# Motion Estimation Techniques

Béatrice Pesquet-Popescu, Marco Cagnazzo, Frédéric Dufaux
*TELECOM ParisTech*

## Table of content

# Glossary

**2D motion vector field:** the 2D motion vector field is defined as the projection of the 3D objects motion onto the 2D image plane.

**Aperture problem**: when observing a moving structure through a small aperture, different physical motions appear indistinguishable

**Block matching**: Motion estimation algorithms based on the matching of blocks between two frames, with the objective to minimize a dissimilarity measure.

**Brightness constancy**: Assumption that a pixel intensity remains constant along a motion trajectory. In other words, variations in time of the pixel intensity are exclusively due to the objects displacements.

**Dense motion field**: Motion field which represents motion by assigning one motion vector to each image pixel.

**Lambertian reflectance**: An ideal diffusely reflecting surface, whose apparent brightness is the same regardless of an observer view angle.

**Motion compensation**: To perform temporal processing on pixels along motion trajectories, i.e. shifted by a motion vector, instead of on co-located pixels. In video coding, motion compensated prediction refers to the prediction of a block by using a shifted previously encoded block, where the shift correspond to the estimated motion vector.

**Motion trajectory**: The path that a pixel follows through space and time when considering an image sequence as a three-dimensional continuous spatio-temporal field.

**Multi-resolution motion estimation**: Techniques based on a multi-resolution or multi-scale data representation, which first compute a coarse estimate of the motion field at the lowest resolution level and then progressively refine it at successively higher resolution levels.

**Occlusion, Disocclusion**: An occlusion refers to a region or object which is partially or fully hidden by another object closer to the camera. A disocclusion denotes a newly appearing region or object which was previously occluded.

**Optical flow:** the optical flow is defined as the apparent motion of the brightness pattern. In other words, the optical flow captures the spatio-temporal variation of pixel intensities in an image sequence.

**Outlier**: an outlier is sample which markedly deviates from the rest of the data samples.

**Sprite**: a sprite, also known as mosaic or panoramic image, refers to a large composite image obtained by aligning and blending pixels from multiple displaced images

**Rate-distortion**: in the context of lossy data compression, the optimal minimization of the date rate in order to be able to reconstruct the source without exceeding a given distortion, or reciprocally, the minimization of the reconstruction distortion for a given data rate.

**Region of support**: The region of support is the set of image pixels to which a motion model applies.

**Parametric motion model**: A parametric motion model represents the motion of a region characterized by a coherent motion with a set of parameters, also known as motion parameters.

## Nomenclature

| | |
|---|---|
| 2D: | Two-dimensional |
| 3D: | Three-dimensional |
| AE: | Angular Error |
| AVC: | Advanced Video Coding |
| DCT: | Discrete Cosine Transform |
| DFD: | Displaced Frame Difference |
| DWT: | Discrete Wavelet Transform |
| EE: | Error in flow Endpoint |
| FFT: | Fast Fourier Transform |
| FIR: | Finite Impulse Response |
| GMC: | Global Motion Compensation |
| GME: | Global Motion Estimation |
| HEVC: | High Efficiency Video Coding |
| HVS: | Human Visual System |
| HSV: | Hue, Saturation, Value |
| IE: | Interpolation Error |
| IIR: | Infinite Impulse Response |
| LMC: | Local Motion Compensation |

| | |
|---|---|
| LME: | Local Motion Estimation |
| LMedS: | Least Median of Squares |
| LMS: | Least Mean Square |
| LTS: | Least-Trimmed Squares |
| MAE: | Mean Absolute Error |
| MB: | MacroBlock |
| MC: | Motion Compensation |
| ME: | Motion Estimation |
| MPEG: | Moving Picture Experts Group |
| MSE: | Mean Squared Error |
| MVF: | Motion Vector Field |
| NE: | Normalized interpolation Error |
| OBMC: | Overlapped Block Motion Compensation |
| PB: | Prediction Block |
| PSNR: | Peak Signal-to-Noise Ratio |
| RD: | Rate-Distortion |
| RDO: | Rate-Distortion Optimization |
| RMS: | Root-Mean-Square |
| RS: | Recursive Search |
| SAD: | Sum of Absolute Difference |
| SSD: | Sum of Squared Difference |
| SSIM: | Structural SIMilarity |
| SVD: | Singular Value Decomposition |
| TV: | Total Variation |
| VIF: | Visual Information Fidelity |
| VOP: | Video Object Plan |
| ZN-SSD: | Zero-mean Normalized SSD |

# 1 Introduction

Digital video is becoming ubiquitous, thanks to tremendous technological progresses over the last decades, with widespread applications in information technology, telecommunications, consumer electronics and entertainment.

In video sequences, motion is a key source of information. Motion arises due to moving objects in the 3D scene, as well as camera motion. Apparent motion, also known as optical flow, captures the resulting spatio-temporal variations of pixel intensities in successive images of a sequence. The purpose of motion estimation techniques is to recover this information by analyzing the image content. Efficient and accurate motion estimation is an essential component in the domains of image sequence analysis, computer vision and video communication.

In the context of image sequence analysis and computer vision, the objective of motion estimation algorithms is to precisely and faithfully model the motion in the scene. This information is fundamental for video understanding and object tracking. Relevant applications include video surveillance, robotics, autonomous vehicles navigation, human motion analysis, quality control in manufacturing, video search and retrieval, and video restoration. Accurate motion is also important in some video processing tasks such as frame rate conversion or de-interlacing.

As far as video coding is concerned, compression is achieved by exploiting data redundancies in both the spatial and temporal dimensions. Spatial redundancies reduction is largely achieved by transform-coding, e.g. using the Discrete Cosine Transform (DCT) or the Discrete Wavelet Transform (DWT), which effectively compacts the signal energy into a few significant coefficients. In turn, temporal redundancies are reduced by means of predictive coding. Observing that temporal correlation is maximized along motion trajectories, motion compensated prediction is used for this purpose. In this context, the main objective of motion estimation is no longer to find the 'true' motion in the scene, but rather to maximize compression efficiency. In other words, motion vectors should provide a precise prediction of the signal. Moreover, the motion information should enable a compact representation, as it has to be transmitted as overhead in the compressed code stream. Efficient motion estimation is key to achieve high compression in video coding applications such as TV broadcasting, Internet video streaming, digital cinema, DVD, Blu-ray Disc, and video-conferencing,

We should also note a duality between motion estimation and segmentation operations. More specifically, in order to correctly estimate motion, regions of homogeneous motion need to be known. Conversely, for accurate segmentation of these regions, it is necessary to previously perform motion estimation. This problem can be tackled by joint motion estimation and segmentation techniques.

However, in this chapter, we will exclusively focus on the motion estimation aspects.

As a final remark, perception of motion by the Human Visual System (HVS) is also an important topic. Measurement and interpretation of visual motion is discussed in (Hildreth, 1984) and (Ullman, 1979). Better understanding of human perception could help to improve current motion estimation techniques, or lead to new approaches.

Based on the above discussion, motion estimation is clearly a vast and complex topic. The purpose of this chapter is to give a broad overview of motion estimation techniques with a special emphasis on video compression requirements.

As a complement to this chapter, readers may refer to earlier surveys, including (Tziritas and Labit, 1994), (Dufaux and Moscheni, 1995), (Stiller and Konrad, 1999), and (Fleet and Weiss, 2006).

# 2 Motion representation and models

## 2.1. 2D motion vector field and optical flow

While we all have an intuitive understanding of the concept of motion, this notion deserves to be clarified in the case of digital video processing.

Motion is unambiguously defined in the physical three-dimensional (3D) world. However, hen capturing an image, a two-dimensional (2D) projection of the 3D scene is performed. Straightforwardly, the motion arising in an image sequence is also the direct product of the projection of the objects displacement in the 3D scene.

In particular, we can make a distinction between two different concepts: the 2D motion vector field and the optical flow. More specifically, the 2D motion vector field is defined as the projection of the 3D objects motion onto the 2D image plane (Horn, 1986). In contrast, the optical flow is defined as apparent motion of the brightness pattern (Horn, 1986). In other words, the optical flow captures the spatio-temporal variation of pixel intensities.

The 2D motion vector field and the optical flow often coincide, although it is not mandatory. To better understand the difference, let us consider two simple examples. Figure 1 shows a three-dimensional uniform sphere in pure rotation and under constant illumination. In this case, the 2D motion vector field is non-zero. However, this motion leads to a constant brightness pattern, and therefore the optical flow is zero. Conversely, Figure 2 depicts a static reflecting surface, illuminated by a moving light source. In this case, the 2D motion vector field is zero, whereas the optical flow is non-zero.

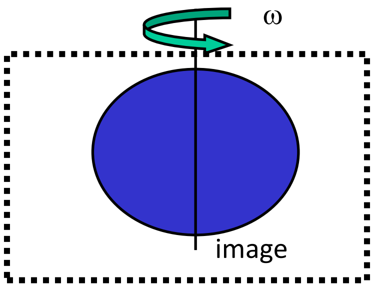

**Figure 1 - Three-dimensional uniform sphere in rotation under constant illumination.**

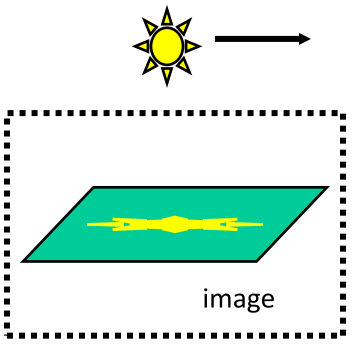

**Figure 2 - Static reflecting surface illuminated by a moving light source.**

In this chapter, we more specifically consider motion estimation techniques for video coding. The objective is to estimate the displacement of pixels between successive images in order to predict the pixel intensities, in other words the optical flow is estimated. However, for the sake of simplicity, we will interchangeably use the terms 2D motion vector field and optical flow throughout the chapter as it is the usage in the video processing community.

## 2.2. The aperture problem

The aperture problem occurs when observing a moving structure through a small aperture. More specifically, under this condition, different physical motions appear indistinguishable. This phenomenon is illustrated in Figure 3. A bar is moving horizontally (top) or vertically (bottom). When seen through a small aperture, which only shows a part of the whole object, both the horizontal and vertical displacements produce the same appearance and therefore cannot be distinguished (right-hand side).

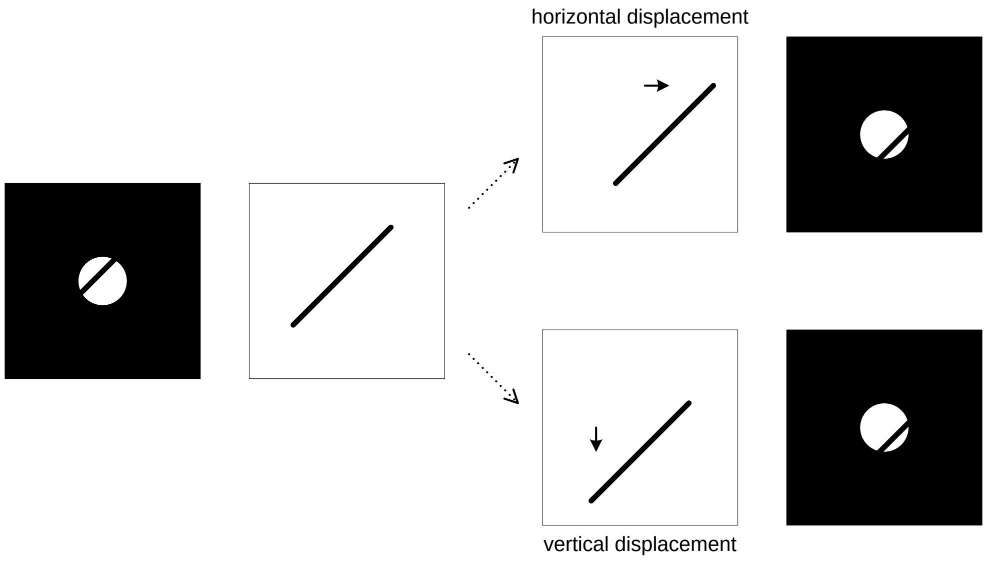

**Figure 3 - Aperture problem: when observing a moving structure through a small aperture, different physical motions appear indistinguishable: (top-right) bar with horizontal displacement seen through aperture, (bottom-right) bar with vertical displacement seen through aperture.**

The same aperture problem happens with motion sensitive neurons in the visual primary cortex (Hildreth, 1984). These neurons, with a finite receptive field, always react to a moving contour that passes through their receptive field, as long as it is consistent with the neuron preferred direction and regardless of the true motion orientation.

## 2.3. Motion representation

In order to define a motion representation, two issues have to be considered. Firstly, a model underlying the motion representation needs to be specified. Secondly, the region of support to which the model applies has to be identified. We discuss these two subjects in more details hereafter.

### 2.3.1. *Brightness constancy motion model and the optical flow equation*

An image sequence can be considered as a three-dimensional continuous spatio-temporal field: *f(x,y,t)*, where *(x,y)* are the spatial coordinates and *t* is the time index. However, in practice we only dispose of a discrete version of this function:

$$f_{n,m,k} = f(nL_1, mL_2, kT) \ , \qquad (1)$$

where $L_1$ and $L_2$ are the sampling steps in the spatial domain, *T* is the temporal sampling step, and *n*, *m* and *k* are integers.

In what follows, we shall make the following assumptions:

- A pixel intensity remains unchanged along a motion trajectory. This assumption is known as the brightness constancy constraint. In other words, the variations in time of the pixel intensity are due to the displacements of different objects present in the scene. The brightness constancy constraint implies that the illumination is uniform and the scene is Lambertian.
- The motion appears locally as a translation (which is the simplest, though effective, motion model).

We then have the following brightness constancy motion model:

$$f(x,y,t+T) = f(x-\Delta x, y-\Delta y, t) \ , \tag{2}$$

where $\Delta x$ and $\Delta y$ are, respectively, the horizontal and vertical components of the displacement. Actually, these components depend on the spatial position, the temporal index $t$ and the temporal sampling step $T$, but for the sake of simplicity, we will not make this dependence appear explicitly in the following.

We can now derive the *motion constraint* or the *optical flow equation*. A first order Taylor expansion of the motion model defined in Eq. (2) leads to:

$$\begin{aligned} f(x,y,t+T) = f(x,y,t) - \Delta x \frac{\partial f}{\partial x}(x,y,t) - \Delta y \frac{\partial f}{\partial y}(x,y,t) + \\ + o(|\Delta x| + |\Delta y|) \end{aligned} \ . \tag{3}$$

Let us define the displacement vector $D$ as

$$D = \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \ . \tag{4}$$

We remark that the vector containing the partial derivatives of the illumination field $f$ with respect to the coordinates $x$ and $y$ is nothing else than the spatial gradient of the image,

$$\nabla f(x,y,t) = \begin{pmatrix} \dfrac{\partial f}{\partial x}(x,y,t) \\ \dfrac{\partial f}{\partial y}(x,y,t) \end{pmatrix} \ . \tag{5}$$

Then we can write the same equation in a vector form, as follows:

$$f(x,y,t+T) = f(x,y,t) - D^T \nabla f(x,y,t) + o(\|D\|) \ , \tag{6}$$

where $D(x,y)^T \nabla f(x,y,t)$ is the inner product of the two vectors.

Now, we define the *velocity vector field* in the sequence, also called *optical flow*, as:

$$V = \begin{pmatrix} u \\ v \end{pmatrix} = \lim_{T \to 0} \frac{D}{T} \ . \tag{7}$$

With this definition, the original equation can be re-written as:

$$V^T \nabla f(x,y,t) + \frac{\partial f}{\partial t}(x,y,t) = 0 \ , \tag{8}$$

and if we develop it, we can write :

$$u(x,y)\frac{\partial f}{\partial x}(x,y,t) + v(x,y)\frac{\partial f}{\partial y}(x,y,t) + \frac{\partial f}{\partial t}(x,y,t) = 0 \ . \tag{9}$$

This equation is essential in all motion field estimation. It is called the *motion constraint* or the *optical flow equation*.

We can make the two following observations:

- the determination of the real displacement in the sequence from the motion constraint is under-determined, since we dispose of only one equation and we have two unknown variables (the optical flow components *u* and *v*. This problem, known as the *aperture problem* and already introduced in Sec. 2.2, can be easily understood from the vector form of the equation. In this case, we have an inner product between *V* and the gradient of the image. Clearly, only the component of the velocity vector parallel to the gradient can be determined. The other component is cancelled in the inner product (see Figure 4).
- in order to solve this ambiguity, one generally has to add an homogeneity constraint for the displacement. Some methods to do this will be discussed in Sec. 3.

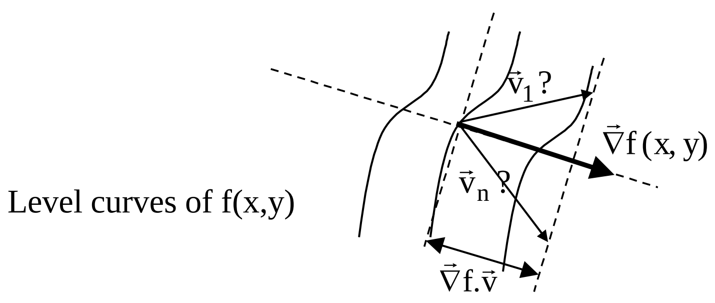

**Figure 4 - Under-determination in the optical flow solution: V1, ..., Vn are several candidates with the same projection of the gradient direction.**

## 2.3.2. *Parametric motion models*

Another approach is to model motion by a set of parameters. Such a model is efficient to represent the motion of a region whose pixels have a coherent motion. Moreover, parametric models result in very compact descriptors, as only a small set of parameters have to be transmitted.

A simple model consists in considering in the image plane a polynomial approximation of the displacement (Tziritas and Labit, 1994). More formally, this polynomial model can be expressed as

$$\begin{pmatrix} d_x \\ d_y \end{pmatrix} = \sum_{i,j} \begin{pmatrix} a_{i,j} \\ b_{i,j} \end{pmatrix} x^i y^j \ , \tag{10}$$

where $d_x$ and $d_y$ are the two components of the motion vector, $a_{i,j}$ and $b_{i,j}$ represent the parameters of the model, and $(x,y)$ are the pixel coordinates.

For example, a first-order approximation leads to a description of the form

$$\begin{aligned} d_x &= t_x + (k + h_1)x + (h_2 - a)y \\ d_y &= t_y + (h_2 + a)x + (k - h_1)y \end{aligned} \ , \tag{11}$$

where $t_x$, $t_y$, $a$, $k$, $h_1$ and $h_2$ are the motion model parameters. In this example, $t_x$ and $t_y$ represent the translation parameters. The parameters $a$, $x_0$, $y_0$, $k$, $h_1$ and $h_2$ are involved in the description of more complex displacements. For instance, as illustrated in Figure 5, a small rotation can be described uniquely using the parameter $a$, a divergent motion vector field will be described by $k$ and a hyperbolic motion will be represented using $h_1$ and $h_2$.

Similar interpretations can be obtained by representing the three-dimensional motions of a rigid body in the space and projecting, under some hypotheses, these motions on the two-dimensional image plane. These hypotheses concern a small vision angle, small displacement orthogonal to the projection plane, small rotations around the axes in the image plane (Tziritas and Labit, 1994).

From this standpoint, polynomial models can be derived:

- translational motion model:

$$\begin{aligned} d_x &= a_1 \\ d_y &= b_1 \end{aligned} \ . \tag{12}$$

- In this case, a very simple zero-order polynomial form is obtained. It can be derived from a rigid translational 3D-motion under orthographic projection. However, this model fails to take into account zoom, rotation, pan and tilt. Nevertheless, thanks to its simplicity, it is widely used in block matching motion estimation techniques (see Sec. 6). Straightforwardly, any 2D shape is preserved after motion compensation using such a model.

- affine motion model:

$$\begin{aligned} d_x &= a_1 + a_2 x + a_3 y \\ d_y &= b_1 + b_2 x + b_3 y \end{aligned} . \tag{13}$$

A first order polynomial form, the affine motion model can be derived from the 3D affine motion of a planar surface under orthographic projection. With the affine model, motion compensation conserves parallel lines.

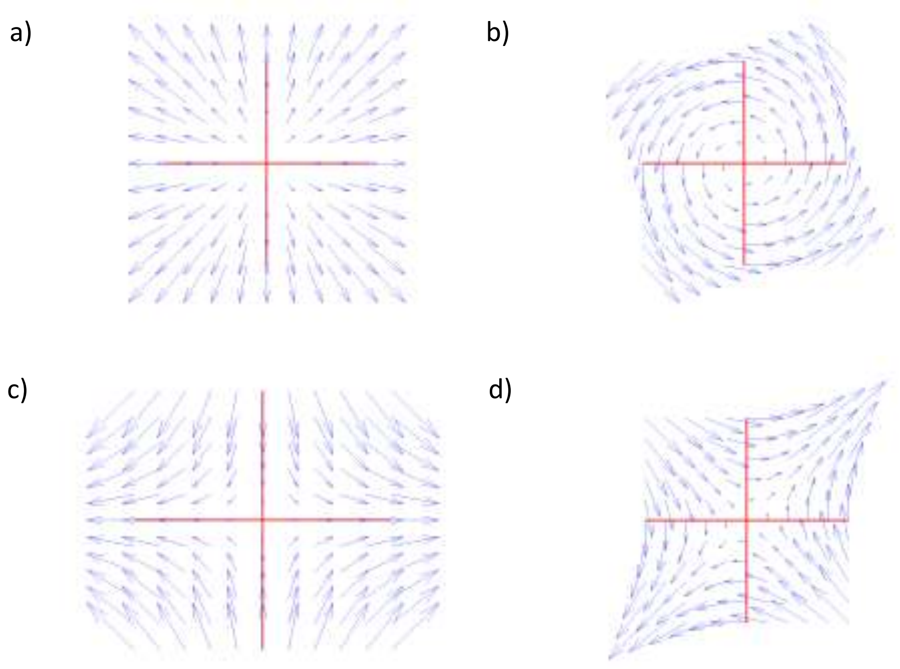

**Figure 5 - Parametric models for motion vector fields : a) divergence (*k* = 0.5); b) rotation (*a* = 0.5); c) hyperbolic field (*h₁* = 0.5); d) hyperbolic field (*h₂* = 0.5).**

- perspective (or projective) motion model:

$$\begin{aligned} d_x &= \frac{a_1 + a_2 x + a_3 y}{1 + a_4 x + b_4 y} \\ d_y &= \frac{b_1 + b_2 x + b_3 y}{1 + a_4 x + b_4 y} \end{aligned} . \tag{14}$$

Considering the 3D affine motion of a planar surface under perspective projection leads to the perspective (or projective) motion model with 8 parameters. Clearly, the affine model is a special case of the perspective one, with $a_4$=$b_4$=0. With perspective model, lines remain lines after motion compensation. One drawback of this model is the difficulty to accurately estimate the parameters in the denominator term.

- quadratic motion model:

$$\begin{aligned} d_x &= a_1 + a_2 x + a_3 y + a_4 xy + a_5 x^2 + a_6 y^2 \\ d_y &= b_1 + b_2 x + b_3 y + b_4 xy + b_5 x^2 + b_6 y^2 \end{aligned} . \tag{15}$$

  This second order polynomial form can be derived from a 3D affine motion of a parabolic surface under orthographic projection. Again, the affine model is a special case, with $a_4$= $a_5$= $a_6$=$b_4$= $b_5$= $b_6$=0. The perspective model is also included as a Taylor approximation. Note that the quadratic model does no longer preserve lines after motion compensation.

- bilinear motion model:

$$\begin{aligned} d_x &= a_1 + a_2 x + a_3 y + a_4 xy \\ d_y &= b_1 + b_2 x + b_3 y + b_4 xy \end{aligned} . \tag{16}$$

  The bilinear model is obtained from the quadratic model by discarding square terms. However, it is not related to any physical 3D motion.

Clearly, the more complex a motion model, the better its ability to precisely represent more complex motions. However, the number of parameters is also higher. Therefore, it makes the estimation process more difficult and complex, and possibly prone to errors. Moreover, a complex motion model does not allow for a compact representation of the motion information.

### 2.3.3. *Region of support*

The region of support is the set of image pixels to which a motion model applies. We can distinguish the following four cases:

- Pixel-based: A motion vector is assigned to each pixel of the image, resulting in a dense motion field. It has the advantage to provide a precise description of the motion. However, from a video coding viewpoint, it entails a costly representation resulting in a large overhead for motion information.
- Region-based: The motion model is applied to a region of the image which is characterized by a coherent motion. In this case, moving objects in the scene have to be identified. In (Diehl, 1991), a method is presented for

segmenting video scenes hierarchically into differently moving objects. A system for representing moving images with sets of overlapping layers using motion analysis in proposed in (Wang and Adelson, 1994). Segmentation-based motion estimation and spatio-temporal segmentation are addressed in (Dufaux and Moscheni, 1996). In (Chang, Tekalp, Sezan, 1997) a Bayesian framework is presented that combines motion estimation and segmentation.

In the context of video coding, such a representation requires to transmit the shape of the region, which entails a bit rate overhead.

- Block-based: As a special case of the region-based support, a very frequent choice is to simply partition the image into blocks. If the block size is sufficiently small, then the assumption that the block is moving in a coherent way is likely to be valid. Another advantage of a block partitioning is that it does not require additional information to represent the shape of the region.
- Global: The region of support simply encompasses the whole image. This case is especially suited to efficiently estimate camera motion. Indeed, camera motion such as dolly, track, boom, pan, tilt or roll, is an essential cinematic technique.

The choices of the region of support and the motion model are closely intertwined. When using a complex parametric motion model, which can handle complex motions, a larger region of support can effectively be used. Conversely, a simple model is often sufficient in conjunction with a small region of support.

### 2.3.4. *Considerations from a video coding viewpoint*

We can clearly observe that, for video coding applications, selection of an optimal motion representation brings up contradictory requirements. On the one hand, the representation has to accurately characterize the motion information, even in the case of complex scenes with multiple moving objects. On the other hand, the representation needs to be compact for further efficient coding, transmission or storage.

From a video coding viewpoint, the first two criteria should lead, foremost, to a good prediction (in other words, small prediction error), and simultaneously, to a low overhead information (therefore, easy-to-encode motion vector fields). This trade-off shows that the ultimate goal is not to obtain the real motion, even though a motion field close to the true motion in the scene avoids artificial discontinuities and reduces the transmission cost of the motion information, but to globally optimize the video coding scheme in a rate-distortion sense.

In practice, in video coding, most schemes are based on a translational motion model combined with a block-based partitioning, as a good trade-off between motion accuracy and the overhead to represent motion information.

# 3 Optical flow approaches

In this section, we present optical flow estimation approaches, which have been mainly developed for image sequence analysis applications and computer vision (Horn, 1986).

The methods entering this category are mainly based on gradient techniques, with the common objective to solve the motion constraint or optical flow equation, as defined in Sec 2.3.1. More precisely, we present two classical optical flow algorithms: Horn-Schunk (Horn and Schunk, 1981) and Lucas-Kanade (Lucas and Kanade, 1981).

## 3.1. Horn-Schunck

We now present one of the methods using this differential approach, based on a global optimization.

The method we describe here uses a minimization of a cost function, which combines the optimization of the optical flow constraint, as defined in Eq. (9), with a constraint on the smoothness of the motion vector field. This method is referred to as the *Horn-Schunck* algorithm (Horn and Schunck, 1981).

The cost function $J_{HS}(V)$ is defined as follow:

$$J_{HS}(V) = \iint_{\Re} \left[ u\frac{\partial f}{\partial x} + v\frac{\partial f}{\partial y} + \frac{\partial f}{\partial t} \right]^2 dxdy + \lambda \iint_{\Re} \left[ \|\nabla u\|^2 + \|\nabla v\|^2 \right] dxdy \qquad (17)$$

,

where $\Re$ is the support on which the optimization is performed, namely it can be a region or the entire image, and λ is a positive constant.

The first term in the above expression is a mean square error on the motion constraint, while the second one is a regularization term: it allows ensuring that the gradient of the motion vector field takes small values ("smoothness" of the solution). In this criterion, λ is the regularization constant, which allows to trade off the influence of the regularization term and the minimization of the motion constraint. Remark that the integrals are on an arbitrary region where the motion is homogeneous, meaning that this technique can be adapted to region-based motion estimation or to a joint segmentation-motion estimation solution.

After some mathematical developments involving the minimization of the cost function, we arrive at the following solution for the two components of the motion vector field:

$$\lambda\nabla^2 u = \frac{\partial f}{\partial x}\left[u\frac{\partial f}{\partial x} + v\frac{\partial f}{\partial y} + \frac{\partial f}{\partial t}\right], \tag{18}$$

$$\lambda\nabla^2 v = \frac{\partial f}{\partial y}\left[u\frac{\partial f}{\partial x} + v\frac{\partial f}{\partial y} + \frac{\partial f}{\partial t}\right]. \tag{19}$$

where $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is the Laplacian operator. The following observations can be made concerning this method:

- This global optimization is quite complex, involving the resolution of a system of partial differential equations.
- The choice of $\lambda$ is critical. On the one hand, it leads to a different smoothness of the field. On the other hand, it influences the numerical stability of the system.

## 3.2. Lucas-Kanade

The *Lucas-Kanade* method is another classical approach for optical flow estimation (Lucas and Kanade, 1981). In this method, the assumption is made that the optical flow is approximately constant in a small neighborhood, instead of adding a smoothness constraint as in the Horn-Schunck algorithm.

More precisely, the motion vector *V*=(*u*,*v*) is assumed to be constant in a local region $\Re$ around the pixel being processed. In other words, the optical flow equation expressed at all pixel locations ($x_i$,$y_i$) ϵ $\Re$ , i=1, ... n, leads to a set of *n* equations:

$$\begin{aligned} &u\frac{\partial f}{\partial x}(x_1,y_1) + v\frac{\partial f}{\partial y}(x_1,y_1) + \frac{\partial f}{\partial t}(x_1,y_1) = 0 \\ &\vdots \\ &u\frac{\partial f}{\partial x}(x_n,y_n) + v\frac{\partial f}{\partial y}(x_n,y_n) + \frac{\partial f}{\partial t}(x_n,y_n) = 0 \end{aligned} \quad . \tag{20}$$

In the Lucas-Kanade method, the optical flow is then obtained by Least Squares minimization of the following cost function $J_{LK}(V)$

$$J_{LK}(V)=\sum_{i=1\ldots n} w_i\left(u\frac{\partial f}{\partial x}(x_i,y_i)+v\frac{\partial f}{\partial y}(x_i,y_i)+\frac{\partial f}{\partial t}(x_i,y_i)\right)^2 , \qquad (21)$$

where the weighting factors $w_i$ have been introduced to give larger weights to pixel locations $(x_i,y_i)$ which are near the region center.

Note that this is also the assumption made in block matching (or region matching) algorithms.

## 3.3. Discussion

Optical flow approaches result in dense motion vector field (one vector per pixel), which is qualitatively interesting for motion analysis applications. However, they also have several weaknesses:

- The derivation of the optical flow equation is based on a first order Taylor series expansion. This only holds under the hypothesis that the motion between two frames is small.
- The equations are written for continuous-time and continuous-space variables and require to estimate the image gradient. For solving them, we need to discretize these variables. However, this sampling process introduces errors in the solution. In particular, gradient computation is sensitive to noise and is therefore subject to errors.
- The smoothness constraint (Horn-Schunck) or the local uniformity constraint (Lucas-Kanade) results in a poor accuracy along moving objects boundaries.

In order to address the above shortcomings, numerous advances have been proposed resulting in improved performance.

Instead of the above formulation based on a L2 norm, the L1 norm is also a frequent choice (Brox, Bruhn, Papenberg *et al.*, 2004), both for the optical flow equation and for the additional constraint (e.g. smoothness). In this case, one refers to the class of Total Variation (TV) methods.

A model to handle changes in illumination and blur is proposed in (Seitz and Baker, 2009). It also includes spatial weighting of the smoothnes constraint. Anisotropic smoothness weighting is considered in (Zimmer, Bruhn, Weickert *et al.*, 2009). The method also applies different weighting to color channels in the HSV color space.

A novel extended coarse-to-fine refinement framework is introduced in (Xu, Jia and Matsushita, 2012). The reliance on the initial flow estimates propagated from a coarser level is reduced. Hence, motion details are better preserved at each scale. An adaptation of the objective function to handle outliers and a new optimization procedure are also proposed.

For a more detailed and in-depth tutorial on optical flow techniques, the reader is referred to (Fleet and Weiss, 2006).

An extensive performance comparison of several algorithms is given in (Barron, Fleet and Beauchemin, 1994). More recently, a new set of benchmarks and evaluation methods for optical flow techniques have been introduced in (Baker, S. Scharstein, D. Lewis, J.P. et al., 2011). A taxonomy of optical flow algorithms is also presented, along with a discussion of recent works. At the time of this writing, the method by (Xu, Jia and Matsushita, 2012) is one of the best performing techniques reported in the on-line optical flow evaluation database at http://vision.middlebury.edu/flow/.

From a video coding perspective, the fact that a dense motion field is obtained is not necessarily a positive point. Indeed, this field has to be encoded, which may result in a high bit rate overhead.

## 4 Pel-recursive approaches

Pel-recursive approaches are among the earliest methods to estimate motion with the objective of video coding applications. Essentially, these techniques recursively estimate the displacement which minimizes the Displaced Frame Difference (DFD) defined as

$$DFD = f(n,m,k) - f(n-\Delta x, m-\Delta y, k-1) \ , \tag{22}$$

Note that here we consider the discrete spatial and temporal coordinates. The image is typically scanned in a raster order, performing the recursion on a pel-by-pel basis.

### 4.1. Netravali-Robbins

The first pel-recursive motion estimation algorithm was proposed by (Netravali and Robbins, 1979). This method aims at minimizing the square of the DFD using a steepest descent technique. Considering the current pixel position (*n,m*) and the motion vector $D_{n,m}^{(i)}$ at iteration *i*, the recursion is defined as

$$D_{n,m}^{(i+1)} = D_{n,m}^{(i)} - \frac{\varepsilon}{2} \nabla_D DFD^2(n,m,\Delta x,\Delta y) \ , \tag{23}$$

where $\nabla_D$ denotes the gradient with respect to *D*, and ε>0 is a constant gain. Developing the second term, we straightforwardly obtain

$$\nabla_D DFD^2(n,m,\Delta x,\Delta y) = 2DFD(n,m,\Delta x,\Delta y)\cdot \nabla f(n-\Delta x, m-\Delta y, t-T) \quad . \qquad (24)$$

Therefore, by substitution, the motion vector update can be rewritten as

$$D_{n,m}^{(i+1)} = D_{n,m}^{(i)} - \varepsilon\, DFD(n,m,\Delta x,\Delta y)\cdot \nabla f(n-\Delta x, m-\Delta y, t-T) \ . \qquad (25)$$

The iteration from *i* to *i*+1 can be carried out at one given pixel location, or from one location to the next one.

In order to make a more robust estimation of the DFD, it can be computed in a neighborhood around the pixel (*n,m*). Furthermore, bilinear interpolation is used when the displacement is not an integer numbers of pixels.

## 4.2. Cafforio-Rocca

In this section, we describe an improved pel-recursive algorithm, known as *Cafforio-Rocca algorithm* (Cafforio and Rocca, 1983).

At each pixel location, in a raster scan order, the following operations are performed:

1) Initialization of the motion vector

As an initial estimate of the motion vector, the vector obtained at the previous position in the scanning order is chosen. Namely, if (*n,m*) is the current position, and $D_{n,m}$ the motion vector at this position, then the initial estimate will be:

$$D_{n,m}^{(0)} = D_{n-1,m} \ . \qquad (26)$$

where $D_{n-1,m}$ is the motion vector estimated at the previous point in the scanning order.

Alternatives can be considered to compute this initial estimate. For example, another possibility is to consider an average of several of the previously estimated motion vectors. Note, however, that the causality induced by the scanning order has to be respected, in order to be able to perform the same operations at the decoder.

2) Reliability test of the initial estimate

After this initialization, the next step is to check that the value of the a priori estimation is correct. This test is necessary in order to take into account the following sources of errors:

- strong variations of the motion vectors from one position to the next, related to objects with different motions;
- the divergence of the motion estimation algorithm itself.

For this purpose, let us consider therefore the criterion built on the following expressions:

$$\begin{aligned} R(n,m) = &\left| f(n,m,k) - f(n-\Delta x^{(0)}, m-\Delta y^{(0)}, k-1) \right| - \\ &\left| f(n,m,k) - f(n,m,k-1) \right| \end{aligned} , \tag{27}$$

with

$$D^{(0)}_{n,m} = \begin{pmatrix} \Delta x^{(0)} \\ \Delta y^{(0)} \end{pmatrix} . \tag{28}$$

In other words, *R* is made of two terms. The first one is the estimation error resulting from the a priori initial prediction and the second one is the simple inter-image difference. We also define a threshold value *s*>0. If (*R*>*s*), the motion estimation is re-initialized to 0, otherwise the a priori estimation at pixel *(n,m)*, as defined in step 1, is kept. More specifically, the criterion is:

$$\begin{array}{ll} \text{if}\left(R(n,m) > s\right) & D^{(1)}_{n,m} = 0 \\ \text{otherwise} & D^{(1)}_{n,m} = D^{(0)}_{n,m} \end{array} . \tag{29}$$

3) Refinement of the estimation

This is the most important step of the algorithm, updating the motion vector. The actual value of the motion vector is computed as the initial value (validated or re-initialized by the reliability test) plus an update vector, as follows:

$$D_{n,m} = D^{(1)}_{n,m} + \delta D_{n,m} . \tag{30}$$

The update vector is chosen so as to minimize a cost function as a trade-off between the prediction error and the amplitude of the update vector. Note that it is similar to the one used in Horn-Schunck (See Sec. 3.1). More precisely, the cost function is written as:

$$\begin{aligned} J(\delta D) = &\left( f(n,m,k) - f(n-\Delta x^{(1)} - \delta x, m-\Delta y^{(1)} - \delta y, k-1) \right)^2 \\ &+ \lambda \left\| \delta D \right\| \end{aligned} , \tag{31}$$

with the previous motion estimate

$$D^{(1)}_{n,m} = \begin{pmatrix} \Delta x^{(1)} \\ \Delta y^{(1)} \end{pmatrix} , \tag{32}$$

the update vector

$$\delta D_{n,m} = \begin{pmatrix} \delta x \\ \delta y \end{pmatrix} , \tag{33}$$

and λ is a positive regularization constant.

With a first order Taylor expansion around the previous estimation, we can write the previous image compensated with the updated vector:

$$\begin{aligned} &f(n-\Delta x^{(1)}-\delta x, m-\Delta y^{(1)}-\delta y, k-1)= \\ &f(n-\Delta x^{(1)}, m-\Delta y^{(1)}, k-1)-\delta D^T \varphi_{n,m}+O\|\delta D\| \end{aligned}, \tag{34}$$

where

$$\varphi_{n,m}=\nabla f(n-\Delta x^{(1)}, m-\Delta y^{(1)}, k-1)\ , \tag{35}$$

is the gradient of the previous image, compensated with the initial motion vector. Thus we can re-write the cost function as

$$J(\delta D)=\left[\varepsilon_{n,m}+\delta D^T \varphi_{n,m}\right]^2+\lambda\|\delta D\|\ , \tag{36}$$

where we introduced

$$\begin{aligned} \varepsilon_{n,m} = \ & f(n-\Delta x^{(1)}-\delta x, m-\Delta y^{(1)}-\delta y, k-1) \\ & -f(n-\Delta x^{(1)}, m-\Delta y^{(1)}, k-1) \end{aligned} \tag{37}$$

which denotes the prediction error using the initial motion vector in order to simplify the notation. The minimum of the cost function with respect to the update vector is obtained by cancelling its derivative. This leads to the following value of the update vector:

$$\delta D_{n,m}=-\frac{\varphi_{n,m}\varepsilon_{n,m}}{\lambda+\|\varphi_{n,m}\|^2}\ . \tag{38}$$

Straightforwardly, the last step of the algorithm consists in updating the motion vector with this quantity:

$$D_{n,m}=D_{n,m}^{(1)}-\frac{\varphi_{n,m}\varepsilon_{n,m}}{\lambda+\|\varphi_{n,m}\|^2}\ . \tag{39}$$

The choice of the regularization parameter λ and of the threshold *s* is very important and can lead to the divergence of the algorithm when they are not well managed.

## 4.3. Discussion

The following remarks can be made regarding pel-recursive approaches such as Netravali-Robbins or Cafforio-Rocca. Firstly, these pel-recursive techniques are relatively easy to implement. Convergence can be slow and the motion estimation is not always of very good quality. This is especially the case when the displacement is important or when there are large motion discontinuities at object borders. This limitation is mostly due to the recursive nature of the algorithm with a causality constraint.

In order to improve convergence and performance, other pel-recursive motion estimation techniques have been proposed. Better adaptation to local statistics is obtained in (Walker and Rao, 1984). A Wiener-based displacement estimation algorithm is introduced in (Biemond, Looijenga and Boekee, 1987). A multiple frames model-based approach is proposed in (Efstratiadis and Katsaggelos, 1990). Finally, a multiple mask regularization technique is introduced in (Efstratiadis and Katsaggelos, 1993).

There are some advantages of the pel-recursive methods over other existing approaches. Provided that the recursion has a quick enough convergence (i.e., it can handle the motion discontinuities), the pel-recursive algorithms may overcome the problem of multiple moving objects. Moreover, as the update vector calculation is based only on previously transmitted data (causality), the decoder can estimate the same motion vector field as the encoder. No overhead information is thus required for transmitting the motion vector field, which is of course a big advantage of these methods in video coding applications. The counterpart is that the decoder has to do the same operations as the encoder in order to find the motion vectors, which will lead to an increased computational complexity.

# 5 Transform-domain approaches

The common idea of transform-domain approaches is to estimate the motion vectors from a measure performed in the transform domain, for instance with a Fourier transform, a Discrete Cosine Transform (DCT) or a Discrete Wavelet Transform (DWT). For this purpose, the effect of the 2D motion on the characteristics of the transform has to be studied.

## 5.1. Motion estimation in the Fourier or DCT domain

Let us first consider the case of motion estimation in the Fourier domain (Haskell, 1974), (Fleet and Jepson, 1990). In this case, a translation in the spatial domain corresponds to a phase-change of the Fourier coefficients. Instead of minimizing a dissimilarity, for instance using Mean Square Error (MSE) or Mean Absolute Error (MAE) as in block matching techniques (see Sec. 6), a phase correlation between blocks is usually performed. Therefore, these methods are sometimes referred to as *phase-correlation* methods.

A common drawback of such methods is that it is difficult to characterize complex motions in the transform domain. As a consequence, a simple motion model has to be adopted, which may adversely affect the precision of the motion estimation process. Nevertheless, correlation-based methods have been successfully applied for the estimation of global motion.

Similarly, the same ideas can be applied in the DCT domain (Koc and Ray Liu, 1994). DCT seems more appropriate than the Fourier transform. Indeed, most video coding schemes are based on a DCT of the residual signal, and it is therefore more coherent to use the same transform for motion estimation and for the prediction error coding. However, such an approach faces the same difficulties as for the Fourier representations.

## 6 Block matching approaches

Block matching methods have been more specifically developed in the framework of image sequence coding. All video coding standards to date, including H.264/AVC (Wiegand, Sullivan, Bjontegaard *et al.*, 2003) and the forthcoming HEVC (Ohm and Sullivan, 2013), are based on this paradigm. This class of techniques is therefore more thoroughly discussed in this section.

Block matching methods enter the category of matching primitive techniques. The aim is to minimize a dissimilarity measure. In particular, the fact that the same motion vector is estimated over an entire block can be seen as an additional "smoothness" constraint for solving the motion equation. In this way, the under-determined system can be solved. In addition, these methods will be quite robust to noise, which is not always the case with methods providing dense motion fields, such as the global optimization and the pel-recursive techniques.

Let us introduce some notation. Let us consider images of size *N* x *M*. A block $B_{p,q}$ is a set of indexes defined starting from (*p*,*q*) and whose size is *P* x *Q*:

$$B_{p,q} = \{p, p+1, \dots, p+P-1\} \times \{q, q+1, \dots, q+Q-1\}. \qquad (40)$$

The current image is divided into non-overlapping rectangular blocks as shown in Figure 6. The typical values for *P* and *Q* are 4, 8 or 16, but larger block-sizes are possible in the upcoming standard HEVC (Ohm and Sullivan, 2013).

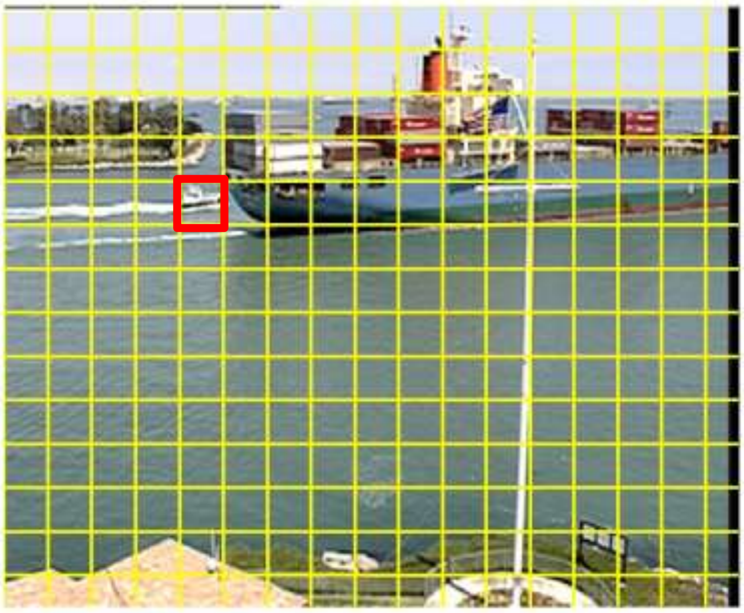

**Figure 6 - An image from the *container* sequences, divided into non-overlapping blocks of 32 x 32 pixels. A single block $B_{p,q}$ is highlighted.**

In any case, for all the pixels in the block, a single motion vector is computed. With a small abuse of notation, we will refer to the vector of image intensity values within the block as $f_k(B_{p,q})$:

$$f_k(B_{p,q}) = \left[f(p,q,k), f(p+1,q,k),\ldots, f(p+P-1,q,k),\ldots,\right.$$
$$\left. f(p+P-1,q+Q-1,k)\right]^T \quad . \qquad (41)$$

The block matching motion estimation is performed by computing a similarity measure between $f_k(B_{p,q})$ and $f_h(B_{p-i,q-j})$, *i.e.* by comparing the intensity values in a block of the current image $f_k$ and those in a block of a second image $f_h$, called *reference.* This second block is displaced from the initial position *(p,q)* by a vector *D*=(*i,j*) that is called *candidate* motion vector.

The case where *h=k-1,*(i.e. when we look for the displacement of blocks in the previous image) is called *forward* motion estimation and is shown in Figure 7.

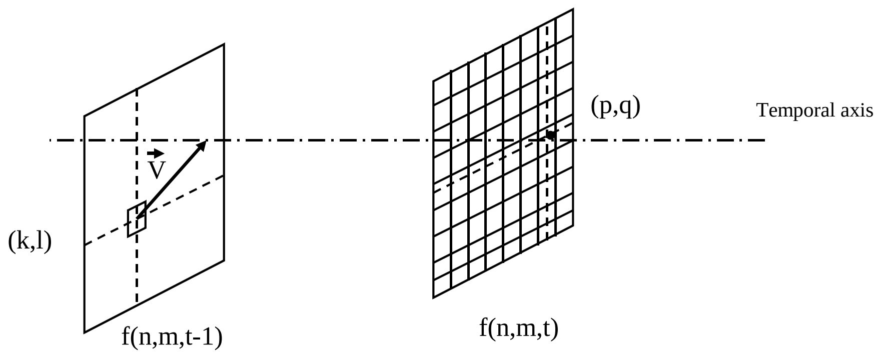

**Figure 7 - Block matching algorithm with forward motion estimation.**

The candidate vector that minimizes a suitable similarity measure between the blocks is the estimated vector for the block $B_{p,q}$ :

$$\left(\hat{i},\hat{j}\right)=\underset{(i,j)\in W}{\arg\min}\, d\left[f_k(B_{p,q}),\, f_h(B_{p-i,q-j})\right], \tag{42}$$

where *W* is a suitable set of candidate vectors, commonly called search window. For a given block, the function to be minimized in Eq. (42) depends only on the candidate vector, so we will shorten it as *J(i,j).* As a consequence, Eq. (42) becomes:

$$\left(\hat{i},\hat{j}\right)=\underset{(i,j)\in W}{\arg\min}\, J(i,j), \tag{43}$$

Equations (42) and (43) contain the essence of the method. The various versions of block-matching differ among them for:

1. The search strategy, *i.e.* how the search window is scanned in order to find the best vector
2. The matching criterion, that is the function *d*(.,.) used in Eq. (42)
3. The block size and shape

These elements are addressed in the next subsections, which will be concluded by considering some other issues such as sub-pixel accuracy and recent advances.

## 6.1. Search window and search strategy

The motion vector is selected within a suitable set of candidate vectors. This set is called *search window* and represents the area where the most similar block will be searched. In the most common case, the search window corresponds to a rectangular area centered in the block $B_{p,q}$ of the reference image. An example of search window is shown in Figure 8.

The structure of the search windows has a huge impact on both the complexity of the motion estimation algorithm and on its precision. Therefore the choices of the search window and of the associated search strategy are critical for a motion estimation algorithm.

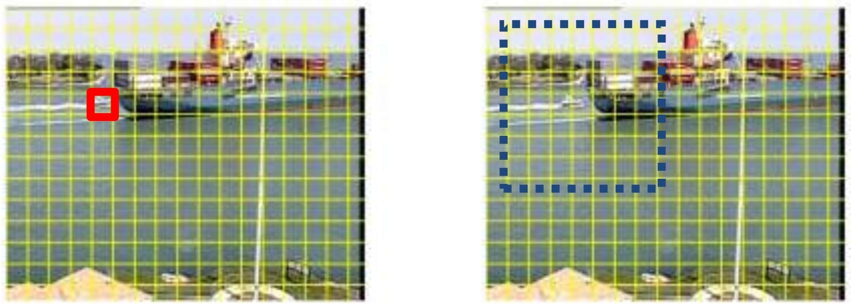

**Figure 8 - Left: the current image with the block $B_{p,q}$ highlighted. Right: the reference image. The search window centered in *(p,q)* is shown.**

### 6.1.1. *Full search*

A very intuitive choice for the search area is the whole reference image: the block $B_{p,q}$ is compared to all possible blocks of the reference image. However this approach is extremely complex, since the number of candidate blocks is ($N$-$P$) x ($M$-$Q$) ≈ $NM$, and for each candidate block we have to compute the similarity measure.

However, it is not necessary to consider all pixels in the reference image: according to the characteristics of the movement and to the resolution of the image, typically it suffices to consider a rectangular area centered in the position *(p,q)*.

Formally, the search window $W$ is defined as a set of vectors:

$$W = \{-A,\ldots,-1,0,1,\ldots,A\} \times \{-B,\ldots,-1,0,1,\ldots,B\} . \quad (44)$$

The horizontal and vertical size of the search area can be different, according to the fact that horizontal movements are usually wider than vertical ones in natural videos. However, for the sake of simplicity, $A$ and $B$ have very often the same value, as we will assume in the following. Also, we refer to $n$=2$A$+1 as the *width* of the search windows.

In order to solve Eqs. (42) or (43) with this definition of $W$, we need to compute the criterion for $n^2$ candidates: this approach is called *full search*.

Full search assures to find the best motion vector (i.e. the one minimizing the criterion) within all displacements of no more than $A$ pixels in whichever direction, but has a relatively large complexity, proportional to $n^2$. However, this technique lends itself to a parallel implementation since the estimation can be performed independently for each block of the current image.

Other techniques exist that allow to find a motion vector with a smaller complexity, but they are suboptimal, in the sense that they cannot assure that the best vector among all those having components bounded by $A$ will be found. The most popular sub-optimal search strategies are described in the following.

### 6.1.2. *Fast search: definitions*

In order to reduce the computational complexity of the full search method presented above, sub-optimal techniques have been proposed. The basic idea is to select a subset of $\{-A,\dots A\}^2$ as search window. In this way, we still may estimate large movements, but with less than $n^2$ computations of the criterion.

Fast search techniques are generally iterative. We start with a search window made up of a few and relatively large vectors, and with an initial candidate vector $(i_0,j_0)$, e.g. (0,0).

At the $k$-th step, the current best vector $(i_k,j_k)$ is selected by full search in the current search window $W_k$. Then, the search window is modified: typically it is centered on $(i_k,j_k)$ and possibly scaled. A proper stopping condition is needed: for example a maximum number of iterations or a condition on the elements of $W_k$.

The general structure of the algorithm is the following:

1. Initialization: we set $k$=0, and we choose $W_1$ and $(i_0 j_0)$
2. While the stop condition is not met,
    a. $k=k+1$
    b. $(i_k, j_k) = \arg\min_{i,j\in W_k} J(i,j)$
    c. $W_{k+1} = \phi[W_k, (i_k, j_k)]$
3. $(\hat{\imath},\hat{\jmath})= (i_k\, j_k)$

Of course, the key feature of this algorithm is the modification of the search window at each step, represented by the function ϕ. We also notice that, if $K$ iterations are performed, the number of criterion computations is

$$C' = \sum_{k=1}^{K} card(W_k) \ . \tag{45}$$

### 6.1.3. *2D-logarithmic search*

In the 2D-logarithimic search (Jain and Jain, 1981), we start with a rough search grid, and we refine it when the current estimated vector is the center of the current search area.

For this case, we need to define the following sets of indexes:

$$\Gamma(1) = \{-1,0,1\}^2 \ , \tag{46}$$

$$\Gamma(2^j) = \{(0,0),(\pm 2^j,0),(0,\pm 2^j)\}, \forall j > 0 \ . \tag{47}$$

We start with a given ($i_0 j_0$), for example the null vector. We select the initial search area $W_1 = \Gamma(2^m)$. Typically, $m$=3 or 4. The value $r_k$=$2^m$ is the current *radius* of the search area. The general step of the algorithm is as follows. If the radius is not 1, the matching criterion is computed for the five candidates of the current window. If the best ($i_k$,$j_k$) candidate is the central one, we halve the search radius and set

$$W_{k+1} = \left\{ (i_k, j_k) + (i,j) \mid (i,j) \in \Gamma\left(\frac{r^k}{2}\right) \right\} \ . \tag{48}$$

$W_{k+1} = \left\{(i_k, j_k) + (i,j) | (i,j) \in \Gamma(\frac{r^k}{2})\right\}$.Otherwise we just center the search window on the best candidate, and keep the same radius for the search area:

$$W_{k+1} = \{(i_k, j_k) + (i,j) \mid (i,j) \in \Gamma(r^k)\} \ . \tag{49}$$

The search patterns are shown in Figure 9.

Finally, if the search radius is equal to 1, the matching criterion is computed on the nine candidates of the current window, which is formed by the previous best candidate and its eight neighbors.

One of the main drawbacks of this algorithm is that the number of iterations is variable and therefore, the complexity of the whole procedure is not perfectly managed.

### 6.1.4. *Conjugate directions search*

The basic idea of this method is to perform one-dimensional searches in each iteration. We start by a searching window of the form

$$W_{k+1} = \{(i_k, j_k) + (i,j) \mid (i,j) \in \Delta W_k\} \ , \tag{50}$$

$W_{k+1} = \{(i_k, j_k) + (i,j) | (i,j) \in \Delta W_k\}$with

$$\Delta W_k = \{(-1,0),(0,0),(1,0)\} \ . \tag{51}$$

This is performed for the horizontal search, and we continue in this direction as long as the current best vector is different from the center of the window and we do not reach a picture boundary. When one of these two cases arises, then we switch to the vertical search, and we set the searching window to

$$\Delta W_k = \{(0,-1),(0,0),(0,1)\} \ . \tag{52}$$

When again a zero motion vector is found, either the estimation is stopped, or a third step can be envisaged, involving a search in the direction given by the original estimation point and the previous estimation. Note however that this step is optional and it can be skipped if a limited-complexity algorithm is targeted.

A similar algorithm is the cross-search (Ghanbari, 1990), where the search is performed separately in each dimension. In this case however, a full exploration of each direction is performed. In other words, we have only two steps, with

$$\Delta W_1 = \{-A, -A+1, \ldots, A\} \times \{0\} , \tag{53}$$

and

$$\Delta W_2 = \{0\} \times \{-A, -A+1, \ldots, A\} . \tag{54}$$

The cross search algorithm requires the computation of the criterion 2*n*-1 times instead of $n^2$ of the full search.

### 6.1.5. *Three-steps search*

As the name shows, the aim of this algorithm is to obtain uniform complexity, corresponding to 3 iterations, regardless of the motion activity (Koga, Iinuma, Hirano *et al.*, 1981). This can be achieved, of course, only at the expense of a certain loss in the precision of the motion vector.

The three-steps search (TSS) algorithm can be seen as a variant of the 2D-logarithmic search with two differences:

1) The search window is always composed by nine elements:
2) $W_{k+1} = \{(i_k, j_k) + (i, j) \mid (i, j) \in \Delta W_k\}$,

   with

   $\Delta W_k = \{-2^k, 0, 2^k\}^2$ .
3) The search radius is halved at each step independently from the estimated vector.

Therefore, if at the first iteration $\Delta W_1 = \{-4,0,4\}^2$ $\Delta W_1 = \{-4,0,4\}^2$, the algorithm will always stop after 3 steps. The corresponding search patterns are shown in Figure 10. The TSS algorithm allows finding displacements of ±7 pixels in both directions with only 25 computations of the criterion (nine for the first step and eight for both the second and the third, since the value of *J* for the center of the search window has been computed at the previous step). For comparison, the full search would have required 225 computations, but would also have provided the best vector.

Some variations of the TSS have been proposed in literature, such as the new TSS (Li, Zeng and Liou, 1994) which is biased towards center points of the search windows and allows an early termination of the algorithm, or the four-steps search (Po and Ma, 1996), which reduces the complexity by checking only a subset of $\Delta W_k$ .

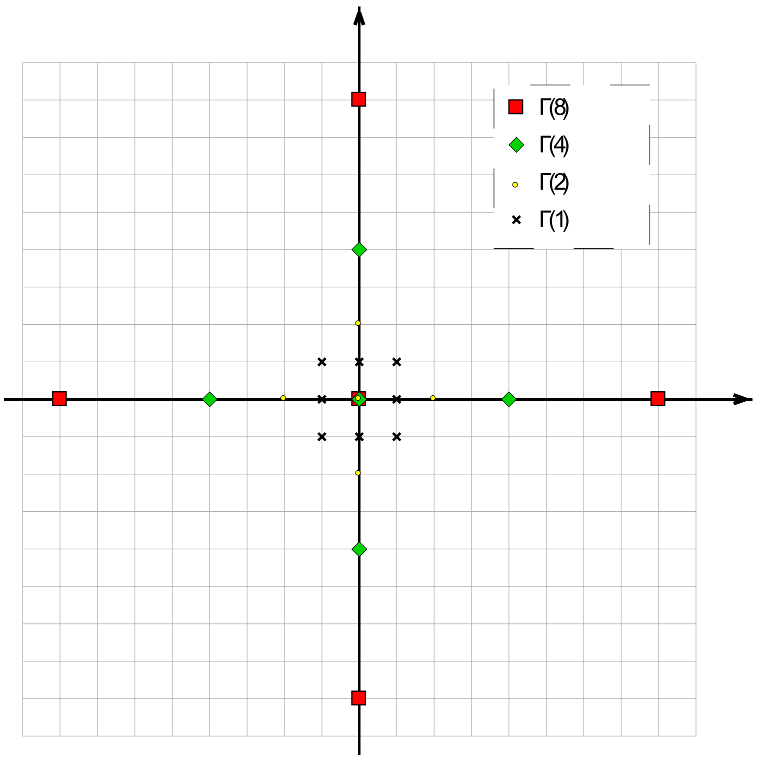

**Figure 9 - 2D-Log: Search patterns at different steps.**

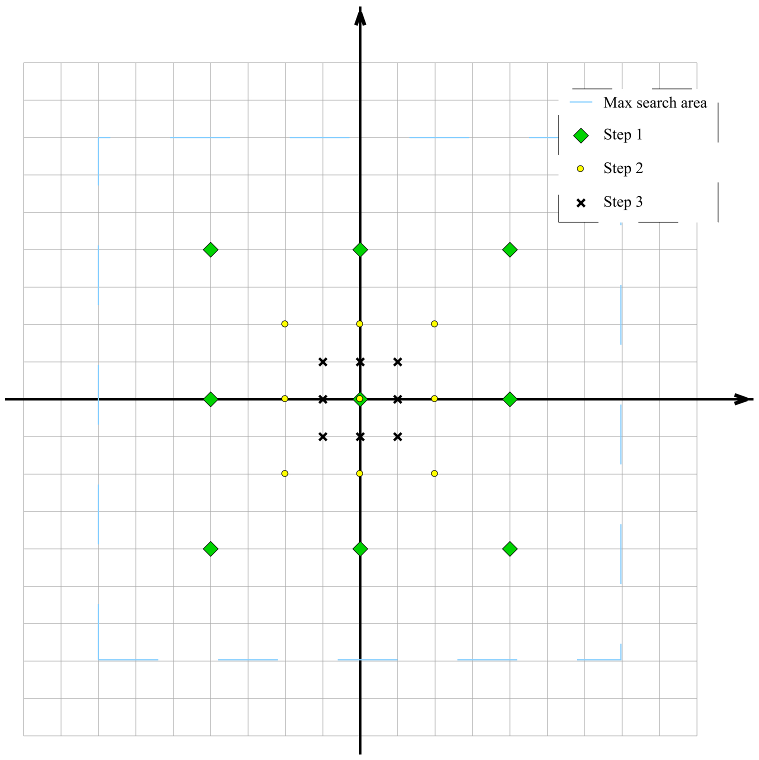

**Figure 10 - Three-steps search: search patterns.**

### 6.1.6. *Diamond search*

The TSS and its variations used to be quite popular until the years 2000's, when a new generation of fast block matching algorithms where developed. These algorithms are not limited to a 15 x 15 search area but still are very fast and effective, and therefore are very commonly used in practical applications such as video encoders.

The diamond search (DS) (Zhu, S.; Ma, 2000) algorithms employs two search patterns as shown in Figure 11. At the first step, a large pattern with a diamond shape is selected (LDSP in Figure 11). According to fact that the best point is the center or not, the next search pattern is respectively changed into a small diamond shape partner (SDSP) or is still a LDSP, as shown in Figure 12. Moreover, only a subset of the pattern points has to be checked at any new step, since the new pattern is always partially superposed to the old one. The algorithm stops when the best point of the SDSP has been found. In other words, in the DS, the LDSP moves in the reference image until the best position is the center of the pattern; then a last iteration of the algorithm is performed using the SDSP.

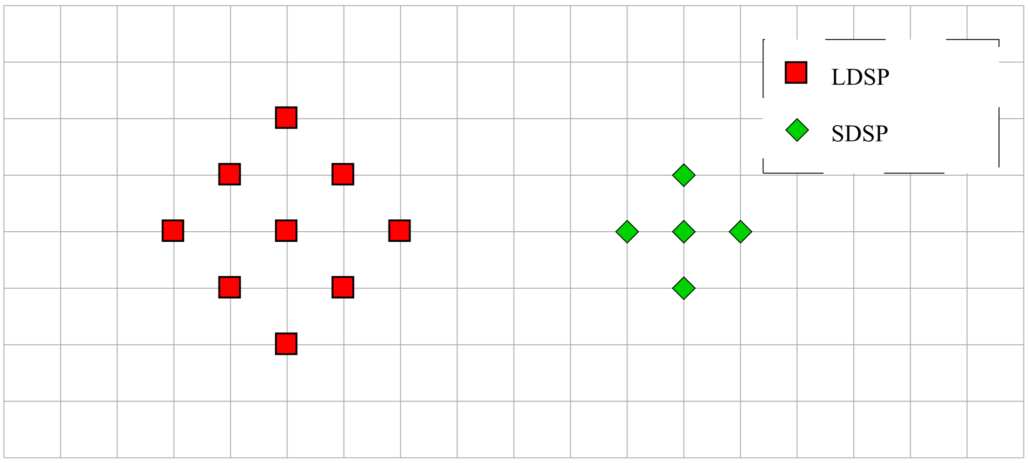

**Figure 11 - Large (LDPS) and small (SPDS) diamond search pattern.**

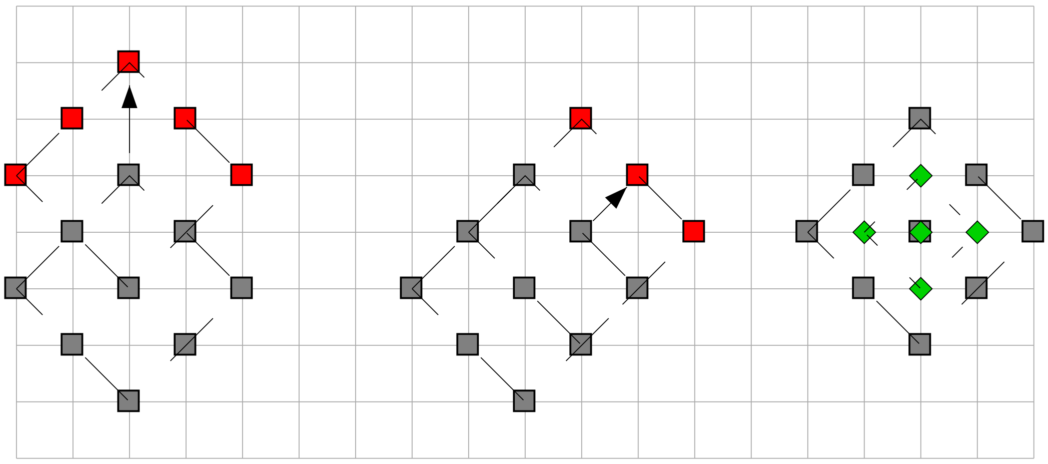

**Figure 12 - The pattern for the next step is different according to the fact that a corner point (left) a side point (center) or the central point (right) are selected.**

The DS algorithm has very good performances since it has not a too small search pattern (which could trap the search algorithm into local minima) neither a too large one, such as the first pattern of the TSS, which could mislead the search path to a wrong direction. Moreover, the search area is not limited since the search pattern keeps moving until the central position of the LDSP is hit.

Simulation experiments show that DS largely outperforms TSS in terms of motion estimation quality, achieving performances similar to NTSS but with a complexity reduction of more than 20%. Thanks to the effectiveness of this technique, it has been integrated in the reference software of the MPEG-4 video coding standard.

### 6.1.7. *Hexagon search*

The DS algorithm has proven that pattern shapes other than a square can lead to fast motion estimation algorithms. However, as one can see from Figure 12, the LDSP moves with a speed of 2 pixels per iteration along the vertical and horizontal directions, while it moves at only $\sqrt{2}$ pixel per iteration in the diagonal directions. This means that the algorithm could require too many iterations to find a diagonal motion vector. Therefore, a hexagonal search algorithm (HS) has been proposed (Zhu, Lin and Chau, 2002), with the patterns shown in Figure 13. The patterns move according to the scheme shown in Figure 14. As in the DS, when the central point of the large pattern is chosen, the algorithm performs a last iteration using the small pattern. The hexagonal search has two advantages with respect to the DS: first, whichever is the pattern motion, only three new points are tested per iteration; second, the pattern is more isotropic than the one of DS, and so diagonal direction are not penalized with respect to the horizontal one. As a consequence, the HS achieve the same motion estimation precision as DS with a complexity reduction close to 40%. The HS and some variations of it are integrated into the reference software of the H.264/MPEG-4 AVC standard (Wiegand, Sullivan, Bjontegaard et al., 2003).

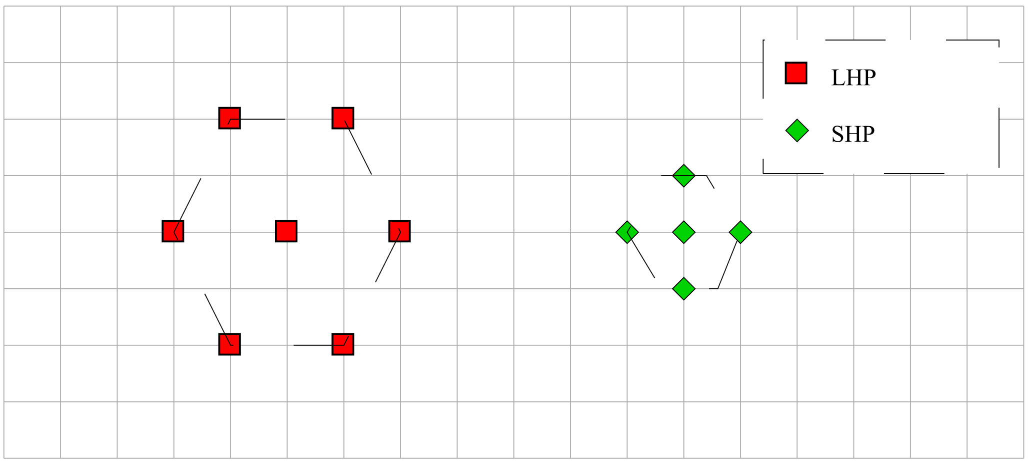

**Figure 13 - Large (LHP) and small (SHP) hexagonal patterns.**

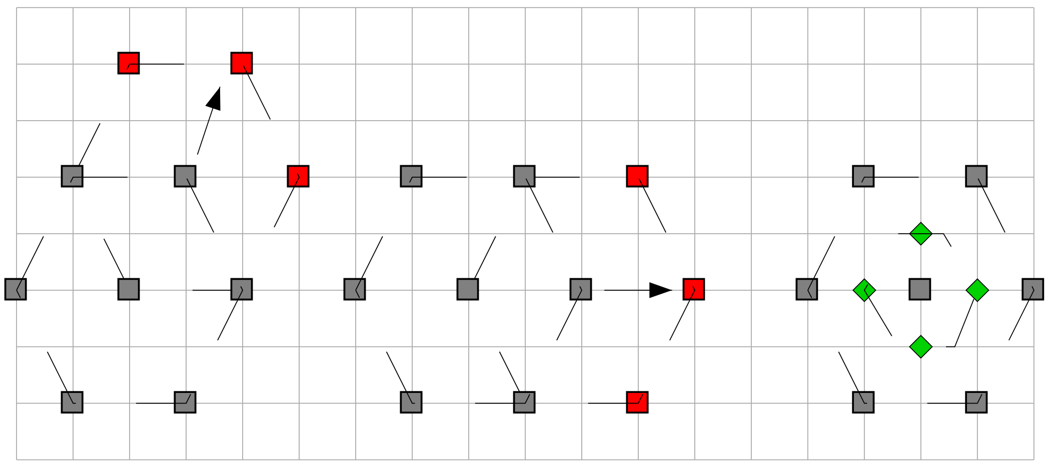

**Figure 14 - The movement of the hexagonal pattern: the speed is two pixels per iteration in the horizontal direction and $\sqrt{5} \approx 2.236$ pixels per iteration in the diagonal directions.**

## 6.2. Matching criterion

In this section we discuss some of the most popular matching criteria for block-based motion estimation.

### 6.2.1. *Norm-based criteria*

As shown in Eqs. (42) or (43), the blocks in the current and in the reference image are matched according to a suitable metric measuring their dissimilarity:

$$J(i,j) = d\left[f_k\left(B_{p,q}\right), f_h\left(B_{p-i,q-j}\right)\right] \ . \tag{55}$$

The most natural approach is to use some distance between the vectors $f_k(B_{p,q})$ and $f_h(B_{p-i,q-j})$, i.e. to compute the $p$-norm of their difference:

$$J(i,j) = \left\| f_k\left(B_{p,q}\right) - f_h\left(B_{p-i,q-j}\right)\right\|_p^p \ . \tag{56}$$

Note that we raise the $p$-norm to the power $p$ in order to get rid of the irrelevant $p$ -order square root. If we set $p$=2, the block matching is performed minimizing the Euclidean distance between the vectors of image samples. In formulas, we have:

$$J_{SSD}(i,j) = \sum_{(n,m)\in B_{p,q}} \left[f(n,m,k) - f(n-i,m-j,h)\right]^2 \ . \tag{57}$$

This matching criterion is referred to as the Sum of Squared Differences (SSD). Note that the using the SSD is equivalent to use the mean square error (MSE) between the current image and the compensated reference. The SSD is very popular, above all for compression applications and this for several reasons. First, the Euclidean distance is a very intuitive and easy to understand metric. Second, by minimizing the SSD, we minimize the mean square error between the current block $f_k(B_{p,q})$ and the block $f_h(B_{p-i,q-j})$, that is used as prediction. Hence, this allows an efficient residual coding. In fact, the Motion Compensated (MC) prediction of the current image, i.e. a prediction of $f_k$ where $f_k(B_{p,q})$ is replaced by $f_h(B_{p-i,q-j})$, is often used to measure the quality of the motion estimation. By definition, no other criterion can provide a better result in this respect (for a given block size and search area). However, the SSD has some drawbacks. First, it is relatively complex, since it requires *PQ* multiplications to be computed (one per each pixel of the block). Second, if some pixels in the image are affected by noise, the square power tends to enhance the associated error, preventing to find the correct motion vector. Third, it does not take into account possible global illumination variations from one image to the other.

In order to alleviate the first two problems, one can resort to the sum of absolute differences (SAD), defined as follows:

$$J_{SAD}(i,j) = \sum_{(n,m)\in B_{p,q}} \left| f(n,m,k) - f(n-i,m-j,h) \right| \quad . \tag{58}$$

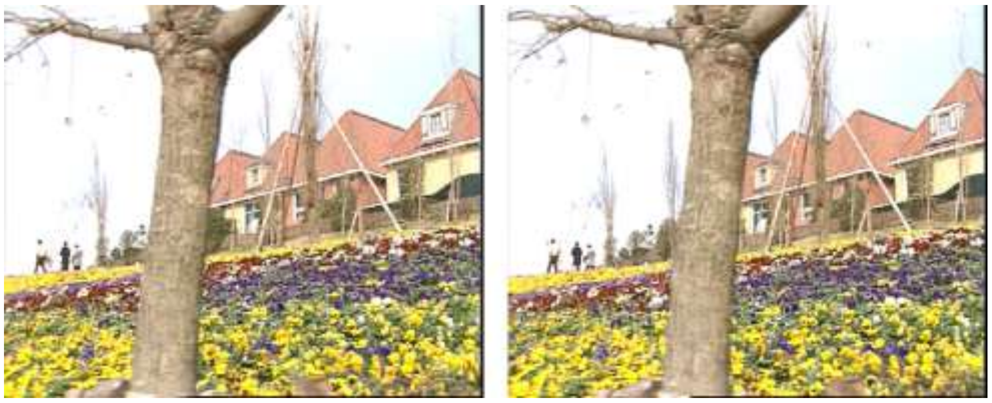

**Figure 15 - Two images (223 and 227 respectively) from the test sequence *flower and garden*.**

The SAD is equivalent to the mean absolute error (MAE). In order to compare the SSD and SAD, we show in Figure 15 two images from a video sequence and in Figure 16 the motion vector fields obtained using block matching with the SSD (left) and the SAD (right) criteria. The two criteria have very close qualities: the MSE of the compensated image achieved by minimizing the SSD is only 0.16 dB smaller than the one obtained using SAD. We also observe that the two fields capture the global lateral motion of the sequence and the different apparent velocity of the foreground (the tree) and the background. However both methods fail at correctly estimating the movement of very homogenous areas (such as the sky in the left part of the image). In particular SSD seems to produce more outliers such as the vectors in red.

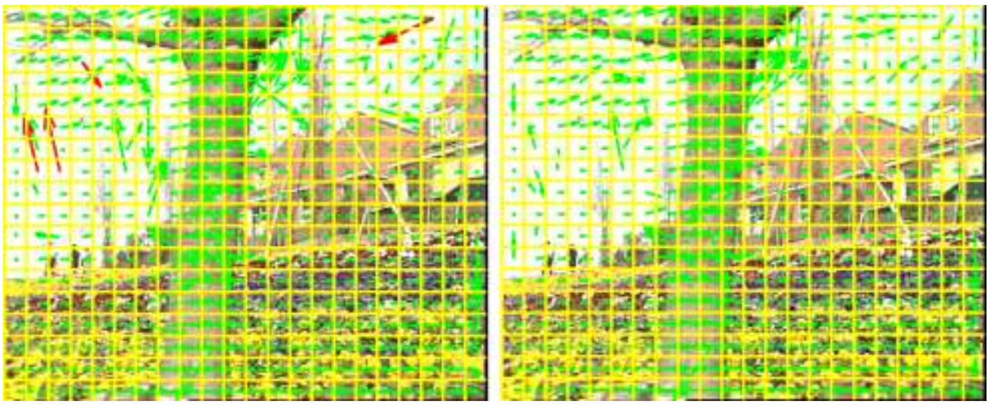

**Figure 16 - Estimated motion vector fields. Left: SSD criterion; right: SAD criterion.**

This lack of regularity affects both the capability of the ME to represent the real motion and the compression performances that can be achieved: an irregular MVF is more expensive to encode than a regular one. For example, the MVF estimated by SSD has an estimated coding cost of 2143 bits, while the slightly more regular field produced by SAD costs 2103 bits.

We can improve the regularity of the MVF by explicitly introducing a smoothness constraint in the criterion:

$$J_{REG}(i,j) = \left\| f_k\left(B_{p,q}\right) - f_h\left(B_{p-i,q-j}\right) \right\|_p^p + \lambda R(i,j) \ , \qquad (59)$$

where *R*(*i,j*) is a suitable cost function and λ is a positive constant. As a consequence, the vector that has the smallest SSD or SAD is not selected if it is too irregular according to *R*. For example *R* could be the norm of the difference between (*i,j*) and a representative of its neighborhood: this would allow having a vector much different from its neighbors only when this is related to a new object (involving a large SSD or SAD reduction). In video compression applications, *R*(*i,j*) is the coding cost of the vector (*i,j*), as we shall describe in more details in Sec. 9.

An example of regularized MVF is shown in Figure 17. We used the criterion in Eq. (59), with *p=2*. The *R* function is the norm of the difference between *(i,j)* and the median vector of its causal neighborhood (three vectors) . The resulting MVF is visually more regular than those shown in Figure 16. This is confirmed by the estimated coding cost, which shrinks to 2008 bits, while the MC prediction quality is reduced by less than 0.1 dB.

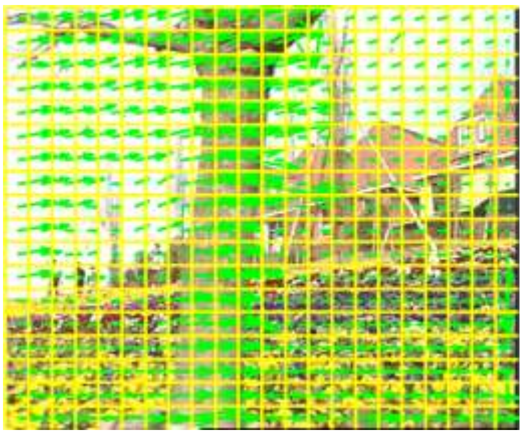

**Figure 17 - Estimated MVF using a regularized SSD criterion.**

### 6.2.2. *An illumination-invariant criterion*

Block-based matching algorithms are mainly used in compression; however, given their simplicity, they have been investigated in the context of other applications, where the objective is to find a MVF as close as possible to the actual motion. However, in these cases SSD and SAD show another limit: they cannot deal with global illumination changes. In this case, we can resort to a variant of SSD which is robust to affine luminance transformations: the Zero-mean Normalized SSD (ZN-SSD). If we refer to the h^th^ element of vector $f_k(B_{p,q})$ as $f_k(B_{p,q})$[h], we can introduce the zero-mean version of vector $f_k(B_{p,q})$ component by component:

$$\tilde{f}_k\left(B_{p,q}\right)[l] = f_k\left(B_{p,q}\right)[l] - \frac{1}{PQ}\sum_h f_k\left(B_{p,q}\right)[h] \ . \tag{60}$$

We define then the ZN-SSD:

$$J_{ZN-SSD}(i,j) = \frac{\sum_{l=1}^{PQ}\left[\tilde{f}_k\left(B_{p,q}\right)[l] - \tilde{f}_h\left(B_{p-i,q-j}\right)[l]\right]^2}{\left(\sum_{l=1}^{PQ}\tilde{f}_k^2\left(B_{p,q}\right)[l]\cdot\sum_{l=1}^{PQ}\tilde{f}_h^2\left(B_{p-i,q-j}\right)[l]\right)^{1/2}} \ . \tag{61}$$

The main disadvantage of ZN-SSD is its computational complexity: for each candidate vector we need to perform about 3*PQ* multiplications.

### 6.2.3. *Correlation-based criteria*

It is known that the cross correlation (i.e. the scalar product) between two vectors is a measure of their similarity, and therefore one might think about using correlation-based criteria to perform block-matching. This is also motivated by the fact that fast algorithms exist to compute the correlation in the frequency domain. However, as we show in the following, the cross correlation itself has some major drawbacks and cannot reliably be used as matching criterion, while the normalized cross-correlation can effectively perform this task.

Let us start by the relationship between norm-based and correlation-based criteria. The SSD between $f_k(B_{p,q})$ and $f_h(B_{p-i,q-j})$ can be written as:

$$\begin{aligned} J_{SSD}(i,j) &= \left\| f_k\left(B_{p,q}\right) - f_h\left(B_{p-i,q-j}\right) \right\|^2 \\ &= \left\langle f_k\left(B_{p,q}\right) - f_h\left(B_{p-i,q-j}\right), f_k\left(B_{p,q}\right) - f_h\left(B_{p-i,q-j}\right) \right\rangle \\ &= \left\| f_k\left(B_{p,q}\right) \right\|^2 - 2\left\langle f_k\left(B_{p,q}\right), f_h\left(B_{p-i,q-j}\right) \right\rangle + \left\| f_h\left(B_{p-i,q-j}\right) \right\|^2 \end{aligned} \tag{62}$$

.

In this expression, $\|f_k(B_{p,q})\|^2$ does not depend on $(i,j)$. If the norm of the displaced block does not vary very much with $(i,j)$, the minimizing the SSD criterion is almost equivalent to maximizing the correlation between the blocks:

$$\begin{aligned} J_{CORR}(i,j) &= \left\langle f_k\left(B_{p,q}\right), f_h\left(B_{p-i,q-j}\right) \right\rangle \\ &= \sum_{(n,m)\in B_{p,q}} f(n,m,k) f(n-i,m-j,h) \end{aligned} . \tag{63}$$

The main advantage in using this criterion is that fast FFT-based implementations exist for the correlation calculation. On the other hand, the cross correlation is not very reliable for the following reasons:

1) In natural images, the block-wise energy $\|f_h(B_{p-i,q-j})\|^2$ actually varies with the position. Therefore the correlation of the original block with the real displaced block can be less than the correlation with a very bright spot in the reference image;
2) The correlation coefficient is sensitive to global amplitude changes, such as those caused by changing global illumination conditions

Therefore, a better criterion is the normalized correlation coefficient, defined as:

$$J_{N-CORR}(i,j) = \frac{\sum_{l=1}^{PQ} \left[ \tilde{f}_k\left(B_{p,q}\right)[l] \, \tilde{f}_h\left(B_{p-i,q-j}\right)[l] \right]}{\left( \sum_{l=1}^{PQ} \tilde{f}_k^2\left(B_{p,q}\right)[l] \cdot \sum_{l=1}^{PQ} \tilde{f}_h^2\left(B_{p-i,q-j}\right)[l] \right)^{1/2}} . \tag{64}$$

We observe that Eq. (64) is very similar to Eq. (61): the only difference is that the zero-mean cross correlation has replaced the zero-mean SSD at the numerator.

Removing the local mean and normalizing allows to mitigate the impact of the two abovementioned problems of cross correlation.

The normalized cross-correlation is typically used for feature tracking in an image sequence: in this case we look for the position of a single feature, which is no longer constrained to have a rectangular support. Therefore, Eq. (64) is simply modified, considering the vector of luminance values of the template instead of $f_k$ and a vector with the same shape (but displaced by $(i,j)$) instead of $f_h$.

The main reason for using normalized cross correlation instead of SSD is that, when the number of pixels of the feature is much smaller than the number of pixels of the image, frequency-domain implementations of Eq. (64) are very efficient in term of computational complexity. They are mainly based on FFT for the computation of the numerator and the computation of a cumulated sum over the image for the denominator.

### 6.2.4. *A criterion for wavelet-based compression*

Norm based criteria, such as SAD, SSD and their regularized version proved to be very effective for predictive video compression, since in this framework motion estimation and compensation are used to minimize the energy of the prediction error. It is interesting to notice that, when the temporal correlation of a video signal is not removed by predictive coding but by transform coding, as in the case of motion-compensated wavelet video coding, these criteria are no longer necessarily optimal. In facts, instead of minimizing the prediction error energy, in this case one should maximize the transform coding gain, defined as the ratio of geometric and arithmetic mean of the temporal wavelet subbands. These subbands are obtained by wavelet filtering along the motion direction, which in turns is computed by motion estimation. Therefore, the motion estimation should find out the trajectory maximizing the coding gain. This can be quite difficult in the general case, but for a class of simple yet effective temporal filters it has been found that optimal motion estimation should be performed on three consecutive frames, evaluating jointly the backward and forward motion vectors (Cagnazzo, Castaldo, André *et al.*, 2005). If $\varepsilon_B$ [$\varepsilon_F$] is the backward [forward] motion compensated error, instead of separately computing the MVFs that minimize $\|\varepsilon_B\|^2$ and $\|\varepsilon_F\|^2$, it has been proved that the optimal MVF are those that jointly minimize $\|\varepsilon_B\|^2 + \|\varepsilon_F\|^2 + \langle\varepsilon_B,\varepsilon_F\rangle$. It is interesting to notice that this criterion involves the computation of norms and correlations.

## 6.3. Sub-pixel accuracy

The sub-pixel accuracy refers to displacements that do not correspond to integer pixel positions on the original grid of the images. The need to estimate finer displacements can be understood from the simple example in Figure 18. The horizontal dimension on the image plane is orthogonal to the drawing plane. We assume that the motion vector from image *k* to *k+2* in the position *(n,m)*, referred to as *v(n,m),* is estimated as (0,-3), i.e. three pixels downward. Assuming a uniform motion in the corresponding time interval, one would infer a displacement of 1.5 pixels downward from frame *k* to frame *k+1*.

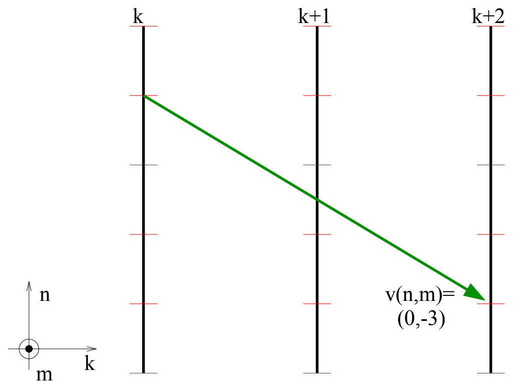

**Figure 18. A schematic representation of the sub-pixel motion.**

The estimation of motion vectors with fractional pixel accuracy is possible and it is especially implemented in block matching algorithms. We describe below the method in the case of half-pixel accuracy, but it is easy to generalize it to other precisions (1/3, 1/4, 1/8 and so on). For a better understanding, the method is illustrated in Figure 19.

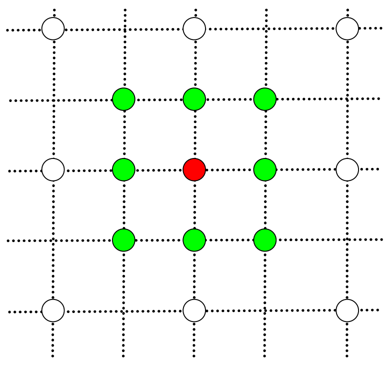

**Figure 19 - Half-pel accuracy motion estimation: the red position on the integer grid is the currently estimated motion vector, the white pixels are its neighbours on the integer grid, the eight green positions on the half-pel accuracy grid have to be tested.**

Once the best fitting for integer pixel accuracy has been found (the red pixel in Figure 18), one has to consider all the candidates at fractional positions *(i,j)* (in green) and to test the same criterion *J(i,j)* (MAE or MSE). The vector minimizing the criterion *J(i,j)* among these positions is the new motion vector, having fractional values. Of course, the difficulty consists in the fact that for testing the matching criterion, we do not dispose of image values at half-pel positions. They have to be interpolated from existing values. A common method used for this operation is the bilinear interpolation, but longer filters can be used for the interpolation.

Let us briefly remind the principle of the bilinear interpolation, with reference to Figure 20.

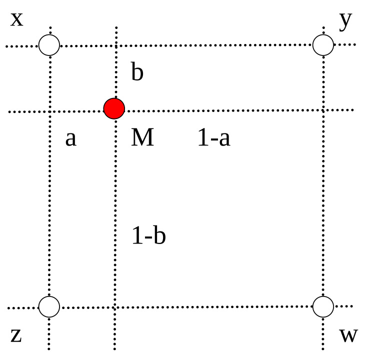

**Figure 20 - Bilinear interpolation: white pixels are on the integer grid, red value has to be interpolated.**

If *x, y, z, w* are the intensity values of the neighbors situated on the integer grid (consider the distance between them equal to 1) and the point to be interpolated has the distance *a* to the left-hand side neighbors and *b* to the upper neighbors, then the value *M* obtained with bilinear interpolation will be:

$$M = (1-a)(1-b)x + a(1-b)y + b(1-a)z + abw \ . \tag{65}$$

In particular, for half-pel accuracy, *a=b=1/2* and then *M = (x+y+z+w)/4*.

The concept of generalized interpolation is introduced in (Lakshman, Schwarz, Blu *et al.*, 2011). A combination of short Infinite Impulse Response (IIR) and Finite Impulse Response (FIR) filters is used, which provides greater design flexibility and better coding performance. An hardware-friendly multiplication-less implementation is also described.

Sub-pixel motion estimation and compensation has been integrated in early video coding standards, and has been improved since then. In MPEG-1 and MPEG-2 a simple bilinear interpolation allowed half-pixel motion precision. In MPEG-4 Part 2 an eight-tap filter is used to compute the interpolated samples.

In H.264/AVC (Wiegand, Sullivan, Bjontegaard *et al.*, 2003), a quarter-pixel accuracy is allowed for motion vectors. A six-taps filter is used to generate half-pixel samples. These values are then rounded to integer values, on which a bilinear filter is performed to produce quarter-pixel samples.

HEVC (Ohm and Sullivan, 2013) further improves the interpolation filters used for fractional ME. An eight-tap filter is used to generate half-pixel samples and a seven-taps one is used for quarter-pixel samples. HEVC takes benefit both from longer filters and from not having intermediate rounding operations.

## 6.4. Variable-size block matching

One severe limitation of block matching motion estimation algorithms is that a single translational motion vector is assigned to all pixels within a block. However, the underlying assumption that the whole block is undergoing a uniform translational motion does not hold in the case of complex scenes with fast moving objects. In particular, it leads to a poor prediction along moving edges which results in block artifacts in the MC frame and decreased compression performance.

To alleviate this important drawback, variable-size block matching techniques, also referred to as multi-grid, have been proposed (Chan, Yu and Constantinides, 1990), (Dufaux and Moscheni, 1995). These techniques are based on the observation that large blocks are sufficient in uniform areas, whereas finer blocks are necessary in highly textured regions or near moving object edges. Straightforwardly, in variable-size block matching techniques, the size of blocks is adapted based on local texture and motion characteristics.

In practice, variable-size block matching techniques essentially proceed as follows. Block matching is first performed on a coarse grid. Then, selected blocks are split. For instance, this decision can be based on a simple threshold on the SAD or SSD of the block, to identify blocks where motion estimation has failed. Each of the new sub-blocks is assigned the motion vector of the parent block, and block matching is then performed again on these sub-blocks. Note that a smaller search window is most often used at this stage, as an initial estimate of the motion vector is already available. The process is iterated until a minimum block size is reached. At each step of the iterative process, selected blocks are most commonly divided into two or four sub-blocks, resulting in a binary- or quad-tree representation. This segmentation information can thus be efficiently represented. An example of the grid resulting from this process is shown in Figure 21. Clearly, large blocks are used in uniform areas, whereas small blocks have been preferred in detailed areas.

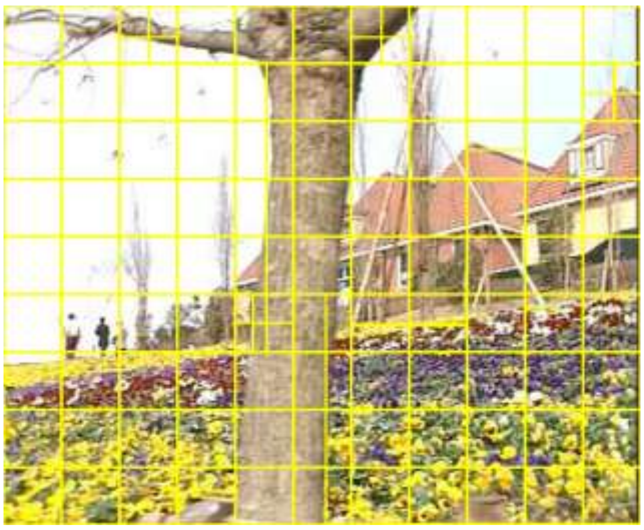

**Figure 21 - Example of final grid using variable-size block matching.**

The projection of the motion vector obtained at a coarse grid to a finer one should have two objectives. Firstly, the projection operator should avoid the propagation into finer levels of erroneous motion vector estimates due to large block sizes. Secondly, it should guarantee the smoothness of the motion vector field. Simply duplicating the parent vector may not be optimal. A bilinear interpolation is also possible. It typically leads to smoother motion fields; however it does not ensure motion consistency along moving object boundaries. A better approach is to select the best initial condition among motion vectors in a neighborhood, as proposed in (Dufaux and Moscheni, 1995).

Regarding the splitting criterion, in the context of video coding, a more appropriate and sophisticate algorithm is to optimize the decision in a rate-distortion sense. More precisely, in this way, the gain of a more accurate motion precision, leading to a reduced residual signal, is weighted against the cost of extra motion vectors to be transmitted. See Sec. 9.1.3 for a more detailed discussion on rate-distortion optimization.

A simple form of variable-size block matching is supported in H.264/AVC (Wiegand, Sullivan, Bjontegaard and Luthra, 2003) and HEVC (Ohm and Sullivan, 2013), as described in more details in Sec. 9.1.

Efficient hardware implementations of variable-size block matching have been proposed. In (Yap and McCanny, 2004), a one-dimensional very large-scale integration architecture is introduced for full-search variable-size block matching. The SAD of a larger block is efficiently computed by re-using the results previously obtained for smaller sub-blocks. In (Chen, Chien, Huang *et al.*, 2006), the impact of variable-size block matching in hardware architectures is first analyzed, and two new hardware architectures are then proposed.

Finally, variable-size or multi-grid block matching techniques can efficiently be combined with multi-resolution approaches, as described in Sec. 7.2. in order to further improve performance.

# 7 Parametric motion estimation

In this section, we consider motion estimation approaches which estimate the parameters of the motion models as defined in Sec. 2.3.2, and more specifically in Eqs. (10) to (16). As previously discussed, these models can be applied to a coherently moving region of support.

An important special case is when a single region of support corresponding to the whole image is selected. In this case, referred to as global motion estimation, the dominant motion is estimated. This dominant motion is resulting from camera motion, such as dolly, track, boom, pan, tilt and roll, which is a widely used cinematic technique in filmmaking and video production.

Hereafter, we describe two classes of techniques for parametric motion estimation. We also discuss difficulties arising due to outliers, and related robust estimators.

## 7.1. Indirect parametric motion estimation

A first class of approaches indirectly computes the motion parameters from a dense motion field rather than from the image pixels. More specifically, a dense motion field is first estimated, and then the parametric motion model is fitted on the obtained motion vectors.

A Least Mean Square (LMS) technique is commonly used for this model fitting. More specifically, the motion parameters are derived from the expressions

$$\begin{aligned} &\min_{\pi} \sum_{(x,y)\in\Re} \left[u(x,y) - \hat{u}_{\pi}(x,y)\right]^2 \\ &\min_{\pi} \sum_{(x,y)\in\Re} \left[v(x,y) - \hat{v}_{\pi}(x,y)\right]^2 \end{aligned}, \qquad (66)$$

where *u*(*x*,*y*) and *v*(*x*,*y*) denote the horizontal and vertical components of the dense motion field, $\hat{u}_{\pi}(x,y)$ and $\hat{v}_{\pi}(x,y)$ the corresponding fully parameterized motion field, $\pi = \{\pi_1, \pi_2, ..., \pi_n\}$ the set of parameters of the model (see Sec. 2.3.2), and $\Re$ the support region for model fitting. The model parameters $\pi$ can then be computed by setting to zero the partial derivatives of Eq. (66) according to $\pi_1$, $\pi_2$, ..., $\pi_n$.

In (Adiv, 1985) and (Wang and Adelson, 1994), the initial dense motion field is estimated using a gradient-based optical flow approach (see Sec. 3). An LMS technique is then used to compute the model parameters. The methods in (Pardas, Salembier and Gonzalez, 1994) and (Tse and Baker, 1991) are similar, however, a block matching technique (see Sec. 6) is rather used in the first step.

A drawback of these approaches is that the performance is significantly influenced by the accuracy of the initial dense motion field. Indeed, LMS is very sensitive to erroneous samples, which may negatively impact the model parameters estimation. Another weakness is that the region of support is assumed to be characterized by a coherent motion which can be closely represented by the motion model. However, this strong assumption may not always hold. To alleviate these two drawbacks, robust estimation can be used, as further discussed in Sec. 7.3.

The same framework can also be used to estimate a parametric motion model in the compressed domain. In this case, block-based motion vectors are readily available from the compressed code stream. Such an approach is proposed in (Smolic, Hoeynck and Ohm, 2000) for a low complexity global motion estimation. To take into account the high likelihood of outliers, a robust M-estimator is used. A similar compressed domain scheme is proposed in (Tok, Glantz, Arvanitidou *et al.*, 2010), relying on the Helmholtz tradeoff estimator as a robust estimator.

## 7.2. Direct parametric motion estimation

A second class of approaches directly computes the model parameters.

Based on the optical flow equation (see Sec. 2.3.1), a gradient-based formulation similar to the dense optical flow approaches is followed. However, as discussed in Sec. 3, the optical flow equation is underconstrained. Thus, an additional constraint is required, namely a smoothness constraint (Horn-Schunck) or a local uniformity constraint (Lucas-Kanade). When including a motion model, the problem becomes implicitly constrained.

More specifically, a parametric gradient-based formulation of the optimization criterion is given by

$$\sum_{(x,y)\in\Re}\left[\hat{u}_\pi(x,y)\frac{\partial f}{\partial x}(x,y)+\hat{v}_\pi(x,y)\frac{\partial f}{\partial y}(x,y)+\frac{\partial f}{\partial t}(x,y)\right]^2 , \qquad (67)$$

where $\hat{u}_\pi(x,y)$ and $\hat{v}_\pi(x,y)$ are the fully parameterized motion field, $\pi=\{\pi_1, \pi_2, ..., \pi_n\}$ the set of parameters of the model (see Sec. 2.3.2), and $\Re$ the support region. By taking the partial derivatives of Eq. (67) according to the parameters $\pi_1, \pi_2, ..., \pi_n$, and setting to zero, the parameters of the motion model can then be derived.

The above formulation uses the optical flow equation, which is based on a first order Taylor series expansion (see Sec. 2.3.1), and is adopted in (Anandan, Bergen, Hanna *et al.*, 1993). However, the first order expansion implicitly assumes that the velocity remains small. As an alternative, a second order Taylor series expansion is preferred in (Hoetter and Thoma, 1988) and (Wu, S.F.; Kittler, 1990). In order to improve robustness, especially when estimating the gradient which is prone to noise, hierarchical schemes are used. More specifically, the process is iterated on a multi-resolution representation by means of a Gauss-Newton minimization algorithm.

A different formulation is proposed in (Szeliski, and Coughlan, 1994) and (Dufaux and Konrad, 2000), where the Sum of Squared Difference between the current frame and the motion compensated previous frame is directly minimized:

$$\sum_{(x,y)\in\Re}\left[e(x,y,t)\right]^2 \ , \tag{68}$$

with

$$e(x,y,t) = f(x-\hat{u}_\pi(x,y), y-\hat{v}_\pi(x,y), t-1) - f(x,y,t) \ . \tag{69}$$

The motion parameters π={ π₁, π₂, ..., πₙ} are computed by minimizing Eq. (68) using the Levenberg-Marquardt iterative non-linear minimization (Press, Flannery, Teukolsky *et al.*, 1988). More specifically, the following expression is obtained

$$\sum_{l=1}^{n} H_{k,l}\delta\pi_l = b_k \ , \tag{70}$$

where $\delta\pi_l$ is the parameter update term

$$\delta\pi_l = \pi_l^{(i+1)} - \pi_l^{(i)} \ , \tag{71}$$

the curvature matrix $H_{k,l}$ (equal to one-half the Hessian matrix) is defined as

$$H_{k,l} = \frac{1}{2}\sum_{(x,y)\in\Re}\frac{\partial^2 e^2}{\partial\pi_k\partial\pi_l} \cong \sum_{(x,y)\in\Re}\frac{\partial e}{\partial\pi_k}\frac{\partial e}{\partial\pi_l} \ , \tag{72}$$

and

$$b_k = \frac{1}{2}\sum_{(x,y)\in\Re}\frac{\partial e^2}{\partial\pi_k} = -\sum_{(x,y)\in\Re} e\frac{\partial e}{\partial\pi_k} \ . \tag{73}$$

The Eq. (70) is solved by Singular Value Decomposition (SVD) (Press, Flannery, Teukolsky *et al.*, 1988) and the process is iterated in a multi-resolution data representation. In (Dufaux and Konrad, 2000), an initial matching step is included in order to find a good initial condition. Moreover, a truncated quadratic function is used in order to increase robustness to outliers.

As an alternative to the above gradient-based approaches, a generalized matching technique is proposed in (Moscheni, Dufaux and Kunt, 1995). More specifically, the model parameters are computed by minimizing a dissimilarity measure. The technique is robust, as it does not rely on a model of the luminance. However, it entails a large computational complexity.

## 7.3. Robust estimation

Outliers are samples that markedly deviate from the prevailing tendency. In the case of parametric motion estimation, the presence of outliers, due to noisy measurements or poorly defined support regions, will lead to inaccurate model estimates. In the case of global motion estimation, foreground moving objects also correspond to outliers.

In order to alleviate the impact of outliers, robust estimation has been proposed (Rosseeuw and Leroy, 1987), (Meer, Mintz, Rosenfeld *et al.*, 1991). One indicator of the performance of a robust estimator is its breakdown point, roughly defined as the highest percentage of outliers that the robust estimator can tolerate.

Three classes of robust estimators can be defined:

- M-estimators: M-estimators are a generalization of maximum likelihood estimators. They involve the minimization of a function of the form:

  $$\sum_i \rho(r_i) \ , \tag{74}$$

  where $r_i$ is the residual error between a data sample and its fitted value, and $\rho$ is a symmetric positive-definite function with a unique minimum at $\rho(x=0)$. With a squared error function, $\rho(x)=x^2$, the rate of increase accelerates for large values of $x$, giving a very large weight to these values. Conversely, for a robust estimator, the function $\rho$ saturates at large values of $x$.

- L-estimators: L-estimators are linear combination of order statistics. Two examples are the median and the α-trimmed-mean.

- R-estimators: R-estimators are based on rank tests.

Robust estimators have been successfully used in motion estimation. Two robust estimators, the Least Median of Squares (LMedS) and the Least-Trimmed Squares (LTS), are used in (Ayer, Schroeter and Bigün, 1994). Tukey's biweight function (Press, Flannery, Teukolsky *et al.*, 1988) is applied in an iterative re-weighting scheme in (Odobez and Bouthemy, 1995) and (Smolic, Hoeynck and Ohm, 2000). A truncated quadratic function is minimized in (Dufaux and Konrad, 2000). Finally, the Helmholtz tradeoff estimator is applied in (Tok, Glantz, Arvanitidou *et al.*, 2010).

# 8 Multi-resolution approaches

Multi-resolution or multi-scale approaches are often used in image and video processing. They have their origin in the Laplacian pyramid introduced in (Burt and Adelson, 1983).

A low-pass pyramid is a set of images with progressively more smoothing and reduced spatial resolution. A smoothing filter is first applied on the original image. This smoothed image is then subsampled, most commonly by a factor of two in both the horizontal and vertical directions (in this case, it is referred to as a dyadic structure). The same filtering and subsampling operations are then applied again on the resulting image, in a recursive way. Different smoothing kernels can be used. The concept of low-pass pyramid is illustrated in Figure 22.

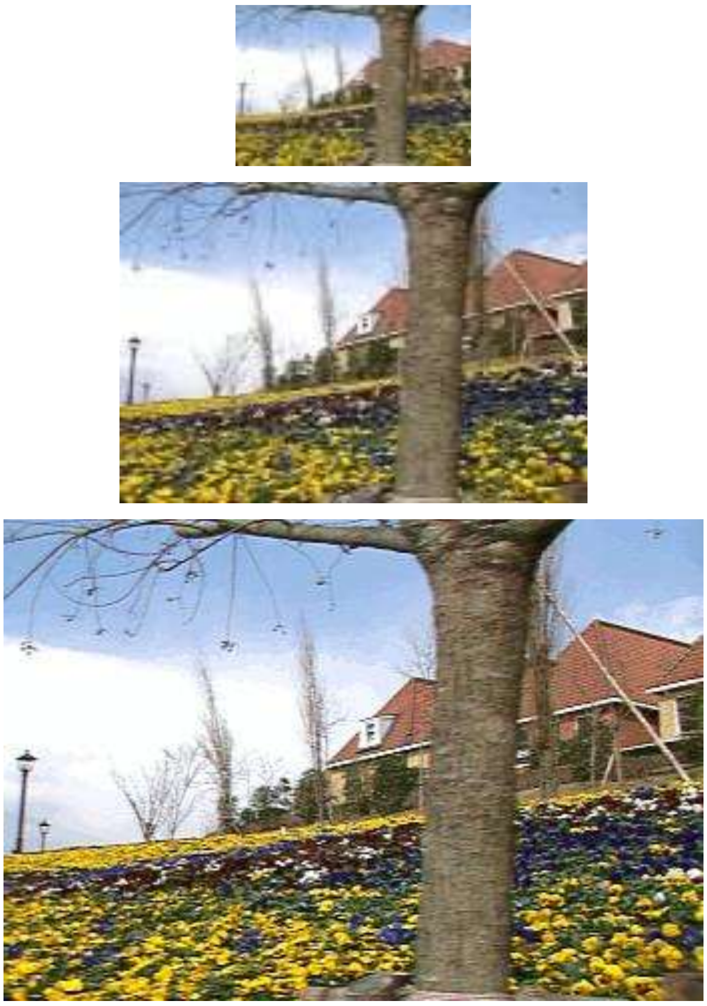

**Figure 22 - Example of low-pass pyramid (3 levels).**

A multi-resolution and multi-scale representation is commonly used, as illustrated at the top of Figure 23. However, a dual representation at multi-resolution but a single scale is also possible, as shown at the bottom of Figure 23. These dual representations are equivalent, as the second one can be derived from the first one by upsampling and interpolation, and vice versa, the first one can be obtained from the second one by filtering and downsampling.

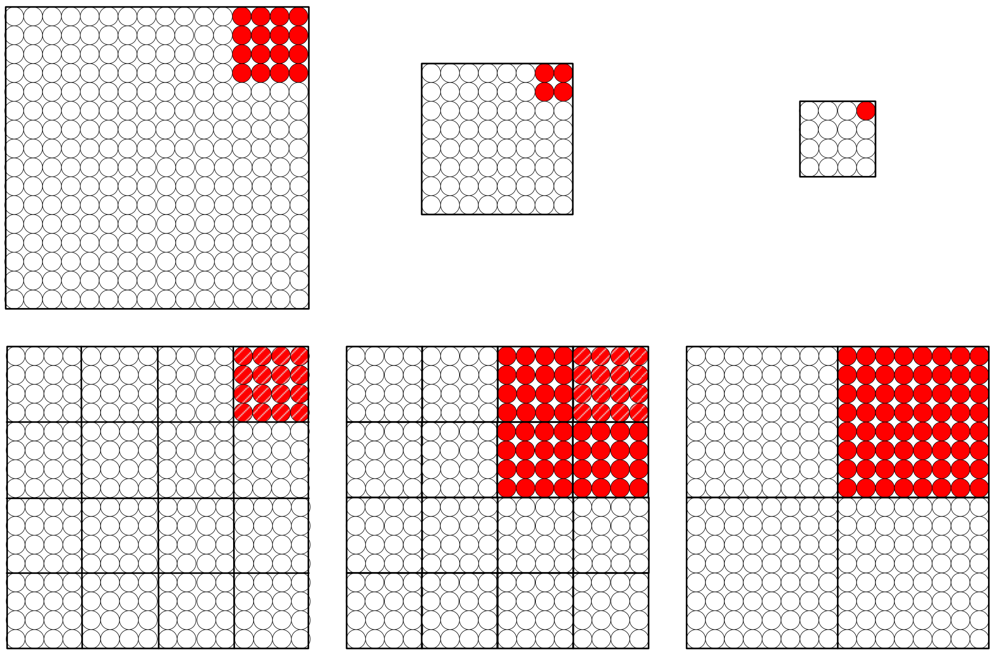

**Figure 23 - Duality between multiple scales (top) and single scale (bottom) for multi-resolution representation.**

In the context of motion estimation, such a multi-resolution or multi-scale representation is very appealing. Thank to the smoothing and spatial subsampling, coarse resolution levels allow to efficiently and robustly estimate large scale motions. Conversely, fine local motions can accurately be estimated at finer resolution levels. An additional advantage of multi-resolution motion estimation techniques is the potentially significant reduction in computational complexity.

Thanks to these appealing advantages, multi-resolution has been widely adopted for motion estimation. Multi-resolution optical flow methods have been proposed in (Burt, Yen and Xu, 1983), (Glazer, 1984) and (Enkelmann, 1988). Similarly, multi-resolution block matching techniques have been introduced in (Anandan, 1987), (Bieling, 1988) and (Dufaux and Moscheni, 1995). In turn, multi-resolution parametric techniques have been proposed in (Black and Anandan, 1996) and (Dufaux and Konrad, 2000).

Most algorithms follow a coarse-to-fine processing. More specifically, motion is first estimated at the coarsest resolution level. As coarse resolution input images are obtained by low-pass filtering and subsampling, noise is largely smoothed out and large-range interactions can be efficiently taken into account. Hence, a robust estimation is obtained, which captures the large trends in motion. The motion field is then projected to the next finer resolution level and iteratively refined. This refinement allows to consider short-range relations and hence to identify local motions and improve accuracy. As a result, a more reliable motion vector field is obtained, with coherent displacements from one scale to the next.

Multi-resolution block matching motion estimation, as proposed in (Anandan, 1987) and (Bieling, 1988), is illustrated in Figure 24 and described in more details. The motivation is to overcome the limitations of block matching techniques, and in particular to reduce blocking artifacts in the motion compensated frame. Block matching is first estimated on a low-pass and subsampled version of the original images, either by full-search or using some fast search techniques (see Sec. 6.1). Thanks to the spatial subsampling, large displacements can efficiently be estimated with a small search window. These motion vectors are then projected on the finer resolution level. If the same block size is used at both resolution levels, a parent block at the coarser level will be projected into four children blocks at the finer level. A simple projection operator is to have these four children blocks inherit the same motion vector of the parent block. Straightforwardly, its amplitude has to be multiplied by two in each direction to take into account the difference of scale. Note that it is also possible to select the best initial condition among motion vectors in a neighborhood, as discussed for variable-block size block matching in Sec. 6.4. The motion vectors are then refined at the finer scale. At this stage, a reduced-size search window around the current estimate can be used in order to guarantee smoothly varying motion vector fields.

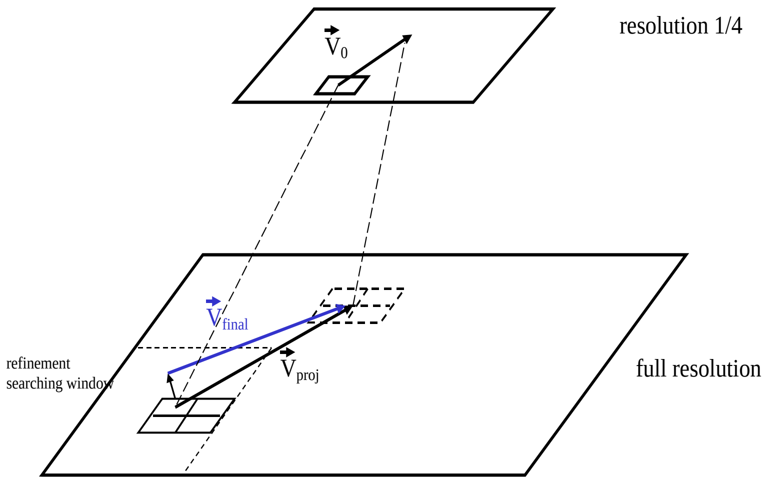

**Figure 24 - Hierarchical motion estimation with refinement at high resolution.**

While most proposed multi-resolution approaches follow the coarse-to-fine processing described above, some techniques have also been proposed which includes fine-to-coarse steps. More specifically, in (Enkelmann, 1988), a strategy is described which returns to coarser resolution levels when the current estimate is unreliable. In (Dufaux, Moccagatta, Rouchouze *et al.*, 1993), a multigrid block matching technique is described, which combines coarse-to-fine and fine-to-coarse steps in order to avoid local minima. Finally, an optical flow multigrid algorithm which passes flow estimates both up and down the multi-resolution levels is proposed in (Bruhn, Weickert, Kohlberger *et al.*, 2006).

It can be straightforwardly observed that a multi-resolution block matching technique, combined with a spatially adaptive grid size, is essentially equivalent to the variable-size block matching described in Sec. 6.4.

In summary, multi-resolution or multi-scale motion estimation approaches are very appealing, as they result in more robust and accurate motion vector fields. Moreover, this performance gain is obtained with a reduced computational complexity. However, a drawback of many approaches is that the obtained motion field is overly smooth and sometimes fails to accurately represent detailed structures and small moving objects.

# 9 Motion compensation

Motion compensation (MC) is used together with motion estimation (ME) to perform temporal predictions in the context of video compression. This approach is so effective that virtually all compression standards resort to it in order to remove the temporal redundancy from video.

Predictive coding consists in computing a prediction of the input signal, and in compressing the prediction error (or residual) instead of the signal itself. If the signal is sufficiently correlated and the prediction is computed in such a way that it could be perfectly reproduced at the decoder side, this predictive coding is more effective than coding the original signal.

Video sequences show a very high temporal correlation, since consecutive images are very similar, and mainly differ for the movement. Therefore a motion compensated temporal prediction is generally very effective, *i.e.* it produces a low-energy residual. Frames in which blocks can be temporally predicted are called *P-frames*.

In video compression, the coding unit is typically a block of pixels. The current terminology for ME/MC is *macroblock*, since the term *block* was reserved for the unit of the spatial transform. However in the upcoming standard HEVC the terminology is clarified since the terms *coding unit*, *prediction unit* and *transform unit* are explicitly introduced. We give some details about HEVC ME/MC in Sec. 9.1.4, and until then we will keep using the terms block and macroblock.

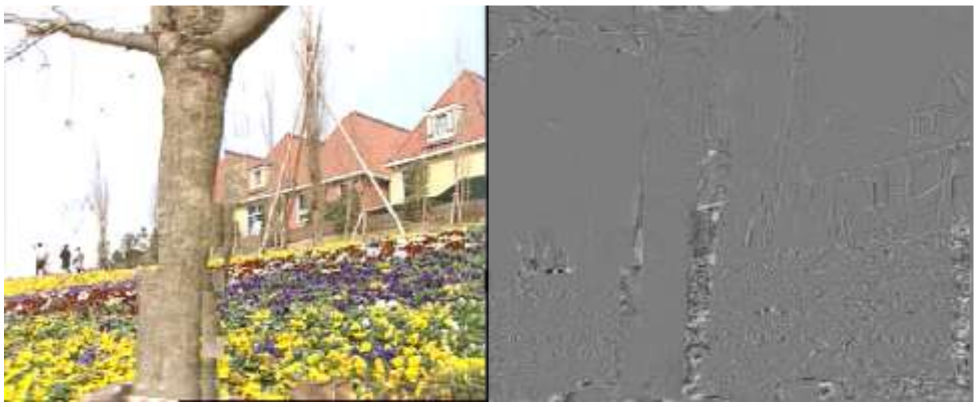

**Figure 25 - Motion compensated version of the reference image and the associated prediction error.**

The motion compensation consists simply in using the block from the reference image in position $(p-\hat{\imath}, q-\hat{\jmath})$ as prediction of the block of the current image in position (*p*,*q*), where $(\hat{\imath}, \hat{\jmath})$ is the vector estimated for the position (*p*,*q*). Therefore, instead of encoding the luminance values $f_k(B_{p,q})$, we have to encode $f_k(B_{p,q})$ - $f_h(B_{p-\hat{\imath},q-\hat{\jmath}})$. In Figure 25 (left) we show the motion compensated prediction of image 227 from the *flower* sequence (Figure 15, right), produced by using the motion vector field shown in Figure 16 (left) on the reference image (image 223, Figure 15, left).

We observe that the prediction is good almost everywhere, except for the blocks that where *disoccluded* in the current image (*i.e.* appeared from behind the tree), and for those that entered the scene from the right. In order to mitigate this kind of problem, *bi-directional* motion-compensated prediction has been introduced since the first video coding standards as MPEG-1: the block belonging to some images (called B-frames) can be predicted not only with blocks from a previous image, but also with blocks from a successive one. Moreover, the prediction can also consist in the average of the two blocks.

We also remark that even with B frames, nothing assures that a good prediction exist in the reference frames; for this reason video standards do not oblige the encoder to use the motion compensated prediction: sometimes is more effective to encode the new block instead of a large prediction error. This choice is up to the encoder.

## 9.1. Motion compensation in H.264/AVC

In H.264/AVC (Wiegand, Sullivan, Bjontegaard *et al.*, 2003) the motion compensation is very effective thanks to several new tools. The most relevant are described in the following. In particular H.264/AVC uses variable block sizes for ME/MC, multiple references and quarter pixel ME precision.

### 9.1.1. *Macroblocks and partitions*

The coding unit in H.264/AVC is the macroblock (MB), *i.e.* a square block of 16 x 16 luminance values, plus the co-located chrominance values. For the sake of simplicity, in the following we will consider only the luminance.

A MB can be coded in several different ways, called *modes*. Motion compensation is used in temporal predictive modes, which differ in the *partition* of the MB. In the 16 x 16 mode, a single motion vector is estimated for the whole MB, and the prediction is the corresponding block of pixels in the reference frame. In the other modes, the MB is divided into rectangular or square parts, and for each one a different motion vector can be encoded. Therefore the predictor of the current MB can be formed by joining blocks from different regions of the reference frame. This allows dealing successfully with blocks where several objects are present. In particular, the MB can be divided into two identical horizontal or vertical rectangles, or into four 8 x 8 blocks. In this case, each of the blocks can undergo a further partition, as shown in the bottom row of Figure 26. In conclusion, for a single temporal predictive MB H.264/AVC allows to encode from one to sixteen motion vectors. In general one can expect that a finer partition gives a better predictor, but this comes at the cost of a higher coding rate for motion.

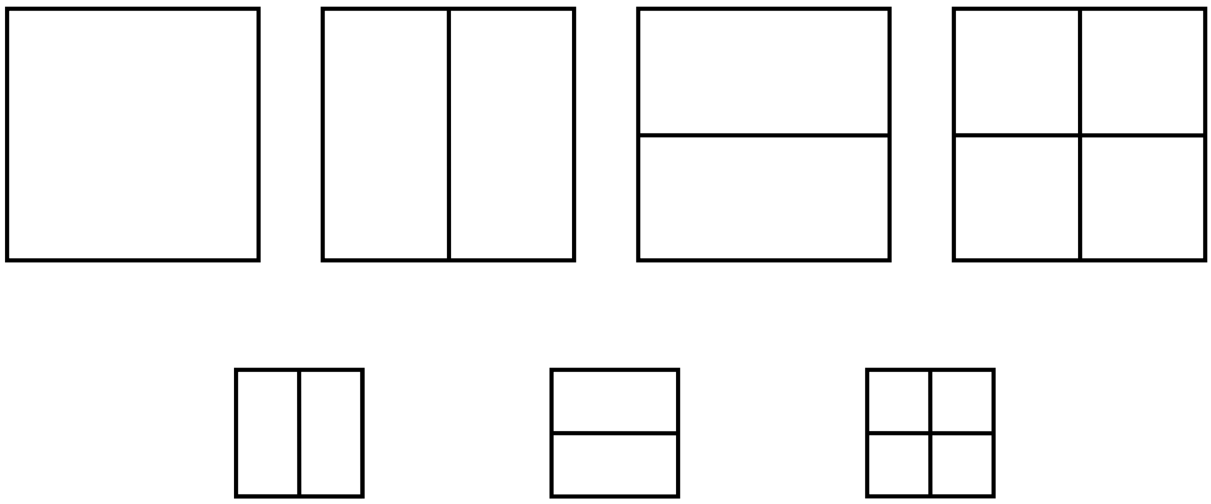

**Figure 26 - H.264/AVC macroblock partitions. First row, from left to right: 16 x 16, 16 x 8, 8 x 16, 8 x 8. If this last partition is selected, each 8 x 8 block can be further partitioned into 8 x 4, 4 x 8 or 4 x 4 sub-blocks.**

### 9.1.2. *Multiple references and generalized P/B frames*

As in previous standards, H.264/AVC allows for mono-directional and bi-directional prediction. However, in H.264/AVC the motion compensation is much more flexible than in the past.

In the case of mono-directional prediction, there is a list of up to 16 images: a MB can be predicted using any reference image in the list: therefore the encoder should write in the encoded bitstream not only the selected motion vector, but also the reference index in the list. Therefore, two macroblocks in the same image can be predicted using blocks that belong to different references, as shown in Figure 27. Likewise, in the case of bi-directional prediction, two lists are kept. A MB can be predicted with blocks from any image in any list, or as a linear combination of a block from the first list and a block from the second one.

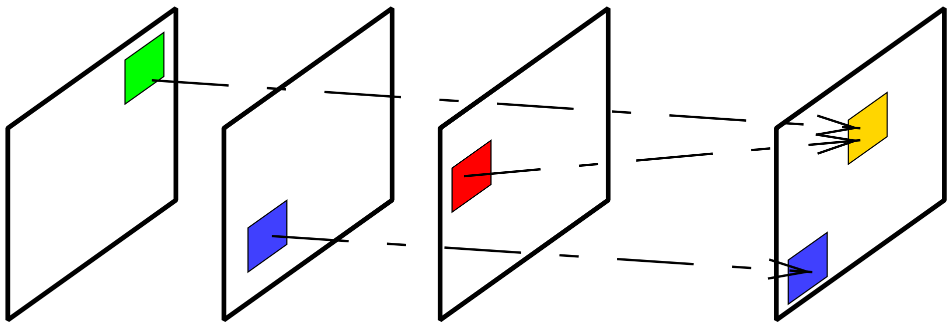

**Figure 27 - Multiple references in H.264/AVC.**

### 9.1.3. *Rate-constraint Lagrangian motion estimation*

In this section we describe a non-normative tool for efficient motion estimation in H.264/AVC, referred to as rate-distortion optimization (RDO) (Wiegand, Schwarz, Joch *et al.*, 2003).

In order to achieve the best possible RD performances, each single decision of the encoder should be taken considering the effect on the final image quality and coding rate, and then solving a constrained optimization problem: typically, the problem is the one of minimizing the distortion with a constraint on the rate. This problem can be solved with a Lagrangian approach. More precisely, given a candidate vector ($i$,$j$), we could perform a complete encoding/decoding process, evaluating the resulting distortion and the coding rate. More precisely, we could compute the motion-compensated residual, then perform the spatial transform, the quantization, the inverse transform and we could add again the prediction, obtaining the decoded macroblock associated to the candidate vector. The resulting distortion would be used as $D(i,j)$. At the same time, we could compute the coding rate as the sum of the rates needed to encode the motion vector, the reference image index if multiple references are possible, and the quantized residual. This would give *R*(*i*,*j*). Finally, the best vector would be the one minimizing

$$J_{RDO}(i,j) = D(i,j) + \lambda R(i,j) \ . \qquad (75)$$

However, this approach is unfeasible in practice, since it would impose an extremely high complexity: each MB is encoded with each possible candidate vector. In practice, a good approximation of Eq. (75) is the following:

$$J_{RDO-ME}(i,j) = J_{SAD}(i,j) + \lambda_{ME} R_{Motion}(i,j) \ . \qquad (76)$$

In other words we use a regularized version of the SAD criterion of Eq. (58), where the regularization function is the rate needed to encode the motion vector.

Finally, let us spend a few words about how the encoder can perform an RD-optimized selection of the MB partition. Since the number of partitions is relatively limited, in this case the encoder can actually use the criterion defined by Eq. (75). For each partition, and for each sub-block of the partition, the contributions to the total distortion and rate are computed and summed up. The mode providing the smallest total cost can be selected as the optimal encoding mode.

### 9.1.4. *Preview of forthcoming HEVC*

In the forthcoming High Efficiency Video Coding standard (Sullivan, Ohm, Han *et al*., 2012), (Ohm and Sullivan, 2013), ME/MC with variable block size is further enhanced with respect to H.264/AVC. More precisely, blocks sizes range from 64 x 64 to 4 x 4. The temporal prediction is performed within a so called prediction unit that in turn can be split into one, two or four prediction blocks (PBs). As in H.264/AVC, a block can be split into two rectangular blocks, but four new split are possible in HEVC, in which the two rectangular blocks do not have the same number of pixels: these splits are known as asymmetrical mode partitions. The possible splits of a block into PBs are shown in Figure 28. The asymmetrical partitions have size *M* x *M*/4 or *M* x 3*M*/4.

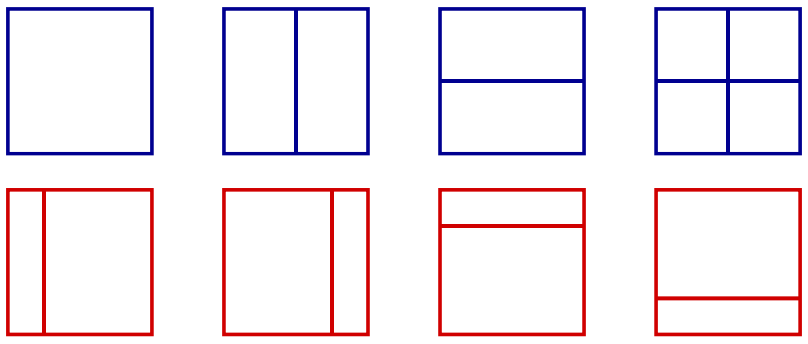

**Figure 28 - HEVC partition types for ME/MC. Top row: symmetrical partitions. Bottom row: asymmetrical partitions.**

## 9.2. Overlapped Block Motion Compensation

When one applies motion compensation on an image using a motion vector field produced with block matching, the resulting image can be affected by blocking artifacts. Motion compensation consist in copying blocks from disparate locations in the reference image and in putting them side-by-side: of course, nothing assures a smooth transition between them. Therefore unnatural image luminance variations appear in correspondence of the block grid, giving rise to annoying visual artifacts, as shown for example in Figure 25 (left).

In order to overcome the block artifacts in the motion-compensated frame, Overlapped Block Motion Compensation (OBMC) has been proposed (Nogaki and Ohta, 1992). This method simply consists in considering for the computation of the motion prediction at the boundaries of a block not only the contribution of the estimated block, but also that of neighboring blocks, leading to a prediction by a weighted average of these two contributions. A schematic example is shown in Figure 29.

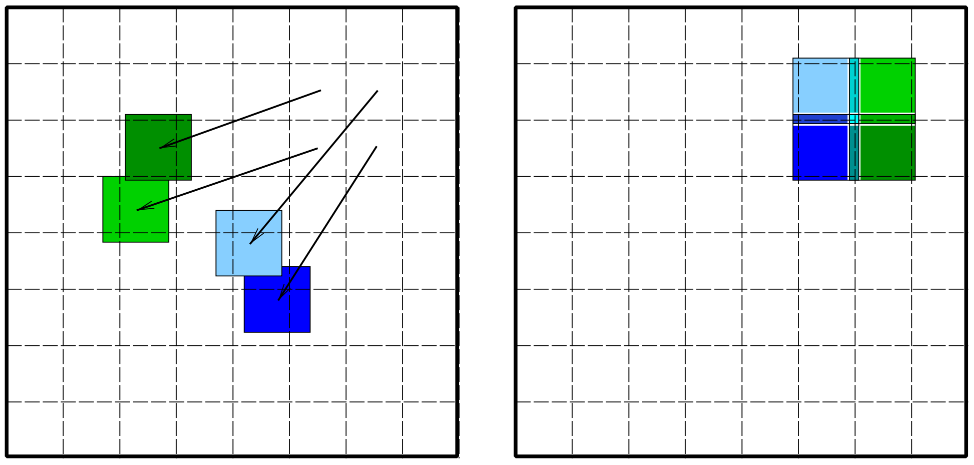

**Figure 29 - Principle of overlapped block motion compensation. Left: we show four motion vectors and the blocks they point toward in the reference image. Right: after motion compensation, the blocks partially overlap, allowing a smooth transition from one block to another.**

An estimation-theoretic analysis of motion compensation is presented in (Orchard and Sullivan, 1994). OBMC is formulated as a probabilistic linear estimator of pixel intensities, which leads to improved prediction.

## 9.3. Global Motion Compensation

Global Motion Compensation (GMC) is especially suited to encode video content with a significant amount of camera motion, such as panning, zooming and tilting. In such a case, the coding efficiency of common Local Motion Compensation (LMC) decreases. On the one hand, a large number of motion vectors have to be transmitted for the moving background. On the other hand, a translational motion model may fail in the presence of camera motion including zoom or rotation. GMC improves upon LMC by building a prediction using global motion parameters. In this way, the dominant motion is accurately represented with very few parameters.

GMC has been standardized in MPEG-4 (Ebrahimi, Dufaux and Nakaya, 2000), more precisely in the Advanced Simple Profile. We now describe this scheme in more detail. The encoder is illustrated in Figure 30. More precisely, it involves the following steps. Both Local Motion Estimation (LME) (i.e. baseline block matching, see Sec. 6) and Global Motion Estimation (GME) (see Sec. 7) are performed. Next, the encoder selects for each MacroBlock (MB) the best prediction between LMC and GMC. Global motion parameters are encoded and transmitted for every frame. In addition, motion vectors are transmitted for every MB encoded using LMC.

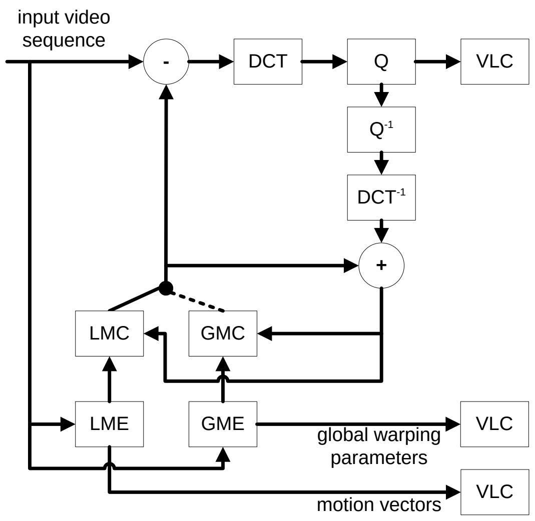

**Figure 30 - MPEG-4 GMC encoder.**

The LMC - GMC decision is not normative. In the MPEG-4 Verification Model (MPEG-4 Video Verification Model, 1998), the following test is used

$$\begin{array}{ll} \text{if } (SAD_{GMC} - P(Q_p, MV_x, MV_y) < SAD_{LMC} & \text{use GMC} \\ \text{otherwise} & \text{use LMC} \end{array}, \quad (77)$$

where $SAD_{GMC}$ (respectively $SAD_{LMC}$) is the sum of absolute difference between the original MB and the GMC prediction (respectively the LMC prediction), $Q_p$ is the quantization parameter, and ($MV_x$, $MV_y$) is the motion vector obtained by LME. The term $P(Q_p, MV_x, MV_y)$ is defined as

$$\begin{aligned} P(Q_p, MV_x, MV_y) = (1-\delta(MV_x, MV_y))(N_B Q_p)/64 \\ +2\delta(MV_x, MV_y)(N_B/2+1) \end{aligned}, \quad (78)$$

with $N_B$ the number of non-transparent pixels in the macroblock (MPEG-4 supports object-based video coding with arbitrary-shape objects), and $\delta(MV_x, MV_y)=1$ when $(MV_x, MV_y)=(0,0)$ and 0 otherwise. The purpose of this term is to give an edge to GMC, especially when $Q_p$ is large (i.e. at low bit rate). It is motivated by the two following observations. Firstly, the gain brought by GMC is in large part due to a reduced amount of overhead motion information since no motion vector is transmitted for MB encoded using the GMC mode. Secondly, the gain resulting from GMC increases at very low bit rates, as in this case the bit rate to transmit motion vectors becomes a larger percentage of the overall bit rate.

The global motion parameters have to be transmitted to the decoder. For this purpose, instead of directly transmitting the parameters of the motion model, displacement of $n$ reference points are encoded, with $n$=1,…,$N$. More precisely, reference points ($i_n$, $j_n$) are positioned at the corners of the current frame (or the bounding box in the case of an arbitrary-shape object), and the corresponding points ($i'_n$, $j'_n$) are computed in the reference frame using the global motion parameters, as shown in Figure 31. Next, the coordinates ($i'_n$, $j'_n$) are quantized to half-pel precision. Finally, the vectors $(u_n, v_n)=(i_n - i'_n, j_n - j'_n)$ are computed and transmitted as differential motion vectors. Four motion models are considered: perspective model where $N$=4 pixels are enough to estimate the model parameters, affine ($N$=3), translation - isotropic magnification - rotation ($N$=2) and translation ($N$=1).

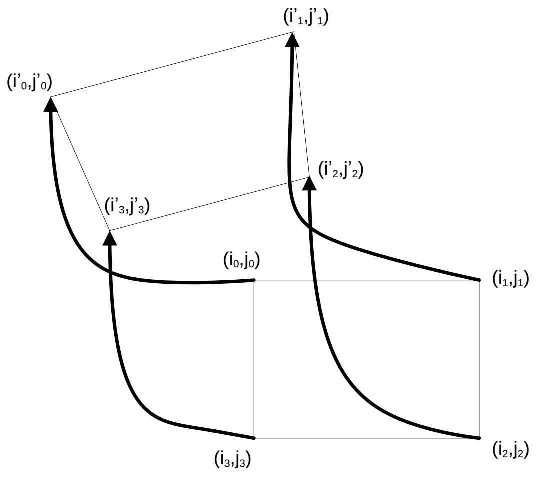

**Figure 31 - Trajectories encoding.**

Similar to the GMC in MPEG-4, a two-stage motion compensation for H.263 was introduced in (Jozawa, Kamikura, Sagata *et al*., 1997). First, GMC is applied to the reference frame, giving a new GMC reference. Next, LMC is performed twice: on the GMC reference and on the regular reference. The best prediction is then selected.

The scheme in (Wiegand, Steinbach and Girod, 2005) combines affine MC with long-term memory MC prediction. Several sets of affine parameters are estimated for sub-regions of the image. Then, for each affine set, the reference picture is warped and added in the long-term memory buffer. Conventional block-based ME and MC is then carried out using all the available reference pictures, and RDO is used for optimal mode selection.In (Glantz, Krutz and Sikora, 2010), a new prediction mode is proposed which combines GMC and temporal filtering of the previously decoded pictures. A rate-distortion optimization is applied on each macroblock to decide whether to use this new prediction mode. The technique is integrated in H.264/AVC, showing improved coding performance.

Exploiting global motion information, an adaptive temporal filter is proposed in (Krutz, Glantz, Tok et al., 2012), as a post-processing for HEVC.

One of the major drawbacks of GMC is an increased computational complexity both at the encoder and decoder sides. The gain achieved straightforwardly depends on the type of motion in the sequence.

## 9.4. Sprites

A sprite, also known as mosaic or panoramic image, refers to a large composite image obtained by aligning and blending pixels from different video frames, see (Teodosio and Bender, 1993), (Irani, Anandan, Hsu, 1995) and (Szeliski, 1996). In the presence of significant camera motion, a sprite can often reconstruct a large panoramic view of the background of the scene by estimating global motion (see Sec. 7). This sprite efficiently captures temporal information, resulting in a very compact representation.

As an alternative to GMC, but with a similar objective, sprite coding as been proposed as an efficient way to represent a video sequence (Irani, Hsu and Anandan, 1995), (Dufaux and Moscheni, 1996), (Lee, Chen, Lin *et al*., 1997). Sprite coding has been standardized in MPEG-4 (Ebrahimi, Dufaux and Nakaya, 2000).

The sprite has typically to be generated off-line, prior to sprite coding. The process is illustrated in Figure 32. Global motion estimation is first performed. Techniques such as those presented in Sec. 7 can be used, usually with a translational, affine or perspective model. Warping (Wolberg, 1990) is then used to align pixels of the current video frame with the background sprite. Finally, the aligned frames are blended and accumulated in the sprite. Note that the sprite is constructed for a region characterized by a coherent motion. This may require a segmentation step, for instance to indentify background and foreground regions.

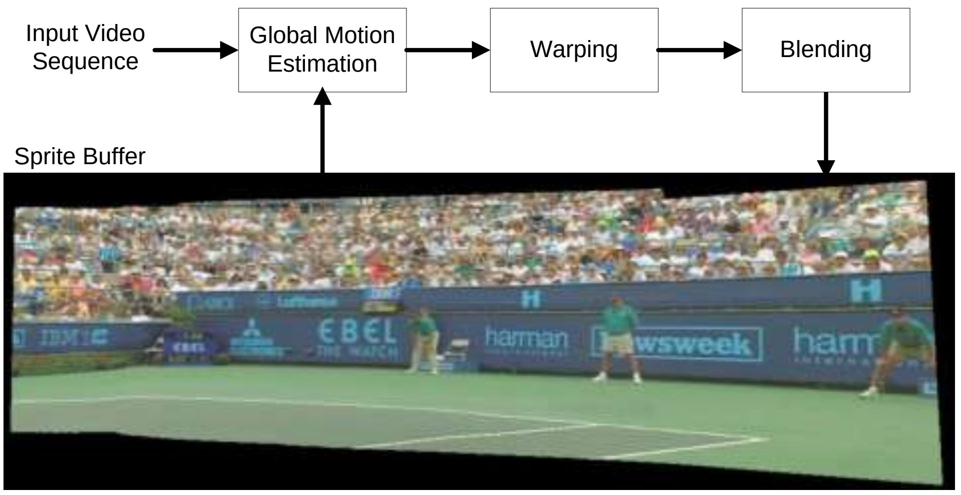

**Figure 32 - Sprite generation process.**

The sprite is in fact a large still image with an associated segmentation mask. In MPEG-4 terminology, it is referred to as a Video Object Plan (VOP) (Ebrahimi, Dufaux and Nakaya, 2000). Its texture and shape can efficiently be encoded using an Intra coding technique. It is then sufficient to transmit the sprite, along with the warping parameters, resulting in a very compact representation. At the decoder side, frames of the video sequence can be reconstructed using the warping parameters and the sprite content. The MPEG-4 sprite encoding and decoding is illustrated in Figure 33 (Ebrahimi, Dufaux and Nakaya, 2000).

Trajectories are encoded in the same way as for the GMC technique in MPEG-4 (see Sec. 9.3).

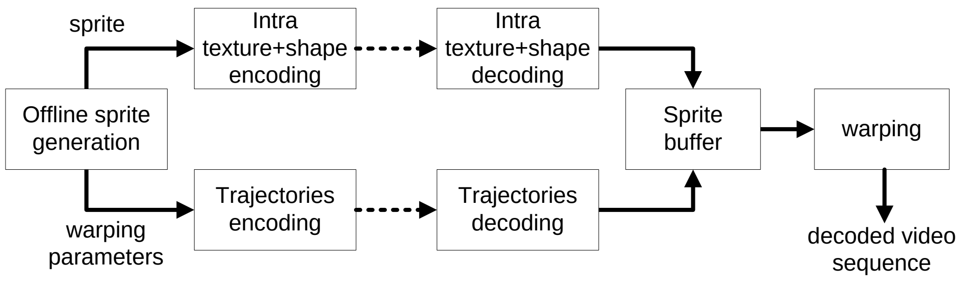

**Figure 33 - Sprite coding in MPEG-4.**

In (Dufaux and Moscheni, 1996), the sprite is dynamically build and used in a motion compensated predictive scheme. In order to improve the sprite generation process, a new blending technique is introduced in (Lu, Gao and Wu, 2003), based on the region reliability. Furthermore, an arbitrary-shape spatial prediction method is proposed to increase the coding performance. A scheme combining sprite coding with automatic background subtraction and H.264/AVC is presented in (Krutz, Glantz, Frater *et al.*, 2011). A rate-distortion optimization technique is also introduced. A long-term global motion estimation technique is proposed in (Smolic, Sikora and Ohm, 1999), using a closed-loop prediction to avoid accumulation of errors, which is especially fitted for sprite coding. An overview of sprite generation and coding is presented in (Farin, Haller, Krutz *et al.*, 2008), along with some recent developments.

Sprite coding can efficiently encode synthetic graphic objects. In the case of natural video, the sprite has to be constructed off-line prior to coding. Therefore, it makes the approach unsuitable for real-time applications. Moreover, sprite coding is only fitting for a video object whose motion can be approximated by a rigid 2-D model. However, this assumption very often holds true for the background.

# 10 Performance assessment criteria for motion estimation algorithms

It is quite difficult to assess the performances of a motion estimation algorithm without including it in a specific application. As we have previously discussed, different properties are desirable when considering image sequence analysis or video coding.

In the context of image sequence analysis, the ability to provide a very accurate motion vector field is primordial, even though the resulting field is dense and more costly to encode. Tests can be set up, implementing artificial motions, in order to check the performance of the algorithm by comparing true and estimated motion vector fields.

When considering video coding, the primary objective is to achieve optimal rate-distortion coding performance. In this case, the ability to reliably estimate the motion present in the scene remains a secondary goal.

## 10.1. Assessment of optical flow techniques

A quantitative assessment is introduced in (Barron, Fleet and Beauchemin, 1994), along with a comparative analysis of several optical flow motion estimation algorithms. However, a very limited data set is used.

More recently, an evaluation methodology is introduced in (Baker, S. Scharstein, D. Lewis, J.P. et al., 2011). The first step is to collect a ground-truth data set. A key difficulty is to be able to derive ground-truth for the optical flow. In order to cover a broad range of content and varying characteristics, four types of video data are considered:

- Real imagery of non-rigidly moving scene: Test video sequences are captured in visible light. In parallel, a dense optical flow ground-truth is captured in UV light using hidden fluorescent painted texture.
- Realistic synthetic imagery: Synthetic sequences are useful, as the motion can be precisely determined. Synthetic sequences obtained by rendering complex scenes with varying amount of motion, realistic textures, and occlusions are considered.
- Imagery for frame interpolation: Test sequences are temporally decimated. The temporally up-converted sequences obtained by frame interpolation can then be compared to the corresponding original sequences. In other words, instead of directly comparing the precision of the obtained motion vectors with a ground-truth field, it is proposed to evaluate the ability of the optical flow to provide an accurate motion-compensated interpolated frame, which may be more important in many application scenarios (Szeliski, 1999).
- Real stereo imagery of rigid scenes: Dense disparity ground-truth is captured for pairs of stereo images, using structured light (Scharstein and Szeliski, 2003).

This test data set is publicly available at http://vision.middlebury.edu/flow/.

We now discuss the methodology and measures used to assess performance. The Angular Error (AE) is often used to compare estimated and ground-truth flows. Let us define ($u_{GT}$,$v_{GT}$) the ground-truth optical flow, and ($u$,$v$) the estimated one. AE is then defined as

$$\mathrm{AE} = \cos^{-1}\left( \frac{1 + u \cdot u_{GT} + v \cdot v_{GT}}{\sqrt{1+u^2+v^2}\sqrt{1+u_{GT}^2+v_{GT}^2}} \right) . \qquad (79)$$

The denominator has for purpose to avoid a divide by zero in case of a null motion vector. However, it has for consequence to arbitrarily weight differently errors depending on the amplitude of the motion vector.

Avoiding this shortcoming, and thus probably more appropriate, the Error in flow Endpoint (EE) is defined as

$$\mathrm{EE} = \sqrt{(u-u_{GT})^2 + (v-v_{GT})^2} \ . \qquad (80)$$

When considering the frame interpolation scenario, the Interpolation Error (IE) is defined as the RMS difference between the ground-truth image, $f_{GT}(x,y)$, and the motion compensated interpolated one $\hat{f}(x,y)$,

$$\mathrm{IE} = \sqrt{\frac{1}{N}\sum\left(\hat{f}(x,y) - f_{GT}(x,y)\right)^2} \ . \qquad (81)$$

Alternatively, a gradient-normalized RMS is used in the Normalized interpolation Error (NE),

$$\mathrm{NE} = \sqrt{\frac{1}{N}\sum\frac{\left(\hat{f}(x,y) - f_{GT}(x,y)\right)^2}{\left\|\nabla f_{GT}(x,y)\right\|^2 + \varepsilon}} \ . \qquad (82)$$

To take into account the observation that optical flow estimation is harder in some regions of the image, error measure statistics are computed over three types of regions: around motion discontinuities, in texture-less regions, and over the whole image.

This evaluation methodology has been widely used by other researchers in the field, who frequently use the publicly available data set to report the performance of their algorithm. Many researchers have also uploaded their results to the website.

## 10.2. Assessment of motion estimation for video coding

In the context of video coding, the quality of the estimated motion field can be evaluated, similarly to Sec. 10.1, in order to assess the ability of the method to estimate the true motion in the scene. In particular, smooth motion vector fields are desired in order to prevent artificial discontinuities in the DFD and to reduce the overhead needed to transmit the motion information. A second measure of the quality of the motion estimation algorithm is the energy of the DFD, giving insight about the quality of the prediction.

However, a more relevant criterion to evaluate a motion estimation algorithm is to consider the global rate-distortion performance when it is used in a given video coding scheme. More specifically, the following aspects have to be considered:

- Objective quality metric: In order to assess the rate-distortion performance of a video coding scheme, the distortion, or equivalently the quality, of the reconstructed sequence should be estimated. PSNR is the most commonly used objective quality metric. It is computed between the reconstructed and original sequences, frame by frame, and in a range of bit rates of interest. However, it has been well-documented that PSNR is not always well correlated with the perceived visual quality (Wang and Bovik, 2009). It is largely due to the fact that PSNR totally ignores properties of the HVS.

  For this reason, perceptually-driven objective quality metrics have been proposed, for instance SSIM (Wang, Bovik, Sheikh et al., 2004), VIF (Sheikh and Bovik, 2006) or PSNR-HVS-M (Ponomarenko, Silvestri, Egiazarian *et al*., 2007). These metrics typically achieve better correlation with perceived visual quality when compared to PSNR.

- Subjective quality assessment: Given the limitations of objective quality metrics, subjective tests are needed in order to reliably and thoroughly assess the visual quality of a video sequence. For this purpose, several protocols have been defined, for instance for TV applications (ITU-R BT.500-12, 2009) or multimedia applications (ITU-R P.910, 2008).

  In the context of motion estimation, it is in particular key to visually inspect the reconstructed video sequence for the presence of distortions resulting from failures of the motion estimation technique.

  Moreover, a mismatch between the motion estimation technique and the coding strategy is another potential cause of distortions. For instance, visible blocking artifacts may be introduced in the case of a block-based motion estimation which, when is followed by a wavelet-based coding of the residual signal, i.e. a transform involving the entire image, may have a worse effect than in the case of an hybrid coding scheme (see also Section 6.2.4) .

Note that video coding standards such as H.264/AVC (Wiegand, Sullivan, Bjontegaard *et al*., 2003) or the forthcoming HEVC (Ohm and Sullivan, 2013), only specify the minimum requirements to guarantee interoperability. In other words, only the syntax and semantic of the code stream, along with the decoding process, are normative. Therefore, although the specific motion estimation technique used may significantly influence the performance of a video encoder, this part of the encoding process is not in the normative scope of the standard.

## 11 Summary and concluding remarks

In this chapter, we have reviewed some of the most important techniques for motion estimation. Motion estimation plays an important role in a broad range of applications encompassing image sequence analysis, computer vision and video communication. As these domains entail different requirements and constraints in terms of performance, we have mainly taken here a video coding view point.

As a preamble, we have first discussed the notion of apparent motion or optical flow. We have also introduced different models for motion representation.

After these preliminaries, we have discussed the main approaches for motion estimation: gradient-based techniques solving the optical flow equation, pel-recursive techniques iteratively minimizing the DFD, transform-domain techniques applied on Fourier, DCT or DWT coefficients, block-matching techniques which are widely adopted in video coding schemes, and finally parametric techniques to estimate the parameters of a motion model.

As a key component in image and video processing, we then described the concept of multi-resolution or multi-scale approaches, which typically leads to more accurate and robust motion vector field, along with reduced computation complexity.

Next, given our emphasis on video coding applications, we explained in more details different methods for motion compensation, including a more thorough description of the various modes enabled in the state-of-the-art H.264/AVC video coding standard.

To complete the chapter, we finally discussed methodologies to effectively assess the performance of motion estimation algorithms.

Motion estimation is a complex subject. In this chapter, we have aimed at a broad and comprehensive overview, although it is certainly not exhaustive. We have more thoroughly discussed some of the fundamental algorithms, and we have complemented them with descriptions of more recent and advanced developments. The large number of references gives the reader the opportunity to further explore different aspects and directions.

With continuous improvements over the last decades, current state-of-the-art motion estimation algorithms typically achieve good performances. Nevertheless, a general standard motion estimation technique remains elusive. In particular, diverse applications imply very different requirements and properties. However, further enhancements can be expected in the future.

## References

Adiv, G. Determining three-dimensional motion and structure from optical flow generated by several moving objects, IEEE Trans. Pattern Anal. Machine Intell. 1985, 7 (4), 384-401.

Anandan, P. A unified perspective on computational techniques for the measurement of visual motion, IEEE Proc. Int. Conf. on Computer Vision, London, UK, 1987.

Anandan, P.; Bergen, J.R.; Hanna, K.J.; Hingorani, R. Hierarchical model-based motion estimation, in M.I. Sezan and R.L. Lagendijk, editors, Motion Analysis and Image Sequence Processing, Kluzer Academic Publishers, pp. 1-22, 1993.

Ayer, S.; Schroeter, P.; Bigün, J. Segmentation of moving objects by robust motion parameter estimation over multiple frames, in Proc. European Conference on Computer Vision (ECCV), Stockholm, Sweden, 1994.

Baker, S. Scharstein, D. Lewis, J.P. et al. A Database and Evaluation Methodology for Optical Flow, Int. J. Comput. Vis. (2011) 92, pp. 1-31.

Barron, J.L.; Fleet, D.J.; Beauchemin, S. Performance of optical flow techniques. International Journal of Computer Vision (Springer), 1994.

Bernard, C. Fast optic flow computation with discrete wavelets, Technical Report, CMAP, Ecole Polytechnique, 1997.

Bernard, C. Wavelets and ill posed problems: optic flow and scattered data interpolation, Ph.D. Thesis, Ecole Polytechnique, 1998.

Bieling, M. Displacement estimation by hierarchical block matching, SPIE Proc. Visual Comm. and Image Processing, Cambridge, MA, Nov. 1988.

Biemond, J.; Looijenga, L.; Boekee, D.E. A pel-recursive Wiener-based displacement estimation algorithm, Signal Processing, 1987, 13(4), pp. 399–412.

Black, M.; Anandan, P. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. ComputerVision and Image Understanding (1996), 63(1), pp. 75–104.

Brox, T.; Bruhn, A.; Papenberg, N.; Weickert, J. High Accuracy Optical Flow Estimation Based on a Theory for Warping, Proc. European Conf. Computer Vision, 2004.

Bruhn, A.; Weickert, J.; Kohlberger, T.; Schnörr, C. A multigrid platform for real-time motion computation with discontinuity-preserving variational methods. International Journal of Computer Vision (2006), 70(3), pp. 257–277.

Burt, P.; Adelson, E. The Laplacian Pyramid as a Compact Image Code, IEEE Trans. on Communications (1983) 9(4), 532–540.

Burt, P.; Yen, C.; Xu, X. Multi-resolution flow-through motion analysis. In Proc. of the IEEE conference on computer vision and pattern recognition, 1983.

Cafforio, C.; Rocca, F. The differential method for motion estimation, Image Sequence Processing and Dynamic Scene Analysis, T.S. Huang, Ed., New York, Springer Verlag, 1983, pp. 104-124.

Cagnazzo, M.; Castaldo, F.; André, T.; Antonini, M.; Barlaud, M. Optimal Motion Estimation for Wavelet Video Coding, IEEE Transactions on Circuits and Systems for Video Technology, (2007), 17(7), pp. 907-911.

Chan, M.H.; Yu, Y.B. ; Constantinides, A.G. Variable size block matching motion compensation with applications to video coding, IEE Proceedings I - Communications, Speech and Vision, 1990, 137 (4), pp. 205 - 212.

Chang, M.M.; Tekalp, A.M.; Sezan, M.I. Simultaneous motion estimation and segmentation, *IEEE Transactions on Image Processing*, 1997, 6 (9), 1326-1333.

Chen, C.-Y.; Chien, S.-Y.; Huang, Y.-W.; Chen, T.-C.; Wang, T.-C.; Chen, L.-G. Analysis and architecture design of variable block-size motion estimation for H.264/AVC, IEEE Trans. on Circuits and Systems I: Regular Papers (2006), 53 (3), pp. 578-593.

Daubechies, I. The wavele transform, time-frequency localization and signal analysis, IEEE Trans. Inform. Theory, (1990), 36 (5), pp. 961-1005.

Diehl, N. Object-oriented motion estimation and segmentation in image sequences, Signal Processing: Image Communication, 1991, 3 (1), 23–56.

Divorra Escoda, O.; Yin, P.; Dai, C.; Li, X. Geometry-Adaptive Block Partitioning for Video Coding, in Proc. *IEEE International Conference on Acoustics, Speech and Signal Processing*, Honolulu, Hawaii, 2007.

Dufaux, F.; Moccagatta, I.; Rouchouze, B.; Ebrahimi, T.; Kunt, M. Motion compensated generic coding of video based on multiresolution data structure, *Optical Engineering* (1993), 32 (7), pp. 1559-1570.

Dufaux, F.; Moscheni, F. Motion estimation techniques for digital TV: a review and a new contribution, Proceedings of the IEEE, 1995, 83(6), pp. 858-876.

Dufaux, F.; Moscheni, F. Segmentation-based motion estimation for second generation video coding techniques, in L. Torres and M. Kunt, editors, *Second*

*Generation Video Coding Techniques*. Kluwer Academic Publishers, pp. 219-263, 1996.

Dufaux, F.; Moscheni, F. Background Mosaicking for Low Bit Rate Video Coding, in *IEEE Proc. Int. Conf. on Image Processing*, Lausanne, Switzerland, 1996.

Dufaux, F.; Konrad, J. Efficient, robust, and fast global motion estimation for video coding, IEEE Transactions on Image Processing, 2000, 9 (3), 497-501

Ebrahimi, T.; Dufaux, F.; Nakaya, Y. MPEG-4 Natural Video – II, in A. Puri and T. Chen, editors, *Multimedia Systems, Standards, and Networks*, Marcel Dekker, 2000, ch. 9, pp. 245-269.

Efstratiadis, S.N.; Katsaggelos, A.K. A model-based pel-recursive motion estimation algorithm, IEEE Proc. International Conference on Acoustics, Speech, and Signal Processing, Albuquerque, NM, 1990.

Efstratiadis, S.N.; Katsaggelos, A.K. An adaptive regularized recursive displacement estimation algorithm, IEEE Trans. on Image Processing, 1993, 2, pp.341-352.

Enkelmann, W. Investigations of multigrid algorithms for the estimation of optical flow fields in image sequences, Computer Vision, Graphics, and Image Processing (1988), 43 (2), pp. 150–177.

Farin, D.; Haller, M.; Krutz, A.; Sikora, T. Recent Developments in Panoramic Image Generation and Sprite Coding, IEEE Proc. Int. Workshop on Multimedia Signal Processing, 2008.

Fleet, D.J.; Jepson, A.D. Computation of component image velocity from local phase information, Int. Journal Computer Vision (1990), 5, pp. 77-104.

Fleet, D.J.; Weiss, Y. Optical Flow Estimation. In Paragios et al.. Handbook of Mathematical Models in Computer Vision. Springer. ISBN 0-387-26371-3, 2006.

Fuldseth, A.; Ramstad, T.A. Subband Video Coding with Smooth Motion Compensation, Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Proc., Atlanta, USA, May 1996.

Ghanbari, M. The cross-search algorithm for motion estimation, IEEE Trans. Commun. 1990, 38, pp. 950–953.

Glantz, A.; Krutz, A; Sikora, T. Adaptive Global Motion Temporal Prediction for Video Coding, in Proc. Picture Coding Symposium, Nagoya, Japan, Dec. 2010.

Glazer, F. Multilevel relaxation in low-level computer vision. A. Rosenfeld (Ed.), Multiresolution Image Processing and Analysis, Springer, Berlin (1984), pp. 312–330.

de Haan, G.; Biezen, P.W.A.C.; Huijgen, H.; Ojo, O.A. True-motion estimation with 3-D recursive search block matching, *IEEE Transactions on Circuits and Systems for Video Technology*, 1993, 3 (5), 368-379.

Haskell, B.G. Frame-to-frame coding of television pictures using two-dimensional Fourier transforms, IEEE Trans. on Information Theory (1974), 20 (1), pp. 119-120.

Hildreth, E. C. The measurement of visual motion. The MIT Press: Cambridge, MA, 1984.

Hoetter, M.; Thoma, R. Image segmentation based on object oriented mapping parameter estimation. Signal Processing, 1988, 15 (3), 315-334.

Horn, B.K.P. Robot Vision; The MIT Press: Cambridge, 1986; pp. 278-298.

Horn, B.K.P.; Schunck, B.G. Determining optical flow, Artif. Intelligence 1981, 17, 185-203.

Hung, E.M.; de Queiroz, R.L.; Mukherjee, D. On Macroblock Partition for Motion Compensation, in Proc. *IEEE International Conference on Image Processing*, Atlanta, GA, 2006

Irani, M.; Anandan, P.; Hsu, S. Mosaic based representations of video sequences and their applications, International Conference on Computer Vision, 1995.

Irani, M.; Hsu, S.; Anandan, P. Mosaic-based video compression, in SPIE Proc. Digital Video Compression: Algorithms and Technologies, San Jose, CA, 1995.

ITU-R BT.500-12, Methodology for the subjective of the quality of television pictures, 2009.

ITU-R P.910, Subjective video quality assessment methods for multimedia applications, 2008.

Jain, J.R.; Jain, A.K. Displacement measurement and its application in interframe image coding, IEEE Trans. Comm. 1981, 29 (12), pp. 1799-1808.

Jozawa, H.; Kamikura, K.; Sagata, A.; Kotera, H.; Watanabe, H. Two-stage motion compensation using adaptive global MC and local affine MC, IEEE Transactions on Circuits and Systems for Video Technology, 1997, 7(1), pp. 75-85.

Kamp, S.; Evertz, M.; Wien, M. Decoder side motion vector derivation for inter frame video coding, in Proc. *IEEE International Conference on Image Processing*, San Diego, CA, 2008.

Koc, U.V.; Ray Liu, K.J. Discrete-Cosine/Sine Transform Based Motion Estimation, IEEE Proc. Int. Conf. on Image Proc., Austin, TX, Nov. 1994, pp. 771-775.

Koga, T.; Iinuma, K.; Hirano, A.; Iijima, Y.; Ishiguro, T. Motion compensated interframe coding for video conferencing, in Proc. Nat. Telecommun. Conf., New Orleans, LA, Nov. – Dec. 1981, pp. G5.3.1–5.3.5

Krutz, A.; Glantz, A.; Frater, M.; Sikora, T. Rate-distortion optimized video coding using automatic sprites, IEEE J. Sel. Topics Signal Process., 2011, 5 (7), pp. 1309–1321.

Krutz, A.; Glantz, A.; Tok, M.; Esche, M.; Sikora, T. Adaptive Global Motion Temporal Filtering for High Efficiency Video Coding, IEEE Transactions on Circuits and Systems for Video Technology, 2012, 22(12).

Lakshman, H.; Schwarz, H.; Blu, T.; Wiegand, T. Generalized interpolation for motion compensated prediction, in *Proc. IEEE International Conference on Image Processing,* Brussels, Belgium, 2011

Lee, M. C.; Chen, W.; Lin, C. B., Gu, C., Markoc, T.;  Zabinsky, S. I.; Szeliski, R. A layered video object coding system using sprite and affine motion model, IEEE Trans. on Circuits and Systems for Video Technology, 1997, 7 (1), pp. 130–145.

Li, R.; Zeng, B.; Liou, M.L. A new three-step search algorithm for block motion estimation, IEEE Trans. Circuits Syst. Video Technol. 1994, 4, pp. 438–442.

Lu, Y.; Gao, W.; Wu, F. Efficient Background Video Coding With Static Sprite Generation and Arbitrary-Shape Spatial Prediction Techniques, IEEE Trans. on Circuits and Systems for Video Technology, 2003, 13(5).

Lucas, B. D.;  Kanade, T. An iterative image registration technique with an application to stereo vision, Proceedings of Imaging Understanding Workshop, 1981, pp. 121--130

Magarey, J.; Kingsbury, N. Motion estimation using a complex-valued wavelet transform, IEEE Trans. on Signal Processing 1998, 46(4), 1069-1084.

Meer, P.; Mintz, D.; Rosenfeld, A.; Kim, D.Y. Robust regression methods for computer vision: A review, Int. J. Comput. Vis. 1991, 6 (1), 59–70.

Moccagatta, I.; Moscheni, F.; Schütz, M.; Dufaux, F. A Motion Field Segmentation to Improve Moving Edges Reconstruction in Video Coding, in Proc. IEEE Int. Conf. on Image Proc., Austin, TX, 1994.

Moscheni, F.; Dufaux, F.; Kunt, M. A new two-stage global/local motion estimation based on a background/foreground segmentation, in IEEE Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), Detroit, MI, 1995.

MPEG-4 Video Verification Model Editing Committee, The MPEG-4 Video Verification Model 12.1, ISO/IEC JTC1/SC29/WG11 N2552, Rome, December 1998.

Netravali, A.; Robbins, J.D. Motion-compensated television coding part I, Bell Syst. Tech. J., 1979, 58(3), 629-668.

Nogaki, S.; Ohta, M. An overlapped block motion compensation for high quality motion picture coding, IEEE Proc. Int. Symp. on Circ. and Systems, San Diego, CA, May 1992, pp. 184-187.

Odobez, J.M.; Bouthemy, P. Robust Multiresolution Estimation of Parametric Motion Models, Journal of Visual Communication and Image Representation, 1995, 6 (4), 348–365.

Ohm, J.-R.; Sullivan, G.J. High Efficiency Video Coding: The next frontier in video compression, IEEE Signal Processing Magazine (2013), 30 (1), pp. 152-158.

Orchard, M.T.; Sullivan, G.J. Overlapped block motion compensation: an estimation-theoretic approach, *IEEE Transactions on Image Processing,* 1994, 3 (5), 693-699.

Pardas, M.; Salembier, P.; Gonzalez, B. Motion and region overlapping motion estimation for segmentation-based video coding, in IEEE Proc. Int. Conf. on Image Processing (ICIP), Austin, TX, 1994.

Po L.M.; Ma, W.C. A novel four-step search algorithm for fast block motion estimation, IEEE Trans. Circuits Syst. Video Technol. 1996, 6, pp. 313–317.

Ponomarenko, N.; Silvestri, F.; Egiazarian, K.; Carli, M.; Astola, J.; Lukin, V. On between-coefficient contrast masking of DCT basis functions, in Proc. of the Third International Workshop on Video Processing and Quality Metrics for Consumer Electronics (VPQM-07), Scottsdale, AZ, January 2007.

Press, W.H.; Flannery, B.P.; Teukolsky, S.A.; Vetterling, W.T. Numerical Recipes in C: The Art of Scientific Computing. Cambridge, U.K.: Cambridge Univ. Press, 1988

Rosseeuw P. J.; Leroy, A. M. Robust regression and outlier detection, New York: Wiley-Interscience, 1987.

Scharstein, D.; Szeliski, R. (2003). High-accuracy stereo depth maps using structured light. In Proc. of the IEEE conference on computer vision and pattern recognition, 2003.

Seitz, S.; Baker, S. Filter flow, in Proc. IEEE Int. Conf. on Computer Vision, 2009.

Sheikh, H.R.; Bovik, A.C. Image information and visual quality, IEEE Trans. on Image Processing (2006), 15 (2),pp. 430- 444.

Simoncelli, E. P. Bayesian multiscale differential optical flow, in H. Jähne and Geissler, Eds., Handbook of computer vision and applications. Academic Press, 1998.

Smolic, A.; Sikora, T.; Ohm, J.-R. Long-Term Global Motion Estimation and Its Application for Sprite Coding, Content Description, and Segmentation, IEEE Trans. on Circuits and Systems for Video Technology, 1999, 9 (8).

Smolic, A.; Hoeynck, M.; Ohm, J.-R. Low complexity global motion estimation from P-frame motion vectors for MPEG-7 applications, in IEEE Proc. Int. Conf. on Image Processing (ICIP) , Vancouver, Canada, 2000.

Stiller, C.; Konrad, J. Estimating motion in image sequences: a tutorial on modeling and computation of 2D motion. IEEE Signal Processing Magazine (1999), 16(4), 70–91.

Sullivan, G. J.; Ohm, J.-R.; Han, W.-J.; Wiegand, T. Overview of the High Efficiency Video Coding (HEVC) Standard, *IEEE Transactions on Circuits and Systems for Video Technology*, 2012, 22 (12), 1649-1668.

Szeliski, R.; Coughlan, J. Hierarchical spline-based image registration, in IEEE Proc. Computer Vision and Pattern Recognition. Seattle, Washington, 1994.

Szeliski, R. Video mosaics for virtual environments, IEEE Comput. Graph. Applicat., 1996, 16, pp. 22–30.

Szeliski, R. Prediction error as a quality metric for motion and stereo. In Proc. of the IEEE international conference on computer vision, 1999.

Teodosio, L.A.; Bender, W. Salient video stills: content and context preserved, in Proc. ACM Int. Conf. on Multimedia, Anaheim, CA, 1993.

Tok, M.; Glantz, A.; Arvanitidou, M.G.; Krutz, A.; Sikora, T. Compressed Domain Global Motion Estimation using the Helmholtz Tradeoff Estimator, in IEEE Proc. Int. Conf. on Image Processing (ICIP), Hong Kong, 2010.

Tse, Y.T.; Baker, R.L. Global zoom/pan estimation and compensation for video compression, in IEEE Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), Toronto, Canada, 1991.

Tziritas, G.; Labit, C. Motion Analysis for Image Sequence Coding,  Advances in Image Communication, Elsevier, 1994.

Ullman, S. The Interpretation of Visual Motion, The MIT Press: Cambrige, MA, 1979.

Walker, D.R.; Rao, K.R.  Improved Pel-recursive motion-estimation, IEEE Trans. Commun, 1984, 32(10), pp. 1128-1134.

Wang, J.Y.A.; Adelson, E.H. Representing moving images with layers, *IEEE Transactions on Image Processing*, 1994, 3 (5), 625-638.

Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: From error visibility to structural similarity, IEEE Trans. Image Processing (2004), 13 (4), pp. 600-612.

Wang, Z.; Bovik, A.C. Mean squared error: love it or leave it? A new look at signal fidelity measures, IEEE Signal Processing Magazine (2009), 26 (1), 98-117

Weber, J.; Malik, J. Robust computation of optical flow in a multi-scale differential framework, International Journal of Computer Vision 1995, 14(1), 5-19.

Wiegand, T.; Schwarz, H.; Joch, A.; Kossentini, F.; Sullivan, G.J. Rate-Constrained Coder Control and Comparison of Video Coding Standards. In *IEEE Transactions on Circuits and Systems for Video Technology*, 2003, 13 (7), 688-703.

Wiegand, T.; Sullivan, G.J.; Bjontegaard, G.; Luthra, A. Overview of the H.264/AVC video coding standard, IEEE Trans. on Circuits and Systems for Video Technology (2003), 13 (7), pp.560-576.

Wiegand, T.; Steinbach, E.; Girod, B. Affine multipicture motion-compensated prediction, *IEEE Transactions on Circuits and Systems for Video Technology*, 2005, 15 (2), 197- 209.

Wolberg, G. Digital Image Warping. IEEE Computer Society Press, Los Alamitos, California, 1990.

Wu, S.F.; Kittler, J. A differential method for simultaneous estimation of rotation, change of scale and translation, Signal Processing: Image Communication, 1990, 2 (1), 69-80.

Yap, S.Y.; McCanny, J.V. A VLSI architecture for variable block size video motion estimation, IEEE Trans. on Circuits and Systems II: Express Briefs (2004), 51 (7), pp. 384- 389.

Xu, L.; Jia, J.; Matsushita, Y. Motion detail preserving optical flow estimation. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2012, 34 (9), 1744-1757.

Zhu, C.; Lin, X.; Chau, L.-P. Hexagon-based search pattern for fast block motion estimation, *IEEE Transactions on Circuits and Systems for Video Technology*, 2002, 12 (5), pp.349-355.

Zhu, S.; Ma, K.-K. A new diamond search algorithm for fast block-matching motion estimation, IEEE Trans. on Image Processing (2000), 9 (2), pp. 287 - 290.

Zimmer, H.; Bruhn, A.; Weickert, J.; Valgaerts, B.R.L.; Salgado, A.; Seidel, H.-P. Complementary Optic Flow, in Proc. Int. Conf. Energy Minimization Methods in Computer Vision and Pattern Recognition, 2009.

## Relevant Websites

Optical Flow - Middlebury Computer Vision: http://vision.middlebury.edu/flow/

Optical Flow Algorithm Evaluation: http://of-eval.sourceforge.net/

The Moving Picture Experts Group website: http://mpeg.chiariglione.org/