Horn-Schunck and Lucas Kanade[1]

Lecture 8

See Sections 4.2 and 4.3 in
Reinhard Klette: Concise Computer Vision
Springer-Verlag, London, 2014

[1] See last slide for copyright information.

Where We Stopped Last Time

LSE optimisation problem defined by those two error terms

$$E_{data}(f) = \sum_{\Omega} [\ u \cdot I_x + v \cdot I_y + I_t\ ]^2$$

$$E_{smooth}(f) = \sum_{\Omega} \quad (\quad u_{x+1,y} - u_{xy})^2 + (u_{x,y+1} - u_{xy})^2 + (v_{x+1,y} - v_{xy})^2 + (v_{x,y+1} - v_{xy})^2$$

Calculate labeling function f which minimizes

$$E_{total}(f) = E_{data}(f) + \ \lambda \cdot E_{smooth}(f)$$

Least-Square Error Optimization

Least-square error (LSE) *optimization* follows a standard scheme:

1. Define an error or energy function.
2. Calculate derivatives of this function with respect to all the unknown parameters.
3. Set derivatives equal to zero and solve this equational system with respect to the unknowns. The result defines a minimum of the error function.

Agenda

Derivatives

We exclude t from the following formulas; we simply use $I(x, y)$

$$\frac{\partial E_{data}}{\partial u_{xy}}(u, v) = 2\left[I_x(x, y)\, u_{xy} + I_y(x, y)\, v_{xy} + I_t(x, y)\right] I_x(x, y)$$

$$\frac{\partial E_{data}}{\partial v_{xy}}(u, v) = 2\left[I_x(x, y)\, u_{xy} + I_y(x, y)\, v_{xy} + I_t(x, y)\right] I_y(x, y)$$

$$\begin{aligned}
\frac{\partial E_{smooth}}{\partial u_{xy}}(u, v) &= -2\left[(u_{x+1,y} - u_{xy}) + (u_{x,y+1} - u_{xy})\right] \\
&\quad + 2\left[(u_{xy} - u_{x-1,y}) + (u_{xy} - u_{x,y-1})\right] \\
&= 2\left[(u_{xy} - u_{x+1,y}) + (u_{xy} - u_{x,y+1})\right. \\
&\quad \left. + (u_{xy} - u_{x-1,y}) + (u_{xy} - u_{x,y-1})\right]
\end{aligned}$$

Simplifications

$$\frac{1}{4} \cdot \frac{\partial E_{smooth}}{\partial u_{xy}}(u, v) = 2\left[u_{xy} - \left[\frac{1}{4}\left(u_{i+1,j} + u_{x,y+1} + u_{i-1,j} + u_{x,y-1}\right)\right]\right]$$

Let $\bar{u}_{xy}$ be the mean value of 4-adjacent pixels

$$\begin{aligned} \frac{1}{4}\frac{\partial E_{smooth}}{\partial u_{xy}}(u, v) &= 2\left[u_{xy} - \bar{u}_{xy}\right] \\ \frac{1}{4}\frac{\partial E_{smooth}}{\partial v_{xy}}(u, v) &= 2\left[v_{xy} - \bar{v}_{xy}\right] \end{aligned}$$

Use λ instead of $\lambda/4$

Equational System

Setting derivatives equal to zero:

$$\begin{aligned} 0 = \lambda \quad & [u_{xy} - \bar{u}_{xy}] \\ & + [I_x(x,y)\, u_{xy} + I_y(x,y)\, v_{xy} + I_t(x,y)]\, I_x(x,y) \\ 0 = \lambda \quad & [v_{xy} - \bar{v}_{xy}] \\ & + [I_x(x,y)\, u_{xy} + I_y(x,y)\, v_{xy} + I_t(x,y)]\, I_y(x,y) \end{aligned}$$

A linear equational system for $2N_{cols}N_{rows}$ unknowns u_{xy} and v_{xy}

Iterative Solution Scheme

An example of a *Jacobi method*, starting with some initial values:

1. *Initialization step:* Values u_{xy}^0 and v_{xy}^0
2. *Iteration Step 0:* Calculate means $\bar{u}_{xy}^0$ and $\bar{v}_{xy}^0$ and values u_{xy}^1 and v_{xy}^1
3. *Iteration Step n:* Use values u_{xy}^n and v_{xy}^n to compute means $\bar{u}_{xy}^n$ and $\bar{v}_{xy}^n$; use those to calculate values u_{xy}^{n+1} and v_{xy}^{n+1}

Proceed for $n \geq 1$ until a stop criterion is satisfied

Solution

The solution is as follows:

$$u_{xy}^{n+1} = \bar{u}_{xy}^{n} - I_x(x,y) \cdot \frac{I_x(x,y)\bar{u}_{xy}^{n} + I_y(x,y)\bar{v}_{xy}^{n} + I_t(x,y)}{\lambda^2 + I_x^2(x,y) + I_y^2(x,y)}$$

$$v_{xy}^{n+1} = \bar{v}_{xy}^{n} - I_y(x,y) \cdot \frac{I_x(x,y)\bar{u}_{xy}^{n} + I_y(x,y)\bar{v}_{xy}^{n} + I_t(x,y)}{\lambda^2 + I_x^2(x,y) + I_y^2(x,y)}$$

Pseudocode Horn-Schunck 1

Use "odd" and "even" arrays for u- and v-values

1: **for** $y = 1$ to N_{rows} **do**
2: **for** $x = 1$ to N_{cols} **do**
3: Compute $I_x(x, y)$, $I_y(x, y)$, and $I_t(x, y)$;
4: Initialize $u(x, y)$ and $v(x, y)$ (in even arrays);
5: **end for**
6: **end for**

$\bar{u}$ and $\bar{v}$ denote means at 4-adjacent pixels

$$\alpha(x, y, n) = \frac{I_x(x, y)\,\bar{u}^n_{xy} + I_y(x, y)\,\bar{v}^n_{xy} + I_t(x, y)}{\lambda^2 + I_x^2(x, y) + I_y^2(x, y)}$$

Pseudocode Horn-Schunck 2

Threshold T is the maximum number of iterations (e.g. $T = 7$)

1: Select weight factor λ; select $T > 1$; set $n = 1$;
2: **while** $n \leq T$ **do**
3: **for** $y = 1$ to N_{rows} **do**
4: **for** $x = 1$ to N_{cols} {in alternation for even or odd arrays} **do**
5: Compute $\alpha(x, y, n)$;
6: Compute $u(x, y) = \bar{u} - \alpha(x, y, n) \cdot I_x(x, y, t)$;
7: Compute $v(x, y) = \bar{v} - \alpha(x, y, n) \cdot I_y(x, y, t)$;
8: **end for**
9: **end for**
10: $n := n + 1$;
11: **end while**

Agenda

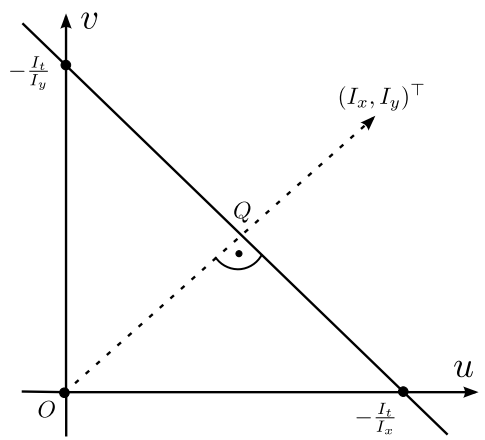
Possible Initializations

1. Initialization with value 0 at all positions of u_{xy} and v_{xy}

Suggested for the original Horn-Schunck algorithm

We have non-zero values u^1_{xy} and v^1_{xy} if $I_x(x,y) \cdot I_t(x,y) \neq 0$ and $I_y(x,y) \cdot I_t(x,y) \neq 0$

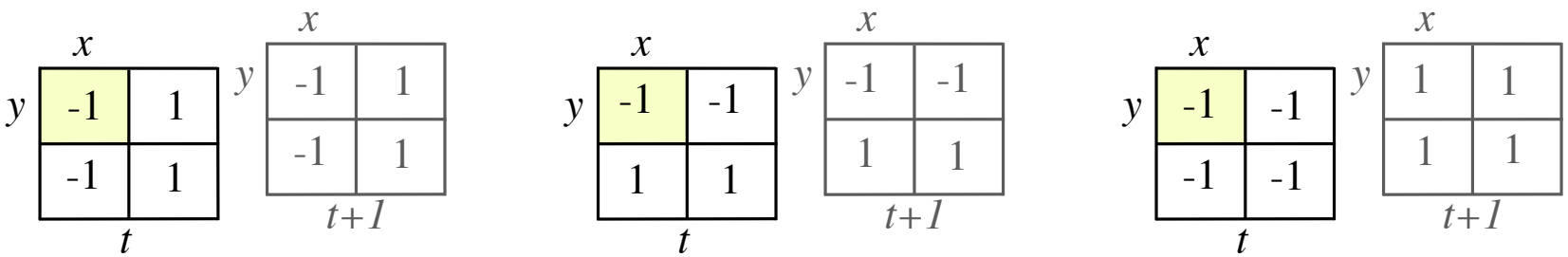
2. Initialization with point Q on straight line in uv-space, at all positions of u_{xy} and v_{xy}



Approximations for I_x, I_y, and I_t

$$
\begin{aligned}
I_x(x,y,t) &= \frac{1}{4}\begin{bmatrix} I(x+1,y,t)+I(x+1,y,t+1) \\ +I(x+1,y+1,t)+I(x+1,y+1,t+1) \end{bmatrix} \\
&\quad -\frac{1}{4}\begin{bmatrix} I(x,y,t)+I(x,y,t+1) \\ +I(x,y+1,t)+I(x,y+1,t+1) \end{bmatrix} \\
I_y(x,y,t) &= \frac{1}{4}\begin{bmatrix} I(x,y+1,t)+I(x,y+1,t+1) \\ +I(x+1,y+1,t)+I(x+1,y+1,t+1) \end{bmatrix} \\
&\quad -\frac{1}{4}\begin{bmatrix} I(x,y,t)+I(x,y,t+1) \\ +I(x+1,y,t)+I(x+1,y,t+1) \end{bmatrix} \\
I_t(x,y,t) &= \frac{1}{4}\begin{bmatrix} I(x,y,t+1)+I(x,y+1,t+1) \\ +I(x+1,y,t+1)+I(x+1,y+1,t+1) \end{bmatrix} \\
&\quad -\frac{1}{4}\begin{bmatrix} I(x,y,t)+I(x,y+1,t) \\ +I(x+1,y,t)+I(x+1,y+1,t) \end{bmatrix}
\end{aligned}
$$

Masks for Approximations for I_x, I_y, and I_t



As in original Horn-Schunck algorithm; many more options

Pyramidal Horn-Schunck Algorithm

Use a regular image pyramid for input frames $I(.,.,t)$

Processing starts at a selected level (of lower resolution)

Obtained results are used for initializing optic flow values at a lower level (of higher resolution)

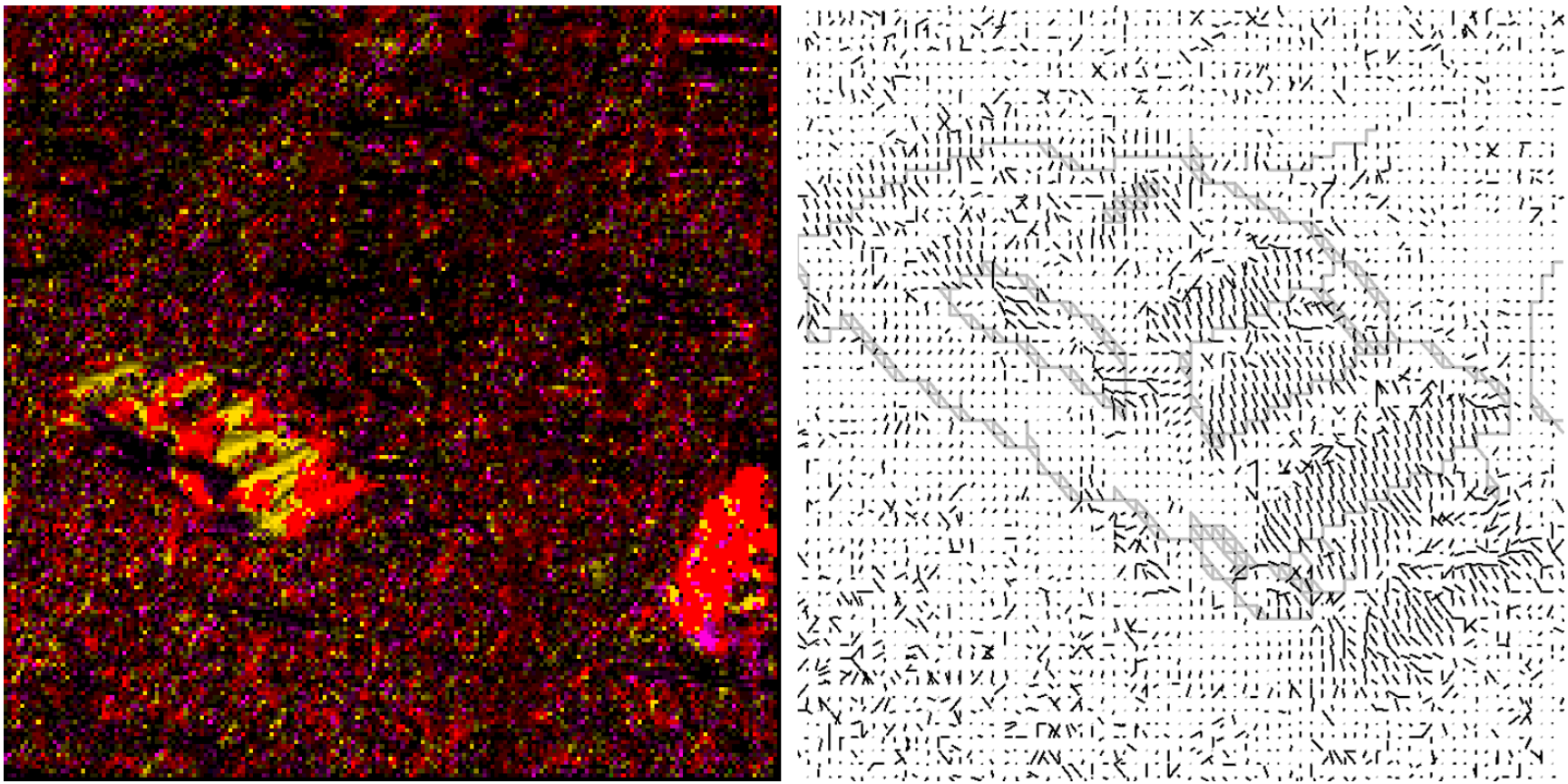
Repeat until full resolution level of original frames is reached

Agenda

Test Data for Comparative Performance Analysis in 1983

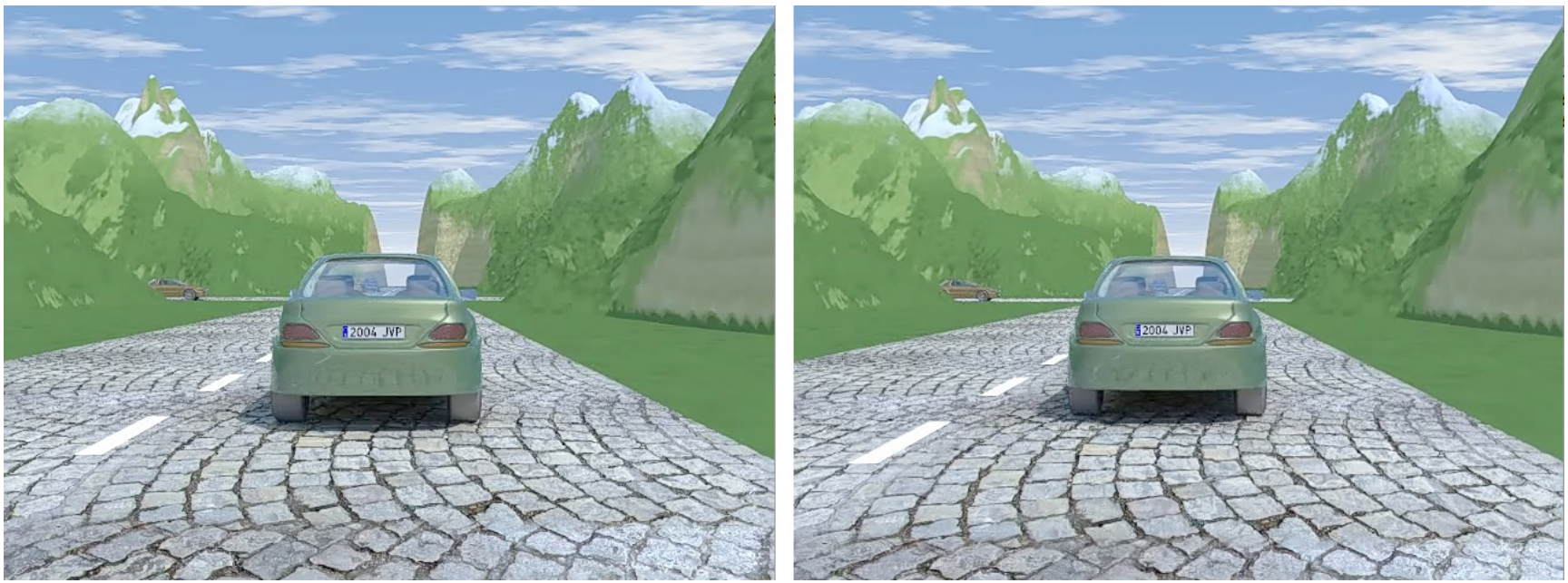
Two subsequent frames of the "Hamburg taxi sequence" (6 frames in total), published in 1983

Visualization of Results in 1995



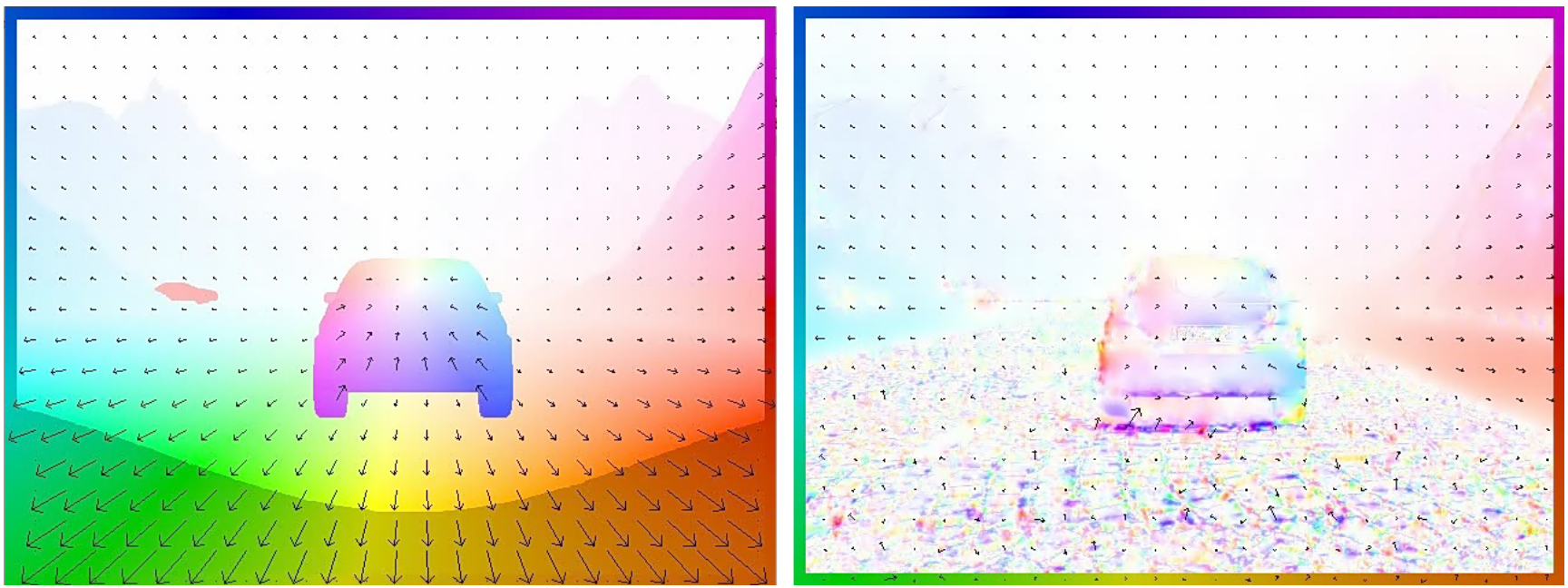
Use of color key or shown as a *needle map*

Test Data for Comparative Performance Analysis in 2007



Two subsequent frames of the first sequence (100 frames long) of Set 2 of EISATS, published online in 2007; synthetic data provide *ground truth* of exact local displacements

Visualization of Results in 2008



Ground truth and result of a pyramidal Horn-Schunck algorithm

Ground Truth

Images recorded in an airplane were and are often used to estimate distances on the ground, or even for 3D reconstruction of whole landscapes or cities.

For evaluating results it was common practice to identify landmarks *on the ground*, such as corners of buildings, and to measure distances or positions of those landmarks.

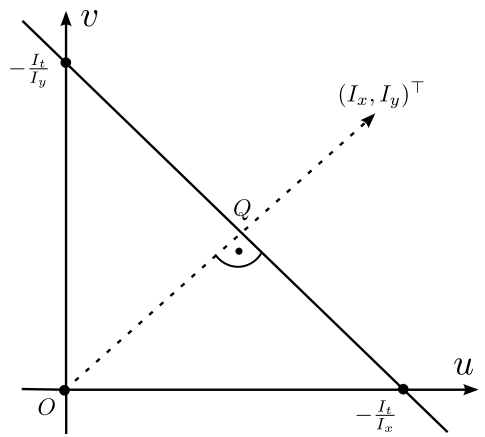
This was the *ground truth*, to be compared with the values calculated based on the images recorded in an airplane.

The term is now in general use for denoting *measured data*, considered to be fairly accurate, thus useful for evaluating algorithms supposed to provide the same data.

Agenda

Optic-Flow Line

We identify point Q on the optic-flow line in uv-space



$I_t = u \cdot I_x + v \cdot I_y$ intersects axes at $(-I_t/I_x, 0)$ and $(0, -I_t/I_y)$

Vector on this line: $[-I_t/I_x, 0]^\top - [0, -I_t/I_y]^\top = [-I_t/I_x, I_t/I_y]^\top$

Orthogonal Vector **a**

Vector $\mathbf{a} = [a_x, a_y]^\top$ from O to Q is orthogonal to this line:

$$\mathbf{a} \cdot [-I_t/I_x, I_t/I_y]^\top = 0$$

Thus

$$a_x \cdot (-I_t/I_x) + a_y \cdot (I_t/I_y) = 0$$

and

$$I_t(a_x \cdot I_y) = I_t(a_y \cdot I_x)$$

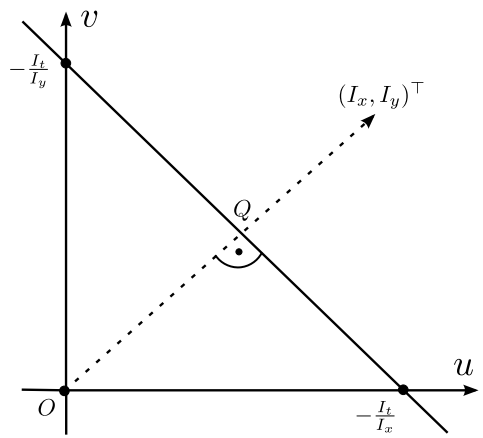
Assuming $I_t \neq 0$ at considered $p = (x, y)$, we have $a_x I_y = a_y I_x$ or

$$\mathbf{a} = c \cdot \mathbf{g}^\circ$$

for some $c \neq 0$, where $\mathbf{g}^\circ$ denotes the unit vector of $\mathbf{g} = [I_x, I_y]^\top$

Point Q Identifies Gradient Flow

Thus: vector **a** is a multiple of the gradient **g**



In other words: Point Q identifies gradient flow

Unit Vector

Vector $\mathbf{a} = [a_1, a_2, \ldots, a_n]^\top$

Magnitude of the vector equals $||\mathbf{a}||_2 = \sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}$

Unit vector

$$\mathbf{a}^\circ = \frac{\mathbf{a}}{||\mathbf{a}||_2}$$

is of magnitude 1 and specifies the direction of vector $\mathbf{a}$

Product $\mathbf{a}^\circ \cdot \mathbf{a}^\circ = ||\mathbf{a}^\circ||_2 \cdot ||\mathbf{a}^\circ||_2 \cdot \cos 0 = 1$

What Constant c?

Vector $\mathbf{a} = c \cdot \mathbf{g}^{\circ}$ satisfies the optic flow equation $\mathbf{u} \cdot \mathbf{g} = -I_t$

$$c \cdot \mathbf{g}^{\circ} \cdot \mathbf{g} = c \cdot ||\mathbf{g}||_2 = -I_t$$

Thus we have c and altogether

$$\mathbf{a} = -\frac{I_t}{||\mathbf{g}||_2} \mathbf{g}^{\circ}$$

as vector representation of point Q

We can use point Q (i.e. gradient flow) for initializing the u and v arrays prior to the iteration

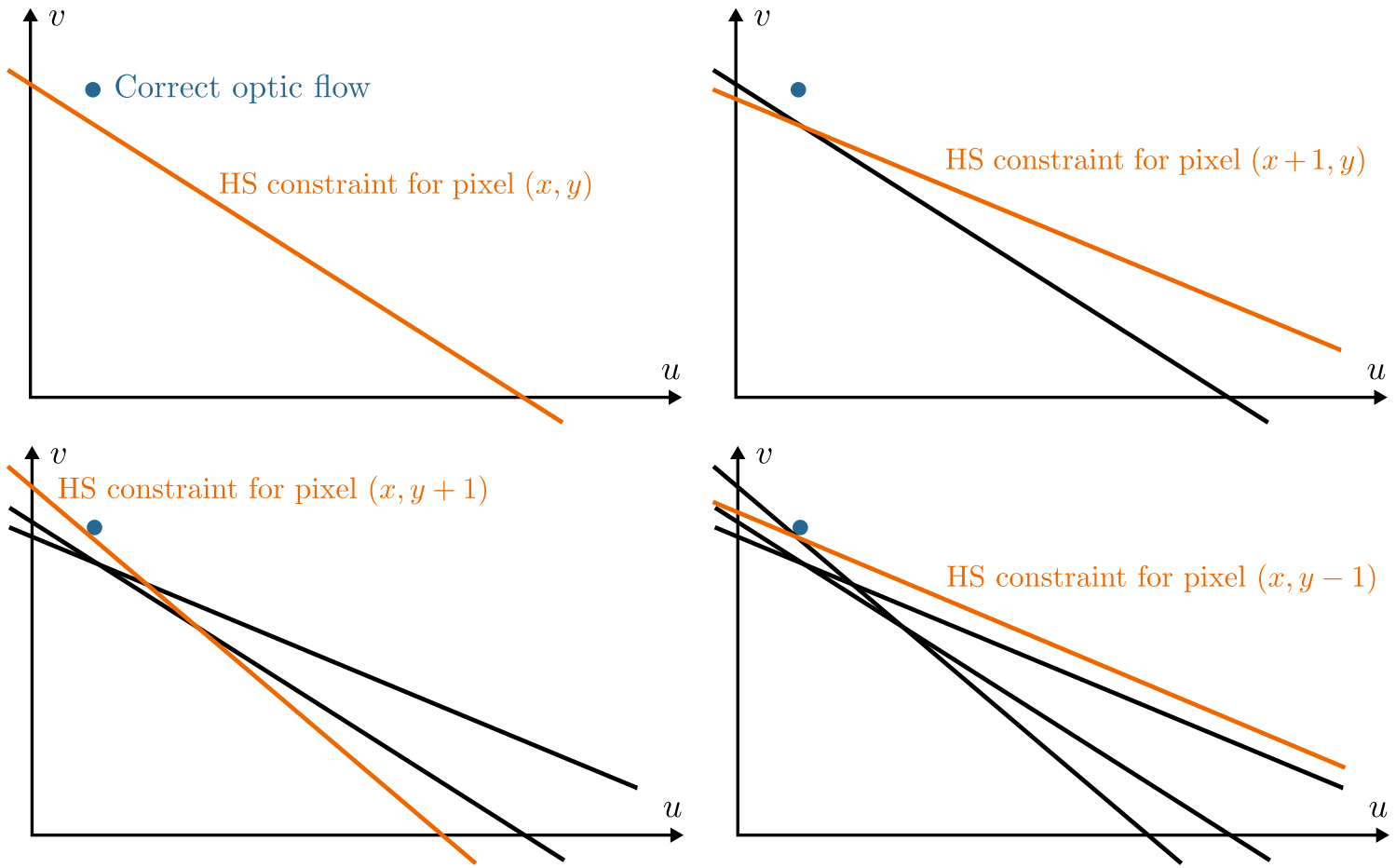
Agenda

Lucas-Kanade Algorithm

Optic flow equation specifies line $u \cdot I_x + v \cdot I_y + I_t = 0$ for $p \in \Omega$

Consider straight lines defined by pixels in a neighborhood

Assume: not parallel, defined by about the same motion



Method of Linear Least Squares

This method is applied in cases of overdetermined equational systems.

The task is to find a solution which minimizes the sum of squared errors (called *residuals*) caused by this solution, for each of the equations.

For example, we only have n unknowns, but $m > n$ linear equations for those; in this case we use a *linear least-squares method*.

Simple Case: Two Lines

Pixel locations p_1 and adjacent p_2
Assume that 2D motion $\mathbf{u}$ is identical at p_1 and p_2
But different unit gradients $\mathbf{g}_1^\circ = (g_{x1}, g_{y1})^\top$ and $\mathbf{g}_2^\circ = (g_{x2}, g_{y2})^\top$
Optic flow equation $\mathbf{u}^\top \cdot \mathbf{g}^\circ = -\frac{I_t}{||\mathbf{g}||_2}$ at both pixels:

$$\begin{aligned} \mathbf{u}^\top \cdot \mathbf{g}_1^\circ(p_1) &= -\frac{I_t}{||\mathbf{g}||_2}(p_1) \\ \mathbf{u}^\top \cdot \mathbf{g}_2^\circ(p_2) &= -\frac{I_t}{||\mathbf{g}||_2}(p_2) \end{aligned}$$

Using b_i on the right-hand side:

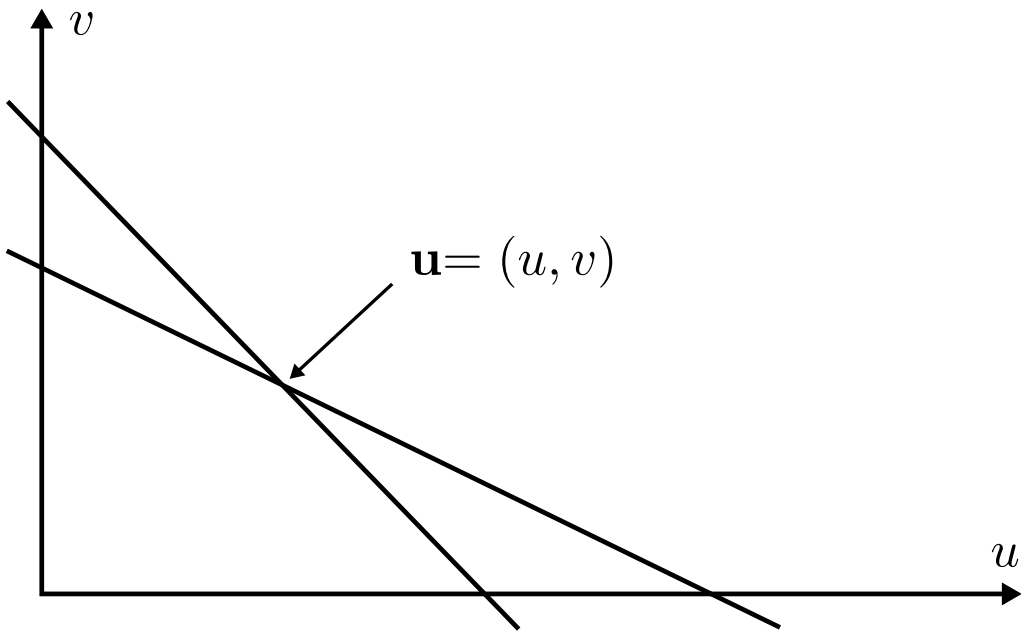
$$\begin{aligned} u g_{x1} + v g_{y1} &= b_1 \\ u g_{x2} + v g_{y2} &= b_2 \end{aligned}$$

Matrix Form

$$\begin{bmatrix} g_{x1} & g_{y1} \\ g_{x2} & g_{y2} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Solvable if matrix on the left is invertible (i.e., non-singular):

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} g_{x1} & g_{y1} \\ g_{x2} & g_{y2} \end{bmatrix}^{-1} \cdot \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$



More Pixels in a Neighborhood

There are errors involved when estimating I_x, I_y, and I_t
Image data are noisy anyway

$$\begin{bmatrix} g_{x1} & g_{y1} \\ g_{x2} & g_{y2} \\ \vdots & \vdots \\ g_{xk} & g_{yk} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}$$

Write as

$$\underbrace{\mathbf{G}}_{k\times 2}\underbrace{\mathbf{u}}_{2\times 1} = \underbrace{\mathbf{B}}_{k\times 1}$$

Solve for $k \geq 2$ in the least-square error sense:

$$\mathbf{G}^\top \mathbf{G}\mathbf{u} = \mathbf{G}^\top \mathbf{B}$$

$$\mathbf{u} = (\mathbf{G}^\top \mathbf{G})^{-1}\mathbf{G}^\top \mathbf{B}$$

Done. $\mathbf{G}^\top \mathbf{G}$ is a 2×2 matrix while $\mathbf{G}^\top \mathbf{B}$ is a 2×1 matrix

Example

Let

$$\mathbf{G}^\top \mathbf{G} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

then

$$(\mathbf{G}^\top \mathbf{G})^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

The rest is simple matrix multiplication

Lucas-Kanade Algorithm

1. Decide for local neighborhood of k pixels and apply uniformly in frames
2. At frame t, estimate I_x, I_y and I_t
3. For each p in Frame t, obtain the equational system and solve it in the least-squares sense

Possibly smooth frames first, e.g. with Gaussian filter with a small standard deviation such as $\sigma = 1.5$

Weights for Contributing Pixels

Weight all the k contributing pixels by positive weights w_i
Current pixel has the maximum weight

$\mathbf{W} = \text{diag}[w_1, ..., w_k]$ a $k \times k$ diagonal matrix of weights

$$\mathbf{W}^\top \mathbf{W} = \mathbf{W}\mathbf{W} = \mathbf{W}^2$$

Task: solve the equation

$$\begin{aligned}
\mathbf{WGu} &= \mathbf{WB} \\
(\mathbf{WG})^\top \mathbf{WGu} &= (\mathbf{WG})^\top \mathbf{WB} \\
\mathbf{G}^\top \mathbf{W}^\top \mathbf{WGu} &= \mathbf{G}^\top \mathbf{W}^\top \mathbf{WB} \\
\mathbf{G}^\top \mathbf{WWGu} &= \mathbf{G}^\top \mathbf{WWB} \\
\mathbf{G}^\top \mathbf{W}^2 \mathbf{Gu} &= \mathbf{G}^\top \mathbf{W}^2 \mathbf{B}
\end{aligned}$$

Solution with Weights

$$\mathbf{u} = [\mathbf{G}^\top \mathbf{W}^2 \mathbf{G}]^{-1} \mathbf{G}^\top \mathbf{W}^2 \mathbf{B}$$

We only accept solutions for cases where eigenvalues of matrix $\mathbf{G}^\top \mathbf{G}$ (unweighted case) or $\mathbf{G}^\top \mathbf{W}^2 \mathbf{G}$ (weighted case) are greater than a chosen threshold, for filtering out "noisy" results

Example

Optic flow calculated with original Lucas-Kanade algorithm;
$k = 25$ for a 5×5 neighborhood

Copyright Information

This slide show was prepared by Reinhard Klette
with kind permission from Springer Science+Business Media B.V.

The slide show can be used freely for presentations.
However, *all the material* is copyrighted.

In case of citation: just cite the book, that's fine.