

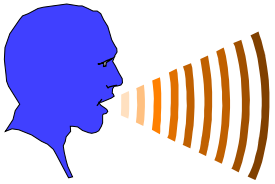



Image and Video Processing

Motion Estimation

Yao Wang
Tandon School of Engineering, New York University

Outline

- 2-D motion vs. optical flow
- Optical flow equation and ambiguity in motion estimation
- General methodologies in motion estimation
  - Motion representation
  - Motion estimation criterion
  - Optimization methods
- Horn and Schunck's Method
- Lucas-Kanade Flow Estimation Method
- Block Matching Algorithm
  - EBMA algorithm
  - Half-pel EBMA
  - Hierarchical EBMA (HBMA)
- Phase Correlation Method
- Deformable image registration

What Causes 2D Motion?

Static camera, moving scene

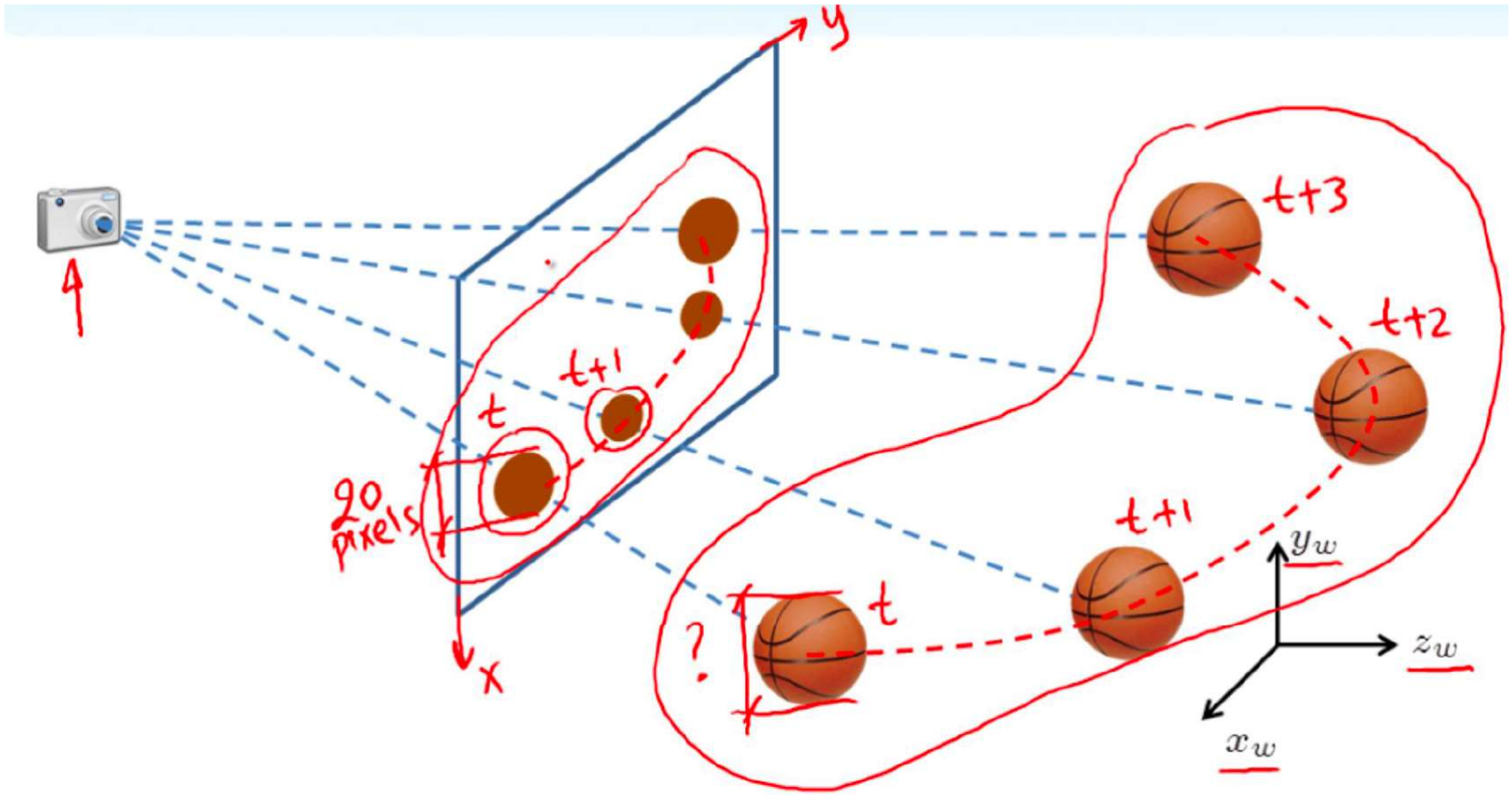
Static scene, moving camera

Moving scene, moving camera

Static camera, moving scene, moving light

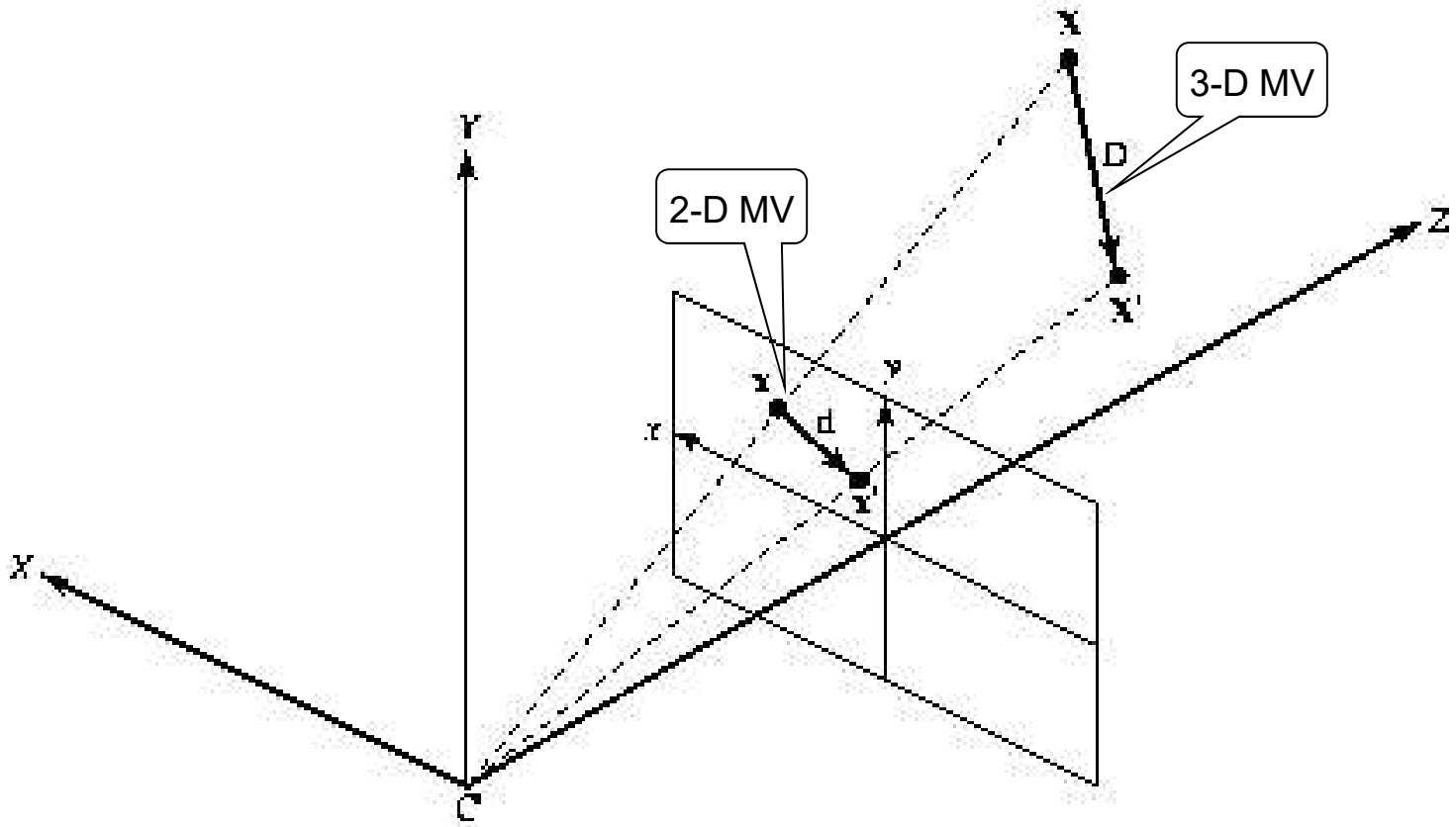
From http://courses.cs.washington.edu/courses/cse576/16sp/Slides/15_Flow.pdf

3D Motion to 2D Motion

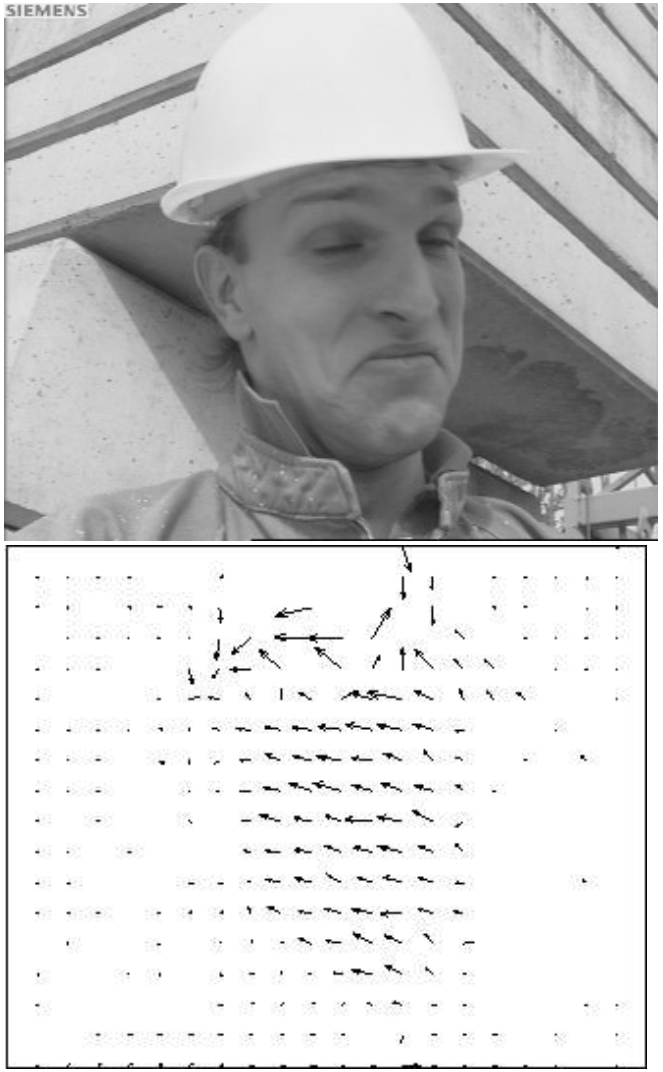


From Katsaggelos's Coursera Course on Image Processing, Lecture on motion estimation.

3-D Motion -> 2-D Motion

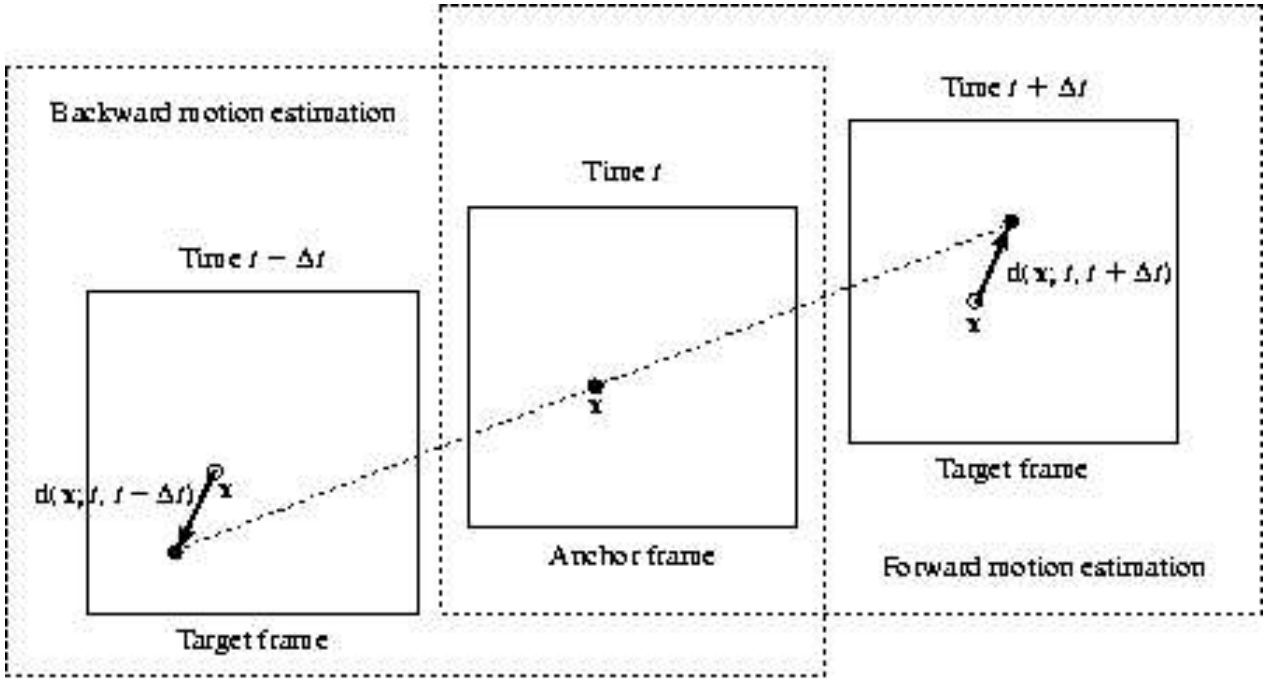


Sample 2D Motion Field



At each pixel (or center of a block) of the anchor image (right), the motion vector describes the 2D displacement between this pixel and its corresponding pixel in the other target image (left)

Motion Field Definition



Anchor frame: $\psi_1(\mathbf{x})$

Target frame: $\psi_2(\mathbf{x})$

Motion parameters: $\mathbf{a}$

Motion vector at a pixel in the anchor frame: $\mathbf{d}(\mathbf{x})$

Motion field: $\mathbf{d}(\mathbf{x};\mathbf{a}), \mathbf{x} \in \Lambda$

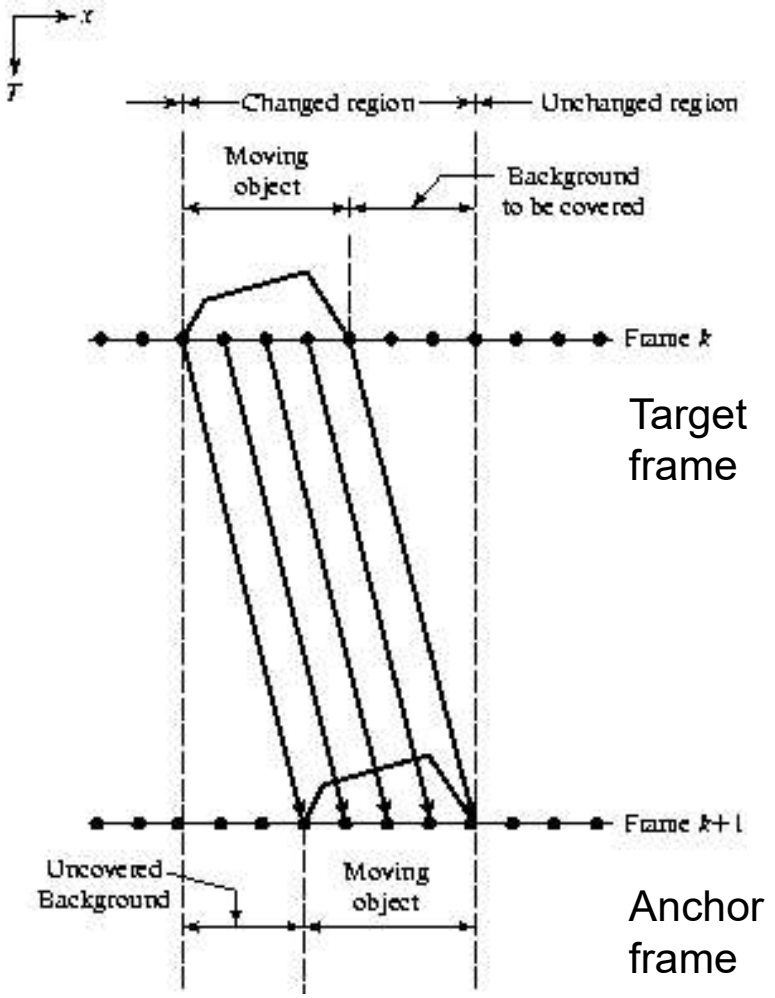
Mapping function:

$$\mathbf{w}(\mathbf{x};\mathbf{a}) = \mathbf{x} + \mathbf{d}(\mathbf{x};\mathbf{a}), \mathbf{x} \in \Lambda$$

$\Psi_1(x)= \Psi_2(w(x;a))=\Psi_2(x+d(x;a))$

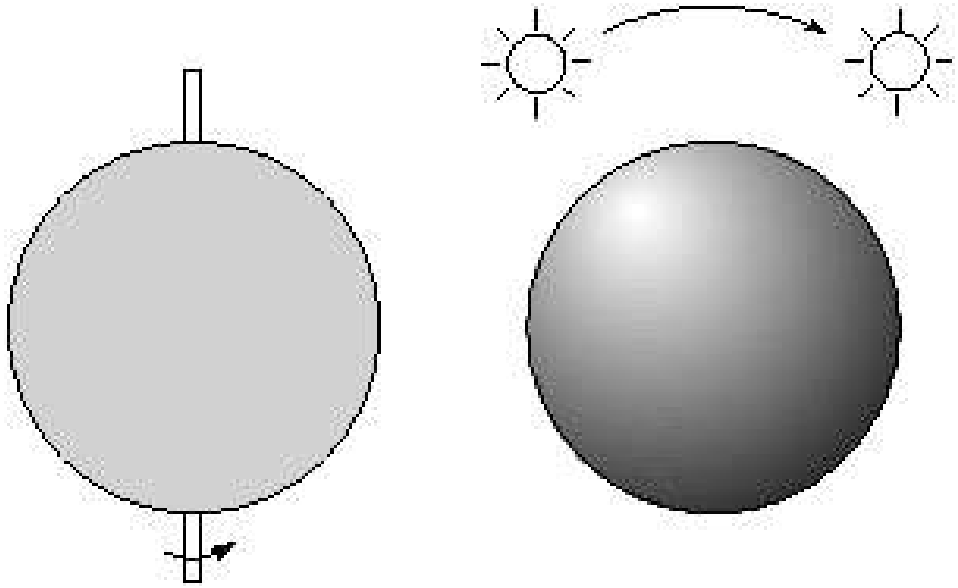
Occlusion Effect

- Motion is undefined in uncovered regions in the anchor frame
- Ideally a 2D motion field should indicate uncovered pixels as occluded instead of giving false MVs



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, depending on the actual 3D motion as well as illumination and object surface texture



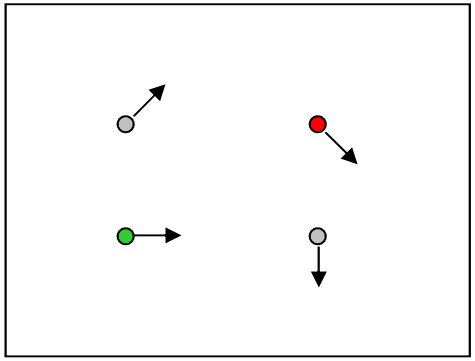
On the left, a sphere is rotating under a constant ambient illumination, but the observed image does not change.

On the right, a point light source is rotating around a stationary sphere, causing the highlight point on the sphere to rotate.

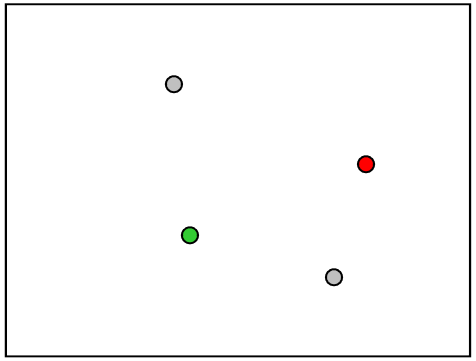
True Motion vs. Perceived Motion over Small Aperture

https://en.wikipedia.org/wiki/Barberpole_illusion

Problem definition: optical flow estimation



H(x,y) (anchor)



I(x,y) (target)

How to estimate pixel motion from image H to image I?

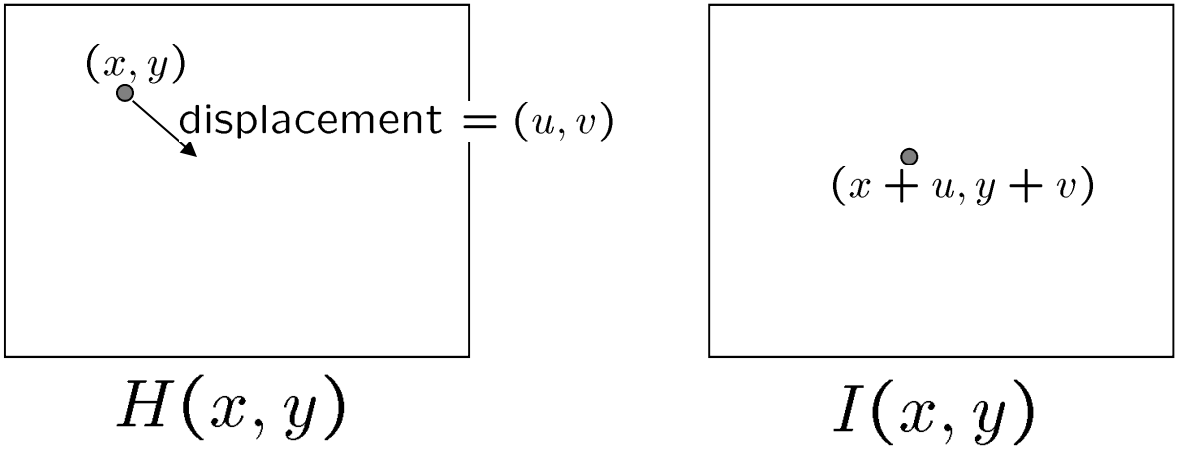
- Solve pixel correspondence problem
  - given a pixel in H look for nearby pixels of the same color in I

Key assumptions

- color constancy: a point in H looks the same in I
  - For grayscale images, this is brightness constancy
- small motion: points do not move very far

Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

Optical flow constraints
(Assuming grayscale images)



- brightness constancy: (x,y) in H is moved to (x+u, y+v) in I
  - H(x,y)= I(x+u, y+v), (u,v) is called flow vector or motion vector
- small motion: (u and v are less than 1 pixel)
  - Using Taylor series expansion of I:

$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

$$\approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

Optical flow equation

Combining these two equations

shorthand: $I_x = \frac{\partial I}{\partial x}$

$$
\begin{aligned}
0 &= I(x+u, y+v) - H(x,y) \\
&\approx I(x,y) + I_x u + I_y v - H(x,y) \\
&\approx (I(x,y) - H(x,y)) + I_x u + I_y v \\
&\approx I_t + I_x u + I_y v \\
&\approx I_t + \nabla I \cdot [u \;\; v]
\end{aligned}
$$

$I_t(x,y) = I(x,y) - H(x,y)$

Optical flow equation:

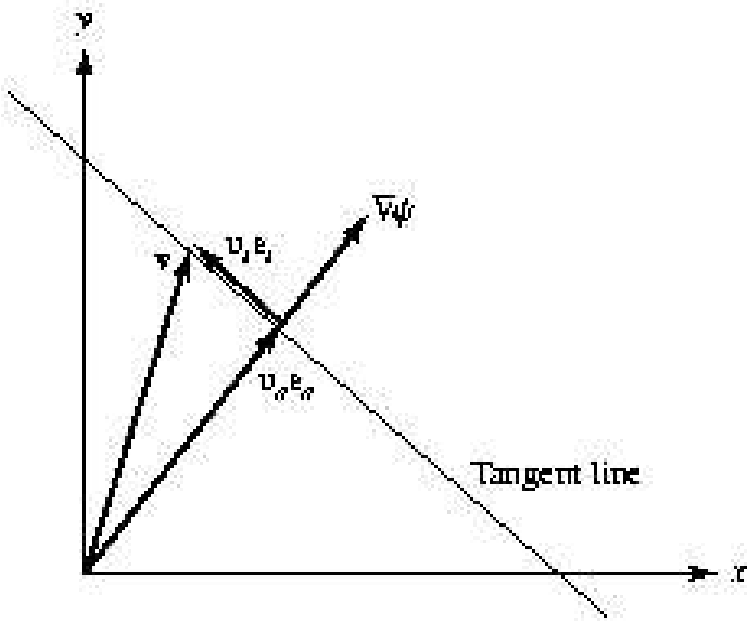
$$I_x(x,y)u(x,y) + I_y(x,y)v(x,y) = H(x,y) - I(x,y)$$

or $\nabla I^T \boldsymbol{u} = -I_t$

Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

Ambiguities in Motion Estimation (Aperture Problem)

- Optical flow equation only constrains the flow magnitude in the gradient direction (v_n)
- The flow vector in the tangent direction (v_t) is under-determined
- In regions with constant brightness ($\nabla \psi = 0$), the flow is also indeterminate
- --> Motion estimation is unreliable in regions with flat texture, more reliable near edges



Optical flow eq: $\nabla\psi^T\, v = -\psi_t$

Representing v as $v = v_n e_n + v_t e_t$

Then: $\nabla\psi^T\, v = v_n \|\nabla\psi\|$

Therefore: $v_n = -I_t/\|\nabla\psi\|$,

v_t is indeterminate

Pop Quiz

- Suppose the image is a vertical bar moving vertically over a flat background
  - Can you observe any change or detect the motion?
  - What if the bar moves horizontally?

  - In this case, the gradient direction is horizontal, the tangent direction is vertical
  - Motion is indeterminate in the tangent direction

Pop Quiz

- Suppose the imaged scene is part of the side of a big box with a constant green color, and the box is moving in the direction parallel to the side
  - Can you observe any change or detect the motion?
  - What if the box moves towards the camera?
  - In this case, the gradient magnitude is zero everywhere
  - Motion is indeterminate

Pop Quiz

- What is difference between true 2D motion and optical flow?
- Can we differentiate the optical flow due to camera motion, lighting change, object motion?
- Which part of the flow vector cannot be determined from images?

Pop Quiz (w/answers)

- What is difference between true 2D motion and optical flow?
  - Optical flow is the “perceived 2D motion”, which may not be the same as the “true 2D motion”
- Can we differentiate the optical flow due to camera motion, lighting change, object motion?
  - When we just look at a small area of the image (aperture), we cannot tell
  - When we look at the whole picture, and using our accumulated knowledge about what we typically see (prior knowledge), we may be able to
- Which part of the flow vector cannot be determined from images?
  - Along the edges
  - In flat regions

General Considerations for Motion Estimation

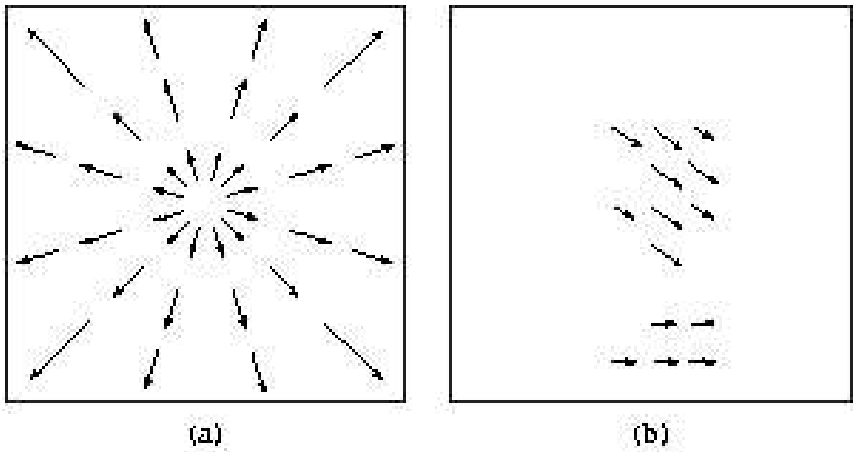
- Two categories of approaches:
  - Feature based: finding corresponding features in two different images and then derive the entire motion field based on the motion vectors at corresponding features.
    - Last lecture!
    - More often used for estimating global motion between the two images due to camera motion (or view angle difference)
  - Intensity based: directly finding MV at every pixel or parameters (**a**) that characterize the motion field based on constant intensity assumption (optical flow equation)
    - More often used for motion compensated prediction and filtering, required in video coding, frame interpolation -> focus of this lecture

Three Problems in Motion Estimation

- How to represent the motion field?
- What criteria to use to estimate motion parameters?
- How to search motion parameters?

Motion Representation

Global:
Entire motion field is represented by a few global parameters (affine, homography)

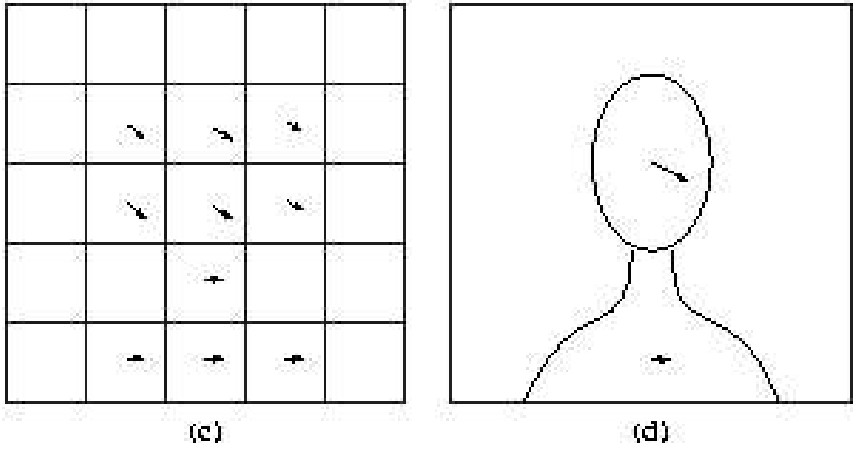


Pixel-based:
One MV at each pixel, with some smoothness constraint between adjacent MVs.

a= MV for each pixel

Block-based:
Entire frame is divided into blocks, and motion in each block is characterized by a few parameters (e.g. a constant MV)

a= MV for each block



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.

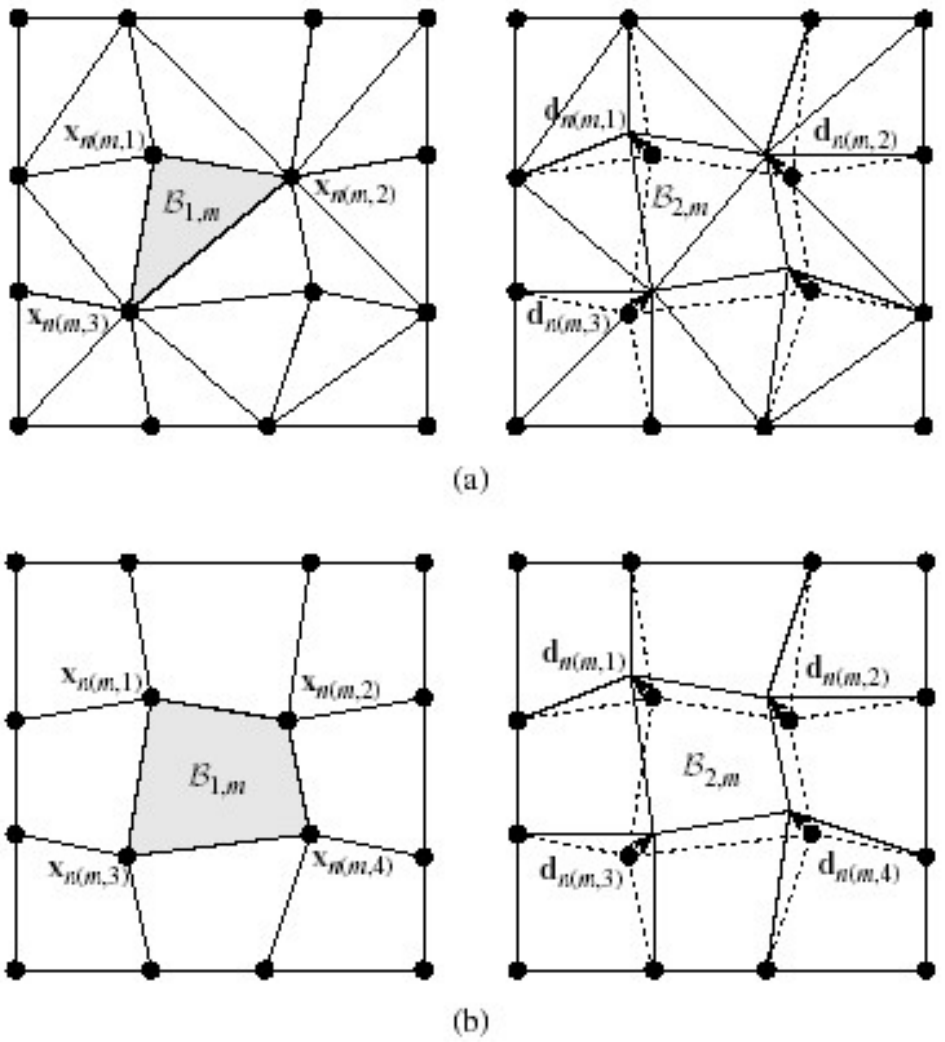
a= Motion parameters for each region

Motion Representation: Mesh-Based

Cover an image with a mesh (triangular or rectangular or irregular), and define MVs at mesh nodes, interpolate MVs at other pixels from nodal MVs.

Mesh is also known as a control grid. Interpolation is done using spline interpolation. This method is also known as spline-based method.

Mostly used for deformable registration in medical images



Motion Estimation Criterion

- To minimize the displaced frame difference (DFD) (based on constant intensity assumption)

$$E_{\text{DFD}}(\mathbf{a}) = \sum_{x\in\Lambda} |\psi_2(\mathbf{x}+\mathbf{d}(\mathbf{x};\mathbf{a})) - \psi_1(\mathbf{x})|^p \rightarrow \min$$

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

a is motion parameter vector that defines the entire motion field

- To satisfy the optical flow equation

$$E_{\text{OF}}(\mathbf{a}) = \sum_{x\in\Lambda} \left| \left(\nabla \psi_2(\mathbf{x})\right)^T \mathbf{d}(\mathbf{x};\mathbf{a}) + \psi_2(\mathbf{x}) - \psi_1(\mathbf{x}) \right|^p \rightarrow \min$$

- To impose additional smoothness constraint using regularization technique (Important in pixel- and block-based representation)

$$E_s(\mathbf{a}) = \sum_{\mathbf{x}\in\Lambda} \sum_{\mathbf{y}\in N_x} \left\| \mathbf{d}(\mathbf{x};\mathbf{a}) - \mathbf{d}(\mathbf{y};\mathbf{a}) \right\|^2$$

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

- Bayesian (MAP) criterion: to maximize the a posteriori probability

$$P(D=\mathbf{d}|\psi_2,\psi_1) \rightarrow \max$$

Relation with previous notation: ψ_2=I, ψ_1=H

Relation Among Different Criteria

- OF criterion is good only if motion is small.
- OF criterion can yield closed-form solution when the objective function is quadratic in MVs.
- When the motion is not small, can use coarse exhaustive search to find a good initial solution, and use this solution to deform target frame, and then apply OF criterion between original anchor frame and the deformed target frame.
- Bayesian criterion can be reduced to the DFD criterion plus motion smoothness constraint

Optimization Methods

- Exhaustive search
  - Typically used for the DFD criterion with p=1 (MAD)
  - Guarantees reaching the global optimal
  - Computation required may be unacceptable when number of parameters to search simultaneously is large!
  - 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
  - Avoid being trapped into a local minimum

Regularization for Dense Motion Estimation

- How do we estimate the motion at each pixel?
- Will yield chaotic motion field if we directly solve the optical flow equation at that pixel (e.g. applying on 3 color channels to get 3 equations)
- General idea:
  - Assuming motion field satisfies some properties (e.g., adjacent pixels have similar MVs)
  - Set up an optimization objective based on the DFD/OF as well as the prior knowledge over the neighborhood
  - Need to find MVs at all pixels $\boldsymbol{u}(\boldsymbol{x})$ simultaneously or iteratively
  - Adding such prior knowledge is known as Regularization

Pixel-Based Motion Estimation

- Horn-Schunck method
  - DFD + motion smoothness criterion
- Multipoint neighborhood method
  - Assuming every pixel in a small block surrounding a pixel has the same MV (Lucas-Kanade method)
- Pel-recurrsive 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
- Recommended reading for recent advances:
  - Sun, Deqing, Stefan Roth, and Michael J. Black. "Secrets of optical flow estimation and their principles." In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2432-2439. IEEE, 2010.

Horn and Schunck's Method (Optional)

$$E_{DFD} = \sum_{\boldsymbol{x}\in\Omega} (I(\boldsymbol{x}+\boldsymbol{u}(\boldsymbol{x})) - H(\boldsymbol{x}))^2, \qquad E_{OF} = \sum\nolimits_{\boldsymbol{x}\in\Omega} (I(\boldsymbol{x}) - H(\boldsymbol{x}) - \nabla I(\boldsymbol{x})^T \boldsymbol{u}(\boldsymbol{x}))^2$$

$$E_{smooth} = \sum\nolimits_{\boldsymbol{x}\in\Omega} \sum\nolimits_{\boldsymbol{y}\in N(\boldsymbol{x})} ||\boldsymbol{u}(\boldsymbol{x}) - \boldsymbol{u}(\boldsymbol{y})||^2$$

$$E = w_{DFD/OF} E_{DFD/OF} + w_{smooth} E_{smooth}$$

Special case:

$$E_{smooth} = \sum\nolimits_{\boldsymbol{x}\in\Omega} ||\boldsymbol{u}(x,y) - \boldsymbol{u}(x,y-1)||^2 + ||\boldsymbol{u}(x,y) - \boldsymbol{u}(x,y+1)||^2 + ||\boldsymbol{u}(x,y) - \boldsymbol{u}(x-1,y)||^2 + ||\boldsymbol{u}(x,y) - \boldsymbol{u}(x+1,y)||^2$$

- Solve minimization problem using gradient descent
  - Update **u**(**x**)={u(x,y), v(x,y)} at each pixel, sequentially and repeatedly
- Minimizing MV difference square between adjacent pixels can lead to overly smooth motion field.
- Better to minimize the absolute difference (=Total variation), which allow motion discontinuity at object boundaries

Lucas-Kanade Method

- Optical flow equation give 1 equation to 2 unknowns at each pixel

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u \ v]$$

- Assume pixels in a small neighborhood around the center pixel have the same motion (u,v)
- If we use a 5x5 window, that gives us 25 equations per pixel!

$$\underset{25\times2}{\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix}}_{A} \underset{2\times1}{\begin{bmatrix} u \\ v \end{bmatrix}}_{d} = - \underset{25\times1}{\begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}}_{b}$$

Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

Lukas-Kanade Method

Prob: we have more equations than unknowns

$$\underset{25\text{x}2}{A}\ \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b} \longrightarrow \text{minimize } \|Ad - b\|^2$$

Solution: solve least squares problem

- minimum least squares solution given by solution (in d) of:

$$\underset{2\text{x}2}{(A^T A)}\ \underset{2\text{x}1}{d} = \underset{2\text{x}1}{A^T b}$$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$A^T A$ $\qquad$ $A^T b$

- The summations are over all pixels in the K x K window, but only use the solution to determine the MV at the center.
- This technique was proposed by Lukas & Kanade (1981)

Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

Conditions for solvability

- Optimal (u, v) satisfies Lucas-Kanade equation

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$A^T A$ $\qquad$ $A^T b$

When is this Solvable?

- A^TA should be invertible
- A^TA should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of A^TA should not be too small
- A^TA should be well-conditioned
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

Eigenvectors of A^TA

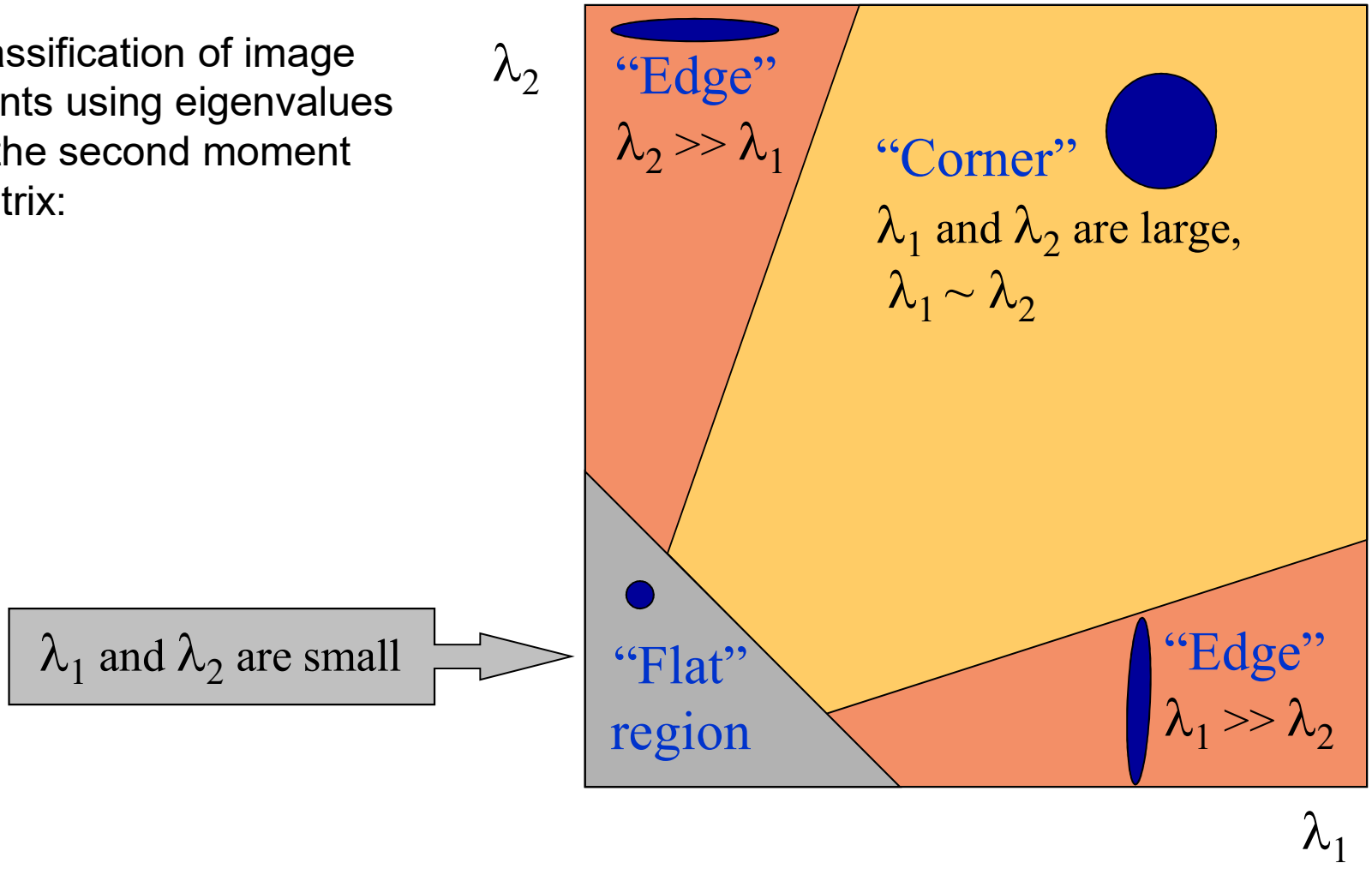
$$A^TA = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y] = \sum \nabla I (\nabla I)^T$$

- Recall the Harris corner detector: $M = A^TA$ is the *second moment matrix*
- The eigenvectors and eigenvalues of M relate to edge direction and magnitude
  - The eigenvector associated with the larger eigenvalue points in the direction of fastest intensity change
  - The other eigenvector is orthogonal to it
  - Motion estimation is reliable only at locations with corners!

Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

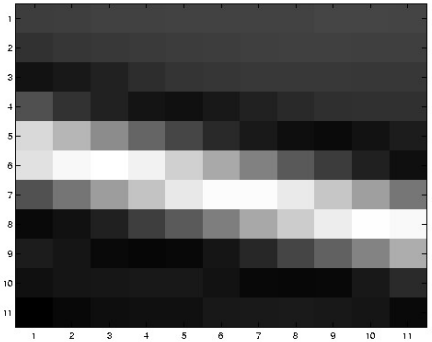
Interpreting the eigenvalues

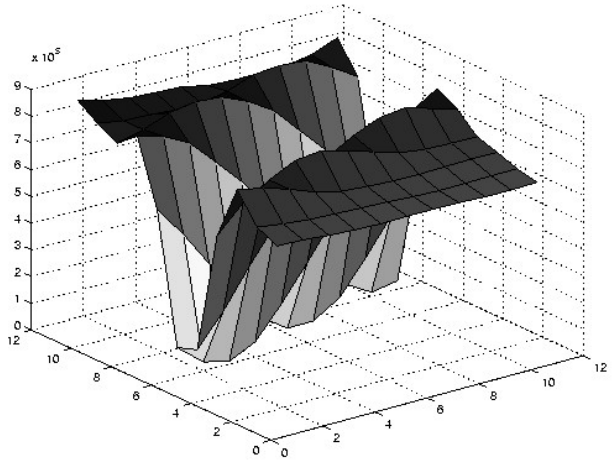
Classification of image points using eigenvalues of the second moment matrix:



Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

Edge





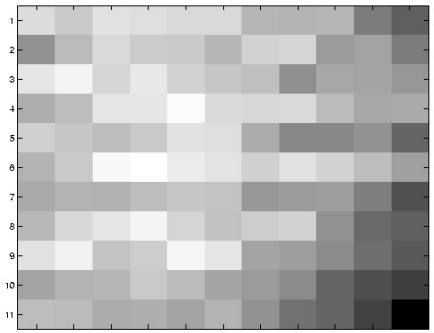
$$\sum \nabla I (\nabla I)^T$$

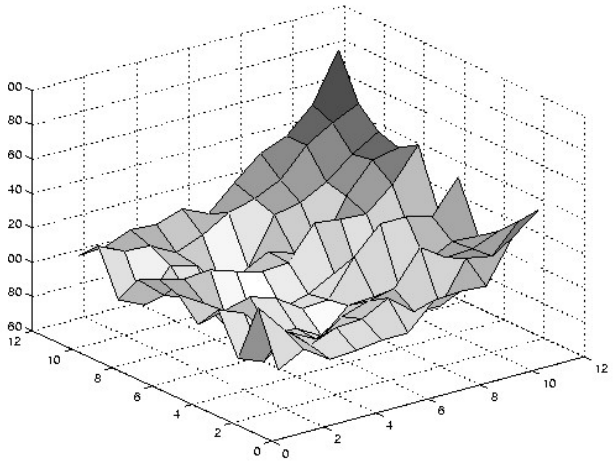
– large gradients orthogonal to edge
– large λ_1, small λ_2
– Motion along edge cannot be determined

Error surface has minimal along the edge

Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

Low texture region





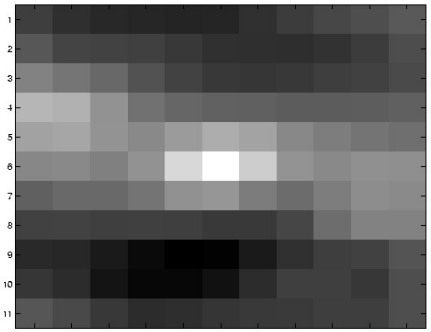
$$\sum \nabla I (\nabla I)^T$$

- gradients have small magnitude
- small λ_1, small λ_2
- Motion around flat region cannot be determined

Error surface fairly flat. Local minima unreliable

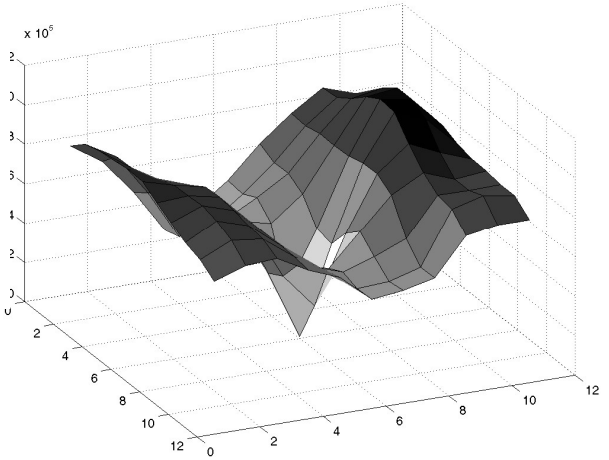
Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

High textured region



$$\sum \nabla I (\nabla I)^T$$

- gradients are different, large magnitudes
- large λ_1, large λ_2
- Motion can be determined well



Error surface has unique minimum

Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

Errors in Lukas-Kanade

What are the potential causes of errors in this procedure?

- Suppose A^TA is easily invertible
- Suppose there is not much noise in the image

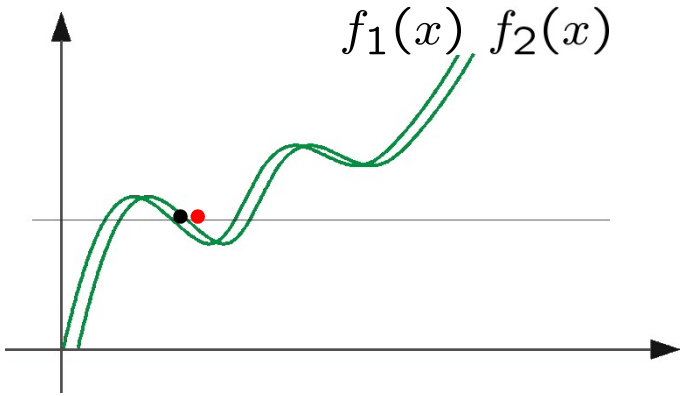
When our assumptions are violated

- Brightness constancy is not satisfied
- The motion is not small
- A point does not move like its neighbors
  - window size is too large
  - what is the ideal window size?

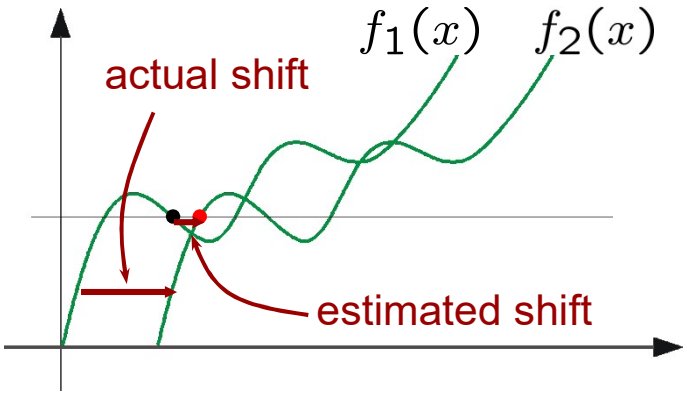
Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

Problem when motion is large

- Optical flow equation is derived from Taylor extension
  - Only correct when the motion is small
- When motion is large, there are ambiguity in estimating the motion by trying to match over a small neighborhood



nearest match is correct (no aliasing)



nearest match is incorrect (aliasing)

Which correspondence is correct?
How to solve this ambiguity?

Figure from Fergus

Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

Iterative Refinement

- Optical flow equation is correct only if true motion is small
- What if a point undergoes large motion (>1pixel)

- Iterative Lukas-Kanade Algorithm
  1. Estimate velocity at each pixel by solving Lucas-Kanade equations
  2. Warp H towards I using the estimated flow field
     - *H'(x,y)=H(x-u,y-v)*
     - *use image warping techniques*
  3. Repeat on I and H' until convergence
  4. Final motion is the sum of motion determined at each step.

Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

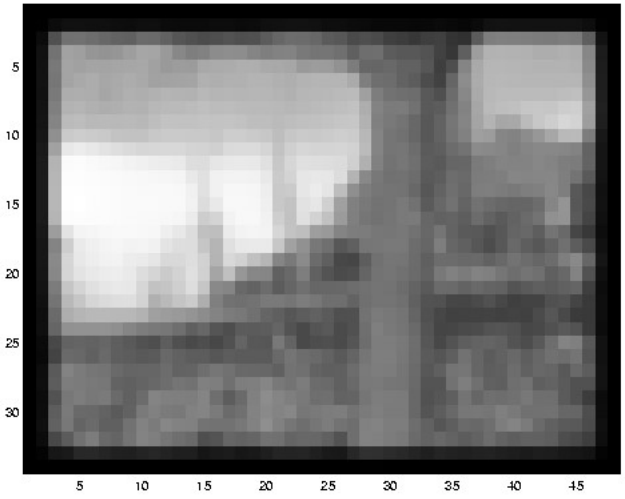
Optical Flow: Iterative Estimation

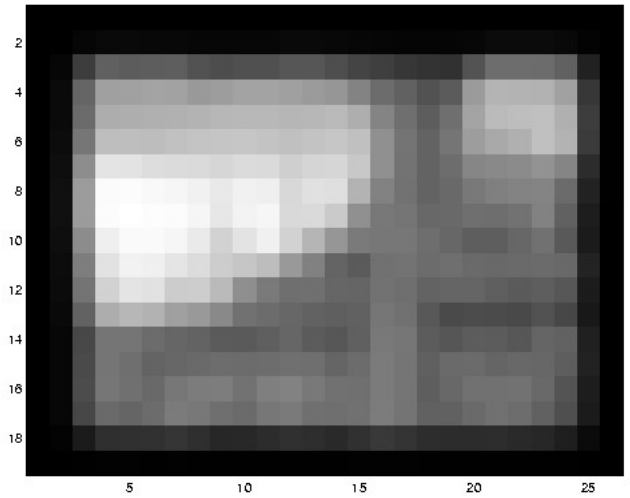
Some Implementation Issues:

- Warping is not easy (ensure that errors in warping are smaller than the estimate refinement)
- Warp one image (H), take derivatives of the other (I) so you don't need to re-compute the gradient after each iteration.
- Often useful to low-pass filter the images before motion estimation (for better derivative estimation, and linear approximations to image intensity)

Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

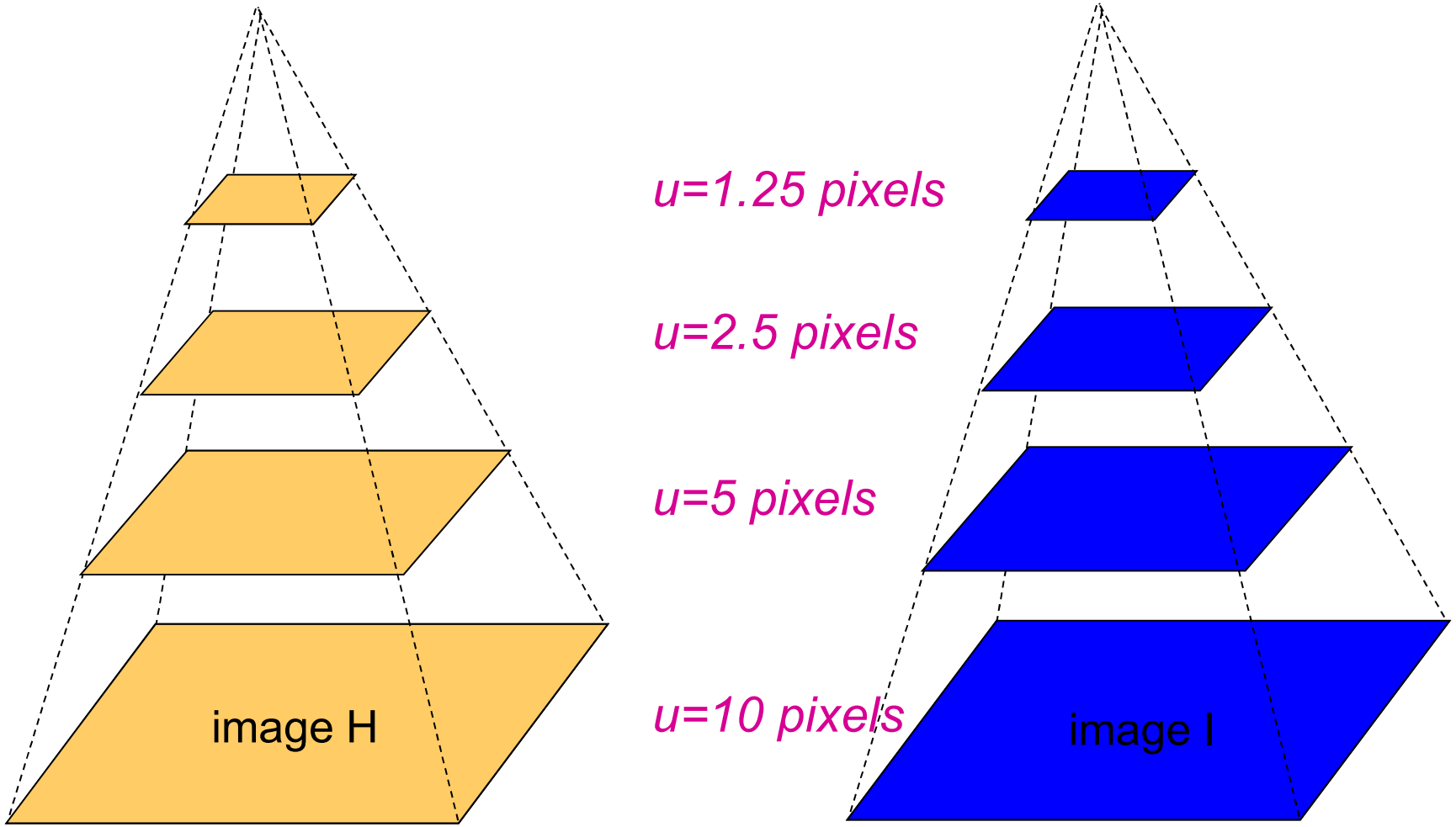
Another Way to Deal With Large Movement
Reduce the resolution!





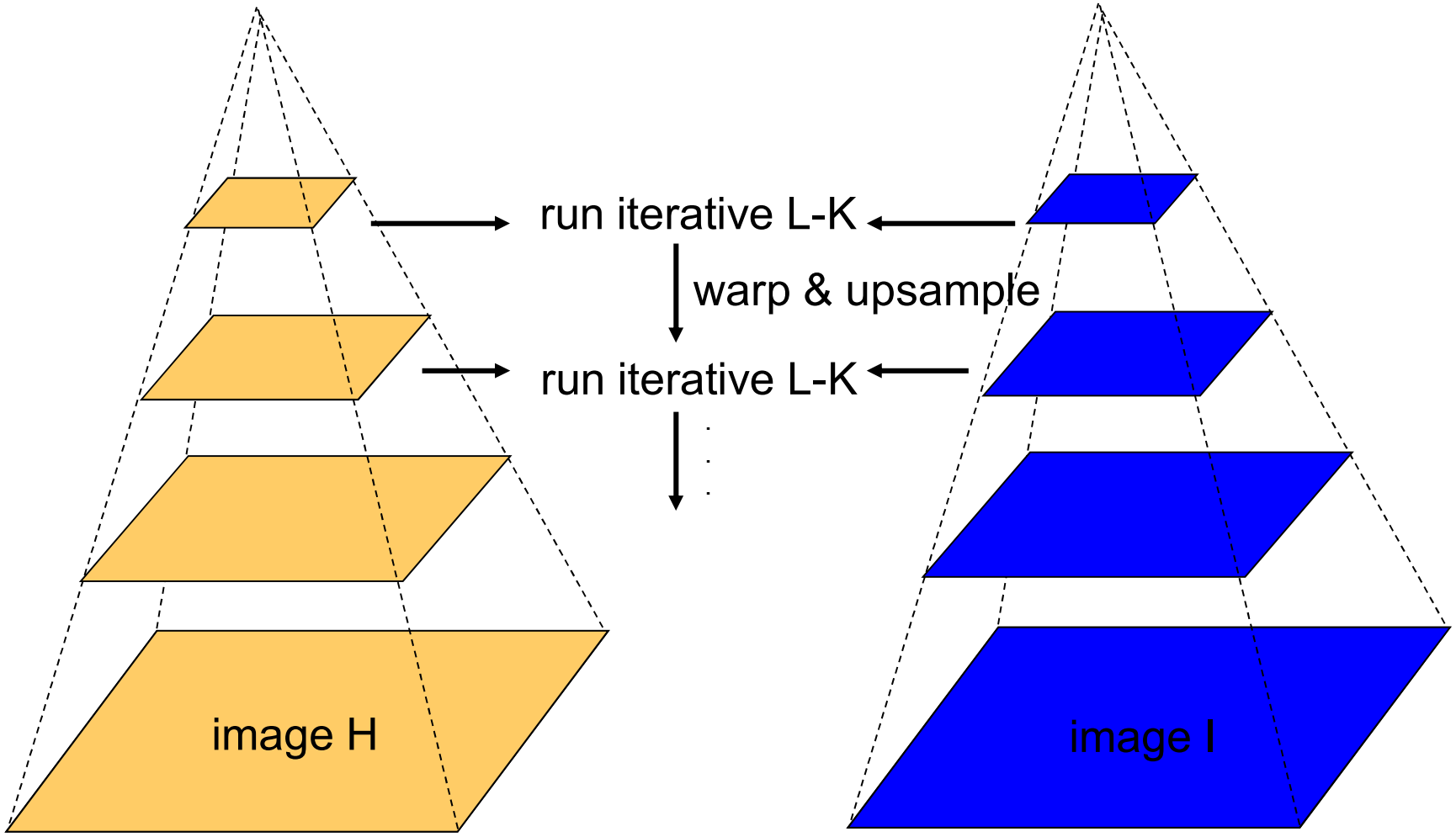
Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

Coarse-to-fine optical flow estimation

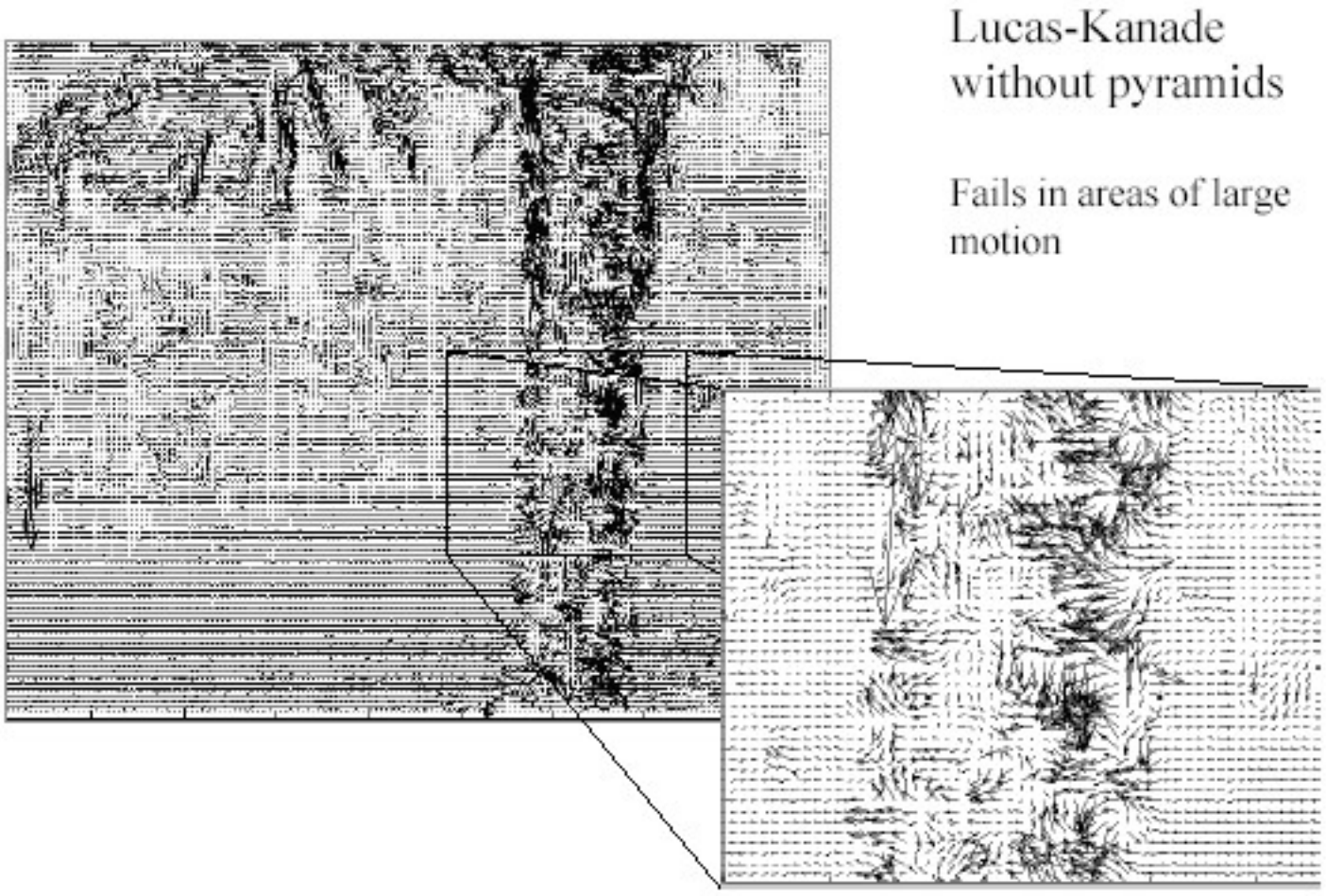


Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf

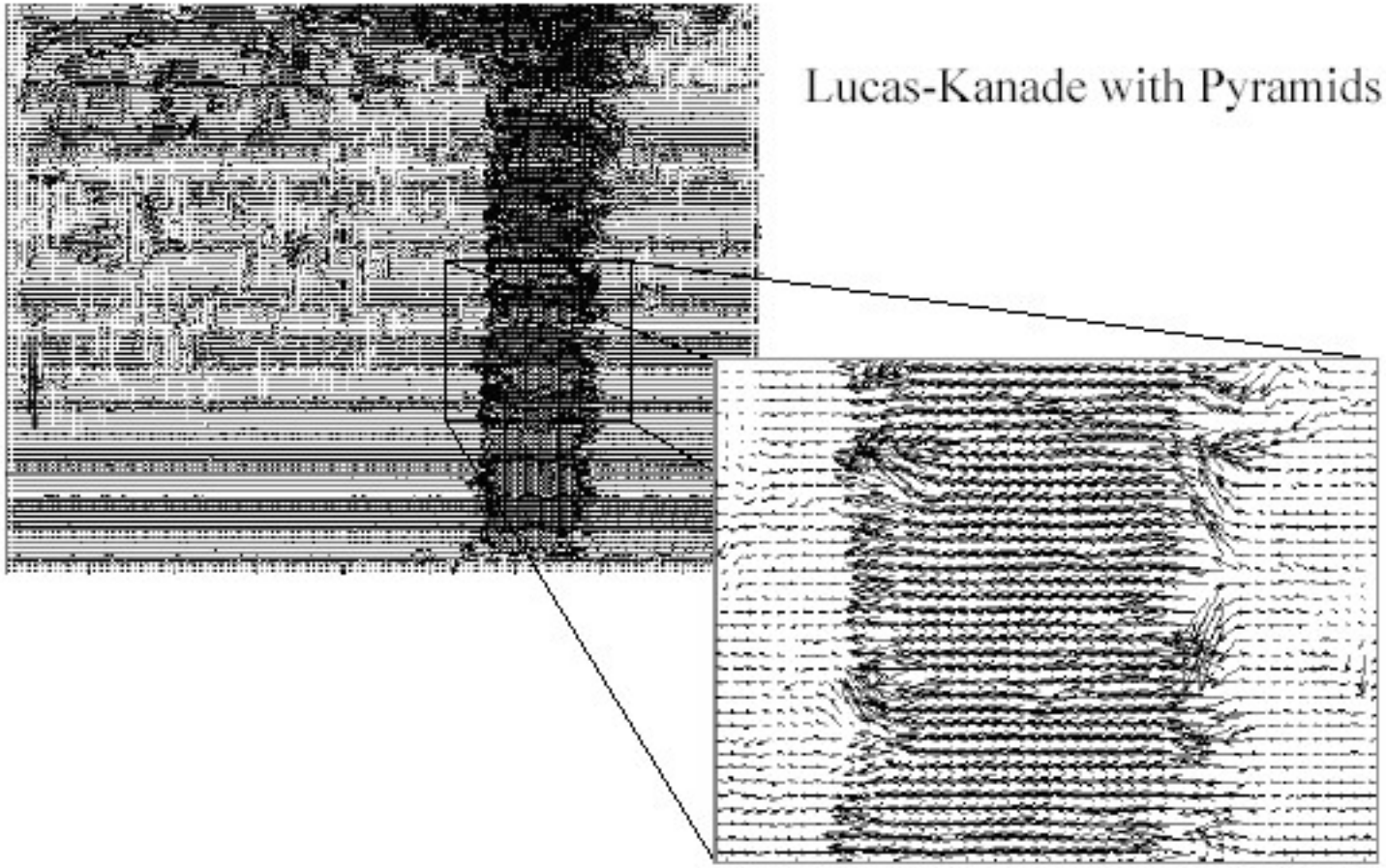
Coarse-to-fine optical flow estimation



Courtesy of Rob Fergus, http://cs.nyu.edu/~fergus/teaching/vision/5_6_Fitting_Matching_Opticalflow.pdf



From Khurram Hassan-Shafique CAP5415 Computer Vision 2003



From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

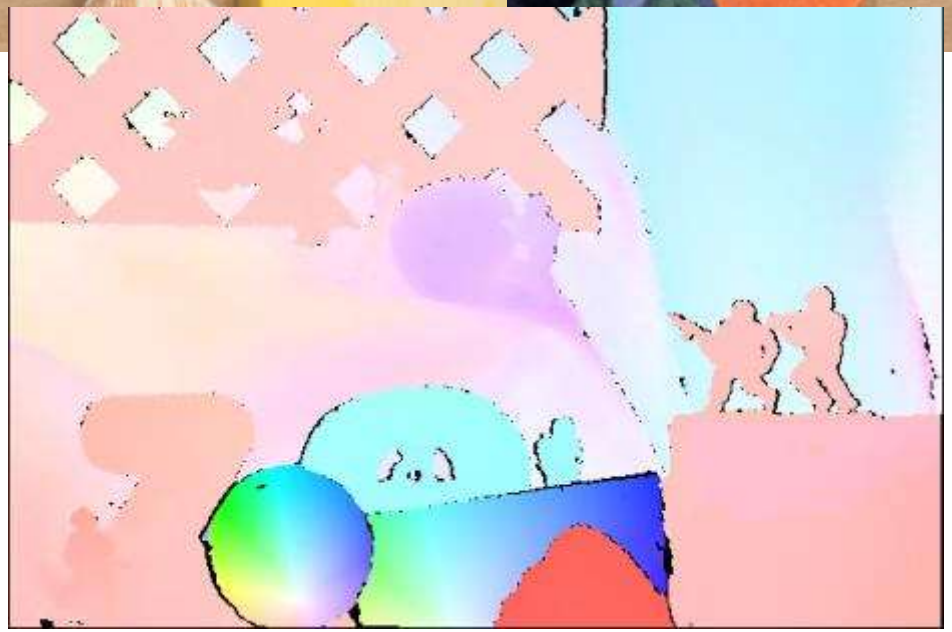
Open Competition for Optical Flow Estimation

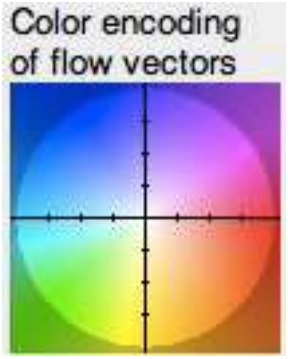
http://vision.middlebury.edu/flow/

- A public domain database containing sample pairs of images with ground truth flow
  - Designed for evaluate motion estimation algorithms between two frames
- Open to submission of algorithms and estimation results
- The site evaluates the accuracy against ground truth
- Advance in many areas in computer vision has been greatly facilitated by such challenges
  - Image classification using deep learning
  - Face detection, recognition
  - Object detection and tracking

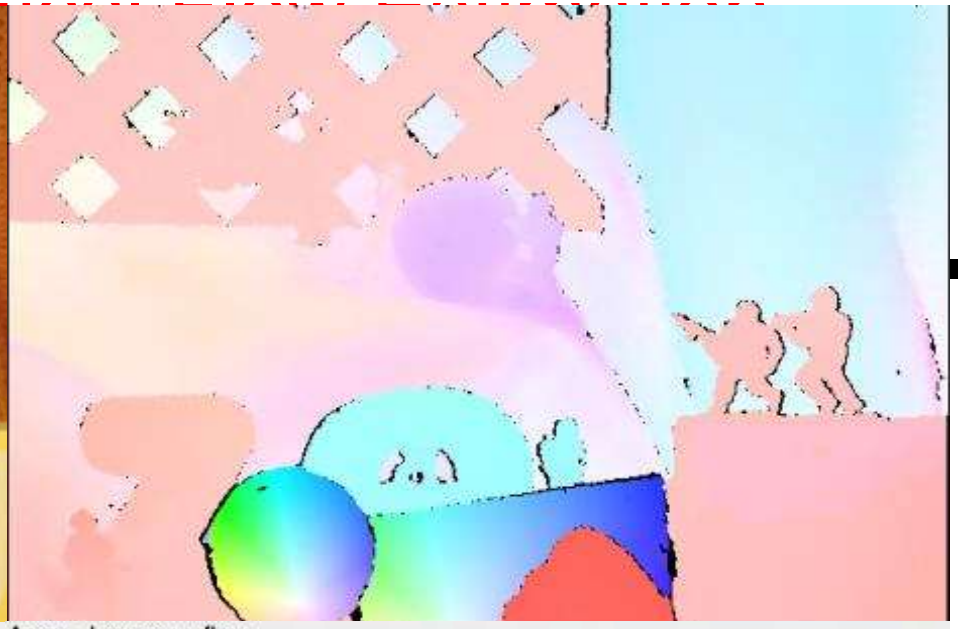
Challenge for Optical Flow Estimation

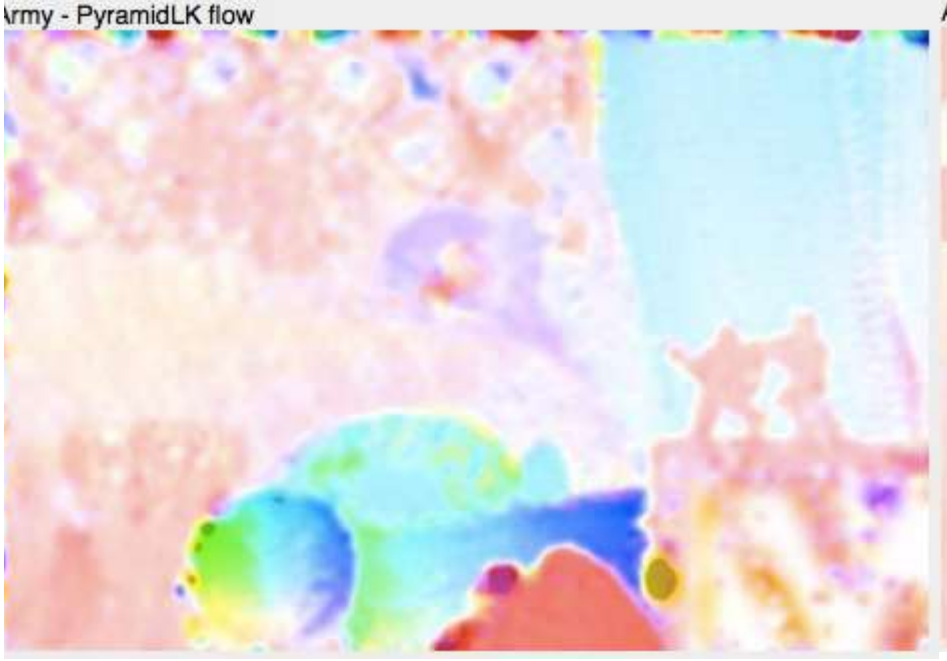
http://vision.middlebury.edu/flow/





Courtesy of Ali Farhadi. From http://courses.cs.washington.edu/courses/cse576/16sp/Slides/15_Flow.pdf





Lucas-Kanade flow with pyramid

Layer++

Courtesy of Ali Farhadi. From http://courses.cs.washington.edu/courses/cse576/16sp/Slides/15_Flow.pdf

Kanade-Lucas-Tomasi (KLT) Tracker

- Detect feature points in the first image where both eigenvalues of the moment matrix are large (similar to Harris corner detector)
- For each feature point in the first image, estimate its motion between first and second frames using a small window surrounding it using the Lucas-Kanade flow estimation method
- Repeat between frame 2 and frame 3, using tracked feature points in Frame 2.
- Verify that the corresponding features between non-adjacent frames satisfy a global affine mapping. Features that are outliers are dropped.
- References:
  - Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.
  - Carlo Tomasi and Takeo Kanade. Detection and Tracking of Point Features. *Carnegie Mellon University Technical Report CMU-CS-91-132*, April 1991.
  - Jianbo Shi and Carlo Tomasi. Good Features to Track. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

Pop Quiz (Homework)

- What is the assumption of the Lucas-Kanade method?
- In what regions we cannot estimate the motion reliably?
- How to handle large motion?
- How does KLT tracker work?

Block-Based Motion Estimation

- Assume all pixels in a block undergo a coherent motion, and search for the motion parameters for each block independently
- Block matching algorithm (BMA): assume translational motion, 1 MV per block (2 parameter)
  - Exhaustive BMA (EBMA)
  - Fast algorithms
- Deformable block matching algorithm (DBMA): allow more complex motion (affine, bilinear)

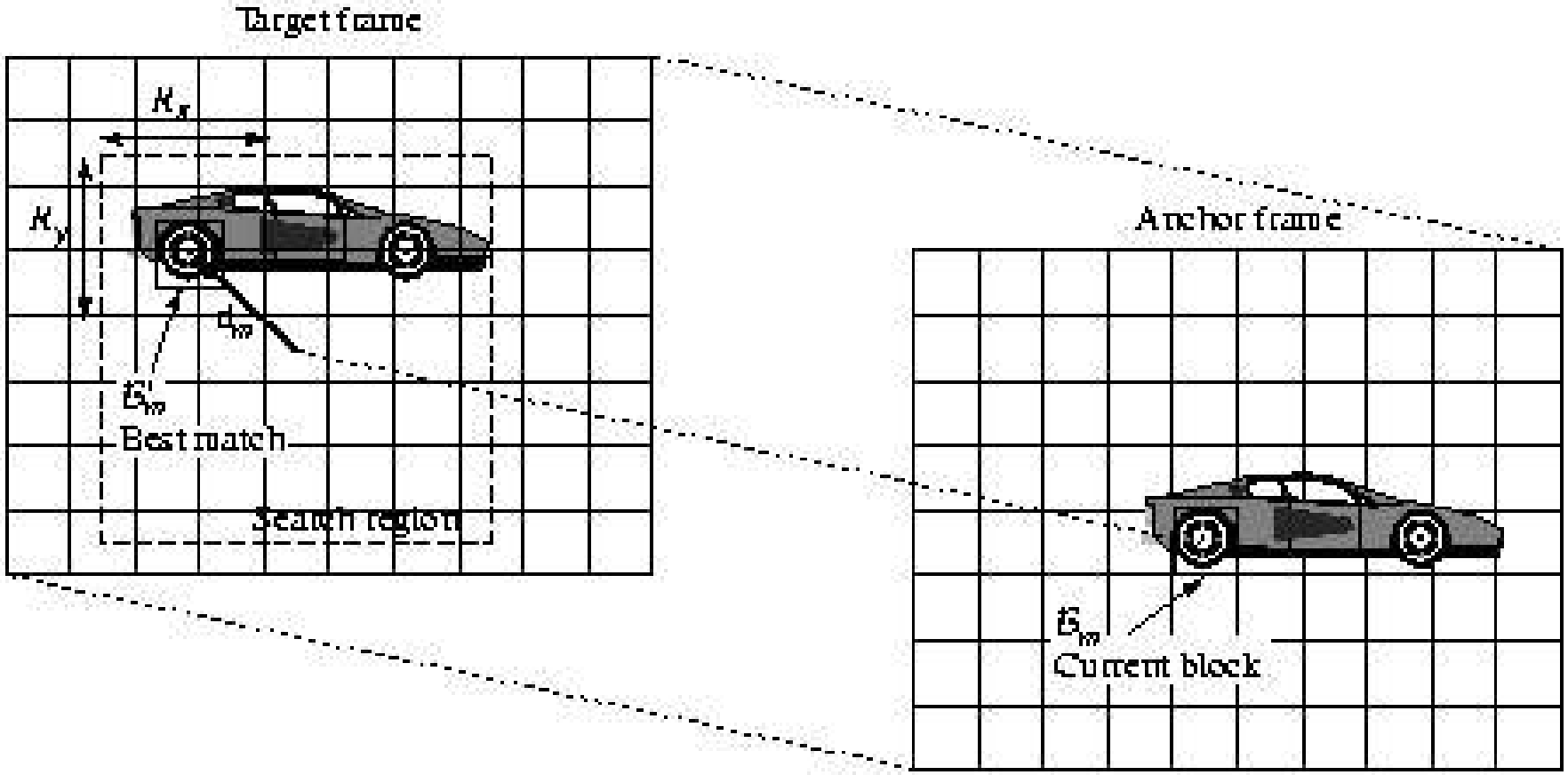
Block Matching Algorithm

- 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
- Minimizing function:

$$E_{\mathrm{DFD}}(\mathbf{d}_{\mathrm{m}}) = \sum_{\mathbf{x}\in B_m} \left|\psi_2(\mathbf{x}+\mathbf{d}_m) - \psi_1(\mathbf{x})\right|^p \rightarrow \min$$

- Optimization method:
  - Exhaustive search (feasible as one only needs to search one MV at a time), using MAD criterion (p=1)
  - Fast search algorithms
  - Integer vs. fractional pel accuracy search

Exhaustive Block Matching Algorithm (EBMA)



Sample 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 i=1:N:height-N,
  for j=1:N:width-N  %for every block in the anchor frame
     MAD_min=256*N*N;mvx=0;mvy=0;
     for k=-R:1:R,
       for l=-R:1:R  %for every search candidate (needs to be modified so that i+k etc are
within the image domain!)
         MAD=sum(sum(abs(f1(i:i+N-1,j:j+N-1)-f2(i+k:i+k+N-1,j+l:j+l+N-1))));
              % calculate MAD for this candidate
         if MAD<MAX_min
              MAD_min=MAD,dy=k,dx=l;
         end;
     end;end;
     fp(i:i+N-1,j:j+N-1)= f2(i+dy:i+dy+N-1,j+dx:j+dx+N-1);
              %put the best matching block in the predicted image
     iblk=(floor)(i-1)/N+1; jblk=(floor)(j-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.

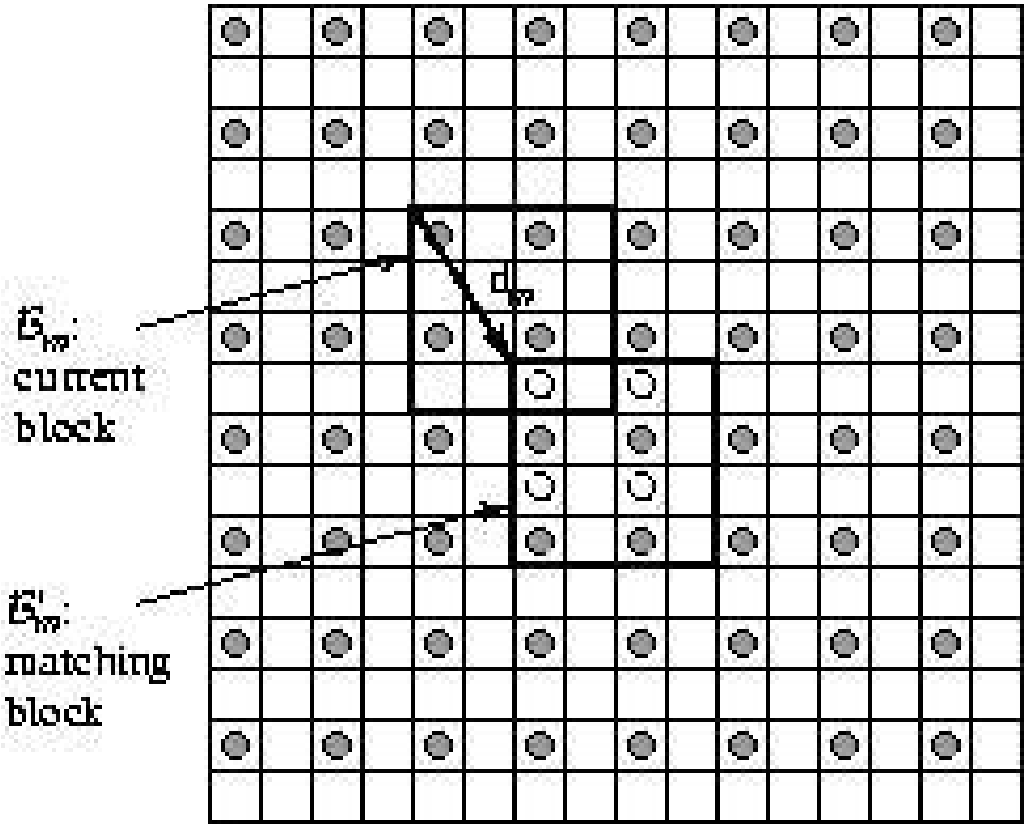
Complexity of Integer-Pel EBMA

- 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 = 2.85x10^8/frame =8.55x10^9/second
- Regular structure suitable for VLSI implementation
- Challenging for software-only implementation

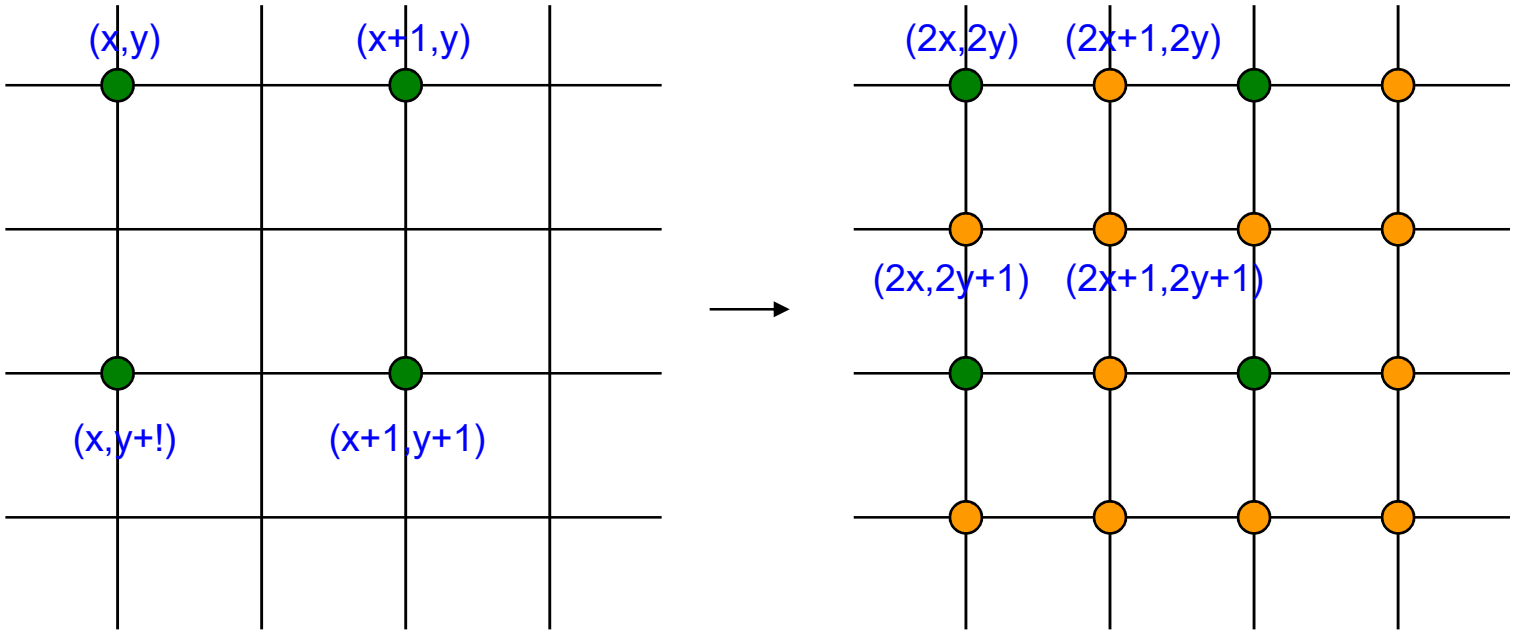
Fractional Accuracy EBMA

- 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



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+1,y])/2
O[2x+1,2y+1]=(I[x,y]+I[x+1,y]+I[x,y+1]+I[x+1,y+1])/4

Implementation for Half-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
%first upsample f2 by a factor of 2 in each direction
f3=imresize(f2, 2,'bilinear') (or use you own implementation!)
for i=1:N:height-N, for j=1:N:width-N  %for every block in the anchor frame
    MAD_min=256*N*N;mvx=0;mvy=0;
    for k=-R:0.5:R, for l=-R:0.5:R  %for every search candidate (needs to be modified!)
        %MAD=sum(sum(abs(f1(i:i+N-1,j:j+N-1)-f2(i+k:i+k+N-1,j+l:j+l+N-1))));
         MAD=sum(sum(abs(f1(i:i+N-1,j:j+N-1)-f3(2*(i+k):2:2*(i+k+N-1),2*(j+l):2:2*(j+l+N-1)))));
            % calculate MAD for this candidate
        if MAD<MAX_min
            MAD_min=MAD,dy=k,dx=l;
        end;
    end;end;
    fp(i:i+N-1,j:j+N-1)= f3(2*(i+dy):2:2*(i+dy+N-1),2*(j+dx):2:2*(j+dx+N-1));
            %put the best matching block in the predicted image
    iblk=(floor)(i-1)/N+1; jblk=(floor)(j-1)/N+1; %block index
    mvx(iblk,jblk)=dx; mvy(iblk,jblk)=dy; %record the estimated MV
end;end;
```

Example: Half-pel EBMA

Pros and Cons with EBMA

- Blocking effect (discontinuity across block boundary) in the predicted image
  - Because the block-wise translation model is not accurate
  - Fix: Deformable BMA
- Motion field somewhat chaotic
  - because MVs are estimated independently from block to block
  - Fix 1: Mesh-based motion estimation
  - Fix 2: Imposing smoothness constraint explicitly
- Noisy MV in the flat region, or if true motion is along the edge direction
- Nonetheless, widely used for motion compensated prediction in video coding
  - Because its simplicity and optimality in minimizing prediction error

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
  - Simplify the error measure (DFD) to reduce the computation involved for each candidate
- 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)

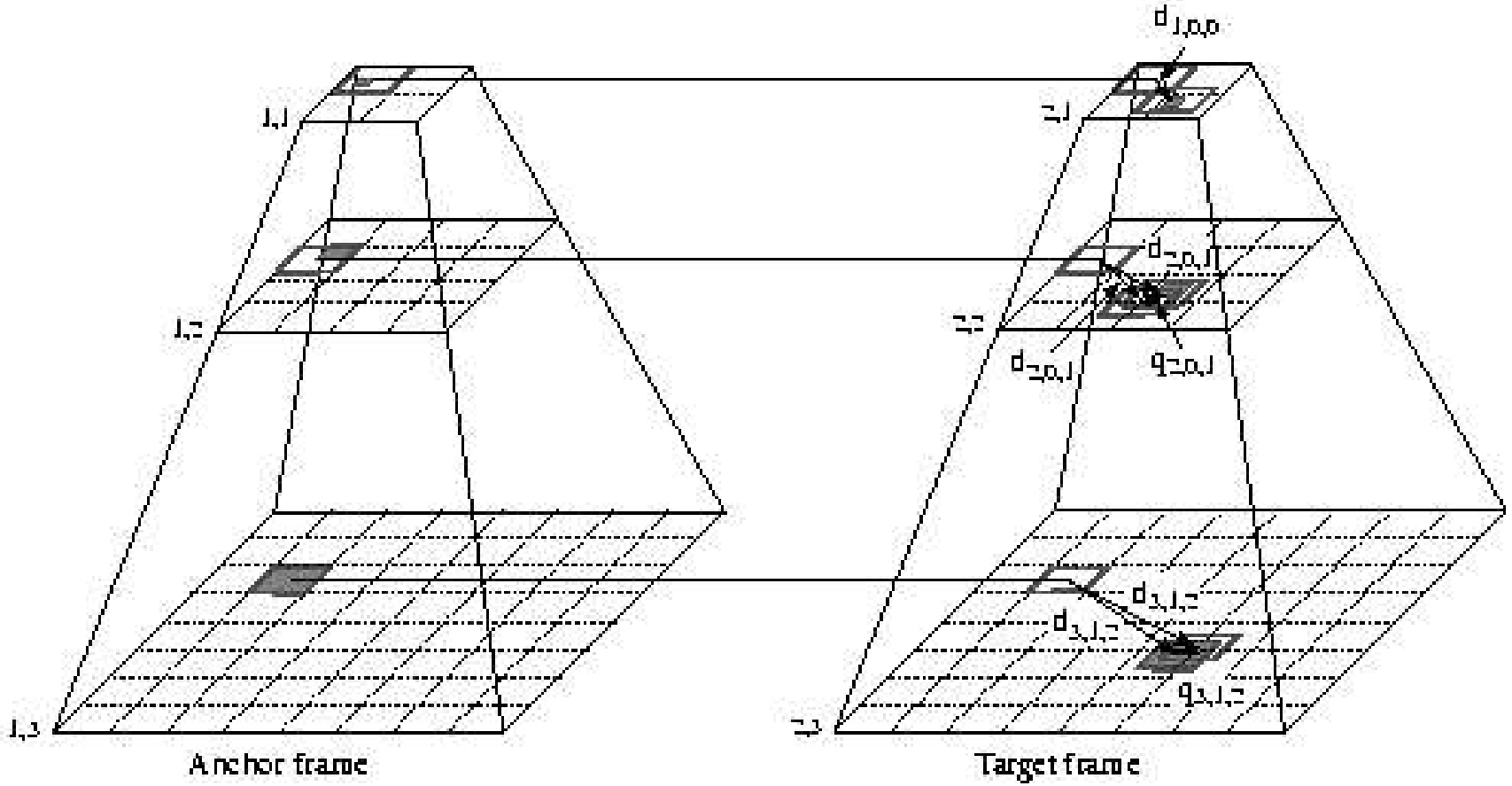
Pop Quiz (Homework)

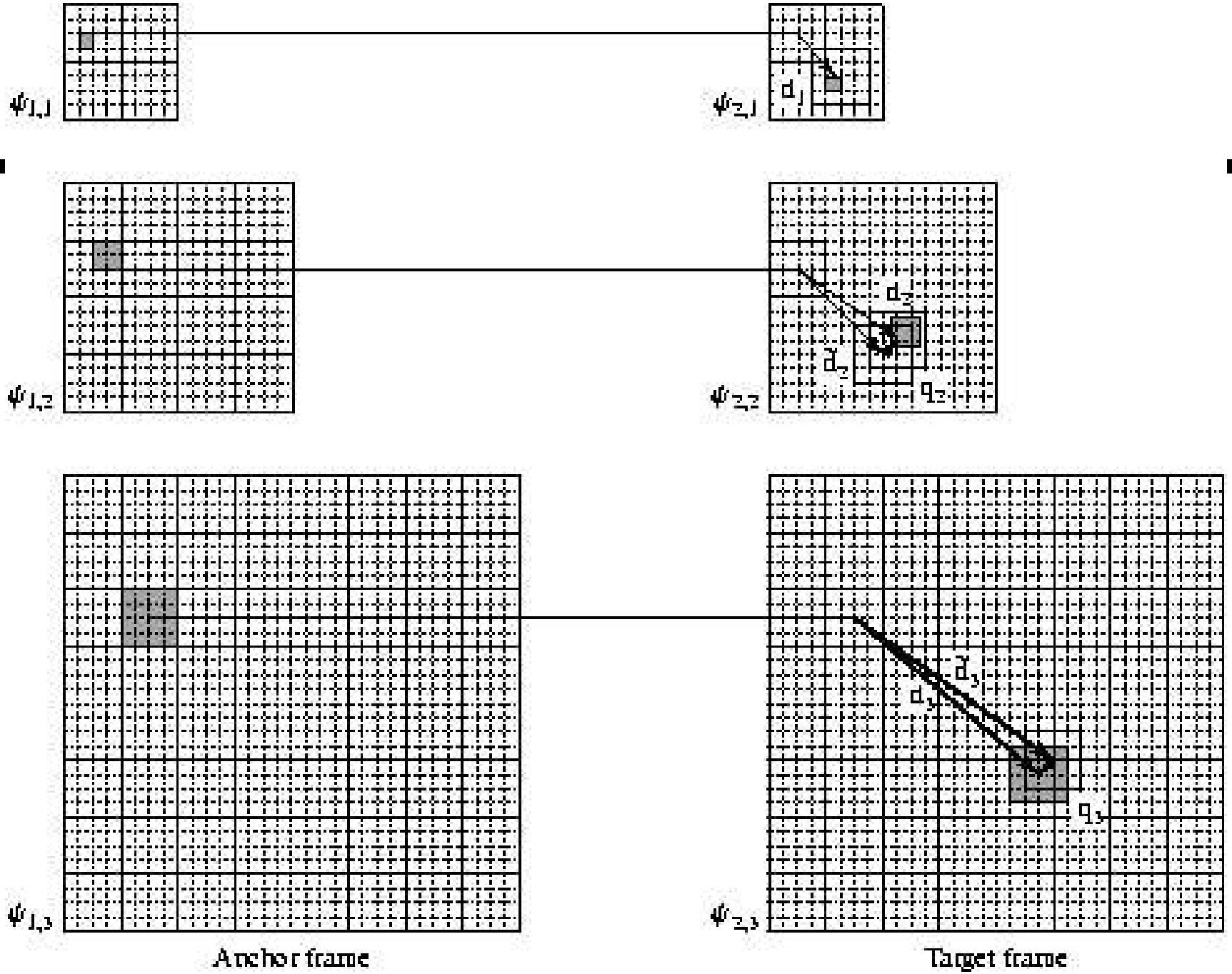
- What is the assumption of EBMA approach?
- Suppose you use a block size BxB, your search range is (-R to R) in both directions, and you only search for integer MVs. How many addition and multiplication you need to do for each block?
- Suppose your image size is MxN. How many addition and multiplication you need to do for the entire image? Does it depend on the block size?
- How do the above numbers change if you use half-pel accuracy search?

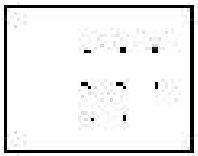
Multi-resolution Motion Estimation

- 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
  - Then 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)

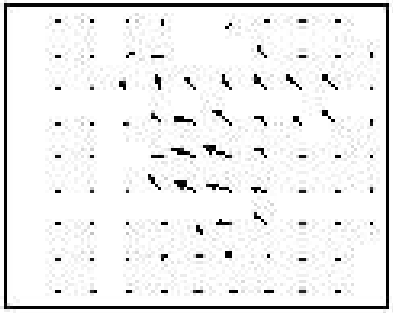



$\psi_{1,1}$
$\psi_{2,1}$
d_1
$\psi_{1,2}$
$\psi_{2,2}$
d_2
d_2
q_2
$\psi_{1,3}$
$\psi_{2,3}$
d_3
d_3
q_3
Anchor frame
Target frame



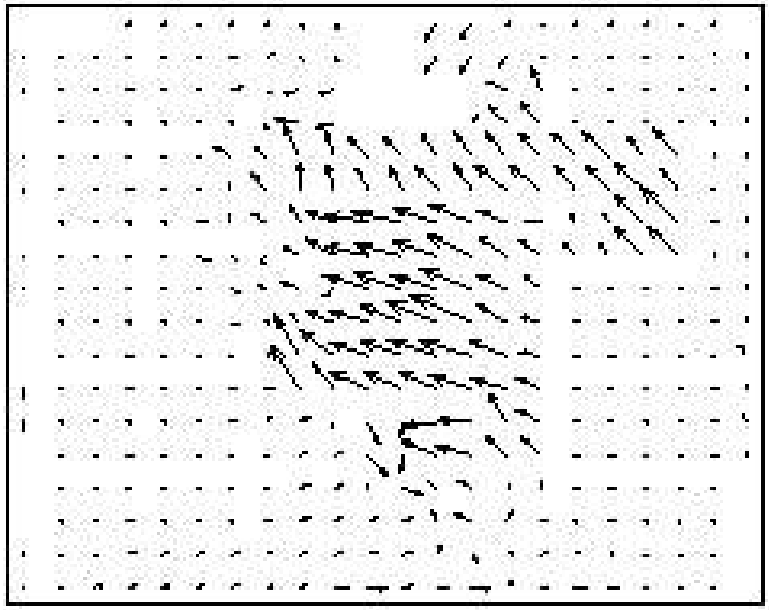
(a)

(b)



(c)

(d)

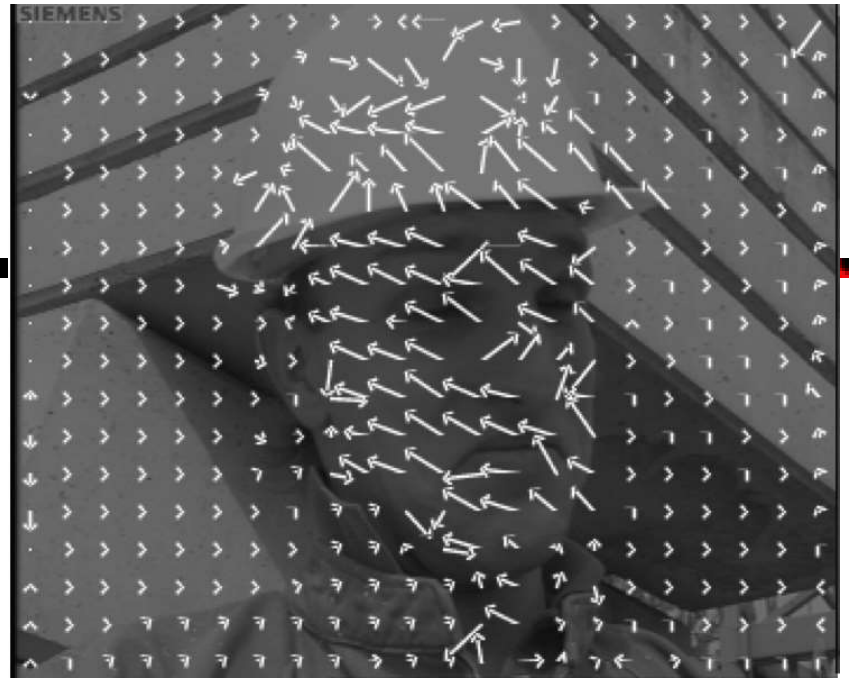


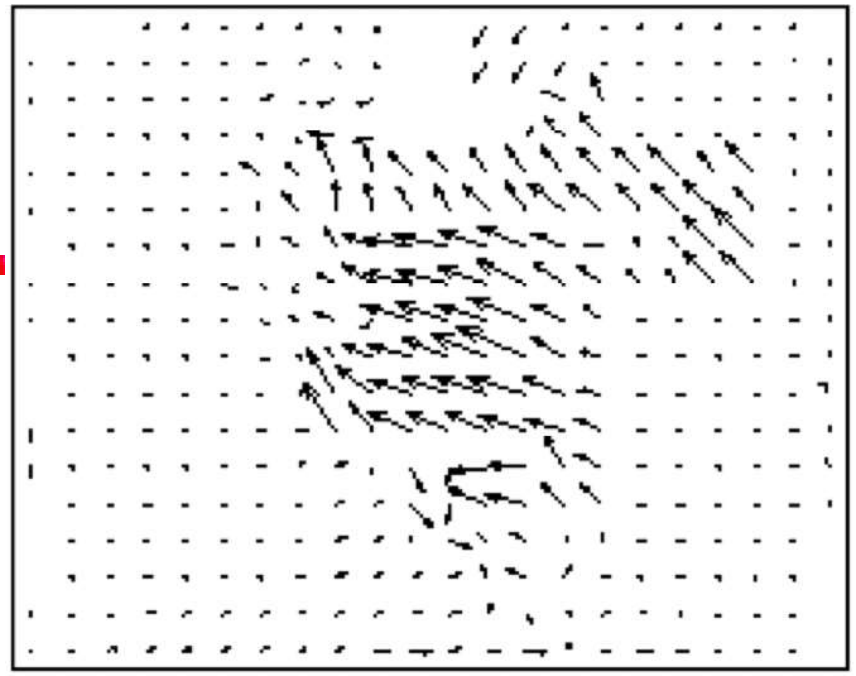
(e)

(f)

Predicted anchor frame (29.32dB)

Example: Three-level HBMA





EBMA, Half-Pel (29.86dB)

HBMA (29.32dB)

Computation Requirement of HBMA

- Assumption
  - 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) (can be smaller)
- 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))

$$\left(M/2^{L-l}\right)^2\left(2R/2^{L-1}+1\right)^2$$

- Total operation count

$$\sum_{l=1}^{L}\left(M/2^{L-l}\right)^2\left(2R/2^{L-1}+1\right)^2 \approx \frac{1}{3}4^{-(L-2)}4M^2R^2$$

- Saving factor:

$$3\cdot 4^{(L-2)} = 3(L=2);\ 12(L=3)$$

Pop Quiz (w/answers)

- What is the benefit of using multi-resolution search?
- Reduce complexity
- Enforce motion smoothness (often physically more accurate solution)

VcDemo Example

VcDemo: **Image and Video Compression Learning Tool**
Developed at Delft University of Technology

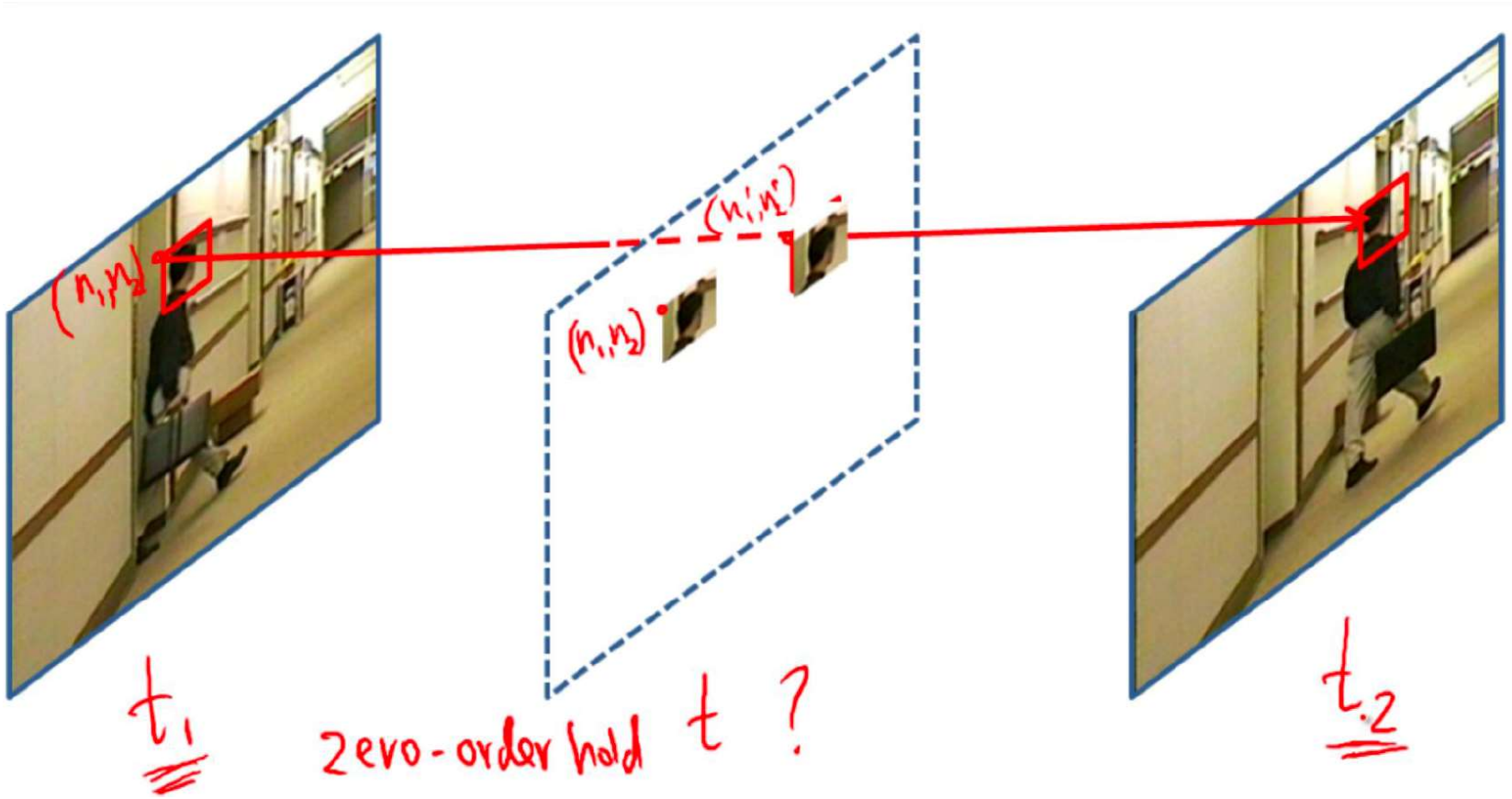
http://insy.ewi.tudelft.nl/content/image-and-video-compression-learning-tool-vcdemo

Use the ME tool to show the motion estimation results with different parameter choices

Applications of Dense Motion Estimation

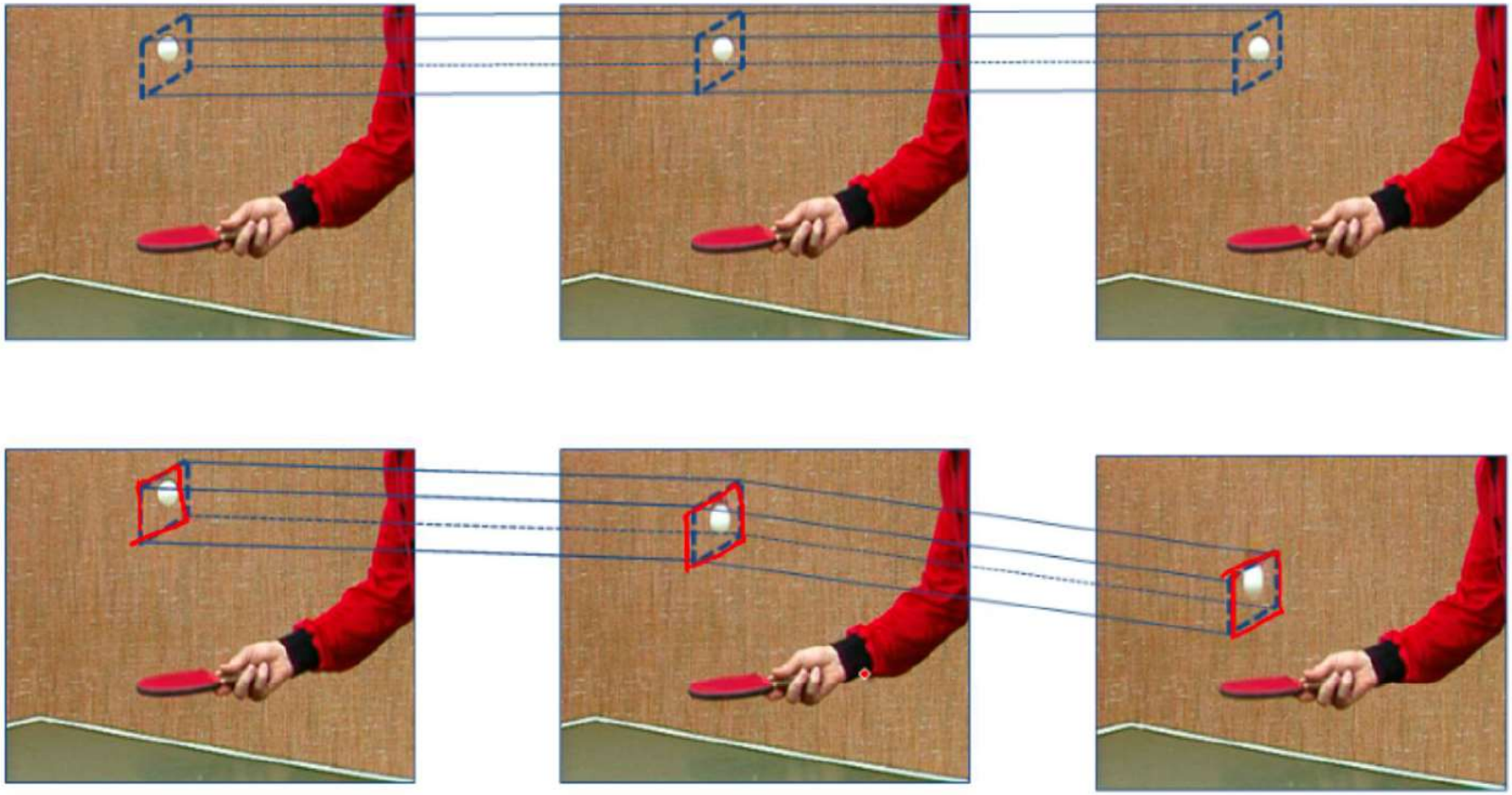
- Registration of two images with elastic deformation
  - Medical applications
  - Image morphing
- Video coding
  - Predicting the next frame with motion-compensated prediction
  - Code the error only
- Frame interpolation
  - Increasing frame rate
  - Filling in lost / missing frames due to transmission loss
- Video filtering
  - To remove noise
  - Temporal filtering along motion trajectory

Motion-Compensated Frame Interpolation



From Katsaggelos's Coursera Course on Image Processing, Lecture on motion estimation.

Motion Compensated Temporal Filtering



From Katsaggelos's Coursera Course on Image Processing, Lecture on motion estimation.

Summary 1: General Methodology

- What causes 2D motion?
  - Object motion projected to 2D
  - Camera motion
  - Optical flow vs. true 2D motion
- Constraints for 2D motion
  - Optical flow equation
  - Derived from constant intensity and small motion assumption
  - Ambiguity in motion estimation (uniquely determined only near corners)
- Estimation criterion:
  - DFD (constant intensity)
  - OF (constant intensity+small motion)
  - Regularization (motion smoothness or other prior knowledge/assumptions)
- Search method:
  - Exhaustive search, gradient-descent, multi-resolution
  - Least squares solution under optical flow equation

Summary 2: Motion Estimation Methods

- Pixel-based motion estimation (also known as optical flow estimation)
  - Most accurate representation, but also most costly to estimate
  - Need to put additional constraints on motion of neighboring pixels
  - Lucas-Kanade method
    - Assuming motion in the neighborhood is the same
  - How to handle large motion: iterative refinement, multiresolution
  - KLT tracker: apply LK method on feature points only
  - Automatically yield fractional accuracy
- Block-based motion estimation, assuming each block has a constant motion
  - Good trade-off between accuracy and speed
  - EBMA and its fast but suboptimal variant is widely used in video coding for motion-compensated temporal prediction.
  - HBMA can not only reduce computation but also yield physically more correct motion estimates

Python Tools for Motion Estimation

- This site describes some Python tools for motion estimation
  - http://docs.opencv.org/2.4/modules/video/doc/motion_analysis_and_object_tracking.html
- Some functions included:
  - cv2.calcOpticalFlowPyrLK()
  - This function computes the flow at a set of feature points, using pyramid representation
  - cv2.calcOpticalFlowFarneback()
  - This function computes dense flow (at every pixel)
- KLT tracker
  - https://github.com/TimSC/PyFeatureTrack (3rd party package)

Video capture/read/write in Python

- Please see following
- http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html

Other Techniques

- Deformable block matching algorithm (DBMA) ([Wang 2002])
  - To allow more complex motion within each block
  - Can set up equation for motion parameters using optical flow equation
- Mesh-based motion estimation (aka spline-based) ([Wang 2002])
  - Estimate MVs at grid points of a mesh, and interpolate MV at other points
  - Spline-based interpolation enforce motion smoothness
- Global motion estimation (next lecture)
- Segmentation of moving objects from background (next lecture)
- Block motion estimation using Phase correlation (optional material)
- Diffeomorphism for deformable image registration (optional material)
- Deep learning based methods

Deep-Learning Based Method

- [PWCnet] Deqin Sun, Xiaodong Yang, Ming-Yu Liu, Jan Kautz, PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. CVPR 2018. https://arxiv.org/abs/1709.02371

- [Flownet2] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks.," in CVPR . 2017, pp. 1647–1655, IEEE Computer Society
- https://lmb.informatik.uni-freiburg.de/Publications/2017/IMKDB17/

- [FlowFields] C. Bailer, B. Taetz, and D. Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In IEEE Int. Conference on Computer Vision (ICCV), 2015.
-

PWCnet: Matching in Feature Space

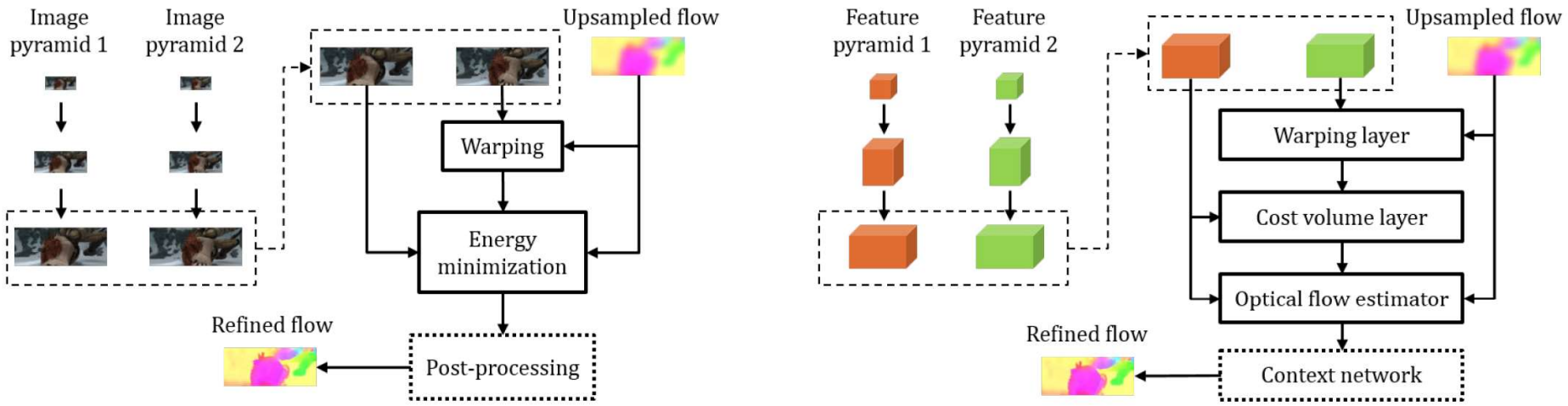


Figure 3. **Traditional coarese-to-fine approach vs. PWC-Net.** Left: Image pyramid and refinement at one pyramid level by the energy minimization approach [7, 9, 19, 45]. Right: Feature pyramid and refinement at one pyramid level by PWC-Net. PWC-Net warps features of the second image using the upsampled flow, computes a cost volume, and process the cost volume using CNNs. Both post-processing and context network are optional in each system. The arrows indicate the direction of flow estimation and pyramids are constructed in the opposite direction. Please refer to the text for details about the network.

[PWCnet] Deqin Sun, Xiaodong Yang, Ming-Yu Liu, Jan Kautz, PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. CVPR 2018. https://arxiv.org/abs/1709.02371

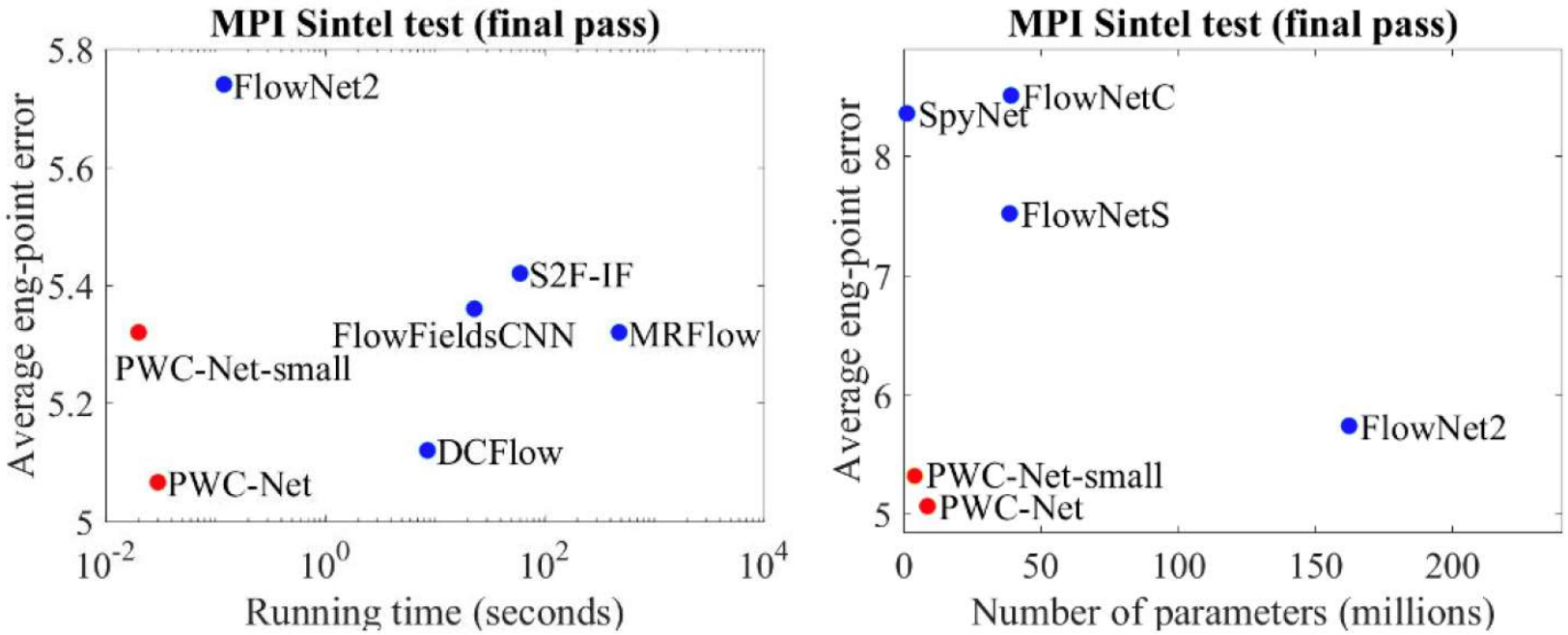


Figure 1. Left: PWC-Net outperforms all published methods on the MPI Sintel final pass benchmark in both accuracy and running time. Right: among existing end-to-end CNN models for flow, PWC-Net reaches the best balance between accuracy and size.

Reading Assignments

- [Szeliski2010] Richard Szeliski, Computer Vision: Algorithms and Applications. 2010. Chap. 8. (We covered most of 8.1, 8.4 in this lecture. You are encouraged to ready 8.3. We will cover 8.2 and 8.5 in the next lecture)
- Optional reading:
- [Wang2002] Wang, et al, Digital video processing and communications.
  - Chap 5: Sec. 5.1, 5.5
  - Chap 6: Sec. 6.1-6.6, Apx. A, B.
- Sun, Deqing, Stefan Roth, and Michael J. Black. "Secrets of optical flow estimation and their principles." In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2432-2439. IEEE, 2010.
- Hill, Derek LG, et al. "Medical image registration." *Physics in medicine and biology* 46.3 (2001): R1. http://research.mholden.org/publications/hill2001pmb.pdf

Motion Estimation and Segmentation Database

- http://vision.middlebury.edu/flow/data/
- http://people.csail.mit.edu/celiu/motionAnnotation/

- More recent
- MPI Sintel flow dataset (animation video)
  - http://sintel.is.tue.mpg.de/
- KITTI flow dataset 2015
  - http://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=flow

Written Assignment (1)

- Answer the quiz questions in the lecture note on slides #50 and #63.
- Consider two successive frames shown below. Using the Lucas-Kanade method to determine the optical flow vector of the center pixel. To determine the horizontal and vertical gradient image, you could simply use difference of two horizontally and vertically adjacent pixels. Your solution should show the horizontal, vertical and temporal gradient image, the moment matrix and the final solution. You should use a 3x3 neighborhood block surrounding the center pixel.

Frame t

0	0	10	10	10
0	0	10	10	10
0	0	10	10	10
10	10	10	10	10
10	10	10	10	10

Frame t+1

0	0	0	10	10
0	0	0	10	10
0	0	0	10	10
10	10	10	10	10
10	10	10	10	10

Written Assignments (2)

- Would you be able to derive the optical flow using the LK method, if the two frames look like the following? Why? What if you use a block matching method?

Frame t

0	0	10	10	10
0	0	10	10	10
0	0	10	10	10
0	0	10	10	10
0	0	10	10	10

Frame t+1

0	0	0	10	10
0	0	0	10	10
0	0	0	10	10
0	0	0	10	10
0	0	0	10	10

MATLAB Assignment (Optional)

1. [Wang2002] Prob. 6.12 (EBMA with integer accuracy)
2. [Wang2002] Prob. 6.13 (EBMA with half-pel accuracy)
3. [Wang2002] Prob. 6.15 (HBMA)

- Note: you can download sample video frames from the course webpage. When applying your motion estimation algorithm, you should choose two frames that have sufficient motion in between so that it is easy to observe effect of motion estimation inaccuracy. If necessary, choose two frames that are several frames apart. For example, foreman: frame 100 and frame 103.

- Try out many of the project ideas given in [Szeliski2010] 8.7.

Optional Material

Motion Estimation Through Phase Correlation

$$x_{t-1}(n_1, n_2) \leftrightarrow X_{t-1}(k_1, k_2) \qquad x_t(n_1, n_2) \leftrightarrow X_t(k_1, k_2)$$

Assume $$x_t(n_1, n_2) = x_{t-1}((n_1 - m_1)_{N_1}, (n_2 - m_2)_{N_2})$$ $N_1 \times N_2$, $0-(N_1-1)$, $0-(N_2-1)$

Then $$X_t(k_1, k_2) = X_{t-1}(k_1, k_2) e^{-j\frac{2\pi}{N_1} m_1 k_1} e^{-j\frac{2\pi}{N_2} m_2 k_2}$$

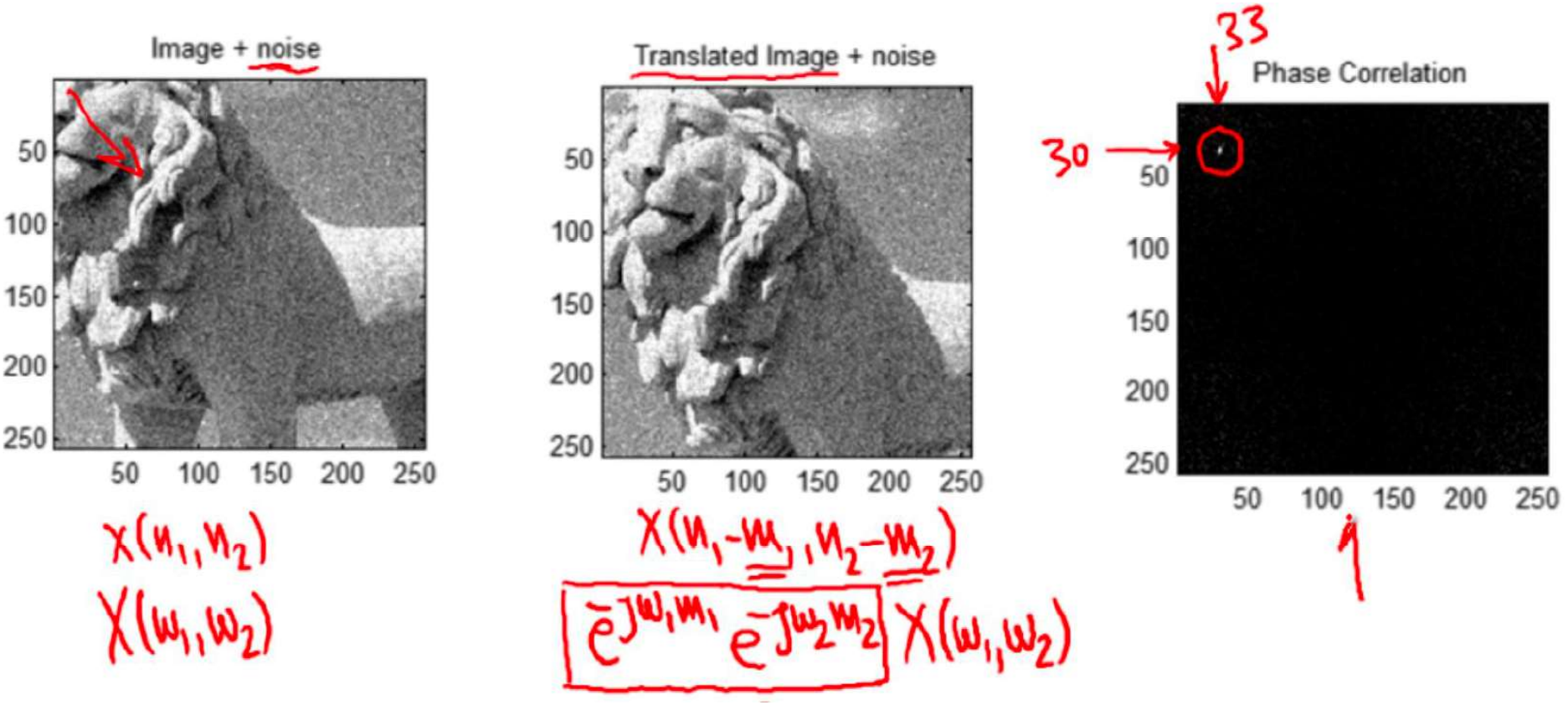
Form: $$C(k_1, k_2) = \frac{X_t(k_1, k_2) \cdot X^*_{t-1}(k_1, k_2)}{|X_t(k_1, k_2) X^*_{t-1}(k_1, k_2)|} = \frac{|X_{t-1}(k_1, k_2)|^2 \, e^{-j\frac{2\pi}{N_1} m_1 k_1} e^{-j\frac{2\pi}{N_2} m_2 k_2}}{|X_{t-1}(k_1, k_2)|^2}$$

(complex conjugate)

$$c(n_1, n_2) = \delta(n_1 - m_1, n_2 - m_2)$$

From Katsaggelos's Coursera Course on Image Processing, Lecture on motion estimation.

Motion Estimation Through Phase Correlation



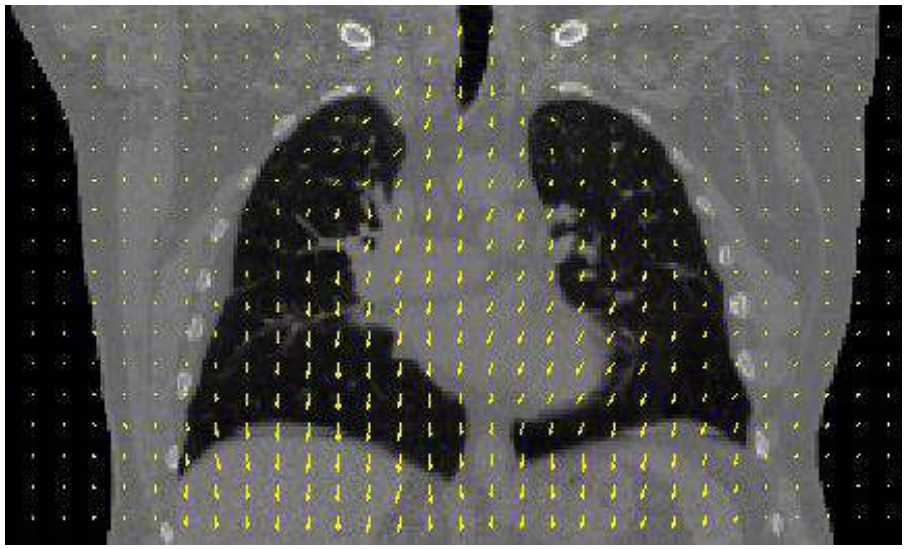
From Katsaggelos's Coursera Course on Image Processing, Lecture on motion estimation.

What if different regions move differently?

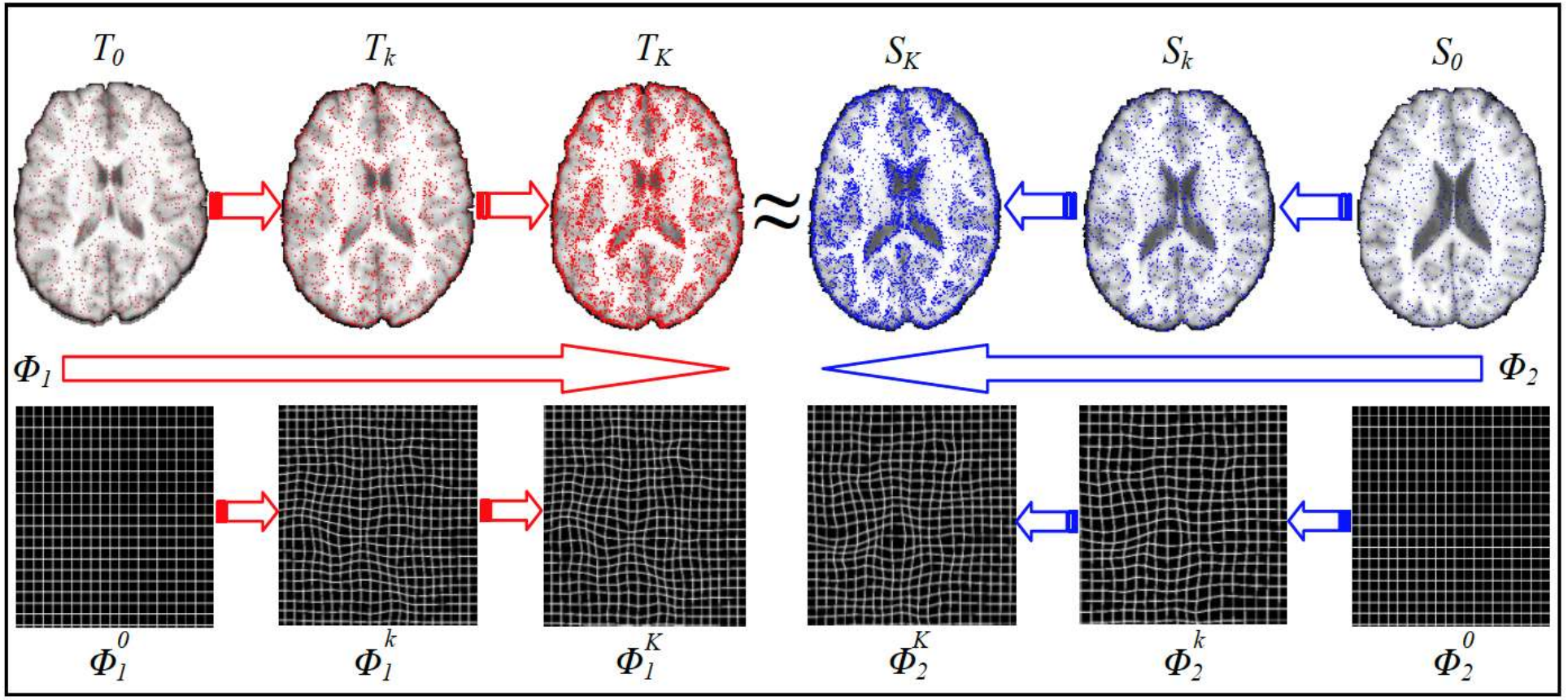
- If we apply FT on entire images, we may see multiple peaks
- Apply phase correlation methods at block level, but need to apply FT on enlarged blocks that cover both the original block and the new block positions

Deformable Image Registration

- In medical applications, we often need to align two images of the same patient taken during different times, or between one image and a reference (atlas).
- Find point-wise correspondence between two images, so that one can be warped to the other.
- Essentially we need to find the dense motion field between the two images
- The registration often has to be done on a 3D volume, as corresponding pixels may not be on the same slice



Diffeomorphism Mapping



- Diffeomorphism Mapping: continuous and invertible

Formalization of Deformable Registration as a Minimization Problem

Images (target I_T, source I_S): $I : \Omega \to \mathbb{R}$

Image domain: $\Omega \subset \mathbb{R}^d$, $d=2,3$

Warped Source Image

$$u' = \arg\min_{u \in \mathrm{H}} E_D(I_T, I_S(\varphi)) + \lambda E_R(u)$$

Displacements are elements of a Hilbert space H, e.g.: $u \in \mathrm{L}^2$

Deformation: $\varphi = \mathrm{Id} + u$, $\varphi : \Omega \to \mathbb{R}^d$

or point-wise: $\varphi(x) = x + u(x)$

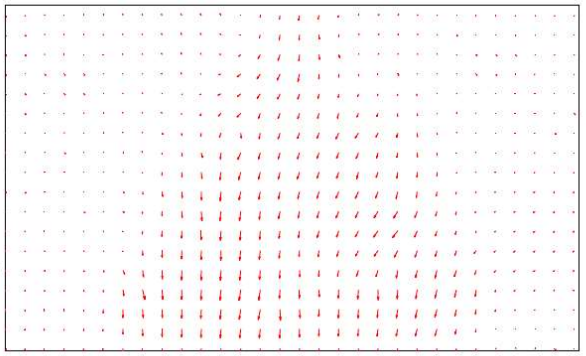
Please note: Highly heterogeneous notation in the field

Displacement:

$u : \Omega \to \mathbb{R}^d$

e.g. $u = [u_x, u_y, u_z]$

difference between original position of a point x and its transformation φ(x)



From: http://campar.in.tum.de/twiki/pub/DefRegTutorial/WebHome/MICCAI_2010_Tutorial_Def_Reg_Darko.pdf

Why do we need Regularization?

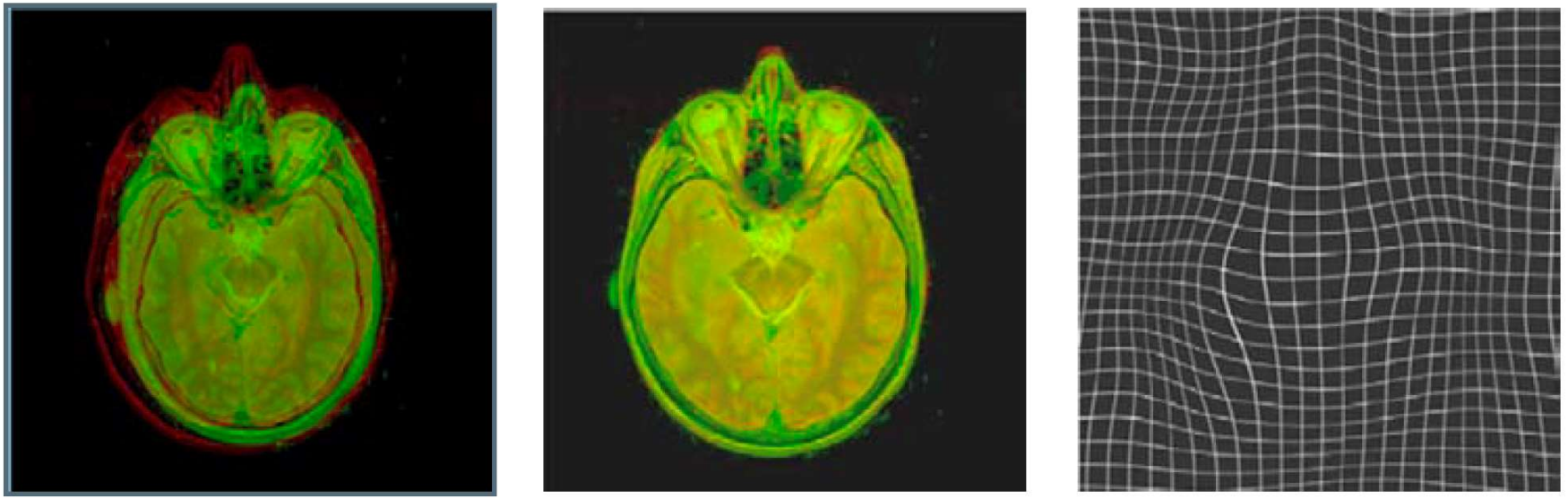
Since minimizing the difference measure is not enough...

- Motivation for regularization:
  - → **Necessity:**
    Minimization of difference measure only can be ill-posed (#measurements < #unknowns)
    (Optical Flow community: "*Aperture Problem*")
  - → **Modeling:**
    Regularization can be used to include prior knowledge, for example about underlying tissue properties
  - → **Practical Reasons:**
    Without regularization: High number of local minima in the energy function (→ bad for optimization)

Many ways to impose regularization → Different methods

From: http://campar.in.tum.de/twiki/pub/DefRegTutorial/WebHome/MICCAI_2010_Tutorial_Def_Reg_Darko.pdf

Example of Deformable Registration



(a) (b) (c)

Figure 8.10 Elastic brain registration (Kybic and Unser 2003) © 2003 IEEE: (a) original brain atlas and patient MRI images overlaid in red–green; (b) after elastic registration with eight user-specified landmarks (not shown); (c) a cubic B-spline deformation field, shown as a deformed grid.

From [Szeliski2010]

Demon's Algorithms

- Basic Idea
  - Iteratively update the motion vector at each pixel based on the image gradient (image warped after iteration)
  - Smooth the new motion field using a Gaussian kernel

Loop until convergence:

$$h = \frac{(I_{\mathrm{T}} - I_{\mathrm{S}}(\varphi_u))\nabla I_{\mathrm{S}}(\varphi_u)}{\|\nabla I_{\mathrm{S}}(\varphi_u)\|^2 + (I_{\mathrm{T}} - I_{\mathrm{S}}(\varphi_u))^2}$$

$$u = u + \tau h$$

$$u = G^{\sigma} * u$$

end

Thirion, J-P. "Image matching as a diffusion process: an analogy with Maxwell's demons." *Medical image analysis* 2.3 (1998): 243-260.

From: http://campar.in.tum.de/twiki/pub/DefRegTutorial/WebHome/MICCAI_2010_Tutorial_Def_Reg_Darko.pdf

Python for Demons' Registration Algorithm

- Package : SimpleITK
- Install : conda install -c simpleitk simpleitk=0.10.0
- Detail :http://insightsoftwareconsortium.github.io/SimpleITK-Notebooks/66_Registration_Demons.html