COMPUTER VISION 2018 - 2019

> OPTICAL FLOW

UTRECHT UNIVERSITY

RONALD POPPE

OUTLINE

Optical flow

- Lucas-Kanade
- Horn-Schunck
- Deepflow

Applications of optical flow

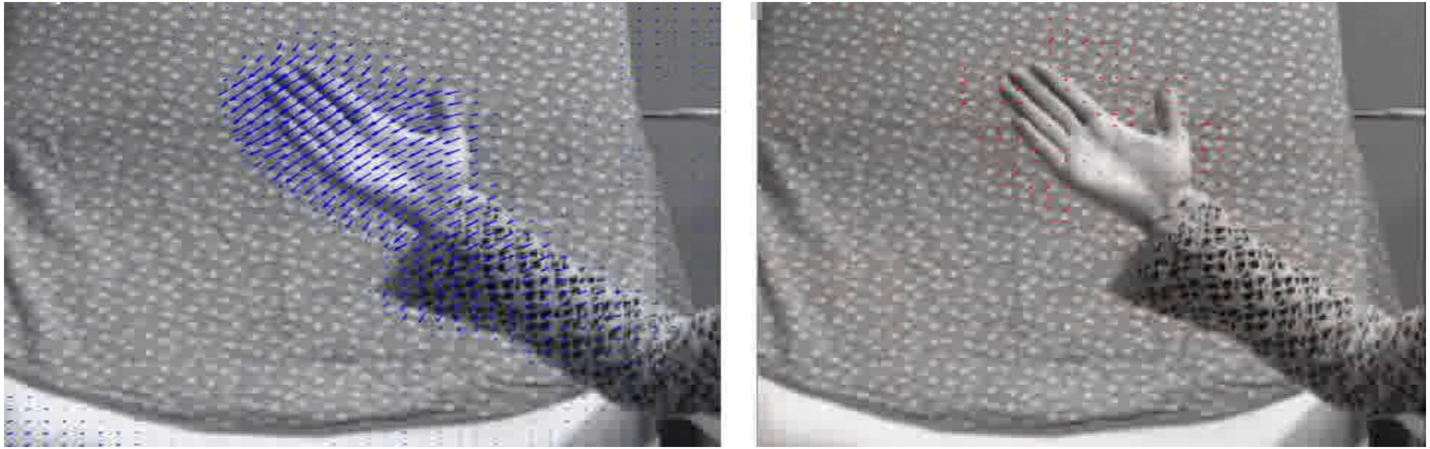
- Optical flow tracking
- Histograms of oriented flow

OPTICAL FLOW

OPTICAL FLOW

The process of estimating the motion field (the trajectory of points) from time-varying image intensity (video)

Or: mapping pixels from one frame to another



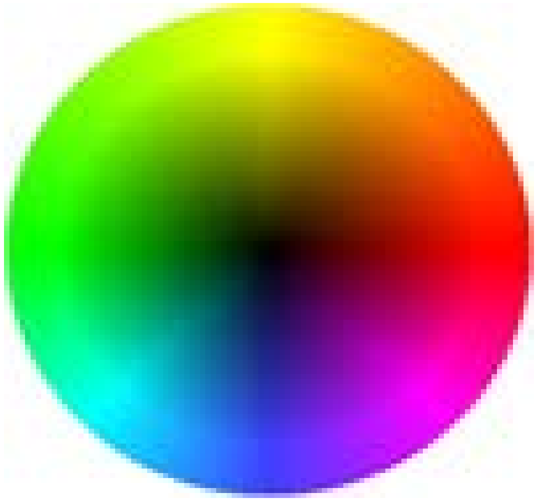
OPTICAL FLOW[2]

For each pixel, we want to know its location in the next frame

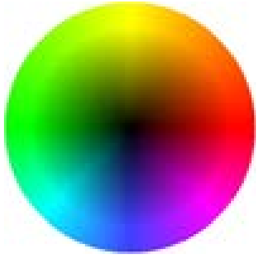
Can be described as a vector:

- Length (amount of movement)
- Direction of movement

Can be color-coded



OPTICAL FLOW[3]

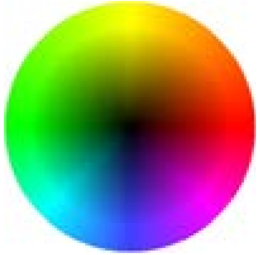


Example: motion in video (Brox et al., PAMI 2011)

http://lmb.informatik.uni-freiburg.de/people/brox/demos.html

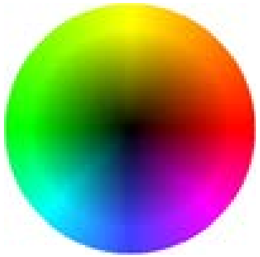
OPTICAL FLOW[4]

Example: motion in video (Brox et al., PAMI 2011)



http://lmb.informatik.uni-freiburg.de/people/brox/demos.html

OPTICAL FLOW[5]



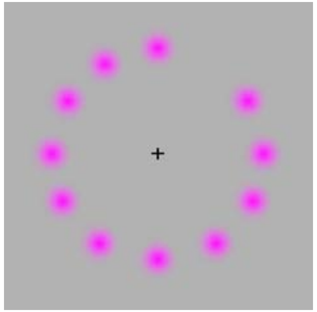
Example: motion in video (Brox et al., PAMI 2011)

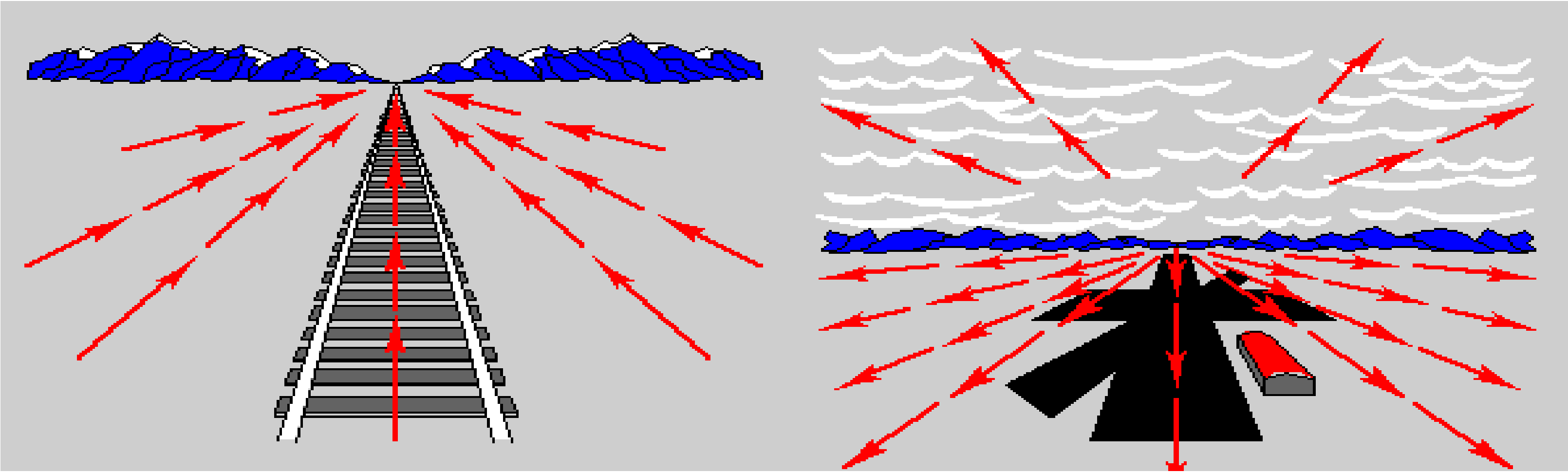
http://lmb.informatik.uni-freiburg.de/people/brox/demos.html

OPTICAL FLOW[6]

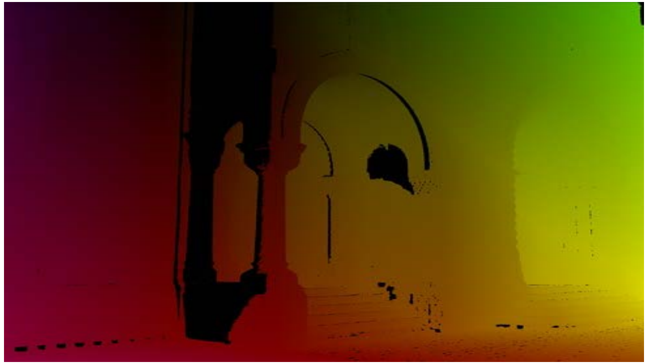
Optical flow is due to:

- Object motion
- Camera motion
- Variations in intensity





APPLICATIONS



Tracking

Estimate depth from flow

- Background subtraction
- Object detection

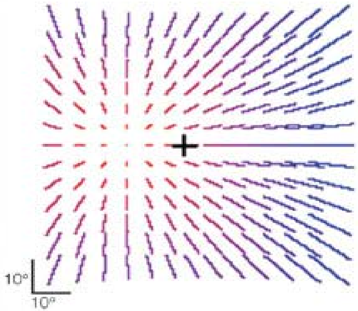


Image stabilization

Activity recognition

left leg kicking | ball moving up | leg retraction | ball coming down

BASIC IDEA

For now, we assume two subsequent frames in a video

Basic idea:

- For each pixel, determine where it is in the next frame
- We call this dense optical flow

Can be done based on differential technique:

- Lucas-Kanade
- Horn-Schunck

Can be done based on matching patches: Deepflow

DIFFERENTIAL TECHNIQUE

ASSUMPTION

Formally:

- (x, y, t) is a pixel (x, y) at time t
- $\boldsymbol{I}(x, y, t)$ the pixel value (intensity)

When analyzing image motion, we use an important assumption based on object appearance:

- The intensity of the pixel under motion does not change
- Brightness constancy assumption (BCA)
- $\boldsymbol{I}(x + dx, y + dy, t + dt) = \boldsymbol{I}(x, y, t)$

ASSUMPTION^2

An image is a 2D projection of a 3D scene!

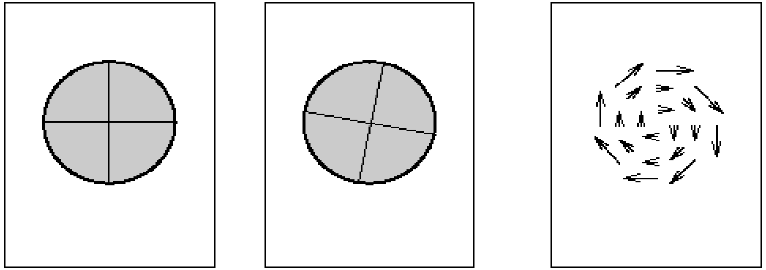
The intensity-constant assumption is typically true for:

- Short-time movement
- Movement paralel to the image plane (in-plane or 2D motion)

But less so for:

- Out-of-plane movement (out-of-plane rotation)
- At “depth borders”
- Occlusions

Discontinuities at the boundaries!

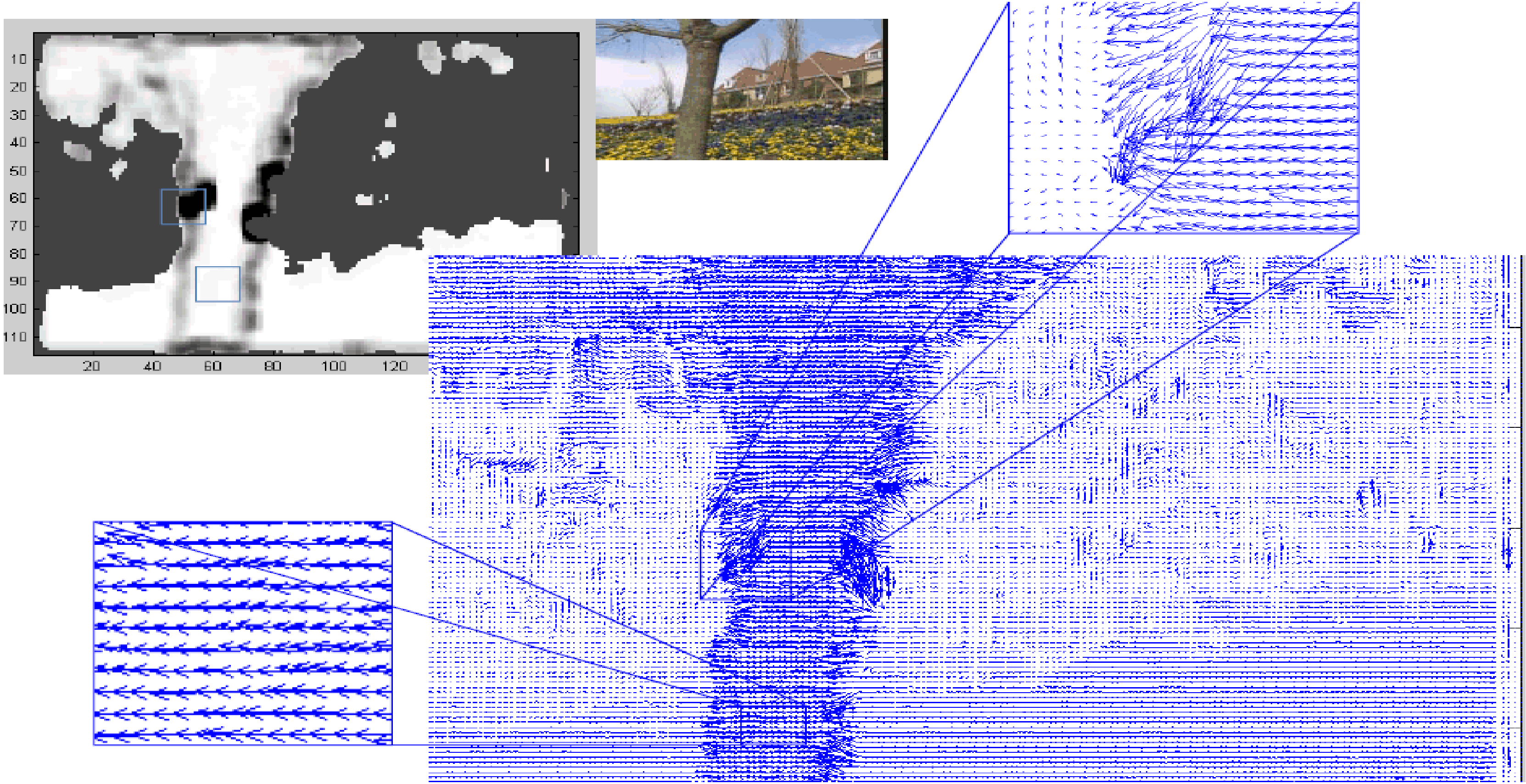


ASSUMPTION[3]

Example video: Garden sequence

http://persci.mit.edu/demos/jwang/garden-layer/orig-seq.html

ASSUMPTION[4]



FORMULATION

Consider a pixel (x,y) **at time** t **with intensity** $I(x, y, t)$

At $t+dt$**, the pixel has moved with** (dx,dy)**:** $(x+dx, y+dy, t+dt)$

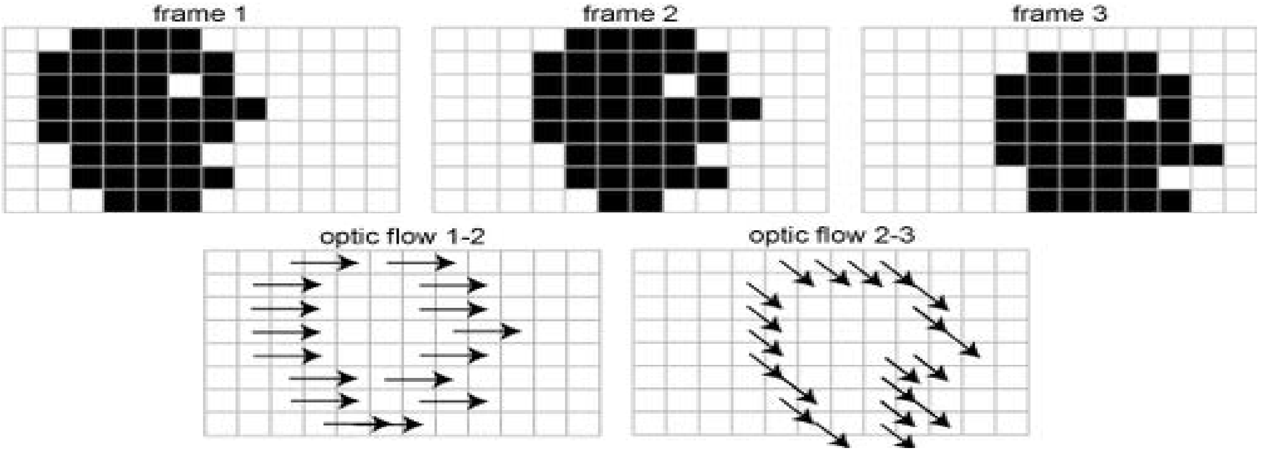
- We use u and v to denote dx/dt and dy/dt
- Vector (u,v) is the apparent 2D motion of the pixel (optical flow)
- We are interested in finding the (u,v) for each pixel

To find the optical flow for an image, we need to find the most likely values of (u,v) **for each pixel**

FORMULATION²

Question: what are the *(u,v)* vectors to go from:

- Frame 1 to frame 2
  - Answer: (2, 0)
- Frame 2 to frame 3
  - Answer: (1, 1)



FORMULATION[3]

Example: $\mathbf{J}(x,y) = \mathbf{I}(x + \mathbf{u}(x, y), y + \mathbf{v}(x, y))$

- $\mathbf{I}(1, 3) = 9$
- $\mathbf{u}(1, 3) = 3$, $\mathbf{v}(1, 3) = 0$
- $\mathbf{J}(1, 3) = \mathbf{I}(1 + \mathbf{u}(1, 3), 3 + \mathbf{v}(1, 3)) = \mathbf{I}(4, 3) = 12$

1	2	3	4
5	6	7	8
9	10	11	12
15	16	13	14

I

5	6	7	8
1	2	3	4
12	11	10	9
13	14	15	16

J

0	0	0	0
0	0	0	0
3	1	-1	-3
2	2	-2	-2

u

1	1	1	1
-1	-1	-1	-1
0	0	0	0
0	0	0	0

v

FORMULATION[4]

Intensity-assumption: $\boldsymbol{I}(x, y, t) = \boldsymbol{I}(x + dx, y + dy, t + dt)$

For a constant intensity (pixel does not change value):

- $\frac{d\boldsymbol{I}}{dt} = \frac{d\boldsymbol{I}(x,y,t)}{dt} = 0$

We will use Taylor series expansion:

- $\boldsymbol{I}(x + \Delta) = \boldsymbol{I}(x) + \Delta\boldsymbol{I}'(x) + \frac{1}{2}\Delta^2\boldsymbol{I}''(x) + \frac{1}{3}\Delta^3\boldsymbol{I}'''(x)$ ...
- Same for *y* and *t*

FORMULATION[5]

With Taylor expansion (only 1st derivative), this becomes:

- $\boldsymbol{I}(x + dx, y + dy + t + dt) = \boldsymbol{I}(x, y, t) + \frac{\partial \boldsymbol{I} dx}{\partial x} + \frac{\partial \boldsymbol{I} dy}{\partial y} + \frac{\partial \boldsymbol{I} dt}{\partial t}$
- Linearized BCA

And its derivative to *t*: $\frac{\partial \boldsymbol{I} dx}{\partial x dt} + \frac{\partial \boldsymbol{I} dy}{\partial y dt} + \frac{\partial \boldsymbol{I}}{\partial t} = 0$

Substituting with *u* and *v*:

- $u = \frac{dx}{dt}, v = \frac{dy}{dt}$
- $\frac{\partial \boldsymbol{I}}{\partial x} u + \frac{\partial \boldsymbol{I}}{\partial y} v + \frac{\partial \boldsymbol{I}}{\partial t} = 0$

FORMULATION[6]

Again: $\frac{\partial \boldsymbol{I}}{\partial x}u + \frac{\partial \boldsymbol{I}}{\partial y}v + \frac{\partial \boldsymbol{I}}{\partial t} = 0$

For convenience, we write:

- $f_x = \frac{\partial \boldsymbol{I}}{\partial x}$, $f_y = \frac{\partial \boldsymbol{I}}{\partial y}$ and $f_t = \frac{\partial \boldsymbol{I}}{\partial t}$
- So: $f_x u + f_y v = -f_t$
- f_x and f_y can be measured from the image → they are the gradients in x- and y-direction!
- Apply derivative masks to first and second frame:
- x: $\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$, y: $\begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}$, t (1st): $\begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix}$, (2nd) $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ (Roberts)

ERROR FUNCTION

We have to estimate *(u,v)* for each pixel

- How to know what a good estimate is?

Remember the intensity-constancy assumption:

- Pixels are not supposed to change in intensity

The "cost" of moving a pixel to a new location:

- $Err(u,v) = \left(f_x u + f_y v + f_t\right)^2$

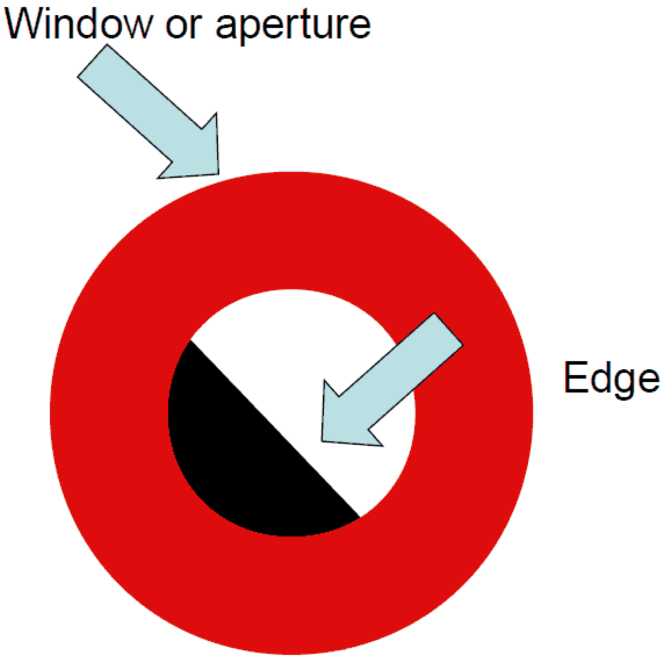
ISSUE

For each pixel, we have equation $f_x u + f_y v = -f_t$

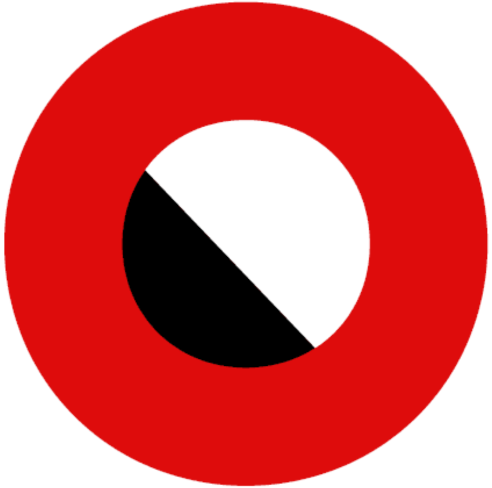
Optical flow:

- *u* and *v* need to be estimated
- Two unknowns, one equation → problem!

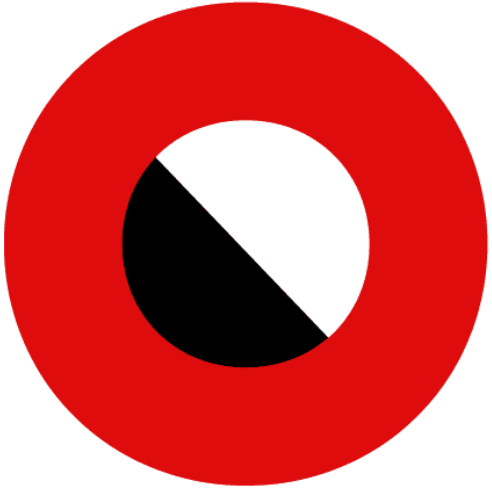
ISSUE 2



ISSUE^3



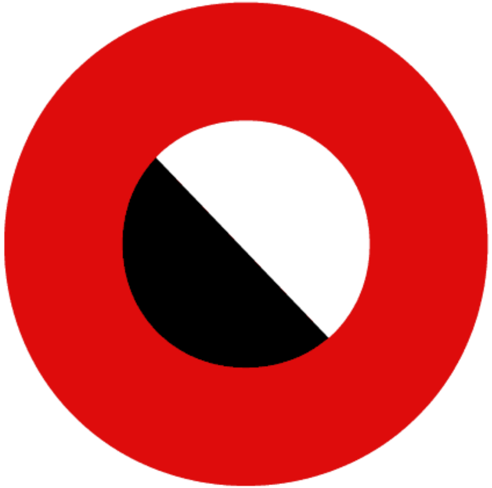
ISSUE[4]



ISSUE 5

ISSUE 6

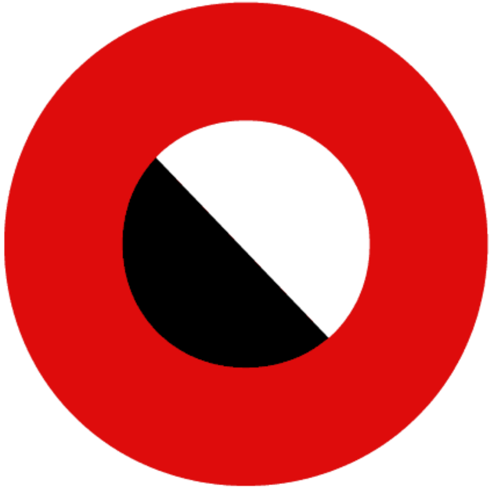
ISSUE 7



ISSUE 8

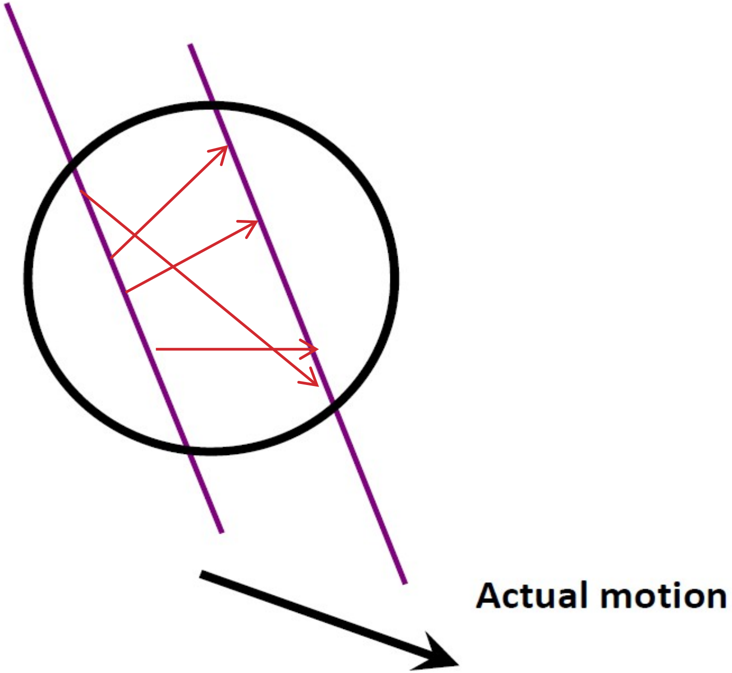
ISSUE 9

ISSUE 10

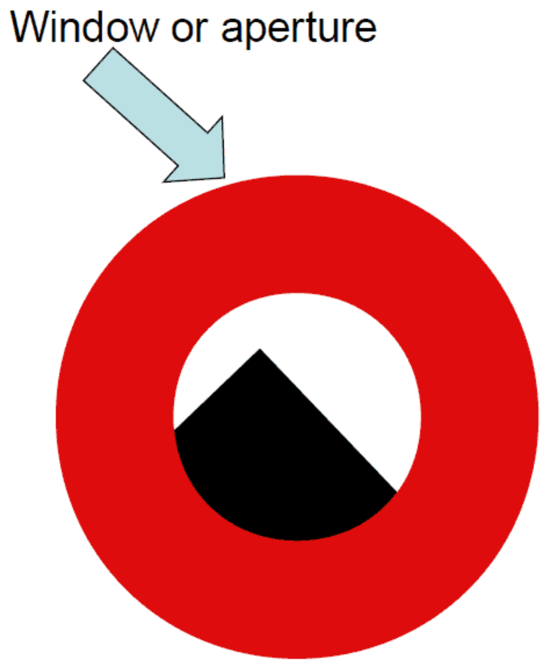


ISSUE[11]

Within the view, we cannot determine the true movement



ISSUE[12]



ISSUE[13]

Animations!

http://web.mit.edu/persci/demos/Motion&Form/demos/two-squares/two-squares.html

ISSUE[13]

So we need more equations to estimate the optical flow *(u,v)* for each pixel

- Aperture problem when f_x and/or f_y are zero (even areas/edges) because of $f_x u + f_y v = -f_t$
- So we need contrast in both horizontal and vertical direction
- Ideally, we include a corner to be sure about the estimate

Two options:

- Look at motion in small region: Lucas-Kanade
- Introduce additional constraints: Horn-Schunck

LUCAS-KANADE

LUCAS-KANADE

Basic idea:

- Assume that neighboring pixels have same optical flow
- Consider a small window around a pixel
- More equations with the same two unknowns → solvable!

Formally:

- For each pixel *i* in window *W*, we have: $f_{x_i}u + f_{y_i}v = -f_{t_i}$
- Written in matrix form: $\begin{bmatrix} f_{x_1} & f_{y_1} \\ \vdots & \vdots \\ f_{x_n} & f_{y_n} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -f_{t_1} \\ \vdots \\ -f_{t_n} \end{bmatrix}$
- *n:* the number of pixels in the window (e.g., *W=5x5* → *n=25*)

LUCAS-KANADE²

So we have a series of equations:

- $\begin{bmatrix} f_{x_1} & f_{y_1} \\ \vdots & \vdots \\ f_{x_n} & f_{y_n} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -f_{t_i} \\ \vdots \\ -f_{t_i} \end{bmatrix}$
- Over-complete system!

Recall our error term:

- $Err(u,v) = \left(f_x u + f_y v + f_t\right)^2$
- Adapted for windows: $Err(u,v) = \sum_i \left(f_{x_i} u + f_{y_i} v + f_{t_i}\right)^2$
- We want to find optical flow *(u,v)* for which the sum of pixel intensity differences is smallest → minimize the error

LUCAS-KANADE[3]

We want to minimize this error:

- $Err(u,v) = \sum_i \left(f_{x_i}u + f_{y_i}v + f_{t_i}\right)^2$
- Typical solution: solve for *u* and *v* separately by setting their partial derivatives to zero

- Use chain rule ($\frac{dx^2}{dx} = 2x, \frac{df^2}{dx} = 2f * \frac{df}{dx}$):
- $\frac{\partial Err(u,v)}{\partial u} = \sum_i 2\left(f_{x_i}u + f_{y_i}v + f_{t_i}\right)f_{x_i} = 0$ and
- $\frac{\partial Err(u,v)}{\partial v} = \sum_i 2\left(f_{x_i}u + f_{y_i}v + f_{t_i}\right)f_{y_i} = 0$
- The 2 term can be dropped (it is a constant)

LUCAS-KANADE[4]

We can write:

- $\sum_i \left(f_{x_i} u + f_{y_i} v + f_{t_i}\right) f_{x_i} = 0$
- $\sum_i \left(f_{x_i} u + f_{y_i} v + f_{t_i}\right) f_{y_i} = 0$

As:

- $\sum_i {f_{x_i}}^2 u + \sum_i f_{x_i} f_{y_i} v = -\sum_i f_{x_i} f_{t_i}$
- $\sum_i f_{x_i} f_{y_i} u + \sum_i {f_{y_i}}^2 v = -\sum_i f_{y_i} f_{t_i}$

And in matrix form:

- $\begin{bmatrix} \sum_i {f_{x_i}}^2 & \sum_i f_{x_i} f_{y_i} \\ \sum_i f_{x_i} f_{y_i} & \sum_i {f_{y_i}}^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum_i f_{x_i} f_{t_i} \\ -\sum_i f_{y_i} f_{t_i} \end{bmatrix}$

LUCAS-KANADE[5]

With the same analogy as:

- $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$
- We can pre-multiply both sides of:
- $\begin{bmatrix} \sum_i {f_{x_i}}^2 & \sum_i f_{x_i} f_{y_i} \\ \sum_i f_{x_i} f_{y_i} & \sum_i {f_{y_i}}^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum_i f_{x_i} f_{t_i} \\ -\sum_i f_{y_i} f_{t_i} \end{bmatrix}$ with the inverse:
- $\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i {f_{x_i}}^2 & \sum_i f_{x_i} f_{y_i} \\ \sum_i f_{x_i} f_{y_i} & \sum_i {f_{y_i}}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i f_{x_i} f_{t_i} \\ -\sum_i f_{y_i} f_{t_i} \end{bmatrix}$, which gives us:
- $\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\sum_i {f_{x_i}}^2 \sum_i {f_{y_i}}^2 - \left(\sum_i f_{x_i} f_{y_i}\right)^2} \begin{bmatrix} \sum_i {f_{y_i}}^2 & -\sum_i f_{x_i} f_{y_i} \\ -\sum_i f_{x_i} f_{y_i} & \sum_i {f_{x_i}}^2 \end{bmatrix} \begin{bmatrix} -\sum_i f_{x_i} f_{t_i} \\ -\sum_i f_{y_i} f_{t_i} \end{bmatrix}$

LUCAS-KANADE[6]

From:

- $\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\sum_i {f_x}_i{}^2 \sum_i {f_y}_i{}^2 - \left(\sum_i {f_x}_i {f_y}_i\right)^2} \begin{bmatrix} \sum_i {f_y}_i{}^2 & -\sum_i {f_x}_i {f_y}_i \\ -\sum_i {f_x}_i {f_y}_i & \sum_i {f_x}_i{}^2 \end{bmatrix} \begin{bmatrix} -\sum_i {f_x}_i {f_t}_i \\ -\sum_i {f_y}_i {f_t}_i \end{bmatrix}$

Eventually, we can calculate the optical flow in both the *u* and *v* direction:

- $u = \frac{-\sum_i {f_y}_i{}^2 \sum_i {f_x}_i f_t + \sum_i {f_x}_i {f_y}_i \sum_i {f_y}_i {f_t}_i}{\sum_i {f_x}_i{}^2 \sum_i {f_y}_i{}^2 - \left(\sum_i {f_x}_i {f_y}_i\right)^2}$
- $v = \frac{\sum_i {f_x}_i f_t \sum_i {f_x}_i {f_y}_i - \sum_i {f_x}_i{}^2 \sum_i {f_y}_i {f_t}_i}{\sum_i {f_x}_i{}^2 \sum_i {f_y}_i{}^2 - \left(\sum_i {f_x}_i {f_y}_i\right)^2}$

LUCAS-KANADE[7]

We could also calculate this directly:

- $\begin{bmatrix} f_{x_1} & f_{y_1} \\ \vdots & \vdots \\ f_{x_n} & f_{y_n} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -f_{t_i} \\ \vdots \\ -f_{t_i} \end{bmatrix}$
- $\boldsymbol{B}[u \quad v]^T = \boldsymbol{C}$, with $\boldsymbol{B} = \begin{bmatrix} f_{x_1} & f_{y_1} \\ \vdots & \vdots \\ f_{x_n} & f_{y_n} \end{bmatrix}$ and $\boldsymbol{C} = \begin{bmatrix} -f_{t_i} \\ \vdots \\ -f_{t_i} \end{bmatrix}$
- $[u \quad v]^T = (\boldsymbol{B}^T\boldsymbol{B})^{-1}\boldsymbol{B}^T\boldsymbol{C}$
- $(\boldsymbol{B}^T\boldsymbol{B})^{-1}$ is the pseudo-inverse

Reworking the right-hand-side will result in the same equations

LUCAS-KANADE[8]

Lucas-Kanade in practice uses small windows:

- 3x3 or 5x5
- Cannot cope with larger movements (outside the window)

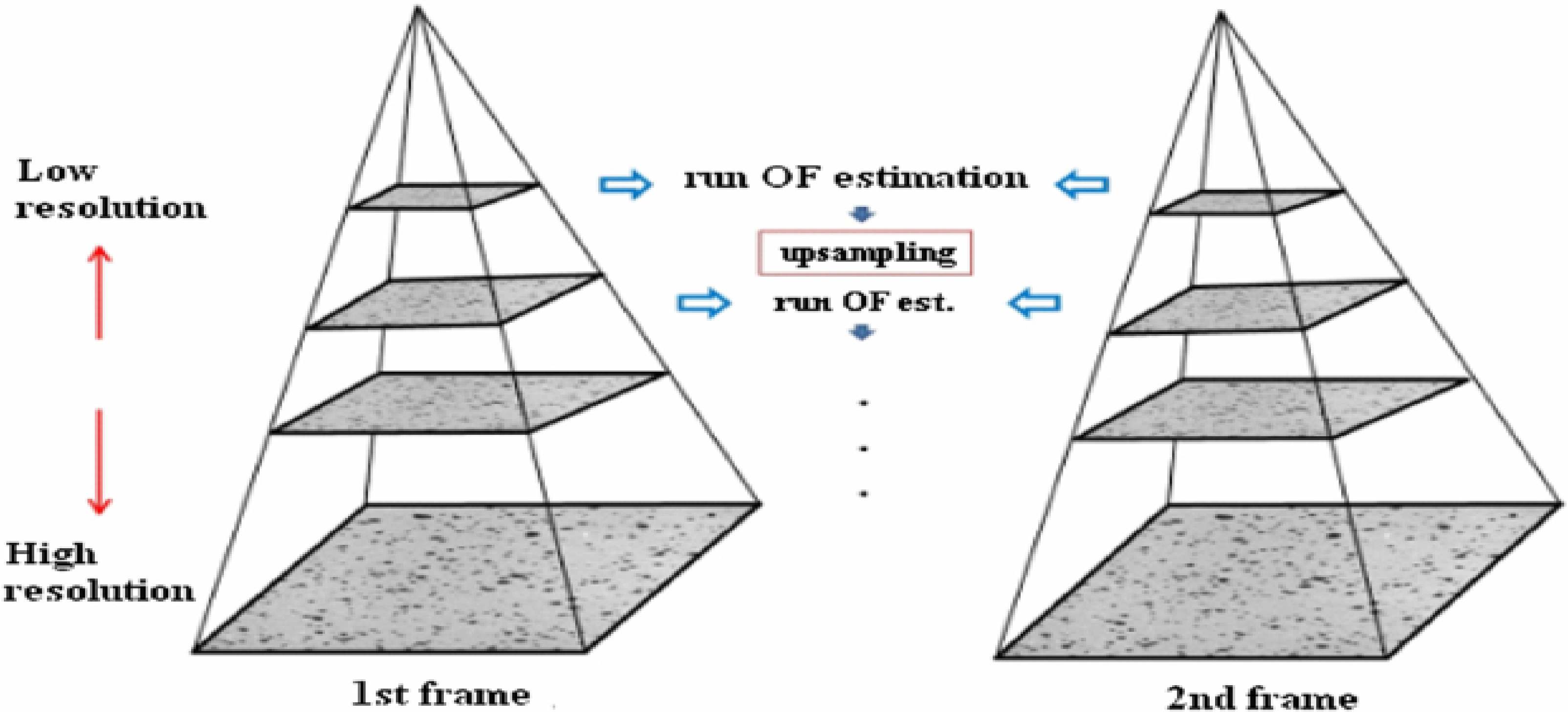
For example:

- With a 5x5 window, *u* and *v* cannot be larger than 2 in any direction
- When the movement is larger, the results are unstable → the best optical flow estimate is out of reach

Solution:

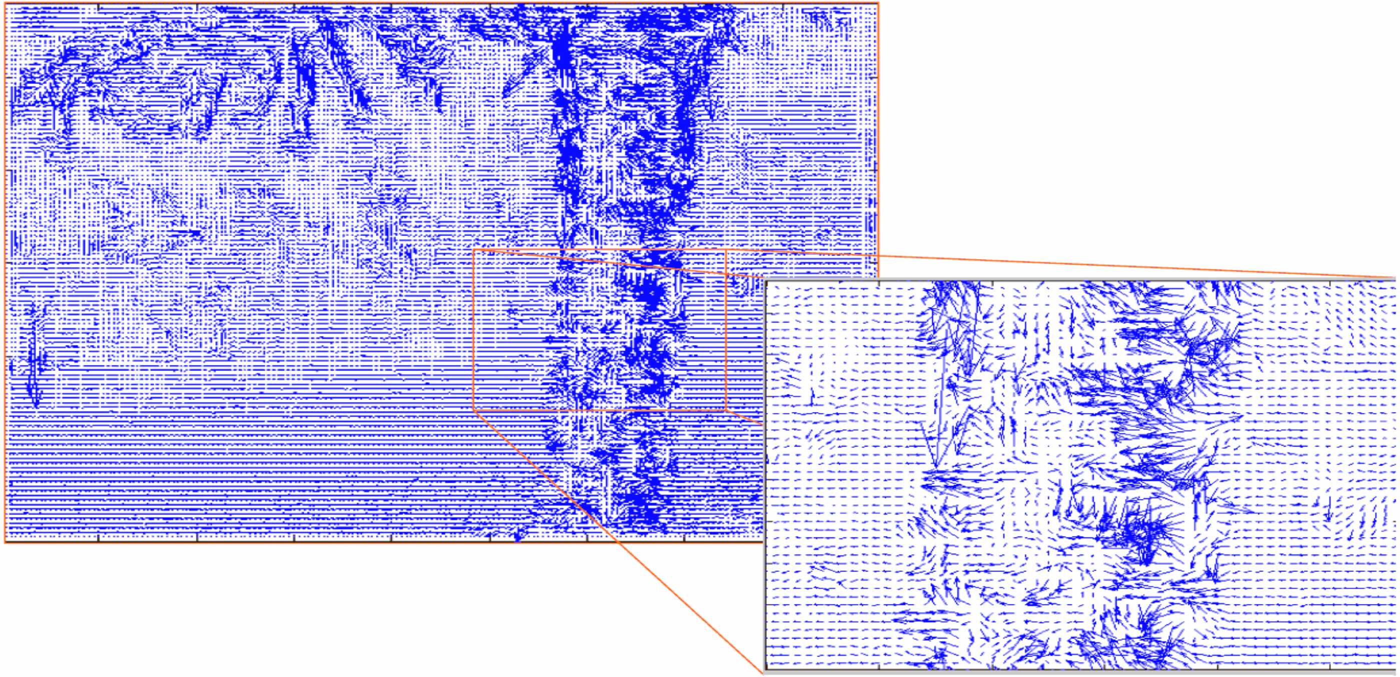
- Use pyramids (recall SIFT lecture)

LUCAS-KANADE[9]



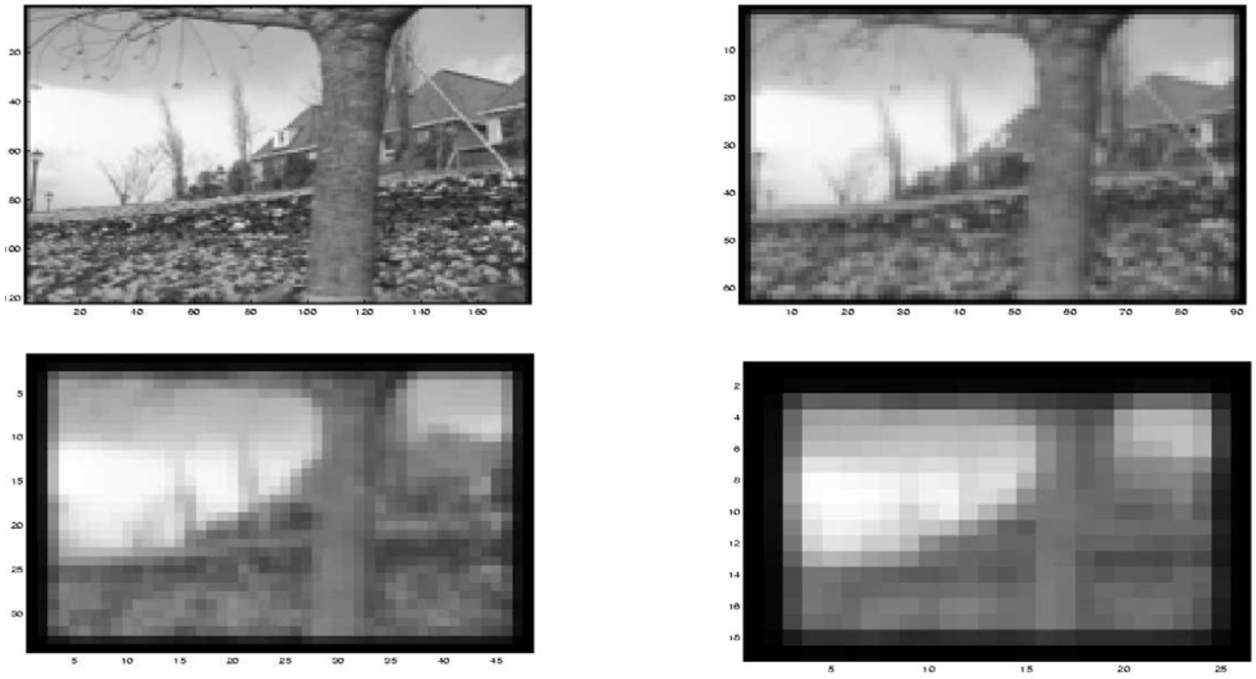
LUCAS-KANADE[10]

Without pyramids: unstable results with larger motion



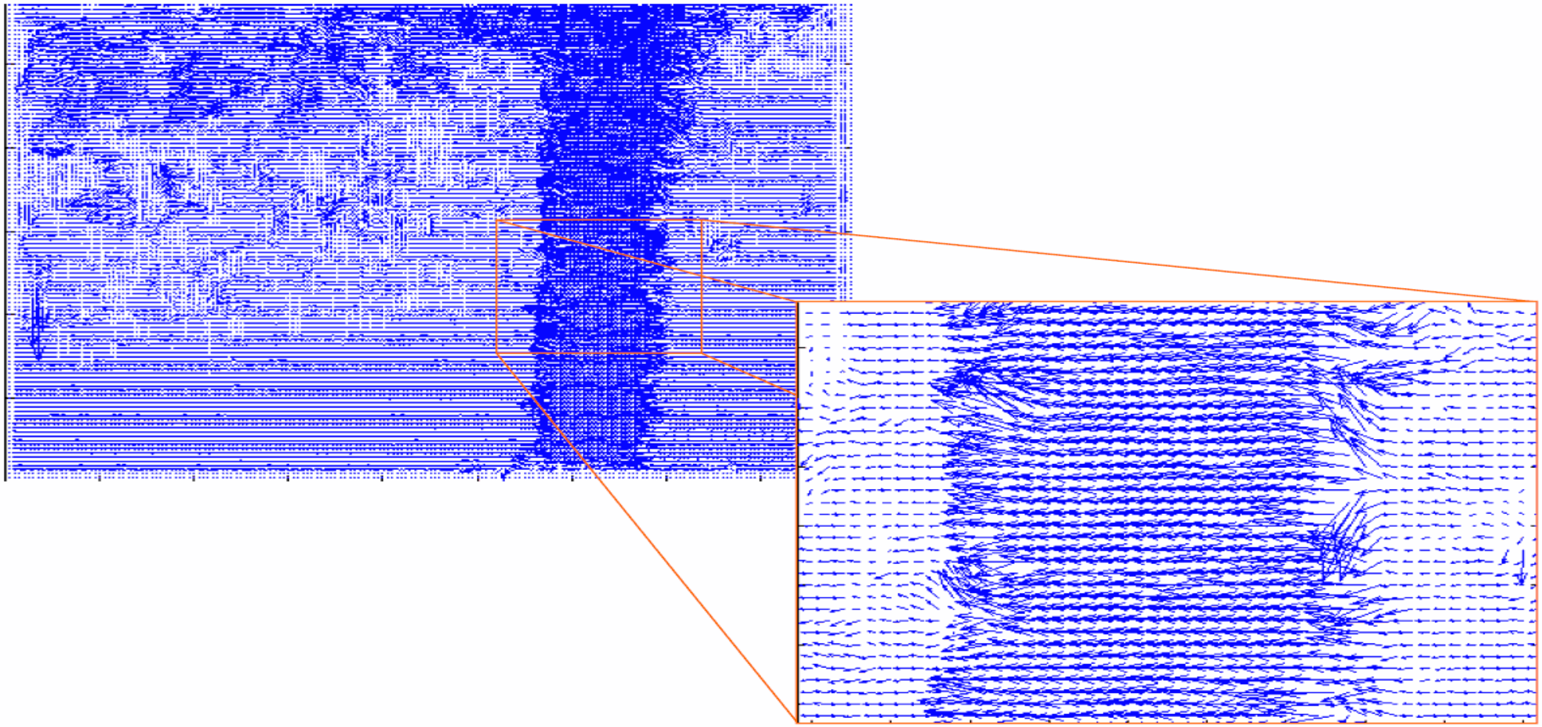
LUCAS-KANADE[11]

Application of pyramids: resolution is much lower



LUCAS-KANADE[12]

With pyramids: stable results with larger motion



LUCAS-KANADE[13]

Some of the properties of Lucas-Kanade:

- Depends on image gradients (f_x and f_y)
- For uniform colors → $f_x=f_y=0$, optical flow is not defined
- Fast!

Usually applied for each pixel: dense optical flow estimation

- Can also be applied only around corners (similar to SIFT) → sparse result

Noise has large effect on gradient, so also on optical flow estimation

- Can be reduced by applying a Gaussian filter

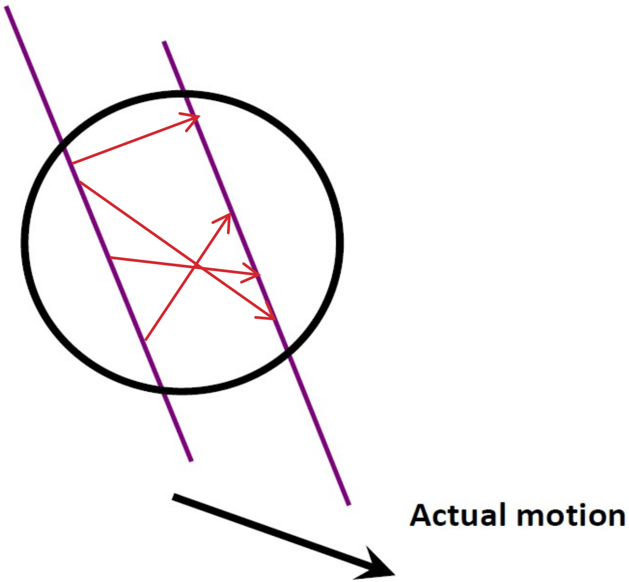
QUESTIONS?

HORN-SCHUNCK

HORN-SCHUNCK

Instead of using a window of pixels, Horn-Schunck optical flow estimation uses a smoothness constraint:

- Smaller values of *u* and *v* are favored over larger ones



HORN-SCHUNCK[2]

Original error function:

- $Err(u,v) = \left(f_x u + f_y v + f_t\right)^2$

Error function in Horn-Schunck:

- $Err(u,v) = \iint \left(f_x u + f_y v + f_t\right)^2 + \lambda\left((\nabla u)^2 + (\nabla v)^2\right) dxdy$
- With ∇ the del operator: $\nabla = \left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \ldots \frac{\partial}{\partial x_n}\right)$ (vector of partial derivatives to each variable)

HORN-SCHUNCK[3]

Horn-Schunck error function:

- $$Err(u,v) = \iint \underbrace{\left(f_x u + f_y v + f_t\right)^2}_{\text{Intensity assumption}} + \underbrace{\lambda}_{\text{Weighting factor}} \underbrace{\left((\nabla u)^2 + (\nabla v)^2\right)}_{\text{Smoothness term}} dx\,dy$$

Again, we want to minimize *(u, v)* to have the lowest error

We are searching for *(u, v)* that:

- Follows the intensity assumption, and
- Minimizes the estimated motion
- Global estimation (hence the integrals)

HORN-SCHUNCK[4]

Properties of Horn-Schunck optical flow:

- Smooth flow (no “jumps”) owing to the smoothness constraints
- Global information, so larger motion possible
- Slow because of iterative nature → also might get stuck in local minimum
- Still: problems at the boundaries → but this an intrisically hard issue

DEEPFLOW

Weinzaepel et al., ICCV 2013 / Revaud, IJCV 2016

- Introduces matching stage to deal with large displacements

Recall Lucas-Kanade:

1. Assumes that motion is similar in small neighborhoods
2. Pyramid-approach to deal with large displacements

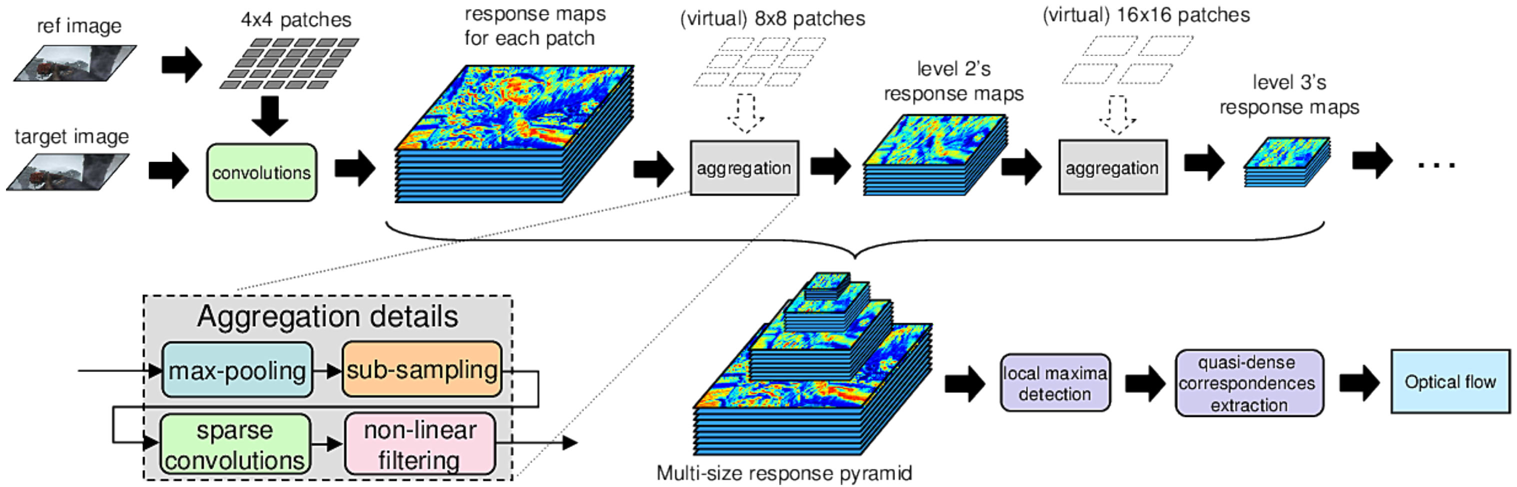
Recall Horn-Schunck:

1. Employs a smoothness term
2. Globally optimal estimation (approximated)

DEEPFLOW^2

Deepflow introduces a matching stage

- First roughly determine the displacement of patches
- Then determine pixel-level flow with smoothness constraint



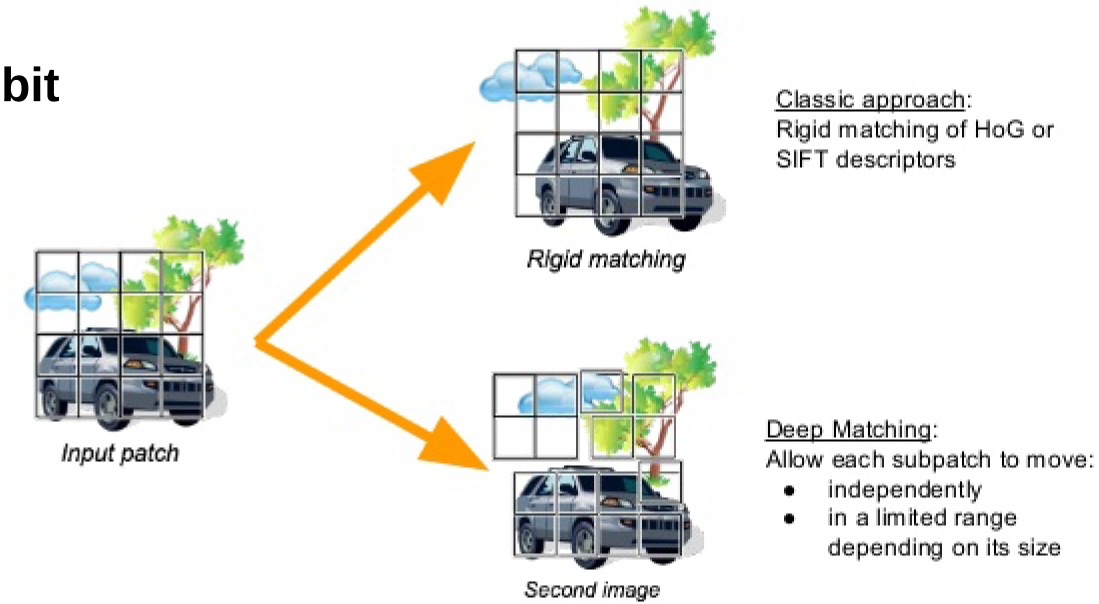
DEEPFLOW[3]

Matching shares ideas from SIFT:

- Encode a region in the input image using 4x4 cells with HOG

Allow each region to move a bit

in the second image



DEEPFLOW[4]

We can convolve the second image with a kernel based on each region

- Produces a response map
- High scores indicate high similarity

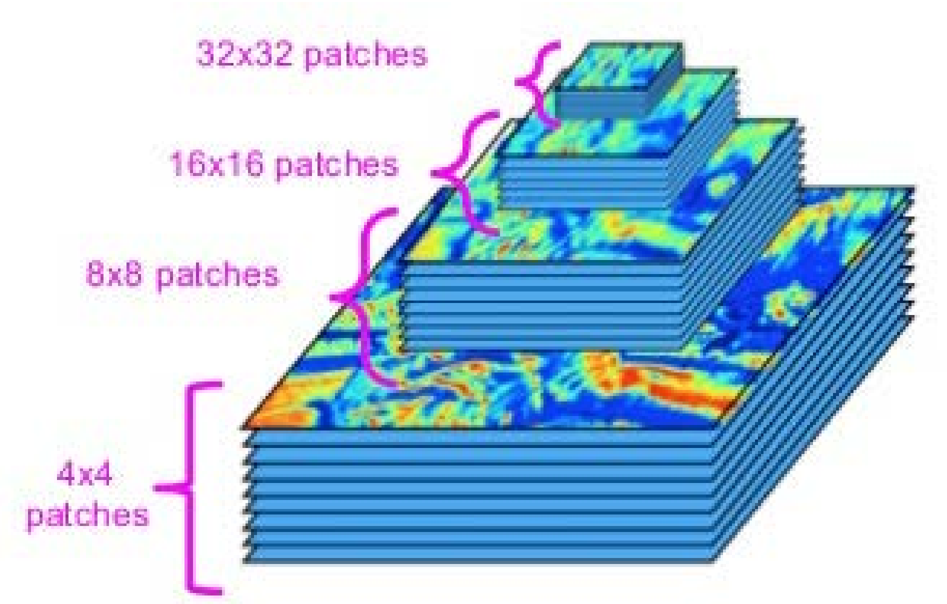
Regions can look a lot like other regions

- Especially with repeated textures (bricks, etc.)
- Processing at multiple scales (downsampling) – similar rationale as pyramids

DEEPFLOW[5]

Multiscale convolution creates a pyramid of response maps

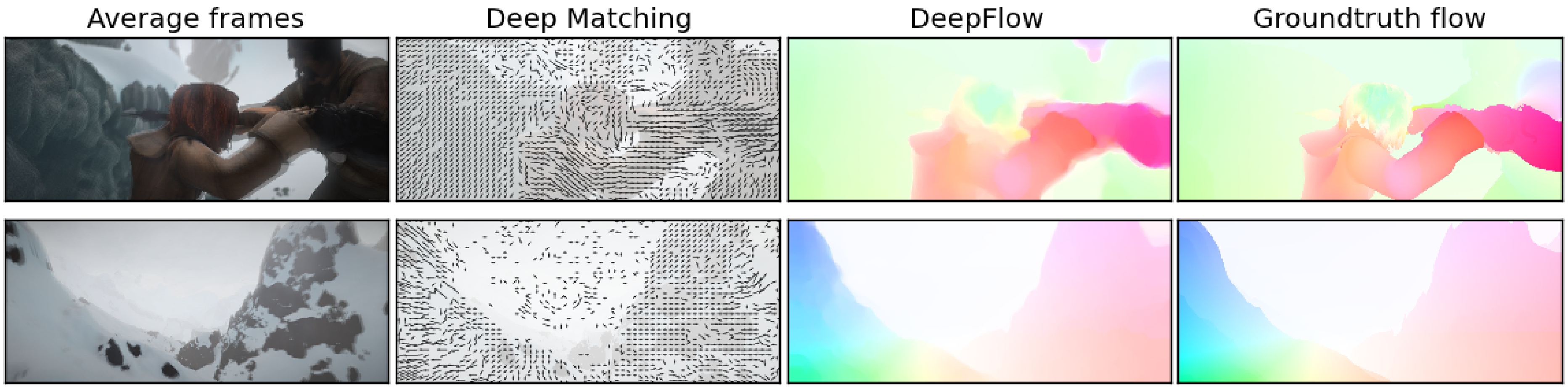
- Determine the optimal displacement as a function of the responses at various scales
- Favor coarser scales



DEEPFLOW[6]

Last step is to combine various sources of information:

- Deformation from the matching
- Data term (gradient information)
- Smoothness term



DEEPFLOW[7]

Relation to other topics:

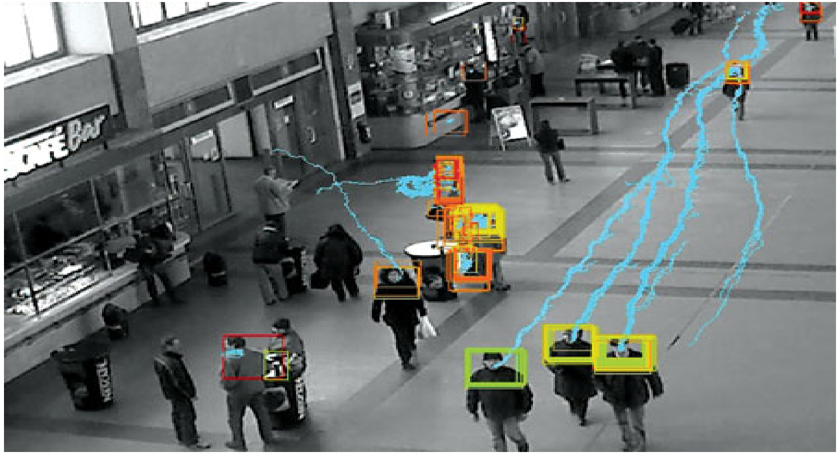
- Coarse-to-fine strategy using pyramid (cf. SIFT)
- Encoding of local regions (cf. SIFT)
- Convolutions
- Deepflow is NOT a deep learning method (no learning but subsequent convolution operations)

APPLICATIONS OF OPTICAL FLOW

OPTICAL FLOW TRACKING

Given an estimate of the movement of a pixel, we can track it

- Kanade-Lucas-Tomasi (KLT) is a well-known tracker



KLT TRACKING

KLT (Kanade-Lucas-Tomasi) tracking works by estimating the optical flow from frame to frame at certain points :

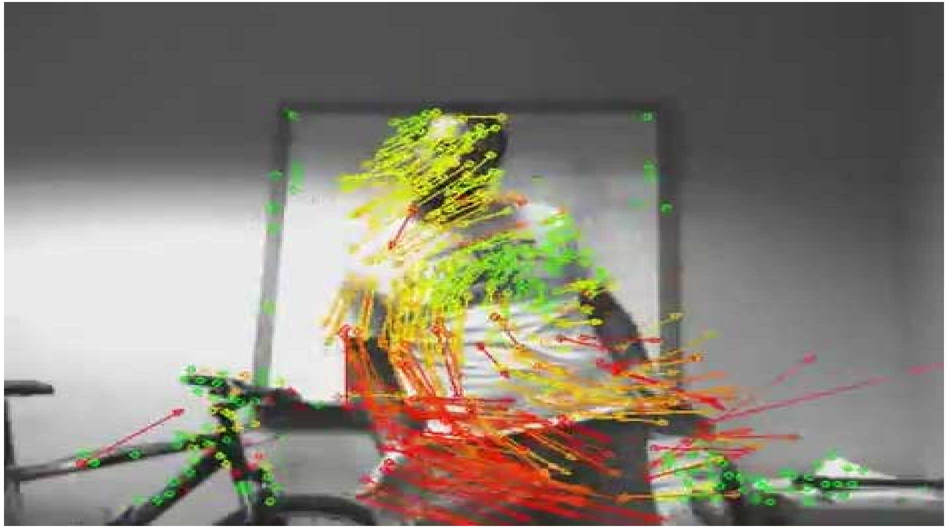
- Typically, Harris corners are used
- Found on corners with strong gradient
- Comparable to interest points (remember from SIFT)

KLT tracking is straightforward for translations of single points, more difficult for:

- More complex transformations (rotations, affine)
- Multiple points (e.g. a region of points belonging to the same object)

KLT TRACKING

Circles are tracked points (movement vectors are multiplied)



http://www.youtube.com/watch?v=E86NLzNbuL8

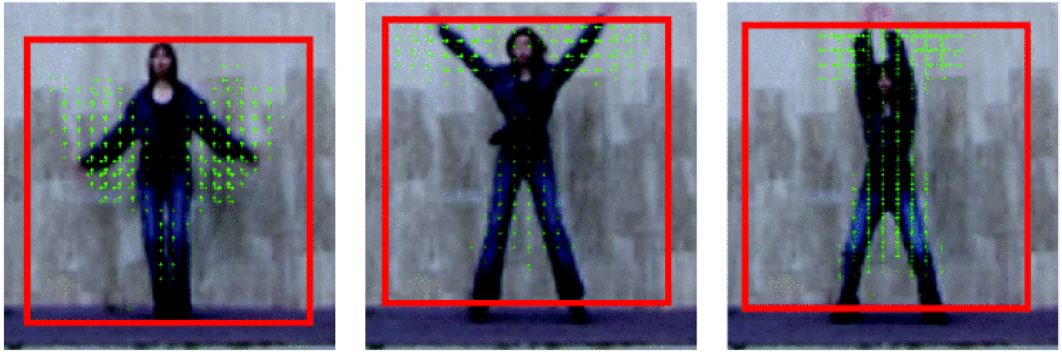
HISTOGRAMS OF ORIENTED FLOW

We can calculate optical flow within a bounding box

- Often, specific movement in parts of the region is meaningful

We can use the histogram of oriented gradients (HOG) concept

- Optical flow has a magnitude and orientation
- Bin flow instead of gradients



QUESTIONS?

ASSIGNMENT

ASSIGNMENT

Assignment 3:

- Deadline is Sunday March 10, 23:00
- No more help sessions for this assignment
- Use Slack and contact Breixo with questions

NEXT LECTURE

Next lecture: Training, classification, detection

- Thursday March 7, 11:00-12:45, RUPPERT-042