Univerza v Ljubljani

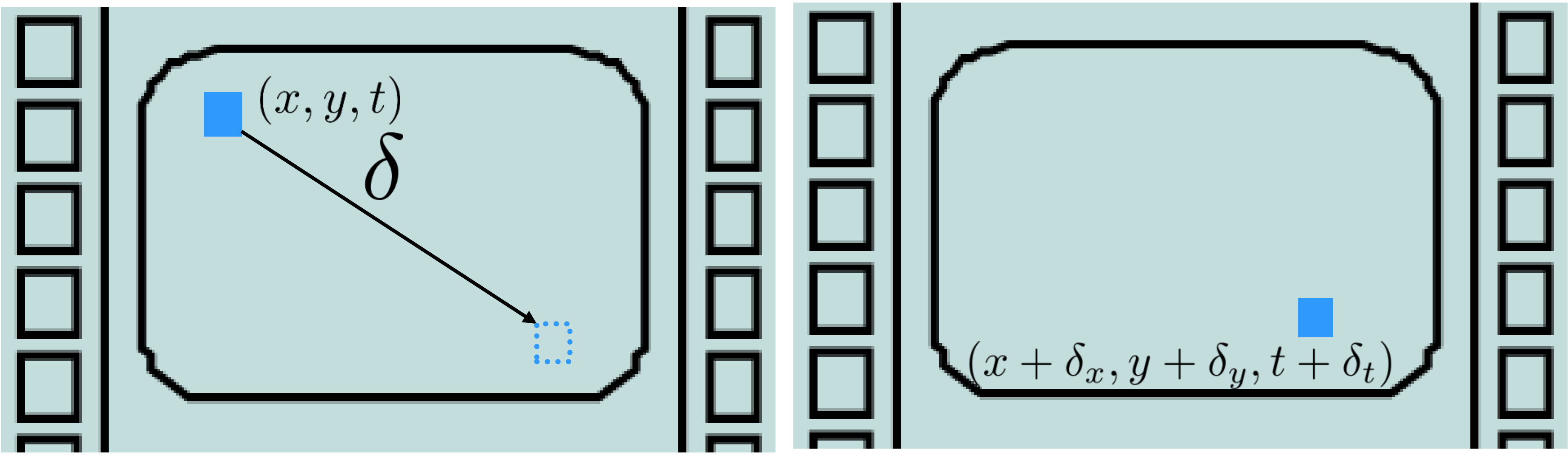
Advanced CV methods
Optical flow 2

Matej Kristan

Laboratorij za Umetne Vizualne Spoznavne Sisteme,
Fakulteta za računalništvo in informatiko,
Univerza v Ljubljani

Previously at ACVM...

- Flow: estimate translation vectors for all pixels just by considering two consecutive images.



frame t

frame t+1

- Assumptions required!

Previously at ACVM…

- Brightness constancy assumption:

$$I(\mathbf{x}) = I(\mathbf{x} + \delta)$$

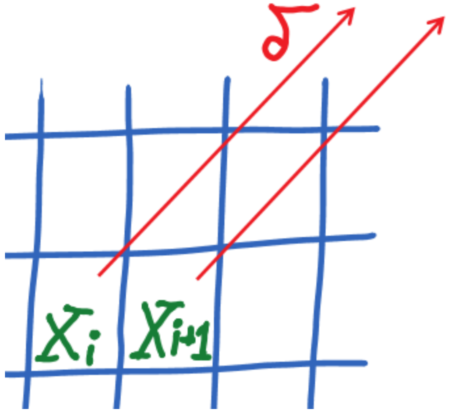
Image at t   Image at t+1

- Small displacement assumption:

$$I(\mathbf{x} + \delta) \approx I(\mathbf{x}) + \nabla I^T \mathbf{J} \delta$$

- Optical flow equation (underdetermined system):

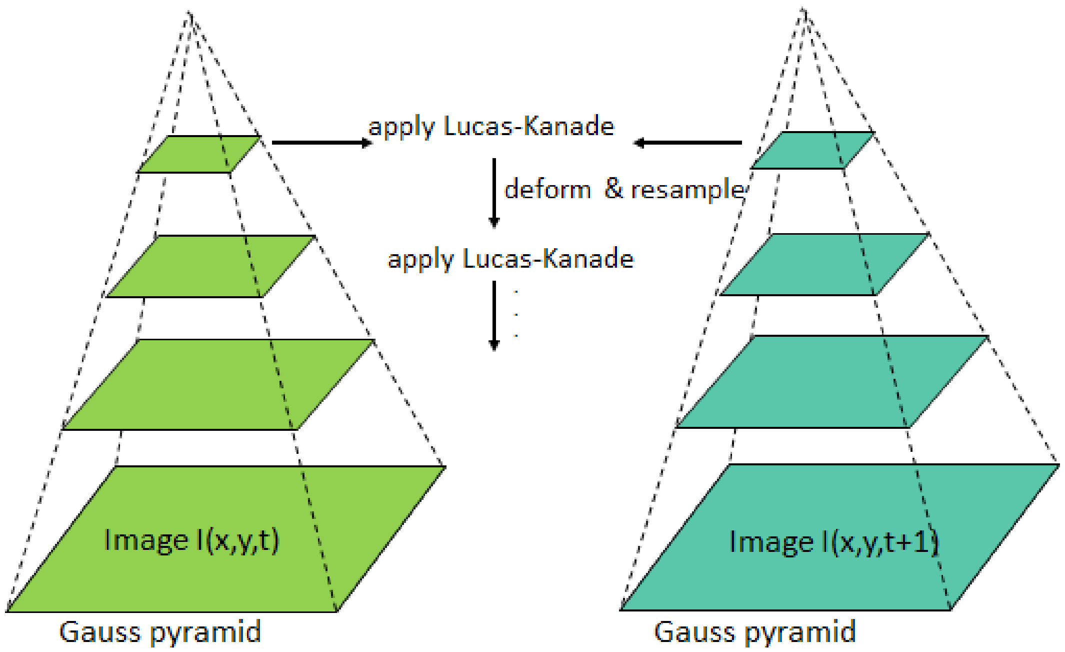
$$I_x(\mathbf{x}_i)\delta_x + I_y(\mathbf{x}_i)\delta_y + I_t(\mathbf{x}_i) = 0$$

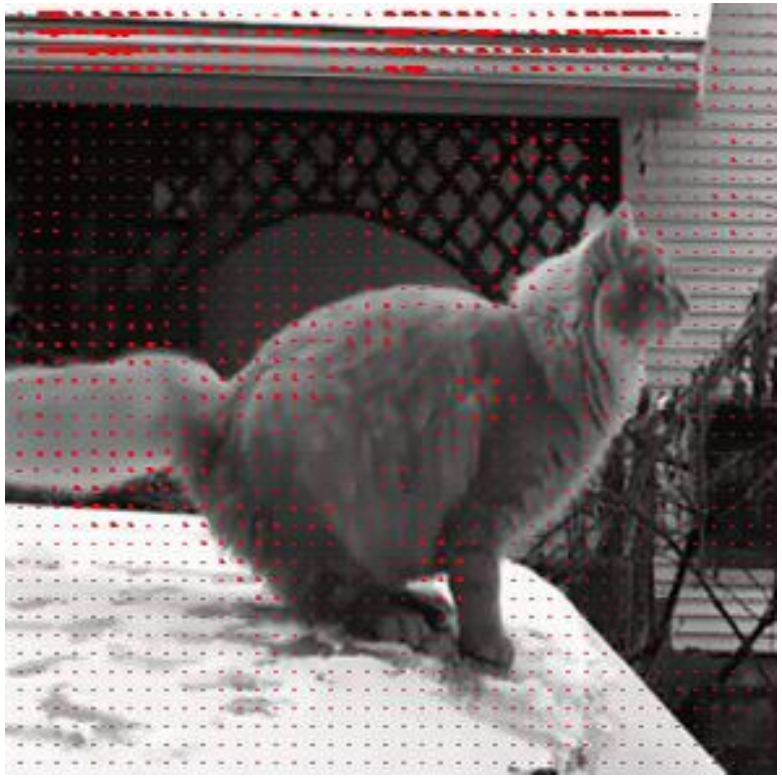


- LK solution: neighboring points move similarly, so we can solve for the displacements via least squares.

Previously at ACVM…

- Small motion assumption often violated!
- Addressed by pyramid implementation of OF





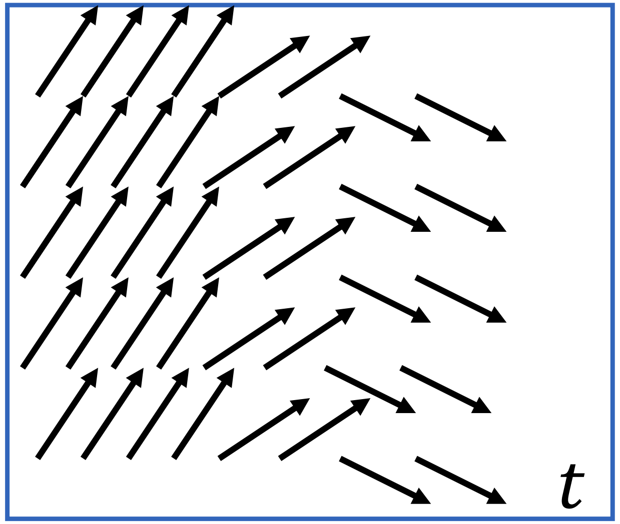
Waffles!

Alternative solution

- Consider optical flow estimation as an energy minimization problem.
- Approach:
  - Define a single energy function E that depends on all flow vectors (flow field) in the image.
  - Minimize E w.r.t. flow field!

  - Energy function in terms of the standard constraints:
    - Brightness and motion constraint
    - Smoothness of the field constraint.

Flow field

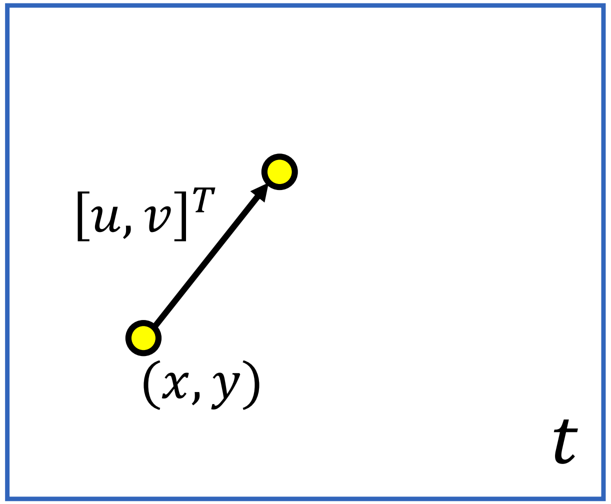


Brightness + Small motion cost

- *Construct a cost E_c that reflects per-pixel flow quality*
- Brightness+Small motion constraint at pixel (x, y):

$$I_x(x,y)u(x,y) + I_y(x,y)v(x,y) + I_t(x,y) = 0$$

Note: the notation changed for clarity, $[\delta_x, \delta_y] = [u, v]$



1.) Cost of (u, v) at pixel (x, y):

$$E_c(u, v, x, y) = ?$$

2.) Sum of costs over all pixels:

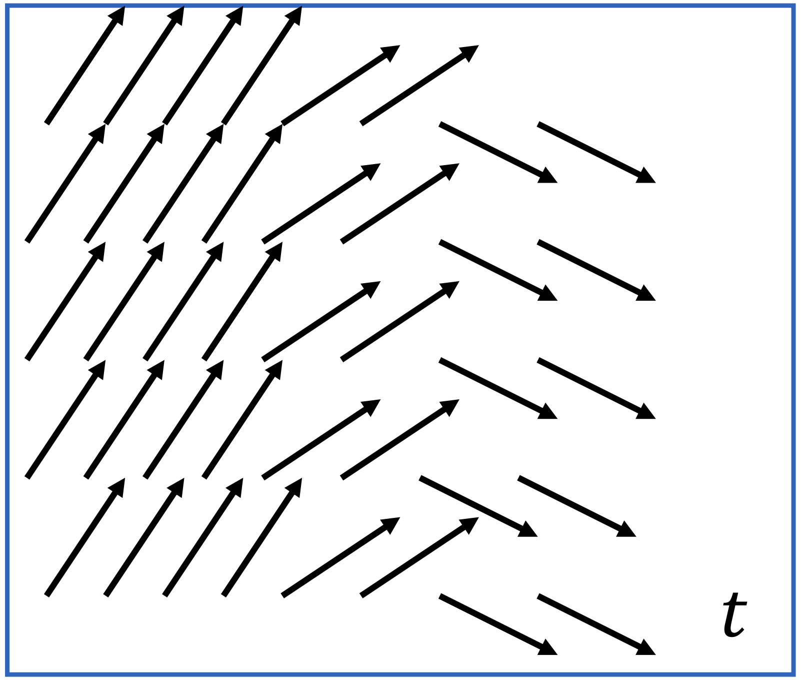
$$E_c = \int_{all\ (x,y)} E_c(u, v, x, y)\, dx dy$$

Flow agrees with constraint → E_c low ; Flow disagrees with constraint → E_c high

Brightness + Small motion cost

- Sum of errors in brightness constancy:

$$E_c = \iint_D (I_x u + I_y v + I_t)^2 \, dx dy$$



D ... image domain

Note:
All variables, I_x, I_y, u, v depend on (x, y) in the above equation!

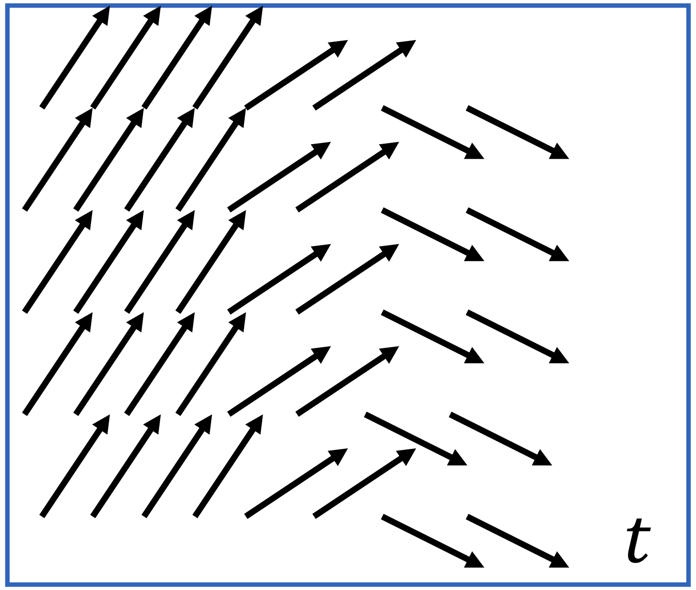
Smoothness cost

- Flow smoothness error:

Smoothness cost

- Error at (x, y): $E_s(\mathrm{u}, \mathrm{v}, \mathrm{x}, \mathrm{y}) = \left(u_x^2 + u_y^2\right) + \left(v_x^2 + v_y^2\right)$
- Smoothness of flow field error:

$$E_s = \iint_D (u_x^2 + u_y^2) + (v_x^2 + v_y^2) dxdy$$



D ... image domain

Note:
All variables, $\mathrm{I_x}, \mathrm{I_y}, u, v$ depend on (x, y) in the above equation!

The flow field energy function

- Flow smoothness error:

$$E_s = \iint_D \left(u_x^2 + u_y^2\right) + \left(v_x^2 + v_y^2\right) dx\,dy$$

- Color constancy + small motion error:

$$E_c = \iint_D \left(I_x u + I_y v + I_t\right)^2 dx\,dy$$

- The final energy function:

$$E = E_c + \alpha E_s$$

B. K. P. Horn and B. Schunck, "*Determining Optical Flow*," Artificial Intelligence, 17 (1981)

Horn-Schunck optical flow

- Find an optical flow *field*, that minimizes the error:

$$E = \iint_D \left(I_x u + I_y v + I_t\right)^2 + \alpha\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right) dx\, dy$$

- In LK we were searching for a single vector that minimizes an energy function in a small neighborhood of a pixel.
- But now we are looking for a *function* that minimizes the error over entire image.
- For this we require the *variational calculus*!

A gentle intro to Euler-Lagrange

- For simple intro see [Taylor, J. R., Classical Mechanics, Section 6.2]
- For inspiring motivation, see [Feynman lectures]
- Find a function that minimizes a functional:

$$E = \int_{x_1}^{x_2} f(q(x), q'(x), x)\, dx$$

- The function should satisfy the Euler-Lagrange equation:

$$\frac{\partial f}{\partial q} - \frac{d}{dx}\left(\frac{\partial f}{\partial q'}\right) = 0$$

$$E = \iint_D \left(I_x u + I_y v + I_t\right)^2 + \alpha\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right) dx\, dy$$

Back to Horn-Schunck

- Find an optical flow *field*, that minimizes the energy

$$E = \iint_D \left(I_x u + I_y v + I_t\right)^2 + \alpha\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right) dx\,dy$$

- Recall that all terms depend on pixel position (x, y):

$$u(x,y), v(x,y), u_x(x,y), u_y(x,y), v_x(x,y), v_y(x,y)$$

- From Wikipedia:
    - E-L for several functions of several variables

$$E = \iint L(\mathrm{u}, \mathrm{v}, \mathrm{u}_x, \mathrm{u}_y, \mathrm{v}_x, \mathrm{v}_y, \mathrm{x}, \mathrm{y}) dxdy$$

$$\frac{\partial L}{\partial u} - \frac{d}{dx}\left(\frac{\partial L}{\partial u_x}\right) - \frac{d}{dy}\left(\frac{\partial L}{\partial u_y}\right) = 0$$

$$\frac{\partial L}{\partial v} - \frac{d}{dx}\left(\frac{\partial L}{\partial v_x}\right) - \frac{d}{dy}\left(\frac{\partial L}{\partial v_y}\right) = 0$$

Back to Horn-Schunck

- From $E = \iint_D \left(I_x u + I_y v + I_t\right)^2 + \alpha\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right) dx\, dy$

$$\frac{\partial L}{\partial u} - \frac{d}{dx}\left(\frac{\partial L}{\partial u_x}\right) - \frac{d}{dy}\left(\frac{\partial L}{\partial u_y}\right) = 0 \text{ and } \frac{\partial L}{\partial v} - \frac{d}{dx}\left(\frac{\partial L}{\partial v_x}\right) - \frac{d}{dy}\left(\frac{\partial L}{\partial v_y}\right) = 0$$

Back to Horn-Schunck

- From $E = \iint_D \left(I_x u + I_y v + I_t\right)^2 + \alpha\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right) dx\, dy$

$$\frac{\partial L}{\partial u} - \frac{d}{dx}\left(\frac{\partial L}{\partial u_x}\right) - \frac{d}{dy}\left(\frac{\partial L}{\partial u_y}\right) = 0 \quad \text{and} \quad \frac{\partial L}{\partial v} - \frac{d}{dx}\left(\frac{\partial L}{\partial v_x}\right) - \frac{d}{dy}\left(\frac{\partial L}{\partial v_y}\right) = 0$$

- The following pair of equations *per pixel* emerge

$$I_x(\mathrm{I}_x u + I_y v + I_t) - \alpha(\mathrm{u}_{xx} + \mathrm{u}_{yy}) = 0$$
$$I_y(\mathrm{I}_x u + I_y v + I_t) - \alpha(v_{xx} + v_{yy}) = 0$$

Try deriving this for home exercise!

Note: $u_{xx} = \frac{\partial^2 u}{\partial x^2}$, $u_{yy} = \frac{\partial^2 u}{\partial y^2}$, etc.

The Horn-Schunck equations

- Rewrite these equations

$$I_x(I_x u + I_y v + I_t) - \alpha(u_{xx} + u_{yy}) = 0$$
$$I_y(I_x u + I_y v + I_t) - \alpha(v_{xx} + v_{yy}) = 0$$

by using the definition of the Laplacian operator Δ:

$$\Delta = \begin{bmatrix} \frac{\partial^2}{\partial x^2} \\ \frac{\partial^2}{\partial y^2} \end{bmatrix}$$

$$I_x(I_x u + I_y v + I_t) - \alpha \Delta u = 0$$
$$I_y(I_x u + I_y v + I_t) - \alpha \Delta v = 0$$

Now solve this by discretization of these terms: $I_x, I_y, I_t, \Delta u, \Delta v$!

Discretization: the derivatives

- I_x, I_y, I_t can be estimated in same way as in LK
- But often the following form is used

(you could smooth the images a bit with a Gaussian first)

$$I_x = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} * I(x,y,t)$$

$$I_y = \begin{bmatrix} -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} * I(x,y,t)$$

$$I_t = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix} * \left(I(x,y,t+1) - I(x,y,t)\right)$$

This is temporal derivative with a bit of spatial smoothing.

Discretization: the Laplacian

- Finite difference approximation of $\Delta u = u_{xx} + u_{yy}$

$$\Delta u = u * g$$ ← convolution by

$$g = \begin{bmatrix} 0 & 1/4 & 0 \\ 1/4 & -1 & 1/4 \\ 0 & 1/4 & 0 \end{bmatrix}$$

- Equal to subtracting the value at pixel from the average of the neighbors

$$g = \left[\begin{bmatrix} 0 & 1/4 & 0 \\ 1/4 & 0 & 1/4 \\ 0 & 1/4 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right]$$

$$u * g = \underbrace{u * \begin{bmatrix} 0 & 1/4 & 0 \\ 1/4 & 0 & 1/4 \\ 0 & 1/4 & 0 \end{bmatrix}}_{\text{average of "neighbors"; } \bar{u}} - u$$

So the Laplacian is written as:

$$\Delta u = \bar{u} - u$$

HS optical flow

- Using the definition $\Delta u = \bar{u} - u$, $\Delta v = \bar{v} - v$:

Try deriving this for home exercise!

$$I_x(I_x u + I_y v + I_t) - \alpha \Delta u = 0 \qquad (I_x^{\,2} + \alpha)u + I_x I_y v = \alpha \bar{u} - I_x I_t$$

$$I_y(I_x u + I_y v + I_t) - \alpha \Delta v = 0 \quad \Rightarrow \quad I_x I_y u + (I_y^{\,2} + \alpha)v = \alpha \bar{v} - I_y I_t$$

- And in matrix form (interesting to compare to LK!):

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

HS optical flow

- Solve for [u,v]:

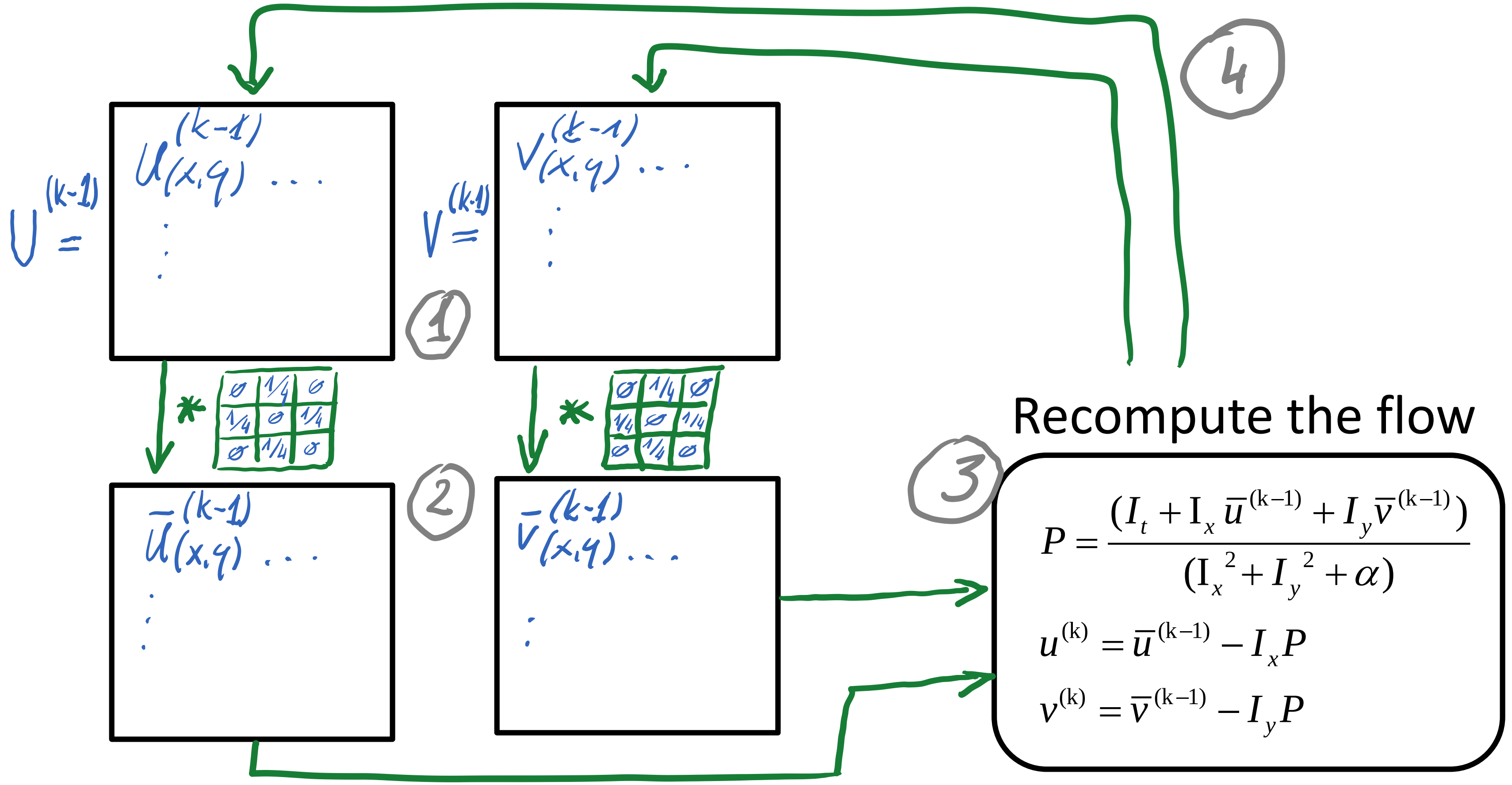
$$\begin{bmatrix} (I_x^2+\alpha) & I_x I_y \\ I_x I_y & (I_y^2+\alpha) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \alpha\bar{u} - I_x I_t \\ \alpha\bar{v} - I_y I_t \end{bmatrix} \longrightarrow \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \bar{u} - I_x P \\ \bar{v} - I_y P \end{bmatrix}, \quad P = \frac{(I_t + \mathrm{I}_x \bar{u} + I_y \bar{v})}{(\mathrm{I}_x^2 + I_y^2 + \alpha)}$$

- Solving simultaneously for all pixels is costly!
- Solve iteratively by Gauss-Siedel-like approach independently for each pixel:

$$\begin{bmatrix} u^{(\mathrm{k})} \\ v^{(\mathrm{k})} \end{bmatrix} = \begin{bmatrix} \bar{u}^{(\mathrm{k}-1)} - I_x P \\ \bar{v}^{(\mathrm{k}-1)} - I_y P \end{bmatrix}, \quad P = \frac{(I_t + \mathrm{I}_x \bar{u}^{(\mathrm{k}-1)} + I_y \bar{v}^{(\mathrm{k}-1)})}{(\mathrm{I}_x^2 + I_y^2 + \alpha)}$$

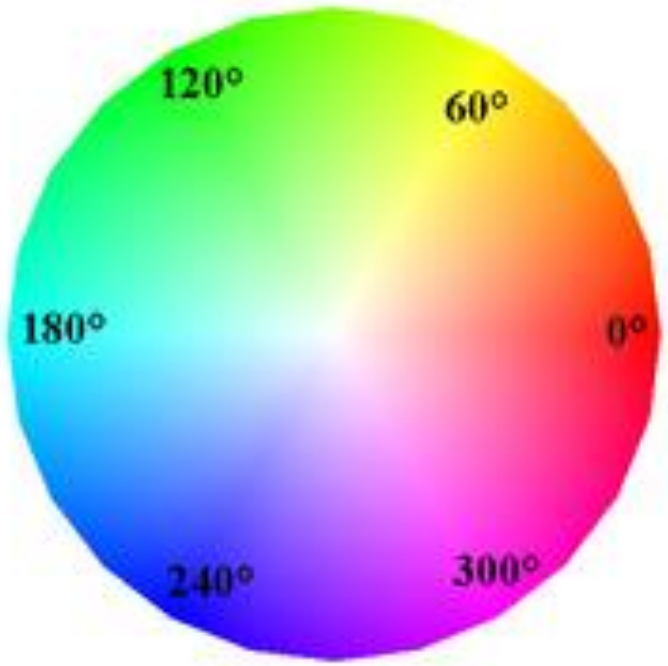
HS flow implementation

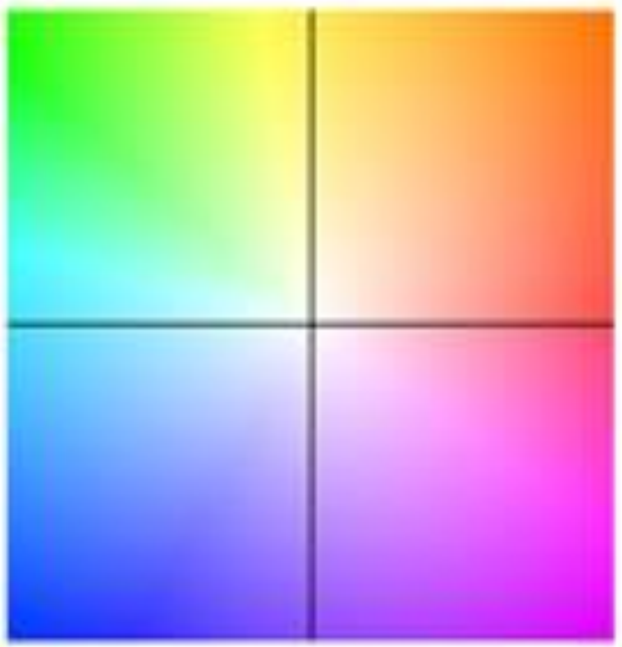
- Initialize U,V to zero, calculate I_x, I_y, I_t in advance
- Repeat the following iterations until "convergence"

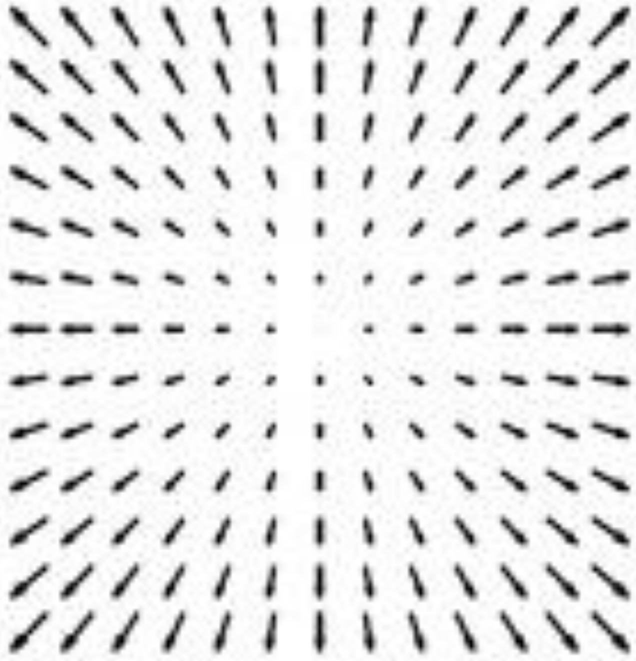


Flow visualization

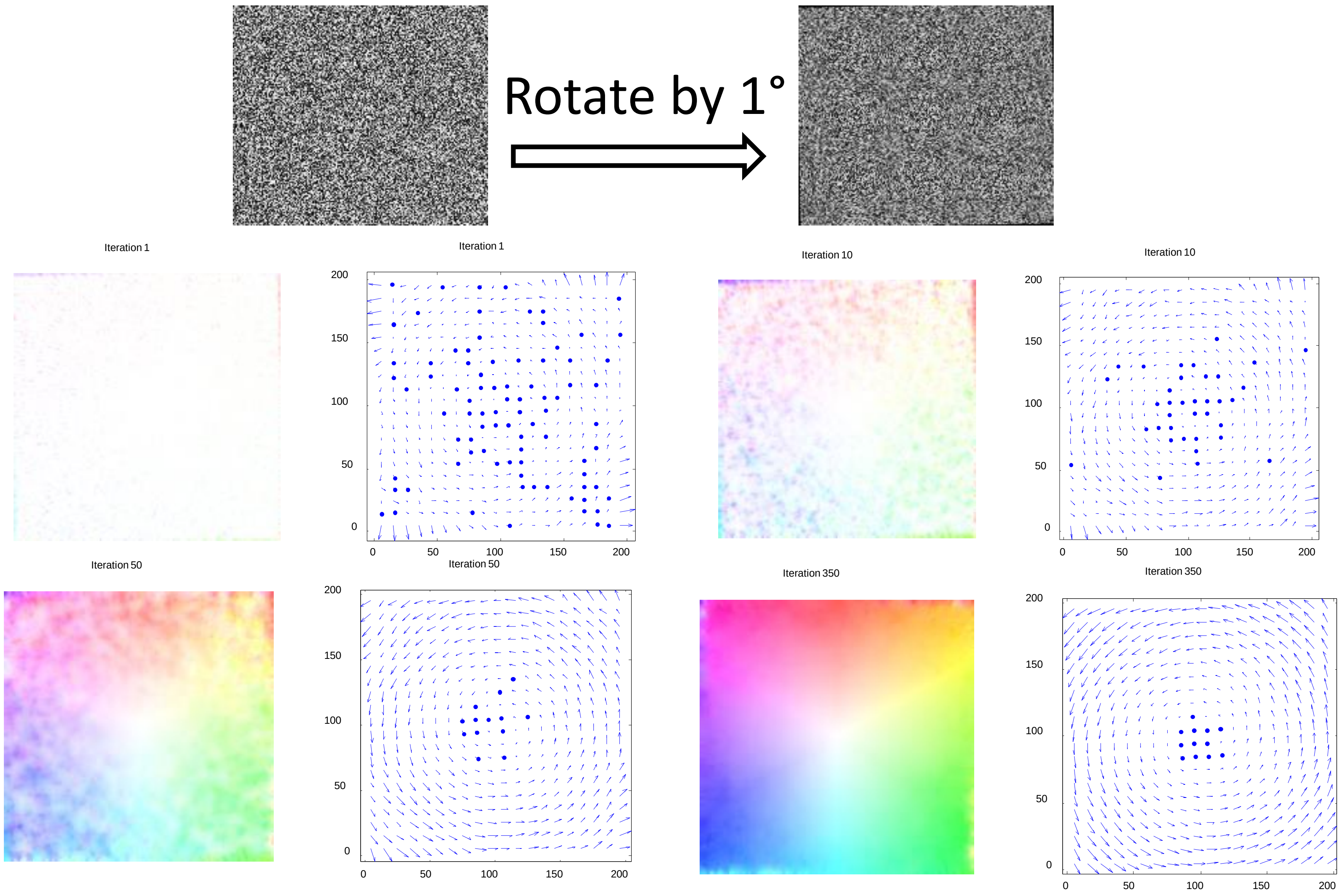
- Angle: Hue
- Magnitude: Saturation





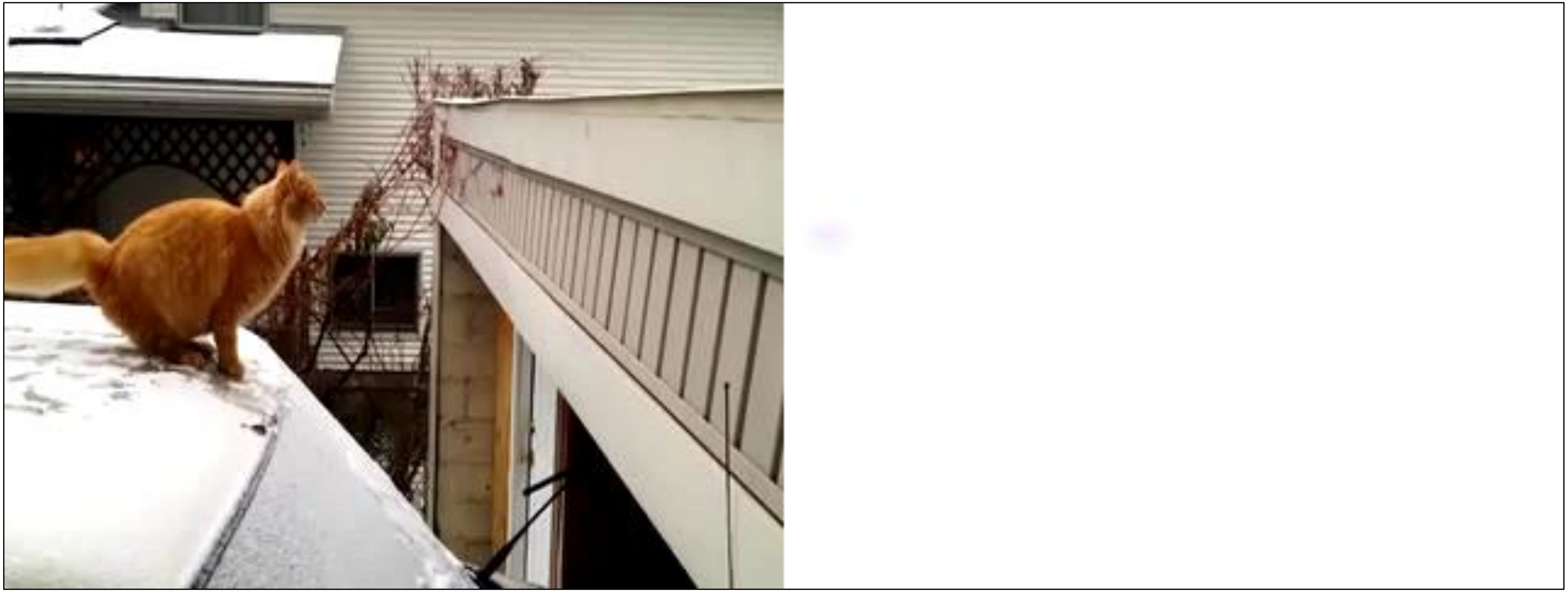


Horn Schunck Iterations



Horn-Schunck example

Back to Waffle the terrible:



Flow in a nutshell

- Brightness constancy assumption: $I(\mathbf{x}) = I(\mathbf{x} + \delta)$
- Small displacement assumption: $I(\mathbf{x} + \delta) \approx I(\mathbf{x}) + \nabla I^T \mathbf{J} \delta$
- Optical flow equation: $I_x(\mathbf{x}_i)\delta_x + I_y(\mathbf{x}_i)\delta_y + I_t(\mathbf{x}_i) = 0$
- Solve the aperture problem:
  - Lucas&Kanade – by least squares
  - Horn&Schunck – by variational calculus
- Worth paying attention
  - Apply pyramids to handle large displacements
  - Apply proper estimates of derivatives
- Robust approaches to handle motion discontinuities

State-of-the-art (from ~10 years ago)

- Not-so SOTA any more:

Baker et al.,A Database and Evaluation Methodology for Optical Flow, IJCV2011

Method	EE		IE		NE	
	Avg	Avg4	Avg	Avg4	Avg	Avg4
Adaptive	4.4	4.5	12.5	11.8	9.8	10.4
Complementary OF	5.7	5.6	12.5	12.4	11.0	9.3
Aniso. Huber-L1	5.8	5.9	4.6	5.4	5.0	5.1
DPOF	6.1	4.2	10.2	9.5	10.9	10.3
TV-L1-improved	7.2	7.4	12.8	9.9	12.7	9.8
CBF	7.8	6.5	3.5	3.1	5.6	4.8
Brox et al.	8.4	8.4	6.3	4.2	7.5	4.8
Rannacher	8.5	9.3	16.0	14.8	14.1	13.2
F-TV-L1	8.8	7.8	7.1	9.6	8.4	9.2
Second-order prior	9.0	9.3	5.5	5.1	5.5	5.1
Fusion	9.4	7.5	10.0	4.9	8.7	6.3
Dynamic MRF	11.1	11.3	14.5	11.8	15.3	11.3
SegOF	11.7	12.8	18.1	17.0	15.3	15.8
Learning Flow	13.3	13.3	15.8	15.5	15.2	15.6
Filter Flow	14.3	11.8	9.7	11.3	11.0	14.0
Graph Cuts	14.5	15.5	13.0	12.1	13.0	11.8
Black & Anandan	15.0	15.4	10.1	14.6	10.1	14.5
SPSA-learn	15.7	17.4	18.0	17.8	19.0	18.4
Group Flow	15.9	18.3	21.1	20.3	19.2	18.8
2D-CLG	17.4	18.8	11.0	11.4	11.6	11.3
Horn & Schunck	18.6	19.3	11.1	14.8	10.4	14.0
TI-DOFE	19.6	20.9	13.5	16.9	12.0	16.1
FOLKI	22.6	22.8	15.9	19.7	18.0	19.8
Pyramid LK	23.7	23.7	22.2	23.4	21.5	23.1

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

Recent SOTA comparison:

The KITTI Vision Benchmark Suite

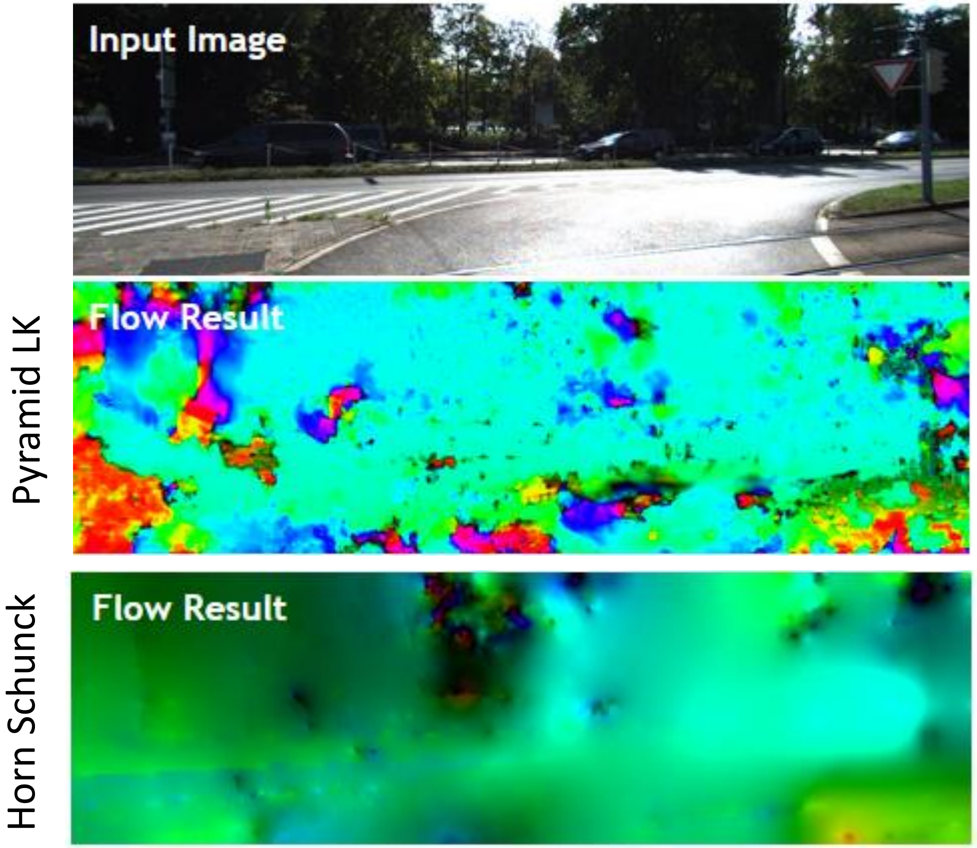
A project of Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago

http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=flow

State-of-the-art

More results on the Kitti benchmark page

- Modern approaches apply convolutional neural networks
- The basic flow equations often used in *unsupervised* learning

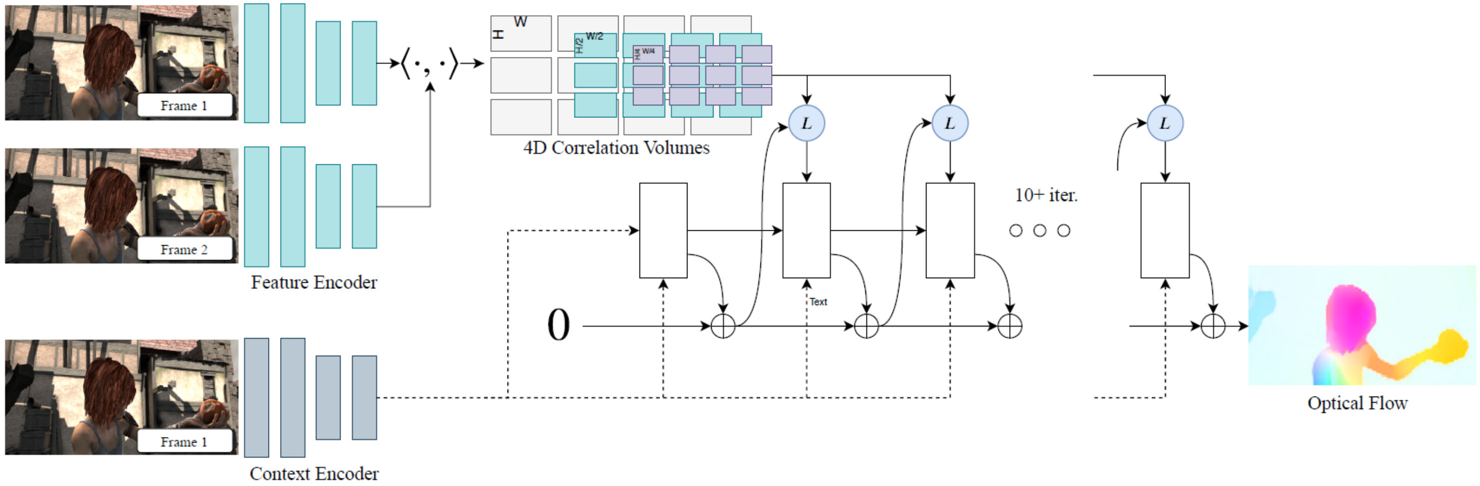


Detail Preserving Propagation for Coarse-to-Fine Matching - Optical Flow Version [PCF-F]

DRISF: Deep rigid instance scene flow [UberATG-DRISF]
(uses stereo – link to paper)

State-of-the-art

- RAFT: iterative refinement (akin to HS idea, but in DL)
  RAFT: Recurrent All Pairs Field Transforms for Optical Flow (code), Best paper award at ECCV 2020



https://www.youtube.com/watch?v=GNwyuhYu7ZI

State-of-the-art

- Like all CNN-based methods, deep flows can be fooled
- Methods being proposed that are resilient to this

Ranjan et al., Attacking Optical Flow, ICCV2019

https://www.youtube.com/watch?v=5nQ7loiPmdA

References

- I recommend to at least superficially read:
  - B. K. P. Horn and B. Schunck, "*Determining Optical Flow*," Artificial Intelligence, 17 (1981), pp. 185-203
- Review of classical stuff:
  - Barron, J.L., Fleet, D.J., and Beauchemin, S. "*Performance of optical flow techniques*". IJCV, 1994, 12(1):43-77
- More recent deep-learning stuff:
  - Dosovitskiy et al., FlowNet: Learning Optical Flow with Convolutional Networks, ICCV2015
  - DRISF: Deep rigid instance scene flow, (link to paper)
  - Papers with code (https://paperswithcode.com/task/scene-flow-estimation/codeless)