

3.8

Motion Detection and Estimation

Janusz Konrad

*INRS-Télécommunications
Institut National de la Recherche Scientifique
Verdun, Québec, Canada H3E 1H6*

3.8.1 Introduction

A video sequence is a much richer source of visual information than a still image. This is primarily due to the capture of *motion*; while a single image provides a snapshot of a scene, a sequence of images registers the dynamics in it. The registered motion is a very strong cue for human vision; we can easily recognize objects as soon as they move even if they are inconspicuous when still. Motion is equally important for video processing and compression for two reasons. First, motion carries a lot of information about spatio-temporal relationships between image objects. This information can be used in such applications as traffic monitoring or security surveillance, for example to identify objects entering/leaving the scene or objects that just moved. Secondly, image properties, such as intensity or color, have a very high correlation in the direction of motion, i.e., they do not change significantly when tracked in the image (the color of a car does not change as the car moves across the image). This can be used for the removal of temporal video redundancy; in an ideal situation only the first image and the subsequent motion need to be transmitted. It can be also used for general temporal filtering of video. In this case, one-dimensional temporal filtering along a motion trajectory, e.g., for noise reduction or temporal interpolation, does not affect the spatial detail in the image.

The above applications require that image points be identified as moving or not (surveillance), or that it be measured how they move (compression, filtering). The first task is often referred to as *motion detection* while the latter as *motion estimation*. The goal of this chapter is to present today's most promising approaches to solving both. Note, that only two-dimensional (2-D) motion of intensity patterns in the image plane, often referred to as *apparent motion*, will be considered. 3-D motion of objects will not be treated here. *Motion segmentation*, i.e., the identification of groups of image points moving similarly, is treated in Chapter 4.8.

The discussion of motion in this chapter will be carried out from the point of view of video processing and compression. Necessarily, the scope of methods reported will not be complete. To present the methods in a consistent fashion, a classification will be made based on *models*, *estimation criteria* and *search strategies* used. This classification will be introduced for two reasons. First, it is essential for the understanding of methods described here and elsewhere in the literature. Secondly, it should help the reader in the development of his/her own motion detection or estimation method.

Although motion detection and estimation still require specialized hardware for real-time execution, the present rapid growth of CPU power available in a personal computer will soon allow execution of motion-related tasks in software on a general CPU. This will certainly spawn new applications and an even greater need for robust, flexible and fast motion detection and estimation algorithms. Hopefully, in designing a new algorithm or understanding an existing one, the reader will be able to exploit the variety of tools presented here.

The chapter is organized as follows. In the next section, the notation is established, followed by a brief review of some tools needed. Then, in Section 3.8.3, motion detection is discussed from the point of view of hypothesis testing and MAP detection. In Section 3.8.4, motion estimation is described in two parts. First, models, estimation criteria and search strategies are discussed. Then, five motion estimation algorithms are described in more detail, of which three are based on models supported by the current video compression standards. Both motion detection and estimation are illustrated by numerous experimental results.

3.8.2 Notation and preliminaries

In this chapter, both continuous and discrete representations of motion and images will be used, with bold characters denoting vectors. Let $\mathbf{x} = (x, y)^T$ be a spatial position of a pixel in continuous coordinates, i.e., $\mathbf{x} \in R^2$ within image limits, and let I_t denote image intensity at time t . Then, $I_t(\mathbf{x}) \in R$ is limited by the dynamic range of the sensing device (e.g., vidicon, CCD). Before images can be manipulated digitally, they need to be sampled and quantized. Let $\mathbf{n} = (n_1, n_2)^T$ be a discretized spatial position in the image that corresponds to \mathbf{x} . Similarly, let k be a discretized position in time, also denoted t_k . The triplet $(n_1, n_2, k)^T$ belongs to a 3-D sampling grid, for example a 3-D lattice (Chapter 7.2). It is assumed here that images are either continuous or discrete simultaneously in position and in amplitude. Consequently, the same symbol I will be used for continuous *and* quantized intensities; the nature of I can be inferred from its argument (continuous-valued for \mathbf{x} and quantized for \mathbf{n}).

Motion in continuous images can be described by a *velocity* vector $\boldsymbol{\nu} = (\nu_1, \nu_2)^T$. While $\boldsymbol{\nu}(\mathbf{x})$ is a velocity at spatial position \mathbf{x} , $\boldsymbol{\nu}_t$ will denote a *velocity field* or *motion field*, i.e., the set of all velocity vectors within the image, at time t . Often the computation of this *dense* representation is replaced by the computation of a small number of motion parameters \mathbf{b}_t with the benefit of reduced computational complexity. Then, $\boldsymbol{\nu}_t$ is approximated by \mathbf{b}_t *via* a known transformation. For discrete images, the notion of velocity is replaced by *displacement* \mathbf{d} .

3.8.2.1 Hypothesis testing

Let y be an *observation* and let Y be the associated random variable. Suppose that there are two hypotheses H_0 and H_1 with corresponding probability densities $p_Y(y|H_0)$ and $p_Y(y|H_1)$, respectively. The goal is to decide from which of the two densities a given y is more likely to have been drawn. Clearly, 4 possibilities exist (true hypothesis/decision): H_0/H_0 , H_0/H_1 , H_1/H_0 , H_1/H_1 . Whereas H_0/H_0 and H_1/H_1 correspond to correct choices, H_0/H_1 and H_1/H_0 are erroneous. In order to take a decision, a *decision criterion* is needed that attaches some relative importance to the four possible scenarios.

Under the *Bayes criterion*, two *a priori* probabilities P_0 and $P_1 = 1 - P_0$ are assigned to the two hypotheses H_0 and H_1 , respectively, and a cost is assigned to each of the four scenarios listed above. Naturally, one would like to design a decision rule so that on average the cost associated with making a decision based on y is minimal. By computing an average risk and by assuming that costs associated with erroneous decisions are higher than those associated with the corresponding correct decisions, it can be shown that an optimal decision can be made according to the following rule [24, Chapter 2]:

$$\frac{p_Y(y|H_1)}{p_Y(y|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \theta \frac{P_0}{P_1}. \quad (3.8.1)$$

The quantity on the left is called the *likelihood ratio* and θ is a constant dependent on the costs of the 4 scenarios. Since these costs are determined in advance, θ is a fixed threshold. If P_0 and P_1 are pre-determined as well, the above hypothesis test compares the likelihood ratio with a given threshold. Alternatively, the prior probabilities can be made variable; variable-threshold hypothesis testing results.

3.8.2.2 Markov random fields

A Markov random field (MRF) is a multidimensional random process which generalizes the notion of a 1-D Markov process. Below, some essential properties of MRFs are described; for more details the reader is referred to Chapter 4.2 and to the literature (e.g., [9] and references therein).

Let Λ be a sampling grid in R^N and let $\eta(\mathbf{n})$ be a *neighborhood* of $\mathbf{n} \in \Lambda$, i.e., a set of such \mathbf{n} 's that $\mathbf{n} \notin \eta(\mathbf{n})$ and $\mathbf{n} \in \eta(\mathbf{l}) \Leftrightarrow \mathbf{l} \in \eta(\mathbf{n})$. The first-order neighborhood consists of immediate top, bottom, left and right neighbors of \mathbf{n} . Let \mathcal{N} be a *neighborhood system*, i.e., a collection of neighborhoods of all $\mathbf{n} \in \Lambda$.

A random field Υ over Λ is a multidimensional random process where each site $\mathbf{n} \in \Lambda$ is assigned a random variable. A random field Υ with the following properties:

1. $P(\Upsilon = v) > 0, \quad \forall v \in \Gamma$, and
2. $P(\Upsilon_{\mathbf{n}} = v_{\mathbf{n}} | \Upsilon_{\mathbf{l}} = v_{\mathbf{l}}, \forall \mathbf{l} \neq \mathbf{n}) = P(\Upsilon_{\mathbf{n}} = v_{\mathbf{n}} | \Upsilon_{\mathbf{l}} = v_{\mathbf{l}}, \forall \mathbf{l} \in \eta(\mathbf{n})), \quad \forall \mathbf{n} \in \Lambda, \forall v \in \Gamma$,

where P is a probability measure, is called a Markov random field with state space Γ .

In order to define the Gibbs distribution, the concepts of clique and potential function are needed. A *clique* c defined over Λ with respect to \mathcal{N} is a subset of Λ such that either c consists of a single site or every pair of sites in c are neighbors, i.e., belong to η . The set of all cliques is denoted by \mathcal{C} . Examples of a two-element spatial clique $\{\mathbf{n}, \mathbf{l}\}$ are two immediate horizontal, vertical or diagonal neighbors. A *Gibbs distribution* with respect to Λ and \mathcal{N} is a probability measure π on Γ such that

$$\pi(v) = \frac{1}{Z} e^{-U(v)/T},$$

where the constants Z and T are called the *partition function* and *temperature*, respectively, and the *energy function* U is of the form

$$U(v) = \sum_{c \in \mathcal{C}} V(v, c).$$

$V(v, c)$ is called a *potential function*, and depends only on the value of v at sites that belong to the clique c .

The equivalence between Markov random fields and Gibbs distributions is provided through the important *Hammersley-Clifford theorem* which states that Υ is a MRF on Λ with respect to \mathcal{N} if and only if its probability distribution is a Gibbs distribution with respect to Λ and \mathcal{N} . The equivalence between MRFs and Gibbs distributions results in a straightforward relationship between qualitative properties of a MRF and its parameters via the potential functions V . Extension of the Hammersley-Clifford theorem to vector MRFs is straightforward (new definition of a state is needed).

3.8.2.3 MAP estimation

Let Y be a random field of observations and let Υ be a random field that we want to estimate based on Y . Let y, v be their respective realizations. For example, y could be a difference between two images while v could be a field of motion detection labels. In order to compute v based on y , a powerful tool is the maximum *a posteriori* probability (MAP) estimation, expressed as follows:

$$\hat{v} = \arg \max_v P(\Upsilon = v | y) = \arg \max_v P(Y = y | v) \cdot P(\Upsilon = v), \quad (3.8.2)$$

where $\max_v P(\Upsilon = v | y)$ denotes the maximum of the posterior probability $P(\Upsilon = v | y)$ with respect to v and \arg denotes the argument of this maximum, i.e., such \hat{v} that $P(\Upsilon = \hat{v} | y) \geq P(\Upsilon = v | y)$ for any v . Above, the Bayes rule was used and $P(Y = y)$ was omitted since it does not depend on v . If $P(\Upsilon = v)$ is the same for all realizations v , then only the likelihood $P(Y = y | v)$ is maximized, resulting in the *maximum likelihood* (ML) estimation.

3.8.3 Motion detection

Motion detection is, arguably, the simplest of the three motion-related tasks, i.e., detection, estimation and segmentation. Its goal is to identify which image points, or more generally which regions, of the image have moved between two time instants. As such, motion detection applies only to images acquired with a static camera. However, if camera motion can be counteracted, e.g., by global motion estimation and compensation, then the method equally applies to images acquired with a moving camera [14, Chapter 8].

It is essential to realize that the motion of image points is not perceived directly but rather through intensity changes. However, such intensity changes over time may be also induced by camera noise or illumination changes. Moreover, object motion itself may induce small intensity variations or even none at all. The latter will happen in the rare case of exactly constant luminance and color within the object. Clearly, motion detection from time-varying images is not as easy as may seem initially.

3.8.3.1 Hypothesis testing with fixed threshold

Fixed-threshold hypothesis testing belongs to the simplest motion detection algorithms as it requires very few arithmetic operations. Several early motion detection methods belong to this class, although originally they were not developed as such.

Let $H_{\mathcal{M}}$ and $H_{\mathcal{S}}$ be two hypotheses declaring an image point at \mathbf{n} as moving (\mathcal{M}) and stationary (\mathcal{S}), respectively. Let's assume that $I_k(\mathbf{n}) = I_{k-1}(\mathbf{n}) + q$ and that q is a noise term (Chapter 4.4), zero-mean Gaussian with variance σ^2 in stationary areas and zero-mean uniformly distributed in $[-L, L]$ in moving areas. The motivation is that in stationary areas only camera noise will distinguish same-position pixels at t_{k-1} and t_k , while in moving areas this difference is attributed to motion and therefore unpredictable. Then, let

$$\rho_k(\mathbf{n}) = I_k(\mathbf{n}) - I_{k-1}(\mathbf{n})$$

be an observation (3.8.1) upon which we intend to select one of the two hypotheses. With the above assumptions and after taking the natural logarithm of both sides of (3.8.1) the hypothesis test can be written as follows:

$$\rho_k^2(\mathbf{n}) \underset{\mathcal{S}}{\overset{\mathcal{M}}{\gtrless}} \theta, \quad (3.8.3)$$

where the threshold θ equals $2\sigma^2 \ln((2L \cdot P_{\mathcal{S}})/(\sqrt{2\pi\sigma^2} \cdot P_{\mathcal{M}}))$. A similar test can be derived for a Laplacian-distributed noise term q ; in (3.8.3) $\rho_k^2(\mathbf{n})$ is replaced by $|\rho_k(\mathbf{n})|$ and θ is computed accordingly. Such a test was used in the early motion detection algorithms. Note that both the Laplacian and Gaussian cases are equivalent under appropriate selection of θ . Although θ includes the prior probabilities, they are usually fixed in advance as is the noise variance, and thus the test is parameterized by one constant.

The above pixel-based hypothesis test is not robust to noise in the image; for small θ 's "noisy" detection masks result (many isolated small regions or even pixels) while for large θ 's only object boundaries and its most textured parts are detected. To attenuate the impact of noise, the method can be extended by measuring the temporal differences over a spatial window \mathcal{W} with N points:

$$\frac{1}{N} \sum_{\mathbf{n} \in \mathcal{W}} \rho_k^2(\mathbf{n}) \underset{\mathcal{S}}{\overset{\mathcal{M}}{\gtrless}} \theta.$$

This approach exploits the fact that typical variations of camera characteristics, such as camera noise, can be closely approximated by an additive white noise model; by averaging over \mathcal{W} the noise impact is reduced. Still, the method is not very robust and therefore is usually followed by some post-processing (such as median filtering, suppression of small regions, etc.). Moreover, since the classification at

position \mathbf{n} is done based on all pixels within \mathcal{W} , the resolution of the method is reduced; a moving image point affects the decision of many of its neighbors. This method can be further improved by modeling the intensities I_{k-1} and I_k within \mathcal{W} by a polynomial function, e.g., linear or quadratic [11].

The methods discussed thus far cannot deal with illumination changes between t_{k-1} and t_k ; any intensity change caused by illumination variation is interpreted as motion. In the case of a global illumination change (across the whole image) a normalization of intensities can be used to match the second-order statistics in I_{k-1} and I_k . By allowing a linear transformation $\hat{I}_k(\mathbf{n}) = aI_k(\mathbf{n}) + b$, coefficients a and b can be found so that such statistics at t_k and t_{k-1} are equal, i.e., $\hat{\mu}_k = \mu_{k-1}$ and $\hat{\sigma}_k^2 = \sigma_{k-1}^2$ where $\hat{\mu}$ and $\hat{\sigma}$ are, respectively, mean and variance of the normalized image and μ and σ are those for the original image. Solving for a and b yields:

$$\hat{I}_k(\mathbf{n}) = \frac{\sigma_{k-1}}{\sigma_k} (I_k(\mathbf{n}) - \mu_k) + \mu_{k-1}.$$

This transformation helps in the case of global or almost-global illumination change only. However, a typical illumination change varies with \mathbf{n} and moreover is often localized. The above normalization can be made adaptive on a region-by-region basis but for fixed regions (e.g., blocks) the improvements are marginal and it is unclear how to adapt the region shape.

A different approach to handling illumination change is to compare intensity gradients rather than intensities themselves, i.e., to construct the following test:

$$\frac{1}{N} \sum_{\mathbf{n} \in \mathcal{W}} \|\nabla I_k(\mathbf{n}) - \nabla I_{k-1}(\mathbf{n})\| \underset{\mathcal{S}}{\overset{\mathcal{M}}{\geq}} \theta,$$

where $\nabla = (\partial/\partial x, \partial/\partial y)^T$ is the spatial gradient and $\|\cdot\|$ is a suitable norm, e.g., Euclidean or city-block distance. This approach, applied to the polynomial-based intensity model [11], has been shown to increase robustness in the presence of illumination changes [18]. However, the method can handle the multiplicative nature of illumination only approximately (the intensity gradient above is not invariant under intensity scaling). To address this issue in more generality, shading models extensively used in computer graphics must be employed [18].

3.8.3.2 Hypothesis testing with adaptive threshold

The motion detection methods presented thus far were based solely on image intensities and made no *a priori* assumptions about the nature of moving areas. However, moving 3-D objects usually create compact, closed boundaries in the image plane, i.e., if an image point is declared moving, it is likely that its neighbor is moving as well (unless the point is on a boundary) and the boundary is smooth rather than rough. To take advantage of this *a priori* information, hypothesis testing can be combined with Markov random field models.

Let E_k be a MRF of all labels assigned at time t_k , and let e_k be its realization. Let's assume for the time being that $e_k(\mathbf{n})$ is known for all \mathbf{n} except \mathbf{l} . Since the estimation process is iterative, this assumption is not unreasonable; previous estimates are known at $\mathbf{n} \neq \mathbf{l}$. Thus, the estimation process is reduced to deciding between $e_k(\mathbf{l}) = \mathcal{M}$ and $e_k(\mathbf{l}) = \mathcal{S}$. Let the label field resulting from $e_k(\mathbf{l}) = \mathcal{M}$ be denoted by $e_k^{\mathcal{M}}$ and that produced by $e_k(\mathbf{l}) = \mathcal{S}$ be $e_k^{\mathcal{S}}$. Then, based on (3.8.1) the decision rule for $e_k(\mathbf{l})$ can be written as follows:

$$\frac{p(\rho_k|e_k^{\mathcal{M}})}{p(\rho_k|e_k^{\mathcal{S}})} \underset{\mathcal{S}}{\overset{\mathcal{M}}{\geq}} \theta \frac{P(E_k = e_k^{\mathcal{S}})}{P(E_k = e_k^{\mathcal{M}})}, \quad (3.8.4)$$

where P is a probability distribution governing the MRF E_k . By making the simplifying assumption that the temporal differences $\rho_k(\mathbf{n})$ are conditionally independent, i.e., $p(\rho_k|e_k) = \prod_{\mathbf{n}} p(\rho_k(\mathbf{n})|e_k(\mathbf{n}))$,

equation (3.8.4) can be further re-written:

$$\frac{p(\rho_k(\mathbf{l})|H_{\mathcal{M}})}{p(\rho_k(\mathbf{l})|H_{\mathcal{S}})} \underset{\mathcal{S}}{\overset{\mathcal{M}}{\gtrless}} \theta \frac{P(E_k = e_k^{\mathcal{S}})}{P(E_k = e_k^{\mathcal{M}})}. \quad (3.8.5)$$

The hypothesis $H_{\mathcal{M}}$ means that $e_k(\mathbf{l}) = \mathcal{M}$ and $H_{\mathcal{S}}$ means that $e_k(\mathbf{l}) = \mathcal{S}$. All constituent probability densities from the left-hand side of equation (3.8.4), except at \mathbf{l} , cancel out since $e_k^{\mathcal{M}}$ and $e_k^{\mathcal{S}}$ differ only at \mathbf{l} . Although the conditional independence assumption is reasonable in stationary areas (temporal differences are mostly due to camera noise), it is less so in the moving areas. However, a convincing argument based on experimental results can be made in favor of such independence [1].

To increase the detection robustness to noise, the temporal differences should be pooled together, for example within a spatial window \mathcal{W}_l centered at \mathbf{l} . This leads to the evaluation of the likelihood for all ρ_k within \mathcal{W}_l given the hypothesis $H_{\mathcal{M}}$ or $H_{\mathcal{S}}$ at \mathbf{l} . Under the assumption of zero-mean Gaussian density p with variances $\sigma_{\mathcal{M}}^2$ and $\sigma_{\mathcal{S}}^2$ for $H_{\mathcal{M}}$ and $H_{\mathcal{S}}$, respectively, and assuming that $\sigma_{\mathcal{M}}^2 \gg \sigma_{\mathcal{S}}^2$, the final hypothesis becomes:

$$\sum_{\mathbf{n} \in \mathcal{W}_l} \rho_k^2(\mathbf{n}) \underset{\mathcal{S}}{\overset{\mathcal{M}}{\gtrless}} 2\sigma_{\mathcal{S}}^2 \left(-\ln \theta + N \ln \frac{\sigma_{\mathcal{M}}}{\sigma_{\mathcal{S}}} + \ln \frac{P(E_k = e_k^{\mathcal{S}})}{P(E_k = e_k^{\mathcal{M}})} \right), \quad (3.8.6)$$

where N is the number of pixels in \mathcal{W}_l . In case the *a priori* probabilities are identical or fixed (independent of the realization e_k), the overall threshold depends only on model variances. Then, for increasing $\sigma_{\mathcal{S}}^2$, the overall threshold rises as well thus discouraging \mathcal{M} labels (as camera noise increases only large temporal differences should induce moving labels). Conversely, for decreasing $\sigma_{\mathcal{S}}^2$, the threshold falls, thus biasing the decision towards moving labels. In the limit, as $\sigma_{\mathcal{S}}^2 \rightarrow 0$, the threshold becomes 0; for a noiseless camera even the slightest temporal difference will induce a moving label.

By suitably defining the *a priori* probabilities one can adapt the threshold in response to the properties of e_k . Since the required properties are object compactness and smoothness of its boundaries, a simple MRF model supported on a first-order neighborhood [9] with two-element cliques $c = \{\mathbf{n}, \mathbf{l}\}$ and the following potential function

$$V_{\mathbf{n}\mathbf{l}} = \begin{cases} 0 & \text{if } e_k(\mathbf{n}) = e_k(\mathbf{l}), \\ \beta & \text{if } e_k(\mathbf{n}) \neq e_k(\mathbf{l}), \end{cases} \quad (3.8.7)$$

is appropriate. Whenever a neighbor of \mathbf{n} has a different label than $e_k(\mathbf{n})$, a penalty $\beta > 0$ is incurred; summed over the whole field it is proportional to the length of the moving mask boundary. Thus, the resulting prior (Gibbs) probability

$$P(E_k = e_k) = \frac{1}{Z} \exp\left(-\frac{1}{T} \sum_{\{\mathbf{n}, \mathbf{l}\}} V_{\mathbf{n}\mathbf{l}}\right)$$

will increase for configurations with smooth boundaries and will reduce for those with rough boundaries. More advanced models, for example based on second-order neighborhood systems with diagonal cliques, can be used similarly [1].

Note that the MRF model facilitates the use of adaptive thresholds: if $P(E_k = e_k^{\mathcal{S}}) > P(E_k = e_k^{\mathcal{M}})$, the fixed part of the threshold in (3.8.6), i.e., the first two terms, will be augmented by a positive number thus biasing the decision towards a static label. Conversely, for $P(E_k = e_k^{\mathcal{S}}) < P(E_k = e_k^{\mathcal{M}})$ the bias is in favor of a moving label.

3.8.3.3 MAP detection

The MRF label model introduced in the previous section can be combined with another Bayesian criterion, namely the MAP criterion (Section 3.8.2.3). To find the MAP estimate of the label field E_k ,

the posterior probability $P(E_k = e_k | \rho_k)$, or its Bayes equivalent $p(\rho_k | e_k) \cdot P(E_k = e_k)$, needs to be maximized.

Let's consider the likelihood $p(\rho_k | e_k)$. One of the questionable assumptions made in the previous section was the conditional independence of the ρ_k given e_k (3.8.5). To alleviate this problem, let $|I_k(\mathbf{n}) - I_{k-1}(\mathbf{n})|$ be an observation modeled as $\rho_k(\mathbf{n}) = \xi(e_k(\mathbf{n})) + q(\mathbf{n})$ where q is zero-mean uncorrelated Gaussian noise with variance σ^2 and

$$\xi(e_k(\mathbf{n})) = \begin{cases} 0 & \text{if } e_k(\mathbf{n}) = \mathcal{S}, \\ \alpha & \text{if } e_k(\mathbf{n}) = \mathcal{M}. \end{cases}$$

Above, α is considered to be an average of the observations in moving areas. For example, α could be computed as an average temporal difference for previous-iteration moving labels e_k or previous-time moving labels e_{k-1} . Clearly, ξ attempts to closely model the observations since for a static image point it is zero, while for a moving point it tracks average temporal intensity mismatch; the uncorrelated q should be a better approximation here than in the previous section.

Under the uncorrelated Gaussian assumption for the likelihood $p(\rho_k | e_k)$ and a Gibbs distribution for the *a priori* probability $P(E_k = e_k)$, the overall energy function can be written as follows:

$$\begin{aligned} U(\rho_k, e_{k-1}, e_k) &= \frac{1}{2\sigma^2} \sum_{\mathbf{n}} (\rho_k(\mathbf{n}) - \xi(e_k(\mathbf{n})))^2 + \sum_{\{\mathbf{n}, \mathbf{l}\}} V_s(e_k(\mathbf{n}), e_k(\mathbf{l})) \\ &+ \sum_{\{t_{k-1}, t_k\}} V_t(e_{k-1}(\mathbf{n}), e_k(\mathbf{n})). \end{aligned}$$

The first term measures how well the current label $e_k(\mathbf{n})$ explains the observation $\rho_k(\mathbf{n})$. The other terms measure how contiguous the labels are in the image plane (V_s) and in time (V_t). Both V_s and V_t are specified similarly to (3.8.7) thus favoring spatial and temporal similarity of the labels [6]. This basic model can be enhanced by selecting a more versatile model for ξ [6] or a more complete prior model for including spatio-temporal, as opposed to purely spatial and temporal, cliques [15]. The above cost function can be optimized using various approaches, such as those discussed in Section 3.8.4.3 (simulated annealing, iterated conditional modes or highest confidence first). The latter method, based on an adaptive selection of visited labels according to their impact on the energy U (most influential visited first), gives the best compromise between performance (final energy value) and computing time.

3.8.3.4 Experimental comparison of motion detection methods

Fig. 3.8.1 shows the original images as well as the binary label fields obtained using the MAP detection mechanism discussed in Section 3.8.3.3, as well as those obtained by fixed-threshold hypothesis test (3.8.3). Note the compactness of the detection mask and the smoother boundary in case of the MAP estimate as opposed to the noisy detection result obtained by thresholding. The improvement is due primarily to the *a priori* knowledge incorporated into the algorithm.

3.8.4 Motion estimation

As mentioned in the introduction, knowledge of motion is essential for both the compression and processing of image sequences. Although compression is often considered to be encompassed by processing, a clear distinction between these two terms will be made here. Methods explicitly reducing the number of bits needed to represent a video sequence will be classified as video compression techniques. For example, motion-compensated hybrid (predictive/DCT) coding is exploited today in all video compression standards, i.e., H.261, H.263, MPEG-1, MPEG-2, MPEG-4 (Chapters 6.1, 6.4 and 6.5). On the other hand, methods that do not attempt such a reduction but transform the video sequence,

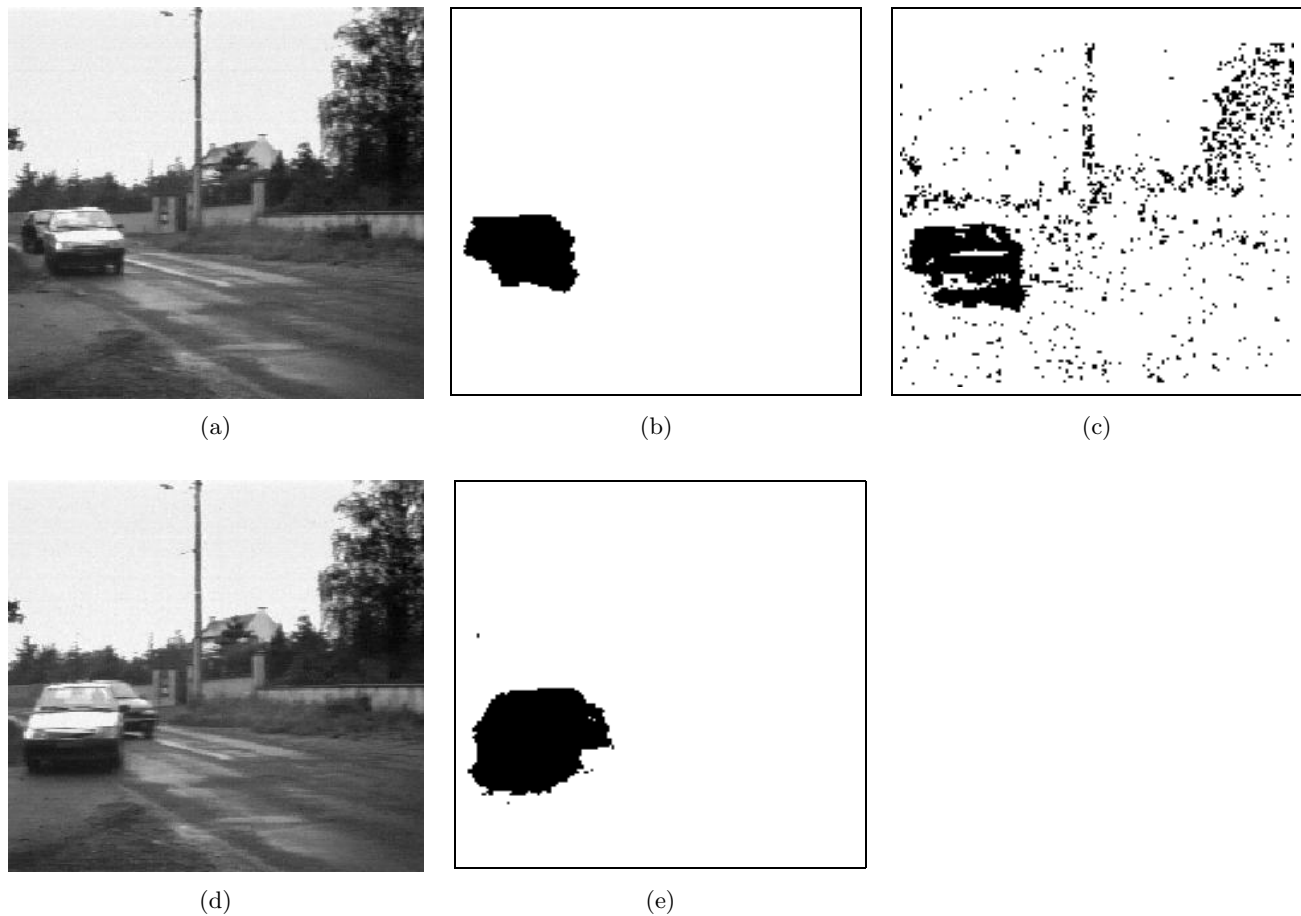


Figure 3.8.1: Motion detection results for two frames of a road traffic sequence of images: (a) frame #137; (b) MAP-detected motion for frame #137; (c) motion in frame #137 detected by fixed-threshold hypothesis test; (d) frame #143; (e) MAP-detected motion for frame #143. (From Bouthemy and Lalonde [6]. Reproduced with permission of the International Society for Optical Engineering (SPIE)).

e.g., to improve quality, will be considered to belong to video processing methods. Examples of video processing are motion-compensated noise reduction (Chapter 3.9), motion-compensated interpolation (Chapter 3.9), motion-based video segmentation (Chapter 4.8).

The above classification is important from the point of view of the goals of motion estimation that, in turn, influence the choice of models and estimation criteria. In the case of video compression, the estimated motion parameters should lead to the highest compression ratio possible (for a given video quality). Therefore, the computed motion need not resemble the *true* motion of image points as long as some minimum bit rate is achieved. In video processing, however, it is the *true* motion of image points that is sought. For example, in motion-compensated temporal interpolation (Fig. 3.8.2) the task is to compute new images located between the existing images of a video sequence (e.g., video frame rate conversion between NTSC and PAL scanning standards). In order that the new images be consistent with the existing ones, image points must be displaced according to their true motion as otherwise “jerky” motion of objects would result. This is a very important difference that influences the design of motion estimation algorithms and, most importantly, that usually precludes a good performance of a compression-optimized motion estimation algorithm in video processing and *vice versa*.

In order to develop a motion estimation algorithm, three important elements need to be considered: *models*, *estimation criteria* and *search strategies*. They will be discussed next, but no attempt will be made to include an exhaustive list pertaining to each of them. Clearly, this cannot be considered a

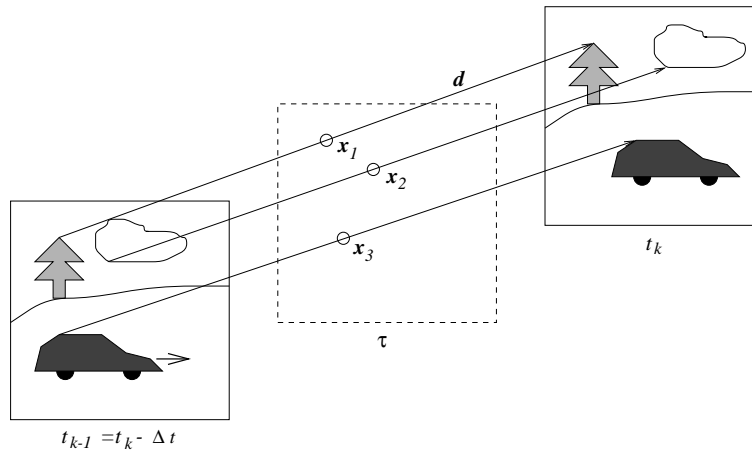


Figure 3.8.2: Motion-compensated interpolation between images at time $t_k - \Delta t$ and t_k . Motion compensation is essential for smooth rendition of moving objects. Shown are three motion vectors that map the corresponding image points at time $t_k - \Delta t$ and t_k onto image at time τ .

universal classification scheme of motion estimation algorithms, but it is very useful in understanding the properties and merits of various approaches. Then, five practical motion estimation algorithms will be discussed in more detail.

3.8.4.1 Motion models

There are two essential models in motion estimation: a *motion model*, i.e., how to represent motion in an image sequence, and a model relating motion parameters to image intensities, called an *observation model*. The latter model is needed since, as was mentioned before, computation of motion is carried out indirectly by examining intensity changes.

Spatial motion models

The goal is to estimate the motion of image points, i.e., the *2-D motion* or *apparent motion*. Such a motion is a combination of projections of the motion of objects in a 3-D scene and of 3-D camera motion. Whereas camera motion affects the movements of all or almost all image points, the motion of 3-D objects only affects a subset of image points corresponding to objects' projections. Since, in principle, the camera-induced motion can be compensated for by either estimating it (Section 3.8.5.1) or by physically measuring it at the camera, we need to model the object-induced motion only. Such a motion depends on:

1. image formation model, e.g., perspective, orthographic projection [23],
2. motion model of 3-D object, e.g., rigid-body with 3-D translation and rotation, 3-D affine motion,
3. surface model of 3-D object, e.g., planar, parabolic.

In general, these relationships are fairly complex but two cases are relatively simple and have been used extensively in practice. For an orthographic projection and arbitrary 3-D surface undergoing 3-D translation, the resulting 2-D instantaneous velocity at position \mathbf{x} in the image plane is described by a 2-D vector:

$$\boldsymbol{\nu}(\mathbf{x}) = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \quad (3.8.8)$$

where parameters $\mathbf{b} = (b_1, b_2)^T = (\nu_1, \nu_2)^T$ depend on camera geometry and 3-D translation parameters. This 2-D translational model has proven very powerful in practice, especially in video compression, since locally it provides a close approximation for most natural images.

The second powerful, yet simple, parametric model is that of orthographic projection combined with 3-D affine motion of a planar surface. It leads to the following 6-parameter *affine* model [22, Chapter 6]:

$$\boldsymbol{\nu}(\mathbf{x}) = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} + \begin{pmatrix} b_3 & b_4 \\ b_5 & b_6 \end{pmatrix} \mathbf{x} \quad (3.8.9)$$

where, again, $\mathbf{b} = (b_1, \dots, b_6)^T$ is a vector of parameters related to the camera as well as 3-D surface and motion parameters. Clearly, the translational model above is a special case of the affine model. More complex models have been proposed as well but, depending on the application, they do not always improve the precision of motion estimates. In general, the higher the number of motion parameters, the more precise the description of motion. However, an excessive number of parameters may be detrimental to the performance. This depends on the number of degrees of freedom, i.e., model complexity (dimensionality of \mathbf{b} and the functional dependence of $\boldsymbol{\nu}$ on x, y) versus the size of the region of support (see below). A complex model applied to a small region of support may lead to an actual increase in the estimation error compared to a simpler model such as in (3.8.9).

Temporal motion models

The trajectories of individual image points drawn in the (x, y, t) space of an image sequence can be fairly arbitrary since they depend on object motion. In the simplest case, trajectories are linear, such as the ones shown in Fig. 3.8.2. Assuming that the velocity $\boldsymbol{\nu}_t(\mathbf{x})$ is constant between $t = t_{k-1}$ and τ ($\tau > t$), a linear trajectory can be expressed as follows

$$\mathbf{x}(\tau) = \mathbf{x}(t) + \boldsymbol{\nu}_t(\mathbf{x}) \cdot (\tau - t) = \mathbf{x}(t) + \mathbf{d}_{t,\tau}(\mathbf{x}), \quad (3.8.10)$$

where $\mathbf{d}_{t,\tau}(\mathbf{x}) = \boldsymbol{\nu}_t(\mathbf{x}) \cdot (\tau - t)$ is a displacement vector¹ measured in the positive direction of time, i.e., from t to τ . Consequently, for linear motion the task is to find the two components of the velocity $\boldsymbol{\nu}$ or displacement \mathbf{d} for each \mathbf{x} . This simple motion model embedding the two-parameter spatial model (3.8.8) has proven to be a powerful motion estimation tool in practice.

A natural extension of the linear model is a quadratic trajectory model, accounting for acceleration of image points, which can be described by

$$\mathbf{x}(\tau) = \mathbf{x}(t) + \boldsymbol{\nu}_t(\mathbf{x}) \cdot (\tau - t) + \frac{1}{2} \cdot \mathbf{a}_t(\mathbf{x}) \cdot (\tau - t)^2. \quad (3.8.11)$$

The model is based on two velocity (linear) variables and two acceleration (quadratic) variables $\mathbf{a} = (a_x, a_y)^T$ thus accounting for second-order effects. This relatively new model has recently been demonstrated to greatly benefit such motion-critical tasks as frame rate conversion [7] due to its improved handling of variable-speed motion present in typical videoconferencing images (e.g., hand gestures, facial expressions).

The above models require two (3.8.10) or four (3.8.11) parameters at each position \mathbf{x} . To reduce the computational burden, parametric (spatial) motion models can be combined with the temporal models above. For example, the affine model (3.8.9) can be used to replace $\boldsymbol{\nu}_t$ (3.8.10) and then applied over a suitable region of support. This approach has been successfully used in various region-based motion estimation algorithms. Recently, a similar parametric extension of the quadratic trajectory model ($\boldsymbol{\nu}_t$ and \mathbf{a}_t replaced by affine expressions) has been proposed [14, Chapter 4] but its practical importance remains to be verified.

¹In the sequel, the dependence of \mathbf{d} on t and τ will be dropped whenever it is clear between what time instants \mathbf{d} applies.

Region of support

The set of points \boldsymbol{x} to which a spatial and temporal motion model applies is called a *region of support*, denoted \mathcal{R} . The selection of a motion model and a region of support is one of the major factors determining the precision of the resulting motion parameter estimates. Usually, for a given motion model, the smaller the region of support \mathcal{R} , the better the approximation of motion. This is due to the fact that over a larger area motion may be more complicated and thus require a more complex model. For example, the translational model (3.8.8) can fairly well describe motion of one car in a highway scene while this very model would be quite poor for the complete image. Typically, the region of support for a motion model belongs to one of the four types listed below. Fig. 3.8.3 shows schematically each type of region.

1. $\mathcal{R} = \text{the whole image}$

A single motion model applies to all image points. This model is suitable for the estimation of camera-induced motion (Section 3.8.5.1) as very few parameters describe the motion of all image points. This is the most constrained model (relatively small number of motion fields can be represented), but with the fewest parameters to estimate.

2. $\mathcal{R} = \text{one pixel}$

This model applies to a single image point (position \boldsymbol{x}). Typically, the translational spatial model (3.8.8) is used jointly with the linear (3.8.10) or quadratic temporal model (3.8.11). This pixel-based or *dense* motion representation is the least constrained one since at least two parameters describe the movement of each image point. Consequently, a very large number of motion fields can be represented by all possible combinations of parameter values, but computational complexity is, in general, high.

3. $\mathcal{R} = \text{rectangular block of pixels}$

This motion model applies to a rectangular (or square) block of image points. In the simplest case the blocks are non-overlapping and their union covers the whole image. A spatially-translational (3.8.8) and temporally-linear (3.8.10) motion of a rectangular block of pixels has proven to be a very powerful model and is used today in all digital video compression standards, i.e., H.261, H.263, MPEG-1 and MPEG-2 (Chapters 6.1, 6.4 and 6.5). It can be also argued that a spatially-translational but temporally-quadratic (3.8.11) motion has been implicitly exploited in the MPEG standards since in the *B* frames two independent motion vectors are used (two non-collinear motion vectors starting at \boldsymbol{x} can describe both velocity and acceleration). Although very successful in hardware implementations, due to its simplicity, the translational model lacks precision for images with rotation, zoom, deformation, and is often replaced by the affine model (3.8.9).

4. $\mathcal{R} = \text{irregularly-shaped region}$

This model applies to all pixels in a region \mathcal{R} of arbitrary shape. The reasoning is that for objects with sufficiently smooth 3-D surface and 3-D motion, the induced 2-D motion can be closely approximated by the affine model (3.8.9) applied linearly over time (3.8.10) to the image area arising from object projection. Thus, regions \mathcal{R} are expected to correspond to object projections. This is the most advanced motion model that has found its way into standards; a square block divided into arbitrarily-shaped parts, each with independent translational motion, is used in the MPEG-4 standard (Chapter 6.5).

Observation models

Since motion is estimated (and observed by the human eye) based on the variations of intensity and/or color, the assumed relationship between motion parameters and image intensity plays a very important

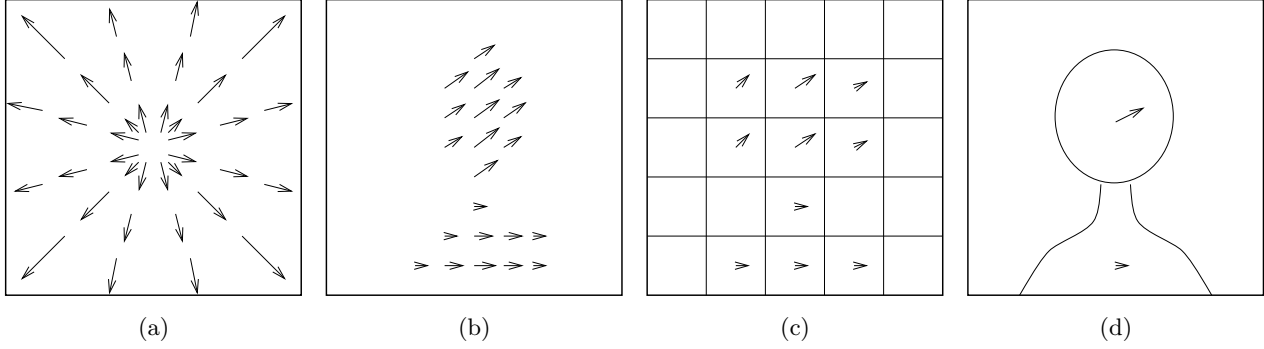


Figure 3.8.3: Schematic representation of motion for the four regions of support \mathcal{R} : (a) whole image, (b) pixel, (c) block and (d) arbitrarily-shaped region. The implicit underlying scene is “head-and-shoulders” as captured by the region-based model in (d).

role. The usual, and reasonable, assumption made is that image intensity remains constant along a motion trajectory, i.e., that objects do not change their brightness and color when they move. For temporally-sampled images, this means that $I_k(\mathbf{x}(t_k))=I_{k-1}(\mathbf{x}(t_{k-1}))$. Using the relationship (3.8.10) with $t = t_{k-1}$ and $\tau = t_k$, and assuming spatial sampling of the images, this condition can be expressed as follows

$$I_k(\mathbf{n}) = I_{k-1}(\mathbf{n} - \mathbf{d}). \quad (3.8.12)$$

The above equation, however, cannot be used directly to solve for \mathbf{d} since in practice it does not hold exactly due to noise q , aliasing, etc., present in the images, i.e., $I_k(\mathbf{n}) = I_{k-1}(\mathbf{n} - \mathbf{d}) + q(\mathbf{n})$. Therefore, \mathbf{d} must be found by minimizing a function of the error between $I_k(\mathbf{n})$ and $I_{k-1}(\mathbf{n} - \mathbf{d})$. Moreover, equation (3.8.12) applied to a single image point \mathbf{n} is insufficient since \mathbf{d} contains at least 2 unknowns. Both issues will be treated in depth in the next section.

Let's consider now the continuous case. Let s be a variable along a motion trajectory. Then, the constant-intensity assumption translates into the following constraint equation

$$\frac{dI}{ds} = 0, \quad (3.8.13)$$

i.e., the directional derivative in the direction of motion is zero. By applying the chain rule, the above equation can be written as the well-known *motion constraint equation* [10]

$$\frac{\partial I}{\partial x}\nu_1 + \frac{\partial I}{\partial y}\nu_2 + \frac{\partial I}{\partial t} = (\nabla I)^T \boldsymbol{\nu} + \frac{\partial I}{\partial t} = 0, \quad (3.8.14)$$

where $\nabla=(\partial/\partial x, \partial/\partial y)^T$ denotes the spatial gradient and $\boldsymbol{\nu}=(\nu_1, \nu_2)^T$ is the velocity to be estimated. The above constraint equation, whether in the above continuous form or as a discrete approximation, has served as the basis for many motion estimation algorithms. Note that, similarly to (3.8.12), equation (3.8.14) applied at single position (x, y) is underconstrained (one equation, two unknowns) and allows to determine the component of velocity $\boldsymbol{\nu}$ in the direction of the image gradient ∇I only [10]. Thus, additional constraints are needed in order to uniquely solve for $\boldsymbol{\nu}$ [10]. Also, equation (3.8.14) does not hold exactly for real images and usually a minimization of a function of $(\nabla I)^T \boldsymbol{\nu} + \frac{\partial I}{\partial t}$ is performed.

Since color is a very important attribute of images, a possible extension of the above models would be to include chromatic image components into the constraint equation. The assumption is that in the areas of uniform intensity but substantial color detail, the inclusion of a color-based constraint could prove beneficial. In such a case, (3.8.13) and (3.8.14) would hold with a multicomponent (vector) function replacing I .

The assumption about intensity constancy is usually only approximately satisfied, but it is particularly violated when scene illumination changes. As an alternative, a constraint based on the spatial gradient's constancy in the direction of motion can be used [2]

$$\frac{d\nabla I}{ds} = \vec{0}. \quad (3.8.15)$$

This equation can be re-written as follows:

$$\begin{bmatrix} \partial^2 I / \partial x^2 & \partial^2 I / \partial x \partial y \\ \partial^2 I / \partial x \partial y & \partial^2 I / \partial y^2 \end{bmatrix} \boldsymbol{\nu} + \frac{\partial(\nabla I)}{\partial t} = \vec{0}. \quad (3.8.16)$$

It relaxes the constant-intensity assumption but requires that the amount of dilation or contraction, and rotation in the image be negligible², a limitation often satisfied in practice. Although both (3.8.15) and (3.8.16) are linear vector equations in 2 unknowns, in practice they do not lend themselves to the direct computation of motion, but need to be further constrained by a motion model. The primary reason for this is that equation (3.8.16) holds only approximately. Furthermore, it is based on second-order image derivatives that are difficult to compute reliably due to the high-pass nature of the operator; usually image smoothing must be performed first.

The constraints discussed above find different applications in practice. A discrete version of the constant-intensity constraint (3.8.14) is often applied in video compression since it yields small motion-compensated prediction error. Although motion can be computed also based on color using a vector equivalent of equation (3.8.14), experience shows that the small gains achieved do not justify the substantial increase in complexity. However, motion estimation from color data is useful in video processing tasks (e.g., motion-compensated filtering, resampling), where motion errors may result in visible distortions. Moreover, the multicomponent (color) constraint is interesting for estimating motion from multiple data sources (e.g., range/intensity data).

3.8.4.2 Estimation criteria

The models discussed need to be incorporated into an estimation criterion that will be subsequently optimized. There is no unique criterion for motion estimation since its choice depends on the task at hand. For example, in compression an average performance (prediction error) of a motion estimator is important, whereas in motion-compensated interpolation the worst case performance (maximum interpolation error) may be of concern. Moreover, the selection of a criterion may be guided by the processor capabilities on which the motion estimation will be implemented.

Pixel-domain criteria

Most of the criteria arising from the constant-intensity assumption (3.8.12) aim at the minimization of a function (e.g., absolute value) of the following error

$$\varepsilon_k(\mathbf{n}) = I_k(\mathbf{n}) - \tilde{I}_k(\mathbf{n}), \quad \forall \mathbf{n} \in \Lambda \quad (3.8.17)$$

where $\tilde{I}_k(\mathbf{n}) = I_{k-1}(\mathbf{n} - \mathbf{d}(\mathbf{n}))$ is called a motion-compensated prediction of $I_k(\mathbf{n})$. Since, in general, \mathbf{d} is real-valued, intensities at positions $\mathbf{n} - \mathbf{d}$ outside of the sampling grid Λ must be recovered by a suitable interpolation. For estimation methods based on matching, C^0 interpolators that assure continuous interpolated intensity (e.g., bilinear) are sufficient, while for methods based on gradient descent C^1 interpolators giving both continuous intensity *and* its derivative are preferable for stability reasons.

²Even when the constant-intensity assumption is valid, the intensity gradient changes its amplitude under dilation or contraction and its direction under rotation.

Motion fields calculated solely by the minimization of the prediction error are sensitive to noise if the number of pixels in \mathcal{R} is not large compared to the number of motion parameters estimated or if the region is poorly textured. However, such a minimization may yield good estimates for parametric motion models with few parameters and reasonable region size.

A common choice for the estimation criterion is the following sum

$$\mathcal{E}(\mathbf{d}) = \sum_{\mathbf{n} \in \mathcal{R}} \Phi(I_k(\mathbf{n}) - \tilde{I}_k(\mathbf{n})) \quad (3.8.18)$$

where Φ is a non-negative real-valued function. The often-used quadratic function $\Phi(\varepsilon) = \varepsilon^2$ is not a good choice since a single large error ε (an outlier) overcontributes to \mathcal{E} and biases the estimate of \mathbf{d} . A more robust function is the absolute value $\Phi(\varepsilon) = \alpha|\varepsilon|$ since the cost grows linearly with error (Fig. 3.8.4.a.). Since it does not require multiplications, it is the criterion of choice in practical video encoders today. An even more robust criterion is based on the Lorentzian function $\Phi(\varepsilon) = \log(1 + \varepsilon^2/2\omega^2)$ that grows slower than $|x|$ for large errors. The growth of the cost for increasing errors ε is adjusted by the parameter ω (a Lorentzian function for two different values of ω is shown in Fig. 3.8.4.a.).

Since for matching algorithms the continuity of Φ is not important (no gradient computations), non-continuous functions based on the concept of the truncated quadratic:

$$\Phi_{tq}(\varepsilon, \theta, \beta) = \begin{cases} \varepsilon^2 & |\varepsilon| < \theta, \\ \beta & \text{otherwise,} \end{cases} \quad (3.8.19)$$

are often used (Fig. 3.8.4.b). If $\beta = \theta^2$, the usual truncated quadratic results, fixing the cost of outliers at θ^2 . An alternative is to set $\beta = 0$ with the consequence that the outliers have zero cost and do not contribute to the overall criterion \mathcal{E} . In other words, the criterion is defined only for non-outlier pixels, and therefore the estimate of \mathbf{d} will be computed solely on the basis of reliable pixels.

The similarity between $I_k(\mathbf{n})$ and its prediction $\tilde{I}_k(\mathbf{n})$ can be also measured by a cross-correlation function:

$$\mathcal{C}(\mathbf{d}) = \sum_{\mathbf{n}} I_k(\mathbf{n}) I_{k-1}(\mathbf{n} - \mathbf{d}(\mathbf{n})). \quad (3.8.20)$$

Although more complex computationally than the absolute value criterion due to the multiplications, this criterion is an interesting and practical alternative to the prediction error-based criteria (Section 3.8.5.3). Note that a cross-correlation criterion requires maximization unlike the prediction-based criteria.

For a detailed discussion of robust estimation criteria in the context of motion estimation the reader is referred to the literature (e.g., [5] and references therein).

Frequency-domain criteria

Although the frequency-domain criteria are less used in practice today than the space/time-domain methods, they form an important alternative. Let $\hat{I}_k(\mathbf{u}) = \mathcal{F}[I_k(\mathbf{n})]$ be a spatial (2-D) discrete Fourier transform (DFT) of the intensity signal $I_k(\mathbf{n})$, where $\mathbf{u} = (u, v)^T$ is a 2-D discrete frequency (see Chapter 2.3). Suppose that the image I_{k-1} has been uniformly shifted to create the image I_k , i.e., that $I_k(\mathbf{n}) = I_{k-1}(\mathbf{n} - \mathbf{z})$. This means that only translational global motion exists in the image and all boundary effects are neglected. Then, by the shift property of the Fourier transform

$$\mathcal{F}[I_{k-1}(\mathbf{n} - \mathbf{z})] = \hat{I}_{k-1}(\mathbf{u}) e^{-j2\pi\mathbf{u}^T \mathbf{z}}, \quad (3.8.21)$$

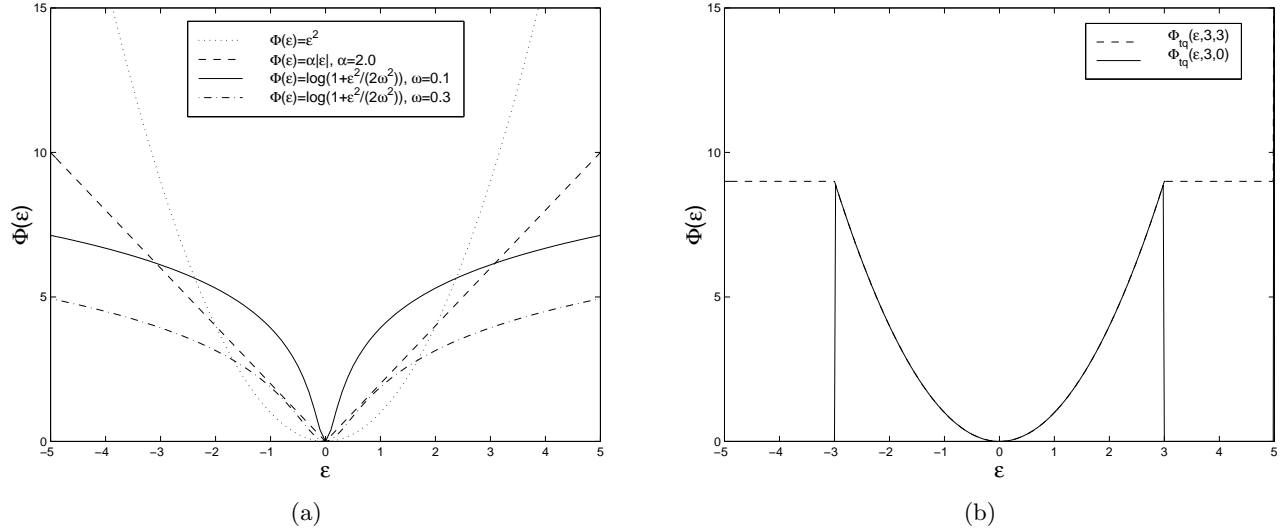


Figure 3.8.4: Comparison of estimation criteria: (a) quadratic, absolute value, Lorentzian functions (two different ω 's); and (b) truncated-quadratic functions.

where \mathbf{u}^T denotes a transposed vector \mathbf{u} . Since the amplitudes of both Fourier transforms are independent of \mathbf{z} while the argument difference

$$\arg\{\mathcal{F}[I_k(\mathbf{n})]\} - \arg\{\mathcal{F}[I_{k-1}(\mathbf{n})]\} = -2\pi\mathbf{u}^T\mathbf{z}$$

depends linearly on \mathbf{z} , the global motion can be recovered by evaluating the phase difference over a number of frequencies and solving the resulting over-constrained system of linear equations. In practice, this method will work only for single objects moving across a uniform background. Moreover, the positions of image points to which the estimated displacement \mathbf{z} applies are not known; this assignment must be performed in some other way. Also, care must be taken of the non-uniqueness of the Fourier phase function which is periodic.

A Fourier-domain representation is particularly interesting for the cross-correlation criterion (3.8.20). Based on the Fourier transform properties (Chapter 2.3) and under the assumption that the intensity function I is real-valued, it is easy to show that:

$$\mathcal{F}[\mathcal{C}(\mathbf{d})] = \mathcal{F}\left[\sum_{\mathbf{n}} I_k(\mathbf{n})I_{k-1}(\mathbf{n} - \mathbf{d})\right] = \widehat{I}_k(\mathbf{u})\widehat{I}_{k-1}^*(\mathbf{u}) \quad (3.8.22)$$

where the transform is applied in spatial coordinates only and \widehat{I}^* is the complex conjugate of \widehat{I} . This equation expresses spatial cross-correlation in the Fourier domain, where it can be efficiently evaluated using the DFT.

Regularization

The criteria described thus far deal with the underconstrained nature of equation (3.8.14) by applying the motion measurement to either a region, such as a block of pixels, or to the whole image (frequency-domain criteria). In consequence, resolution of the computed motion may suffer.

To maintain motion resolution at the level of original images, the pixel-wise motion constraint equation (3.8.14) can be used but, to address its underconstrained nature, needs to be combined with another constraint. In typical real-world images the moving objects are close to being rigid. Upon projection onto the image plane this induces very similar motion of neighboring image points within the object's projection area. In other words, the motion field is locally smooth. Therefore, a motion field

$\boldsymbol{\nu}_t$ must be sought that satisfies the motion constraint (3.8.14) as close as possible and simultaneously is as smooth as possible. Since gradient is a good measure of local smoothness, this may be achieved by minimizing the following criterion [10]:

$$\mathcal{E}(\boldsymbol{\nu}) = \int_{\mathcal{D}} \left(\nabla^T I(\mathbf{x}) \boldsymbol{\nu}(\mathbf{x}) + \frac{\partial I(\mathbf{x})}{\partial t} \right)^2 + \lambda (\|\nabla(\nu_1(\mathbf{x}))\|^2 + \|\nabla(\nu_2(\mathbf{x}))\|^2) d\mathbf{x} \quad (3.8.23)$$

where \mathcal{D} is the domain of the image. This formulation is often referred to as *regularization* [2] (see also the discussion of regularized image recovery in Chapter 3.11). Note that the smoothness constraint may be also viewed as an alternative spatial motion model to those described in Section 3.8.4.1.

Bayesian criteria

Bayesian criteria form a very powerful probabilistic alternative to the deterministic criteria described thus far. If motion field \mathbf{d}_k is a realization of a vector random field \mathbf{D}_k with a given *a posteriori* probability distribution, and image I_k is a realization of a scalar random field \mathcal{I}_k , then the MAP estimate of \mathbf{d}_k (Section 3.8.2.3) can be computed as follows [12]:

$$\begin{aligned} \hat{\mathbf{d}}_k &= \arg \max_{\mathbf{d}} P(\mathbf{D}_k = \mathbf{d}_k | \mathcal{I}_k = I_k; I_{k-1}) \\ &= \arg \max_{\mathbf{d}} P(\mathcal{I}_k = I_k | \mathbf{D}_k = \mathbf{d}_k; I_{k-1}) \cdot P(\mathbf{D}_k = \mathbf{d}_k; I_{k-1}). \end{aligned} \quad (3.8.24)$$

In this notation, the semicolon indicates that subsequent variables are only deterministic parameters. The first (conditional) probability distribution denotes the likelihood of image I_k given displacement field \mathbf{d}_k and the previous image I_{k-1} , and therefore is closely related to the observation model. In other words, this term quantifies how well a motion field \mathbf{d}_k explains the change between the two images. The second probability $P(\mathbf{D}_k = \mathbf{d}_k; I_{k-1})$ describes the prior knowledge about the random field \mathbf{D}_k , such as its spatial smoothness, and therefore can be thought of as a motion model. It becomes particularly interesting when \mathbf{D}_k is a MRF. By maximizing the product of the likelihood and the prior probabilities one attempts to strike a balance between motion fields that give a small prediction error and those that are smooth. It will be shown in Section 3.8.5.5 that maximization (3.8.24) is equivalent to an energy minimization.

3.8.4.3 Search strategies

Once models have been identified and incorporated into an estimation criterion, the last step is to develop an efficient (complexity) and effective (solution quality) strategy for finding the estimates of motion parameters.

For a small number of motion parameters and a small state space for each of them, the most common search strategy when minimizing a prediction error, like (3.8.17), is *matching*. In this approach, motion-compensated predictions $\tilde{I}_k(\mathbf{n}) = I_{k-1}(\mathbf{n} - \mathbf{d}(\mathbf{n}))$ for various motion candidates \mathbf{d} are compared (matched) with the original images $I_k(\mathbf{n})$ within the region of support of the motion model (pixel, block, etc.). The candidate yielding the best match for a given criterion becomes the optimal estimate. For small state spaces, as is the case in block-constant motion models used in today's video coding standards, the full state space of each motion vector can be examined (exhaustive search) but partial search often gives almost as good results (Section 3.8.5.2).

As opposed to matching, *gradient-based techniques* require an estimation criterion \mathcal{E} that is differentiable. Since this criterion depends on motion parameters *via* the image function, as in $I_{k-1}(\mathbf{n} - \mathbf{d}(\mathbf{n}))$, to avoid non-linear optimization I is usually linearized using a Taylor expansion with respect to $\mathbf{d}(\mathbf{n})$. Due to the Taylor approximation, the model is applicable only in a small vicinity of the initial \mathbf{d} . Since initial motion is usually assumed to be zero, it comes as no surprise that gradient-based estimation

is reported to yield accurate estimates only in regions of small motion; the approach fails if motion is large. This deficiency is usually compensated for by a hierarchical or multiresolution implementation [17, Chapter 1], (Chapter 4.1). An example of hierarchical gradient-based method is reported in Section 3.8.5.1.

For motion fields using a spatial non-causal model, such as based on a MRF, simultaneous optimization of thousands of parameters may be computationally prohibitive³. Therefore, *relaxation techniques* are usually employed to construct a series of estimates such that consecutive estimates differ in one variable at most. In case of estimating a motion field \mathbf{d} a series of motion fields $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \dots$ is constructed so that any two consecutive estimates $\mathbf{d}^{(k-1)}, \mathbf{d}^{(k)}$ differ at most at a single site \mathbf{n} . At each step of the relaxation procedure the motion vector at a single site is computed; vectors at other sites remain unchanged. Repeating this process results in a propagation of motion properties, such as smoothness, that are embedded in the estimation criterion. Relaxation techniques are most often used in dense motion field estimation, but they equally apply to block-based methods.

In *deterministic relaxation*, such as Jacobi or Gauss-Seidel, each motion vector is computed with probability 1, i.e., there is no uncertainty in the computation process. For example, a new local estimate is computed by minimizing the given criterion; variables are updated one after another and the criterion is monotonically improved step by step. Deterministic relaxation techniques are capable of correcting spurious motion vectors in the initial state $\mathbf{d}^{(0)}$ but they often get trapped in a local optimum near $\mathbf{d}^{(0)}$. Therefore, the availability of a good initial state is crucial.

The *highest confidence first* (HCF) algorithm [8] is an interesting variant of deterministic relaxation that is insensitive to the initial state. The distinguishing characteristic of the method is its site visiting schedule that is not fixed but driven by the input data. Without going into the details, the HCF algorithm initially selects motion vectors that have the largest potential for reducing the estimation criterion \mathcal{E} . Usually, these are vectors in highly-textured parts of an image. Later, the algorithm includes more and more motion vectors from low-texture areas, thus building on the neighborhood information of sites already estimated. By the algorithm's construction, the final estimate is independent of the initial state. The HCF is capable of finding close to optimal MAP estimates at a fraction of the computational cost of the globally-optimal methods.

A deterministic algorithm specifically developed to deal with MRF formulations is called *iterated conditional modes* (ICM) [3]. Although it does not maximize the *a posteriori* probability, it finds reasonably close approximations. The method is based on the division of sites of a random field into N sets such that each random variable associated with a site is independent of other random variables in the same set. The number of sets and their geometry depend on the selected cliques of the MRF. For example, for the first-order neighborhood system (Section 3.8.2.2), $N=2$ and the two sets look like a chess board. First, all the sites of one set are updated to find the optimal solution. Then, the sites of the other set are examined with the state of the first set already known. The procedure is repeated until a convergence criterion is met. The method converges quickly but does not lead to as good solutions as the HCF approach.

The dependence on a good initial state is eliminated in *stochastic relaxation*. In contrast to the deterministic relaxation, the motion vector \mathbf{v} under consideration is selected randomly (both its location \mathbf{x} and parameters \mathbf{b}) thus allowing (with a small probability) a momentary deterioration of the criterion [12]. In the context of minimization, such as in *simulated annealing* [9], this allows the algorithm to “climb” out of local minima and eventually reach the global minimum. Stochastic relaxation methods, although easy to implement and capable of finding excellent solutions, are very slow in convergence.

³There exist methods based on causal motion models that are computationally inexpensive, e.g., pel-recursive motion estimation, but their accuracy is usually lower than that of methods based on non-causal motion models.

3.8.5 Practical motion estimation algorithms

3.8.5.1 Global motion estimation

As discussed in Section 3.8.4.1 camera motion induces motion of all image points and therefore is often an obstacle to solving various video processing problems. For example, to detect motion in images obtained by a mobile camera, camera motion must be compensated first [14, Chapter 8]. Global motion compensation (GMC) plays also an important role in video compression since only a few motion parameters are sufficient to greatly reduce the prediction error when images to be encoded are acquired, for example, by a panning camera. GMC has been included in version 2 of the MPEG-4 video compression standard (Chapter 6.5).

Since camera motion is limited to translation and rotation, and affects all image points, a spatially-parametric (e.g., affine (3.8.9)) and temporally-linear (3.8.10) motion model supported on the whole image is appropriate. Under the constant-intensity observation model (3.8.12), the pixel-based quadratic criterion (3.8.18) leads to the following minimization:

$$\min_{\mathbf{b}} \mathcal{E}(\boldsymbol{\nu}), \quad \mathcal{E}(\boldsymbol{\nu}) = \sum_{\mathbf{n}} \varepsilon^2(\mathbf{n}), \quad \varepsilon(\mathbf{n}) = I_k(\mathbf{n}) - I_{k-1}(\mathbf{n} - \boldsymbol{\nu}(\mathbf{n}) \cdot (t_k - t_{k-1})), \quad (3.8.25)$$

where the dependence of $\boldsymbol{\nu}$ on \mathbf{b} is implicit (3.8.9) and $t_k - t_{k-1}$ is usually assumed to equal 1. To perform the above minimization, gradient descent can be used. However, since this method gets easily trapped in local minima, an initial search for approximate translation components b_1 and b_2 (3.8.9), that can be quite large, needs to be performed. This search can be executed, for example, using the three-step block matching (Section 3.8.5.2).

Since the dependence of the cost function \mathcal{E} on \mathbf{b} is nonlinear, an iterative procedure is typically used:

$$\mathbf{b}^{n+1} = \mathbf{b}^n + \mathbf{H}^{-1} \mathbf{c}$$

where \mathbf{b}^n is the parameter vector \mathbf{b} at iteration n , \mathbf{H} is a $K \times K$ matrix equal to 1/2 of the Hessian matrix of \mathcal{E} (i.e., matrix with elements $\partial^2 \mathcal{E} / \partial b_k \partial b_l$), \mathbf{c} is a K -dimensional vector equal to -1/2 of $\nabla \mathcal{E}$, and K is the number of parameters in the motion model (6 for affine). The above equation can be equivalently written as $\sum_l H_{kl} \Delta b_l = c_k$, where $\Delta \mathbf{b} = \mathbf{b}^{n+1} - \mathbf{b}^n$ and

$$\begin{aligned} H_{kl} &= \frac{1}{2} \sum_{\mathbf{n}} \frac{\partial^2 \varepsilon^2(\mathbf{n})}{\partial b_k \partial b_l} = \sum_{\mathbf{n}} \left(\frac{\partial \varepsilon(\mathbf{n})}{\partial b_k} \frac{\partial \varepsilon(\mathbf{n})}{\partial b_l} + \varepsilon(\mathbf{n}) \frac{\partial^2 \varepsilon(\mathbf{n})}{\partial b_k \partial b_l} \right) \approx \sum_{\mathbf{n}} \left(\frac{\partial \varepsilon(\mathbf{n})}{\partial b_k} \frac{\partial \varepsilon(\mathbf{n})}{\partial b_l} \right), \\ c_k &= -\frac{1}{2} \sum_{\mathbf{n}} \frac{\partial \varepsilon^2(\mathbf{n})}{\partial b_k} = -\sum_{\mathbf{n}} \varepsilon(\mathbf{n}) \frac{\partial \varepsilon(\mathbf{n})}{\partial b_k}. \end{aligned}$$

The approximation above is due to dropping the second-order derivatives (see [16, page 683] for justification).

In order to handle large velocities and to speed up computations, the method needs to be implemented hierarchically. Thus, an image pyramid is built with spatial pre-filtering and sub-sampling applied between each two levels. The computation starts at the top level of the pyramid (lowest resolution) with b_1 and b_2 estimated in the initial step and the other parameters set to zero. Then, gradient descent is performed by solving for $\Delta \mathbf{b}$, e.g., using singular value decomposition, and updating $\mathbf{b}^{n+1} = \mathbf{b}^n + \Delta \mathbf{b}$ until a convergence criterion is met. The resulting motion parameters are projected onto a lower level of the pyramid⁴ and the gradient descent is repeated. This cycle is repeated until the bottom of the pyramid is reached.

⁴The projection is performed by scaling the translation parameters b_1 and b_2 by 2 and leaving the other 4 parameters unchanged.

Since the global motion model applies to all image points, it cannot account for local motion. Thus, points moving independently of the global motion may generate large errors $\varepsilon(\mathbf{n})$ and thus bias an estimate of the global motion parameters. The corresponding pixels are called *outliers* and, ideally, should be eliminated from the minimization (3.8.25). This can be achieved by using a robust criterion (Fig. 3.8.4) instead of the quadratic. For example, a Lorentzian function or a truncated quadratic can be used, but both provide a non-zero cost for outliers. This reduces the impact of outliers on the estimation but does not eliminate it completely. To exclude the impact of outliers altogether, a modified truncated quadratic should be used such as $\Phi_{tg}(\varepsilon, \theta, 0)$ defined in (3.8.19). This criterion effectively limits the summation in (3.8.25) to the non-outlier pixels and is used only in the gradient descent part of the algorithm. The threshold θ can be fixed or it can be made adaptive, e.g., by limiting the false alarm rate.

Fig. 3.8.5 shows outlier pixels for two images “Foreman” and “Coastguard” declared using the above method based on the 8-parameter perspective motion model [22, Chapter 6]. Note the clear identification of outliers in the moving head, on the boats and in the water. The outliers tend to appear at intensity transitions since it is there that any inaccuracy in global motion caused by a local (inconsistent) motion will induce large error ε ; in uniform areas of local motion the error ε remains small. By excluding the outliers from the estimation, the accuracy of computed motion parameters is improved. Since the true camera motion is not known for these two sequences, the improvement was measured in the context of the GMC mode of MPEG-4 compression⁵. In comparison with non-robust global motion estimation ($\Phi_{tg}(\varepsilon, \infty, \cdot)$), the robust method presented resulted in bit rate reduction of 8% and 15% for “Foreman” and “Coastguard”, respectively [13].

3.8.5.2 Block matching

Block matching is the simplest algorithm for the estimation of local motion. It uses a spatially-constant (3.8.8) and temporally-linear (3.8.10) motion model over a rectangular region of support. Although, as explained in Section 3.8.4.1, the translational 2-D motion is only valid for the orthographic projection and 3-D object translation, this model applied locally to a small block of pixels is quite accurate for a large variety of 3-D motions. It has proven accurate enough to serve as a basis for most of the practical motion estimation algorithms used today. Due to its simplicity and regularity (the same operations are performed for each block of the image), block matching can be relatively easily implemented in VLSI. Today, block matching is the only motion estimation algorithm massively implemented in VLSI and used for encoding within *all* video compression standards (see Chapters 6.4 and 6.5).

In video compression, motion vectors \mathbf{d} are used to eliminate temporal video redundancy *via* motion-compensated prediction (3.8.17). Hence, the goal is to achieve as low prediction error $\varepsilon_k(\mathbf{n})$ as possible, which is equivalent to the constant-intensity observation model. By applying this model within a pixel-based criterion, the method can be described by the following minimization:

$$\min_{\mathbf{d}_m \in \mathcal{P}} \mathcal{E}(\mathbf{d}_m), \quad \mathcal{E}(\mathbf{d}_m) = \sum_{\mathbf{n} \in \mathcal{B}_m} \Phi(I_k(\mathbf{n}) - I_{k-1}(\mathbf{n} - \mathbf{d}_m)) \quad \forall \mathbf{m} \quad (3.8.26)$$

where \mathcal{P} is the search area to which \mathbf{d}_m belongs, defined as follows:

$$\mathcal{P} = \{\mathbf{n} = (n_1, n_2) : -P \leq n_1 \leq P, -P \leq n_2 \leq P\},$$

and \mathcal{B}_m is an $M \times N$ block of pixels with the top-left corner coordinate at $\mathbf{m} = (m_1, m_2)$. The goal is to find the best, in the sense of the criterion Φ , displacement vector \mathbf{d}_m for each block \mathcal{B}_m . This is illustrated graphically in Fig. 3.8.6(a); a block is sought within image I_{k-1} that best matches the current block in I_k .

⁵MPEG-4 encoder (version 2) can send parameters of global motion for each frame. Consequently, for each macroblock it can make a decision as to whether to perform the temporal prediction based on the global motion parameters or the local macroblock motion. The benefit of GMC is that only few motion parameters (e.g., 8) are sent for the whole frame. The GMC mode is beneficial for sequences with camera motion or zoom.

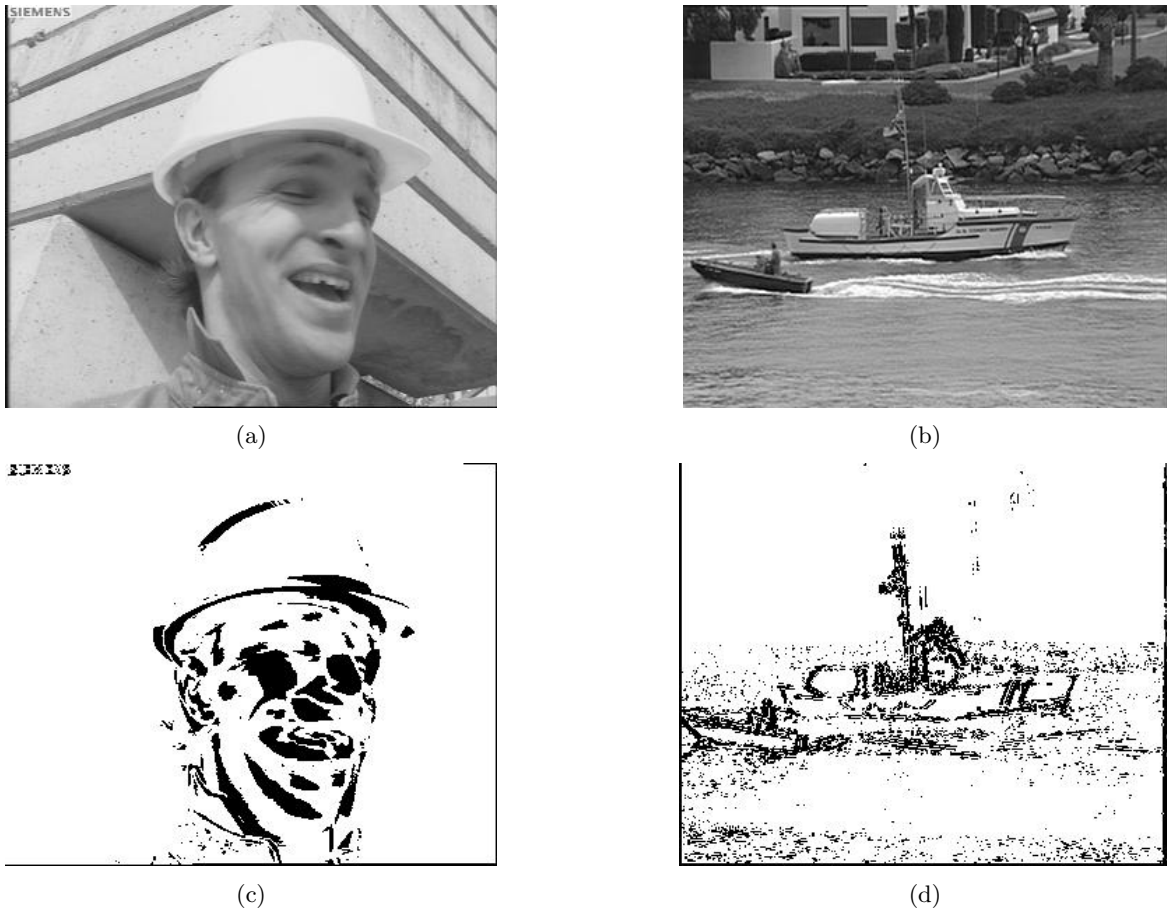


Figure 3.8.5: (a-b) Original images from CIF sequences “Foreman” and “Coastguard”, and (c-d) pixels declared as outliers (black) according to a global motion estimate.

Estimation criterion

Although an average error is used in (3.8.26), other measures are possible, such as a maximum error (min-max estimation). To fully define the estimation criterion, the function Φ must be established. Originally, $\Phi(x) = x^2$ was often used in block matching, but it was soon replaced by an absolute error criterion $\Phi(x) = |x|$ for its simplicity (no multiplications) and robustness in the presence of outliers. Other improved criteria have been proposed, such as based on median of squared errors, however their computational complexity is significantly higher.

Also, simplified criteria have been proposed to speed up the computations, for example, based on adaptive quantization to 2 bits or on pixel sub-sampling [4]. Usually, a simplification of the original criterion Φ leads to a suboptimal performance. However, with an adaptive adjustment of the criterion’s parameters (e.g., quantization levels, decimation patterns) a close-to-optimal performance can be achieved at significantly reduced complexity.

Search methods

Exhaustive search for $d_m \in \mathcal{P}$ that gives the lowest error \mathcal{E} is computationally costly. An “intelligent” search, whereby only the more likely candidates from \mathcal{P} are evaluated, usually results in substantial computational savings. One popular technique for reducing the number of candidates is the *logarithmic search*. Assuming that $P = 2^k - 1$ and denoting $P_l = (P + 1)/2^l$, where k and l are integers, the new

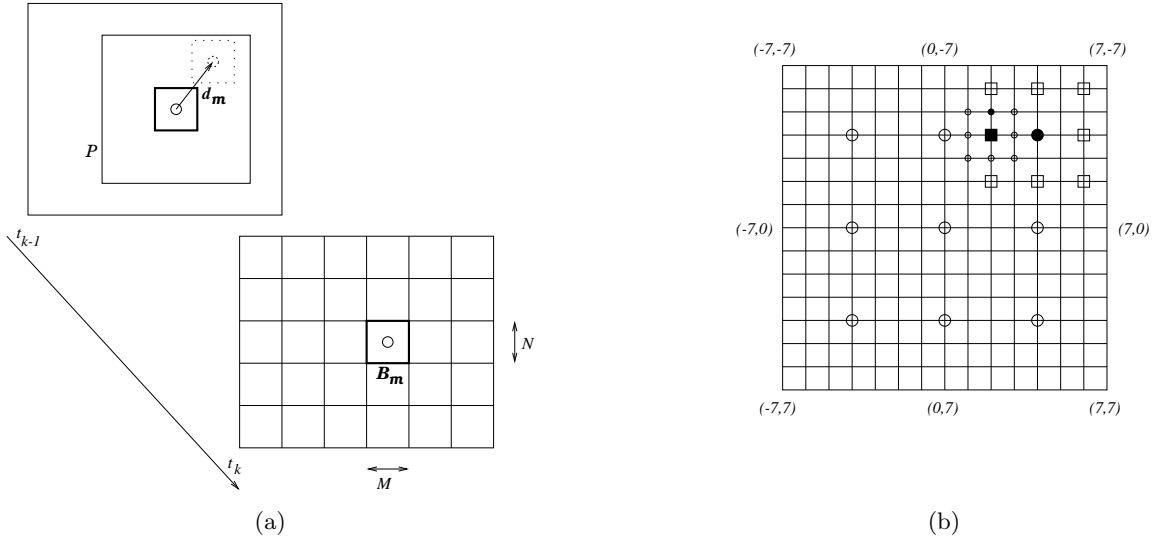


Figure 3.8.6: (a) Block matching between block \mathcal{B}_m at time t_k (current image) and all possible blocks in the search area \mathcal{P} at time t_{k-1} . (b) Three-step search method. Large circles denote level 1 (\mathcal{P}_1), squares denote level 2 (\mathcal{P}_2) and small circles denote level 3 (\mathcal{P}_3). The filled-in elements denote the best match found at each level. The final vector is $(2, 5)$.

reduced-size search area is established as follows:

$$\mathcal{P}_l = \{\mathbf{n} : \mathbf{n} = (\pm P_l, \pm P_l) \text{ or } \mathbf{n} = (\pm P_l, 0) \text{ or } \mathbf{n} = (0, \pm P_l) \text{ or } \mathbf{n} = (0, 0)\}$$

i.e., \mathcal{P}_l is reduced to the vertices, midway points between vertices and the central point of the half-sized original rectangle \mathcal{P} . For example, for $P = 7$, \mathcal{P}_1 consists of the following candidates: $(-4, -4)$, $(-4, 4)$, $(4, -4)$, $(4, 4)$, $(-4, 0)$, $(4, 0)$, $(0, -4)$, $(0, 4)$, $(0, 0)$. The search starts with the candidates from \mathcal{P}_1 . Once the best match is found, the new search area \mathcal{P}_2 is centered around this match and the procedure is repeated for candidates from \mathcal{P}_2 . Note that the error \mathcal{E} does not have to be evaluated for the $(0, 0)$ candidate since it had been evaluated at the previous level. The procedure is continued with subsequently reduced search spaces. Since typically only 3 levels ($l = 1, 2, 3$) are used, such a method is often referred to as the *three-step search* (Fig. 3.8.6(b)).

In the logarithmic search above, at each step a 2-D search is performed. An alternative approach is to perform 1-D searches only, usually in orthogonal directions. Examples of block matching algorithms based on 1-D search methods are:

1. *one-at-a-time search* [19]: In this method, first a minimum of \mathcal{E} is sought in one, for example horizontal, direction. Then, given the horizontal estimate, a vertical search is performed. Subsequently, a horizontal search is performed given the previous vertical estimate, and so on. In the original proposal, 1-D minima closest to the origin were examined only, but later a 1-D full-search was used to avoid the problem of getting trapped close to the origin. Note that the searches are not independent since each relies on the result of the previous one.
2. *parallel hierarchical one-dimensional search* [4]: This method also performs 1-D searches in orthogonal directions (usually horizontal and vertical) but independently of each other, i.e., the horizontal search is performed simultaneously with the vertical search since it does not depend on the outcome of the latter. In addition, the 1-D search is implemented hierarchically. First, every K -th location from \mathcal{P} is taken as a candidate for a 1-D search. Once the minima in both directions are identified, new 1-D searches begin with every $K/2$ -th location from \mathcal{P} , and so on. Typically, horizontal and vertical searches using every 8-th, 4-th, 2-nd and finally every pixel (within the limits of \mathcal{P}) are performed.

A remark is in order at this point. All the fast search methods are based on the assumption that the error \mathcal{E} has a single minimum for all $\mathbf{d}_m \in \mathcal{P}$, or in other words, that \mathcal{E} increases monotonically when moving away from the best-match position. In practice this is rarely true since \mathcal{E} depends on \mathbf{d}_m via the intensity I , which can be arbitrary. Therefore, multiple local minima often exist within \mathcal{P} and a fast search method can be easily trapped in anyone of them whereas an exhaustive search will always find the “deepest” minimum. This is not a very serious problem in video coding since a sub-optimal motion estimate translates into an increased prediction error (3.8.17) that will be entropy coded and at most will result in a rate increase. It is a serious problem, however, in video processing where *true* motion is sought and any motion errors may result in uncorrectable distortions. A good review of block matching algorithms can be found in [4].

3.8.5.3 Phase correlation

As discussed above, block matching can precisely estimate local displacement but must examine all possible candidates (exhaustive search). At the same time methods based on the frequency-domain criteria (Section 3.8.4.2) are capable of identifying global motion but cannot localize it in the space/time domain. By combining the two approaches, the *phase correlation* method [21] is able to exploit advantages of both approaches. First, likely candidates are computed using a frequency-domain approach and then they are assigned a spatial location by local block matching.

Recall the cross-correlation criterion $\mathcal{C}(\mathbf{d})$ expressed in the Fourier domain (3.8.22). By normalizing $\mathcal{F}[\mathcal{C}(\mathbf{d})]$ and taking the inverse transform one obtains:

$$\Psi_{k-1,k}(\mathbf{n}) = \mathcal{F}^{-1} \left\{ \frac{\widehat{I}_k(\mathbf{u})\widehat{I}_{k-1}^*(\mathbf{u})}{|\widehat{I}_k(\mathbf{u})\widehat{I}_{k-1}^*(\mathbf{u})|} \right\}. \quad (3.8.27)$$

$\Psi_{k-1,k}(\mathbf{n})$ is a normalized correlation computed between two images I_k and I_{k-1} . In the special case of a global translation ($I_k(\mathbf{n}) = I_{k-1}(\mathbf{n} - \mathbf{z})$), by using the transform (3.8.21) it can be easily shown that the correlation surface becomes a Kronecker delta function ($\delta(\mathbf{x})$ equals 0 for $\mathbf{x} \neq 0$ and 1 for $\mathbf{x} = 0$):

$$\Psi_{k-1,k}(\mathbf{n})|_{I_k(\mathbf{n})=I_{k-1}(\mathbf{n}-\mathbf{z})} = \mathcal{F}\{e^{-j2\pi\mathbf{u}\cdot\mathbf{z}}\} = \delta(\mathbf{n} - \mathbf{z}).$$

In practice, when neither the global translation nor intensity constancy hold, $\Psi_{k-1,k}$ is a surface with numerous peaks. These peaks correspond to dominant displacements between I_{k-1} and I_k , and, if identified, are very good candidates for fine-tuning by, for example, block matching. Note that no explicit motion model has been used thus far, while the observation model, as usual, is that of constant intensity and the estimation criterion is the cross-correlation function. In practice, the method can be implemented as follows [21]:

1. divide I_{k-1} and I_k into large blocks, e.g., 64×64 (motion range of ± 32 pixels), and take a fast Fourier transform (FFT) of each block,
2. compute $\Psi_{k-1,k}$ using same-position blocks in I_{k-1} and I_k ,
3. take inverse FFT of $\Psi_{k-1,k}$ and identify most dominant peaks,
4. use the coordinates of the peaks as the candidate vectors for block matching of 16×16 blocks.

The phase correlation can also initialize pixel-based based estimation. Moreover, the correlation over a large area (64×64) permits the recovery of sub-pixel displacements by interpolating the correlation surface. Note that since the discrete Fourier transform (implemented via FFT) assumes signal periodicity, intensity discontinuities between left and right, and between top and bottom block boundaries may introduce spurious peaks.

The phase correlation method is basically an efficient maximization of a correlation-based error criterion. The shape of the maxima of the correlation surface is weakly dependent on the image content and the measurement of their locations is relatively independent of illumination changes. This is due, predominantly, to the normalization in (3.8.27). However, rotations and zooms cannot be easily handled since the peaks in $\Psi_{k-1,k}$ are hard to distinguish due to spatial smoothness of the corresponding motion fields.

3.8.5.4 Optical flow via regularization

Recall the regularized estimation criterion (3.8.23). It uses a translational/linear motion model at each pixel under the constant-intensity observation model and a quadratic error criterion.

To find the functions ν_1 and ν_2 , implicitly dependent on \mathbf{x} , the functional needs to be minimized, which is a problem in the calculus of variations. The Euler-Lagrange equations yield [10]:

$$\begin{aligned}\lambda \nabla^2 \nu_1 &= \left(\frac{\partial I}{\partial x} \nu_1 + \frac{\partial I}{\partial y} \nu_2 + \frac{\partial I}{\partial t} \right) \frac{\partial I}{\partial x}, \\ \lambda \nabla^2 \nu_2 &= \left(\frac{\partial I}{\partial x} \nu_1 + \frac{\partial I}{\partial y} \nu_2 + \frac{\partial I}{\partial t} \right) \frac{\partial I}{\partial y},\end{aligned}$$

where $\nabla^2 = \partial^2/\partial x^2 + \partial^2/\partial y^2$ is the Laplacian operator. This pair of elliptic partial differential equations can be solved iteratively using finite-difference or finite-element discretization (see Chapter 3.11 for other examples of regularization).

An alternative is to formulate the problem directly in the discrete domain. Then, the integral is replaced by a summation while the derivatives are replaced by finite differences. In [10], for example, an average of first-order differences computed over a $2 \times 2 \times 2$ cube was used. By differentiating this discrete cost function, a system of equations can be computed and subsequently solved by Jacobi or Gauss-Seidel relaxation. This discrete approach to regularization is a special case of the MAP estimation presented next.

3.8.5.5 MAP estimation of dense motion

The MAP formulation (3.8.24) is very general and requires further assumptions. The likelihood relates one image to another *via* \mathbf{d}_k . Since $I_k(\mathbf{n}) = I_{k-1}(\mathbf{n} - \mathbf{d}(\mathbf{n})) + \varepsilon_k(\mathbf{n})$, the characteristics of this likelihood reside in ε_k . It is clear that for good displacement estimates ε_k should behave like a noise term. It can be shown that the statistics of ε_k are reasonably close to those of a zero-mean Gaussian distribution, although a generalized Gaussian is a better fit [20]. Therefore, assuming no correlation among $\varepsilon_k(\mathbf{n})$ for different \mathbf{n} , $P(\mathcal{I}_k = I_k | \mathbf{D}_k = \mathbf{d}_k; I_{k-1})$ can be fairly accurately modeled by a product of zero-mean Gaussian distributions.

The prior probability is particularly flexible when \mathbf{D}_k is assumed to be a MRF. Then, $P(\mathbf{D}_k = \mathbf{d}_k; I_{k-1})$ is a Gibbs distribution (Section 3.8.2.2) uniquely specified by cliques and a potential function. For example, for two-element cliques $\{\mathbf{n}, \mathbf{l}\}$ the smoothness of \mathbf{D}_k can be expressed as follows:

$$V_s(\mathbf{d}(\mathbf{n}), \mathbf{d}(\mathbf{l})) = \|\mathbf{d}(\mathbf{n}) - \mathbf{d}(\mathbf{l})\|^2, \quad \forall \{\mathbf{n}, \mathbf{l}\} \in C.$$

Clearly, for similar $\mathbf{d}(\mathbf{n})$ and $\mathbf{d}(\mathbf{l})$ the potential V_s is small and thus the prior probability is high, whereas for dissimilar vectors this probability is small.

Since both likelihood and prior probability distributions are exponential in this case, the MAP estimation (3.8.24) can be re-written as energy minimization:

$$\hat{\mathbf{d}}_k = \arg \min_{\mathbf{d}} \left(\frac{1}{2\sigma^2} \sum_{\mathbf{n}} (I_k(\mathbf{n}) - I_{k-1}(\mathbf{n} - \mathbf{d}(\mathbf{n})))^2 + \sum_{\mathbf{l} \in \eta(\mathbf{n})} \|\mathbf{d}(\mathbf{n}) - \mathbf{d}(\mathbf{l})\|^2 \right). \quad (3.8.28)$$

The above energy can be minimized in various ways. To attain the global minimum, simulated annealing (Section 3.8.4.3) should be used. Given sufficiently many iterations the method is theoretically capable of finding the global minimum, however at considerable computational cost. On the other hand the method is easy to implement [12]. A faster alternative is the deterministic ICM method that does not find a true MAP estimate, although usually finds a close enough solution in a fraction of time taken by simulated annealing. An even more effective method is the HCF method, although its implementation is a little bit more complex.

It is worth noting that the formulation (3.8.28) comprises, as a special case, the discrete formulation of the optical flow computation described in Section 3.8.5.4. Consider the constraint (3.8.14). Multiplying both sides by $\Delta t = t_k - t_{k-1}$ it becomes $I^x d_1 + I^y d_2 + I^t \Delta t = 0$, where I^x , I^y , I^t are discrete approximations to horizontal, vertical and temporal derivatives, respectively. This constraint is not satisfied exactly and as it turns out $I^x d_1 + I^y d_2 + I^t \Delta t$ is a noise-like term with similar characteristics to the prediction error ε_k . This is not surprising since both originate from the same constant-intensity hypothesis. By replacing the prediction error in (3.8.28) with this new term one obtains a cost function equivalent to the discrete formulation of the optical flow problem (Section 3.8.5.4) [10].

The minimization (3.8.28) leads to smooth displacement fields \mathbf{d}_k , also at object boundaries which is undesirable. To relax the smoothness constraint at object boundaries, explicit models of motion discontinuities (line field) [12] or of motion segmentation labels (segmentation field) [20] can be easily incorporated into the MRF formulation, although their estimation is far from trivial.

3.8.5.6 Experimental comparison of motion estimation methods

To demonstrate the impact of various motion models, Fig. 3.8.7 shows results for the QCIF sequence “Carphone”. Both the estimated displacements and the resulting motion-compensated prediction errors are shown for pixel-based (dense), block-based and segmentation-based motion models. The latter motion estimate was obtained by minimizing the mean squared error ($\Phi(\varepsilon) = \varepsilon^2$ in (3.8.18)) within each region \mathcal{R} from Fig. 3.8.7.c for the affine motion model (3.8.9).

Note the lack of detail due to the low resolution (16×16 blocks) of the block-based approach, but approximately correct motion of objects. The pixel-based model results in a smooth estimate with more spatial detail but at the cost of reduced precision. The segmentation-based motion estimate shows both better accuracy and detail. Although the associated segmentation (Fig. 3.8.7.c) does not correspond exactly to the objects as perceived by humans, it nevertheless closely matches object boundaries. The motion of the head and of the upper body is well-captured, but the motion of landscape in the car window is exaggerated due to the lack of image detail. As for the prediction error, note the blocking artifacts for the block-based motion model (31.8dB⁶) but a very small error for the pixel-based model (35.9dB). The region-based model results in a slightly higher prediction error (35.5dB) than the pixel-based model, but significantly lower than that of the block model.

3.8.6 Perspectives

In the last two decades, motion detection and estimation have moved from research laboratories to specialized products. This has been made possible by two factors. First, enormous advances in VLSI have facilitated practical implementation of CPU-hungry motion algorithms. Secondly, new models and estimation algorithms have lead to improved reliability and accuracy of the estimated motion. With the continuing advances in VLSI, the complexity constraints plaguing motion algorithms will become less of an issue. This should allow practical implementation of more advanced motion models and estimation criteria, and, in turn, further improve the accuracy of the computed motion. One of the

⁶The prediction error is defined as follows: $10 \log(255^2 / \mathcal{E}(\mathbf{d}))$ [dB] for \mathcal{E} from (3.8.18) with quadratic Φ and \mathcal{R} being the whole image.

promising approaches studied today is the joint motion segmentation and estimation that effectively combines the detection and estimation discussed separately in this chapter.

Bibliography

- [1] T. Aach and A. Kaup, "Bayesian algorithms for adaptive change detection in images sequences using markov random fields," in *Signal Process., Image Commun.*, vol. 7, pp. 147–160, 1995.
- [2] M. Bertero, T. Poggio, and V. Torre, "Ill-posed problems in early vision," *Proc. IEEE*, vol. 76, pp. 869–889, Aug. 1988.
- [3] J. Besag, "On the statistical analysis of dirty pictures," *J. Roy. Stat. Soc.*, vol. B 48, pp. 259–279, 1986.
- [4] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*. Kluwer Academic Publishers, 1997.
- [5] M. Black, *Robust incremental optical flow*. PhD thesis, Yale University, Department of Computer Science, Sept. 1992.
- [6] P. Bouthemy and P. Lalande, "Recovery of moving object masks in an image sequence using local spatiotemporal contextual information," *Opt. Eng.*, vol. 32, no. 6, pp. 1205–1212, 1993.
- [7] M. Chahine and J. Konrad, "Estimation and compensation of accelerated motion for temporal sequence interpolation," *Signal Process., Image Commun.*, vol. 7, pp. 503–527, Nov. 1995.
- [8] P. Chou and C. Brown, "The theory and practice of Bayesian image labelling," *Intern. J. Comput. Vis.*, vol. 4, pp. 185–210, 1990.
- [9] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 6, pp. 721–741, Nov. 1984.
- [10] B. Horn, *Robot vision*. Cambridge, MA: MIT Press, 1986.
- [11] Y. Hsu, H.-H. Nagel, and G. Rekers, "New likelihood test methods for change detection in image sequences," *Comput. Vis. Graph. Image Process.*, vol. 26, pp. 73–106, 1984.
- [12] J. Konrad and E. Dubois, "Bayesian estimation of motion vector fields," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, pp. 910–927, Sept. 1992.
- [13] J. Konrad and F. Dufaux, "Improved global motion estimation for N3," Tech. Rep. MPEG97/-M3096, ISO/IEC JTC1/SC29/WG11, Feb. 1998.
- [14] H. Li, S. Sun, and H. Derin, eds., *Video compression for multimedia computing – Statistically based and biologically inspired techniques*. Kluwer Academic Publishers, 1997.
- [15] F. Luthon, A. Caplier, and M. Liévin, "Spatiotemporal MRF approach with application to motion detection and lip segmentation in video sequences," *Signal Process.*, 1999 (to appear).
- [16] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical recipes in C: The art of scientific computing*. Cambridge University Press, 2-nd ed., 1992.
- [17] I. Sezan and R. Legendijk, eds., *Motion analysis and image sequence processing*. Kluwer Academic Publishers, 1993.
- [18] K. Skifstad and R. Jain, "Illumination independent change detection for real world image sequences," *Comput. Vis. Graph. Image Process.*, vol. 46, pp. 387–399, 1989.
- [19] R. Srinivasan and K. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans. Commun.*, vol. 33, pp. 888–896, Aug. 1985.

- [20] C. Stiller, "Object-based estimation of dense motion fields," *IEEE Trans. Image Process.*, vol. 6, pp. 234–250, Feb. 1997.
- [21] A. Tekalp, *Digital video processing*. Prentice Hall PTR, 1995.
- [22] L. Torres and M. Kunt, eds., *Video coding: Second generation approach*. Kluwer Academic Publishers, 1996.
- [23] E. Trucco and A. Verri, *Introductory techniques for 3-D computer vision*. Prentice Hall, 1998.
- [24] H. van Trees, *Detection, estimation and modulation theory*. New York: John Wiley and Sons, 1968.

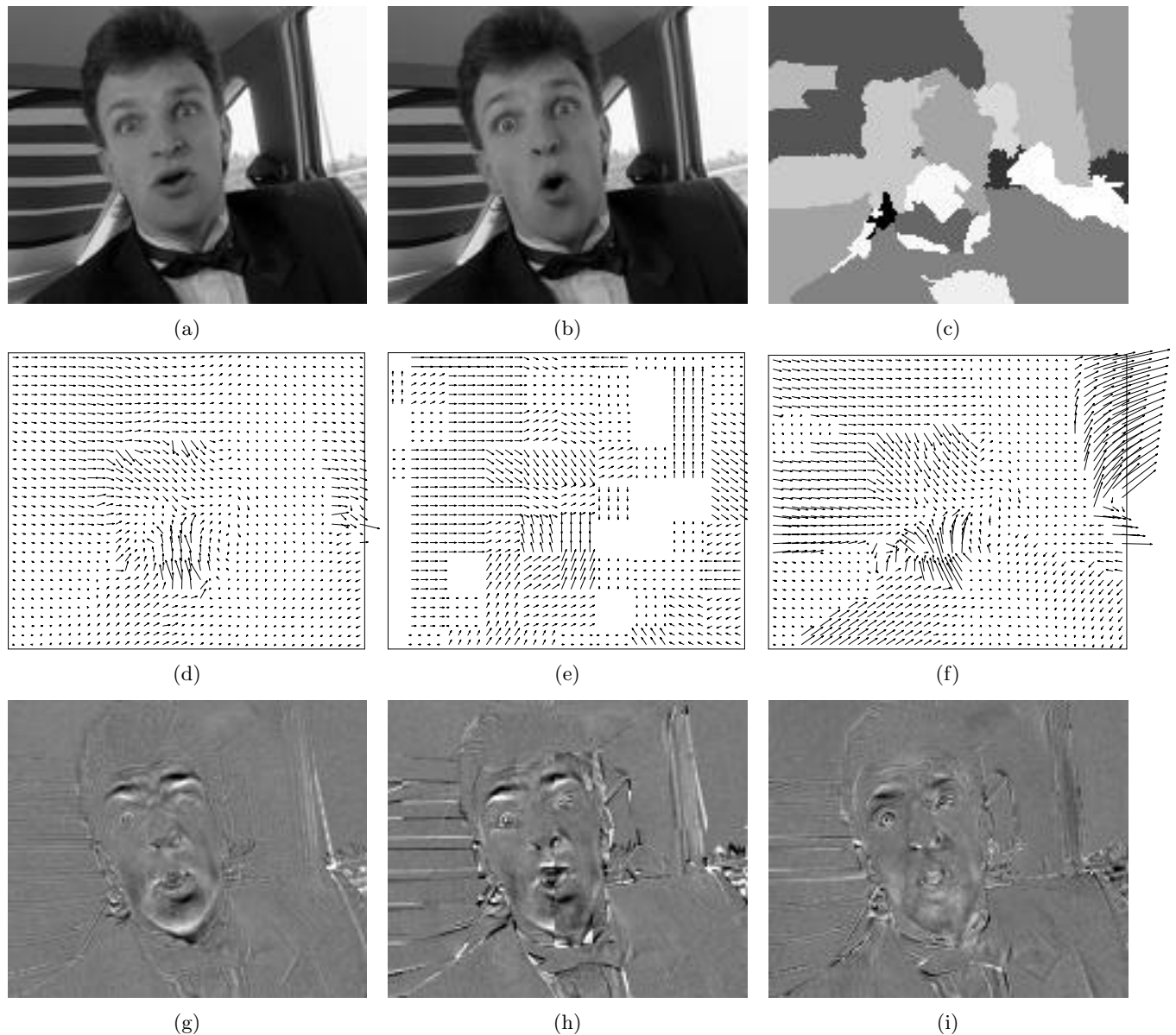


Figure 3.8.7: Original frames (a) #168 and (b) #171 from QCIF sequence “Carphone”; (c) motion-based segmentation of frame #171; motion estimates (sub-sampled by 4) and the resulting motion-compensated prediction error (magnified by 2) at frame #171 for: (d,g) dense-field MAP estimation; (e,h) 16×16 block matching; (f,i) region-based estimation for segments from (c). (From Konrad and Stiller [14, Chapter 4]. Reproduced with permission of Kluwer Academic Publishers.)