Motion Estimation
Outline

• 2-D motion vs. optical flow
• General methodologies in motion estimation
  • Motion representation
  • Motion estimation criterion
  • Optimization methods
  • Gradient descent methods
• Pixel-based motion estimation
• Block-based motion estimation
• Multiresolution motion estimation
• Deformable block matching algorithm (DBMA)
• Mesh-based motion estimation
• Global motion estimation
• Region-based motion estimation
Reading Material

• A. M. Tekalp, Digital video processing, Prentice Hall, 2015
  • Chapter 4.1, 4.2
  • Chapter 4.3, 4.4, 4.5, 4.6

  • Chapter 5.1, 5.3.2, 5.5
  • Chapter 6.1 – 6.4 (skip 6.4.5, 6.4.6), 6.7, 6.9
  • Appendix A and B: Gradients and steepest descent

• R. Szeliski, Computer Vision: Algorithms and Applications, Springer 2010, Chapter 8
Summary (last class)

- 3D Motion
  - Rigid vs. non-rigid motion
- Camera model: 3D $\rightarrow$ 2D projection
  - Perspective projection vs. orthographic projection
- What causes 2D motion?
  - Object motion projected to 2D
  - Camera motion
- Models corresponding to typical camera motion and object motion
  - Piece-wise projective mapping is a good model for projected rigid object motion
  - Can be approximated by affine or bilinear functions
  - Affine functions can also characterize some global camera motions
- Ways to represent motion
  - Pixel-based, block-based, region-based, global, etc.
Motion Field Corresponding to Different 2-D Motion Models

Translation

Affine

Bilinear

Projective
Region of Support for Motion Representation

Global:
Entire motion field is represented by a few global parameters.

Pixel-based:
One MV at each pixel, with some smoothness constraint between adjacent MVs.

Block-based:
Entire frame is divided into blocks, and motion in each block is characterized by a few parameters.

Region-based:
Entire frame is divided into regions, each region corresponding to an object or sub-object with consistent motion, represented by a few parameters.
Motion Estimation Algorithms

• Motion representation
  • Pixel-based, block-based, mesh, global motion, …

• Optimization criteria
  • Minimize displaced frame difference, optical flow, while subject to constraints, …

• Optimization strategies
  • Gradient descent, exhaustive search, …
Motion Estimation Outline

• 2D motion and optical flow
• Motion estimation
  • General methodologies
  • Pixel-based
  • Block-based
2-D Motion vs. Optical Flow

- 2-D Motion: Projection of 3-D motion, depending on 3D object motion and projection operator
- Optical flow: “Perceived” 2-D motion based on changes in image pattern, also depends on illumination and object surface texture

On the left, a sphere is rotating under a constant ambient illumination, but the observed image does not change => optical flow is zero, but the motion field is not

On the right, a point light source is rotating around a stationary sphere, causing the highlight point on the sphere to rotate => motion field is zero, but the optical flow is not
Optical Flow Equation (1)

• When illumination condition is unknown, the best one can do it to estimate optical flow

• Constant intensity assumption -> Optical flow equation

Under “constant intensity assumption”:

\[ \psi(x + d_x, y + d_y, t + d_t) = \psi(x, y, t) \]

Taylor’s expansion, when \(d_x, d_y, d_t\) are small (ignore higher order terms):

\[ \psi(x + d_x, y + d_y, t + d_t) = \psi(x, y, t) + \frac{\partial \psi}{\partial x} d_x + \frac{\partial \psi}{\partial y} d_y + \frac{\partial \psi}{\partial t} d_t \]

Compare the above two equations, we have the optical flow equation:

\[ \frac{\partial \psi}{\partial x} d_x + \frac{\partial \psi}{\partial y} d_y + \frac{\partial \psi}{\partial t} d_t = 0 \quad \text{or} \quad \frac{\partial \psi}{\partial x} v_x + \frac{\partial \psi}{\partial y} v_y + \frac{\partial \psi}{\partial t} = 0 \]

Assuming \(d_t\) is small, so that \(v_x = d_x / d_t\)
Optical Flow Equation (2)

• Constant intensity assumption

• Apply chain rule

\[
\frac{d\psi(x, y, t)}{dt} = 0
\]

\[
\frac{\partial \psi}{\partial x} \frac{dx}{dt} + \frac{\partial \psi}{\partial y} \frac{dy}{dt} + \frac{\partial \psi}{\partial t} = \frac{\partial \psi}{\partial x} v_x + \frac{\partial \psi}{\partial y} v_y + \frac{\partial \psi}{\partial t} = 0
\]

Two approaches, same assumption, same answer
Optical Flow Equation (3)

\[
\frac{\partial \psi}{\partial x} v_x + \frac{\partial \psi}{\partial y} v_y + \frac{\partial \psi}{\partial t} = 0 \quad \text{or} \quad \nabla \psi^T v + \frac{\partial \psi}{\partial t} = 0
\]

\[
\nabla \psi = \begin{bmatrix} \frac{\partial \psi}{\partial x}, \frac{\partial \psi}{\partial y} \end{bmatrix}^T
\]

is the spatial gradient vector

- Can only estimate motion in direction of the spatial gradient
- Applies to a single point only
Ambiguities in Motion Estimation

- Optical flow equation only constrains the flow vector in the gradient direction $v_n$.
- The flow vector in the tangent direction ($v_t$) is under-determined $\Rightarrow$ any point on the tangent line is a solution to the optical flow equation.
- In regions with constant brightness ($\nabla \psi = 0$), the flow is indeterminate $\Rightarrow$ Motion estimation is unreliable in regions with flat texture, more reliable near edges.

\[
v = v_n e_n + v_t e_t
\]

$v_n \|\nabla \psi\| + \frac{\partial \psi}{\partial t} = 0$

$e_n$: normal direction
$e_t$: tangent direction
Inaccuracies of the Flow Equation

• Object boundaries
  • Motion estimation more reliable around strong edges, but strong edges are likely to be where two objects move differently

• Occlusion
  • No correspondence exists for (un)covered background

• Constant brightness
  • Flow vector is indeterminate, because no perceived brightness changes when the surface has flat pattern
  • Consistent with the relation between spatial and temporal frequency discussed earlier
Aperture Problem in Motion Estimation

• “Aperture” means the small window over which to apply the constant intensity assumption
• Motion at $X_1$ cannot be determined due to only one spatial gradient direction in aperture 1
• Motion at $X_2$ can be determined accurately, because the image has gradient in two different directions in aperture 2
Two categories of approaches for Motion Estimation

- **Feature based** (more often used in object tracking, 3D reconstruction from 2D)
  - Find motion only for sparse points
  - Impose a motion model to estimate a dense field
- **Intensity based** (based on constant intensity assumption)
  - More often used for motion compensated prediction
  - Required in video coding, frame interpolation
General Considerations for Motion Estimation

• Three important questions
  • How to represent the motion field?
    (ex: dense or sparse? Region or )
  • What optimization criteria to use to estimate motion parameters?
    • Depends on the application; compression minimize average prediction error; motion-compensated interpolation minimize maximum interpolation error
  • How to search for the best motion parameters?
Mix-and-match

- Motion representation
- Optimization criteria
- Optimization strategies

Examples:
- Pixel-based representation, DFD, gradient descent
- Pixel-based representation, OF, least-square
- Block-based representation, DFD, exhaustive search
- Block-based representation, DFD, hierarchical search
- Mesh representation, DFD, iterative search
- Global motion
Notations

Anchor frame: $\psi_1(x)$
Tracked frame: $\psi_2(x)$
Motion parameters: $a$
Motion field: $d(x; a), x \in \Lambda$
Mapping function: $w(x; a) = x + d(x; a), x \in \Lambda$
Motion Estimation Criteria

- Minimize the displaced frame difference (DFD)
  \[ E_{\text{DFD}}(\mathbf{a}) = \sum_{x \in \Lambda} |\psi_2(x + \mathbf{d}(x; \mathbf{a})) - \psi_1(x)|^p \rightarrow \min \]

  \( p = 1: \text{MAD}; \quad P = 2: \text{MSE} \)

- Satisfy the optical flow (OF) equation
  \[ E_{\text{OF}}(\mathbf{a}) = \sum_{x \in \Lambda} \left| (\nabla \psi_1(x))^T \mathbf{d}(x; \mathbf{a}) + \psi_2(x) - \psi_1(x) \right|^p \rightarrow \min \]

- Impose additional smoothness constraint using regularization technique
  (Important in pixel- and block-based representation)
  \[ E_s(\mathbf{a}) = \sum_{x \in \Lambda} \sum_{y \in N_x} \| \mathbf{d}(x; \mathbf{a}) - \mathbf{d}(y; \mathbf{a}) \|^2 \]

  \[ w_{DFD} E_{\text{DFD}}(\mathbf{a}) + w_s E_s(\mathbf{a}) \rightarrow \min \]

- Bayesian (MAP) criterion: to maximize the a posteriori probability
  \[ P(D = \mathbf{d} | \psi_2, \psi_1) \rightarrow \max \]
Relation Among Different Criteria

• OF criterion is good only if motion is small
• OF criterion can often yield closed-form solution as the objective function is quadratic in MVs
• When the motion is not small, can iterate the solution based on the OF criterion to satisfy the DFD criterion
• Bayesian criterion can be reduced to the DFD criterion plus motion smoothness constraint
• More in the textbook (Wang, Chapter 6.2.2)
Optimization Strategies to Find Min. or Max.

• Exhaustive search
  • Typically used for the DFD criterion with $p=1$ (MAD)
  • Guarantees reaching the global optimal
  • Computation required may be unacceptable when there are many parameters to search simultaneously!
  • Fast search algorithms reach sub-optimal solution in shorter time

• Gradient-based search
  • Typically used for the DFD or OF criterion with $p=2$ (MSE)
    • the gradient can often be calculated analytically
    • When used with the OF criterion, closed-form solution may be obtained
  • Reaches the local optimal point closest to the initial solution

• Multi-resolution search
  • Search from coarse to fine resolution, faster than exhaustive search
  • Less likely to be trapped into a local minimum
High-level Framework

• Motion representation
• Optimization criteria
• Optimization strategies

• Mix and match
  • Pixel-based representation, DFD, gradient descent
  • Pixel-based representation, OF, least-squares
  • Block-based representation, DFD, exhaustive search
  • Block-based representation, DFD, hierarchical search
  • Mesh representation, DFD, iterative search
  • Global motion
Block Matching Algorithm

- **Overview**
  - Assume all pixels in a block undergo a translation, denoted by a single MV
  - Estimate the MV for each block independently, by minimizing the DFD error over this block
  - Discrete search for optical flow

- **Minimizing function**

\[
E_{DFD}(d_m) = \sum_{x \in B_m} |\psi_2(x + d_m) - \psi_1(x)|^p \rightarrow \min
\]

- **Optimization method**
  - Exhaustive search (feasible as one only needs to search one MV for all pixels in the block), using MAD criterion (p=1)
  - Fast search algorithms
  - Integer vs. fractional pel accuracy search
Exhaustive Block Matching Algorithm (EBMA)
Complexity of Integer-Pel EBMA

- Assumption
  - Image size: MxM
  - Block size: NxN
  - Search range: (-R,R) in each dimension
  - Search stepsize: 1 pixel (assuming integer MV)

- Operation counts (1 operation=1 “-”, 1 “+”, 1 “*”):
  - Each candidate position: $N^2$
  - Each block going through all candidates: $(2R+1)^2 N^2$
  - Entire frame: $(M/N)^2 (2R+1)^2 N^2 = M^2 (2R+1)^2$
    - Independent of block size!

- Example: M=512, N=16, R=16, 30 fps
  - Total operation count = $2.85 \times 10^8$/frame = $8.55 \times 10^9$/second

- Regular structure suitable for VLSI implementation
- Software-only implementation slow
Pseudo-code/Matlab Script for Integer-pel EBMA

%f1: anchor frame; f2: target frame, fp: predicted image;
%mvx,mvy: store the MV image
%widthxheight: image size; N: block size, R: search range

for ii=1:N:height-N,
    for jj=1:N:width-N  %for every block in the anchor frame
        MAD_min=256*N*N;mvx=0;mvy=0;
        for kk=-R:1:R,
            for ll=-R:1:R  %for every search candidate
                MAD=sum(sum(abs(f1(ii:ii+N-1,jj:jj+N-1)-f2(ii+kk:ii+kk+N-1,jj+ll:jj+ll+N-1))));
                % calculate MAD for this candidate
                if MAD<MAX_min
                    MAD_min=MAD,dy=kk,dx=ll;
                end;
            end;
        end;
        fp(ii:ii+N-1,jj:jj+N-1)= f2(ii+dy:ii+dy+N-1,jj+dx:jj+dx+N-1);  %put the best matching block in the predicted image
        iblk=(floor)(ii-1)/N+1; jblk=(floor)(jj-1)/N+1; %block index
        mvx(iblk,jblk)=dx; mvy(iblk,jblk)=dy; %record the estimated MV
    end;
end;

Note: A real working program needs to check whether a pixel in the candidate matching block falls outside the image boundary and such pixel should not count in MAD. This program is meant to illustrate the main operations involved. Not the actual working matlab script.
Fractional Accuracy EBMA

• Real MV may not always be multiples of pixels. To allow sub-pixel MV, the search stepsize must be less than 1 pixel

• Half-pel EBMA: stepsize=1/2 pixel in both dimension

• Difficulty:
  • Target frame only have integer pels

• Solution:
  • Interpolate the target frame by factor of two before searching
  • Bilinear interpolation is typically used

• Complexity:
  • 4 times of integer-pel, plus additional operations for interpolation

• Fast algorithms:
  • Search in integer precisions first, then refine in a small search region in half-pel accuracy
Half-Pel Accuracy EBMA
Bilinear Interpolation

\[
\begin{align*}
O[2x,2y] &= I[x,y] \\
O[2x+1,2y] &= (I[x,y]+I[x+1,y])/2 \\
O[2x,2y+1] &= (I[x,y]+I[x,y+1])/2 \\
O[2x+1,2y+1] &= (I[x,y]+I[x+1,y]+I[x,y+1]+I[x+1,y+1])/4
\end{align*}
\]
Example: Half-pel EBMA
Problems with EBMA

• Motion field is chaotic
  • Each block’s motion vector is computed independently
  • Many possible matches, especially in smooth regions

• DFD is not uniformly small within block
  • Poor motion model:
    • Block may contain multiple motions
    • Block does not undergo translation
    • Illumination changes

• DFD is not uniformly small across block boundaries
  • Poor motion model: Adjacent pixels can have very different motions

• Slow
Minimizing Problems with EBMA

• Motion field is chaotic
  • Use hierarchical search
  • Impose smoothness constraints (including mesh-based model)

• DFD is not uniformly small within block
  • Improve motion model (Deformable and mesh-based models; region-based estimation; compensate for variable illumination)

• DFD is not uniformly small across block boundaries
  • Mesh-based motion models; compute pixel-based motion

• Slow
  • Use fast algorithms and hierarchical search
Fast Algorithms for BMA

• Key idea to reduce the computation in EBMA:
  • Reduce # of search candidates:
    • Only search for those that are likely to produce small errors
    • Predict possible remaining candidates, based on previous search result
  • Reduce the computation for each candidate by simplifying the DFD error measure
    • Subsample and don’t compute DFD on all possible pixels

• Classical fast algorithms
  • Three-step
  • 2D-log
  • Conjugate direction

• Many new fast algorithms have been developed since then
  • Some suitable for software implementation, others for VLSI implementation (memory access, etc.)
Multi-resolution Motion Estimation

• Problems with BMA
  • Unless exhaustive search is used, the solution may not be global minimum
  • Exhaustive search requires extremely large computation
  • Block wise translation motion model is not always appropriate

• Multiresolution approach
  • Aim to solve the first two problems
  • First: Estimate the motion in a coarse resolution over low-pass filtered, down-sampled image pair
    • Can usually lead to a solution close to the true motion field
  • Second: Modify the initial solution in successively finer resolution within a small search range
    • Reduce the computation
  • Can be applied to different motion representations, but we will focus on its application to BMA
Hierarchical Block Matching Algorithm (HBMA)
Example: Three-level HBMA
Computation Requirement of HBMA

• Definitions
  • Image size: MxM; Block size: NxN at every level; Levels: L
  • Search range:
    • 1st level: R/2^(L-1) (Equivalent to R in L-th level)
    • Other levels: R/2^(L-1) (could be smaller – since motion error is likely to be small)

• Operation counts for EBMA
  • Image size M, block size N, search range R
  • # operations: \( M^2 (2R + 1)^2 \)

• Operation counts at L-th level (Image size: M/2^(L-l))
  \[
  \left( \frac{M}{2^{L-l}} \right)^2 \left( \frac{2R}{2^{L-1}} + 1 \right)^2
  \]

• Total operation count
  \[
  \sum_{l=1}^{L} \left( \frac{M}{2^{L-l}} \right)^2 \left( \frac{2R}{2^{L-1}} + 1 \right)^2 \approx \frac{1}{3} 4^{-(L-2)} 4M^2 R^2
  \]

• Saving factor: \( 3 \cdot 4^{(L-2)} = 3(L = 2); 12(L = 3) \)
Summary (so far this class)

• Constraints for 2D motion
  • Optical flow equation
  • Derived from constant intensity and small motion assumption
  • Ambiguity in motion estimation

• Estimation criterion:
  • DFD (constant intensity)
  • OF (constant intensity + small motion)

• Search method:
  • Exhaustive search, gradient-descent, multi-resolution

• Pixel-based motion estimation
  • Most accurate representation, but also most costly to estimate

• Block-based motion estimation
  • Good trade-off between accuracy and speed
  • EBMA and its fast but suboptimal variants are widely used in video coding for motion-compensated temporal prediction.
Optimization Strategies to Find Min. or Max.

• Exhaustive search
  • Typically used for the DFD criterion with p=1 (MAD)
  • Guarantees reaching the global optimal
  • Computation required may be unacceptable when there are many parameters to search simultaneously!
  • Fast search algorithms reach sub-optimal solution in shorter time

• Gradient-based search
  • Typically used for the DFD or OF criterion with p=2 (MSE)
    • the gradient can often be calculated analytically
    • When used with the OF criterion, closed-form solution may be obtained
  • Reaches the local optimal point closest to the initial solution

• Multi-resolution search
  • Search from coarse to fine resolution, faster than exhaustive search
  • Less likely to be trapped into a local minimum
Gradient Descent Method

• Iteratively update the current estimate in the direction opposite the gradient direction

\[ x^{(t+1)} = x^{(t)} - \alpha \left. \frac{\partial J}{\partial x} \right|_{x^{(t)}} \]

• The solution depends on the initial condition. Reaches the local minimum closest to the initial condition

• Choice of step size:
  • Fixed stepsize: Stepsize must be small to avoid oscillation, requires many iterations
  • Steepest gradient descent (adjust stepsize optimally)
Newton’s Method

- Newton’s method

\[ x^{(l+1)} = x^{(l)} - \alpha [H(x^{(l)})]^{-1} \frac{\partial J}{\partial x} \bigg|_{x^{(l)}} \]

\[ [H(x)] = \frac{\partial^2 J}{\partial x^2} = \begin{bmatrix}
    \frac{\partial^2 J}{\partial x_1 \partial x_1} & \frac{\partial^2 J}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 J}{\partial x_1 \partial x_K} \\
    \frac{\partial^2 J}{\partial x_2 \partial x_1} & \frac{\partial^2 J}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 J}{\partial x_2 \partial x_K} \\
    \vdots & \vdots & \ddots & \vdots \\
    \frac{\partial^2 J}{\partial x_K \partial x_1} & \frac{\partial^2 J}{\partial x_K \partial x_2} & \cdots & \frac{\partial^2 J}{\partial x_K \partial x_K}
\end{bmatrix} \]

- Converges faster than 1st order method (i.e. requires fewer number of iterations to reach convergence)
- Requires more calculation in each iteration
- More prone to noise (gradient calculation is subject to noise, more so with 2nd order than with 1st order)
- May not converge if \( a \geq 1 \). Should choose a appropriate to reach a good compromise between guaranteeing convergence and the convergence rate
Newton-Raphson Method

- Newton-Ralphson method
  - Approximate $2^{nd}$ order gradient with product of $1^{st}$ order gradients
  - Applicable when the objective function is a sum of squared errors
  - Only needs to calculate $1^{st}$ order gradients, yet converge at a rate similar to Newton’s method.

$$J(x) = \frac{1}{2} \sum_k e_k^2(x).$$

$$\frac{\partial J}{\partial x} = \sum \frac{\partial e_k}{\partial x} e_k(x)$$

$$[H] = \frac{\partial^2 J}{\partial x^2} = \sum \frac{\partial e_k}{\partial x} \left( \frac{\partial e_k}{\partial x} \right)^T + \frac{\partial^2 e_k}{\partial x^2} e_k(x) \approx \sum \frac{\partial e_k}{\partial x} \left( \frac{\partial e_k}{\partial x} \right)^T$$

$$x^{(t+1)} = x^{(t)} - \alpha [H(x^{(t)})]^{-1} \frac{\partial J}{\partial x} \bigg|_{x^{(t)}}$$
Reminder: high-level framework

• Motion representation
• Optimization criteria
• Optimization strategies

• Mix and match
  • Pixel-based representation, DFD, gradient descent
  • Pixel-based representation, OF, least-squares
  • Block-based representation, DFD, exhaustive search
  • Block-based representation, DFD, hierarchical search
  • Mesh representation, DFD, iterative search
  • Global motion
Pixel-Based Motion Estimation

- Multipoint neighborhood method
  - Assuming every pixel in a small block surrounding a pixel has the same MV

- Horn-Schunck (1981) method
  - OF + smoothness criterion

- Pel-recursive method
  - MV for a current pel is updated from those of its previous pels, so that the MV does not need to be coded
  - Developed for early generation of video coder
Multipoint Neighborhood Method

• Estimate the MV at each pixel independently, by minimizing the optimization criterion over a neighborhood surrounding this pixel

• Every pixel in the neighborhood B(x) is assumed to have the same MV

• Case 1: Gradient descent with DFD criterion

\[
E_{DFD}(d_n) = \sum_{x \in B(x_n)} w(x)|\psi_2(x + d_n) - \psi_1(x)|^2 \rightarrow \min
\]

• Case 2: Least-squares with OF criterion

\[
E_{OF}(d_n) = \sum_{x \in B(x_n)} w(x)|(\nabla \psi_1(x))^T d_n + \psi_2(x) - \psi_1(x)|^2 \rightarrow \min
\]
Example: Gradient Descent Method

\[ E_{DFD}(d_n) = \sum_{x \in B(x_n)} w(x)|\psi_2(x + d_n) - \psi_1(x)|^2 \rightarrow \min \]

\[ g(d_n) = \frac{\partial E}{\partial d_n} = \sum_{x \in B(x_n)} w(x) e(x + d_n) \frac{\partial \psi_2}{\partial x} \bigg|_{x+d_n} \]

First order gradient descent:

\[ d_n^{(l+1)} = d_n^{(l)} - \alpha g(d_n^{(l)}) \]
Example: Gradient Descent Method

\[ E_{DFD}(d_n) = \sum_{x \in B(x_n)} w(x)[\psi_2(x + d_n) - \psi_1(x)]^2 \rightarrow \min \]

\[ g(d_n) = \frac{\partial E}{\partial d_n} = \sum_{x \in B(x_n)} w(x)e(x + d_n) \frac{\partial \psi_2}{\partial x} \bigg|_{x+d_n} \]

\[ [H(d_n)] = \frac{\partial^2 E}{\partial d_n^2} = \sum_{x \in B(x_n)} w(x) \frac{\partial \psi_2}{\partial x} \left( \frac{\partial \psi_2}{\partial x} \right)^T \bigg|_{x+d_n} + w(x)e(x + d_n) \frac{\partial^2 \psi_2}{\partial x^2} \bigg|_{x+d_n} \]

\[ \approx \sum_{x \in B(x_n)} w(x) \frac{\partial \psi_2}{\partial x} \left( \frac{\partial \psi_2}{\partial x} \right)^T \bigg|_{x+d_n} \]

First order gradient descent:

\[ d_n^{(l+1)} = d_n^{(l)} - \alpha \ g(d_n^{(l)}) \]

Newton - Raphson method:

\[ d_n^{(l+1)} = d_n^{(l)} - \alpha \left[ H(d_n^{(l)}) \right]^{-1} g(d_n^{(l)}) \]
Least-Square Solution: OF Criterion

\[ E_{OF}(d_n) = \sum_{x \in B(x_n)} w(x)[(\nabla \psi_1(x))^T d_n + \psi_2(x) - \psi_1(x)]^2 \rightarrow \min \]

\[ \frac{\partial E}{\partial d_n} = 2 \sum_{x \in B(x_n)} w(x) (\nabla \psi_1(x))^T d_n + \psi_2(x) - \psi_1(x) \nabla \psi_1(x) = 0 \quad \text{Unique minimum} \]

\[ d_{n, \text{opt}} = \left[ \sum_{x \in B(x_n)} w(x) \nabla \psi_1(x) (\nabla \psi_1(x))^T \right]^{-1} \left[ \sum_{x \in B(x_n)} w(x) (\psi_1(x) - \psi_2(x)) \nabla \psi_1(x) \right] \]

The solution is good only if the actual MV is small. When this is not the case, one should iterate the above solution, with the following update:

\[ \psi_2^{(l+1)}(x) = \psi_2(x + d_n^{(l)}) \]

\[ d_n^{(l+1)} = d_n^{(l)} + \Delta_n^{(l+1)} \]

where \( \Delta_n^{(l+1)} \) denote the MV found at that iteration

Intuitively, this takes the target image and shifts it by the best known vector. This makes the small-motion approximation more valid.
Horn and Schunck (1981)

• Pixel-based motion
• Combine flow equation with smooth-motion constraint

\[
\sum_{x \in \Lambda} \left( \frac{\partial \psi}{\partial x} v_x + \frac{\partial \psi}{\partial y} v_y + \frac{\partial \psi}{\partial t} \right)^2 + w_s \left( \| \nabla v_x \|^2 + \| \nabla v_y \|^2 \right)
\]

• All gradients approximated with local differences
• Iterate; eventually information from regions with strong gradient infiltrate both
  • Into regions with nearly zero gradient
  • Across object boundaries
High-level Framework

- Motion representation
- Optimization criteria
- Optimization strategies
- Mix and match
  - Pixel-based representation, DFD, gradient descent
  - Pixel-based representation, OF, least-square
  - Block-based representation, DFD, exhaustive search
  - Block-based representation, DFD, hierarchical search
  - Deformable block and mesh representations
  - Global motion
Deformable Block Matching Algorithm
Overview of DBMA

• Three steps:
  • Partition the anchor frame into regular blocks
  • Model the motion in each block by a more complex motion
    • The 2-D motion caused by a flat surface patch undergoing rigid 3-D motion can be approximated well by projective mapping
    • Projective Mapping can be approximated by affine mapping and bilinear mapping
    • Various possible mappings can be described by a node-based motion model
  • Estimate the motion parameters block by block independently
    • Discontinuity problem cross block boundaries still remain
  • Still cannot solve the problem of multiple motions within a block or changes due to illumination effect!
Affine and Bilinear Model

- **Affine (6 parameters):**
  - Good for mapping triangles to triangles
  \[
  \begin{bmatrix}
  d_x(x, y) \\
  d_y(x, y)
  \end{bmatrix} =
  \begin{bmatrix}
  a_0 + a_1x + a_2y \\
  b_0 + b_1x + b_2y
  \end{bmatrix}
  \]

- **Bilinear (8 parameters):**
  - Good for mapping blocks to quadrangles
  \[
  \begin{bmatrix}
  d_x(x, y) \\
  d_y(x, y)
  \end{bmatrix} =
  \begin{bmatrix}
  a_0 + a_1x + a_2y + a_3xy \\
  b_0 + b_1x + b_2y + b_3xy
  \end{bmatrix}
  \]
Representing Deformable Blocks

1. Represent with polynomial coefficients of the model

2. Represent with nodal motion
   • Motion vector for each node
   • Interpolation kernel for pixels inside node (kernel depends on the motion model)

   • Advantages of second representation
     • More efficient communication (polynomial coefficients need high precision)
     • Easier to define both search range and search stepsize
     • All parameters are equally important
     • All parameters need same degree of precision
Problems with DBMA

• Motion discontinuity across block boundaries, because nodal MVs are estimated independently from block to block
  • Fix: mesh-based motion estimation
  • First apply EBMA to all blocks

• Cannot do well on blocks with multiple moving objects or changes due to illumination effect
  • Three mode method
    • First apply EBMA to all blocks
    • Blocks with small EBMA errors have *translational motion*
    • Blocks with large EBMA errors may have *non-translational motion*
      • First apply DBMA to these blocks
      • Blocks still having errors are *non-motion compensable*
Mesh-Based Motion Estimation

Partition frame into non-overlapping polygons

Describe using nodal motion
- Motion of the nodes
- Interpolation kernel

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

When they move, nodes must stay in a mesh
Mesh-based vs. Block-based Motion Estimation

(a) Block-based backward ME

(b) Mesh-based backward ME

(c) Mesh-based forward ME
Mesh Generation and Motion Estimation

• Two problems:
  • Given a mesh in the anchor frame, determine nodal positions in the target frame – Motion estimation
  • Set up the mesh in the anchor frame, so that the mesh conforms with object boundaries – Mesh generation
    • Backward ME: can use either regular mesh or object adaptive mesh at each new frame
      • Motion estimation is easier with a regular mesh, but adaptive mesh can yield more accurate result
    • Forward ME:
      • Only need to establish a mesh for the initial frame. Meshes in the following frames depend on the nodal MVs between successive frames
      • To accommodate appearing/disappearing objects, the mesh geometry needs to be updated
Example: Half-pel EBMA
Mesh-based method (29.72dB)

EBMA (29.86dB)

EBMA vs. Mesh-based Motion Estimation
Global Motion Estimation

- Global motion:
  - Camera moving over a stationary scene
    - Most projected camera motions can be captured by affine mapping
    - Assumes the scene moves in its entirety – very rare
    - Can decompose scene into several major regions, each moving differently (region-based motion estimation)

- Determine global motion parameters for all pixels:
  - Direct estimation
  - Indirect estimation

- Exempt some pixels from the global motion estimation:
  - Iteratively determine the motion parameters and the set of pixels
  - Robust estimator
Direct Estimation

- Parameterize the DFD error in terms of the motion parameters, and estimate these parameters by minimizing the DFD error

\[ E_{DFD} = \sum_{n \in N} w_n |\psi_2(\mathbf{x}_n + \mathbf{d}(\mathbf{x}_n; \mathbf{a})) - \psi_1(\mathbf{x}_n)|^p \]

Weighting \( w_n \) coefficients depend on the importance of pixel \( \mathbf{x}_n \).

Ex: Affine motion:

\[
\begin{bmatrix}
  d_x(\mathbf{x}_n; \mathbf{a}) \\
  d_y(\mathbf{x}_n; \mathbf{a})
\end{bmatrix} = \begin{bmatrix}
  a_0 + a_1 x_n + a_2 y_n \\
  b_0 + b_1 x_n + b_2 y_n
\end{bmatrix}, \quad \mathbf{a} = [a_0, a_1, a_2, b_0, b_1, b_2]^T
\]

Exhaustive search or gradient descent method can be used to find \( \mathbf{a} \) that minimizes \( E_{DFD} \)
Indirect Estimation

- First find the dense motion field using pixel-based or block-based approach (e.g. EBMA)
- Then parameterize the resulting motion field using the motion model through least squares fitting

\[ E_{fit} = \sum w_n (d(x_n; a) - d_n)^2 \]

Affine motion:
\[ d(x_n; a) = [A_n]a, \]
\[ [A_n] = \begin{bmatrix} 1 & x_n & y_n & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_n & y_n \end{bmatrix} \]

\[ \frac{\partial E_{fit}}{\partial a} = \sum w_n [A_n]^T ([A_n]a - d_n) = 0 \]

\[ a = \left( \sum w_n [A_n]^T [A_n] \right)^{-1} \left( \sum w_n [A_n]^T d_n \right) \]

Weighting \( w_n \) coefficients depend on the accuracy of estimated motion at \( x_n \)
Robust Estimator

Concept: iteratively removing “outlier” pixels

1. Initialize the region to include all pixels in a frame
2. Apply the direct or indirect method over all pixels in the region
3. Evaluate errors (\(E_{DFD}\) or \(E_{fit}\)) at all pixels in the region
4. Eliminate “outlier” pixels with large errors
5. Repeat steps 2-4 for the remaining pixels in the region
Illustration of Robust Estimator

Fitting a line to the data points by using LMS and robust estimators
Region-Based Motion Estimation

• Assumption: the scene consists of multiple objects, with the region corresponding to each object (or sub-object) having a coherent motion
  • Physically more correct than block-based, mesh-based, global motion model

• Method:
  • Region First: Segment the frame into multiple regions based on texture/edges, then estimate motion in each region using the global motion estimation method
  • Motion First: Estimate a dense motion field, then segment the motion field so that motion in each region can be accurately modeled by a single set of parameters
  • Joint region-segmentation and motion estimation: iterate the two processes
Summary

• Fundamentals:
  • Optical flow equation
    • Derived from constant intensity and small motion assumption
    • Ambiguity in motion estimation
  • How to represent motion:
    • Pixel-based, block-based, region-based, global, etc.
  • Estimation criterion:
    • DFD (constant intensity)
    • OF (constant intensity + small motion)
  • Search method:
    • Exhaustive search, gradient-descent, multi-resolution
Summary

• Basic techniques
  • Pixel-based motion estimation
  • Block-based motion estimation
    • EBMA, integer-pel vs. half-pel accuracy, Fast algorithms

• More advanced techniques
  • Multiresolution approach
    • Avoid local minima, smooth motion field, reduced computation
  • Deformable block matching algorithm (DBMA)
    • To allow more complex motion within each block
  • Mesh-based motion estimation
    • To enforce continuity of motion across block boundaries
  • Global motion estimation
    • Good for estimating camera motion
  • Region-based motion estimation
    • More physically correct: allow different motion in each sub-object region