px]	**Ratio**
Akademie	8.4	0.002	9.1	0.002	8.8	0.002	9.1	0.002
Balls	-8.4	0.002	-7.0	0.002	-8.2	0.002	-4.7	0.001
Boys Running	-11	0.001	-11	0.001	-9.3	0.001	-6.2	0.001
Boys Villa	-23	0.002	-22	0.002	-17	0.001	-9.3	0.001
Eye	-2.8	0.001	-3.3	0.001	-3.4	0.001	-0.7	< 0.001
Fire Department	4.9	< 0.001	5.8	< 0.001	9.3	0.001	14	0.001
Girl	-0.9	< 0.001	-0.6	< 0.001	-1.6	< 0.001	-2.4	< 0.001
Lake	-3.6	0.001	-3.6	0.001	1.1	< 0.001	2.1	< 0.001
Snowy Landscape	-7.5	0.001	-7.3	0.001	-5.6	< 0.001	-1.7	< 0.001

Table 5.7: Average difference of the number of denoised pixels and ratio wrt the size of the frame by using the SSD and SAD objective functions on the block matching algorithm with pixel decimation, the additional subsampling and early cut-off ($[px] \times 10^3$).

	9×9		9×11		11×13		16×16	
	Diff. [px]	**Ratio**	**Diff.** [px]	**Ratio**	**Diff.** [px]	**Ratio**	**Diff.** [px]	**Ratio**
Akademie	24	0.005	25	0.006	21	0.005	14	0.003
Balls	54	0.012	51	0.011	49	0.011	38	0.008
Boys Running	170	0.021	170	0.021	180	0.022	180	0.022
Boys Villa	710	0.051	730	0.053	750	0.054	770	0.056
Eye	52	0.011	51	0.011	52	0.011	51	0.011
Fire Department	69	0.006	65	0.005	60	0.005	53	0.004
Girl	44	0.003	24	0.002	42	0.003	86	0.006
Lake	69	0.015	67	0.015	61	0.013	55	0.012
Snowy Landscape	130	0.009	50	0.003	64	0.004	140	0.010

Plots for the full search algorithm with no decimation applied to the Akademie, Fire Department and Boys Villa sequences are shown on Figure 5.10, which visually displays the negligible difference between the functions, even though the denoising rates are very different. The Fire Department sequence (black line) has the lowest denoising rate due to its randomness, while the Akademie and Boys Villa sequences are very similar. Figure 5.11 shows how early cut-offs obtain better results with the SAD function, as discussed earlier. A similar phenomenon can be observed also on Figure 5.12, although the difference is smaller.

Figures 5.13 and 5.14 display the variation on the Eye sequence when applying the block matching algorithm with pixel decimation and block subsampling with and without early cut-off, which again return better results for the SAD in the latter case, even though the improvement is less noticeable. Moreover, the block subsampling operation did not introduce any relevant loss in the amount of denoised pixels.

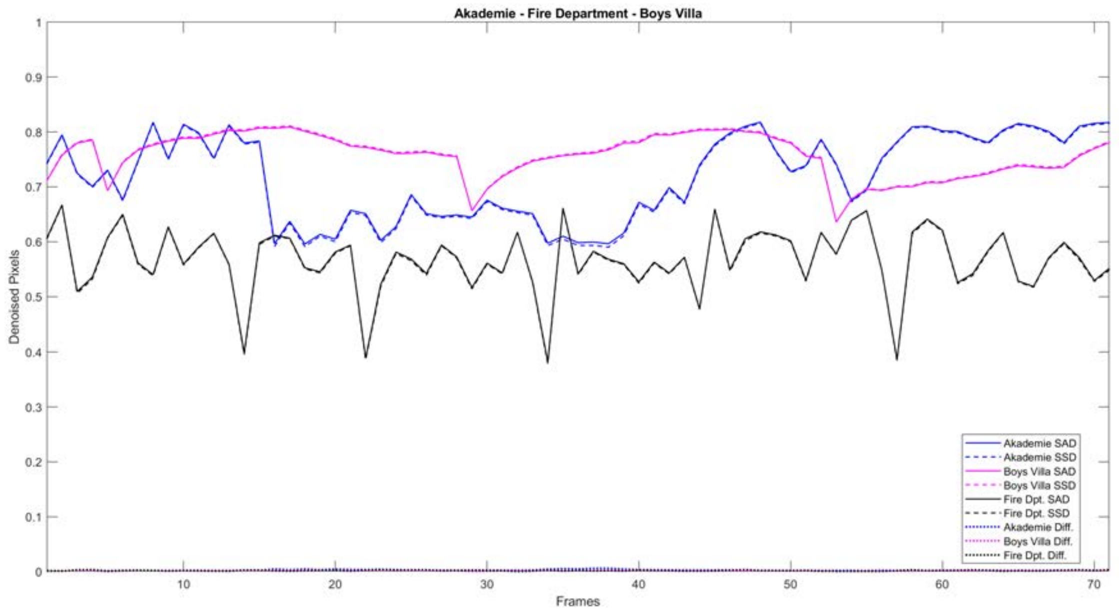

Figure 5.10: Denoised pixels per frame for the Akademie, Boys Villa and Fire Department sequences, on the full search algorithm with pixel decimation, and a block size of 9×9 px. The denoising rates are different for each sequence, with Fire Department having the worst performance, but the difference between the two metrics is negligible in all cases.

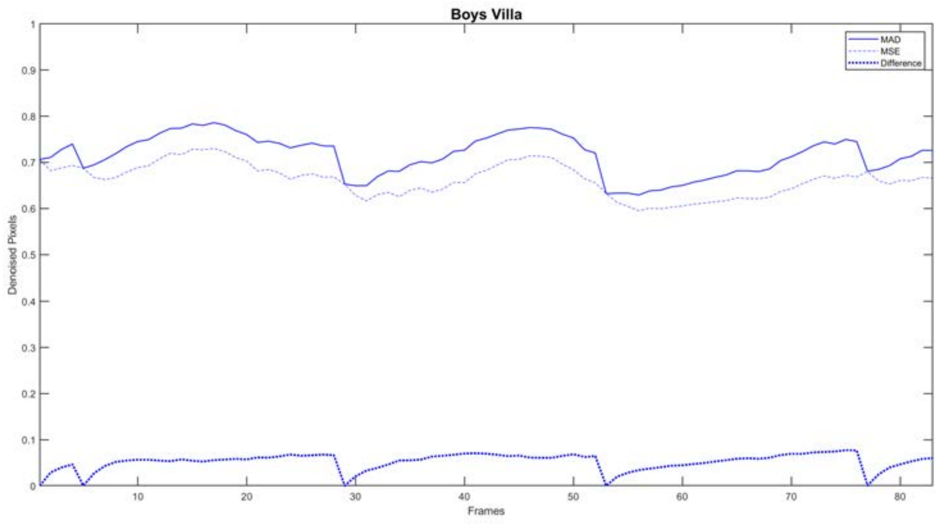

Figure 5.11: Denoised pixels per frame for the Boys Villa sequence, on the full search algorithm with pixel decimation and early cut-off, and a block size of 9×9 px. The introduction of early cut-offs improved the results obtained with the SAD function.

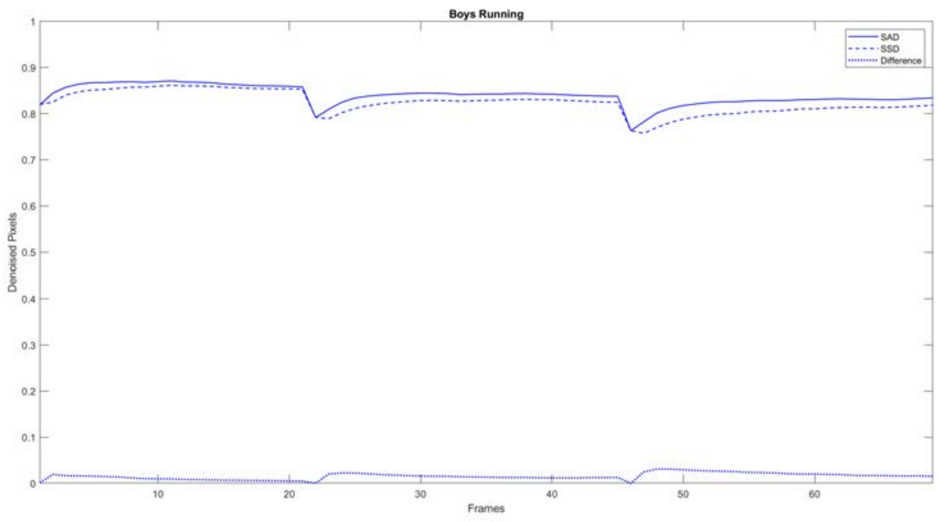

Figure 5.12: Denoised pixels per frame for the Boys Running sequence, on the full search algorithm with pixel decimation and early cut-off, and a block size of 9×9 px. As for Figure 5.11, the introduction of early cut-offs returned better results with the SAD function.

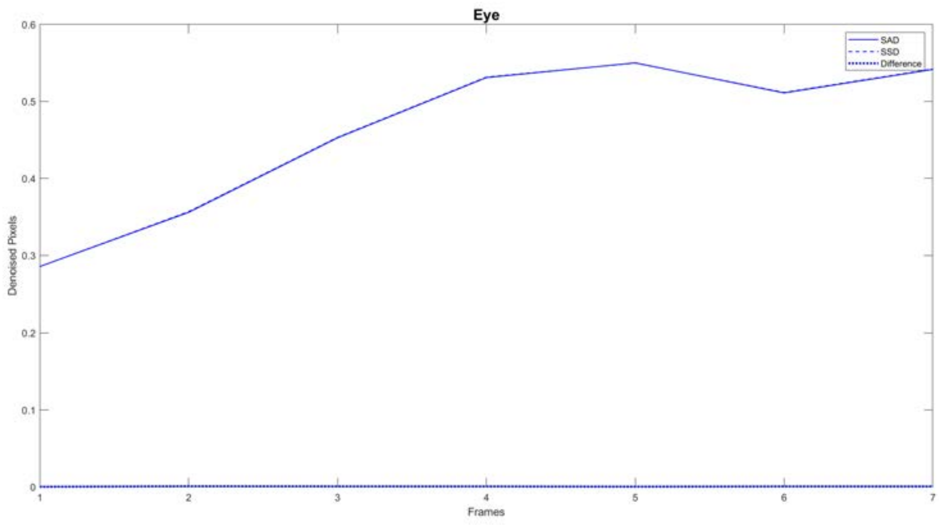

Figure 5.13: Denoised pixels per frame for the Eye sequence, when applying pixel decimation and block subsampling, and a block size of 9×9 px. The difference is negligible.

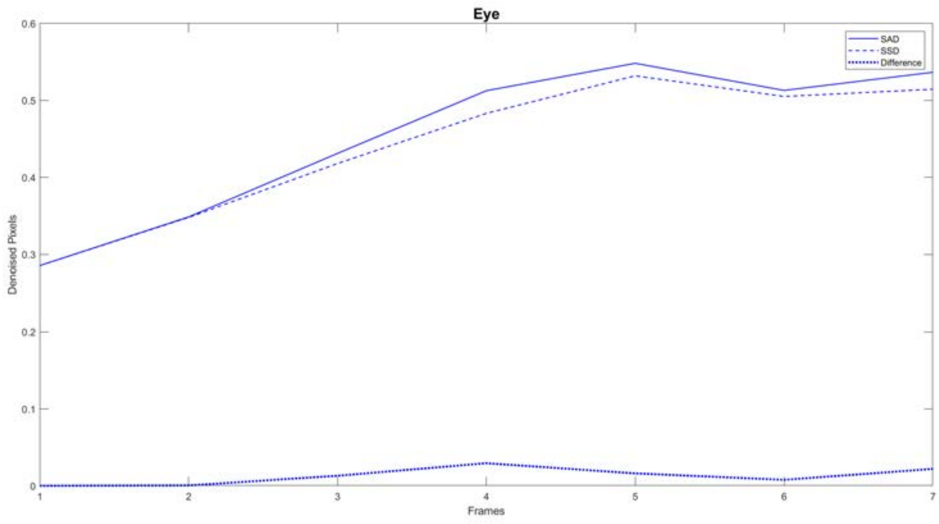

Figure 5.14: Denoised pixels per frame for the Eye sequence, when applying pixel decimation, block subsampling and early cut-off, and a block size of 9×9 px. Results obtained with the SAD are slightly better than with the SSD function.

Denoising Improvement Since it was found that the SAD is the preferable objective function, only the results obtained with this metric are discussed in the following paragraph. The ground truth to the improvement obtained through the block matching algorithms is given by the denoising rate with no motion estimation.

Tables 5.8 to 5.14 report the denoising rates obtained by applying the proposed algorithms. The best results (that is, the best improvements compared to denoising with no motion estimation) are given by sequences with gradual and slow movements, as it was anticipated earlier, such as *Akademie* and *Lake*, with an average gain of almost 30%; the performance remains stable when changing the block size. Reasonable outcomes are returned when moving subjects are present too, like in *Boys Running* and *Boys Villa*, with a slight gain as the block size increases. On the contrary, accuracy tends to be reduced when motion is faster and irregular, as it can be seen for example on the *Eye, Fire Department* and *Snowy Landscape* sequences, with less than 10% gain. As this sequence is very long, it presents a wide variety of features that affect motion estimation, which are highlighted in the tables by the huge difference between the minimum and the maximum denoising rates achieved. Moreover, when there is little or no movement, such as in *Girl*, there is just a slight improvement from the ground truth, as motion estimation is minor. Nevertheless, performance seems to improve with bigger block sizes.

An interesting result is given by the *Balls* sequence: introducing early cut-offs seems to improve the denoising rates. As the sequence becomes darker, it is possible that early cut-offs increase the robustness of the algorithms to noise brought by the lack of light, making the predictions more precise.

The average denoising rates of all the algorithms are also reported on Table 5.15, for a clear comparison between the approaches.

Bigger block sizes facilitate the detection of certain artifacts such as zooms or fast moving objects. This is why non panoramic sequences profited from larger blocks with better denoising rates. Pans, on the other hand, did not present any noticeable improvement, as they already benefit of a smooth and easy to detect movement.

Some of the most meaningful results are also presented on the plots of Figures 5.15 to 5.18, in order to provide a graphical representation. Figure 5.15, in particular, shows a drastic improvement of the average denoising rate, except for an initial downfall, which is due to a camera vibration as the panoramic movement starts. Similar kinds of noise may dramatically reduce the quality of the prediction and denoising.

Table 5.8: Comparison between the denoising rate without any motion estimation and with the full search algorithm with no pixel decimation.

		No BM	9×9	9×11	11×13	16×16
Akademie	**min**	0.37	0.59	0.60	0.59	0.59
	max	0.61	0.83	0.83	0.83	0.83
	avg	0.45	0.73	0.73	0.73	0.73
Balls	**min**	0.72	0.78	0.79	0.80	0.80
	max	0.86	0.90	0.90	0.91	0.91
	avg	0.77	0.82	0.82	0.83	0.84
Boys Running	**min**	0.70	0.77	0.77	0.77	0.77
	max	0.79	0.88	0.88	0.89	0.90
	avg	0.74	0.85	0.86	0.86	0.87
Boys Villa	**min**	0.56	0.64	0.64	0.64	0.64
	max	0.62	0.81	0.82	0.83	0.84
	avg	0.59	0.76	0.77	0.78	0.79
Eye	**min**	0.28	0.29	0.29	0.29	0.29
	max	0.50	0.57	0.57	0.58	0.59
	avg	0.42	0.48	0.48	0.48	0.49
Fire Department	**min**	0.35	0.38	0.38	0.38	0.38
	max	0.66	0.67	0.67	0.67	0.67
	avg	0.55	0.57	0.57	0.57	0.57
Girl	**min**	0.88	0.90	0.91	0.91	0.91
	max	0.91	0.94	0.94	0.94	0.94
	avg	0.90	0.92	0.92	0.92	0.93
Lake	**min**	0.34	0.46	0.46	0.46	0.45
	max	0.42	0.83	0.84	0.85	0.86
	avg	0.37	0.65	0.66	0.66	0.66
Snowy Landscape	**min**	0.26	0.26	0.26	0.26	0.26
	max	0.96	0.95	0.95	0.95	0.95
	avg	0.64	0.67	0.68	0.68	0.69

Table 5.9: Comparison between the denoising rate without any motion estimation and with the full search algorithm with pixel decimation.

		No BM	9×9	9×11	11×13	16×16
Akademie	**min**	0.37	0.59	0.59	0.59	0.59
	max	0.61	0.83	0.83	0.83	0.83
	avg	0.45	0.73	0.73	0.73	0.73
Balls	**min**	0.72	0.77	0.77	0.77	0.80
	max	0.86	0.90	0.90	0.90	0.91
	avg	0.77	0.81	0.81	0.82	0.83
Boys Running	**min**	0.70	0.76	0.76	0.77	0.76
	max	0.79	0.88	0.88	0.89	0.90
	avg	0.74	0.85	0.85	0.86	0.87
Boys Villa	**min**	0.56	0.63	0.64	0.64	0.64
	max	0.62	0.80	0.81	0.83	0.84
	avg	0.59	0.75	0.76	0.77	0.79
Eye	**min**	0.28	0.29	0.29	0.29	0.29
	max	0.50	0.57	0.57	0.57	0.58
	avg	0.42	0.47	0.48	0.48	0.49
Fire Department	**min**	0.35	0.38	0.38	0.38	0.38
	max	0.66	0.66	0.66	0.67	0.67
	avg	0.55	0.56	0.56	0.57	0.58
Girl	**min**	0.88	0.90	0.90	0.90	0.91
	max	0.91	0.93	0.93	0.94	0.94
	avg	0.90	0.92	0.92	0.92	0.92
Lake	**min**	0.34	0.46	0.46	0.46	0.45
	max	0.42	0.83	0.84	0.85	0.85
	avg	0.37	0.65	0.65	0.66	0.66
Snowy Landscape	**min**	0.26	0.26	0.26	0.26	0.26
	max	0.96	0.94	0.94	0.94	0.95
	avg	0.64	0.67	0.67	0.68	0.68

Table 5.10: Comparison between the denoising rate without any motion estimation and with the full search algorithm with pixel decimation and early cut-off.

		No BM	9×9	9×11	11×13	16×16
Akademie	**min**	0.37	0.56	0.56	0.56	0.56
	max	0.61	0.82	0.82	0.82	0.83
	avg	0.45	0.73	0.73	0.73	0.73
Balls	**min**	0.72	0.78	0.78	0.79	0.79
	max	0.86	0.90	0.90	0.90	0.91
	avg	0.77	0.82	0.82	0.82	0.83
Boys Running	**min**	0.70	0.76	0.76	0.77	0.77
	max	0.79	0.87	0.88	0.88	0.89
	avg	0.74	0.84	0.84	0.85	0.85
Boys Villa	**min**	0.56	0.63	0.63	0.64	0.64
	max	0.62	0.79	0.79	0.80	0.81
	avg	0.59	0.72	0.72	0.73	0.73
Eye	**min**	0.28	0.29	0.29	0.29	0.29
	max	0.50	0.53	0.53	0.53	0.54
	avg	0.42	0.44	0.44	0.44	0.45
Fire Department	**min**	0.35	0.36	0.36	0.36	0.35
	max	0.66	0.66	0.66	0.66	0.65
	avg	0.55	0.55	0.55	0.55	0.55
Girl	**min**	0.88	0.90	0.91	0.91	0.91
	max	0.91	0.93	0.93	0.94	0.94
	avg	0.90	0.92	0.92	0.92	0.92
Lake	**min**	0.34	0.46	0.46	0.46	0.45
	max	0.42	0.83	0.84	0.84	0.84
	avg	0.37	0.64	0.64	0.65	0.65
Snowy Landscape	**min**	0.26	0.26	0.26	0.26	0.26
	max	0.96	0.94	0.94	0.94	0.95
	avg	0.64	0.67	0.67	0.67	0.67

Table 5.11: Comparison between the denoising rate without any motion estimation and with the block matching algorithm with pixel decimation and block subsampling.

		No BM	9×9	9×11	11×13	16×16
Akademie	**min**	0.37	0.59	0.59	0.59	0.59
	max	0.61	0.83	0.83	0.83	0.83
	avg	0.45	0.73	0.73	0.73	0.73
Balls	**min**	0.72	0.77	0.78	0.79	0.80
	max	0.86	0.90	0.90	0.90	0.91
	avg	0.77	0.81	0.82	0.83	0.84
Boys Running	**min**	0.70	0.76	0.76	0.77	0.77
	max	0.79	0.88	0.88	0.89	0.90
	avg	0.74	0.85	0.85	0.86	0.88
Boys Villa	**min**	0.56	0.63	0.64	0.64	0.64
	max	0.62	0.80	0.81	0.82	0.83
	avg	0.59	0.75	0.76	0.77	0.78
Eye	**min**	0.28	0.29	0.29	0.29	0.29
	max	0.50	0.55	0.55	0.56	0.56
	avg	0.42	0.46	0.46	0.47	0.47
Fire Department	**min**	0.35	0.37	0.37	0.37	0.37
	max	0.66	0.66	0.66	0.66	0.66
	avg	0.55	0.55	0.56	0.56	0.56
Girl	**min**	0.88	0.90	0.90	0.90	0.91
	max	0.91	0.93	0.93	0.94	0.94
	avg	0.90	0.91	0.92	0.92	0.92
Lake	**min**	0.34	0.46	0.46	0.46	0.45
	max	0.42	0.84	0.85	0.85	0.86
	avg	0.37	0.65	0.65	0.65	0.66
Snowy Landscape	**min**	0.26	0.26	0.26	0.26	0.26
	max	0.96	0.94	0.94	0.94	0.95
	avg	0.64	0.66	0.67	0.67	0.67

Table 5.12: Comparison between the denoising rate without any motion estimation and with the block matching algorithm with pixel decimation, block subsampling and early cut-off.

		No BM	9×9	9×11	11×13	16×16
Akademie	**min**	0.37	0.58	0.58	0.58	0.58
	max	0.61	0.83	0.83	0.83	0.83
	avg	0.45	0.73	0.73	0.73	0.73
Balls	**min**	0.72	0.79	0.79	0.80	0.80
	max	0.86	0.90	0.90	0.90	0.91
	avg	0.77	0.82	0.83	0.83	0.84
Boys Running	**min**	0.70	0.76	0.76	0.77	0.77
	max	0.79	0.88	0.88	0.89	0.89
	avg	0.74	0.85	0.85	0.86	0.86
Boys Villa	**min**	0.56	0.63	0.63	0.63	0.64
	max	0.62	0.80	0.81	0.82	0.83
	avg	0.59	0.74	0.75	0.76	0.77
Eye	**min**	0.28	0.29	0.29	0.29	0.29
	max	0.50	0.55	0.55	0.56	0.56
	avg	0.42	0.45	0.46	0.46	0.46
Fire Department	**min**	0.35	0.36	0.36	0.36	0.36
	max	0.66	0.66	0.66	0.66	0.65
	avg	0.55	0.55	0.55	0.55	0.55
Girl	**min**	0.88	0.90	0.90	0.90	0.91
	max	0.91	0.93	0.93	0.94	0.94
	avg	0.90	0.92	0.92	0.92	0.92
Lake	**min**	0.34	0.46	0.46	0.46	0.45
	max	0.42	0.84	0.84	0.85	0.85
	avg	0.37	0.64	0.64	0.65	0.65
Snowy Landscape	**min**	0.26	0.26	0.26	0.26	0.26
	max	0.96	0.94	0.94	0.94	0.95
	avg	0.64	0.66	0.66	0.67	0.67

Table 5.13: Comparison between the denoising rate without any motion estimation and with the block matching algorithm with pixel decimation and ulterior block subsampling.

		No BM	9×9	9×11	11×13	16×16
Akademie	**min**	0.37	0.58	0.58	0.59	0.58
	max	0.61	0.82	0.82	0.82	0.82
	avg	0.45	0.73	0.73	0.73	0.73
Balls	**min**	0.72	0.76	0.76	0.77	0.78
	max	0.86	0.90	0.90	0.90	0.91
	avg	0.77	0.80	0.80	0.81	0.82
Boys Running	**min**	0.70	0.76	0.76	0.77	0.77
	max	0.79	0.87	0.87	0.88	0.89
	avg	0.74	0.84	0.84	0.85	0.86
Boys Villa	**min**	0.56	0.63	0.64	0.64	0.64
	max	0.62	0.80	0.80	0.82	0.83
	avg	0.59	0.75	0.75	0.77	0.78
Eye	**min**	0.28	0.29	0.29	0.29	0.29
	max	0.50	0.57	0.57	0.57	0.58
	avg	0.42	0.47	0.48	0.48	0.48
Fire Department	**min**	0.35	0.37	0.37	0.37	0.37
	max	0.66	0.66	0.66	0.66	0.66
	avg	0.55	0.56	0.56	0.56	0.56
Girl	**min**	0.88	0.90	0.90	0.90	0.91
	max	0.91	0.93	0.93	0.93	0.94
	avg	0.90	0.91	0.92	0.92	0.92
Lake	**min**	0.34	0.46	0.46	0.46	0.45
	max	0.42	0.83	0.83	0.84	0.85
	avg	0.37	0.64	0.65	0.65	0.66
Snowy Landscape	**min**	0.26	0.26	0.26	0.26	0.26
	max	0.96	0.94	0.94	0.94	0.95
	avg	0.64	0.67	0.67	0.68	0.68

Table 5.14: Comparison between the denoising rate without any motion estimation and with the block matching algorithm with pixel decimation, ulterior block subsampling and early cut-off.

		No BM	9×9	9×11	11×13	16×16
Akademie	**min**	0.37	0.55	0.56	0.56	0.56
	max	0.61	0.82	0.82	0.82	0.82
	avg	0.45	0.73	0.73	0.73	0.73
Balls	**min**	0.72	0.77	0.77	0.78	0.78
	max	0.86	0.90	0.90	0.90	0.91
	avg	0.77	0.81	0.81	0.82	0.82
Boys Running	**min**	0.70	0.76	0.77	0.77	0.77
	max	0.79	0.86	0.87	0.87	0.88
	avg	0.74	0.853	0.84	0.84	0.85
Boys Villa	**min**	0.56	0.62	0.62	0.63	0.63
	max	0.62	0.77	0.78	0.78	0.79
	avg	0.59	0.70	0.71	0.71	0.72
Eye	**min**	0.28	0.29	0.29	0.29	0.29
	max	0.50	0.53	0.53	0.54	0.54
	avg	0.42	0.44	0.44	0.45	0.45
Fire Department	**min**	0.35	0.35	0.35	0.35	0.35
	max	0.66	0.66	0.66	0.66	0.65
	avg	0.55	0.54	0.54	0.54	0.54
Girl	**min**	0.88	0.90	0.90	0.91	0.91
	max	0.91	0.93	0.93	0.93	0.94
	avg	0.90	0.91	0.92	0.92	0.92
Lake	**min**	0.34	0.46	0.46	0.46	0.45
	max	0.42	0.83	0.83	0.84	0.84
	avg	0.37	0.63	0.63	0.64	0.64
Snowy Landscape	**min**	0.26	0.26	0.26	0.26	0.26
	max	0.96	0.94	0.94	0.94	0.95
	avg	0.64	0.66	0.66	0.67	0.67

Table 5.15: Overview table where the average denoising rates of all the algorithms, with a block size of 16×16, are summarized, for a clear comparison between them. The first Table shows the results without subsampling, while the results with subsampling are reported on the second Table.

	No BM	**Full Search No Pixel Decimation**	**Full Search With Px. Decimation**	**Full Search + Early Cut-offs**
Akademie	0.45	0.73	0.73	0.73
Balls	0.77	0.84	0.83	0.83
Boys Running	0.74	0.87	0.87	0.85
Boys Villa	0.59	0.79	0.79	0.73
Eye	0.42	0.49	0.49	0.45
Fire Department	0.55	0.57	0.58	0.55
Girl	0.90	0.93	0.92	0.92
Lake	0.37	0.66	0.66	0.65
Snowy Landscape	0.64	0.69	0.68	0.67

	Block Subsampling	**Block Subsampling + Cut-offs**	**Ulterior Subsampling**	**Ulterior Subsampling + Cut-Offs**
Akademie	0.73	0.73	0.73	0.73
Balls	0.84	0.84	0.82	0.82
Boys Running	0.88	0.86	0.86	0.85
Boys Villa	0.78	0.77	0.78	0.72
Eye	0.47	0.46	0.48	0.45
Fire Department	0.56	0.55	0.56	0.54
Girl	0.92	0.92	0.92	0.92
Lake	0.66	0.65	0.66	0.64
Snowy Landscape	0.67	0.67	0.68	0.67

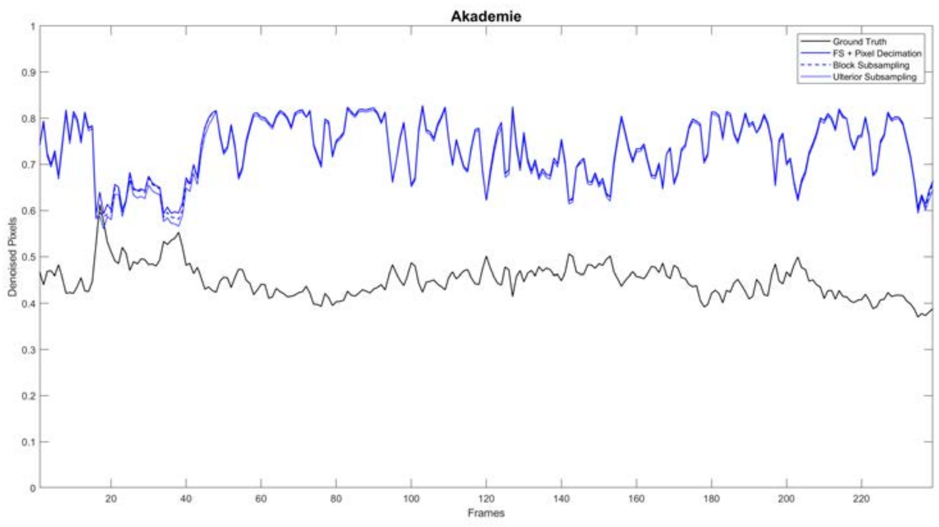

Figure 5.15: Comparison of the denoising rates of the algorithms wrt the ground truth for the Akademie sequence with a block size of 16×16 px. The improvement wrt the ground truth is very noticeable.

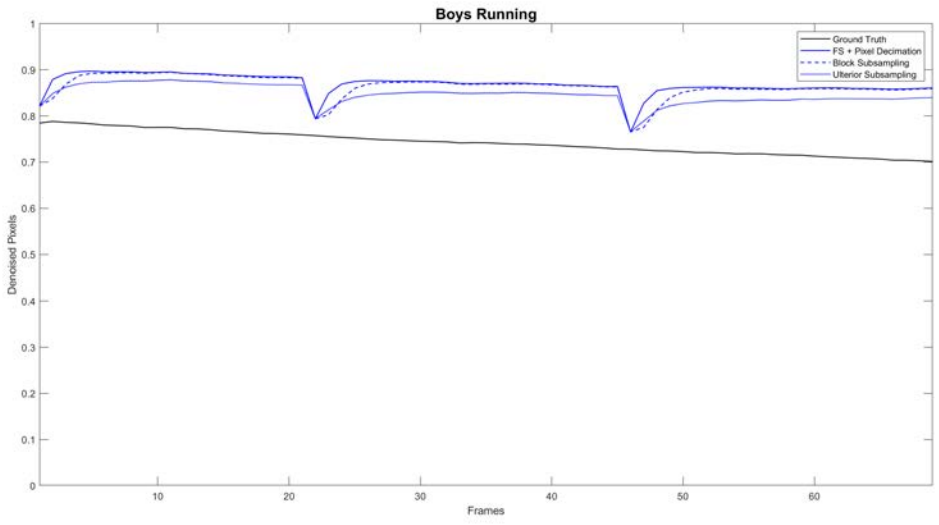

Figure 5.16: Comparison of the denoising rates of the algorithms wrt the ground truth for the Boys Running sequence with a block size of 16×16 px. There is a slight improvement in the denoisining rate wrt the ground truth.

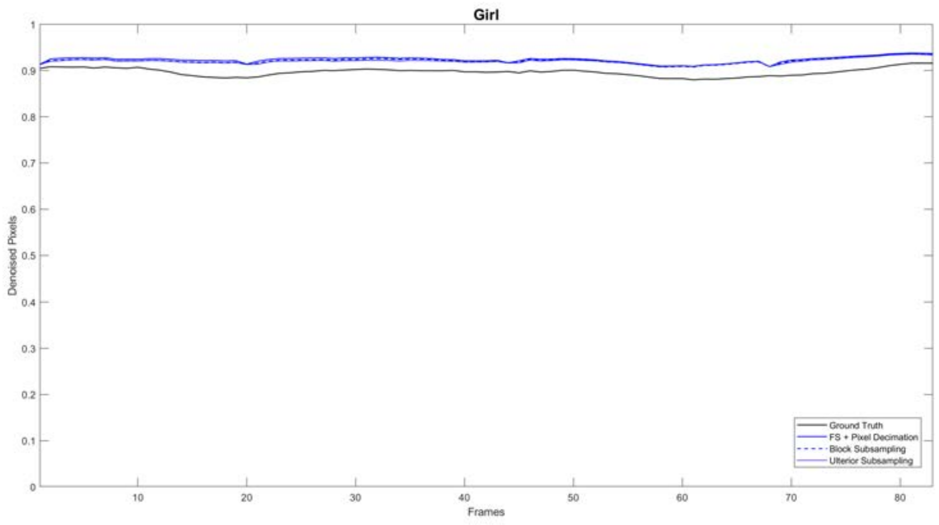

Figure 5.17: Comparison of the denoising rates of the algorithms wrt the ground truth for the Girl sequence with a block size of 16×16 px. As the sequence does not involve lots of movement, the improvement is minor, and all the algorithms achieve similar results.

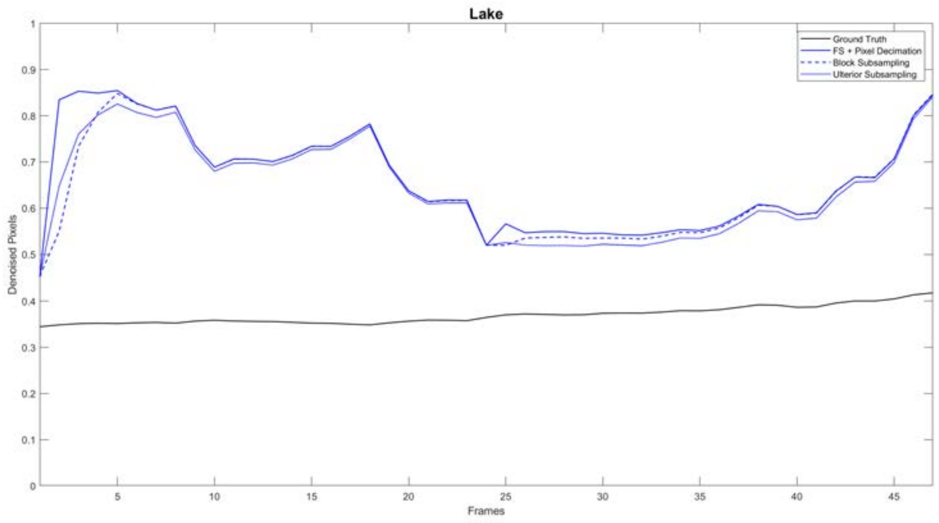

Figure 5.18: Comparison of the denoising rates of the algorithms wrt the ground truth for the Lake sequence with a block size of 16×16 px. The improvement for all the algorithms is noticeable.

Although denoising rates are similar, the performance of the algorithms was also measured through the MSE per pixel, as reported on Figures 5.19 to 5.27. In most cases, the proposed approaches have a very similar MSE per pixel to the full search algorithm with no decimation, although as the precision of the algorithm decreases, the error metric slightly increases. This implies that comparable results can be achieved with less and faster computations.

In Figure 5.19, some noticeable peaks can be observed at the beginning of the sequence. The camera vibration caused by the beginning of the movement reduced the accuracy of the prediction, until camera stabilization was reached. The *Balls* sequence presents a gradual reduction of the scene luminosity, which is reported by the growth of the error, shown in Figure 5.20. Peaks can be observed also on Figure 5.21, that resulted in the two drops on the denoising rate seen in Figure 5.16.

In general, higher error rates result in lower predictive quality, and therefore in poor denoising rates. This phenomenon is mostly visible on the Snowy Landscape sequence, where a huge increase of the error on the middle part of the sequence (shown in Figure 5.27), produced by the fast movement of the car and of the camera following it, is followed by a massive drop in the denoising rate. Indeed, motion estimation is not well suited to detect fast movements, represented in subsequent frames by large steps from the initial position of an object to the next one.

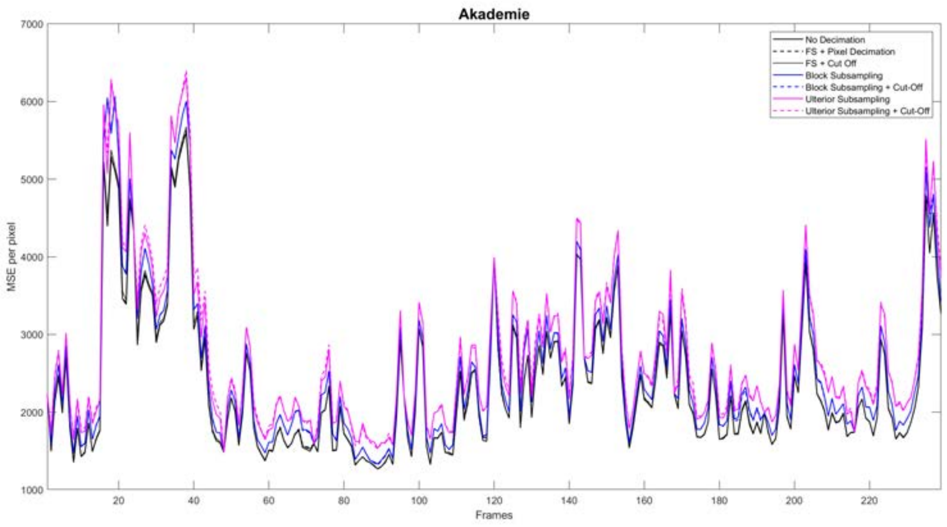

Figure 5.19: MSE per pixel for all the algorithms applied to the Akademie sequence with a block size of 16×16 px. The outcomes are very similar, with a minor increase of the MSE as the precision of the prediction is reduced.

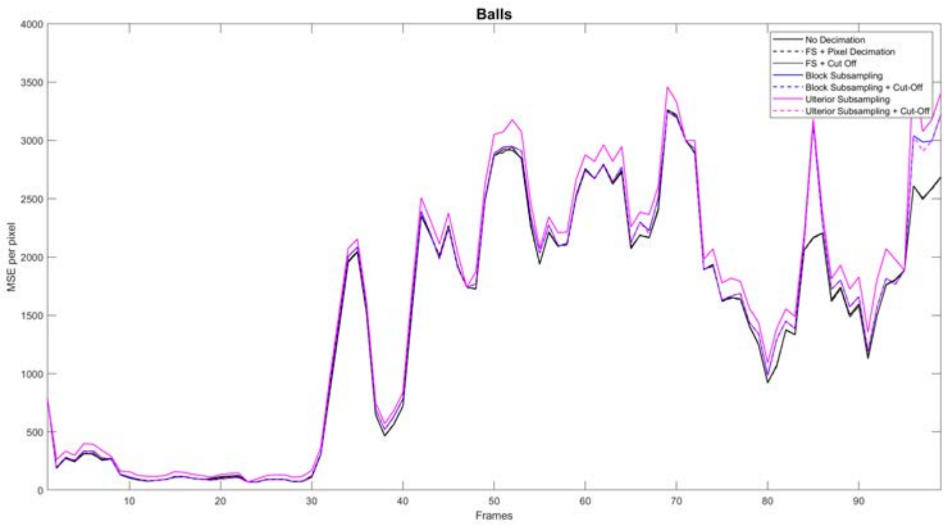

Figure 5.20: MSE per pixel for all the algorithms applied to the Balls sequence with a block size of 16×16 px. Only some minor peaks due to the ulterior subsampling are visible.

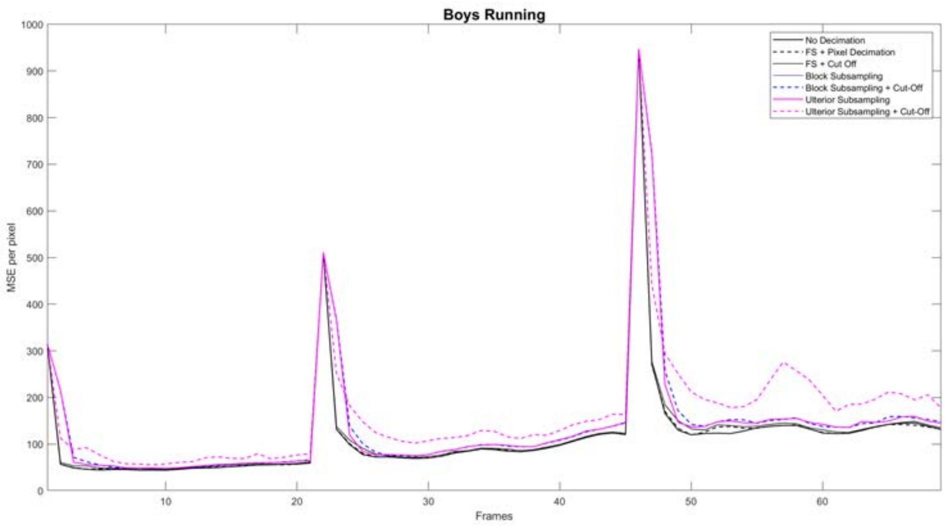

Figure 5.21: MSE per pixel for all the algorithms applied to the Boys Running sequence with a block size of 16×16 px. The results are very similar, with a slight increase of the metric for the ulterior subsampling + cut-off algorithm.

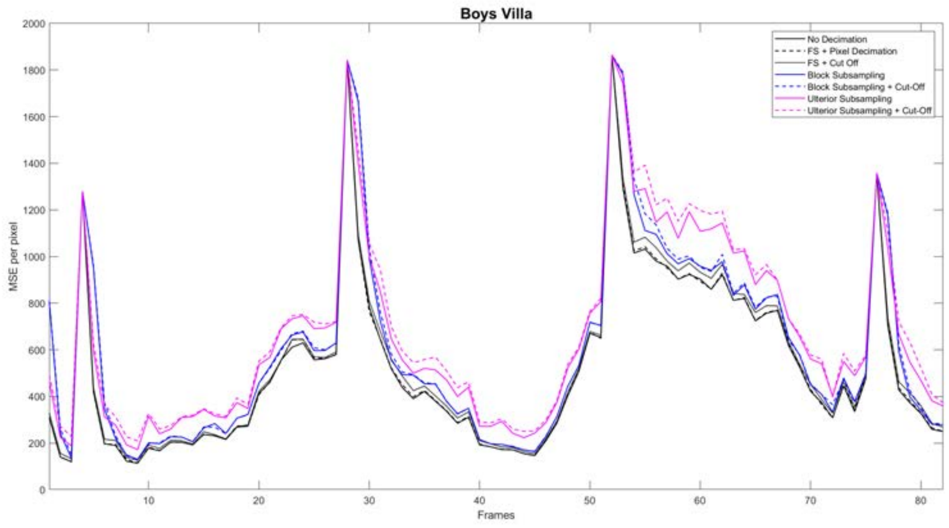

Figure 5.22: MSE per pixel for all the algorithms applied to the Boys Villa sequence with a block size of 16×16 px. There is a slight increase of the metric value as the precision of the algorithms is reduced.

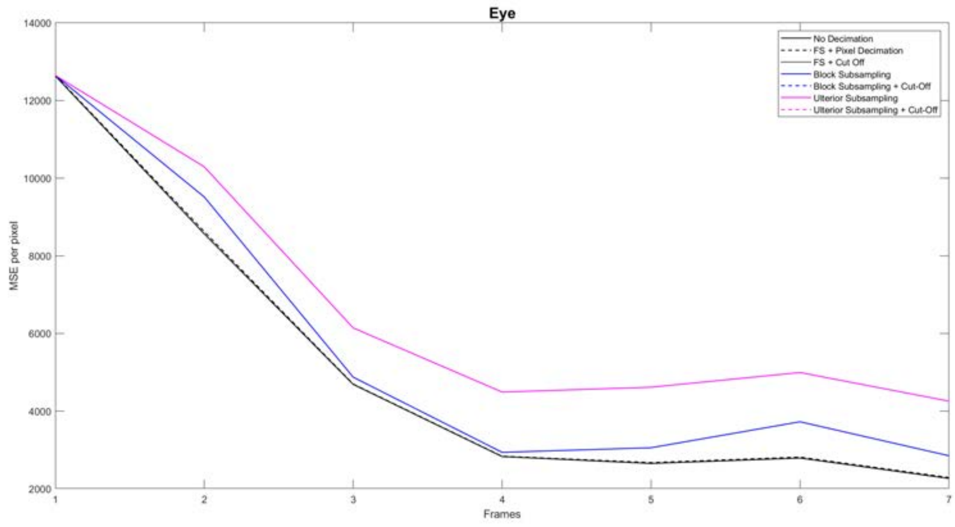

Figure 5.23: MSE per pixel for all the algorithms applied to the Eye sequence with a block size of 16×16 px. As the sequence is very fast, there is a visible reduction in the accuracy of the prediction.

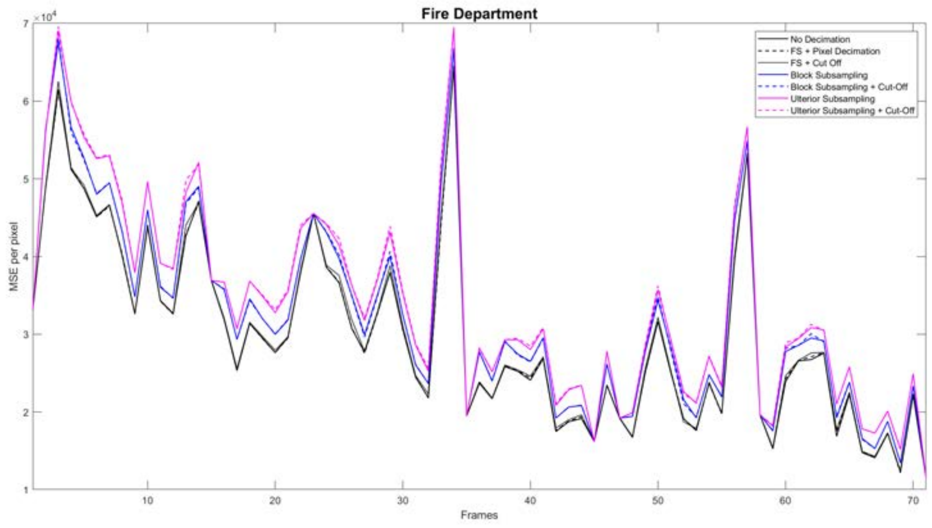

Figure 5.24: MSE per pixel for all the algorithms applied to the Fire Department sequence with a block size of 16×16 px.

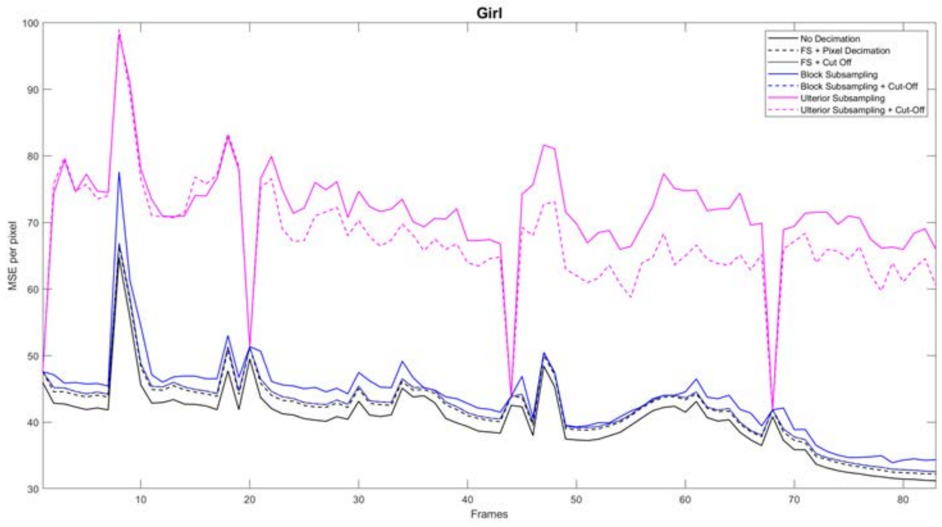

Figure 5.25: MSE per pixel for all the algorithms applied to the Girl sequence with a block size of 16×16 px. As the overall measure is not large, the reduction of the accuracy is more visible.

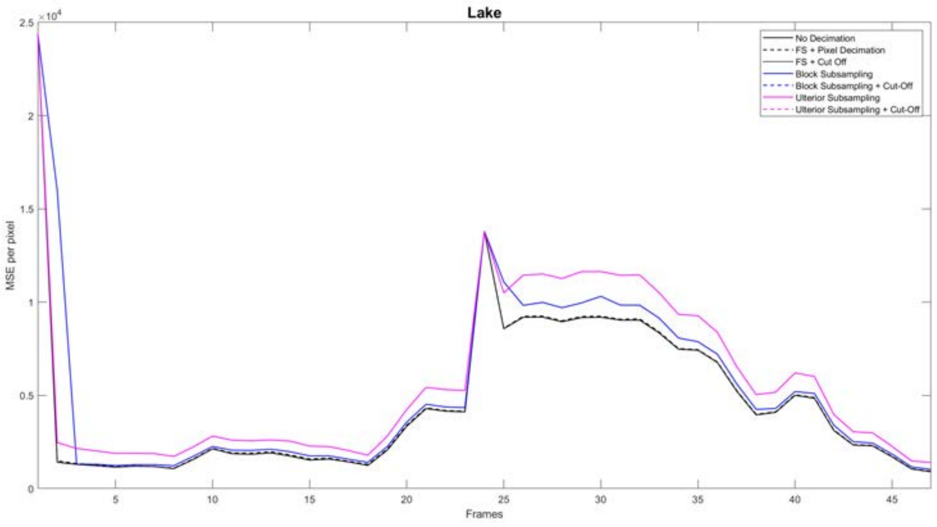

Figure 5.26: MSE per pixel for all the algorithms applied to the Lake sequence with a block size of 16×16 px. The results are akin to the previous ones.

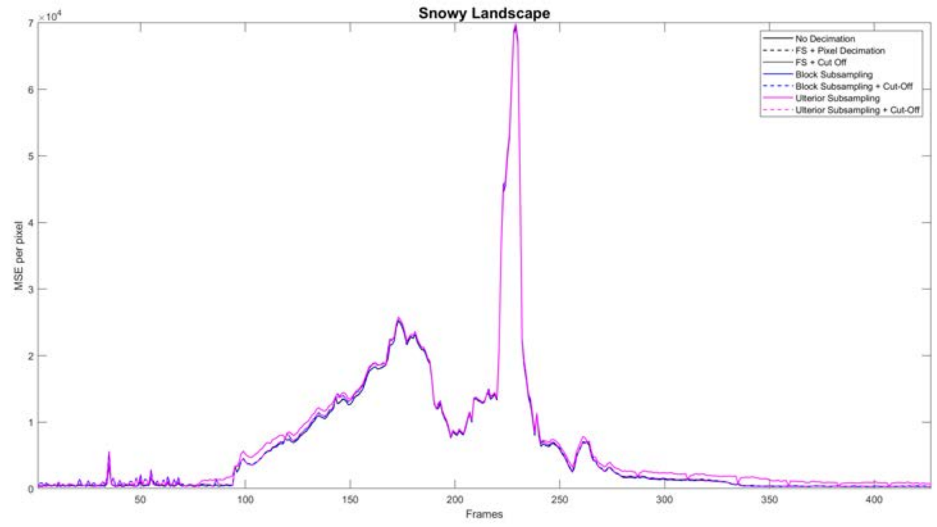

Figure 5.27: MSE per pixel for all the algorithms applied to the Snowy Landscape sequence with a block size of 16×16 px.

Early Cut-Offs The introduction of early cut-offs drastically increased the speed of the algorithms, without suffering the loss of excessive accuracy, as it was earlier argued. Cut-offs allow to reduce the number of searched location to less than 10, when the full search algorithm would require 21 iterations as it was set. Also, as anticipated, the MSE per pixel of sequences estimated with early cut-off algorithms is equal or lower to the counterpart that performed the search on the full window in most cases, or with negligible fluctuations.

Histograms displaying the normalized cut-off probabilities for some sequences are shown in Figures 5.28 to 5.36. It must be pointed out that most of the times, when the algorithm performs 21 iterations, it is due to the cyclical full search it performs to refresh the motion vectors.

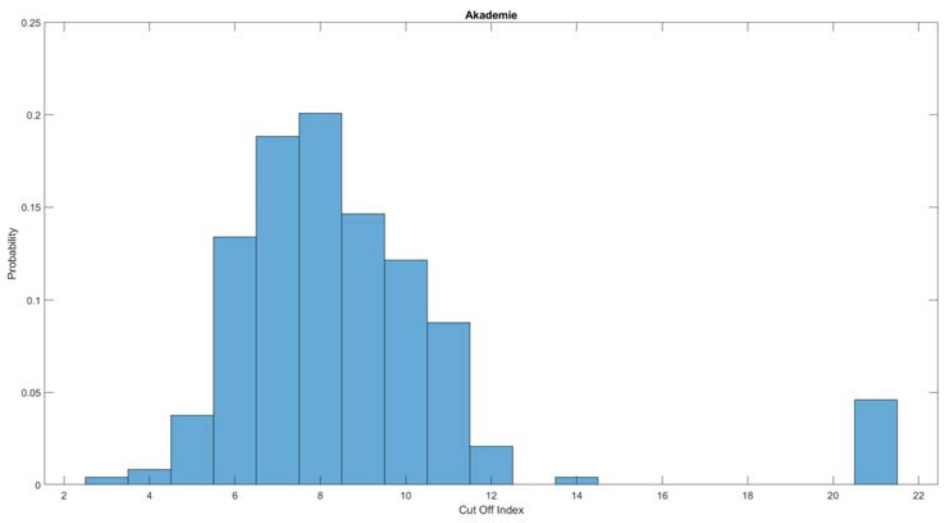

Figure 5.28: Cut-off probability of the Akademie sequence for a block size of 16×16 px.

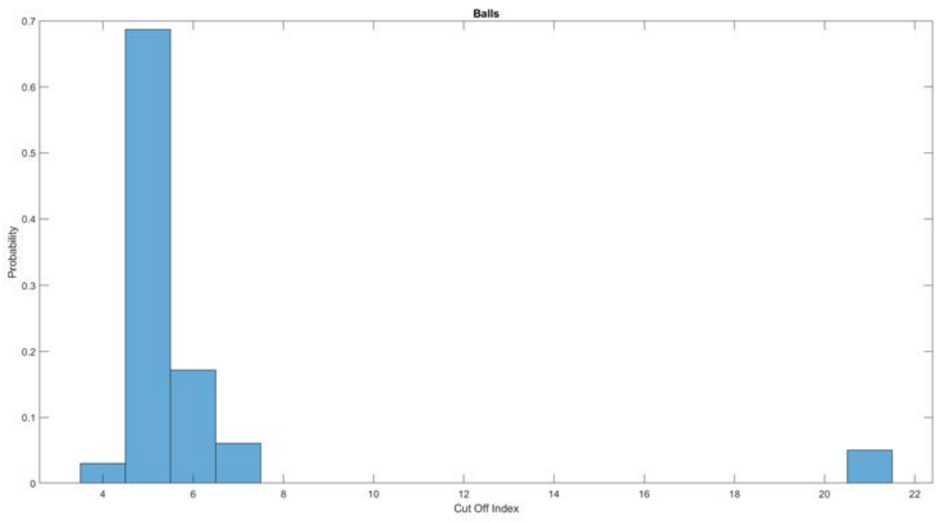

Figure 5.29: Cut-off probability of the Balls sequence for a block size of 16×16 px.

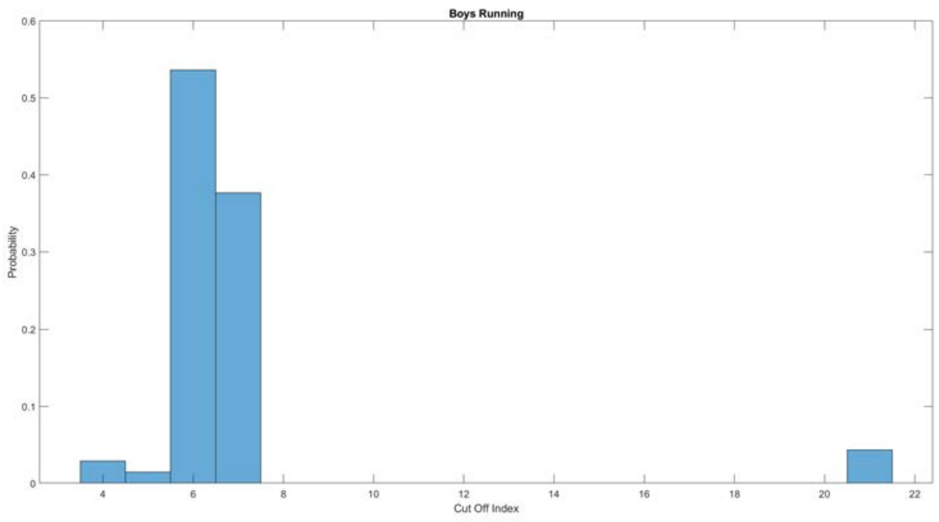

Figure 5.30: Cut-off probability of the Boys Running sequence for a block size of 16×16 px.

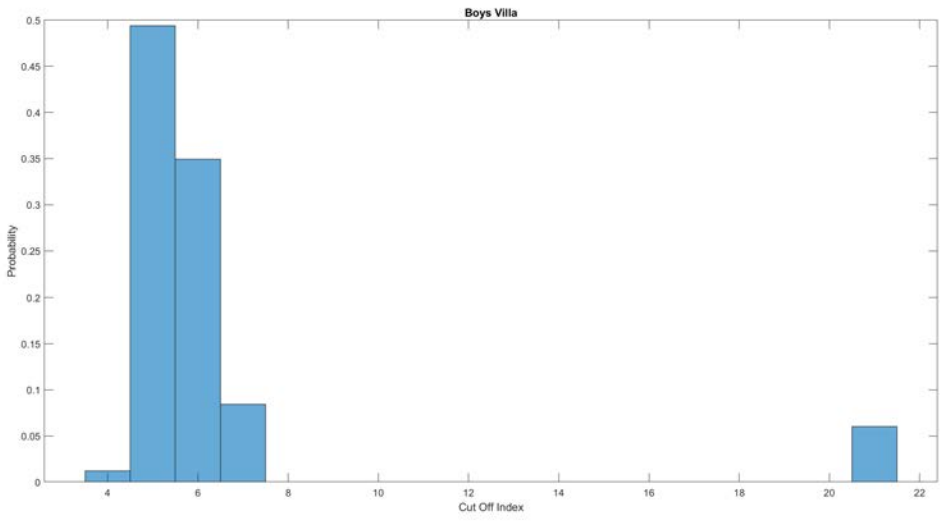

Figure 5.31: Cut-off probability of the Boys Villa sequence for a block size of 16×16 px.

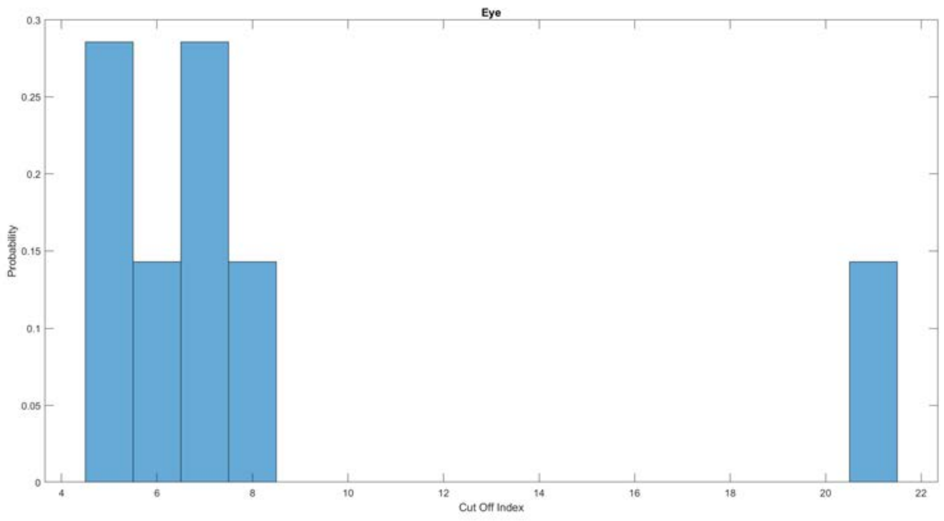

Figure 5.32: Cut-off probability of the Eye sequence for a block size of 16×16 px.

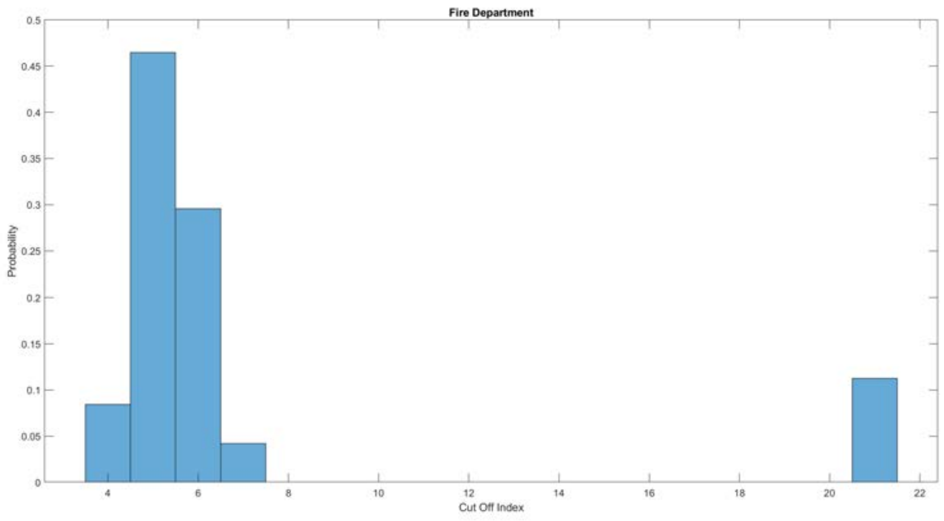

Figure 5.33: Cut-off probability of the Fire Department sequence for a block size of 16×16 px.

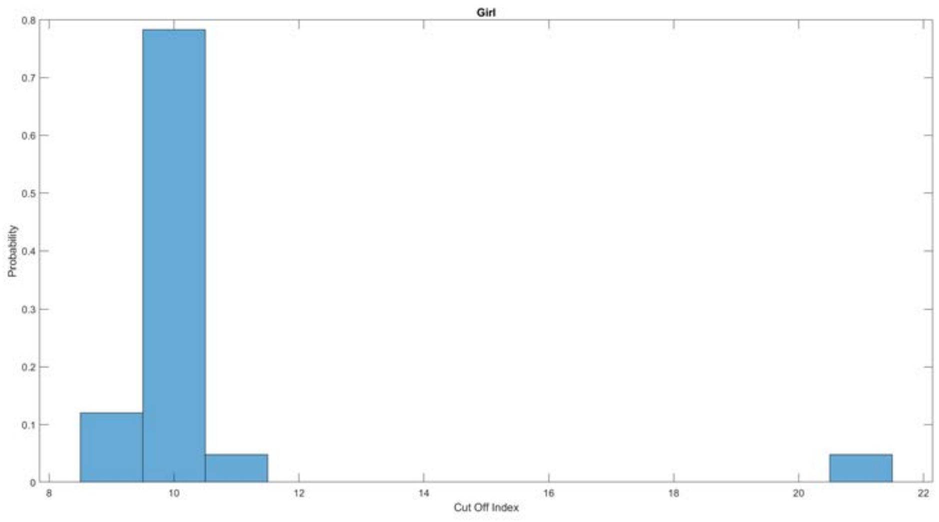

Figure 5.34: Cut-off probability of the Girl sequence for a block size of 16×16 px.

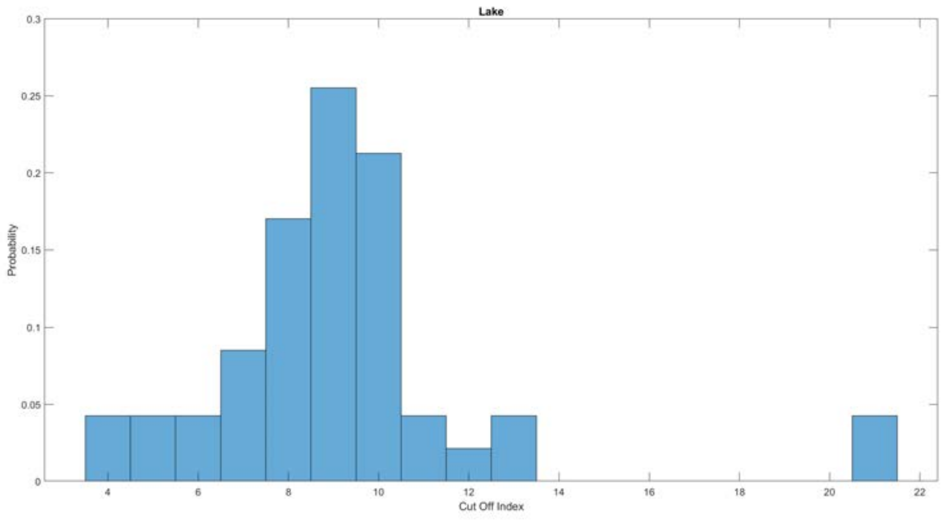

Figure 5.35: Cut-off probability of the Lake sequence for a block size of 16×16 px.

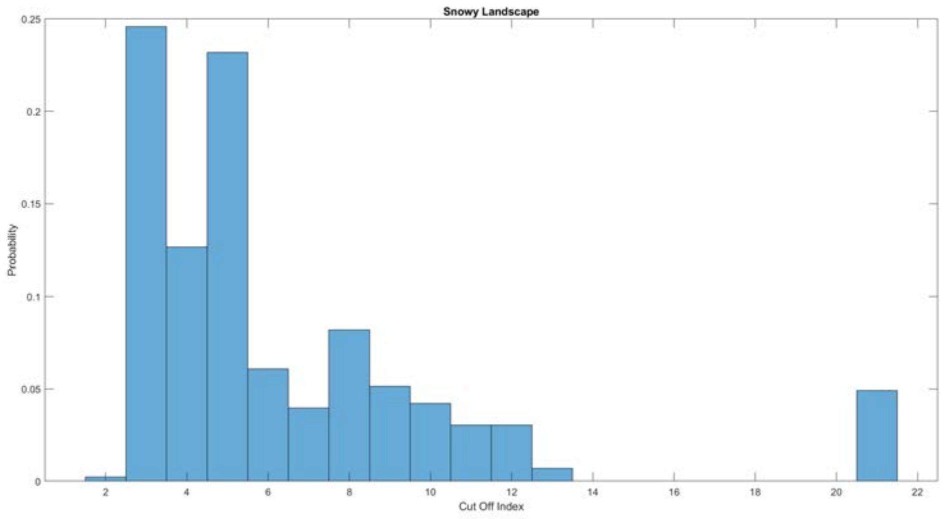

Figure 5.36: Cut-off probability of the Snowy Landscape sequence for a block size of 16×16 px.

Motion Vectors Finally, a brief discussion on the statistics of motion vectors is presented. Only the full search with no pixel decimation algorithm is considered, as the conclusions naturally extend to the other algorithms. It must be noted that motion vectors are multiples of two, because the algorithms were run on RAW sequences recorded using a Bayer pattern, which has a size of 2×2 pixels. Pixels must be compared only with others of the corresponding color.

As predicted, the majority of the motion vectors was determined to belong in a neighborhood of the coordinates $[0, 0]$ for all the sequences. Therefore, the algorithms certainly benefit from the early cut-offs, that allow to jump directly to the subsequent iteration without wasting time searching on meaningless locations. Peaks in horizontal movements can be seen on the two *Boys* sequences. Overall, all plots show a prevalence of small movements. Small horizontal movements can also be observed on the *Akademie* and *Balls* sequences, while vertical motion is predominant on in the *Lake* sequence.

The *Fire Department* sequence presents a lot of random movement, that is reported by the significant presence of non-zero motion vectors in all directions, although the majority are concentrated on the center, due to the still background of the scene.

Relevant examples are shown on Figure 5.37, 5.38, 5.39, 5.40 and 5.41.

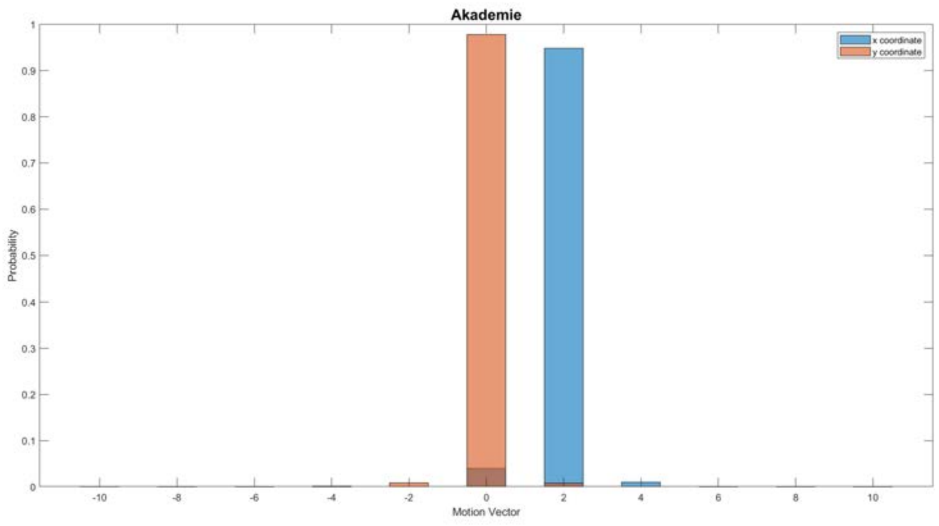

Figure 5.37: Motion vectors estimated on the Akademie sequence with the full seach algorithm with no pixel decimation and a block size of 16×16 px.

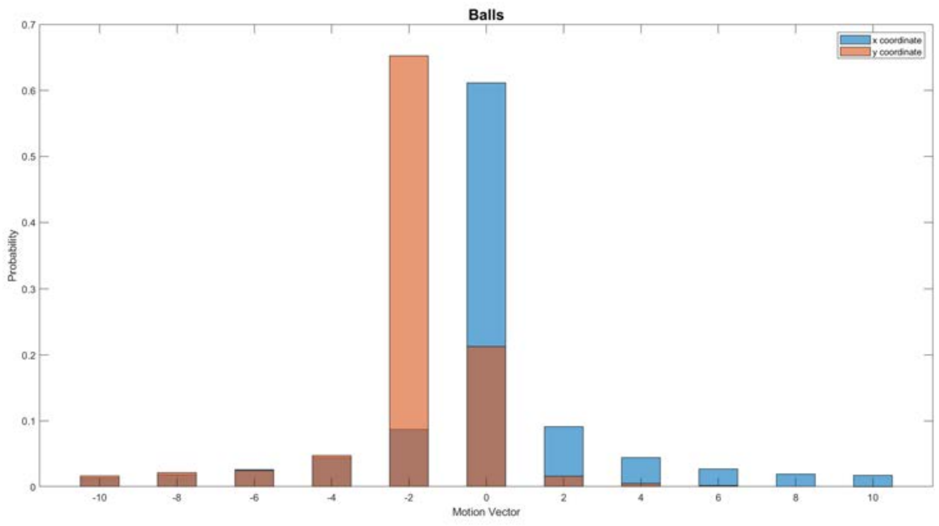

Figure 5.38: Motion vectors estimated on the Balls sequence with the full seach algorithm with no pixel decimation and a block size of 16×16 px.

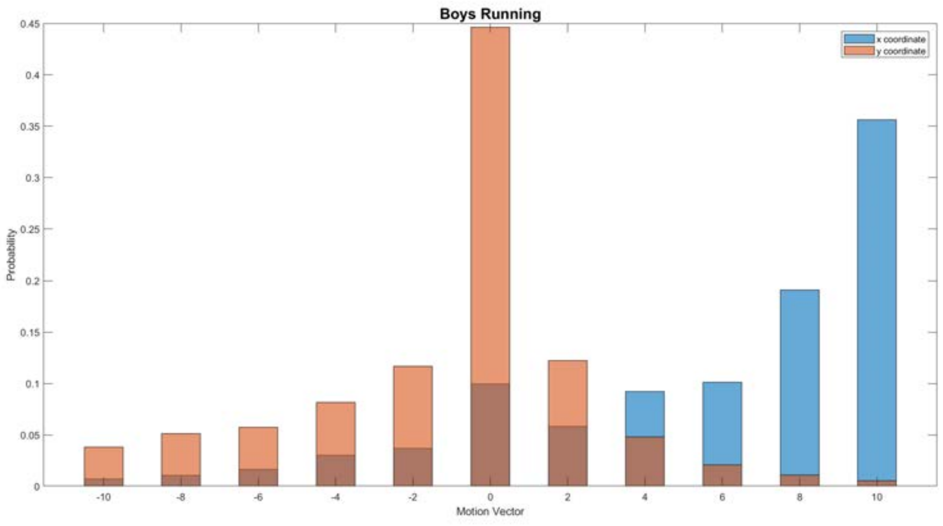

Figure 5.39: Motion vectors estimated on the Boys Running sequence with the full seach algorithm with no pixel decimation and a block size of 16×16 px.

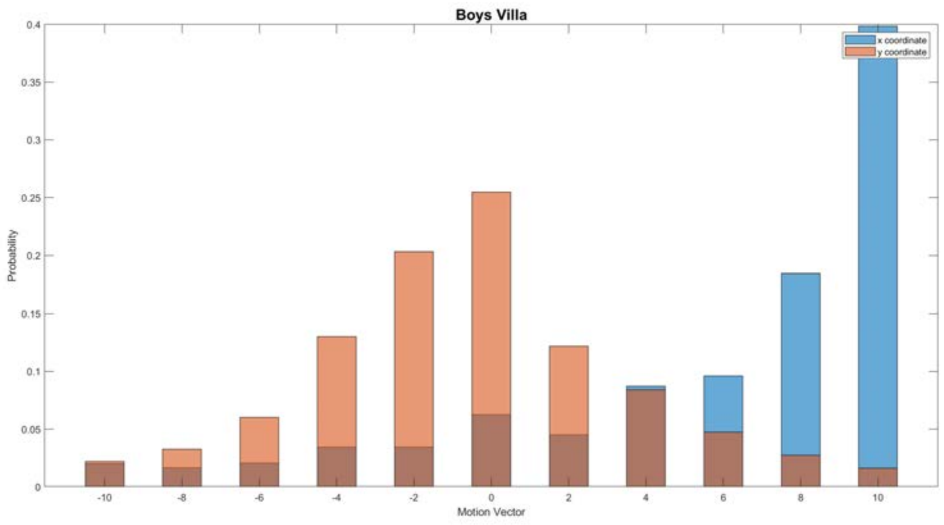

Figure 5.40: Motion vectors estimated on the Boys Villa sequence with the full seach algorithm with no pixel decimation and a block size of 16×16 px.

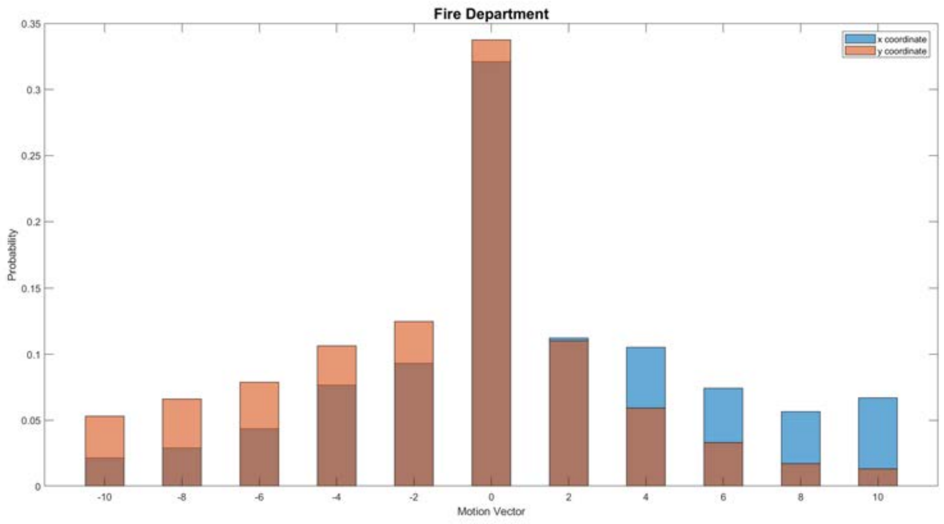

Figure 5.41: Motion vectors estimated on the Fire Department sequence with the full seach algorithm with no pixel decimation and a block size of 16×16 px.

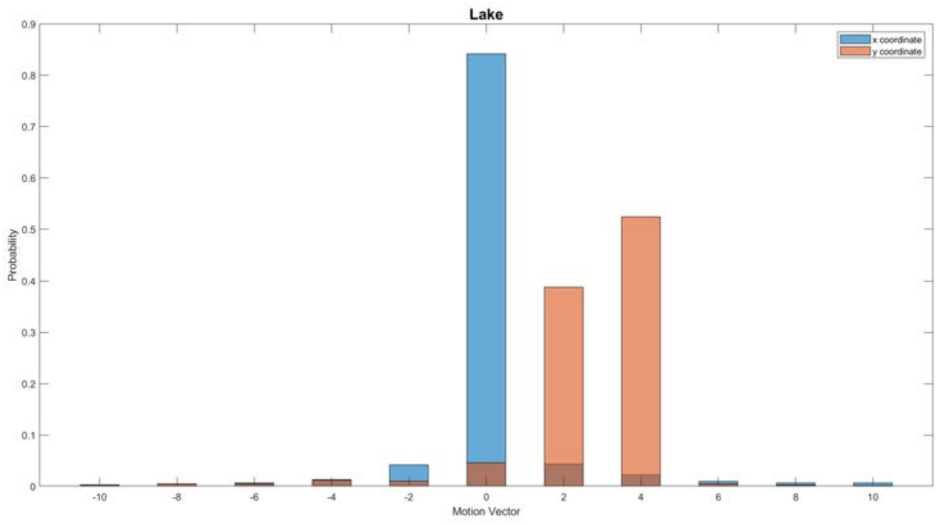

Figure 5.42: Motion vectors estimated on the Lake sequence with the full seach algorithm with no pixel decimation and a block size of 16$\times$16 px.

6
Conclusion

In this thesis, three different approaches aimed at reducing the computational complexity of the Full Search Block Matching Algorithm have been discussed. To achieve this, precision in the detection of the optical flow is traded for speed. The main goal of this research is a real-time application that could be directly applied to a camera system, to increase the performance of the already implemented temporal denoiser, during the recording.

The algorithms proposed include a rework of the Full Search approach by exploiting previous motion estimation to predict the future position of moving objects, reducing the amount of locations searched for the matching, and a gradual reduction of the number of pixels used in the motion estimation, i.e, a minimization of the operations. This is performed by introducing pixel decimation, block subsampling and early cut-offs. Other expedients adopted include the minimization of stressful hardware operations such as multiplications and divisions, for example by employing the SAD objective function, instead of the MAD, as they are conceptually equivalent, with the latter requiring a division.

The results proved that even by drastically reducing the complexity by using the above techniques, the denoising results are comparable with the Full Search Block Matching Algorithms. Early cut-off and motion vector statistics prove that the assumption on which the algorithms rely on, that most of the times the optical flow within subsequent frames varies at a slow pace, seems to be accurate and reliable. In fact, early cut-offs statistics show that on average the search was interrupted after six iterations (compared to a set maximum of twenty one), while motion vector statistics exhibit a vast predominance of small movements. Early cut-offs significantly reduced the time required by the algorithms to compute the matching on a whole sequence.

Motion compensation seemed not to perform well on fast moving sequences, where the quality of the prediction and denoising rates dropped significantly. However, this limit is compensated by the fact that the human eye is not able to capture details when looking at fast moving objects, therefore visual impurities go by unnoticed.

It is important to apply the block matching algorithm using as a reference the already denoised frame from the previous iteration. Applying denoising is necessary as it will aid the prediction by providing a reference image affected by less artifacts, granting more precise results. However, the denoising algorithm is very computationally demanding too, and it was the most time consuming operation during the testing phase.

Future work might involve exploiting the results obtained to further enhance the algorithm, such us optimizing the parameters used to match hardware constraints, in view of applying it to FPGA technology. It is important to note that early cut-offs are unlikely to be implemented in FPGA technology, as it is hard to manage variable complexity on hardware due to synchronization issues. Usually implementations are executed by taking into account the worst case scenario. It is also suggested to introduce a certain level of parallelization in both the block matching and denoising procedures, to speed up the process.

Listing of figures

Listing of tables

References

[1] E. Ballan, "Evaluation of block-matching parameters for motion compensated temporal denoising."

[2] K. Sayood, *Introduction to Data Compression.* Morgan Kaufmann, 2012.

[3] B. E. Bayer, "Color imaging array," Patent.

[4] B. Horn and B. Schunk, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[5] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proc. of the DARPA Image Understanding Workshop*, pp. 121–130, 1984.

[6] A. N. Netravali and J. D. Robbins, "Motion-compensated television coding: Part i," *The Bell System Technical Journal*, vol. 58, no. 3, pp. 631–670, March 1979.

[7] R. Srinivasan and K. Rao, "Predictive coding based on efficient motion estimation," *IEEE Transactions on Communications*, vol. 33, no. 8, pp. 888–896, August 1985.

[8] H. G. Musmann, P. Pirsch, and H. . Grallert, "Advances in picture coding," *Proceedings of the IEEE*, vol. 73, no. 4, pp. 523–548, April 1985.

[9] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Dec 2012.

[10] Wenjie Yin, Haiwu Zhao, Guoping Li, Guozhong Wang, and Guowei Teng, "A block based temporal spatial nonlocal mean algorithm for video denoising with multiple resolution," in *2012 6th International Conference on Signal Processing and Communication Systems*, Dec 2012, pp. 1–4.

[11] K. J. Boo and N. K. Bose, "A motion-compensated spatio-temporal filter for image sequences with signal-dependent noise," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 3, pp. 287–298, June 1998.

[12] A. A. Yahya, J. Tan, B. Su, and K. Liu, "Video denoising based on spatial-temporal filtering," in *2016 6th International Conference on Digital Home (ICDH)*, Dec 2016, pp. 34–37.

[13] A. H. Y. I. T. I. T. Koga, K. Iinuma, "Motioncompensated interframe coding for video conferencing," *Proc. National Telecommunication Conference*, 1981.

[14] Reoxiang Li, Bing Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438–442, Aug 1994.

[15] Shan Zhu and Kai-Kuang Ma, "A new diamond search algorithm for fast block matching motion estimation," in *Proceedings of ICICS, 1997 International Conference on Information, Communications and Signal Processing. Theme: Trends in Information Systems Engineering and Wireless Multimedia Communications (Cat.*, vol. 1, Sep. 1997, pp. 292–296 vol.1.

[16] Yao Nie and Kai-Kuang Ma, "Adaptive rood pattern search for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 11, no. 12, pp. 1442–1449, Dec 2002.

[17] N. Purnachand, L. N. Alves, and A. Navarro, "Fast motion estimation algorithm for hevc," in *2012 IEEE Second International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, Sep. 2012, pp. 34–37.

[18] S. Acharjee, D. Biswas, N. Dey, P. Maji, and S. S. Chaudhuri, "An efficient motion estimation algorithm using division mechanism of low and high motion zone," in *2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*, March 2013, pp. 169–172.

[19] B. N. J. A. Z. Wei, D. Lee and B. Edwards, "Fpga-based embedded motion estimation sensor," *International Journal of Reconfigurable Computing*, 2008.

[20] Y.-g. Xue, H.-y. Su, J. Ren, M. Wen, C.-y. Zhang, and L.-q. Xiao, “A highly parallel and scalable motion estimation algorithm with gpu for hevc,” *Scientific Programming*, vol. 2017, pp. 1–15, 10 2017.

[21] J. Olivares, I. Benavides, J. Hormigo, J. Villalba, and E. Zapata, “Fast full-search block matching algorithm motion estimation alternatives in fpga,” in *2006 International Conference on Field Programmable Logic and Applications*, Aug 2006, pp. 1–4.

[22] J. Watkinson, *The Engineer’s Guide to Motion Compensation.* Snell e Wilcox, 1994.

[23] Yui-Lam Chan and Wan-Chi Siu, “A new block motion vector estimation using adaptive pixel decimation,” in *1995 International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, May 1995, pp. 2257–2260 vol.4.

[24] Lei Hua, Lei Xie, and Huifang Chen, “A color interpolation algorithm for bayer pattern digital cameras based on green components and color difference space,” in *2010 IEEE International Conference on Progress in Informatics and Computing*, vol. 2, Dec 2010, pp. 791–795.

[25] B. Liu and A. Zaccarin, “New fast algorithms for the estimation of block motion vectors,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, no. 2, pp. 148–157, April 1993.

Acknowledgments

FIRST OF ALL, I would like to thank my supervisor Professor Giancarlo Calvagno and my co-supervisor Stefano Andriani, for the great opportunity that they offered me, along with all the advice and help they gave me during the course of these months.

I would also like to thank my family and my closest friends, for all the support and encouragement they provided me, besides the *patience* they had, during all these years of university.

Thank you