Master in
Computer Vision
Barcelona

Module M6 : Video Analysis

Lectures 3 & 4: Motion estimation. Optical flow

Lecturer: Ramon Morros

Outline

- **Introduction**
- **Direct exploration technique:**
  - Block matching
- **Gradient-based techniques:**
  - Frame-level: Affine model
  - Dense flow: local approach: Lucas – Kanade
  - Dense flow: global approach with smoothness constraints: Horn-Schunck
- **Other techniques:**
  - Brox
  - Deep learning methods
- **Feature tracking**

Motion – Introduction (I)

- **Motion field**: projection of 3D scene motion into the image
- Camera motion adds to the motion field
- Optical flow: **apparent** 2D motion of brightness patterns in the image
  - Note: apparent motion can be caused by lighting changes without any actual motion

> - Motion field is what we want to know.
> - Optical flow is what we can estimate.

- **GOAL**: Recover image motion at each pixel from optical flow

Motion – Introduction (II)

- Applications
  - Video coding
    - Exploit temporal redundancy in videos
    - Motion compensation + error coding
  - Special effects

    http://www.fxguide.com/featured/art_of_optical_flow/
  - Structure from motion
    - Estimate 3D structures from 2D image sequences
  - Tracking
    - Locate objects over time
  - Many others …

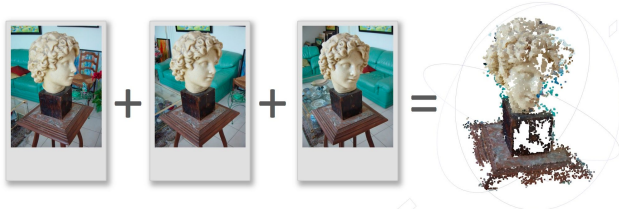


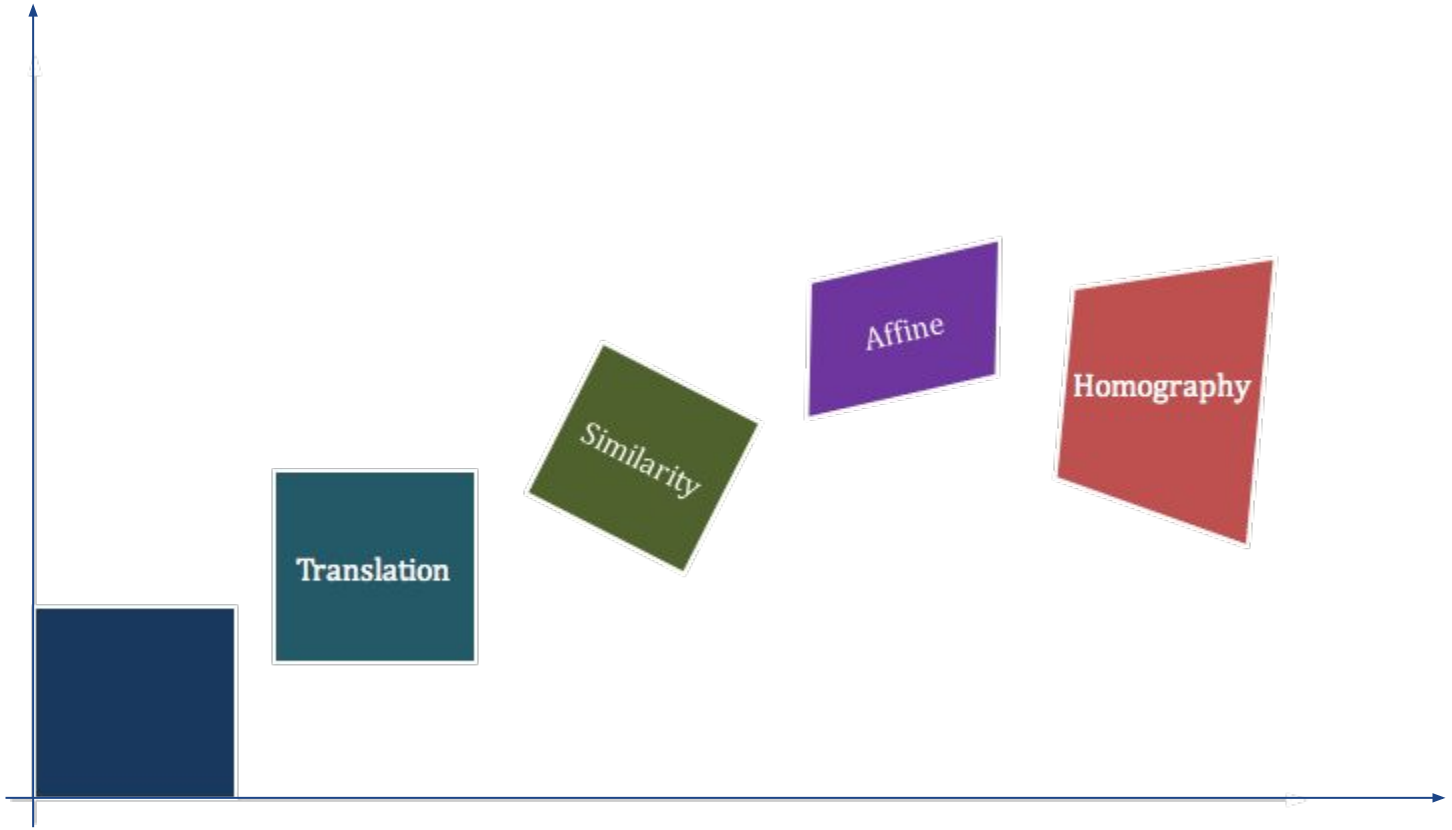
Image: Jianxiong Xiao, Princeton Vision & Robotics

Motion – Parametric motion models (I)

- Each image zone that presents a **homogeneous motion** is assigned a given type of motion model (e.g.: translational) with specific parameters (e.g.: d_x = 3 pixels, d_y = -5 pixels):
  - **Segmentation problem**: Image zones with homogeneous motion have to be determined.
  - **Restrictions on the motion estimation**: The use of a specific parametric motion model imposes constraints in the final optical flow.

- The **most common types of parametric motion models** are:
  - Translational model
  - Affine model

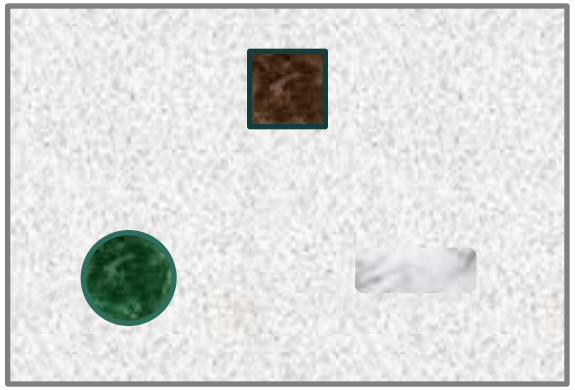
Motion – Parametric motion models (II)

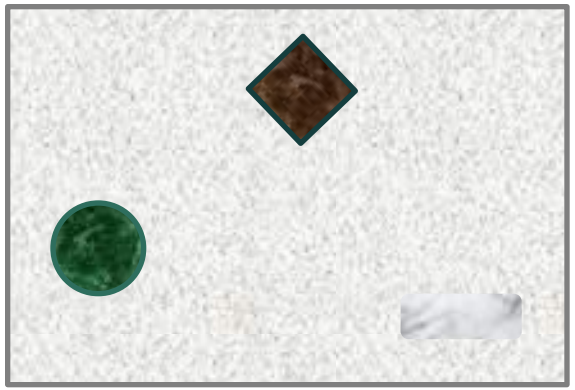


Source: Ce Liu (celiu@microsoft.com)

Motion – Parametric motion models (III)

- Example of the problems in parametric motion modeling. Let us assume that **translational motion** is imposed:





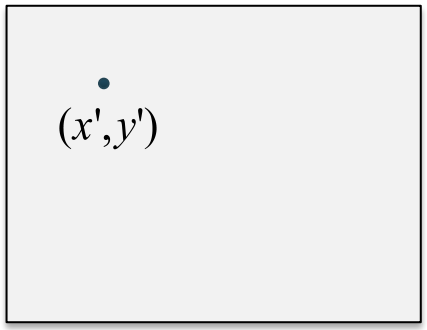
- **Background**: Static
  d_x = 0 pixels, d_y = 0 pixels
- **Green ball**: Translational motion
  d_x = -3 pixels, d_y = -7 pixels
- **Grey rectangle**: The boundaries of the object are difficult to define. The motion parameters are difficult to estimate.
- **Brown square**: Type of motion (rotation) does not fit the model. If the motion model allows rotation, every optical flow component has to be computed anyway for each pixel. **How?**

$$\begin{pmatrix} d_x \\ d_y \end{pmatrix} = \begin{pmatrix} a_1 & a_2 \\ a_4 & a_5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a_3 \\ a_6 \end{pmatrix} \qquad \forall (x, y) \in R$$

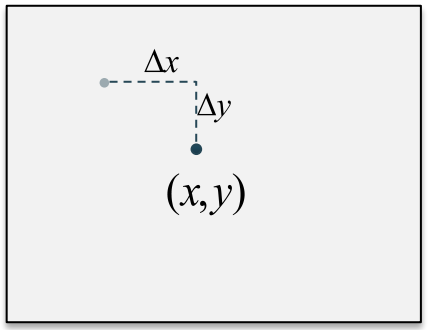
Optical flow - Hypothesis

- Changes in pixel values (color/gray level) are **only due to the motion** in the scene
- A pixel value in the image at time t -Δt can always be found (and associated to a pixel) in the image at time t.

$$(X', Y', Z') \rightarrow (x', y')\Big|_{t-\Delta t} \qquad (X, Y, Z) \rightarrow (x, y)\Big|_{t}$$



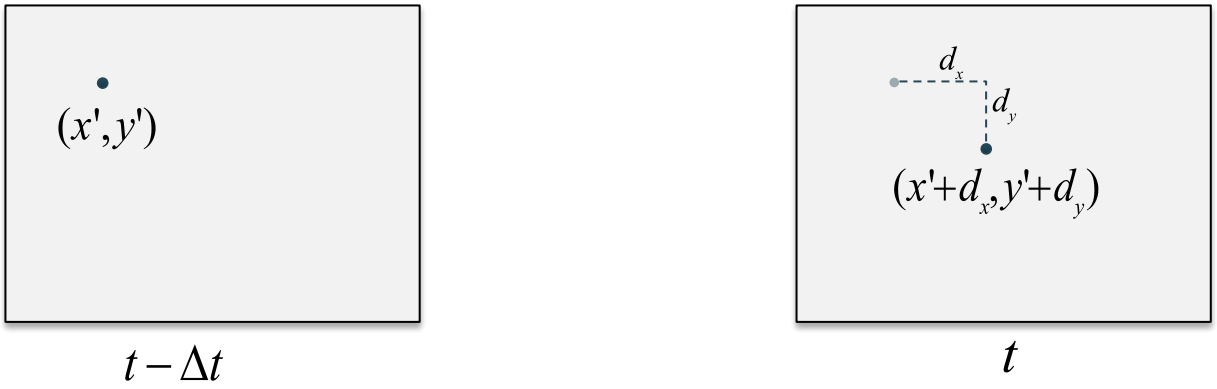
$I_{t-\Delta t}$



I_t

$$x = x' + \Delta x$$
$$y = y' + \Delta y$$

Optical flow – Brightness constancy



Brightness constancy equation:

$$I(x, y, t) = I(x', y', t - \Delta t) = I(x - \Delta x, y - \Delta y, t - \Delta t)$$

$$I(\vec{r}, t) = I(\vec{r} - \vec{D}(\vec{r}), t - \Delta t)$$

$$\vec{D} = (\Delta x(x, y), \Delta y(x, y))$$

Optical flow – Motion compensated image

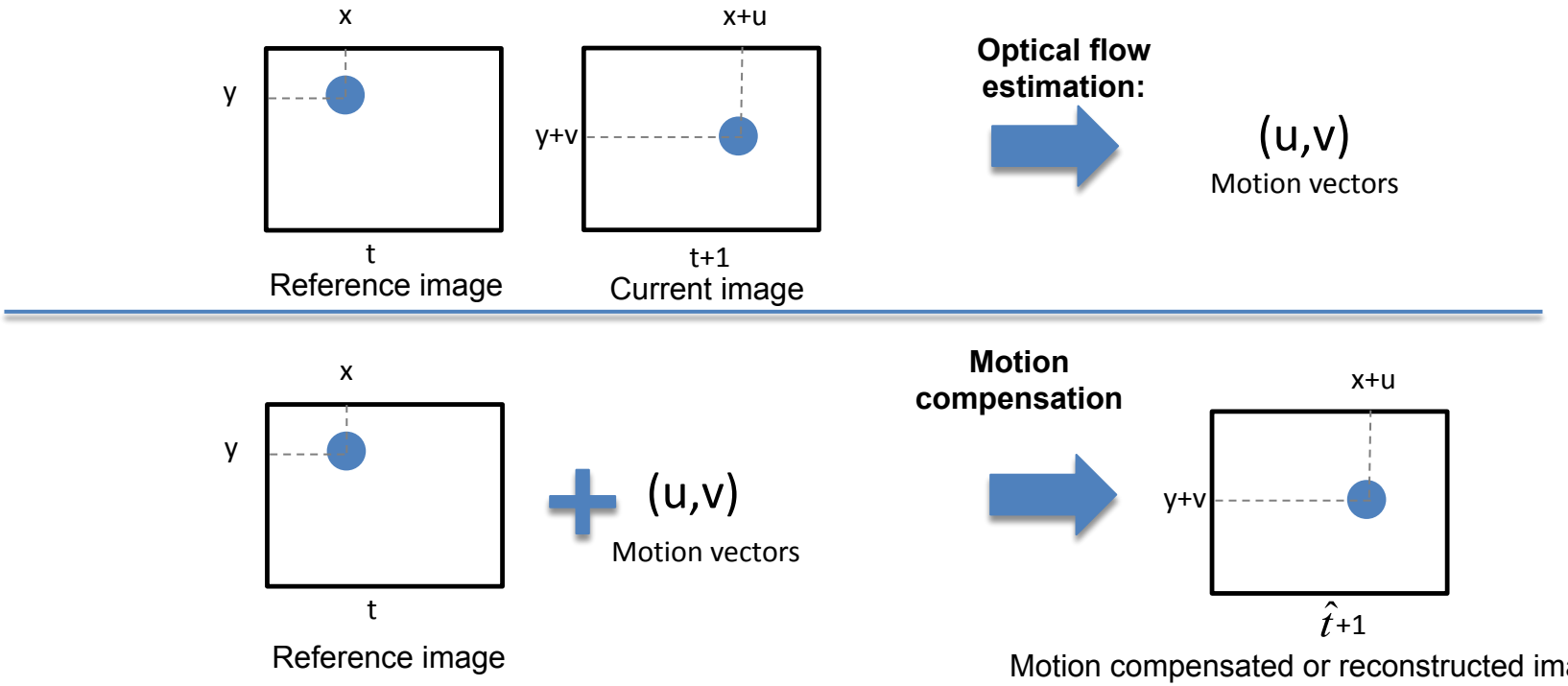
- $\boldsymbol{D}(\boldsymbol{r})$ is a **vector function** that contains the optical flow values. It associates to each pixel a displacement vector

$$I(\vec{r}, t) = I(\vec{r} - D(\vec{r}), t - \Delta t)$$

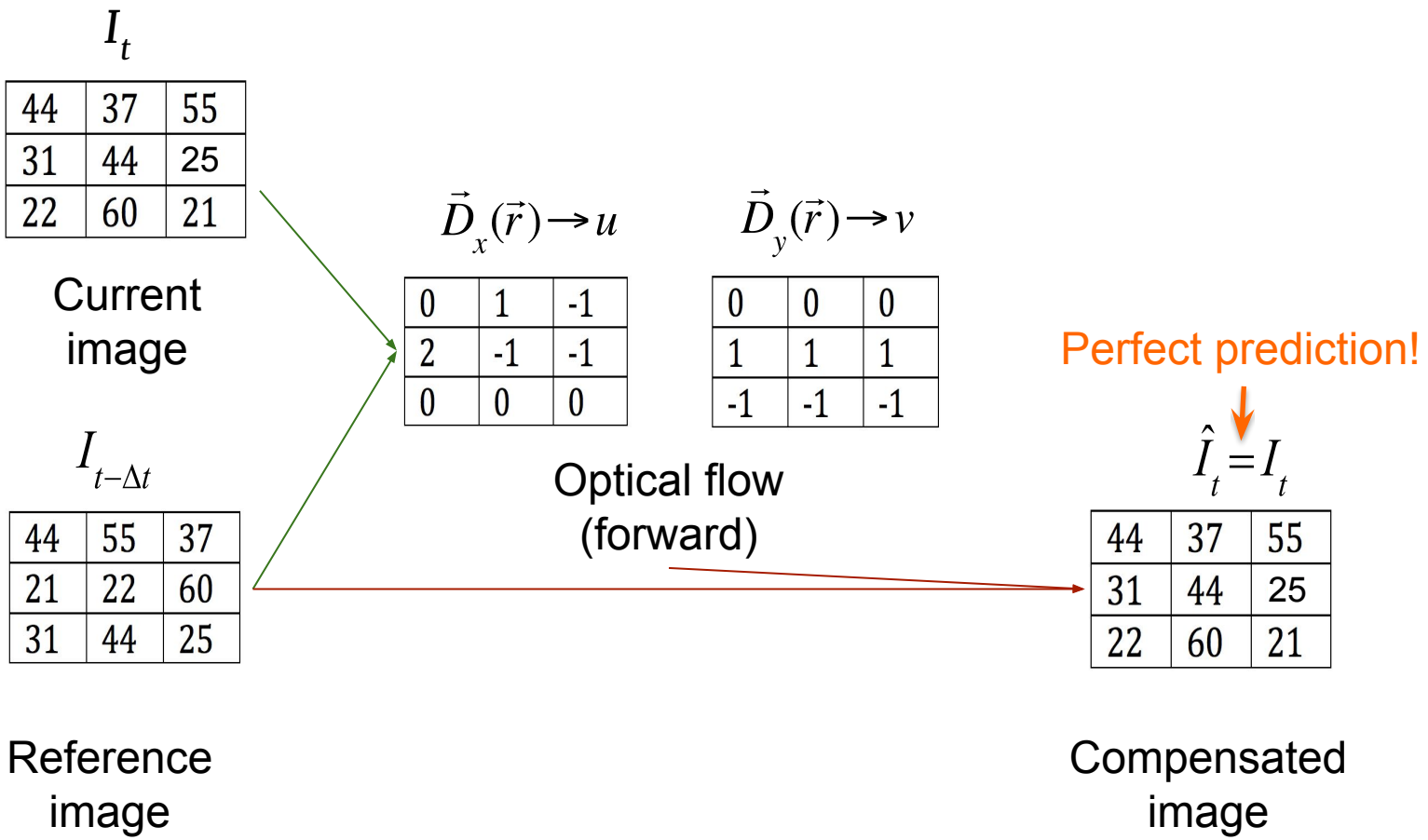
- The previous hypothesis implies that $I(\mathbf{r}, t)$ can be **perfectly predicted** by a reference image $I(\mathbf{r}, t\text{-}\Delta t)$, thanks to the optical flow.
- Pixels in the reference image are **compensated**, through the optical flow, to be co-located with their related pixels in the current image.
- The reference image to which the optical flow has been applied is known as **motion compensated image**.

Optical flow – Motion compensated image

- Motion compensation describes a picture in terms of the transformation of a reference picture to the current picture
- Images are synthesized from previously transmitted/stored images plus motion information



Optical flow – Motion compensated image



Optical flow – Displaced Frame Difference (I)

The hypothesis $I(\vec{r},t) = I(\vec{r} - D(\vec{r}), t - \Delta t)$ is too restrictive and, quite often, no optical flow can be found actually fulfilling it.

In order to relax the constraint introduced by this hypothesis, an **estimation of the optical flow** is assumed.

As a result, the **Displaced Frame Difference** (DFD) is defined as:

$$DFD\left(\vec{r}, \hat{\vec{D}}(\vec{r})\right) = I(\vec{r},t) - I(\vec{r} - \hat{\vec{D}}(\vec{r}), t - \Delta t)$$

Optical flow – Displaced Frame Difference (II)

$$DFD\left(\vec{r}, \hat{\vec{D}}(\vec{r})\right) = I(\vec{r}, t) - I(\vec{r} - \hat{\vec{D}}(\vec{r}), t - \Delta t)$$

- Given an **estimation of the optical flow** between the current image and a reference image, the **DFD** is an image that contains the difference between every pixel in the current image and its associated pixel in the **reference image**.
- The **closer to 0 are the DFD values**, the closer to fulfill the hypothesis is the estimated optical flow, and the better the estimated optical flow is supposed to be.
- DFD values provide a measure of the **optical flow estimation quality**.

Optical flow – DFD based computation

- **Criterion**: Look for an (approximated) optical flow that minimizes a **measure of error based on the DFD**.

$$\hat{\vec{D}}(\vec{r}) = \arg\min_{\vec{D}(\vec{r})} \left[\sum_{\vec{r}\in\mathbb{R}} \left\| DFD\left(\vec{r}, \vec{D}(\vec{r})\right) \right\| \right]$$

R: The whole image.

$$\hat{\vec{D}}(\vec{r}) = \arg\min_{\vec{D}(\vec{r})} \left[\sum_{\vec{r}\in\mathbb{R}} \left\| I(\vec{r}, t) - I(\vec{r} - \hat{\vec{D}}(\vec{r}), t - \Delta t) \right\| \right]$$

- Typically, the **absolute error** or the **square error** is used:
  - Absolute error: **fast to compute**
  - Square error: **easy to derive mathematical properties**

Optical flow – Classification of techniques

- It is a **nonlinear optimization** problem, defined over **several** (a large number of) **dimensions**:
  - Nonlinear since it involves **absolute** or **square** functions and *D* is a function of *I*.
  - Several dimensions since, in the worst case, **two displacement components** have to be obtained **for each pixel**.

Techniques can be classified depending on:

- **Data used in the process**:
  - **Direct exploration techniques**: only the values of the original functions (images) are used.
  - **Differential or gradient based techniques**: the values of the original functions (images) as well as their derivatives are used.
- **Motion constraints:**
  - **Non-parametric models**: the optical flow values are not constrained by any parametric function (Very high dimension).
  - **Parametric models**: the optical flow is parameterized over an area that is assumed to have homogenous motion (Lower dimension).

Optical flow – DFD definition with parametric motion models

$$DFD\left(\vec{r}, \hat{\vec{D}}(\vec{r})\right) = I(\vec{r}, t) - I(\vec{r} - \hat{\vec{D}}(\vec{r}), t - \Delta t)$$

When a parametric motion model is used, the optical flow depends on the pixel position ($\mathbf{r}$) and the model parameters (p_i) assigned to this region:

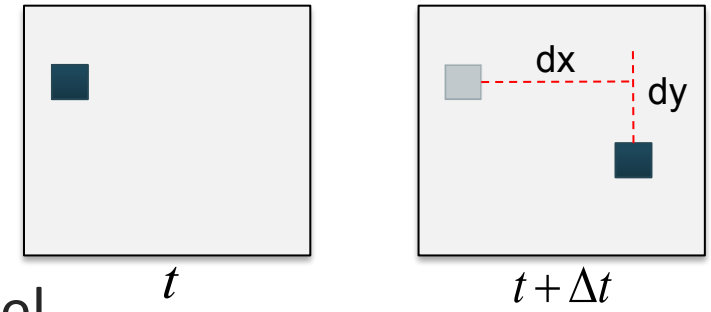
$$DFD\left(\vec{r}, \hat{\vec{D}}(\vec{r}, p_i)\right) = I(\vec{r}, t) - I(\vec{r} - \hat{\vec{D}}(\vec{r}, p_i), t - \Delta t)$$

Optical flow – Parametric motion models (I)

Translational vs. affine motion models

dx = x' - x, dy = y' - y

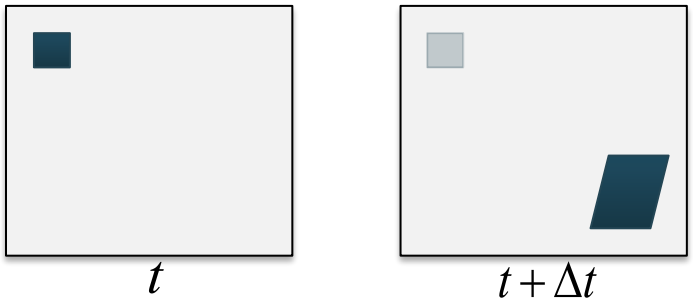
- Translational model:
  - Motion of a group of pixels can be described using a translation



$$\vec{r}' = \vec{r} + \boldsymbol{b}$$

$$\boldsymbol{b} = \begin{bmatrix} b_x \\ b_y \end{bmatrix}$$

- Affine model
  - Motion of a group of pixels can be described using a translation and a deformation



$$\vec{r}' = \boldsymbol{A}\vec{r} + \boldsymbol{b}$$

$$A = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} b_x \\ b_y \end{bmatrix}$$

Optical flow – Parametric motion models (II)

Translational vs. affine motion models

- Translational model:

$$\vec{r}' = X + \boldsymbol{b}$$

$$\boldsymbol{b} = \begin{bmatrix} b_x \\ b_y \end{bmatrix}$$

$$I(\vec{r} + \boldsymbol{b}, t + \Delta t) = I(\vec{r}', t)$$

$$x' = x + b_x$$
$$y' = y + b_y$$

- Affine model:

$$\vec{r}' = \boldsymbol{A}X + \boldsymbol{b}$$

$$A = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} b_x \\ b_y \end{bmatrix}$$

$$I(\boldsymbol{A}\vec{r} + \boldsymbol{b}, t + \Delta t) = I(\vec{r}', t)$$

$$x' = a_1 \cdot x + a_2 \cdot y + b_x$$
$$y' = a_3 \cdot x + a_4 \cdot y + b_y$$

Optical flow – Computation

- **Criterion**: Look for the optical flow that minimizes the **error based on the DFD**. The minimization can be done region by region:

$$\hat{\vec{D}}(\vec{r}, p_i) = \arg\min_{p_i} \left[\sum_{\vec{r} \in \mathbb{R}} \left\| DFD\left(\vec{r}, \vec{D}(\vec{r}, p_i)\right) \right\| \right]$$

R: a given region.

$$DFD\left(\vec{r}, \hat{\vec{D}}(\vec{r}, p_i)\right) = I(x, y, t) - I(x - d_x(x, y, p_i), y - d_y(x, y, p_i), t - \Delta t)$$

  - This is an optimization problem with a **reduced number of dimensions**:
    - Number of regions x Number of model parameters
  - **Two main types** of approaches exist:
    - **Direct exploration** and **Gradient** techniques

Outline

- Introduction
- **Direct exploration technique:**
  - Block matching
- Gradient-based techniques:
  - Frame level: Affine model
  - Dense flow: local approach: Lucas – Kanade
  - Dense flow: global approach with smoothness constraints: Horn-Schunck
- Other techniques:
  - Brox
  - Deep learning methods
- Feature tracking

Direct exploration – Definition of the solution space

For each region R:

- The solution is searched in an **n-dimensional space** (n : number of parameters of the model)
- The possible parameters values are **bounded** and **quantized**:
  - **Bounded**: Maximum acceptable **variation**
  - **Quantized**: Minimum acceptable **precision**

$$P_i < p_i \leq P_i + L_i \quad i = 1, 2, \cdots, n \qquad p_i \in \{P_i, P_i + \Delta_1, \cdots, P_i + \Delta_k, \cdots P_i + L_i\}$$

- The function to be optimized (error measure) is evaluated at **a subset of points of the n-dimensional space** (possible motion parameter values defined by the bounding and the quantization):
  - **Exhaustive search**: The absolute minimum is found but the search may be time consuming.
  - **Fast search techniques**: Faster, suboptimal solutions.

Direct exploration – Block Matching (I)

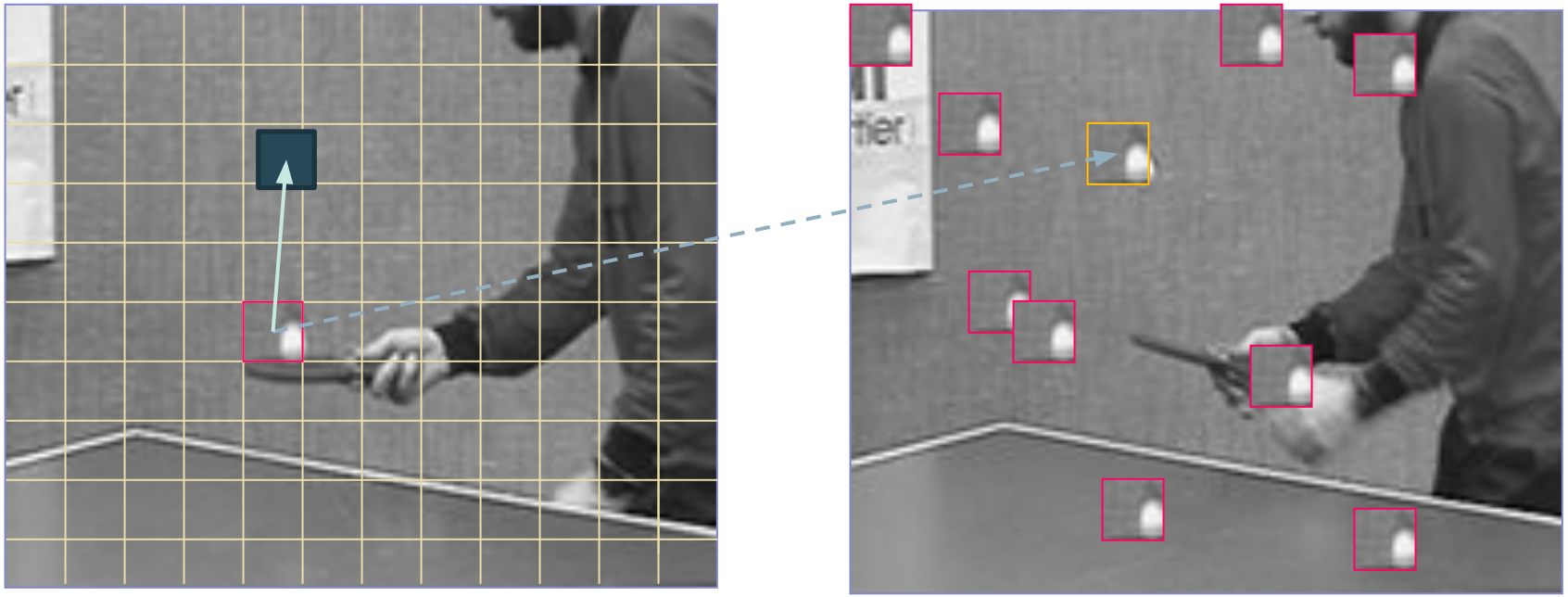
Simple & extensively used example

- **Region definition**: fixed division of the image into square blocks
  - Issue: The **block size** has to be defined.
- **Motion model**: All the pixels in a block are assumed to move with a **translational motion**.
  - Issue: The **set of possible displacements** (motion parameter values) has to be defined.
- **Algorithm**:
  - For each block of the image under study, the algorithm searches the block of the reference image with **minimum Euclidean distance (maximum correlation)**.
  - The **difference of position** of both blocks defines the block displacement.

Direct exploration – Block Matching (II)

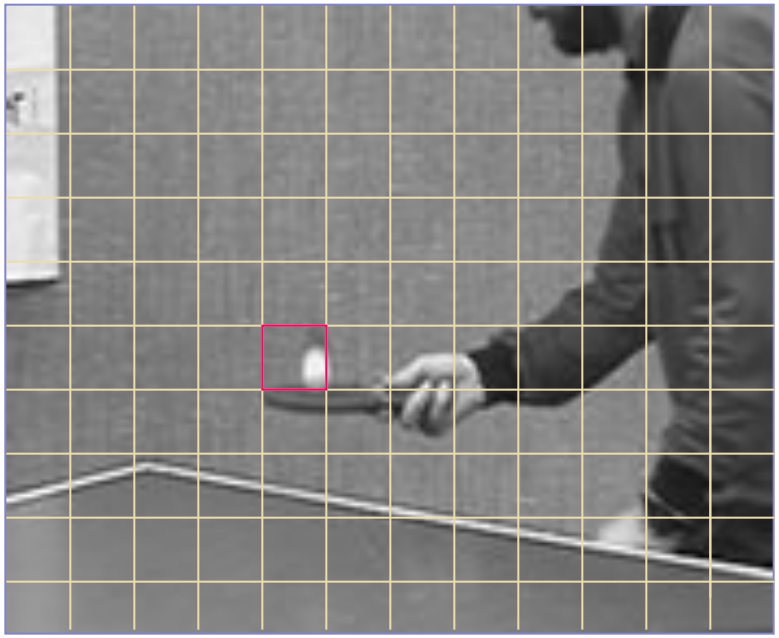
- **Region definition**: A fixed division of the image into **square blocks**.
- **Motion model**: All the pixels in a block are assumed to move with a **translation**.

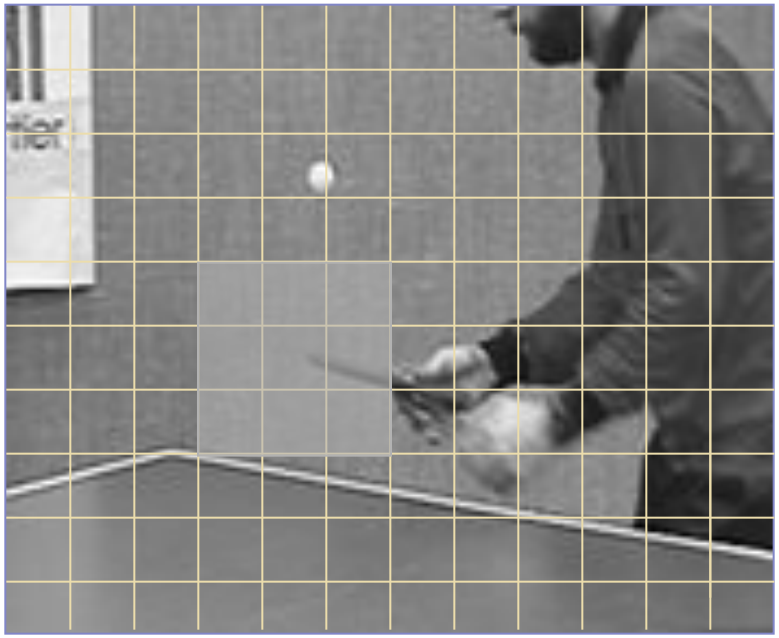
- In the example, the displacement parameters have not been bounded: the **search window is the whole image**.



Direct exploration – Block Matching (III)

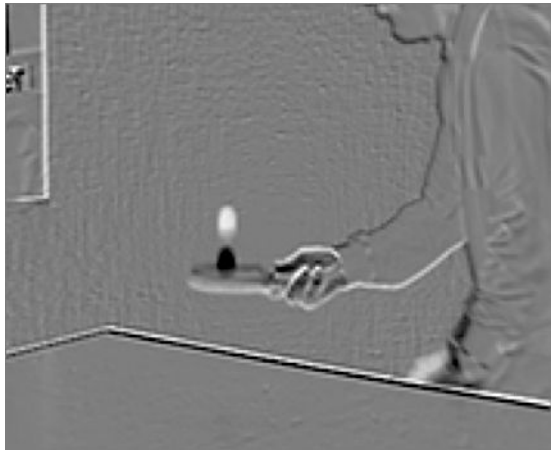
- **Block size (N):** A **compromise** between:
  1. large enough to have **statistically relevant data** and
  2. small **enough to represent part of an object** under translation. Typically 16x16 pixels.
- **Search area (P)**: It is related to the range of **expected movement**: P pixels in every direction: (2P+N)x(2P+N) pixels. Typically P = N.
- **Quantization step:** Related to the **accuracy** of the estimated motion. Typically, 1 pixel but it can go down to 1/4 of pixel (need of upsizing interpolation in this case).

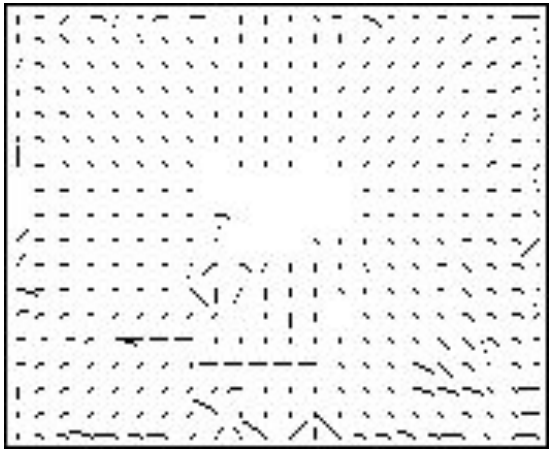


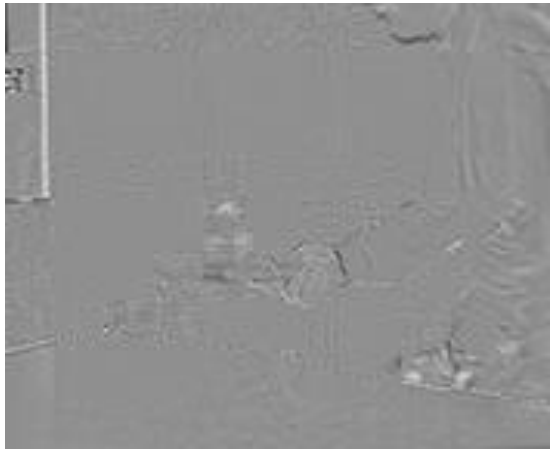


Direct exploration – Block Matching (IV)

Example of **Block Matching** based motion estimation







- **Table-tennis #40**
- **Table-tennis #41**
- **Direct difference**
- **Motion vectors**
- **Displaced Frame Difference**

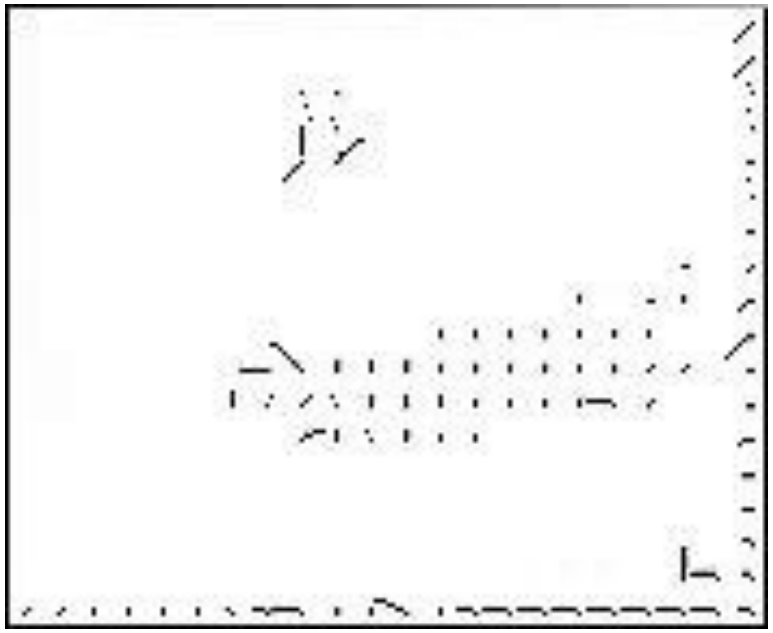
Direct exploration – Block Matching (V)

Example of **Block Matching** based motion estimation

***Table-tennis* sequence**

Direct exploration – Block Matching (VI)

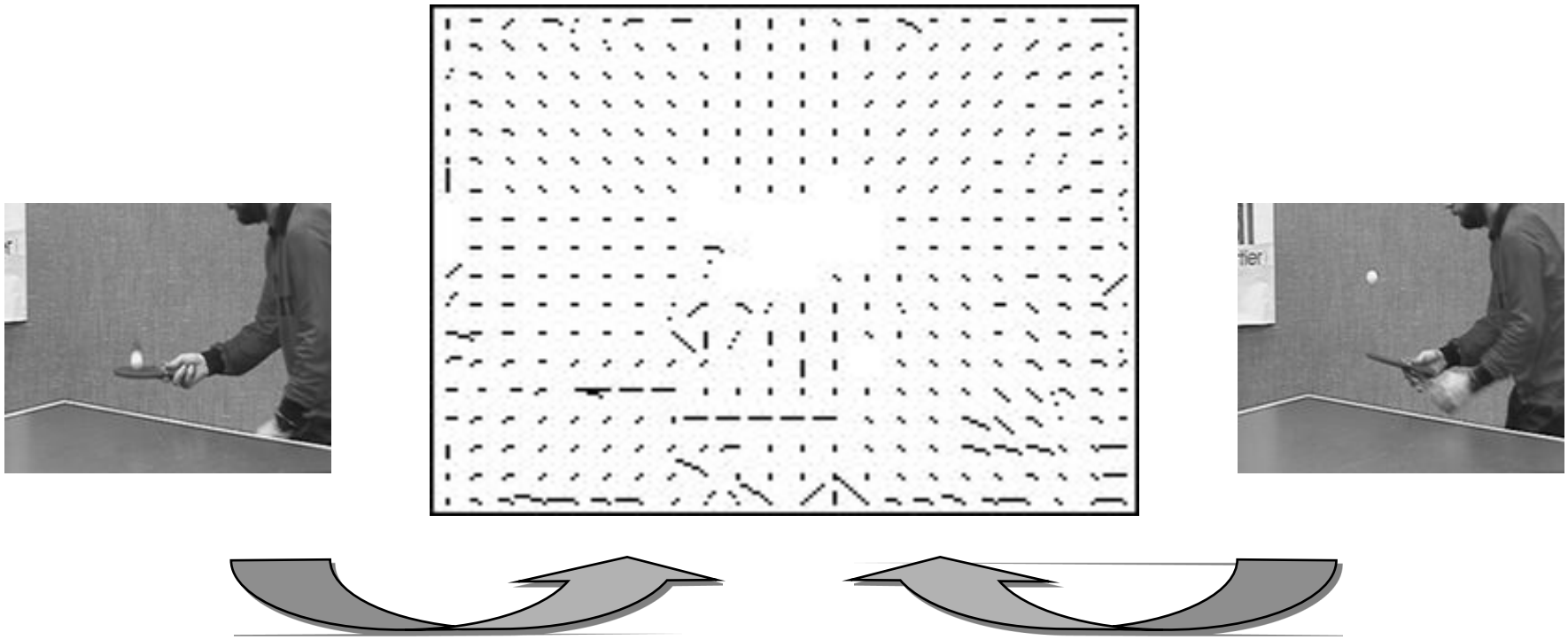
Example of **Block Matching** based motion estimation



Motion vectors of the *Table-tennis* sequence

Direct exploration – Block Matching (VII)

- The motion vectors obtained using the Block Matching algorithm **do not accurately represent the motion of the objects** in the scene.



Direct exploration – Block Matching (VIII)

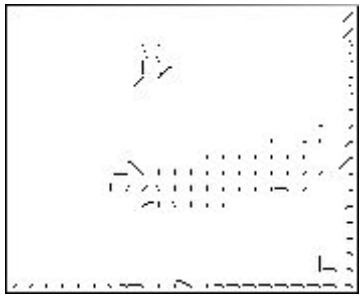
Example of **Block Matching** based motion estimation

Displaced Frame Difference of the *Table-tennis* sequence

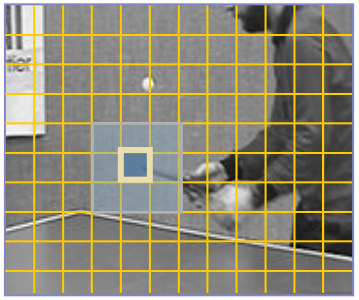
Direct exploration – Block Matching (IX)

Further comments on the BM implementation of the previous example.

- Several blocks have been assigned the **(0, 0) motion vector**.
  - Typically, **the function to be optimized is quite noisy,** presenting multiple, small local minima.
  - Neighbor blocks may present **similar vector values**, but different due to these noisy minima.
  - In coding, solutions around (0, 0) leading to values slightly lower than (0, 0) are usually **substituted by (0, 0)**, which results in a chain of symbols cheaper to code.



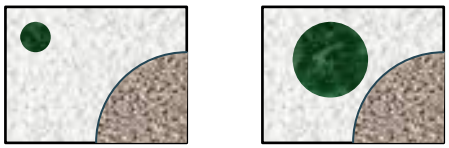
- Vectors have been obtained using an **exhaustive search**.
  - Block size NxN (16x16).
  - Search area (2P+N)x(2P+N) with P = N
  - Possible positions for a given block (2P+1)x(2P+1)
  - Quantization step: 1 pixel.



Total number of subtractions per block using exhaustive search: (2P+1)x(2P+1)xNxN (**278.784**). **Non-optimal techniques are necessary**.

Direct exploration – Block Matching (X)

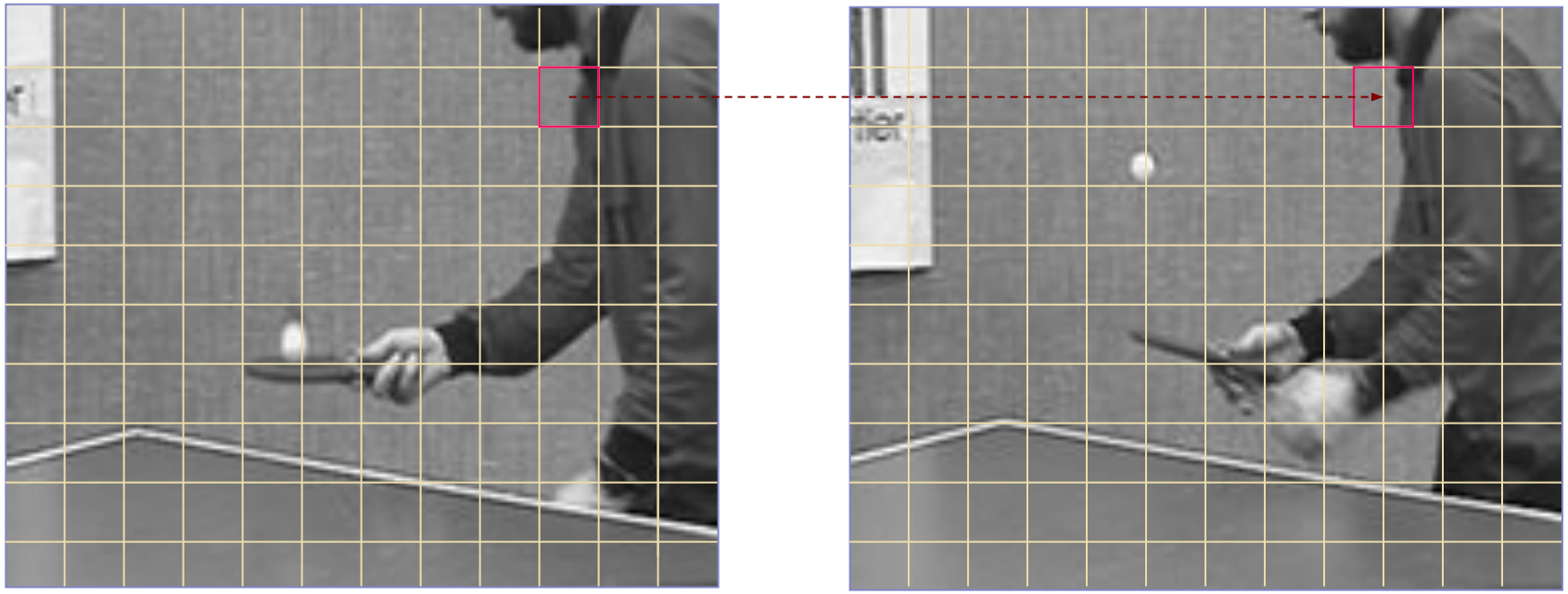
- Vectors representing the motion between two consecutive images can be defined **from the past to the future** (where does this pixel go to?) or **from the future to the past** (where does this pixel come from?).
- **Forward motion estimation**: All pixels in the past image are associated to a pixel in the current image (but the contrary cannot be ensured).
- **Backward motion estimation**: All pixels in the current image are associated to a pixel in the past image (but the contrary cannot be ensured).
  - In the context of coding, the image estimation provided by **the backward motion compensation** has been shown to provide a result with lower error estimation energy.



Motion estimation provides with an injection (but not a bijection) between pixels of the image to be estimated and those of the reference image.

Direct exploration – Forward motion compensation (I)

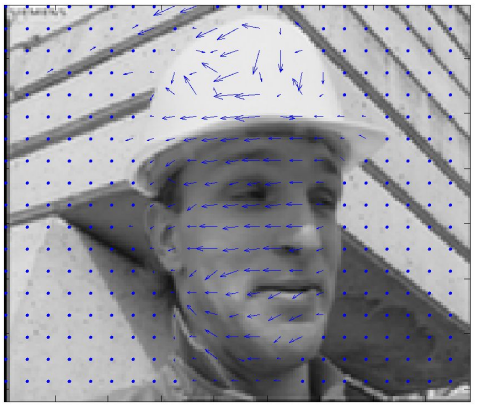
- All pixels in the past image are associated to a pixel in the current image (but the contrary cannot be ensured).
- Some pixels in the current image **may not have a reference** in the previous one.
- **Holes may appear** in a forward motion compensated image



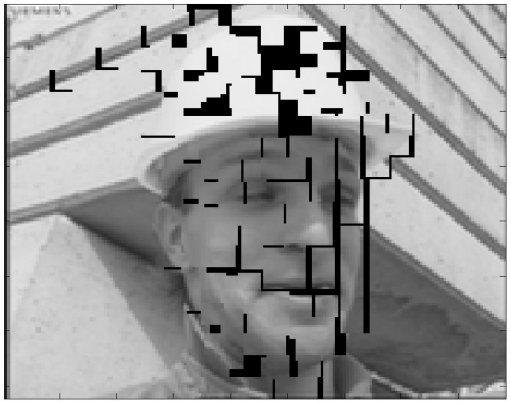
Direct exploration – Forward motion compensation (II)

Frame #(t-1)

Frame #t

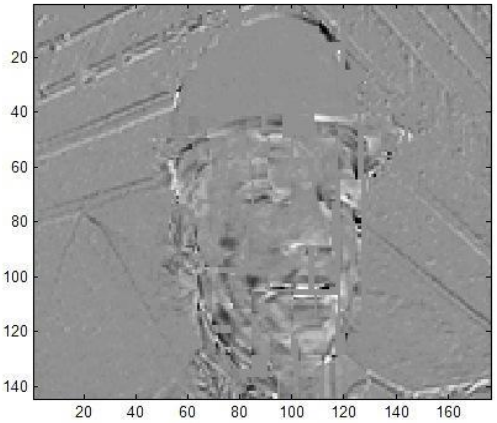


Motion vector field



Forward MC Frame

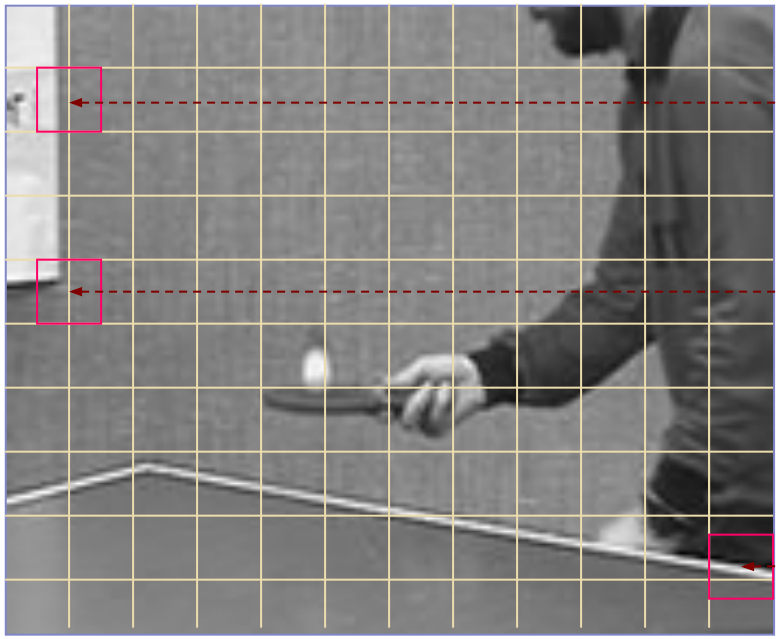
Forward MC frame filling non estimated pixels



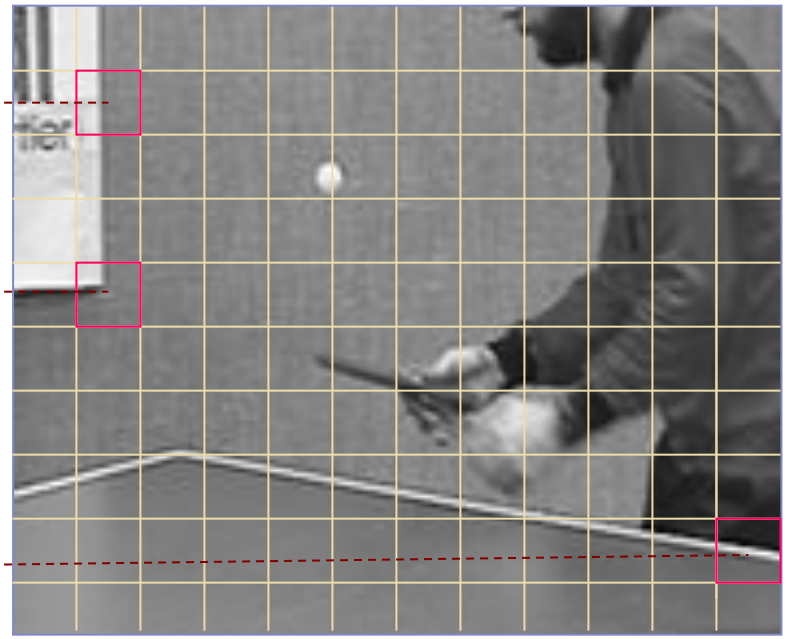
Error prediction filling unconnected pixels

Direct exploration – Backward motion compensation (I)

- All pixels in the current image are associated to a pixel in the past image (but the contrary cannot be ensured).
- All pixels in the current image **have a reference** in the previous one.
- **There are no holes** in a backward motion compensated image



Past image

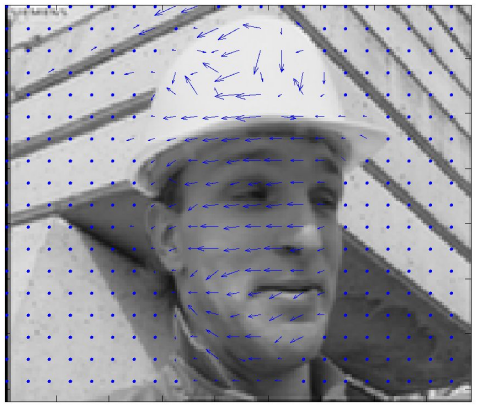


Current image

Direct exploration – Backward motion compensation (II)

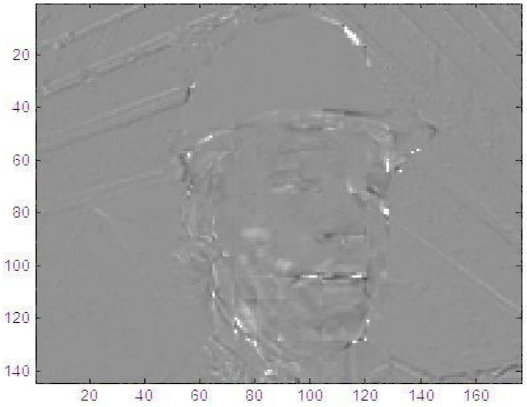
Frame #(t-1)

Frame #t

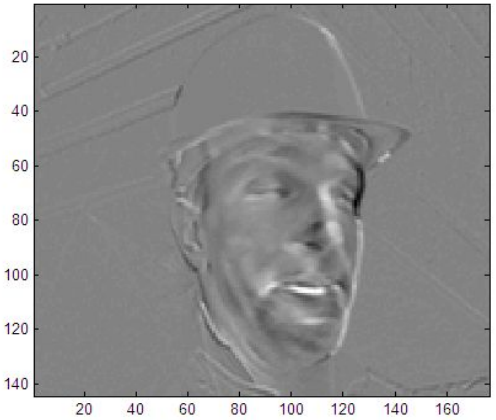


Motion vector field

Motion compensated frame #t



MC Prediction Error Frame #t



Frame difference

Direct exploration – Backward motion compensation (III)

Example of use of the **motion information for image estimation** in video

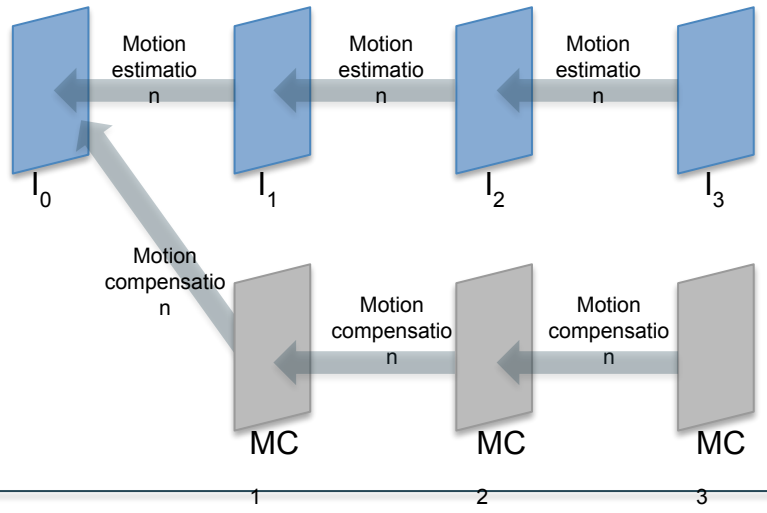
Original sequence *Foreman*

Direct exploration – Backward motion compensation (IV)

- The information in the **previous motion compensated image** is used for image compensation. **What is it happening in this sequence?**

Second example:

- The information in the **previous motion compensated image** is used for compensation.



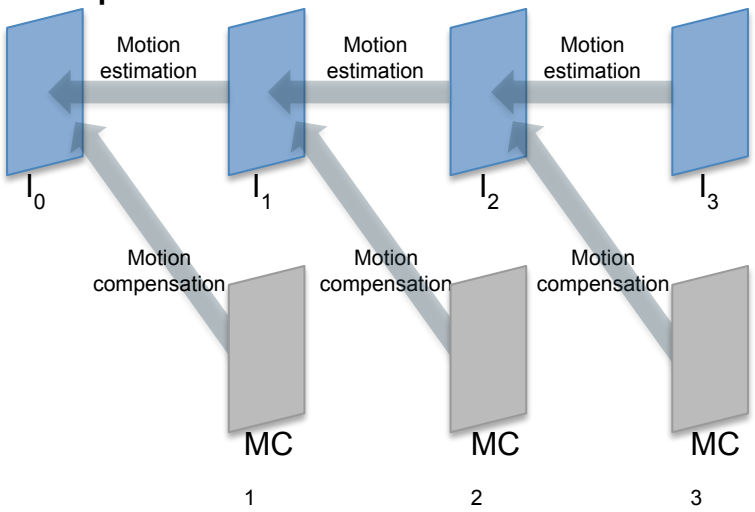
Motion compensated *Foreman* sequence with error propagation

Direct exploration – Backward motion compensation (IV)

- The use of **motion provides a very good estimation** of the image sequence

First example:

- The information in the **previous original image** is used for image compensation.



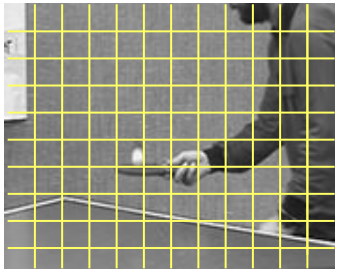
Motion compensated *Foreman* sequence without error propagation

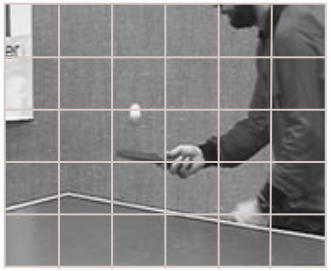
Application to video coding – Exploiting redundancy

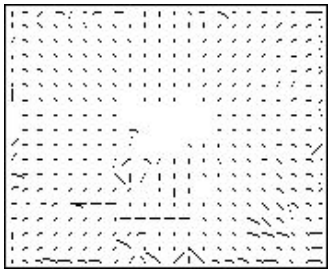
- The most common video coding techniques are based in the so-called **hybrid approach** that combines the exploitation of the spatial and temporal redundancy in video sequences.
- **In the absence of temporal information**, the basic tool for exploiting the **spatial redundancy** is to divide the image into blocks (typically of 8x8 pixels) and to quantize the coefficients of the **Discrete Cosine Transform** of each block.
- **If temporal information is available:**
  - The basic tool for exploiting the **temporal redundancy** is to divide the current image into blocks (typically of 16x16 pixels) and to estimate it by motion compensation of the reference images, using the **Block Matching** approach.
  - The **motion compensation error** (the DFD image) still presents some spatial redundancy that can be exploited by dividing the image into blocks and quantizing the coefficients of the **DCT** of each block.

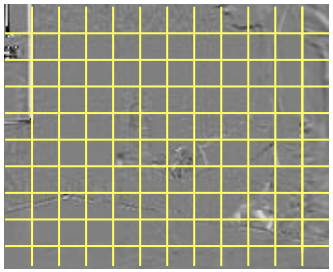
Application to video coding – Information to be transmitted

- The **initial image** in the sequence (or shot) has no reference image and has to be coded using DCT.
  - **Quantized DCT coefficients for each block** are encoded and transmitted
- **Subsequent images** are encoded through a temporal prediction: backward motion estimation & compensation.
  - **Motion vectors for each block** are transmitted
  - Given their importance, **they are not further quantized**.
- The motion compensated image is computed and its associated **motion compensation error** is coded using DCT:
  - **Quantized DCT coefficients for each block** are encoded and transmitted









Outline

- Introduction
- Direct exploration technique:
  - Block matching
- **Gradient-based techniques:**
  - Frame level: Affine model,
  - Dense flow: local approach: Lucas – Kanade
  - Dense flow: global approach with smoothness constraints: Horn-Schunck
- Other techniques:
  - Brox
  - Deep learning methods
- Feature tracking

Global gradient based techniques – Problem statement (I)

- **Criterion**: Look for an (approximated) optical flow that minimizes a **measure of error based on the DFD**.

$$\hat{\vec{D}}(\vec{r}, p_i) = \arg\min_{p_i} \left[\sum_{\vec{r} \in \mathbb{R}} \left\| DFD\left(\vec{r}, \vec{D}(\vec{r}, p_i)\right) \right\| \right]$$

R: the whole image

- In the Block Matching case, **the image was divided into regions** and the minimization was done **region by region**.
- Now, the **whole image** is going to be assumed to be a single region and the **camera motion** is going to be estimated.
  - Several pixels may be removed from that region (**Foreground outliers**).
- Typically, an **affine motion model** (6 parameters) is assumed and a **gradient-based optimization technique** is adopted.

Global gradient based techniques – Iterative approach (I)

- The problem can be solved using an **iterative approach** following a **gradient descent strategy.**
- Gradient descent techniques look for a solution in an iterative way:

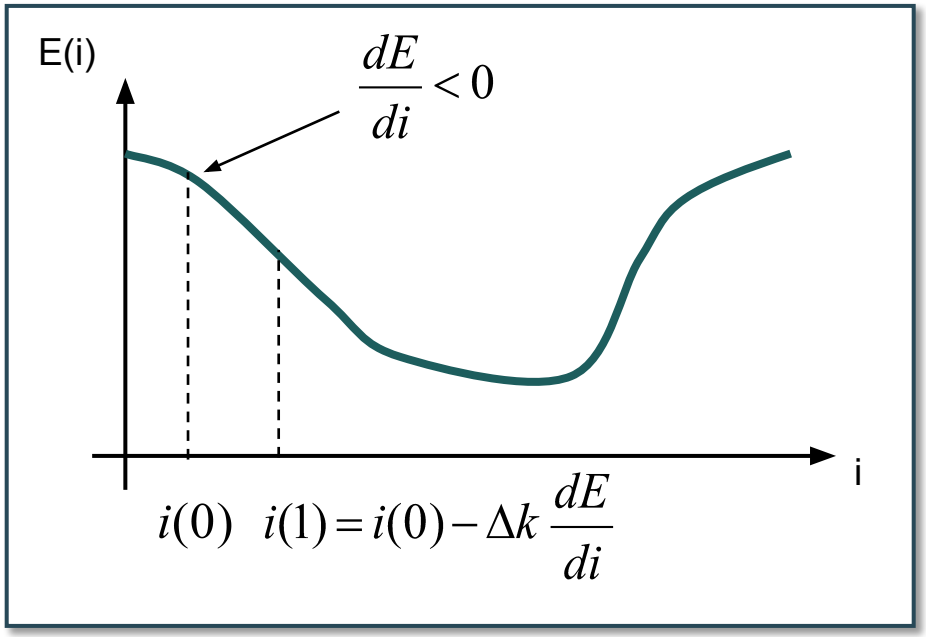
$$\text{Minimum of } E(i) \quad \Rightarrow \quad \frac{dE(i)}{di} = 0$$

$$\frac{\partial i(k)}{\partial k} = -\frac{\partial E(i)}{\partial i}$$

In practice:

$$i(k) - i(k-1) \propto -\frac{\partial E(i)}{\partial i}$$

$$i(k) = i(k-1) - \mu \frac{\partial E(i)}{\partial i}$$



Global gradient based techniques – Iterative approach (II)

- Since the optical flow is fully parameterized by a single set of motion parameters, the **optimization** can be done **over the vector of motion parameters**.

- The problem is defined on a P**-dimensional space**:
  - P **number of parameters** of the model; typically, six.

$$p_i^{k+1} = p_i^k - \mu \frac{\partial}{\partial p_i} \sum_{\vec{r} \in \mathbf{R}} DFD^2(\vec{r}, t, p_i^k)$$

Global gradient based techniques – Iterative approach (III)

- To analyze the problem, we are going to study the derivation with respect to **one of the motion model parameters**.

$$\frac{\partial}{\partial p_i}\sum_{\vec{r}\in\mathbf{R}} DFD^2(\vec{r},t,p_i^k) = 2\sum_{\vec{r}\in\mathbf{R}} DFD(\vec{r},t,p_i^k)\frac{\partial}{\partial p_i}DFD(\vec{r},t,p_i^k)$$

$$\frac{\partial}{\partial p_i}DFD(\vec{r},t,p_i^k) = \frac{\partial}{\partial p_i}\left(I(\vec{r},t) - I(\vec{r}-\vec{D}(\vec{r},p_i^k),t-\Delta t)\right)$$
$$= \frac{\partial}{\partial p_i}\left(I(x,y,t) - I(x-d_x(x,y,p_i^k),y-d_y(x,y,p_i^k),t-\Delta t)\right)$$

$$= \begin{cases} \frac{\partial}{\partial x}I(\vec{r}-\vec{D}(\vec{r},p_i^k),t-\Delta t)\frac{\partial(x-d_x(x,y,p_i^k))}{\partial p_i} \\ \frac{\partial}{\partial y}I(\vec{r}-\vec{D}(\vec{r},p_i^k),t-\Delta t)\frac{\partial(y-d_y(x,y,p_i^k))}{\partial p_i} \end{cases}$$

Gradient of motion compensated image

Global gradient based techniques – Iterative approach (IV)

$$p_i^{k+1} = p_i^k - 2\mu \sum_{\vec{r}\in\mathbf{R}} DFD(\vec{r}, t, p_i^k) \nabla I(\vec{r} - \vec{D}(\vec{r}, p_i^k), t - \Delta t) \begin{pmatrix} \frac{\partial d_x(x,y,p_i^k)}{\partial p_i} \\ \frac{\partial d_y(x,y,p_i^k)}{\partial p_i} \end{pmatrix}$$

Example of computation of the derivative of the motion field:

$$\begin{pmatrix} d_x \\ d_y \end{pmatrix} = \begin{pmatrix} a_1 & a_2 \\ a_4 & a_5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a_3 \\ a_6 \end{pmatrix} \qquad \forall (x,y) \in R$$

$$d_x = a_1 x + a_2 y + a_3 \Rightarrow \frac{\partial d_x(x,y,a_1^k)}{\partial a_1} = x$$

a1= [x, 0]
a2 = [y, 0]
a3 = [1,0]
a4 = [0, x]
a5 = [0,y]
a6 = [0, 1]

Global gradient based techniques – Iterative approach (V)

- The final iterative solution may be compactly expressed as

$$p_i^{k+1} = p_i^k - 2\mu \sum_{\vec{r} \in \mathbf{R}} DFD(\vec{r}, t, p_i^k) \nabla I(\vec{r} - \vec{D}(\vec{r}, p_i^k), t - \Delta t) \frac{\partial}{\partial p_i} \left(\vec{D}(\vec{r}, p_i^k) \right)$$

- Given an initial estimation of the motion model parameters, the **optimal motion model parameters** can be computed by an iterative process that:
  - **Calculates the DFD** using the motion model parameters at iteration k.
  - **Computes the spatial gradient of the motion compensated image**, using the motion model parameters at iteration k.
  - **Computes the derivative** of the optical flow with respect to the motion model parameters at iteration k.

Global gradient based techniques – Application to mosaic creation (I)

- A typical application is the creation of a **mosaic image** from several images in a video, or the stitching of several images to create a panoramic view.

Video sequence obtained with a camera mounted in a small surveillance plane to access remote areas

Global gradient based techniques – Application to mosaic creation (II)

The proposed application requires solving **several problems**:

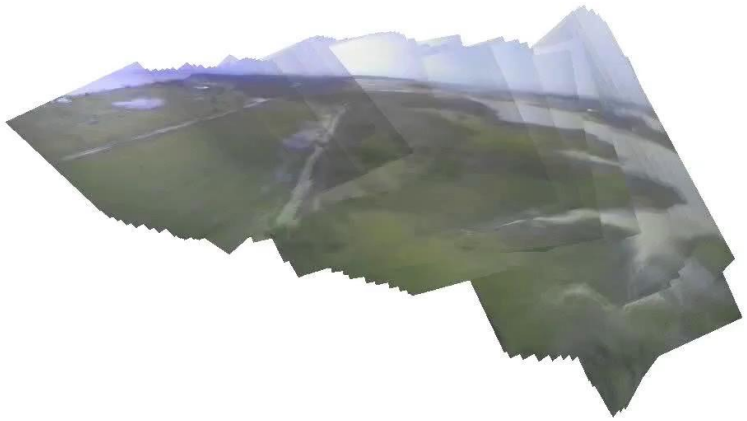
- **Global motion estimation**:
  - Definition of the **motion model**: application dependent.
  - Parts of the image without reference: **searching area**.
  - Presence of **moving objects**: outliers.

- **Warping and blending of the mosaic image**:
  - Position of the selected pixels: **interpolation**.
  - Value of the selected pixels: **filtering**.

- The same technology applies to **super-resolution images**.

Global gradient based techniques – Application to mosaic creation (III)

- The movement of the camera is modeled by **a parametric motion model** (affine in this case) and the parameters between consecutive images are estimated.



Mosaic image created from the video sequence obtained by a camera mounted in a small surveillance plane

Outline

- Introduction
- Direct exploration technique:
  - Block matching
- **Gradient-based techniques:**
  - Frame level: Affine model
  - Dense flow: local approach: Lucas – Kanade
  - Dense flow: global approach with smoothness constraints: Horn-Schunck
- Other techniques:
  - Brox
  - Deep learning methods
- Feature tracking

Local gradient based techniques – Lucas-Kanade (I)

- **Criterion**: Look for an (approximated) optical flow that minimizes a **measure of error based on the DFD**.

$$\hat{\vec{D}}(\vec{r}, p_i) = \arg\min_{p_i} \left[\sum_{\vec{r} \in \mathbf{R}} \left\| DFD\left(\vec{r}, \vec{D}(\vec{r}, p_i)\right) \right\| \right]$$

R: a given region.

- Regions are defined by taking a small window around each pixel.
- Typically, a **translation motion model** (2 parameters) is assumed and a first order (linear) **gradient-based optimization technique** is adopted.
- Some assumptions are made: small motion, brightness constancy and motion coherence inside the regions

Local gradient based techniques – Lucas-Kanade (II)

- Back to the **brightness constancy** equation:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \qquad (1)$$

- We take Taylor expansion of the right expression:

$$I(x+\Delta x, y+\Delta y, t+\Delta t) \approx I(x, y, t) + I_x \cdot \Delta x + I_y \cdot \Delta y + I_t \Delta t \quad (+ \text{H.O.T}) \to 0$$

Assumption: Small displacement!!!

- Comparing with (1), we obtain:

$$I_x \cdot \Delta x + I_y \cdot \Delta y + I_t \Delta t = 0 \quad \Rightarrow \quad I_x \cdot \frac{\Delta x}{\Delta t} + I_y \cdot \frac{\Delta y}{\Delta t} + I_t \frac{\Delta t}{\Delta t} = 0$$

$$I_x \cdot u + I_y \cdot v + I_t = 0$$

$$u = \frac{\Delta x}{\Delta t}, \quad v = \frac{\Delta y}{\Delta t}$$

$$\nabla I \cdot V^T + I_t = 0 \qquad V = [u\ v]$$

Local gradient based techniques – Lucas-Kanade (III)

- **Taylor theorem**: in calculus, **Taylor's theorem** gives an approximation of a *k*-times differentiable function around a given point *a* by a polynomial of degree *k*, called the *k*th-order **Taylor polynomial**[1].
- The first-order Taylor polynomial is the linear approximation of the function
- The first order approximation is valid for a small region around *a*.

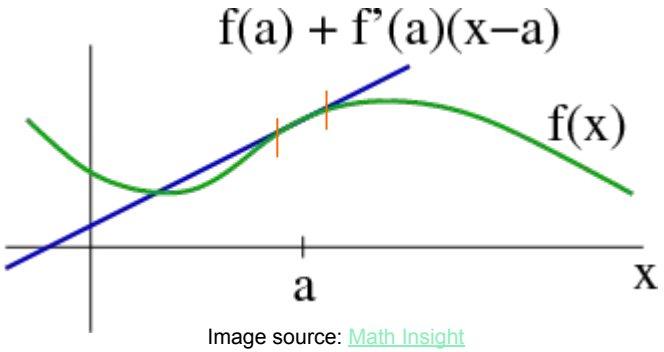
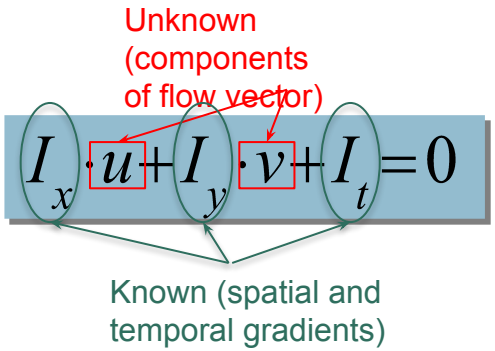


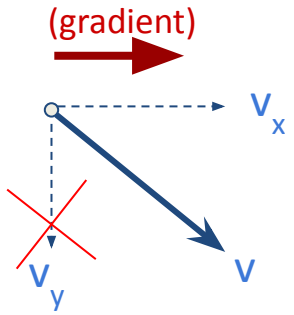
Image source: Math Insight

(1) https://en.wikipedia.org/wiki/Taylor%27s_theorem

Local gradient based techniques – Lucas-Kanade (IV)

Optical flow equation



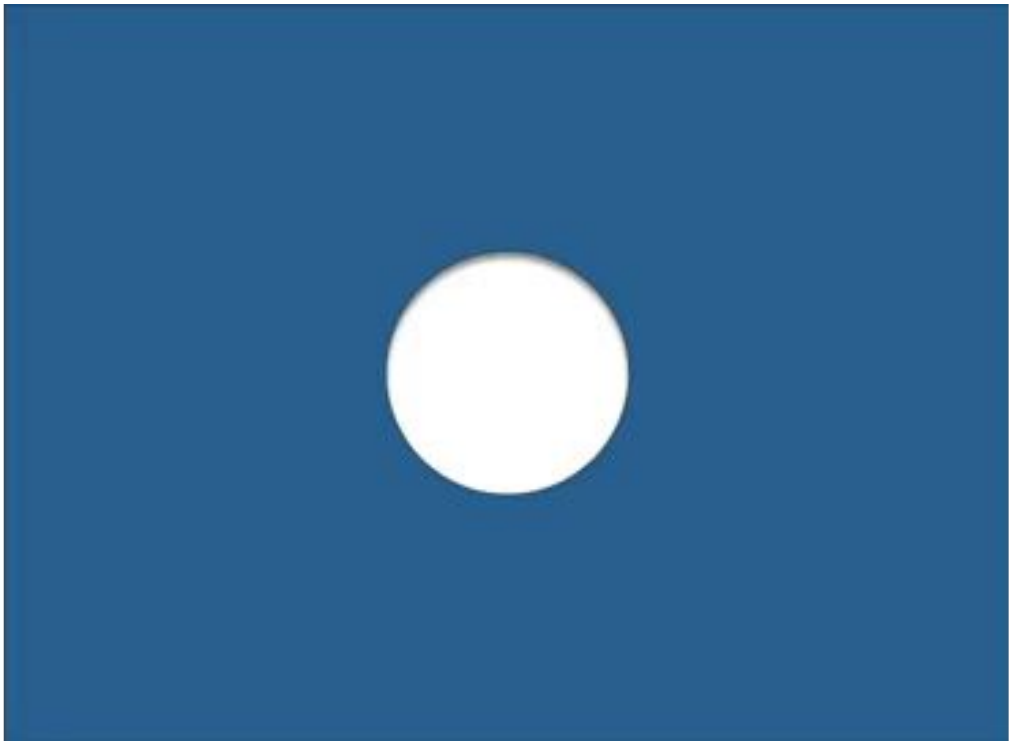


- **One equation, two unknowns!! (u,v)**

- The component of the flow perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

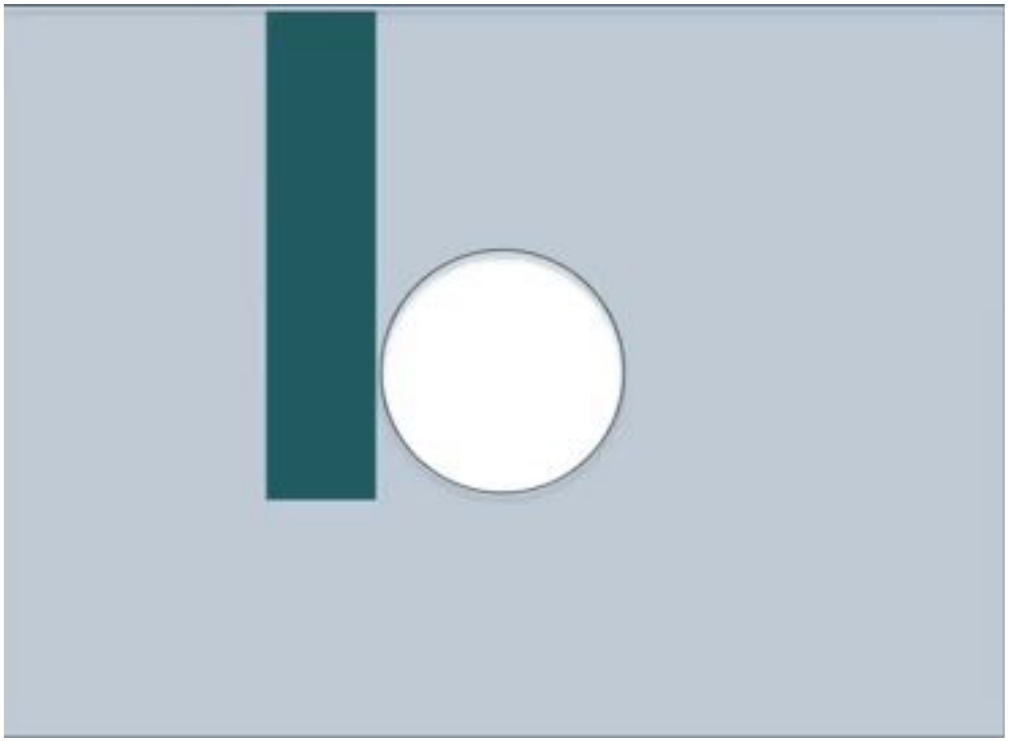
Local gradient based techniques – Lucas-Kanade (V)

Aperture problem



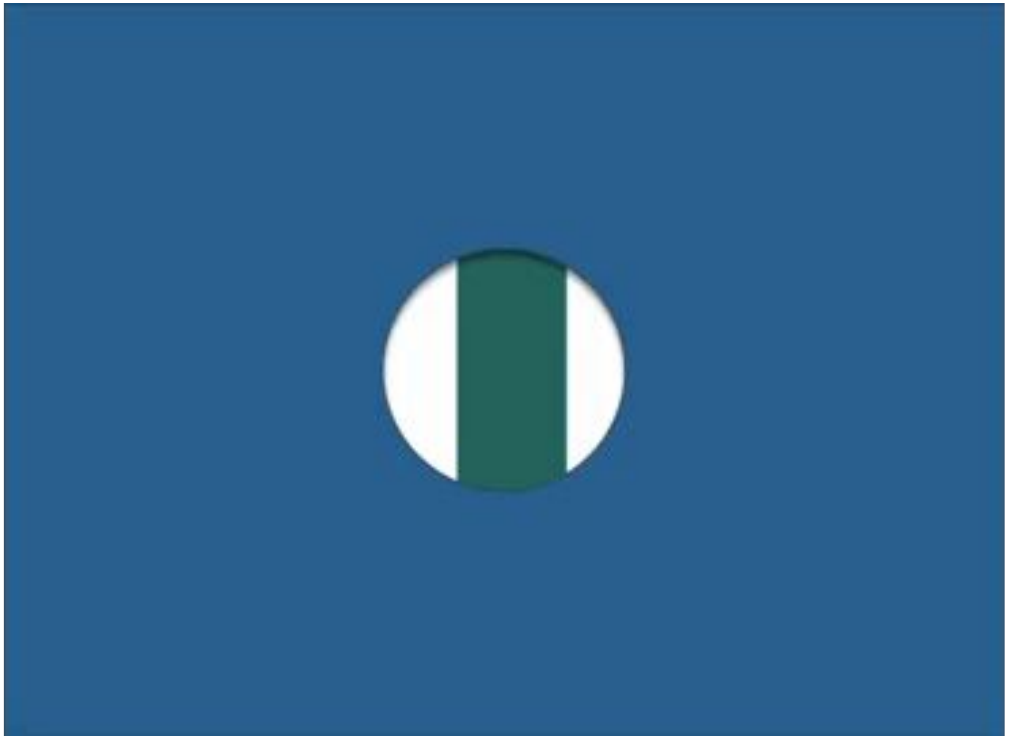
Local gradient based techniques – Lucas-Kanade (VI)

Aperture problem



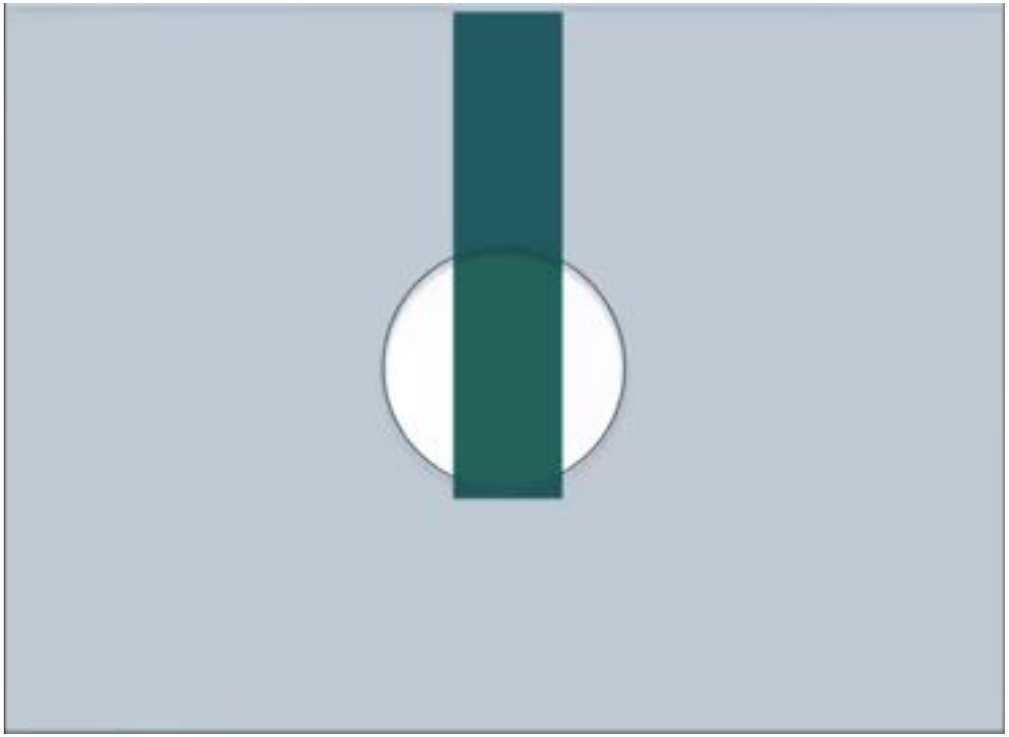
Local gradient based techniques – Lucas-Kanade (VII)

Aperture problem



Local gradient based techniques – Lucas-Kanade (VIII)

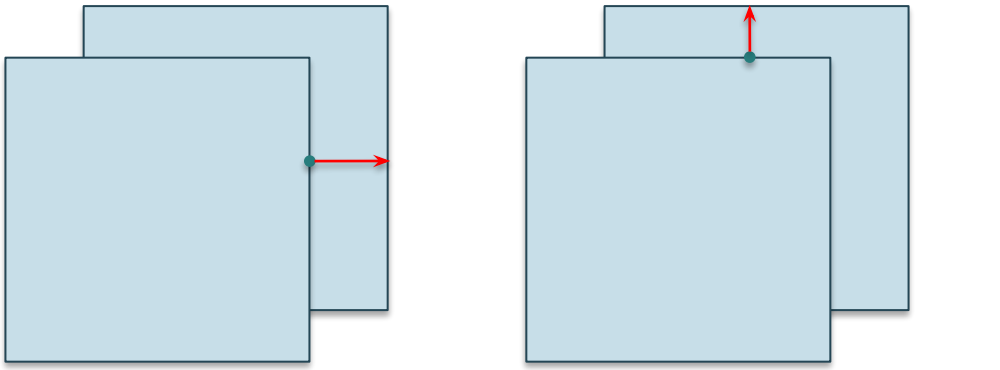
Aperture problem



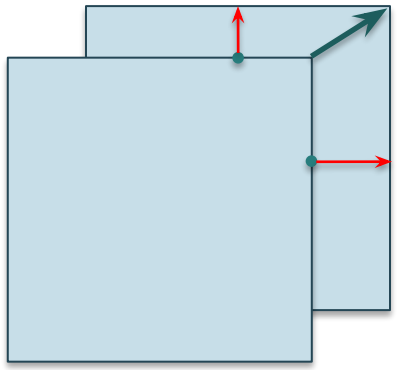
Local gradient based techniques – Lucas-Kanade (IX)

Solving the aperture problem

- Use gradients with more than one direction



Aperture problem with single gradient direction



Using more than one direction Correct solution can be inferred

Local gradient based techniques – Lucas-Kanade (X)

Solving the aperture problem

- We need to add more equations for each pixel
- Use a **spatial coherence constraint**: pretend the pixel's neighbors have the same (u, v) →**Translational model**
  - Using an NxN window around each pixel, we obtain N^2 equations per pixel

$$\nabla I(p_i) \cdot V + I_t(p_i) = 0$$

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{N^2}) & I_y(p_{N^2}) \end{bmatrix} \times \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{N^2}) \end{bmatrix} \qquad A \cdot v = b$$

B. Lucas and T. Kanade. **An iterative image registration technique with an application to stereo vision**. In Proceedings of the International Joint Conference on Artificial Intelligence, pp. 674–679, 1981.

Local gradient based techniques – Lucas-Kanade (XI)

Solving the aperture problem

- This system has more equations than unknowns and thus it is usually over-determined.
- The Lucas-Kanade method obtains a compromise solution by the least squares principle.

$$A^T A \cdot V = A^T \boldsymbol{b} \Rightarrow$$

$$V = (A^T A)^{-1} A^T \boldsymbol{b}$$

Structure tensor!!
(aka Hessian matrix)

$$A^T A = \begin{bmatrix} \sum_i w_i I_x^2 & \sum_i w_i I_x I_y \\ \sum_i w_i I_x I_y & \sum_i w_i I_x^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}$$

Source: http://en.wikipedia.org/wiki/Lucas-Kanade_method

Local gradient based techniques – Lucas-Kanade (XII)

Solving the aperture problem

$$V = (A^T A)^{-1} A^T \boldsymbol{b}$$

- System can be solved when:
  - $(A^T A)$ is invertible
  - $(A^T A)$ is not too small (eigenvalues not too small)
  - $(A^T A)$ is well conditioned ($\frac{\lambda_1}{\lambda_2}$ not too large)

- **Harris corners** provide good points where to compute motion
  - Corners, textured regions ✓
  - Contours ✗
  - Low textured regions ✗

Local gradient based techniques – Lucas-Kanade assumptions

Assumptions

- Brightness constancy
  - Changes in pixel values are **only due to the motion** in the scene
  - Failures due to illumination changes → Illumination compensation
- A pixel value in the image at time $t+\Delta t$ can always be found (and associated to a pixel) in the image at time t
  - Fails due to occlusions, camera motion, objects entering/leaving the scene, etc.
- Small motion:
  - Motion is usually higher than one pixel →Iterative, multiresolution methods
- Motion coherence
  - Pixel's neighbors have the same motion (u, v)

    → More complex motion models (other than translation) can be used

    → Motion segmentation

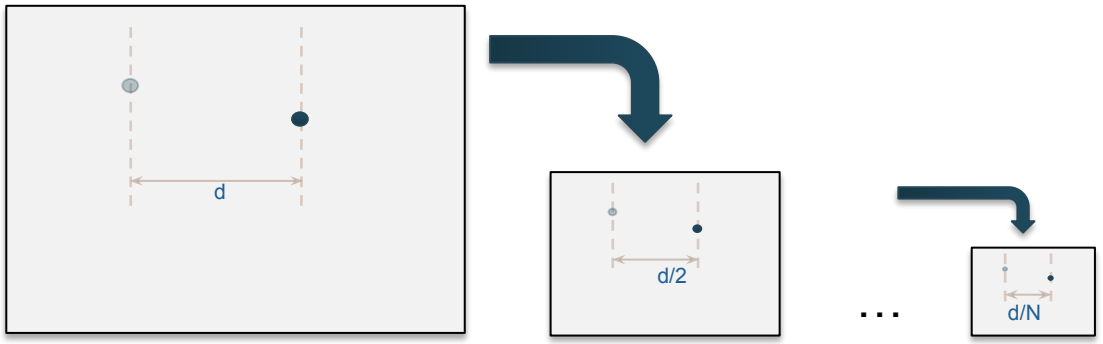
Local gradient based techniques – Lucas-Kanade & large motion (I)

Iterative estimation

1. Initialize (u,v) = (0,0)
2. Compute (u,v) by L-K
3. Motion compensation using (u, v)
4. Repeat steps 2-3 until small change

Local gradient based techniques – Lucas-Kanade & large motion (II)

Multi-resolution approach

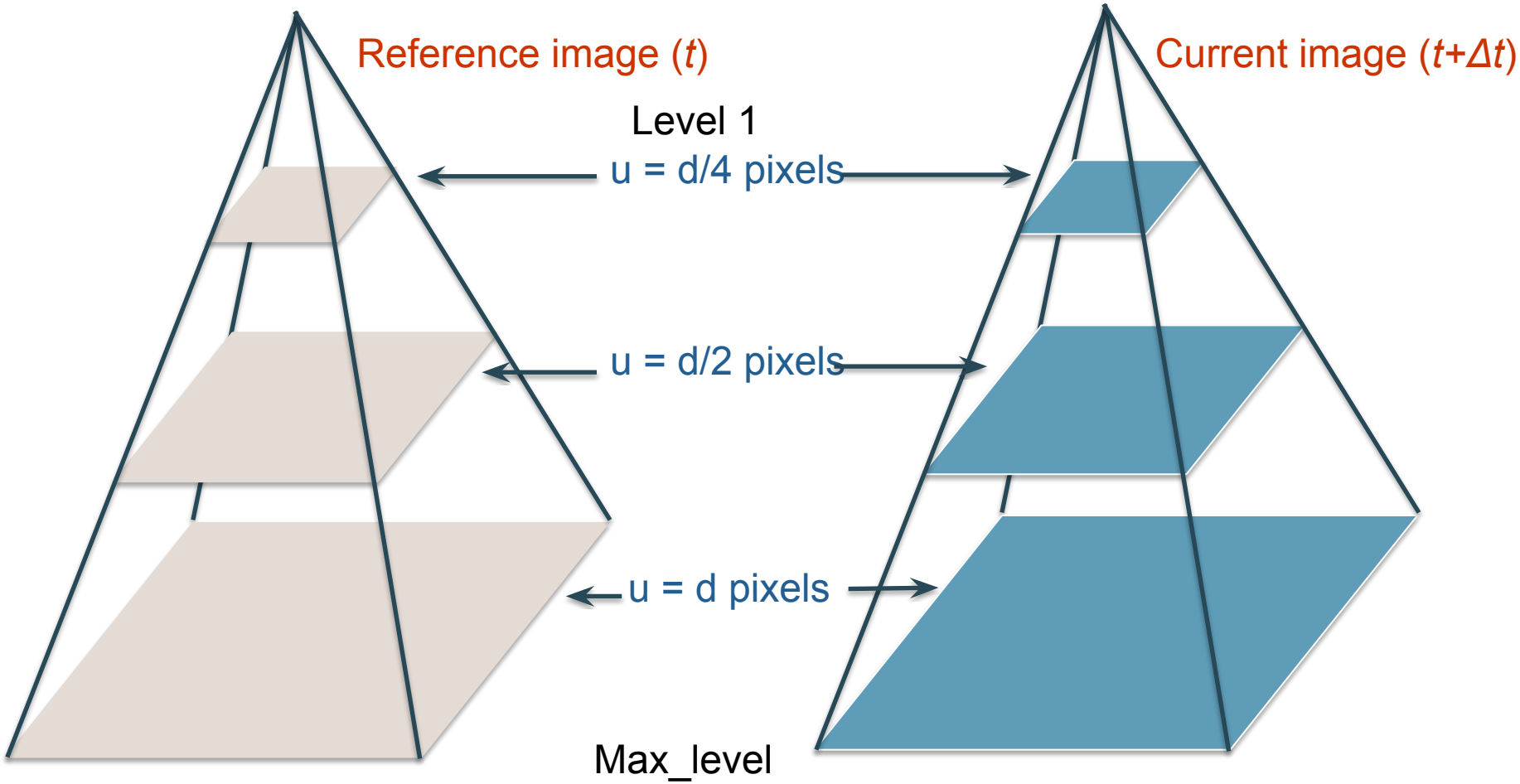


Large motion

Small motion!

Local gradient based techniques – Lucas-Kanade & large motion (III)

Multi-resolution Lucas – Kanade



Local gradient based techniques – Lucas-Kanade & large motion (IV)

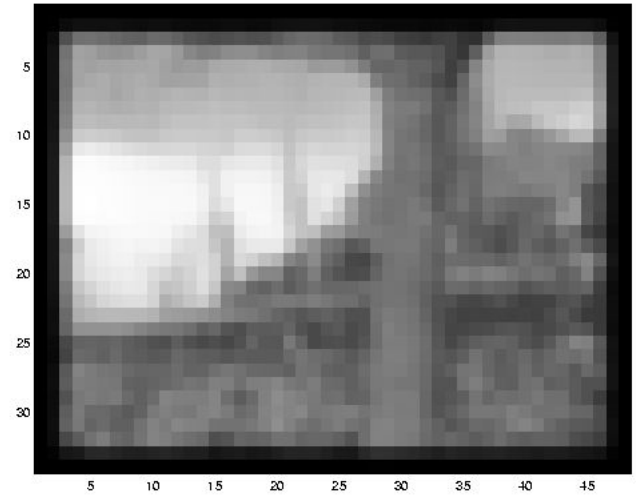
Multi-resolution Lucas – Kanade

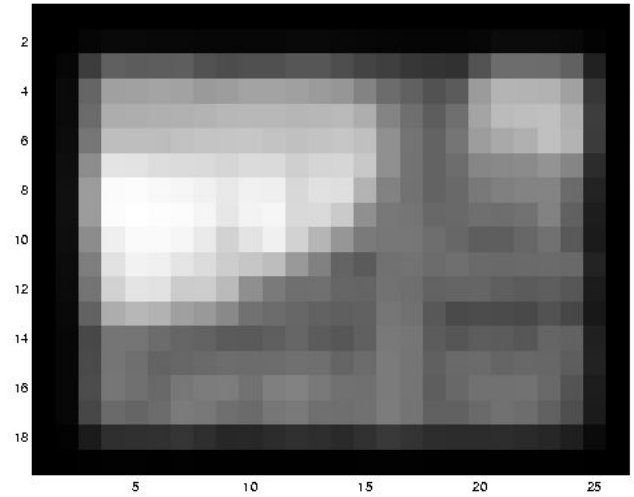
- Compute optical flow using iterative Lucas-Kanade at level **lev** = 1
- For **lev** = 2 to max_level:
  - $(u^*_{lev}, v^*_{lev}) = \text{upsample}\ (u^*_{lev-1}, v^*_{lev-1})$
  - $(u^*_{lev}, v^*_{lev}) = 2 \cdot (u^*_{lev}, v^*_{lev})$
  - Compensate reference using (u^*_{lev}, v^*_{lev})
  - Apply iterative Lucas-Kanade using compensated reference
  - Add the corrections: $u_{lev} = u^*_{lev} + u'_{lev}$

    $v_{lev} = v^*_{lev} + v'_{lev}$

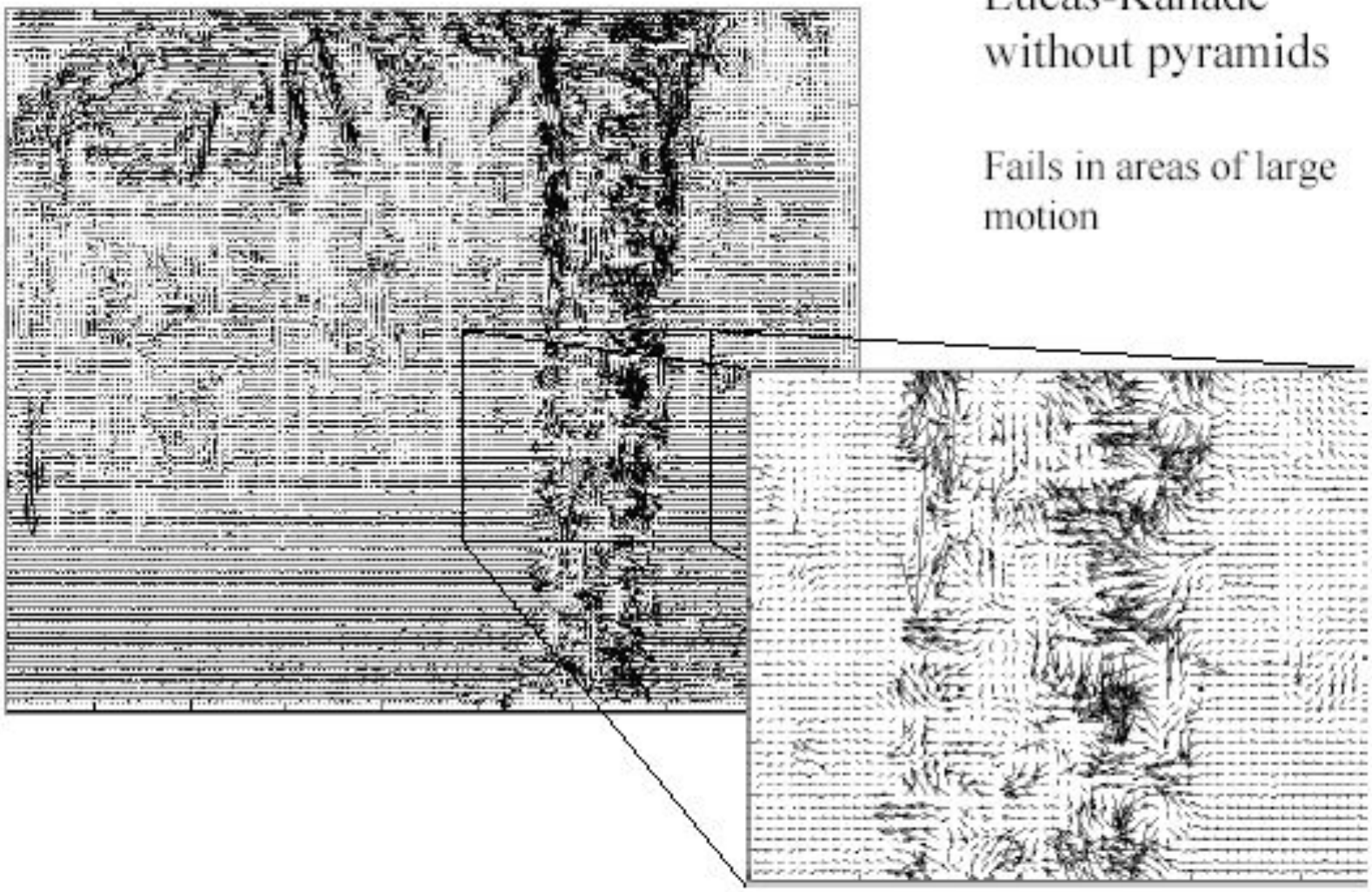
Local gradient based techniques – Lucas-Kanade example (I)

Local gradient based techniques – Lucas-Kanade example (II)

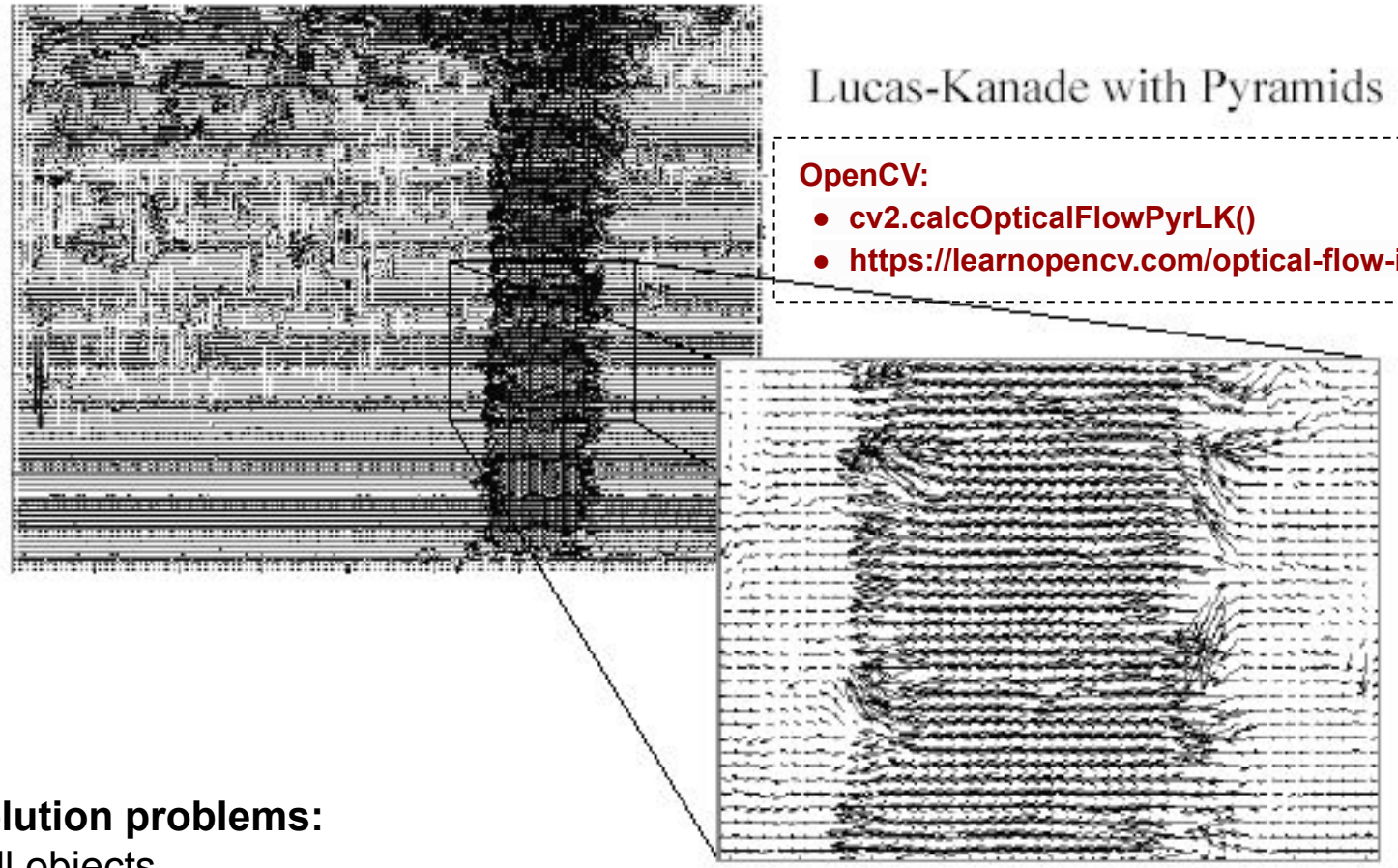




Local gradient based techniques – Lucas-Kanade example (III)



Local gradient based techniques – Lucas-Kanade example (IV)



OpenCV:

- **cv2.calcOpticalFlowPyrLK()**
- **https://learnopencv.com/optical-flow-in-opencv**

Multiresolution problems:

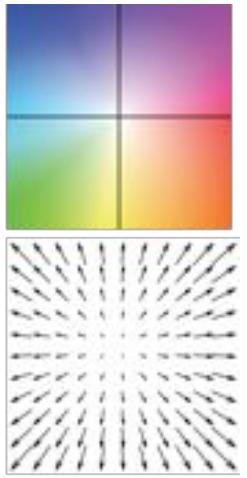
- Small objects
- Errors at initial stages can not be corrected

Optical flow visualization

- Too messy to plot flow vector for every pixel
- Map flow vectors to color
  - Magnitude: saturation
  - Orientation: hue

Input two frames

Ground-truth flow field



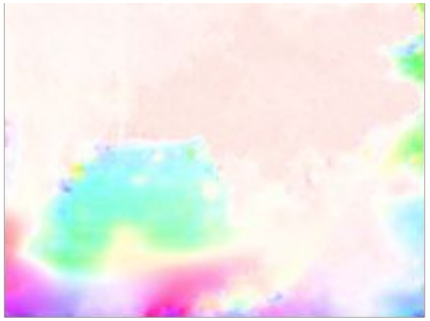
Visualization code
(Baker et al., 2007)

Source: Ce Liu (celiu@microsoft.com)

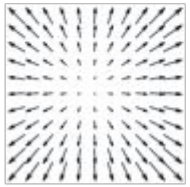
Local gradient based techniques – Lucas-Kanade

- Median filtering can be applied for spatial smoothness

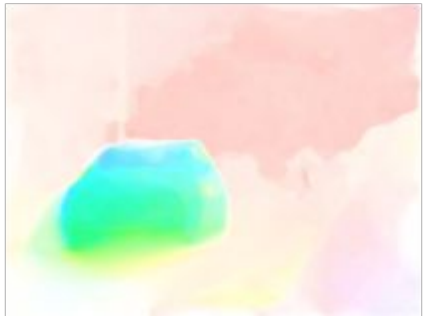
Input two frames



Coarse-to fine LK



Visualization code



Coarse-to fine LK with median filtering

Source: Ce Liu (celiu@microsoft.com)

Outline

- Introduction
- Direct exploration technique:
  - Block matching
- **Gradient-based techniques:**
  - Frame level: Affine model
  - Dense flow: local approach: Lucas – Kanade
  - Dense flow: global approach with smoothness constraints: Horn-Schunck
- Other techniques:
  - Brox
  - Deep learning methods
- Feature tracking

Global gradient based techniques – Horn-Schunck

- Local motion is ambiguous
- L-K fails at lines and flat regions (aperture problem)

Solution → Impose spatial smoothness to the flow field
(adjacent pixels should move together as much as possible)

- Horn-Schunck:

$$E = \iint \underbrace{(I_x u + I_y v + I_t)^2}_{\text{Data term}} + \alpha^2 \underbrace{(|\nabla u|^2 + |\nabla v|^2)}_{\text{Smoothing term}} dx dy$$

(1st order brightness constancy eq.)

$$(u, v) = \arg\min\{E\}$$

Assumes smoothness in the flow over all the image

α : smoothness coefficient

$$|\nabla u|^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 = u_x^2 + u_y^2$$

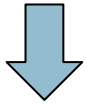
Global gradient based techniques – Horn-Schunck

- How to minimize? → Euler-Lagrange!

$$min\left\{\iint_{\Omega} L(x,y,f,f_x,f_y)dxdy\right\} \quad \Rightarrow \quad \frac{\partial L}{\partial f} - \frac{\partial}{\partial x}\frac{\partial L}{\partial f_x} - \frac{\partial}{\partial y}\frac{\partial L}{\partial f_y} = 0$$

- In Horn-Schunck:

$$L(u,v,u_x,u_y,v_x,v_y) = (I_x u + I_y v + I_t)^2 + \alpha(u_x^2 + u_y^2 + v_x^2 + v_y^2)$$



$$\frac{\partial L}{\partial u} = 2(I_x u + I_u v + I_t)I_x$$

$$\frac{\partial L}{\partial u_x} = 2\alpha u_x, \frac{\partial}{\partial x}\frac{\partial L}{\partial u_x} = 2\alpha u_{xx}, \frac{\partial L}{\partial u_y} = 2\alpha u_y, \frac{\partial}{\partial y}\frac{\partial L}{\partial u_y} = 2\alpha u_{yy}$$

https://en.wikipedia.org/wiki/Euler%E2%80%93Lagrange_equation

Global gradient based techniques – Horn-Schunck

- The Euler-Lagrange equations for Horn-Schunck are:

$$\begin{cases} (I_x u + I_y v + I_t) I_x - \alpha^2 \Delta u = 0 \\ (I_x u + I_y v + I_t) I_y - \alpha^2 \Delta v = 0 \end{cases} \qquad \Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

- In practice, the Laplacian can be approximated as:

$$\Delta u(x,y) = \bar{u}(x,y) - u(x,y)$$

where $\bar{u}$ is a weighted average of u calculated in a neighborhood around the pixel at location (x,y)

- Then, the above equations can be written as:

$$\begin{cases} (I_x^2 + \alpha^2) u + I_x I_y v = \alpha^2 \bar{u} - I_x I_t \\ I_x I_y u + (I_y^2 + \alpha^2) v = \alpha^2 \bar{v} - I_y I_t \end{cases}$$

Source: Wikipedia [https://en.wikipedia.org/wiki/Horn-Schunck_method]

Global gradient based techniques – Horn-Schunck

- These equations are linear in *u* and *v* and may be solved for each pixel in the image

$$\begin{cases} (I_x^2 + \alpha^2)u + I_x I_y v = \alpha^2 \bar{u} - I_x I_t \\ I_x I_y u + (I_y^2 + \alpha^2)v = \alpha^2 \bar{v} - I_y I_t \end{cases}$$

- The solution depends on the neighboring values of the flow field, so it must be repeated once the neighbors have been updated.
- The following iterative scheme is derived:

$$u^{k+1} = \bar{u}^k - \frac{I_x(I_x\bar{u}^k + I_y\bar{v}^k + I_t)}{\alpha^2 + I_x^2 + I_y^2}$$

$$v^{k+1} = \bar{v}^k - \frac{I_y(I_x\bar{u}^k + I_y\bar{v}^k + I_t)}{\alpha^2 + I_x^2 + I_y^2}$$

Global gradient based techniques – Horn-Schunck

bob:
pip install bob.ip.optflow.hornschunck

- Horn-Schunck in short:

$$(u, v) = \arg\min \iint (I_x u + I_y v + I_t)^2 + \alpha^2(|\nabla u|^2 + |\nabla v|^2) dx dy$$

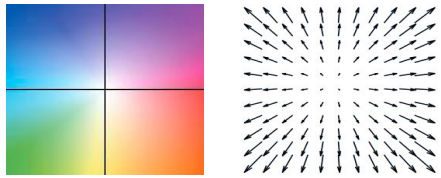
$$u^{k+1} = \bar{u}^k - \frac{I_x(I_x \bar{u}^k + I_y \bar{v}^k + I_t)}{\alpha^2 + I_x^2 + I_y^2}$$

$$v^{k+1} = \bar{v}^k - \frac{I_y(I_x \bar{u}^k + I_y \bar{v}^k + I_t)}{\alpha^2 + I_x^2 + I_y^2}$$

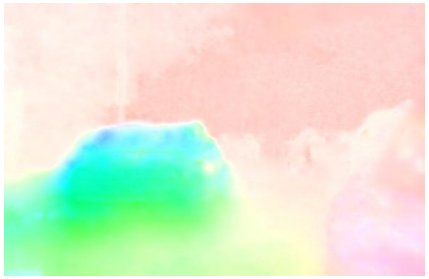
- Strengths:
  - It yields a high density of flow vectors, i.e. the flow information missing in inner parts of homogeneous objects is *filled in* from the motion boundaries.
- Weaknesses:
  - it is more sensitive to noise than local methods.

Global gradient based techniques – Horn-Schunck

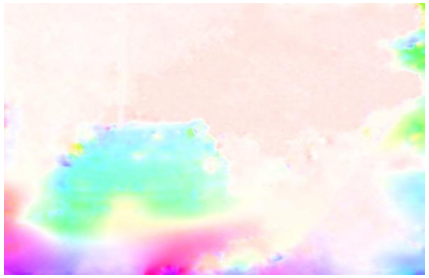
Input two frames



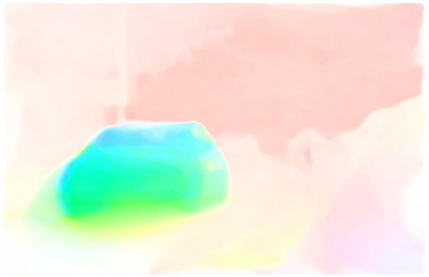
Visualization code



Horn-Schunk



Coarse-to fine LK



Coarse-to fine LK with median filtering

Source: Ce Liu (celiu@microsoft.com)

Global gradient based techniques – Robust methods

- H-S is a Gaussian Markov Random Field (GMRF)

$$\iint (I_x u + I_y v + I_t)^2 + \alpha^2 (|\nabla u|^2 + |\nabla v|^2) dx dy$$

- Quadratic term implies Gaussian white noise
- Nevertheless, the difference between two corresponded pixels is caused by:
  - Noise (majority)
  - Occlusion
  - Compression error
  - Lighting change
  - …

- The error function needs to account for these factors

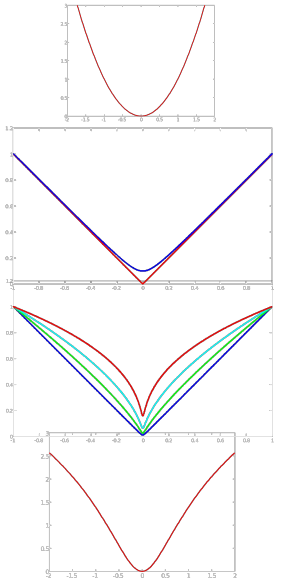
Source: Ce Liu (celiu@microsoft.com)

Global gradient based techniques – Robust methods

- Robust estimation:

$$\iint \psi(|I(x+w) - I(x)|^2) + \alpha^2 \phi(|\nabla u|^2 + |\nabla v|^2) dx dy$$

  - Lucas Kanade: $\alpha = 0,\ \psi(z^2) = z^2$
  - Horn-Schunck: $\alpha > 0,\ \psi(z^2) = z^2,\ \phi(x^2) = z^2$
  - Alternative forms:
    - L1 norm: $\psi(z^2) = \sqrt{z^2 + \epsilon^2}$
    - Sub L1: $\psi(z^2; \eta) = (z^2 + \epsilon^2)^\eta,\ \eta < 0.5$
    - Lorentzian: $\psi(z^2) = log(1 + \gamma z^2)$
    - ...

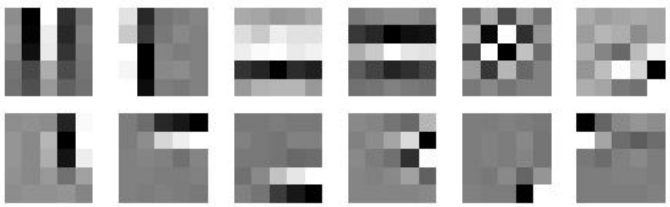


Global gradient based techniques – Robust methods

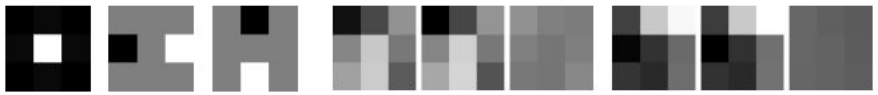
- Robust estimation:

$$\iint \psi(|I(x+w) - I(x)|^2) + \alpha^2 \phi(|\nabla u|^2 + |\nabla v|^2) dx dy$$

- Filters can also be learned (other than gradients) and robust function



[Roth & Black 2005]



[Sun et al. 2010]

Outline

- Introduction
- Direct exploration technique:
  - Block matching
- Gradient-based techniques:
  - Frame level: Affine model
  - Dense flow: local approach: Lucas – Kanade
  - Dense flow: global approach with smoothness constraints: Horn-Schunck
- **Other techniques:**
  - Brox
  - Deep learning methods
- Feature tracking

Other methods – Brox

OpenCV (C++)

cv::cuda::BroxOpticalFlow

- Brox et al:
  - Robust estimation: $\psi(z^2) = \sqrt{z^2 + \epsilon^2}$
  - Non-linearized brightness-constancy assumption
  - Multiresolution

$$E_{Data}(u,v) = \int_\Omega \Psi\left(|I(\mathbf{x}+\mathbf{w}) - I(\mathbf{x})|^2 + \gamma|\nabla I(\mathbf{x}+\mathbf{w}) - \nabla I(\mathbf{x})|^2\right)\mathbf{dx}$$

Non-linearized gray value constancy

Allow for small variations in gray level

$$E_{Smooth}(u,v) = \int_\Omega \Psi\left(|\nabla_3 u|^2 + |\nabla_3 v|^2\right)\mathbf{dx}.$$

$$\nabla_3 := (\partial_x, \partial_y, \partial_t)^\top$$

$$E(u,v) = E_{Data} + \alpha E_{Smooth}$$

Brox, T., Bruhn, A., Papenberg, N., Weickert, J., Pajdla, T., & Matas, J. (2004). High Accuracy Optical Flow Estimation Based on a Theory for Warping. In *European Conference on Computer Vision* (Vol. 3024, pp. 25–36). http://doi.org/10.1007/b97873

Other methods: Matching

Start with something similar to Horh-Schunck

- Gradient constancy
- Energy minimization with smoothing term

+ region matching

+ descriptor matching (long-range)

Large displacement optical flow, Brox et al., CVPR 2009

Slide credit: Derek Hoiem

Other methods: EpicFlow

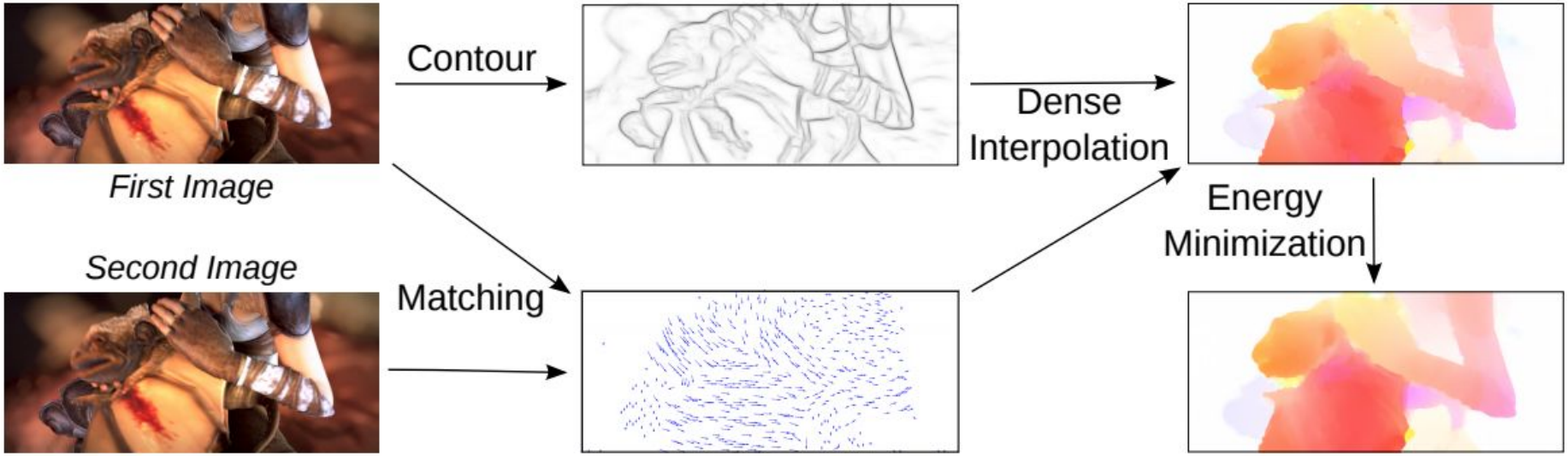


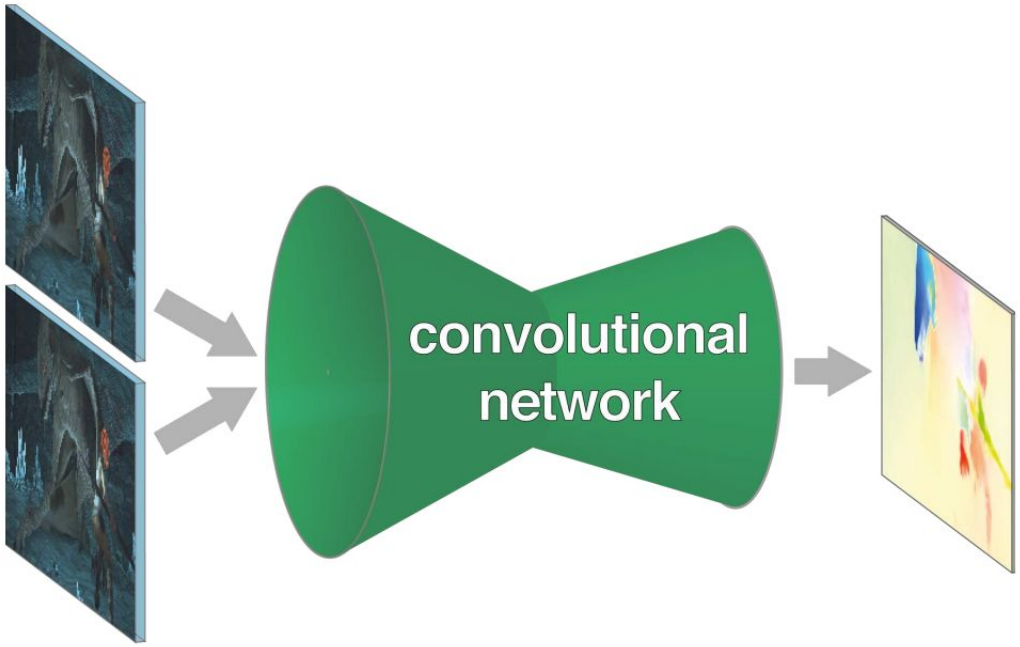
Figure 3. Overview of EpicFlow. Given two images, we compute matches using DeepMatching [34] and the edges of the first image using SED [15]. We combine these two cues to densely interpolate matches and obtain a dense correspondence field. This is used as initialization of a one-level energy minimization framework.

(SED: Structured Edge Detection)

Revaud, J., Weinzaepfel, P., Harchaoui, Z., & Schmid, C. (2015). EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Vol. 7-12 - 2015)

Deep learning techniques: Flownet

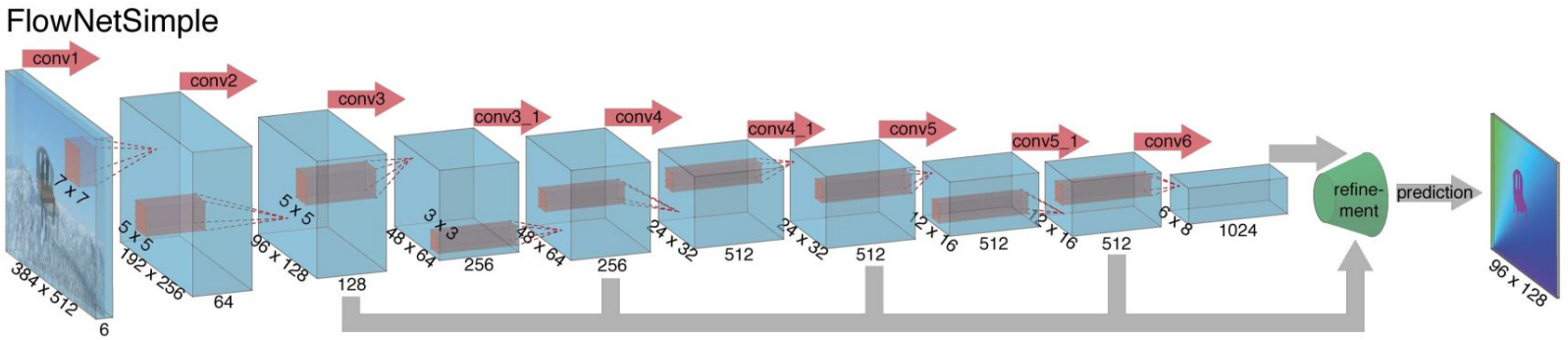
- End-to-end learning of optical flow



Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D. and Brox, T., 2015. FlowNet: Learning Optical Flow With Convolutional Networks. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2758-2766).

Deep learning techniques: Flownet

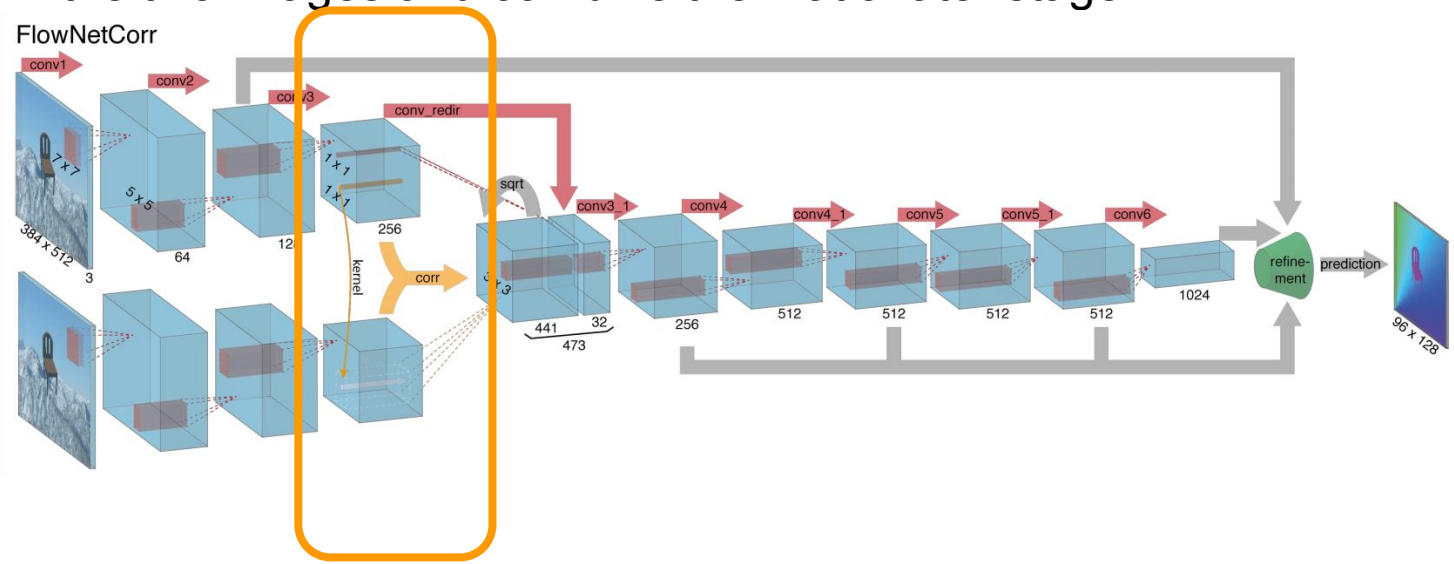
- Contracting:
  - **Option A**: Stack both input images together and feed them through a generic network.



Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D. and Brox, T., 2015. FlowNet: Learning Optical Flow With Convolutional Networks. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2758-2766).

Deep learning techniques: Flownet

- Contracting:
  - **Option B**: Create two separate, yet identical processing streams for the two images and combine them at a later stage.

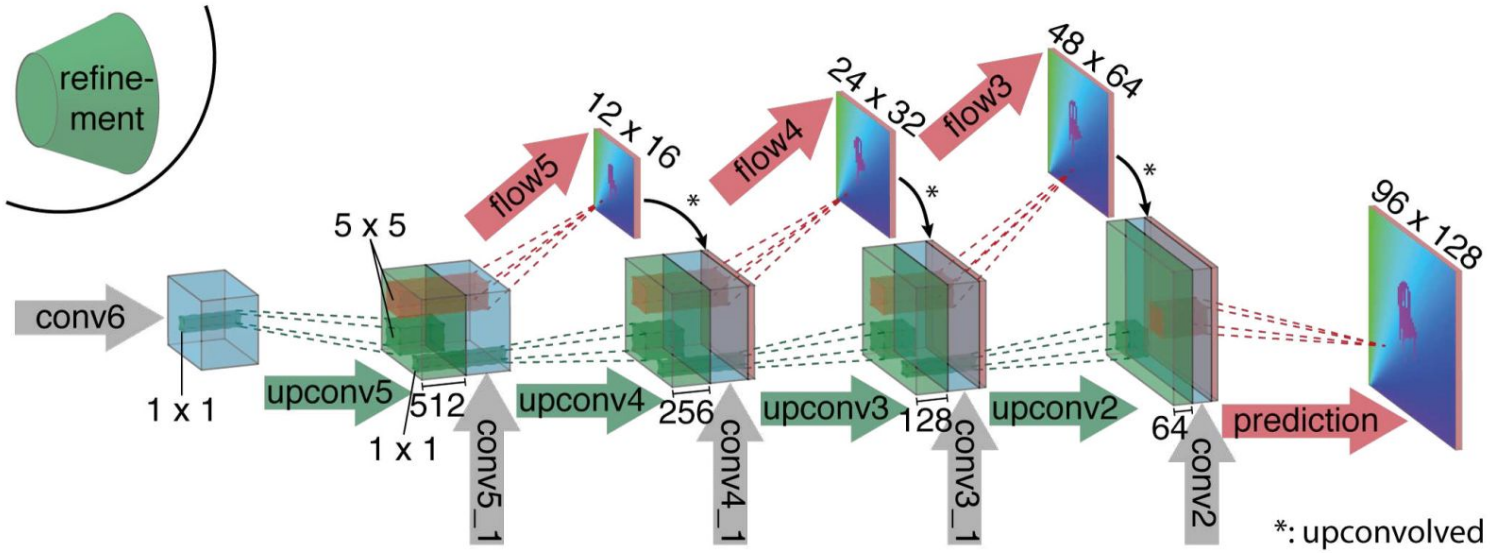


Correlation layer:
Convolution of data patches from the layers to combine.

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D. and Brox, T., 2015. FlowNet: Learning Optical Flow With Convolutional Networks. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2758-2766).

Deep learning techniques: Flownet

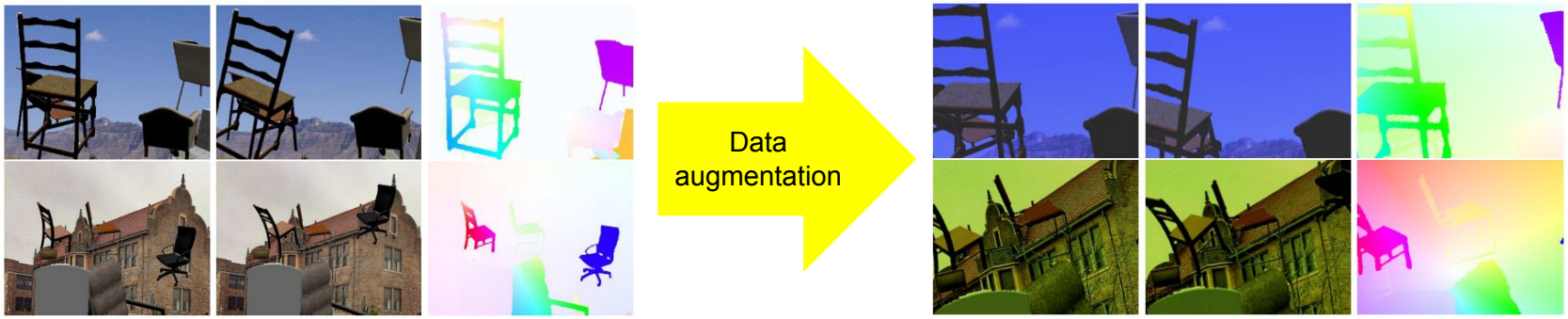
- Expanding:
  - Upconvolutional layers: Unpooling features maps + convolution.
  - Upconvolutioned feature maps are concatenated with the corresponding map from the contractive part.



Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D. and Brox, T., 2015. FlowNet: Learning Optical Flow With Convolutional Networks. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2758-2766).

Deep learning techniques: Flownet

- Data augmentation:
  - Ground truth datasets are scarce to train a ConvNet
  - Synthetic dataset (“Flying chairs”) is generated and augmented (transl., rot., scal., noise, brightness, contrast, gamma and color)



- Convnets trained on these unrealistic data generalize well to existing datasets such as Sintel and KITTI.

Deep learning techniques: Flownet 2

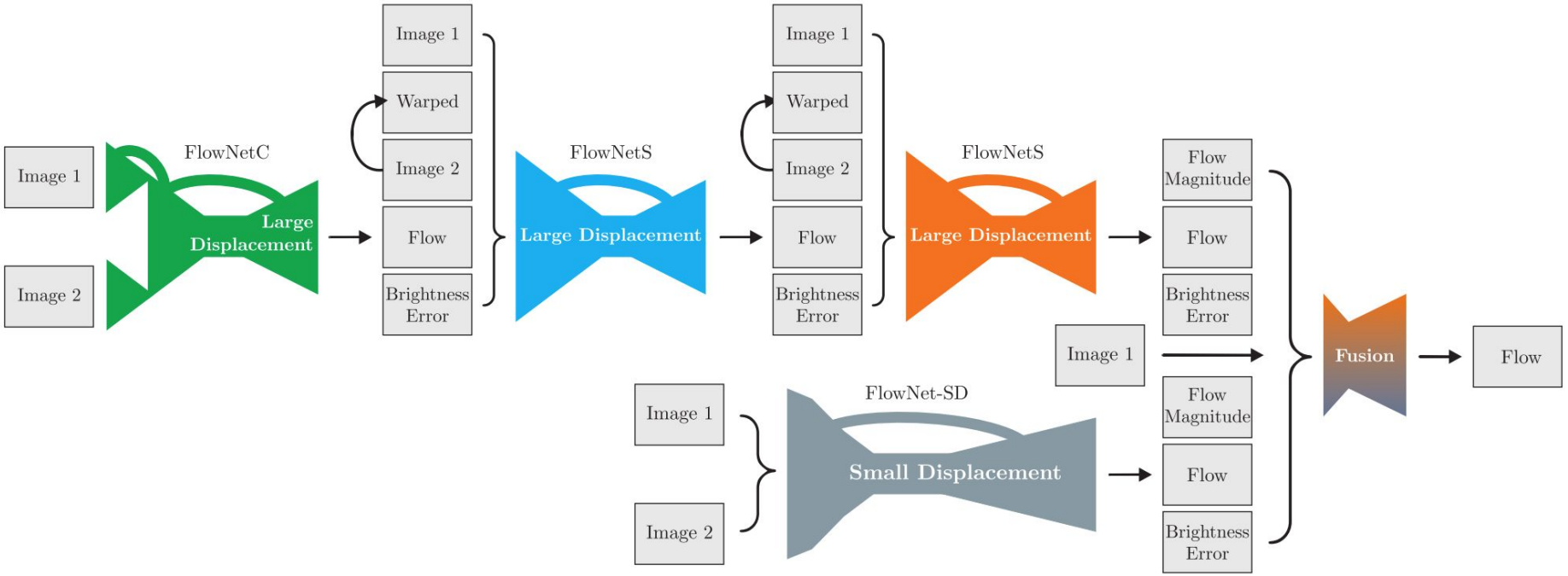


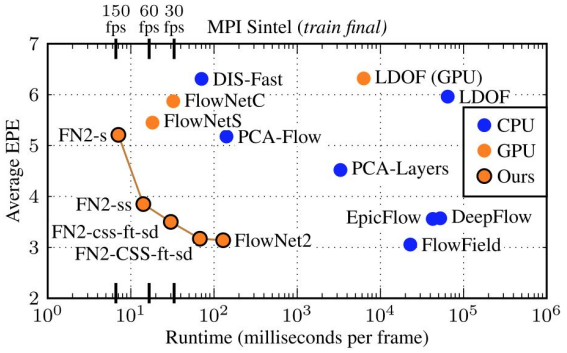
Figure 2. Schematic view of complete architecture: To compute large displacement optical flow we combine multiple FlowNets. Braces indicate concatenation of inputs. *Brightness Error* is the difference between the first image and the second image warped with the previously estimated flow. To optimally deal with small displacements, we introduce smaller strides in the beginning and convolutions between upconvolutions into the FlowNetS architecture. Finally we apply a small fusion network to provide the final estimate.

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., & Brox, T. (2016). FlowNet 2.0 : Evolution of Optical Flow Estimation with Deep Networks. *Cvpr2017*. Retrieved from https://github.com/lmb-freiburg/flownet2

Deep learning techniques: Flownet 2

	Method	Sintel *clean* AEE *train*	Sintel *clean* AEE *test*	Sintel *final* AEE *train*	Sintel *final* AEE *test*	KITTI 2012 AEE *train*	KITTI 2012 AEE *test*	KITTI 2015 AEE *train*	KITTI 2015 Fl-all *train*	KITTI 2015 Fl-all *test*	Middlebury AEE *train*	Middlebury AEE *test*	Runtime ms per frame CPU	Runtime ms per frame GPU
Accurate	EpicFlow† [22]	2.27	4.12	3.56	6.29	**3.09**	**3.8**	9.27	27.18%	**27.10%**	0.31	0.39	42,600	–
	DeepFlow† [32]	2.66	5.38	3.57	7.21	4.48	5.8	10.63	26.52%	29.18%	**0.25**	0.42	51,940	–
	FlowFields [2]	**1.86**	**3.75**	**3.06**	**5.81**	3.33	3.5	**8.33**	**24.43%**	–	0.27	**0.33**	22,810	–
	LDOF (CPU) [7]	4.64	7.56	5.96	9.12	10.94	12.4	18.19	38.11%	–	0.44	0.56	64,900	–
	LDOF (GPU) [27]	4.76	–	6.32	–	10.43	–	18.20	38.05%	–	0.36	–	–	6,270
	PCA-Layers [33]	3.22	5.73	4.52	7.89	5.99	5.2	12.74	27.26%	–	0.66	–	**3,300**	–
Fast	EPPM [4]	–	**6.49**	–	**8.38**	–	9.2	–	–	–	–	**0.33**	–	200
	PCA-Flow [33]	**4.04**	6.83	**5.18**	8.65	**5.48**	**6.2**	**14.01**	**39.59%**	–	**0.70**	–	140	–
	DIS-Fast [16]	5.61	9.35	6.31	10.13	11.01	14.4	21.20	53.73%	–	0.92	–	70	–
	FlowNetS [11]	4.50	6.96‡	5.45	7.52‡	8.26	–	–	–	–	1.09	–	–	**18**
	FlowNetC [11]	4.31	6.85‡	5.87	8.51‡	9.35	–	–	–	–	1.15	–	–	32
FlowNet 2.0	FlowNet2-s	4.55	–	5.21	–	8.89	–	16.42	56.81%	–	1.27	–	–	**7**
	FlowNet2-ss	3.22	–	3.85	–	5.45	–	12.84	41.03%	–	0.68	–	–	14
	FlowNet2-css	2.51	–	3.54	–	4.49	–	11.01	35.19%	–	0.54	–	–	31
	FlowNet2-css-ft-sd	2.50	–	3.50	–	4.71	–	11.18	34.10%	–	0.43	–	–	31
	FlowNet2-CSS	2.10	–	3.23	–	**3.55**	–	**8.94**	29.77%	–	0.44	–	–	69
	FlowNet2-CSS-ft-sd	2.08	–	3.17	–	4.05	–	10.07	30.73%	–	0.38	–	–	69
	FlowNet2	**2.02**	**3.96**	**3.14**	6.02	4.09	–	10.06	30.37%	–	**0.35**	**0.52**	–	123
	FlowNet2-ft-sintel	(1.45)	4.16	(2.01)	**5.74**	3.61	–	9.84	**28.20%**	–	0.35	–	–	123
	FlowNet2-ft-kitti	3.43	–	4.66	–	(1.28)	**1.8**	(2.30)	(8.61%)	**11.48%**	0.56	–	–	123

Table 4. Performance comparison on public benchmarks. AEE: Average Endpoint Error; Fl-all: Ratio of pixels where flow estimate is wrong by both ≥ 3 pixels and $\geq 5\%$. The best number for each category is highlighted in bold. See text for details. †*train* numbers for these methods use slower but better "improved" option. ‡For these results we report the fine-tuned numbers (FlowNetS-ft and FlowNetC-ft).



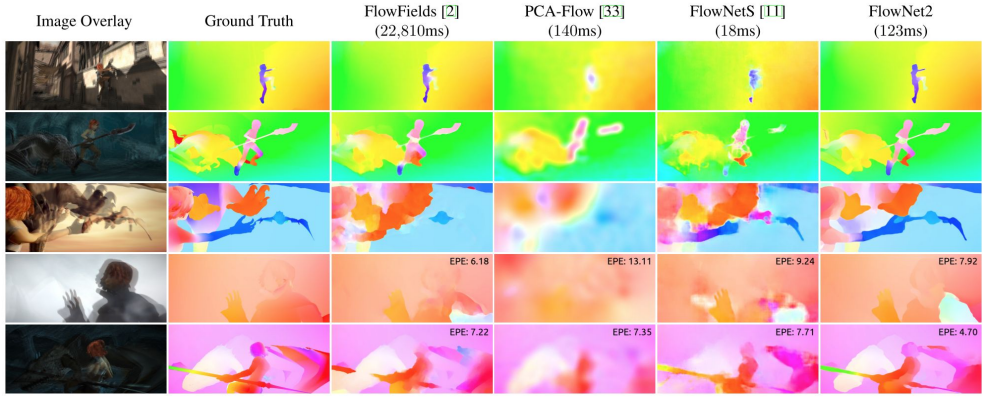


Figure 6. Examples of flow fields from different methods estimated on Sintel. FlowNet2 performs similar to FlowFields and is able to extract fine details, while methods running at comparable speeds perform much worse (PCA-Flow and FlowNetS).

Deep learning techniques: RAFT

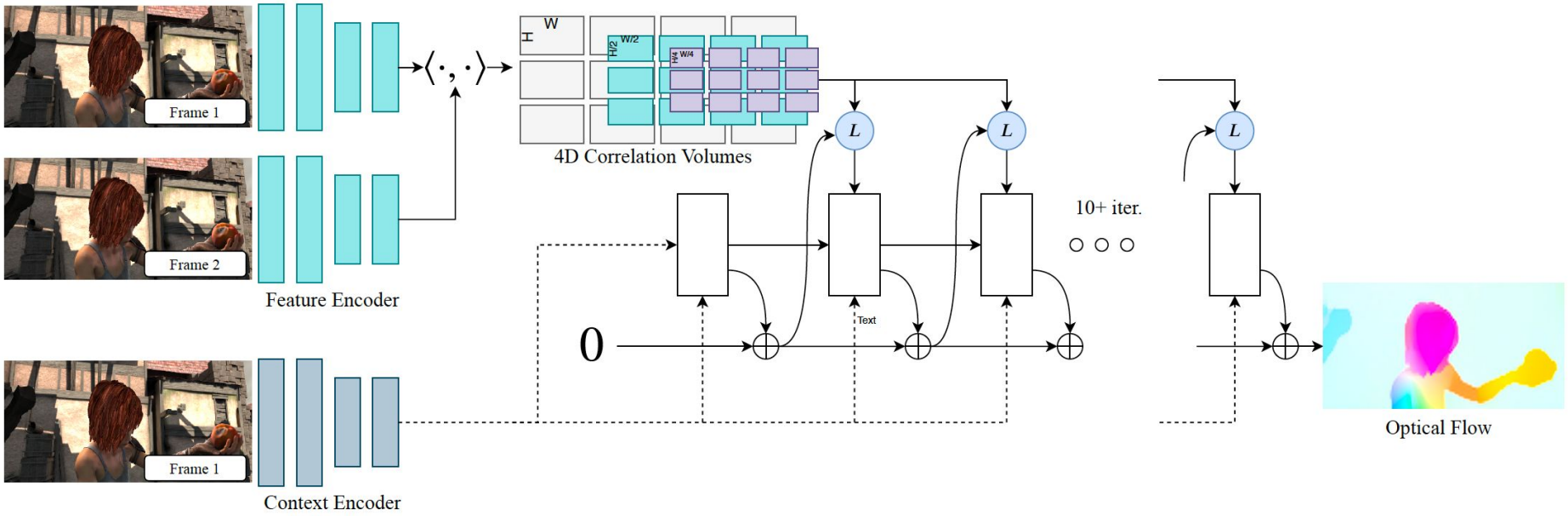


Fig. 1: RAFT consists of 3 main components: (1) A feature encoder that extracts per-pixel features from both input images, along with a context encoder that extracts features from only I_1. (2) A correlation layer which constructs a 4D $W \times H \times W \times H$ correlation volume by taking the inner product of all pairs of feature vectors. The last 2-dimensions of the 4D volume are pooled at multiple scales to construct a set of multi-scale volumes. (3) An *update operator* which recurrently updates optical flow by using the current estimate to look up values from the set of correlation volumes.

Zachary Teed and Jia Deng, **RAFT: Recurrent All-Pairs Field Transforms for Optical Flow,** ECCV 2020

Deep learning techniques: RAFT

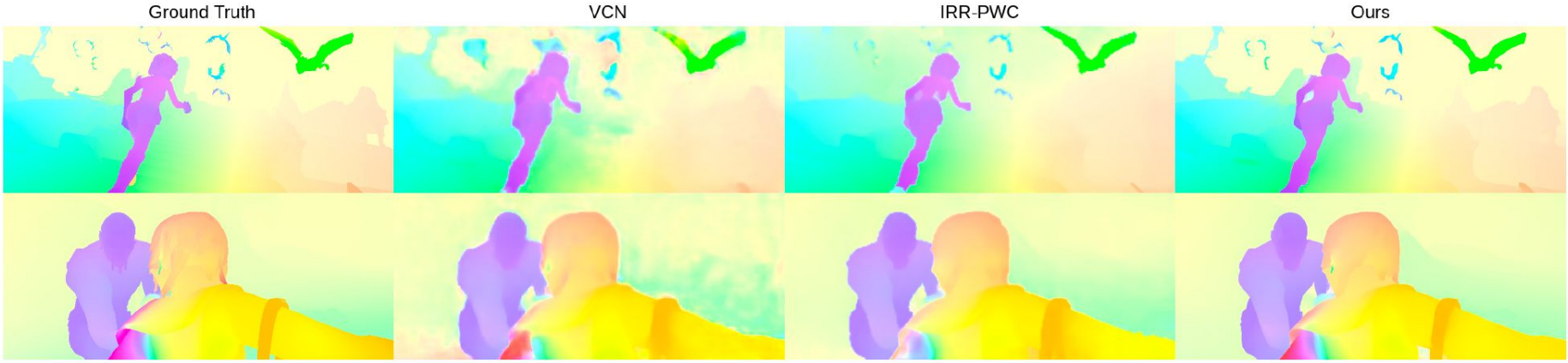


Fig. 3: Flow predictions on the Sintel test set.

Zachary Teed and Jia Deng, **RAFT: Recurrent All-Pairs Field Transforms for Optical Flow,** ECCV 2020

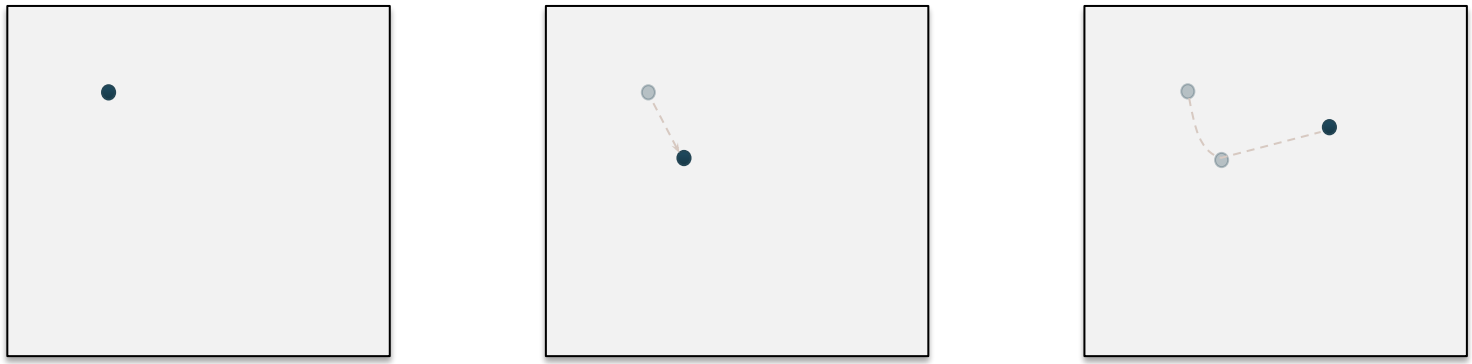
Outline

- Introduction
- Direct exploration technique:
  - Block matching
- Gradient-based techniques:
  - Global approach: Affine model
  - Local approach: Lucas – Kanade
  - Local approach with smoothness constraints: Horn-Schunk
- Other techniques:
  - Brox
  - Deep learning methods
- **Feature tracking**

Feature tracking

Problem:

- Follow visual features along multiple frames



- Optical flow estimation is defined over a pair of images
- Extending temporally (from one frame to another), may result in inconsistent tracks

Feature tracking – Shi-Tomasi (I)

- Use dissimilarity (ε) as a measure of feature quality
  - Quantifies the change of appearance of a feature
  - Residue between first and current frame
  - If dissimilarity grows to large, the feature should be abandoned

$$\epsilon = \sum_{w} \sum [I(\mathbf{A}\vec{r} + \mathbf{b}, t + \Delta t) - I(\vec{r}, t))]^2 \, w(\vec{r}) d\vec{r}$$

- Use of two motion models:
  - Translation provides more reliable results when inter-frame camera translation is small
  - Affine changes are necessary to compare distant frames to determine dissimilarity

Shi, J. and Tomasi, C. “Good features to track”. In CVPR’94, pp. 593–600

Feature tracking – Shi-Tomasi (II)

- Find good features (eigenvalue of structure tensor)
- Use Lucas-Kanade to track with pure translation
- Use affine registration with first feature patch
- Terminate tracks whose dissimilarity gets too large
- Start new tracks when needed

Feature tracking – Shi-Tomasi (III)

Figure 1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

Figure 2: The traffic sign windows from frames 1,6,11,16,21 as tracked (top), and warped by the computed deformation matrices (bottom).

Feature tracking – Shi-Tomasi (III)

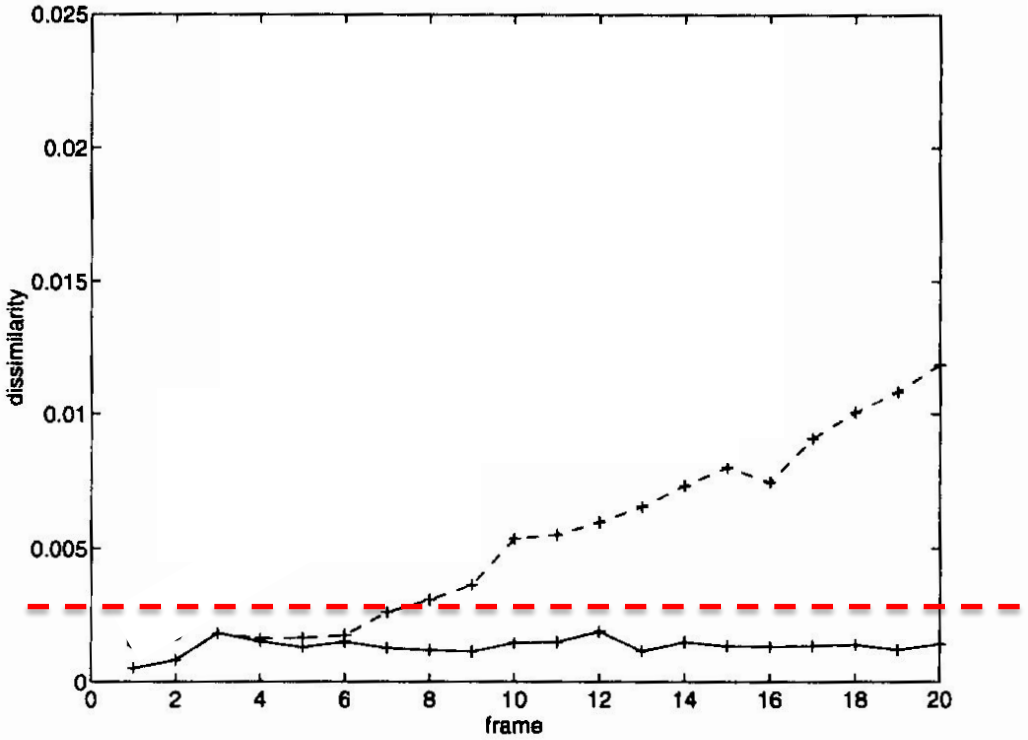


Figure 3: Pure translation (dashed) and affine motion (solid) dissimilarity measures for the window sequence of figure 1 (plusses)

Threshold for valid features

Object tracking using features

Initial detection + tracking

Detect features

Propagate features

Iterate

- Detect in first frame
- Find features
- Track features
- Determine object from features
- Iterate

Tracking by detection

Detect faces

- Detect in all frames
- Find features
- Track features
- Relate objects in the same track
- Iterate