UNIVERSIDADE FEDERAL DE JUIZ DE FORA

INSTITUTO DE CIÊNCIAS EXATAS

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Variable Block Size Motion Estimation Algorithms

Fábio Luiz Marinho de Oliveira

JUIZ DE FORA

FEVEREIRO, 2014

Variable Block Size Motion Estimation Algorithms

Fábio Luiz Marinho de Oliveira

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Marcelo Bernardes Vieira

JUIZ DE FORA

FEVEREIRO, 2014

Variable Block Size Motion Estimation Algorithms

Fábio Luiz Marinho de Oliveira

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Marcelo Bernardes Vieira
Doutor em Ciência da Computação

Rodrigo Luis de Souza da Silva
Doutor em Engenharia Civil

Marcelo Caniato Renhe
Mestre em Engenharia de Sistemas e Computação

JUIZ DE FORA
07 DE FEVEREIRO, 2014

À família pelo apoio e sustento.

Aos amigos, pela presença e motivação.

Resumo

Este trabalho pretende realizar uma comparação de métodos de estimação de movimento entre imagens, mais precisamente entre os algoritmos *Full Search*, *Four Step Search* e suas respectivas versões com tamanho de bloco variável. Para essa análise, além de critérios de eficiência, é usada a taxa de reconhecimento da base de dados KTH. O trabalho apresenta os conceitos e algoritmos de casamento de blocos que fundamentam essa comparação.

Palavras-chave: Estimação de movimento, casamento de blocos, tamanho de bloco variável

Abstract

This work intends to make a comparison between motion estimation methods, more precisely between Full Search, Four Step Search and their respective variable block size versions. For such analysis, along with efficiency criteria, the recognition rate from KTH dataset is used. The work presents block matching concepts and algorithms that underlie the comparison.

Keywords: Motion Estimation, block matching, variable block size

Agradecimentos

A todos a minha família, pelo encorajamento e apoio.

Ao professor Marcelo Bernardes pela orientação, motivação e paciência, sem a qual este trabalho não se realizaria.

Aos companheiros do Grupo de Computação Gráfica, pela ajuda e inspiração.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

"This is how you do it: you sit down at the keyboard and you put one word after another until its done. It's that easy, and that hard".

Neil Gaiman

Contents

List of Figures

List of Tables

Lista de Abreviações

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
BMA	Block Matching Algorithm
VSBMA	Variable-size Block Matching Algorithm
FS	Full Search
4SS	Four Step Search
MAD	Mean Absolute Difference
SAD	Sum of Absolute Differences

1 Introduction

Detecting movement in a sequence of images is an important research field of computer vision. Several applications rely on the quality and efficiency of methods carrying out this task. Motion Estimation is one of these methods and consists in tracking the movement of regions between frames of a video.

The assumption is that if there were continuous motion in the image sequence, several blocks from one image could be found on the next one, but in different positions. This was described as a “piecewise translation” by Jain and Jain [6], who first introduced the technique. So, the goal would be to find where these blocks, possibly representing objects, are in the following frame.

The algorithms for motion estimation, called Block Matching Algorithms (BMA), vary mainly on search strategies and error functions. Search strategies are more efficient ways of analysing the image in order to find the best match for a region. Since it is highly unlikely that two regions will have exactly the same pixel intensities, an error function is used as criterion for which block provides the best match, measuring how similar these regions are.

A variant of the BMA is the Variable-size Block Matching Algorithm (VSBM), which does not maintain a fixed size for the blocks analysed throughout the computation. In the VSBM, the sizes of the blocks often change via split or merge, depending on results yielded by the block matching routine.

This work presents two search strategies coupled with two error functions and also presents a VSBM approach applied with both strategies. The search strategies are: Full Search and Four Step Search. The error functions are: Sum of Absolute Differences and Mean Absolute Difference.

Chapter 2 provides the fundamentals of motion estimation necessary for the understanding of this work. Chapter 3 presents the different methods implemented. Chapter 4 contains the experimental results and comparison of the algorithms. Chapter 5 brings the conclusion and future works.

1.1 Problem Definition

Let J_n and J_{n+1} be two consecutive frames in a video. The displacement vector field D_n is defined as $D_n(b) = [d_x(b_x, b_y), d_y(b_x, b_y)]$ for each block b on J_n with coordinates (b_x, b_y). Each vector (d_x, d_y) points to the position of the block b^* in J_{n+1} which minimizes some error function ϵ, so

$$(d_x, d_y) = (b_x^* - b_x, b_y^* - b_y), min\{\epsilon(b, b^*)\}.$$

Ideally, b and b^* have very similar pixel intensity configurations, $\epsilon(b, b^*)$ is close to zero, and the field D_n is a good representation of object motion between the two frames.

1.2 Objectives

The main objective of this work is the study and implementation of motion estimation methods, focusing on the variable block sizes approach.

As secondary objectives, there is efficiency and quality comparison between the methods and the investigation on how motion estimation can contribute to generate descriptors on motion detection and classification problems.

2 Fundamentals

2.1 Basic Concepts and Definitions

2.1.1 Images

An image can be defined as a function $f : \mathbb{R}^2 \to \mathbb{R}^n, (x, y) \mapsto \vec{f}(x, y)$, where x and y are called space coordinates, $n \in \mathbb{N}$ is the number of channels, and the amplitude f for any coordinates (x, y) is called intensity or brightness level. For $n = 1$, we have $f : \mathbb{R}^2 \to \mathbb{R}, (x, y) \mapsto f(x, y)$, a scalar function representing the grey scale level on each point [3].

Digital Images

In order to represent an image on a computer, it has to be captured by sensors. However, the data obtained via these sensors do not reflect the continuity of the image with respect to the space coordinates and the amplitude. These sensors are usually restricted to a finite number of positions and finite range of intensities captured. Even when these restrictions do not apply to the sensor, they do for the image representation on the computer, which has a finite number of bits. So, at some point between the acquisition and storage of the image, the discretization of both space coordinates and amplitude has to occur. The discretization of the domain into a finite number of regular intervals is called sampling. The discretization of the co-domain is called quantization.

A digital image is an image which can be represented in a finite number of bits. In other words, it is an image with both domain and co-domain discrete. For purposes of simplicity, the discretization process maps the coordinates into integers so that $x = 0, 1, 2, ..., M - 1$ and $y = 0, 1, 2, ..., N - 1$ and the intensities into $f(x, y) = 0, 1, 2, ..., L - 1$. This way, each channel of the image could be represented by a matrix $A_{M \times N}$ such that $a_{ij} = f(x = i, y = j) = f(i, j), i = 0, 1, 2, ..., M - 1, j = 0, 1, 2, ..., N - 1$. Each element of this matrix corresponds to a pair of coordinates and is called picture element, pel, or pixel [3].

Image Operations

Despite its matrix representations, operations involving one or more images are carried out pixel by pixel.

The sum, difference, product and division between two images f and g are written as

$$s(x, y) = f(x, y) + g(x, y)$$

$$d(x, y) = f(x, y) - g(x, y)$$

$$p(x, y) = f(x, y) \times g(x, y)$$

$$v(x, y) = f(x, y) \div g(x, y)$$

respectively, for each pair of coordinates (x,y) [3].

The result of these operations may, sometimes, be out of the intensity range $[0, L-1]$. In this case, the result may not be an image but still have some relevance for the image processing. In this work, this will be the case when calculating the difference between two images, which may yield results below zero. This is an important operation in the block matching process, since the error metrics used rely on the difference operation.

2.1.2 Digital Videos

In a similar way to how images are defined, videos can be defined as a function $g : \mathbb{R}^3 \to \mathbb{R}^n, (x, y, t) \mapsto \vec{g}(x, y, t)$, where x and y are called space coordinates, t is called time coordinate, $n \in \mathbb{N}$ is the number of channels, and the amplitude g for any coordinates (x, y, t) is called intensity or brightness level. Digital videos are videos where x, y, t and g are discrete. We can look at a digital video as a finite sequence of images, with one $f(x, y)$ for each t. Each image in this sequence is called frame [3].

2.2 Motion Estimation Concepts and Definitions

This section presents the concepts and terminology concerning motion estimation that will frequently be used in this work.

2.2.1 Block

A rectangular $s_x \times s_y$ block of an image with domain D is defined as a subimage, or a set of pixels $b = \{(x, y) \in D | b_x \leq x \leq b_x + s_x, b_y \leq y \leq b_y + s_y\}$, where b_x and b_y are the coordinates for the top-left pixel of the block. These coordinates are used to identify the whole block.

For each block in a frame, a corresponding block is searched for in the next frame, hence the name Block Matching Algorithm.

In the Variable-size Block Matching Algorithm, the sizes s_x and s_y can vary between blocks, so they are used along with the coordinates to identify the block.

2.2.2 Search Window

During the block matching process, one frame is called “reference frame” and the following is called “target frame”.

A search window is the set of blocks on the target frame which are examined. The search window is usually centred on the block from the reference frame which is being matched. This restriction is imposed in order to reduce computational cost of analysing a whole frame.

An important observation that can be made about the search window is that it limits the size of the motion vectors calculated through the algorithm, since all the candidates for matching are assumed to be somewhat close to reference frame block. Due to this limitation, the algorithm is incapable of capturing abrupt motion in the small frame time interval.

2.2.3 Search Strategy

The search strategy is the main part of the block matching algorithm. It is the series of steps, or an algorithm *per se*, through which the search window is explored in order to find the best match for a block.

The main goal of a search strategy is to reduce the computational effort of the BMA retaining the quality of the results compared to examining each block in the search window.

Many strategies have been proposed over the years, such as New Three Step Search [9], Four Step Search [14], Simple Efficient Search [10], Diamond Search [19], and Adaptive Rood Pattern Search [12]. To illustrate the gain in performance proportioned by a search strategy, this work presents the results yielded by the Full Search, which is the brute force approach, and the Four Step Search.

2.2.4 Error Function

The error function is the measure of similarity between two blocks. This function is defined in terms of the pixel intensities in the pair of blocks being analysed.

A variety of error functions can be found in the literature being applied as error criteria for block matching algorithms [2, 6, 7], differing in complexity, outlier handling, topology, among other features.

In the BMA, the error function ϵ is the sole criterion used to find the best match for a block. The best match is the one that minimizes ϵ.

3 Block Matching Algorithms

3.1 Overview

This section presents the Block Matching Algorithm and a brief explanation of each of its steps.

Algorithm 1: Block Matching Algorithm

```
Data: Images f,g
Result: Vector set D
Divide image $f$ into a set of blocks $B$;
forall the $b^f \in B$ do
    $b^* \leftarrow b^f$;
    $cost_{b^*} \leftarrow \infty$;
    $C \leftarrow$ SearchStrategy$(g, W, C, b^*)$;
    while $C \neq \varnothing$ do
        forall the $b \in C$ do
            $cost_b \leftarrow \epsilon(b^f, b)$;
            if $cost_b < cost_{b^*}$ then
                $b^* \leftarrow b$;
                $cost_{b^*} \leftarrow cost_b$;
                $(d_x, d_y) \leftarrow (b_x - b^f_x, b_y - b^f_y)$;
            end if
        end forall
        $C \leftarrow$ SearchStrategy$(g, W, C, b^*)$;
    end while
    $D \leftarrow D \cup \{(d_x, d_y)\}$;
end forall
```

The Block Matching Algorithm consists of a series of minimization processes of a function ϵ, one for each block b^f in the reference frame f. The search space for each one of these minimization problems is bounded by the search window W and is explored through a search strategy.

In the algorithm presented above, *SearchStrategy* is assumed to be able to create and/or update a set of candidate blocks C from the image g, for each iteration of the search. This update is based on the search window, on the current candidate set, and on the current best matching block. The cost function ϵ is evaluated for every candidate b in C, and the displacement vector $d(d_x, d_y)$ is the difference between the reference frame block $b^f(b^f_x, b^f_y)$ and the block $b^*(b^*_x, b^*_y)$ which minimizes ϵ.

When the candidates set C is empty, meaning that the search has finished, the algorithm proceeds to the next block in B, until all blocks from the reference frame f have been matched with a block from the target frame g. BMA's output is the displacement vector set D, which contains a translation vector d for each block $b^f \in B$.

Figure 3.1 shows a frame with displacement vectors computed by BMA drawn over it. All the vectors are evenly spaced, since all blocks have the same size.

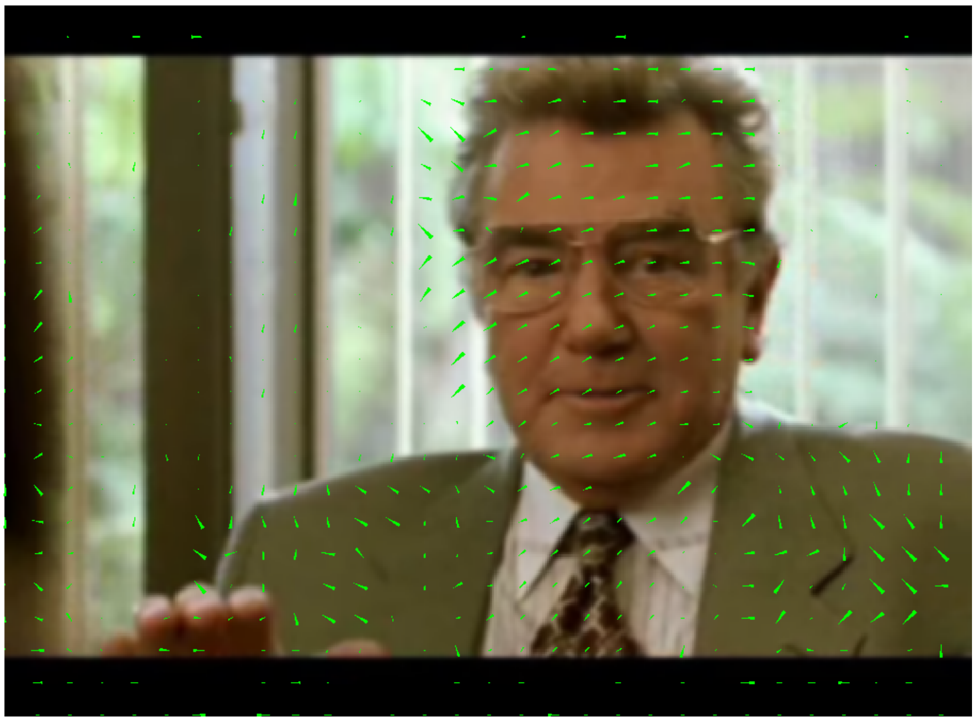


Figure 3.1: BMA displacement vectors.

3.2 Search Strategies

This section describes the search strategies implemented. The Full Search was chosen as a reference for how much computational effort the block matching task may require if a proper strategy is not employed. The Four Step Search was chosen because it is a simple strategy and yet it shows the performance gain that can be obtained, without compromising the quality of the results.

3.2.1 Full Search

The Full Search (FS) is the simplest and most thorough strategy. Once a search window is established, the full search consists of trying to match every possible block inside this window. For example, in a 15×15 window, 225 comparisons are necessary in order to find the best match.

This is a very expensive strategy, but it guarantees that the best match found is indeed the one which minimizes the error function in the search window.

3.2.2 Four Step Search

The Four Step Search (4SS) [14] is a steepest descent based strategy. It consists of four different search patterns used during its four steps.

Starting from the center of a 15×15 window, the first step looks at 9 locations in a 5×5 window. At any step, if the least error is found at the center of search pattern the search jumps to fourth step. If the least error is at one of the eight locations except the center, then this location becomes the search origin and the search moves to the second step. The search pattern is still maintained as 5×5 pixels wide. Depending the least error location, 4SS might end up checking errors at 3 or 5 additional locations. If the least error is found at a corner, the second step checks its 5 neighbours that have not been checked on the first step. If the least error is at the side, the second step checks its 3 neighbours. The third step is exactly the same as the second step, except that it always leads to the fourth step. In the fourth step the pattern size is shrunk to 3×3. The location with the least error is the best matching block and the motion vector is set to point to that position.

The patterns are shown in Figure 3.2. Figure 3.2a contains the first step pattern. Figures 3.2b and 3.2c show both cases for second and third step patterns. Figure 3.2d shows the fourth step pattern. Figure 3.3 shows an example procedure.

Since 4SS is a steepest descent based method, it is subject to being trapped in local minima, but the quality difference is small and well worth the gain in efficiency, compared to the FS. In the best case scenario, 17 points are checked and in the worst case scenario, 27 points are checked out of the 225 in the 15×15 window.

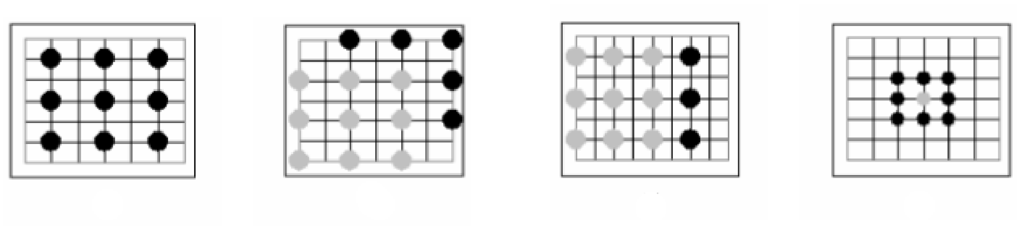

(a) (b) (c) (d)

Figure 3.2: 4SS patterns [14].

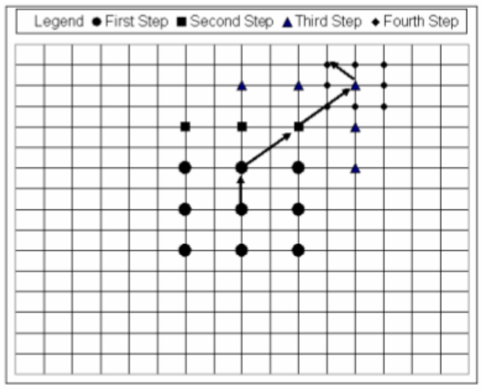

Legend ● First Step ■ Second Step ▲ Third Step ◆ Fourth Step

Figure 3.3: 4SS steps [14].

3.3 Error Functions

3.3.1 Sum of Absolute Differences

The Sum of Absolute Differences (SAD) [4] between two images f and g in a measurement window W is defined as follows

$$\epsilon_{SAD}(d_x, d_y) = \sum_{x,y \in W} |f(x, y) - g(x + d_x, y + d_y)|$$

where (x, y) are the pixel coordinates and (d_x, d_y) is the motion vector.

Notice that SAD just accumulates the absolute differences in the measurement window W. This means that bigger windows tend to produce higher ϵ_{SAD} values than smaller windows.

3.3.2 Mean Absolute Difference

The Mean Absolute Difference (MAD) [18] is calculated as follows

$$\epsilon_{MAD}(d_x, d_y) = \frac{1}{N} \sum_{x,y \in W} |f(x,y) - g(x+d_x, y+d_y)|$$

subject to the same definitions as before and N being the number of pixels in W.

Contrary to SAD, MAD takes into account the size of the measurement window. By dividing the accumulated differences by the number of pixels, different sizes of windows have little to no effect on ϵ_{MAD} values. This is an important regard since the VSBMA relies on an fixed error threshold to decide whether to split a block or not.

3.4 Variable Size Block Matching Algorithm

This section presents the Variable Block Matching Algorithm [2, 15], and a brief explanation of each of its steps.

Algorithm 2: Variable Size Block Matching Algorithm

```
Data: Images f,g
Result: Vector set D
Divide image $f$ into a set of blocks $B$;
forall the $b^f \in B$ do
    $b^* \leftarrow b^f$;
    $cost_{b^*} \leftarrow \infty$;
    $C \leftarrow \text{SearchStrategy}(g, W, C, b^*)$;
    while $C \neq \varnothing$ do
        forall the $b \in C$ do
            $cost_b \leftarrow \epsilon(b^f, b)$;
            if $cost_b < cost_{b^*}$ then
                $b^* \leftarrow b$;
                $cost_{b^*} \leftarrow cost_b$;
                $(d_x, d_y) \leftarrow (b_x - b^f_x, b_y - b^f_y)$;
            end if
        end forall
        if $cost_{b^*} > threshold$ then
            $C \leftarrow C \cup \text{Split}(b)$;
        end if
        $C \leftarrow \text{SearchStrategy}(g, W, C, b^*)$;
    end while
    $D \leftarrow D \cup \{(d_x, d_y)\}$;
end forall
```

VSBMA works just like BMA, except for the additional function *Split*. This function divides the block b passed as parameter into four smaller blocks and adds them to the candidate list. These smaller blocks can be further divided into even smaller blocks, until they fall below a fixed error threshold or a minimum block size is reached. This way, a quad-tree structure emerges, with leaf nodes corresponding to blocks of varying sizes [2]. This tree is used in order to properly code the segmentation of the image, as shown on Figure 3.4. The goal is to make the edge of the blocks coincide with the borders of the objects in the scene, forming regions with uniform intensity, just like in picture segmentation [5].

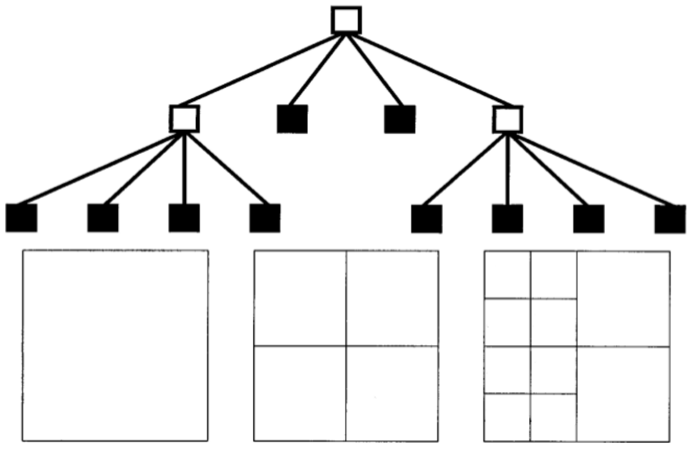


Figure 3.4: Quad tree image segmentation [16].

Figure 3.5 shows a frame with displacement vectors computed through VSBMA drawn over it. Hotter coloured vectors correspond to bigger blocks and colder coloured vectors correspond to smaller blocks. The size of the vectors are proportional to their norms. This example suggests that the motion of more homogeneous regions of the image can be represented by a single vector, while more detailed regions need more vectors in order to properly represent its motion.

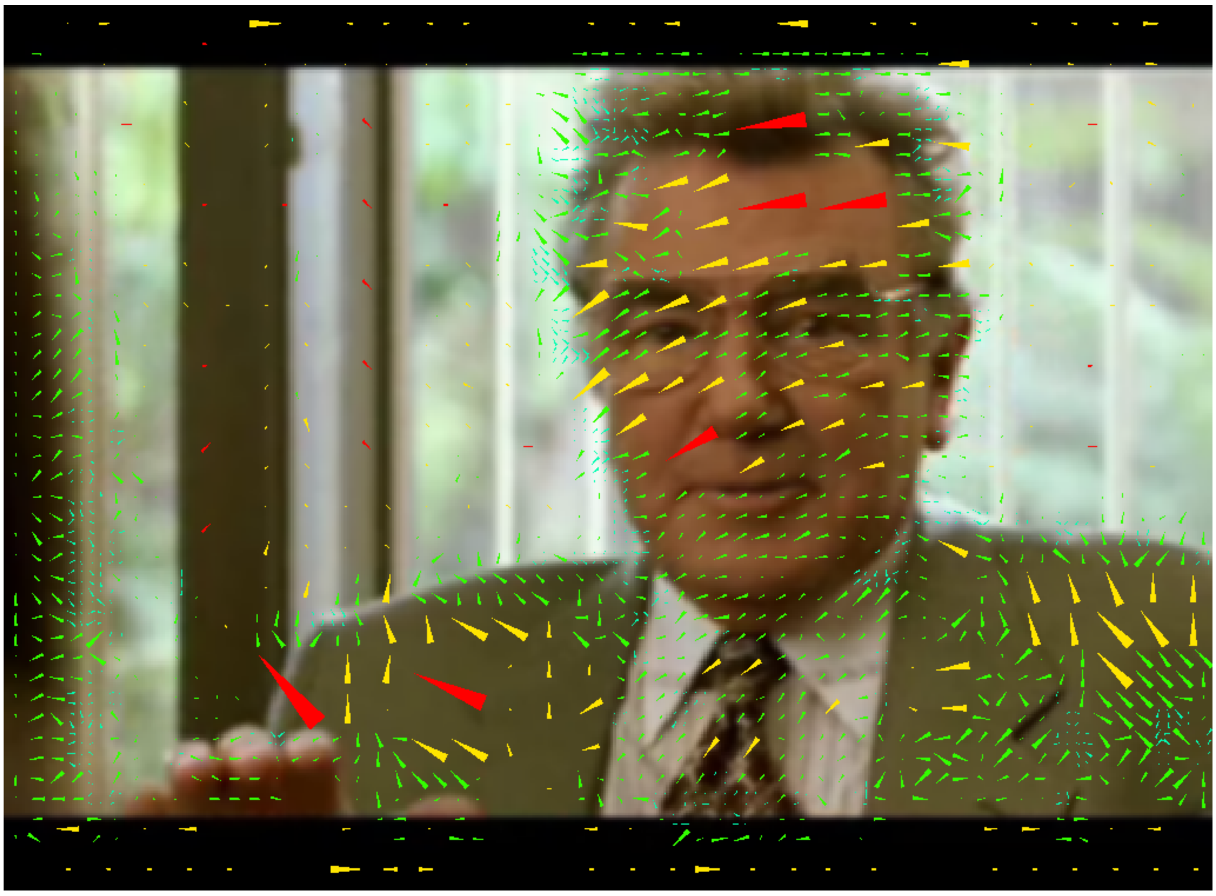


Figure 3.5: VSBMA displacement vectors.

4 Experimental Results

4.1 Efficiency Evaluation

In order to make a efficiency comparison between the algorithms, two criteria are considered: average frame rate and average number of blocks per frame, per video. The frame rate serves as running speed measurement, while the number of blocks measures memory efficiency. The number of blocks can also be related to running speed, since the split operations required to create more blocks also require function calls and block comparisons.

All tests were run on a Intel®Core^{TM}i7-3632QM 2.2GHz with 6GB memory. These tests were made on a 2391 compressed video database, with resolution of 160×120, frame rate of 25fps and average duration of 4 seconds.

BMA efficiency results are shown on Table 4.1. As expected, FS performs one order of magnitude slower than 4SS. It is possible to compute 4SS in real time, since it presents frame rates superior to 25fps on all cases. Also, SAD performs slightly faster than MAD, due to division operations not needed to compute SAD. Since the size of the blocks are fixed, the amount of blocks per frame does not change between tests with the same block size.

Table 4.1: BMA efficiency results. Darker shades of gray highlight the best results.

Parameters			**Avg frame rate**	**Number of blocks**
Block Size	Strategy	Error Function		
8	FS	MAD	6.1492	300
8	FS	SAD	6.4307	300
8	4SS	MAD	53.9517	300
8	4SS	SAD	55.2892	300
16	FS	MAD	7.2568	70
16	FS	SAD	7.4416	70
16	4SS	MAD	60.1998	70
16	4SS	SAD	62.2743	70
24	FS	MAD	8.5012	30
24	FS	SAD	8.9444	30
24	4SS	MAD	68.6432	30
24	4SS	SAD	69.4906	30
32	FS	MAD	10.6040	15
32	FS	SAD	10.9181	15
32	4SS	MAD	79.4199	15
32	4SS	SAD	79.6133	15

For VSBMA, the comparison table is split in two, since one more parameter has to be considered, the splitting threshold. Table 4.2 shows the results using MAD as the error function and Table 4.3 shows the results using SAD. In both cases, the threshold values vary around the mean error, approximately 0.02 for MAD and 5.2 for SAD, to produce variation on the segmentation quad-tree depth.

Table 4.2: VSBMA-MAD efficiency results. Darker shades of gray highlight the best results.

Parameters			**Avg frame rate**	**Avg blocks**
Block Size	Strategy	Threshold		
8	FS	0.01	5.7695	300.4539
8	FS	0.02	6.6796	300.3864
8	FS	0.05	7.3968	300.3242
8	4SS	0.01	40.3573	300.4701
8	4SS	0.02	46.7789	300.4209
8	4SS	0.05	50.6304	300.3678
16	FS	0.01	5.8594	70.3342
16	FS	0.02	6.3491	70.3239
16	FS	0.05	7.5028	70.2836
16	4SS	0.01	43.7113	70.3073
16	4SS	0.02	50.9870	70.2997
16	4SS	0.05	58.1089	70.2661
24	FS	0.01	5.9359	30.1240
24	FS	0.02	7.2140	30.1219
24	FS	0.05	8.3355	30.1049
24	4SS	0.01	45.8166	30.1173
24	4SS	0.02	55.1667	30.1152
24	4SS	0.05	64.1783	30.0991
32	FS	0.01	5.9292	15.1754
32	FS	0.02	8.3002	15.1718
32	FS	0.05	10.8208	15.1478
32	4SS	0.01	46.0447	15.1702
32	4SS	0.02	57.2291	15.1672
32	4SS	0.05	69.9777	15.1445

Through table 4.2, it is possible to observe that the variable block size routine causes some sensible impact on the frame rate, but not on the number of blocks, when using MAD. The frame rate decreases about 10% to 20%, while the number of blocks increases less than 0.7%, when compared to BMA results. Even so, it is still possible to compute 4SS with variable sized blocks in real time.

As shown in Table 4.3, SAD has much more impact on block sizes than MAD. The threshold value plays an important role. When the threshold is low, the segmentation trees are much deeper. This happens because SAD values grow faster as the block sizes get bigger. The smallest differences between pixels, when accumulated in a big 32×32 or 16×16 block, are almost guaranteed to go above the error threshold. As for speed, SAD outperforms MAD, in general, since every split operation for MAD comes at the cost of four extra division operations.

Table 4.3: VSBMA-SAD efficiency results. Darker shades of gray highlight the best results.

Parameters			**Avg frame rate**	**Avg blocks**
Block Size	Strategy	Threshold		
8	FS	3	7.0903	329.2612
8	FS	5	7.6541	311.9919
8	FS	7	7.6081	306.1324
8	FS	10	7.6537	302.9779
8	4SS	3	54.7578	329.5360
8	4SS	5	56.2760	313.0274
8	4SS	7	56.4500	306.9313
8	4SS	10	57.3187	303.5935
16	FS	3	6.1122	143.6343
16	FS	5	6.9305	109.5896
16	FS	7	7.1620	96.5009
16	FS	10	7.3943	87.1845
16	4SS	3	51.0604	142.3622
16	4SS	5	54.4792	109.4417
16	4SS	7	57.3006	96.4079
16	4SS	10	59.5514	87.3178
24	FS	3	5.5233	106.0835
24	FS	5	6.6730	80.0724
24	FS	7	7.6792	65.9465
24	FS	10	8.0449	54.3488
24	4SS	3	42.3479	103.2254
24	4SS	5	49.9035	78.5384
24	4SS	7	53.6687	65.2164
24	4SS	10	57.5309	53.9471
32	FS	3	4.2342	105.4149
32	FS	5	5.9097	71.0749
32	FS	7	7.1490	56.5982
32	FS	10	7.9498	44.9034
32	4SS	3	39.1349	104.1251
32	4SS	5	45.5658	70.8307
32	4SS	7	50.3448	56.3691
32	4SS	10	55.8548	44.8931

4.2 Quality Evaluation on Human Action Recognition Application

Human action recognition have been an active field of research over many years now. Several distinct techniques have been employed, like optical flow [11] and 3-D gradients [8]. But so far, motion estimation has not been thoroughly explored as an action recognition tool, even though it has been used in an array of applications where motion is a relevant feature. Muralidhar et al [13] proposes an architecture for video encoding and compression based on variable block size motion estimation. Amel et al [1] uses motion estimation to detect shot boundaries in video sequences.

In this work, an orientation tensor based descriptor for KTH dataset is generated from both ME algorithms presented. This dataset and the computation of the descriptor is described on the following sections.

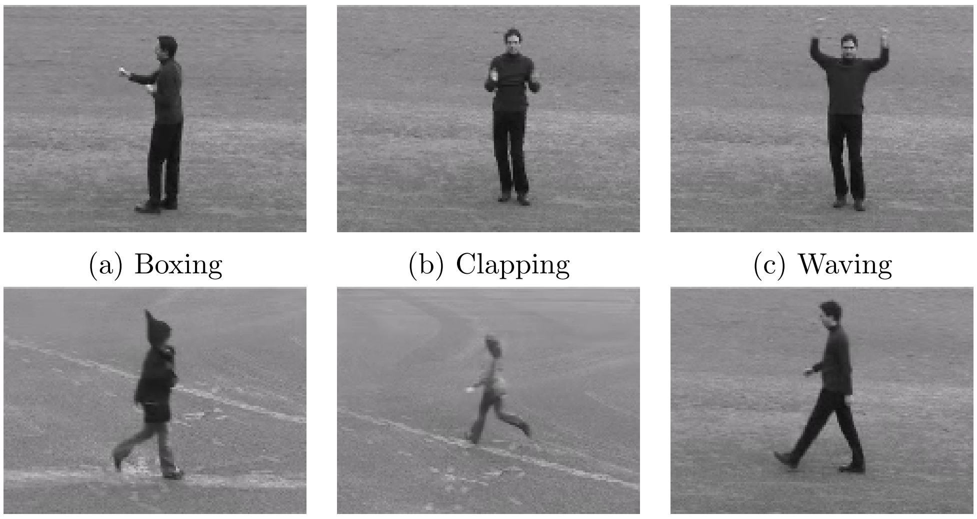
4.2.1 KTH Dataset

KTH is a database of 2391 video sequences of six human actions: walking, running, jogging, boxing, hand waving and hand clapping. These actions are performed by 25 people in four different scenarios: outdoors, outdoors with scale variation, outdoors with different clothes, and indoors. The sequences have a resolution of 160×120 pixels and 25fps frame rate [17].

4.2.2 Descriptor

The motion descriptor is obtained through three steps. The first one is to calculate the displacement vectors with BMA or VSBMA. The second step is to convert these vectors into polar coordinates and build a histogram $\vec{h} = \{h_l\}$, $l \in [1, n]$, where each one of the n bins represents an angle θ interval and is populated as the following equation:

$$h_l = \sum_{i,j} d(i,j) \cdot \omega(i,j), \tag{4.1}$$


(a) Boxing (b) Clapping (c) Waving

(d) Jogging (e) Running (f) Walking

Figure 4.1: KTH dataset categories.

where $\omega(i, j)$ is a Gaussian weighting function with standard deviation $\sigma = 0.01$. This means that each vector adds to the bin with the closest θ value and to the neighbouring bins in a mild way. The third step is to calculate an orientation tensor as such:

$$\mathbf{T} = \vec{h} \cdot \vec{h}^T, \tag{4.2}$$

which is a condensed representation for the motion between a pair of frames.

The tensor for each pair of frames and for all the video sequences in the dataset are then accumulated and normalized with L_2 Frobenius norm, so that it is possible to compare different video sequences regardless of their length or resolution.

4.2.3 Recognition Rates

The quality measure used is the output of a SVM classifier, which takes the descriptors for the whole database and divides them into three groups: a training set, a test set and a validation set. The accuracy shown in Tables 4.4 and 4.5 refer to the percentage of correct action predictions from the descriptor.

Table 4.4 shows the recognition rates for all BMA test runs. These recognition rates indicate that the error function have little to no impact at all on the recognition rates, whereas search strategy and block size play a major role on the descriptor's accuracy. The case with block size 32 achieves poor results due to the big blocks encompassing too heterogeneous regions, and thus failing to capture the detailed motion within. The case with block size 8 shows two very different results, indicating that even though the block size may be appropriate to capture the fine motion, it may also be more subject to noise and compression artefacts, and thus misleadingly capturing background motion. The cases with block sizes 16 and 24 seems to strike a balance between the two previous cases, achieving the best results.

Table 4.4: BMA recognition rates. Darker shades of gray highlight the best results.

Parameters			**Accuracy**
Block Size	Strategy	Error Function	
8	FS	MAD	73.9
8	FS	SAD	73.9
8	4SS	MAD	79.3
8	4SS	SAD	79.3
16	FS	MAD	79.2
16	FS	SAD	79.2
16	4SS	MAD	78.1
16	4SS	SAD	78.1
24	FS	MAD	76.8
24	FS	SAD	76.7
24	4SS	MAD	79.6
24	4SS	SAD	79.6
32	FS	MAD	75.1
32	FS	SAD	75.1
32	4SS	MAD	75.1
32	4SS	SAD	75.1

Table 4.5 shows the best recognition rates obtained with VSBMA. The table presents only SAD results. This is due to MAD tests yielding inferior results than SAD, in all cases tested.

Compared to its BMA counterparts, VSBMA tests show increments on recognition rates ranging from 1% up to 11%. This is a solid improvement, considering VSBMA still retains BMA's real time computation capability.

There is a tendency of increasingly better results as the block sizes get bigger. This happens because the segmentation process tends to include all relevant motion information from the cases with smaller block sizes into the cases with bigger block sizes. The only cases where this tendency is not followed are the cases with block size 16 and one of the cases with block size 24. Note though, that on all of these cases, the segmentation threshold is very far from the average error. On the high threshold case, VSBMA faces the same problem as BMA, where the block may contain more than one direction of motion. On the low threshold cases, the blocks might have become even smaller than the object they are supposed to encapsulate, throughout the segmentation process. Since all the vectors have the same weight in the descriptor computation, regardless of their size, having too many or too few vectors brings back the same problems found with BMA: failure to capture fine motion and sensitivity to noise.

Table 4.5: VSBMA best recognition rates. Darker shades of gray highlight the best results.

Parameters			**Accuracy**
Block Size	Strategy	Threshold	
8	4SS	3	81.2
8	4SS	5	80.3
8	4SS	7	80.8
8	FS	10	80.4
16	4SS	3	81.9
16	FS	5	83.4
16	FS	7	82.9
16	FS	10	81.9
24	4SS	3	80.9
24	4SS	5	82.2
24	4SS	7	82.6
24	4SS	10	82.6
32	4SS	3	83.9
32	FS	5	84.0
32	4SS	7	84.4
32	4SS	10	84.6

5 Conclusion

This work intends to make a comparison between motion estimation methods. For that purpose, it begins with the introduction to the concepts and definitions related to Motion Estimation. Then it presents two approaches for ME: the Block Matching Algorithm and the Variable Size Block Matching Algorithm, both brought along with two different search strategies: the Full Search and the Four Step Search; and two error functions: Sum of Absolute Differences and Mean Absolute Difference.

One important regard is that this work has no intent to compare state-of-the-art Motion Estimation methods nor compete with state-of-the-art human action recognition methods. It is but an experiment on integrating these two fields.

Future works may include several improvements, both on VSBMA and on its use for action recognition. Better exploration of the parameters, adaptive threshold values, block merging operations and different block geometry are a few examples of improvements that can be made on VSBMA. As for human action recognition, the integration of VSBMA and other techniques and datasets is yet to be made.

Bibliography

[1] Abdelati Malek Amel, Ben Abdelali Abdessalem, and Mtibaa Abdellatif. Video shot boundary detection using motion activity descriptor. *Journal of Telecommunications*, 2, 2010.

[2] M.H. Chan, Y.B. Yu, and A.G. Constantinides. Variable size block matching motion compensation with applications to video coding. volume 137, No.4, pages 205–212, 1990.

[3] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Pearson, 2009.

[4] Roger A. Horn and Charles R. Johnson, editors. *Matrix Analysis*. Cambridge University Press, New York, NY, USA, 1986.

[5] Steven L. Horowitz and Theodosios Pavlidis. Picture segmentation by a tree traversal algorithm. *J. ACM*, 23(2), April 1976.

[6] Jaswant R. Jain and Anil K. Jain. Displacement measurement and its application in interframe image coding. *IEEE Transactions on Communications*, COM-29, No 12:1799–1808, 1981.

[7] Jong Won Kim and Sang Uk Lee. Hierarchical variable block size motion estimation technique for motion sequence coding. *Optical Engineering*, 33(8):2553–2561, 1994.

[8] Alexander Kläser, Marcin Marszałek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *British Machine Vision Conference*, pages 995–1004, sep 2008.

[9] Renxiang Li, Bing Zeng, and Ming L. Liou. A new three-step search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 4, No. 4:438–442, 1994.

[10] Jianhua Lu and Ming L. Liou. A simple and efficient search algorithm for block-matching motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 7, No. 2:429–433, 1997.

[11] V.F. Mota, E.A. Perez, M.B. Vieira, L.M. Maciel, F. Precioso, and P.H. Gosselin. A tensor based on optical flow for global description of motion in videos. *Conference on Graphics, Patterns and Images, XXV SIBGRAPI*, 1:298–301, 2012.

[12] Yao Nie and Kai-Kuang Ma. Adaptive rood pattern search for fast block-matching motion estimation. *IEEE Transactions on Image Processing*, 11(12):1442–1449, 2002.

[13] I. Ranjith Kumar P. Muralidhar, C.B. Rama Rao. Efficient architecture for variable block size motion estimation of h.264 video encoder. *International Conference on Solid-State and Integrated Circuit (ICSIC)*, 32:6, 2012.

[14] Lai-Man Po and Wing-Chung Ma. A novel four-step search algorithm for fast block motion estimation. *IEEE Transactions on Circuits and Systems For Video Technology*, 6, No. 3:313–317, 1996.

[15] A. Puri, H.M. Hang, and D.L. Schilling. Interframe coding with variable block-size motion compensation. pages 65–69, 1987.

[16] Injong Rhee, Graham R. Martin, S. Muthukrishnan, and Roger A. Packwood. Quadtree-structured variable-size block-matching motion estimation with minimal error. *IEEE Transactions on Circuits and Systems for Video Technology*, 10, No. 1:42–50, 2000.

[17] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local svm approach. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04)*, ICPR '04, pages 32–36, Washington, DC, USA, 2004. IEEE Computer Society.

[18] Qi Tian, Jie Yu, Q. Xue, N. Sebe, and T.S. Huang. Robust error metric analysis for noise estimation in image indexing. In *Conference on Computer Vision and Pattern Recognition Workshop*, pages 140–140, June 2004.

[19] Shan Zhu and Kai-Kuang Ma. A new diamond search algorithm for fast block-matching motion estimation. *IEEE Transactions on Image Processing*, 9, No. 2:287–290, 2000.