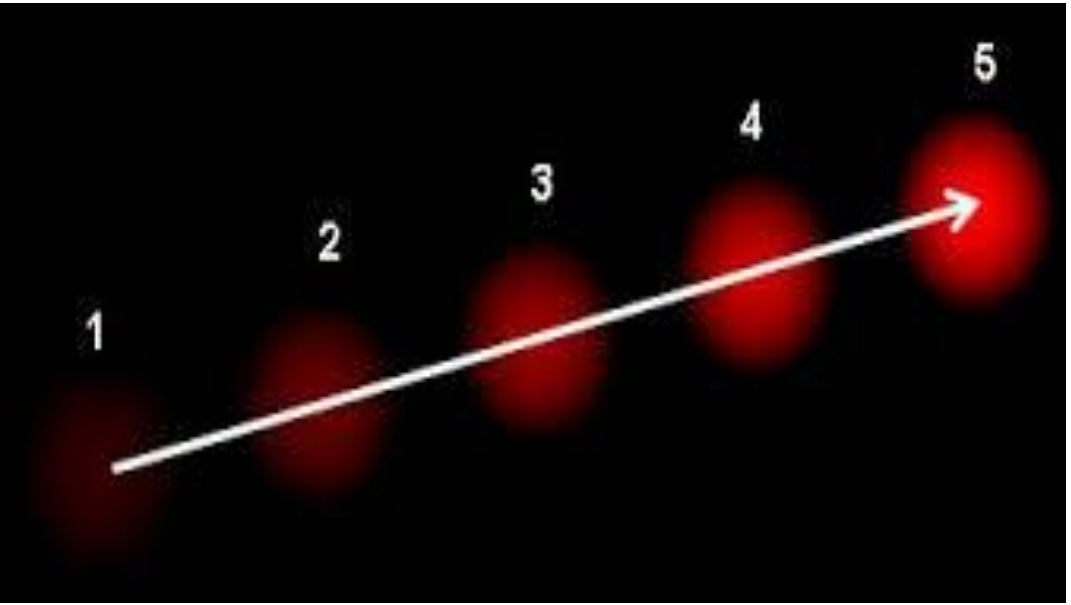
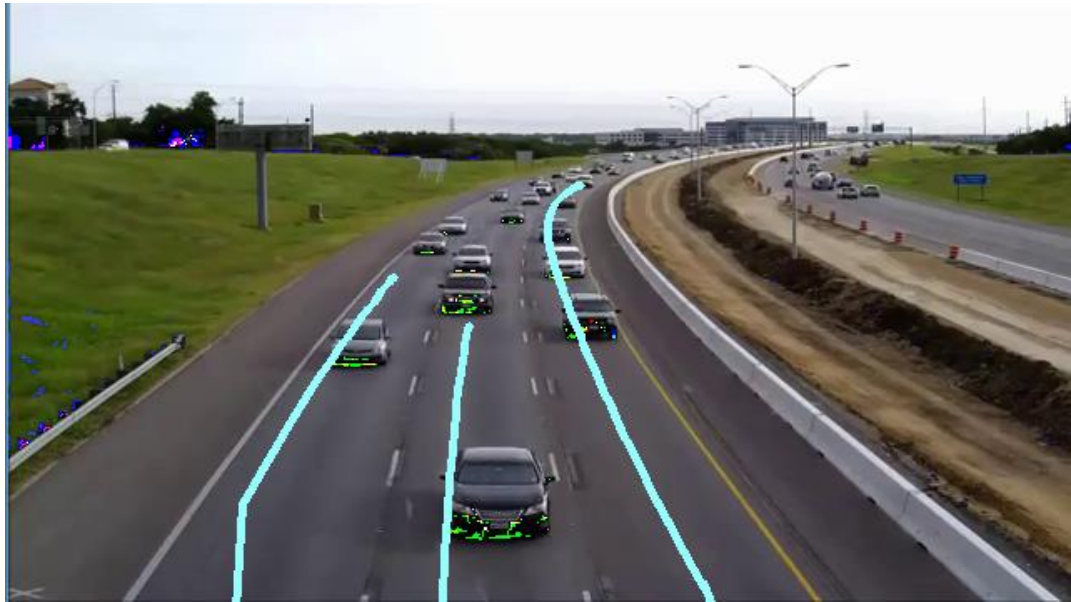
UNIT V

2D Motion Estimation

Part 2

By
B. Shiny Vidya
Assist. Prof





Contents

- **2D Motion Estimation:**
- Pixel Based Motion Estimation
- Block-Matching Algorithm
- Mesh-based Motion Estimation

Pixel Based Motion Estimation

- In pixel-based motion estimation, one tries to estimate a motion vector for every pixel.
- **Constant intensity assumption, for every pixel in the anchor frame**: there are many pixels in the tracked frame that have exactly the same image intensity.
- **Optical flow equation**: the problem is again indeterminate, because there is only one equation for two unknowns.
- To circumvent this problem, there are in general approaches.

  - **Regularization technique** to enforce some smoothness constraints on the motion field, so that the motion vector of a new pixel is constrained by those found for surrounding pixels.

  - **Assume the motion vectors in a neighborhood surrounding each pixel are the same**, and apply the constant intensity assumption or the optical flow equation over the entire neighborhood.

Pixel Based Motion Estimation

Regularization Using Motion Smoothness Constraint

Horn and Schunck proposed to estimate the motion vectors by minimizing the following objective function, which is a combination of the flow-based criterion and a motion smoothness criterion:

$$E\left(\mathbf{v}(\mathbf{x})\right) = \sum_{\mathbf{x}\in\Lambda} \left(\frac{\partial \psi}{\partial x} v_x + \frac{\partial \psi}{\partial y} v_y + \frac{\partial \psi}{\partial t} \right)^2 + w_s \left(\|\boldsymbol{\nabla} v_x\|^2 + \|\boldsymbol{\nabla} v_y\|^2 \right).$$

In their original algorithm, the spatial gradient of v_x and v_y are approximated by $\boldsymbol{\nabla} v_x = [v_x(x,y) - v_x(x-1,y), v_x(x,y) - v_x(x,y-1)]^T$, $\boldsymbol{\nabla} v_y = [v_y(x,y) - v_y(x-1,y), v_y(x,y) - v_y(x,y-1)]^T$. The minimization of the above error function is accomplished by a gradient-based method known as Gauss-Siedel method.

Pixel Based Motion Estimation

Using a Multipoint Neighborhood

In this approach, when estimating the motion vector at a pixel x_n, we assume that the motion vectors of all the pixels in a neighborhood $B(x_n)$ surrounding it are the same, being d_n.

- To determine $\mathbf{d_n}$, minimize the prediction error over $\mathbf{B(x_n)}$.
- **To estimate the motion vector d_n for x_n, we minimize the DFD error over $B(x_n)$:**

$$E_n(\mathbf{d}_n) = \frac{1}{2} \sum_{\mathbf{x} \in \mathcal{B}(\mathbf{x}_n)} w(\mathbf{x}) \left(\psi_2(\mathbf{x} + \mathbf{d}_n) - \psi_1(\mathbf{x})\right)^2 ,$$

where $w(\mathbf{x})$ are the weights assigned to pixel $\mathbf{x}$. Usually, the weight decreases as the distance from $\mathbf{x}$ to $\mathbf{x}_n$ increases.

The gradient with respect to d_n is

$$\mathbf{g}_n = \frac{\partial E_n}{\partial \mathbf{d}_n} = \sum_{\mathbf{x} \in \mathcal{B}(\mathbf{x}_n)} w(\mathbf{x}) e(\mathbf{x}, \mathbf{d}_n) \left. \frac{\partial \psi_2}{\partial \mathbf{x}} \right|_{\mathbf{x}+\mathbf{d}_n} ,$$

where $e(\mathbf{x}, \mathbf{d}_n) = \psi_2(\mathbf{x} + \mathbf{d}_n) - \psi_1(\mathbf{x})$ is the DFD at $\mathbf{x}$ with the estimate $\mathbf{d}_n$.

Pixel Based Motion Estimation

- With a pixel-based motion representation, one would need to **specify a MV for each pixel**, which is **very costly**.
- In **Pel-recursive motion estimation methods**, which are developed for video coding applications, the *MVs are obtained recursively*. Specifically, the MV at a current pixel is updated from those of its neighboring pixels that are coded before.
- Although pel-recursive methods are **very simple**, their motion estimation **accuracy is quite poor**. As a result, the **prediction error is still large**.

Block Matching Motion Estimation

- Smoothness constraints are imposed to regularize the problem in pixel based motion estimation.
- One way of imposing the smoothness constraint on the estimated motion field is to divide the image domain into non-overlapping small regions, called **blocks.**
- If the block is sufficiently *small*, then this model can be quite *accurate*.
- $\mathbf{B_m}$ represents the **m**-th image block, and **M = {1, 2, ….,M}.**

M = Total number of blocks,

The partition into blocks should satisfy

$$\bigcup_{m \in \mathcal{M}} \mathcal{B}_m = \Lambda, \quad \text{and} \quad \mathcal{B}_m \bigcap \mathcal{B}_n = \emptyset, m \neq n.$$

Block Matching Motion Estimation

- **Theoretically, a block can have any polygonal shape**. But in practice, **the square shape is used almost exclusively.**
- The **triangular shape** is more appropriate when the motion in each block is described by an **affine model**.

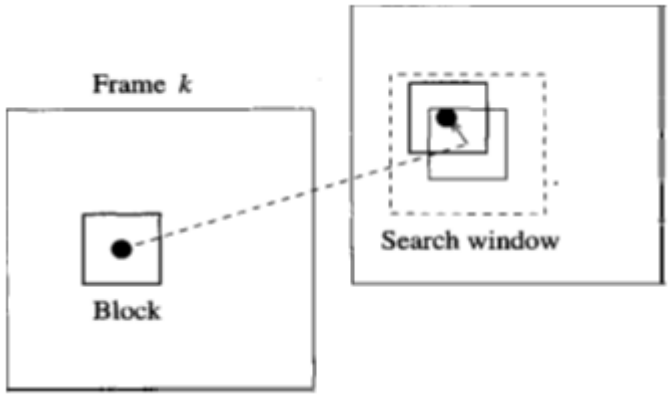


Fig: Block matching with square shape

In the simplest case, the motion in each block is assumed to be constant, i.e., the entire block undergoes a translation. This is called the **block-wise translational model**.

The motion estimation problem is to find a single MV for each block. This type of algorithm is collectively referred as **Block Matching Algorithm (BMA).**

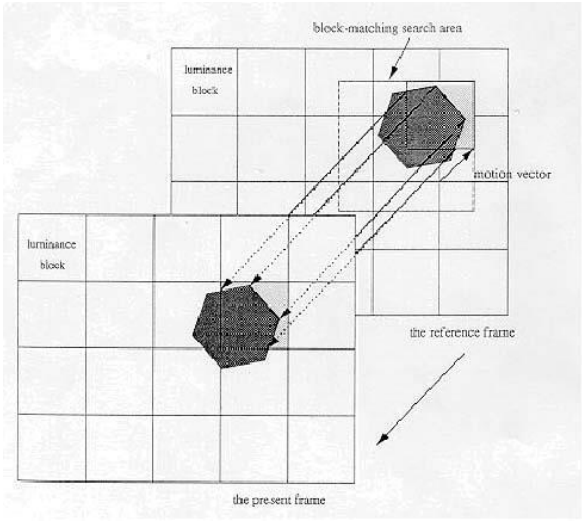


Fig: Block matching with hexagonal shape

Block Matching Motion Estimation

The Exhaustive Search Block Matching Algorithm (EBMA)

Given an image block in the anchor frame $\mathbf{B_m}$, the motion estimation problem at hand is to determine a matching block $\mathbf{B_m'}$ in the tracked frame so that the error between these two blocks is minimized.

The displacement vector $\mathbf{d_m}$ between the spatial positions of these two blocks (the center or a selected corner) is the MV of this block.

Under the block-wise translational model, $\mathbf{w(x;\ a) = x + d_m;\ x \epsilon B_m}$, the error equation can be written as:

$$E(\mathbf{d}_m, \forall m \in \mathcal{M}) = \sum_{m \in \mathcal{M}} \sum_{\mathbf{x} \in \mathcal{B}_m} |\psi_2(\mathbf{x} + \mathbf{d}_m) - \psi_1(\mathbf{x})|^p$$

Block Matching Motion Estimation

The Exhaustive Search Block Matching Algorithm (EBMA)

Because the estimated MV for a block only affects the prediction error in this block, one can estimate the MV for each block individually, by minimizing the prediction error accumulated over this block only, which is:

$$E_m(\mathbf{d}_m) = \sum_{\mathbf{x} \in \mathcal{B}_m} |\psi_2(\mathbf{x} + \mathbf{d}_m) - \psi_1(\mathbf{x})|^p$$

One way to determine the d_m that minimizes the above error is by using an exhaustive search and this method is called exhaustive block matching algorithm (EBMA).

As illustrated in Figure (in next slide) , the EBMA determines the optimal $\mathbf{d_m}$ for a given block $\mathbf{B_m}$ in the anchor frame by comparing it with all candidate blocks $\mathbf{B_m}$**'** in the tracked frame within predefined search region and finding the one with the minimum error.

Block Matching Motion Estimation

The Exhaustive Search Block Matching Algorithm (EBMA)

- The displacement between the two blocks is the estimated motion vector.
- To reduce the computational load, the MAD (Mean Absolute Difference) error (p = 1) is often used.
- The search region is usually symmetric with respect to the current block, up to R_x pixels to the left and right, and up to R_y pixels above and below, as illustrated in the figure.

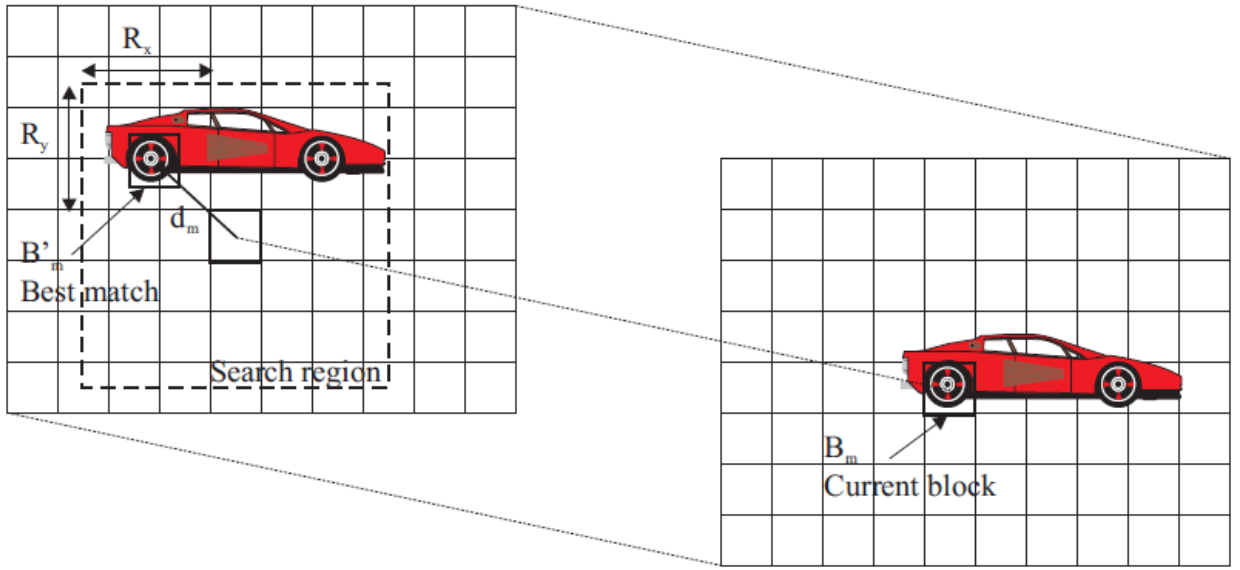


Fig. The search procedure of the exhaustive block matching algorithm.

For an image of size **M x M**, there are $\mathbf{(M/N)^2}$ blocks (assuming M is a multiple of N).
The total number of operations for a complete frame is then $\mathbf{M^2(2R+1)^2}$.
It is interesting to note that the overall computational load is independent of the block size N.

Mesh-based Motion Estimation

Mesh based Motion Representation

- With the mesh-based motion representation, the underlying image domain in the anchor frame is partitioned into **non-overlapping polygonal elements**.
- Each element is defined by a few nodes and links between the nodes, as shown in Fig. 6.14. Such a mesh is also known as a control grid.

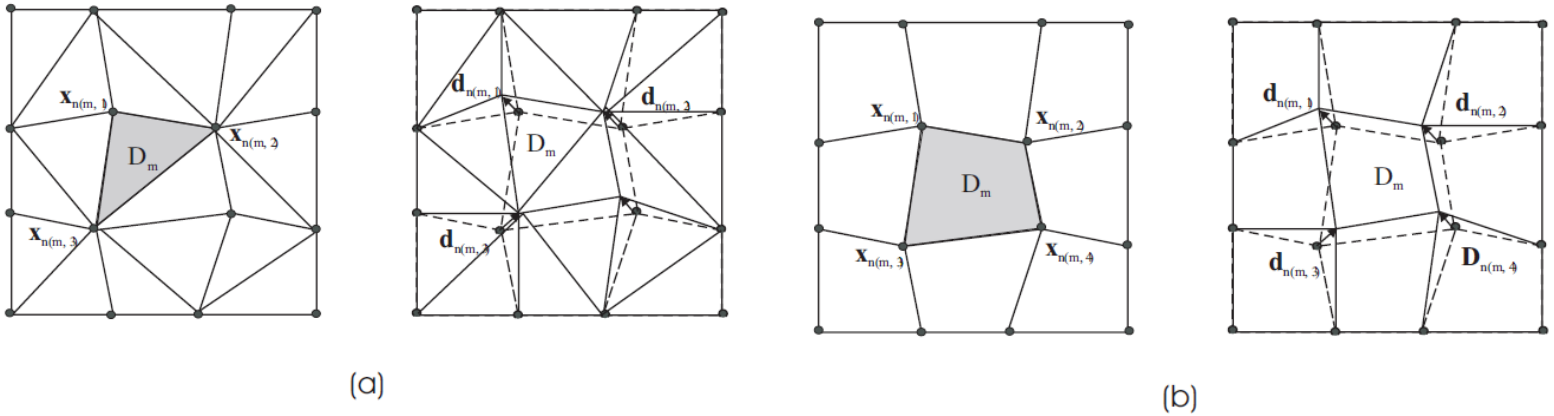


Fig: Illustration of mesh-based motion representation: (a) using a triangular mesh, with 3 nodes attached to each element, (b) using a quadrilateral mesh, with 4 nodes attached to each element. In the shown example, the two meshes have the same number of nodes, but the triangular mesh has twice the number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

In the mesh-based representation, the motion field over the entire frame is described by MVs at the nodes only. The MVs at the interior points of an elements are interpolated from the MVs at the nodes of this element. The nodal MVs are constrained so that the nodes in the tracked frame still form a feasible mesh, with no flip-over elements.

Mesh-based Motion Estimation

Mesh based Motion Representation

- Let the number of **elements** and **nodes** be denoted by **M** and **N** respectively, and the **number of nodes defining each element by K.**
- For convenience, we define the following index sets:

  M = {1, 2, …, M} ; N = {1, 2,…,N}; K = {1, 2,…,K}
- Let the **m-th** element and **n-th** node in frame **t** (t = 1 for the anchor frame and t = 2 for the tracked frame) be denoted by $\mathbf{B_{t,m}}$, m ϵ M and $x_{t,n}$; n ϵ N; and the MV of the n-th node by $\mathbf{d_n = x_{2,n} - x_{1,n}}$.
- The motion field in element $\mathbf{B_{1,m}}$ is related to the nodal MVs $\mathbf{d_n}$ by:

$$\mathbf{d}_m(\mathbf{x}) = \sum_{k \in \mathcal{K}} \phi_{m,k}(\mathbf{x}) \mathbf{d}_{n(m,k)}, \quad \mathbf{x} \in \mathcal{B}_{1,m},$$

where **n(m; k) species the global index of the k-th node in the m-th element**
The function **m;k(x) is the interpolation kernel** associated with node k in element m. It depends on the desired contribution of the k-th node in $B_{1,m}$ to the MV at x. To guarantee continuity across element boundary, the interpolation kernels should satisfy:

$$0 \leq \phi_{m,k}(\mathbf{x}) \leq 1, \sum_k \phi_{m,k}(\mathbf{x}) = 1, \forall \mathbf{x} \in \mathcal{B}_m, \qquad \text{and} \qquad \phi_{m,k}(\mathbf{x}_l) = \delta_{k,l} = \begin{cases} 1 & k = l, \\ 0 & k \neq l. \end{cases}$$

Mesh-based Motion Estimation

Motion Estimation using Mesh-based model

With the mesh-based motion representation, there are in general two sets of problems to be solved:

1) Given a mesh or equivalently nodes in the current frame, how to determine the nodal positions in the tracked frame. This is essentially a motion estimation problem.
2) How to set up the mesh in the anchor frame, so that the mesh conforms to the object boundaries.

- Motion parameters can be estimated by error minimization approach.
- Under the mesh-based motion model, the DFD error is

$$E(\mathbf{d}_n, n \in \mathcal{N}) = \sum_{m \in \mathcal{M}} \sum_{\mathbf{x} \in \mathcal{B}_{1,m}} |\psi_2(\mathbf{w}_m(\mathbf{x})) - \psi_1(\mathbf{x})|^p$$

where

$$\mathbf{w}_m(\mathbf{x}) = \mathbf{x} + \sum_{k \in \mathcal{K}} \phi_{m,k}(\mathbf{x}) \mathbf{d}_{n(m,k)}, \quad \mathbf{x} \in \mathcal{B}_{1,m}$$

We have assumed that a *single mesh* is generated (or propagated from the previous frame in the forward tracking case) for the entire current frame.
Every node in this mesh is tracked to one and only one node in the tracked frame, so that the nodes in the tracked frame still form a mesh that covers the entire frame.