Video Compression - Cours 2

Nicolas Boutry[1]

↪ nicolas.boutry@lrde.epita.fr

[1] Laboratoire de Recherche et Développement de l'EPITA (LRDE), France

Novembre 2019

Outline

Outline

Block matching algorithm (BMA)

[SEE COURSE 1]

Deformable block matching algorithm (DBMA)

Introduction I

- In the BMA introduced previously, each block is assumed to undergo a pure translation,
- This model is not appropriate for blocks undergoing rotation, zooming, ...
- In general, a more sophisticated model, such as the affine, bilinear, or projective mapping, can be used to describe the motion of each block.
- Obviously, it will still cover the translational model as a special case.

Introduction II

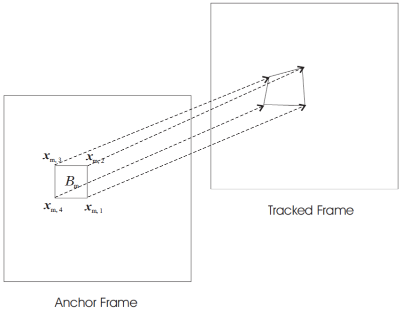


Figure 6.11. The deformable block matching algorithm finds the best matching quadrangle in the tracked frame for each block in the anchor frame. The allowed block deformation depends on the motion model used for the block. Adapted from [39, Fig. 6.9].

- With such models, a block in the reference frame is in general mapped to a non-square quadrangle.
- Therefore, we refer to the case of block-based motion estimation methods using higher order models as deformable block-matching algorithm (DBMA).

Node-based motion representation I

- Here, we introduce a node-based block motion model, which can characterize the same type of motions as the polynomial model, but is easier to interpret and to specify.
- In this model, we assume that a selected number of control nodes in a block can move freely and that the displacement of any interior point can be interpolated from nodal displacements.
- Let the number of control nodes be denoted by K, and the MVs of the control nodes in B_m by $d_{m,k}$, the motion function over the block is described by:

$$d_m(x) = \sum_{k=1}^{K} \phi_{m,k}(x) d_{m,k}, \; x \in B_m$$

Node-based motion representation II

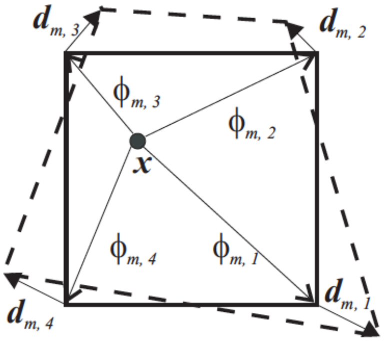


Figure 6.12. Interpolation of motion in a block from nodal MVs.

Node-based motion representation III

- The above equation expresses the displacement at any point in a block as an interpolation of nodal displacements.
- The interpolation kernel $\phi_{m,k}(x)$ depends on the desired contribution of the k-th control point in B_m to x.
- The translational, affine, and bilinear models are special cases of the node-base model with 1, 3, and 4 nodes, respectively.
- A model with more nodes can characterize more complex deformation.

Node-based motion representation IV

- The interpolation kernel in the 1-node case (at the block center or a chosen corner) is a pulse function.
- The interpolation functions in the 3-node case (any three corners in a block) and 4-node (the four corners) cases are affine and bilinear functions respectively.
- Usually, to use an affine model with a rectangular block, the block is first divided into two triangles, and then each triangle is modeled by the 3-node model.

Node-based motion representation V

- Nodal MVs vs. Polynomial coefficients:
  - The nodal MVs can be estimated more easily and specified with a lower precision than for the polynomial coefficients,
  - It is easy to determine appropriate search ranges and search step-sizes for the nodal MVs than the polynomial coefficients (thanks to prior knowledge),
  - All the motion parameters in the node-based representation are equally important, while those in the polynomial representation cannot be treated equally.
  - Specification of the polynomial coefficients requires a high order of precision: a small change in a high order can generate a very different motion field.

Motion estimation using node-based model I

- With the node-based motion model, the motion parameters for any block are the nodal MVs, i.e.,

$$\boldsymbol{a} = [d_k; k \in \mathcal{K}],$$

with $\mathcal{K} = \{1, 2, \ldots, K\}$.

- They can be estimated by minimizing the prediction error over this block, i.e.,

$$E(\boldsymbol{a}) = \sum_{\boldsymbol{x} \in B} (\psi_2(w(\boldsymbol{x}; \boldsymbol{a})) - \psi_1(\boldsymbol{x}))^2$$

where

$$w(\boldsymbol{x}; \boldsymbol{a}) = \boldsymbol{x} + \sum_{k \in \mathcal{K}} \phi_k(\boldsymbol{x}) d_k$$

- As with the BMA, there are many ways to minimize the error, including the exhaustive search and the gradient-based search method.

Motion estimation using node-based model II

- The computational load required for the exhaustive search, however, can be unacceptable in practice, because of the high dimension of the search space.
- Gradient-based methods are more feasible in this case.
- In the following, we derive a Newton-Raphson search algorithm:
  - Define $\boldsymbol{a} = [\boldsymbol{a}_x^T, \boldsymbol{a}_y^T]^T$ with $\boldsymbol{a}_x = [d_{x,1}, d_{x,2}, \ldots, d_{x,K}]^T$ and $\boldsymbol{a}_y = [d_{y,1}, d_{y,2}, \ldots, d_{y,K}]^T$.
  - We can show that:

$$\frac{\partial E}{\partial \boldsymbol{a}}(\boldsymbol{a}) = \left[\frac{\partial E}{\partial \boldsymbol{a}_x}, \frac{\partial E}{\partial \boldsymbol{a}_y}\right]^T$$

with:

$$\frac{\partial E}{\partial \boldsymbol{a}_x}(\boldsymbol{a}) = 2 \sum_{\boldsymbol{x} \in B} e(\boldsymbol{x}; \boldsymbol{a}) \frac{\partial \psi_2(w(\boldsymbol{x}; \boldsymbol{a}))}{\partial x} \phi(\boldsymbol{x}),$$

and

$$\frac{\partial E}{\partial \boldsymbol{a}_y}(\boldsymbol{a}) = 2 \sum_{\boldsymbol{x} \in B} e(\boldsymbol{x}; \boldsymbol{a}) \frac{\partial \psi_2(w(\boldsymbol{x}; \boldsymbol{a}))}{\partial y} \phi(\boldsymbol{x}),$$

Motion estimation using node-based model III

where we define:

$$e(\boldsymbol{x};\boldsymbol{a}) = \psi_2(w(\boldsymbol{x};\boldsymbol{a})) - \psi_1(\boldsymbol{x})$$

and

$$\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \dots, \phi_K(\boldsymbol{x})]^T$$

The proof of the formulas above are at the end of these slides in the appendix!

- By dropping the terms involving second order gradients, the Hessian matrix:

$$[H(\boldsymbol{a})] = \begin{bmatrix} H_{xx}(\boldsymbol{a}) & H_{xy}(\boldsymbol{a}) \\ H_{xy}(\boldsymbol{a}) & H_{yy}(\boldsymbol{a}) \end{bmatrix}$$

can be approximated as:

$$H_{xx}(\boldsymbol{a}) = 2 \sum_{\boldsymbol{x} \in B} \left(\frac{\partial \psi_2}{\partial x} \right)^2 \Bigg|_{w(\boldsymbol{x};\boldsymbol{a})} \boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})^T,$$

$$H_{yy}(\boldsymbol{a}) = 2 \sum_{\boldsymbol{x} \in B} \left(\frac{\partial \psi_2}{\partial y} \right)^2 \Bigg|_{w(\boldsymbol{x};\boldsymbol{a})} \boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})^T,$$

$$H_{xy}(\boldsymbol{a}) = 2 \sum_{\boldsymbol{x} \in B} \frac{\partial \psi_2}{\partial x} \frac{\partial \psi_2}{\partial y} \Bigg|_{w(\boldsymbol{x};\boldsymbol{a})} \boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})^T,$$

Motion estimation using node-based model IV

- The Newton-Raphson update algorithm is:

$$\boldsymbol{a}^{l+1} = \boldsymbol{a}^{l} - \alpha \left[H(\boldsymbol{a}^{l})\right]^{-1} \frac{\partial E}{\partial \boldsymbol{a}}(\boldsymbol{a}^{l})$$

- The update at each iteration thus requires the inversion of the $2K \times 2K$ symmetric matrix $[H]$.
- To reduce numerical computations, we can update the displacements in x and y directions separately:

$$\boldsymbol{a}_x^{l+1} = \boldsymbol{a}_x^{l} - \alpha \left[H_{xx}(\boldsymbol{a}^{l})\right]^{-1} \frac{\partial E}{\partial \boldsymbol{a}_x}(\boldsymbol{a}^{l}),$$

$$\boldsymbol{a}_y^{l+1} = \boldsymbol{a}_y^{l} - \alpha \left[H_{yy}(\boldsymbol{a}^{l})\right]^{-1} \frac{\partial E}{\partial \boldsymbol{a}_y}(\boldsymbol{a}^{l}),$$

- In this case, we only need to invert to $K \times K$ matrices in each update.
- For the 4-node case, $[H]$ is a 8×8 matrix, while $[H_{xx}]$ and $[H_{yy}]$ are 4×4 matrices.

Motion estimation using node-based model V

- As with all gradient-based methods, the above update algorithm may reach a bad local minimum that is far from the global minimum, if the initial solution is not chosen properly.
- A good initial solution can often be provided by the EBMA.

Note-based motion estimation [TP]

[SEE NOTEBOOK]

Mesh-based motion estimation

Introduction I

- With the block-based model used either in BMA or DBMA, motion parameters in individual blocks are independently specified.
- Unless motion parameters of adjacent blocks are constrained to vary smoothly, the estimated motion field is often discontinuous and sometimes chaotic.
- One way to overcome this problem is by using mesh-based motion estimation (see below).
- The reference frame is covered by a mesh and the aim is to find which combined motion of the nodes of this mesh minimizes some prediction error (best matching),
- The motion within each polygonal element is interpolated from nodal's MVs.

Introduction II

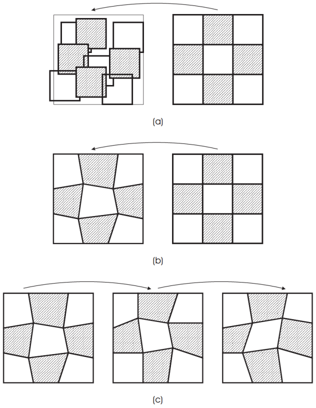


Figure 6.13. Comparison of block-based and mesh-based motion representations: (a) Block-based motion estimation between two frames, using a translational model within each block in the anchor frame; (b) Mesh-based motion estimation between two frames, using a regular mesh at the anchor frame; (d) Mesh-based motion tracking, using the tracked mesh for each new anchor frame.

Introduction III

- As long as the nodes in the current frame still form a feasible mesh, the mesh-based motion representation is guaranteed to be continuous,
- Another benefit of mesh-based representation is that it enables continuous tracking of the same set of nodes over consecutive frames,
- It is desirable in applications require object tracking.
- We can use a same set of nodes for $n \geq 2$ consecutive frames (each mesh represents a partition of the image space !),
- Note that the inherent continuity with the mesh-based representation is not always desired!
- In real-world video sequences, there are often motion discontinuities at object boundaries.

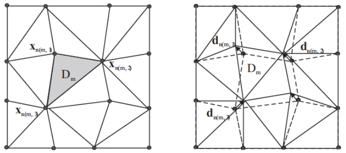
Introduction IV

- As with the block-based representation, the accuracy of the mesh-based representation depends on the number of nodes.
- A very complex motion field can be reproduced as long as a sufficient numbers of nodes are used.
- To minimize the number of nodes required, the mesh should be adapted to the imaged scene,
- If a non-adaptive mesh is used, a larger number of nodes are usually needed to approximate the motion field accurately.

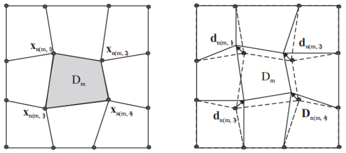
Mesh-based motion representation I

- With the mesh-based motion representation, the underlying image domain in the frames are partitioned into non-overlapping polygonal elements.
- Each polygonal element is defined by a few nodes and links between the nodes.
- Such a mesh is also known as a control grid.
- The motion field over the entire frame is then described by $\underline{\text{MVs at the nodes only}}$.
- The MVs at the interior points of an element are interpolated from the MVs at the nodes of this element.
- Usual constraint: $\underline{\text{no flip-over polygonal elements}}$ (feasible-mesh).

Mesh-based motion representation II



(a)



(b)

Figure 6.14. Illustration of mesh-based motion representation: (a) using a triangular mesh, with 3 nodes attached to each element, (b) using a quadrilateral mesh, with 4 nodes attached to each element. In the shown example, the two meshes have the same number of nodes, but the triangular mesh has twice the number of elements. The left column shows the initial mesh over the anchor frame, the right column the deformed mesh in the tracked frame.

Mesh-based motion representation III

- Notations:
  - $t \in \{1, 2\}$ the time value,
  - $\mathcal{M} = \{1, 2, \ldots, M\}$ (polygonal elements),
  - $\mathcal{N} = \{1, 2, \ldots, N\}$ (nodes),
  - $\mathcal{K} = \{1, 2, \ldots, K\}$ (indices of node in a given polygonal element),
  - Let the m-th element and n-th node in frame t be denoted by $B_{t,m}$, $m \in \mathcal{M}$, and $x_{t,n}$, $n \in \mathcal{N}$, and the MV of the n-th node by $d_n = x_{2,n} - x_{1,n}$.
  - $n(m, k)$ the global index of the k-th node in the m-th polygonal element.
- The motion field in element $B_{1,m}$ is related to the nodal MVs d_n by:

$$d_m(x) = \sum_{k \in \mathcal{K}} \phi_{m,k}(x) d_{n(m,k)}, \; x \in B_{1,m},$$

- The function $\phi_{m,k}(x)$ is the interpolation kernel associated with the node k in the polygonal element m.

Mesh-based motion representation IV

- It depends on the desired contribution of the k-th node in $B_{1,m}$ to the MV at x.
- To guarantee continuity across polygonal element boundaries, the interpolation must satisfy:

$$0 \leq \phi_{m,k}(x) \leq 1, \; \forall x \in B_m, \quad \text{(positivity)}$$

and

$$\sum_k \phi_{m,k}(x) = 1, \; \forall x \in B_m, \quad \text{(unitary total contribution)}$$

and

$$\phi_{m,k}(x_l) = \delta_{k,l} \quad \text{(l local index!)}$$

- In other words, only a part of the nodes are used for each polygonal element.
- If all the polygonal elements have the same shape, then all the interpolation kernels are equal, that is, $\phi_{m,k}(x) = \phi_k(x)$.

Mesh-based motion representation V

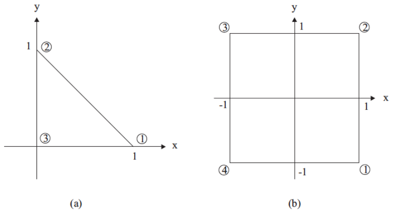


Figure 6.15. (a) A standard triangular element; (b) A standard quadrilateral element (a square).

- The interpolation kernels for the standard triangular element are:

$$\phi_1^t(x,y) = x,$$
$$\phi_2^t(x,y) = y,$$
$$\phi_3^t(x,y) = 1 - x - y.$$

Mesh-based motion representation VI

- The interpolation kernels for the standard quadrilateral element are:

$$\begin{aligned}
\phi_1^q(x,y) &= (1+x)(1-y)/4,\\
\phi_2^q(x,y) &= (1+x)(1+y)/4,\\
\phi_3^q(x,y) &= (1-x)(1+y)/4,\\
\phi_4^q(x,y) &= (1-x)(1-y)/4.
\end{aligned}$$

- We see that the interpolation kernels for these two cases are affine and bilinear functions respectively.

Mesh-based motion representation VII

- Note that a mesh in which each element corresponds to a smooth surface patch in a single object can lead to more accurate ME (since the motion prediction is smooth by construction!).
- An object adaptive mesh would also be more appropriate for motion tracking over a sequence of frames (see on the web at mesh generation problems).
- In the mesh-based model, each node is assigned a single MV, which will affect the interpolated motion functions in four quadrilateral elements attached to this node.
- In node-based model, node *n* can have four MVs! [explain why!]

Motion estimation using mesh-based method I

- With the mesh-based motion representation, the motion parameters include the nodal MVs, that is,

$$\boldsymbol{a} = \{d_n, n \in \mathcal{N}\},$$

- To estimate them, we can again use an error minimization approach.
- Under the mesh-based motion model, the DFD error becomes:

$$E(d_n, n \in \mathcal{N}) = \sum_{m \in \mathcal{M}} \sum_{x \in B_{1,m}} |\psi_2(w_m(x)) - \psi_1(x)|^p,$$

where

$$w_m(x) = x + \sum_{k \in \mathcal{K}} \phi_{m,k}(x) d_{n(m,k)}, \; x \in B_{1,m}$$

Motion estimation using mesh-based method II

- In general, the error function in the last equation is difficult to calculate because of the irregular shape of $B_{1,m}$ (not a rectangle for example).
- To simplify the calculation, we can think of $B_{t,m}$, $t = 1, 2$, as being deformed from a master element with a regular shape.

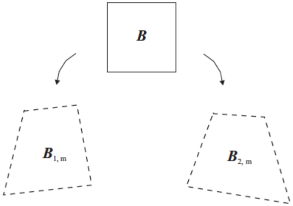


Figure 6.16. Mapping from a master element $\tilde{\mathcal{B}}$ to two corresponding elements in the anchor and tracked frames $\mathcal{B}_{1,m}$ and $\mathcal{B}_{2,m}$.

Motion estimation using mesh-based method III

- Here we only consider the case when all the polygonal elements have the same topology (all squares or all triangles),
- Then they can be mapped from the same master element $\tilde{B}$.
- Let $\tilde{\phi}_k(u)$ represent the interpolation kernel associated with the k-th node in $\tilde{B}$, the mapping functions from $\tilde{B}$ to $B_{t,m}$ are:

$$\tilde{w}_{t,m}(u) = \sum_{k \in \mathcal{K}} \tilde{\phi}_k(u) x_{t,n(m,k)}, \ \forall\, u \in \tilde{B}, \ \forall\, t \in \{1,2\},$$

  where $x_{t,n(m,k)}$ corresponds to the $\underline{\text{position}}$ of the $n(m,k)^{th}$ node ($\underline{\text{and not to a MV !}}$) in the frame at time t,

Motion estimation using mesh-based method IV

- Then the (total) error can be calculated over the master element as:

$$E(d_n, n \in \mathcal{N}) = \sum_{m \in \mathcal{M}} \sum_{u \in \tilde{B}} |\tilde{e}_m(u)|^p \, J_m(u)$$

where

$$\tilde{e}_m(u) = \psi_2(\tilde{w}_{2,m}(u)) - \psi_1(\tilde{w}_{1,m}(u))$$

is the (local) error between the two image frames at points that are both mapped from u is the master element.

- The (weighting) function $J_m(u)$ defined as:

$$J_m(u) = \left| \det \left(\frac{\partial \tilde{w}_{1,m}(u)}{\partial u} \right) \right|$$

denotes the Jacobian of the transformation $\tilde{w}_{1,m}(u)$.

Motion estimation using mesh-based method V

- The gradient of the error function is, when $p = 2$:

$$\frac{\partial E_p}{\partial d_n} = 2 \sum_{m \in \mathcal{M}_n} \sum_{u \in \tilde{B}} \tilde{e}_m(u) \tilde{\phi}_{k(m,n)}(u) \left. \frac{\partial \psi_2(x)}{\partial x} \right|_{\tilde{w}_{2,m}(u)} J_m(u),$$

  where $\mathcal{M}_n$ includes the indices of the elements that are attached to node n, and $k(m,n)$ specifies the local index of node n in the m-th adjacent element.

- Either the first order gradient descent method or the second order Newton-Raphson type of update algorithm could be used.

- The second order method will converge much faster, but it is also more liable to converge to bad local minima.

- The newly updated nodal positions based on the gradient function can lead to overly deformed polygonal elements (including flip-over and obtuse elements).

- In order to prevent such things from happening, one should limit the search range where the updated nodal position can fall into.

Motion estimation using mesh-based method VI

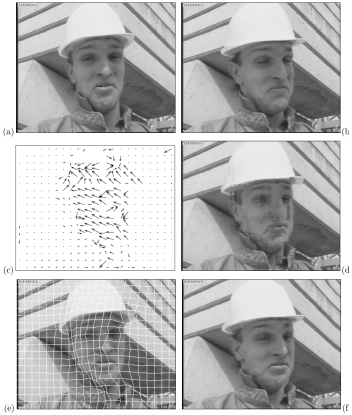


Figure 6.8. Example motion estimation results: (a) the tracked frame; (b) the anchor frame; (c-d) motion field and predicted image for the anchor frame (PSNR=29.86 dB) obtained by half-pel accuracy EBMA ; (e-f) motion field (represented by the deformed mesh overlaid on the tracked frame) and predicted image (PSNR=29.72 dB) obtained by the mesh-based motion estimation scheme in [43].

Motion estimation using mesh-based method VII

- A careful comparison between the predicted image using the mesh-based approach and the actual image will reveal that the eye closing and the mouth movement are not accurately reproduced, and there are some artificial warping artifacts near the jaw and neck,
- The PSNR of the predicted image with mesh-based method is lower than that obtained by the EBMA.

Global motion estimation

Introduction I

- Depending on the context, the mapping function between the two images can be described by a translation, a geometric transformation, an affine mapping, or a projection mapping.
- There are usually at least two objects, a stationnary background and one or more moving foregrounds.
- Fortunately, when the foreground object motion is small and the camera does not move in the Z-direction, the motion field can be approximated well by a global model.

Introduction II

- Such camera motions are quite common in sports videos and movies.
- As long as the effect of the camera motion dominates over other motions (motion of individual small objects), determination of this dominant global motion is still very useful.
- Two approaches are possible for estimating the global motion:
  - minimization of the prediction error,
  - first determine (pixel-/block-wise) MVs, and then using a regression method.

Region-based motion estimation

Introduction I

- There are usually multiple types of motions in the imaged scene,
- By region-based motion estimation, we mean to segment the underlying image frame into multiple regions and estimate the motion parameters of each region.
- The segmentation should be such that a single parametric motion model can represent well the motion in each region.
- The simplest approach is to require each region undergo the same translational motion.
- For a more efficient motion representation, an affine or bilinear or perspective motion model should be used.

Introduction II

- 3 possible approaches:
  1. (1) One first segments the image frame into different regions based on texture homogeneity, the edge information, and then estimates the motion in each region.

     !! We call such a method region-first.

  2. (2) One first estimates the motion field over the entire frame, and then segments the resulting motion field so that the motion in each region can be modeled by a single parameter model.

     ! We call this model motion first.

     !! The resulting region can be further refined subject to some spatial connectivity constraints.

Introduction III

!!! The second problem involves motion-based segmentation.

(3) One <u>jointly</u> estimates region partition and motion in each region.

! In general, this is accomplished by an iterative process, which performs region segmentation and ME alternatively.

Motion-based region segmentation I

- Motion-based segmentation refers to the partitioning of a motion field into multiple regions so that the motion within each region can be described by a single set of motion parameters.
- 2 possible approaches: clustering and layered sequential approach,

Motion-based region segmentation II

- First approach: clustering:
  - Consider the case when the motion model for each region is a pure translation.
  - Then the segmentation task is to group all spatially connected pixels with similar MVs into one region.
  - This can be done easily by an automated clustering method (*k*-means,...).
  - In this process, the spatial connectivity is not considered.
  - Therefore, the resulting regions may contain pixels that are not spatially connected.
  - Post-processing steps may be applied at the end of the iterations of the clustering algorithm to improve the spatial connectivity of the resulting region.
  - When the motion model for each region is not a simple translation, motion-based clustering is not as straightforward.

Motion-based region segmentation III

- This is because we cannot use the similarity between MVs as the criterion for performing clustering.
- One way is to find a set of motion parameters for each pixel.
- Then we can employ the clustering method described previously, by replacing the raw MVs with the motion parameter vector.

Motion-based region segmentation IV

- Second approach: layered (sequential) approach:
  - The motion field in a scene can be decomposed into layers, with the first layer representing the most dominant motion, the second layer the less dominant one, and so on.
  - For example, in a video clip of a tennis play,
    - The background will be the first layer,
    - It usually undergoes a coherent global camera motion,
    - The player is the second layer,
    - It will have different motions for each part of the body,
    - The racket will be the third layer,
    - The ball will be the fourth layer.

Motion-based region segmentation V

- Algorithm:
  - We model the motion field of the entire frame by a single set of parameters,
  - we continuously eliminate outlier pixels from the remaining inlier group until all the pixels within the inlier group can be modeled well.
  - This will yield the first dominant region (corresponding to the inlier region) and its associated motion.
  - The same approach can then be applied to the remaining pixels (the outlier region), to identify the next dominant region and its motion.
  - This process continues until no more outlier pixel exist.

Multi-resolution motion estimation

Introduction I

- Various ME approaches can be reduced (see below) to solving an error minimization problem.
- Major difficulties:
  - many local minima
  - it is not easy to reach the global minimum except if we are close to the initial solution,
  - the amount of computation involved in the minimization process is very high.
- A good alternative is then the multi-resolution approach
- It searches the solution for an optimization problem in successively finer resolutions.
- By first searching the solution in coarse resolution, one can usually obtain a solution that is close to the true motion.
- The total number of searches is reduced (small neighborhoods in each scale).

General formulation I

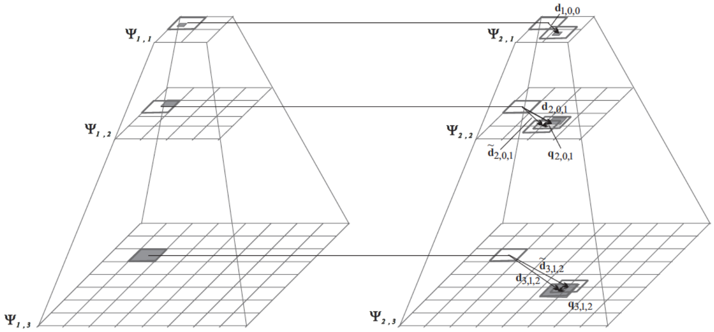


Figure 6.19. Illustration of the Hierarchical Block Matching Algorithm.

Figure: A multi-resolution approach (HBMA)

General formulation II

- In these pyramidal representations, the images at the top are the coarsest levels, and the images at the bottom are the original images.
- The motion field starts from the top coarsest level and progresses to the next finer level repeatedly.
- At each new finer resolution level, the motion field is interpolated and then refined.
- Assume that the number of levels is L, with the L-th level being the original image.
- Let the l-th level images of the reference and current frames be represented by:

$$\psi_{t,l}(x), \forall x \in \Lambda_l, \forall t \in \{1,2\},$$

  where Λ_l is the set of pixels at level l.
- Denote the total motion field obtained from levels 1 to $l-1$ by $d_{l-1}(x)$.

General formulation III

- At the l-th level, we first interpolate $d_{l-1}(x)$ to the resolution of level l, to produce an initial motion estimate:

$$\tilde{d}_l(x) = \mathcal{U}(d_{l-1}(x))$$

  where $\mathcal{U}$ represents the interpolation.

- We then determine the update $q_l(x)$ at this level so that the error:

$$\sum_{x \in \Lambda_l} \left| \psi_{2,l}(x + \tilde{d}_l(x) + q_l(x)) - \psi_{1,l}(x) \right|^p$$

  is minimized.

- The new motion field obtained after this step is:

$$d_l(x) = q_j(x) + \tilde{d}_l(x)$$

General formulation IV

- Upon completion of successive refinements, the total motion at the finest resolution is:

$$d(x) = q_L(x) + \mathcal{U}(q_{L-1}(x) + \mathcal{U}(q_{L-2}(x) + \mathcal{U}(...(q_2(x) + \mathcal{U}(q_1(x) + d_0(x))))))$$

- The initial conditions for the above procedure is $d_0(x) = 0$.
- The benefits of the multi-resolution are two folds:
  - First, the minimization problem at a coarser resolution is less ill-posed than at a finer resolution,
  - The solution obtained at a coarser level is $\underline{\text{closer}}$ to the $\underline{\text{true solution}}$ at that level,
  - At the end, we converge then more easily to the $\underline{\text{global minimum}}$.
  - Second, the estimation at each resolution level can be confined to a $\underline{\text{smaller range}}$,
  - Then the total number of searches is $\underline{\text{smaller}}$ than in the other methods.
- The use of multi-resolution representation for image processing was first introduced by Burt and Adelson (Laplacian Pyramid).

Hierarchical block matching algorithm (HBMA) I

- Here, we introduce a hierarchical block-matching algorithm (HBMA), which is a special case of the multi-resolution approach presented before.
- We assume that the same block size is used at different levels.
- The number of blocks is then reduced by half in each dimension.
- Let the MV for block (m, n) at level l be denoted by $d_{l,m,n}$.
- Starting from level 1, we first find the MVs for all blocks in this level.
- At each new level $l > 1$, for each block, its initial MV $\tilde{d}_{l,m,n}$ is interpolated from a corresponding block in level $l - 1$ by:

$$\tilde{d}_{l,m,n} = \mathcal{U}(d_{l-1,\lfloor m/2\rfloor,\lfloor n/2\rfloor}) = 2\, d_{l-1,\lfloor m/2\rfloor,\lfloor n/2\rfloor}$$

- Then a correction vector $q_{l,m,n}$ is searched, yielding the final estimated MV:

$$d_{l,m,n} = \tilde{d}_{l,m,n} + q_{l,m,n}$$

Hierarchical block matching algorithm (HBMA) II

- Note that using a block width N at level l corresponds to a block width of $2^{l-1}N$ at the full resolution.
- The same scaling applies to the search range and stepsize.
- Therefore, by using the same block size, search range and stepsize at different levels, we actually use a
  larger block size, a larger search range, as well as a larger stepsize in the beginning of the search, and then gradually reduce (by half) these in later steps.

Hierarchical block matching algorithm (HBMA) III

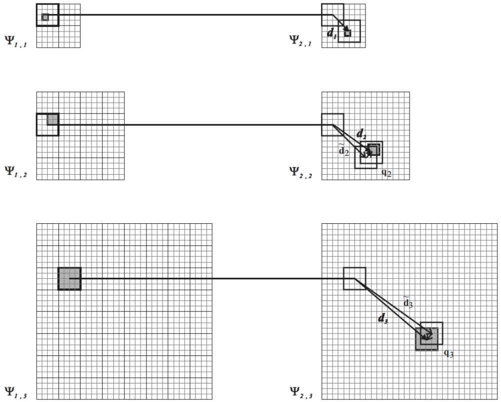


Figure 6.20. An example of using 3-level HBMA for block motion estimation. See Example 6.3.

Hierarchical block matching algorithm (HBMA) IV

- The complexity using HBMA is about:

$$4^{-(L-2)} M^2 R^2$$

  when the image is of size $M \times M$, any block of size is $N \times N$, the search range is R in the finest resolution, and the pyramid has L levels.

- The complexity of the EBMA is $M^2 R^2$, then we obtain a reduction by $2^{-(L-2)}$.

- Typically, the number of levels is 2 or 3.

Hierarchical block matching algorithm (HBMA) V

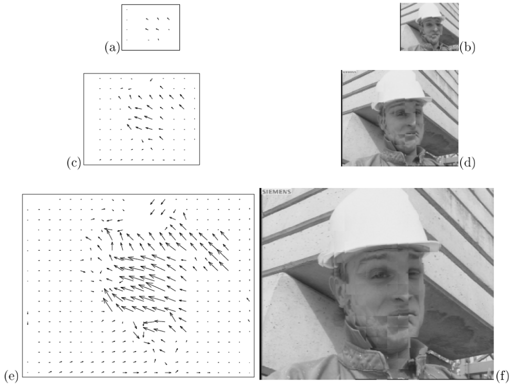


Figure 6.21. Example motion estimation results by HBMA for the two images shown in Fig. 6.8: (a-b) the motion field and predicted image at level 1; (c-d) the motion field and predicted image at level 2; (e-f) the motion field and predicted image at the final level (PSNR=29.32); A three-level HBMA algorithm is used. The block size is 16×16 at all levels. The search range is 4 at all levels with integer-pel accuracy. The result in the final level is further refined by a half-pel accuracy search in the range of ± 1.

Hierarchical block matching algorithm (HBMA) VI

- We can see that the multi-resolution approach indeed yields a smoother motion field than EBMA.
- Visual observation also reveals that this motion field represents more the underlying motion between the two image frames.
- This is true in spite of the fact that the EBMA yields a higher PSNR (but the PSNR is not always a good measure!).
- At the end, EBMA will have required $4.3E+8$ operations and the HBMA $1.1E+7$ operations.
- There are many variants to implementation of HBMA as the quad-tree BMA.
- It starts with a larger block size, and then repeatedly divides a block into four if the matching error for this block is still larger than a given threshold.
- In this case, all the processing is done on the original image resolution.

Outline

Computation of the gradient of the DFD energy in the node-based case model I

- Since we use the node-based motion model, the motion parameters for any block are the nodal MVs, i.e.,

$$\boldsymbol{a} = [d_k ; k \in \mathcal{K}],$$

with $\mathcal{K} = \{1, 2, \ldots, K\}$.

- We are going to use the gradient-based method to minimize the following energy:

$$E(\boldsymbol{a}) = \sum_{\boldsymbol{x} \in B} (\psi_2(w(\boldsymbol{x}; \boldsymbol{a})) - \psi_1(\boldsymbol{x}))^2$$

- The new position $w(\boldsymbol{x}; \boldsymbol{a})$ of each $\boldsymbol{x}$ is predicted w.r.t. the motions d_k of the nodes of each quadrangle B so that:

$$w(\boldsymbol{x}; \boldsymbol{a}) = \boldsymbol{x} + \sum_{k \in \mathcal{K}} \phi_k(\boldsymbol{x}) d_k$$

Computation of the gradient of the DFD energy in the node-based case model II

- We define the motion parameter as:

$$\boldsymbol{a} = [\boldsymbol{a}_x^T, \boldsymbol{a}_y^T]^T$$

  ,
  with $\boldsymbol{a}_x = [d_{x,1}, d_{x,2}, \dots, d_{x,K}]^T$ and $\boldsymbol{a}_y = [d_{y,1}, d_{y,2}, \dots, d_{y,K}]^T$, then $\boldsymbol{a}$ is a vector made of $2K$ scalars:

$$\boldsymbol{a} = [d_{x,1}, d_{x,2}, \dots, d_{x,K}, d_{y,1}, d_{y,2}, \dots, d_{y,K}]^T,$$

- In other words, for $i \in [1, K]$, $a_x^i = d_{x,i}$, and $a_y^i = d_{y,i}$,
- To begin the computation of the derivative of the energy, we separate it into two vectors $\frac{\partial E}{\partial \boldsymbol{a}_x}$ and $\frac{\partial E}{\partial \boldsymbol{a}_y}$,
- So, we have for the moment:

$$\frac{\partial E}{\partial \boldsymbol{a}} = \left[\frac{\partial E}{\partial \boldsymbol{a}_x}, \frac{\partial E}{\partial \boldsymbol{a}_y}\right]^T$$

Computation of the gradient of the DFD energy in the node-based case model III

- We want to compute each of these two derivatives, let us begin with $\frac{\partial E}{\partial \boldsymbol{a}_x}$:
- Let i be an integer in $[1, K]$,

$$\frac{\partial E}{\partial a_x^i} = \frac{\partial}{\partial a_x^i}\left(\sum_{\boldsymbol{x}\in B}(\psi_2(w(\boldsymbol{x};\boldsymbol{a})) - \psi_1(\boldsymbol{x}))^2\right),$$

- Since the sum on $\boldsymbol{x}$ is finite, we obtain:

$$\frac{\partial E}{\partial a_x^i} = \sum_{\boldsymbol{x}\in B} \frac{\partial}{\partial a_x^i}\left((\psi_2(w(\boldsymbol{x};\boldsymbol{a})) - \psi_1(\boldsymbol{x}))^2\right),$$

- Now we can derive:

$$\frac{\partial E}{\partial a_x^i} = \sum_{\boldsymbol{x}\in B} 2\,(\psi_2(w(\boldsymbol{x};\boldsymbol{a})) - \psi_1(\boldsymbol{x}))\,\frac{\partial}{\partial a_x^i}(\psi_2(w(\boldsymbol{x};\boldsymbol{a})) - \psi_1(\boldsymbol{x})),$$

Computation of the gradient of the DFD energy in the node-based case model IV

- Now we can define the error image:

$$e(\boldsymbol{x};\boldsymbol{a}) = \psi_2(w(\boldsymbol{x};\boldsymbol{a})) - \psi_1(\boldsymbol{x}),$$

- We obtain then:

$$\frac{\partial E}{\partial a_x^i} = 2\sum_{\boldsymbol{x}\in B} e(\boldsymbol{x};\boldsymbol{a})\ \frac{\partial}{\partial a_x^i}(\psi_2(w(\boldsymbol{x};\boldsymbol{a})) - \psi_1(\boldsymbol{x})),$$

- Since $\psi_1(\boldsymbol{x})$ does not depend on $\boldsymbol{a}$:

$$\frac{\partial E}{\partial a_x^i} = 2\sum_{\boldsymbol{x}\in B} e(\boldsymbol{x};\boldsymbol{a})\ \frac{\partial}{\partial a_x^i}(\psi_2(w(\boldsymbol{x};\boldsymbol{a}))),$$

Computation of the gradient of the DFD energy in the node-based case model V

- Let us denote w_x and w_y this way:

$$w(\boldsymbol{x};\boldsymbol{a}) = [w_x(\boldsymbol{x};\boldsymbol{a}), w_y(\boldsymbol{x};\boldsymbol{a})]^T,$$

with

$$w_x(\boldsymbol{x};\boldsymbol{a}) = x + \sum_{k\in\mathcal{K}} \phi_k(\boldsymbol{x}) d_{x,k}$$

and

$$w_y(\boldsymbol{x};\boldsymbol{a}) = y + \sum_{k\in\mathcal{K}} \phi_k(\boldsymbol{x}) d_{y,k}$$

Computation of the gradient of the DFD energy in the node-based case model VI

- Then,

$$\frac{\partial}{\partial a_x^i}(\psi_2(w(\boldsymbol{x};\boldsymbol{a}))) = \frac{\partial}{\partial a_x^i}(\psi_2(w_x(\boldsymbol{x};\boldsymbol{a}), w_y(\boldsymbol{x};\boldsymbol{a}))), \tag{1}$$

$$= \frac{\partial \psi_2}{\partial x}(w(\boldsymbol{x};\boldsymbol{a}))\frac{\partial(w_x(\boldsymbol{x};\boldsymbol{a}))}{\partial a_x^i} + \frac{\partial \psi_2}{\partial y}(w(\boldsymbol{x};\boldsymbol{a}))\frac{\partial(w_y(\boldsymbol{x};\boldsymbol{a}))}{\partial a_x^i} \tag{2}$$

- We can easily observe that:

$$\frac{\partial(w_x(\boldsymbol{x};\boldsymbol{a}))}{\partial a_x^i} = \frac{\partial(x + \sum_{k\in\mathcal{K}} \phi_k(\boldsymbol{x}) d_{x,k})}{\partial a_x^i} = \phi_i(\boldsymbol{x})$$

and

$$\frac{\partial(w_y(\boldsymbol{x};\boldsymbol{a}))}{\partial a_x^i} = \frac{\partial(y + \sum_{k\in\mathcal{K}} \phi_k(\boldsymbol{x}) d_{y,k})}{\partial a_x^i} = 0$$

so

Computation of the gradient of the DFD energy in the node-based case model VII

$$\frac{\partial}{\partial a_x^i}(\psi_2(w(\boldsymbol{x};\boldsymbol{a}))) = \frac{\partial \psi_2}{\partial x}(w(\boldsymbol{x};\boldsymbol{a}))\phi_i(\boldsymbol{x}),$$

- Then we finally obtain:

$$\frac{\partial E}{\partial \boldsymbol{a}_x}(\boldsymbol{a}) = 2\sum_{\boldsymbol{x}\in B} e(\boldsymbol{x};\boldsymbol{a})\frac{\partial \psi_2(w(\boldsymbol{x};\boldsymbol{a}))}{\partial x}\phi(\boldsymbol{x}),$$

and by symmetry we also obtain:

$$\frac{\partial E}{\partial \boldsymbol{a}_y}(\boldsymbol{a}) = 2\sum_{\boldsymbol{x}\in B} e(\boldsymbol{x};\boldsymbol{a})\frac{\partial \psi_2(w(\boldsymbol{x};\boldsymbol{a}))}{\partial y}\phi(\boldsymbol{x}),$$

- Done :)

References

Indrajit Chakrabarti, Kota Naga Srinivasarao Batta, S. K. C. (2015).
Motion Estimation for Video Coding.
Springer.

Jie Chen, Ut-Va Koc, K. R. L. (2002).
Design of Digital Video Coding Systems.
Signal Processing and Communication Series, Marcel Dekker, Inc.

Yao Wang, Jorn Ostermann, Y.-Q. Z. (2002).
Video Processing and Communications.
Prentice Hall.