Pixel Labeling: Optic Flow[1]

Lectures 19 and 20

See Material in
Reinhard Klette: Concise Computer Vision
Springer-Verlag, London, 2014

ccv.wordpress.fos.auckland.ac.nz

[1] See last slide for copyright information.

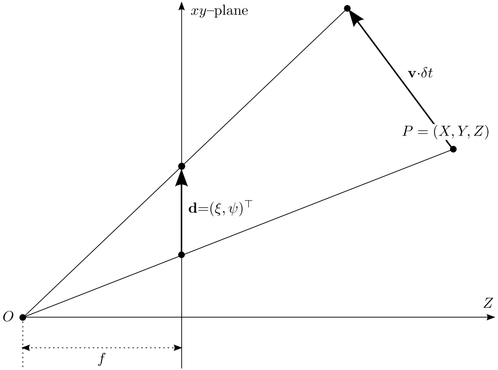
Agenda

Projected Motion

$P = (X, Y, Z)$ projected at $t \cdot \delta t$ into $p = (x, y)$ in $I(.,.,t)$

Camera: *focal length* f, *projection centre* O, looks along *optic axis*

Ideal model defines *central projection* into xy image plane



Projection of motion $\mathbf{v} \cdot \delta t$ into displacement $\mathbf{d}$ in the image plane

2D Motion

Assumptions: motion of P between $t \cdot \delta t$ and $(t+1) \cdot \delta t$

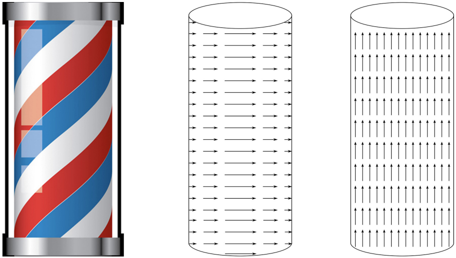
1. linear
2. with constant speed

Local displacement:
Projection $\mathbf{d} = (\xi, \psi)^\top$ of this 3D motion

Visible displacement:
The *optic flow* $\mathbf{u} = [u, v]^\top$ from $p = (x, y)$ to $p = (x + u, y + v)$

Often: optic flow not identical to local displacement

2D Motion $\neq$ Optic Flow



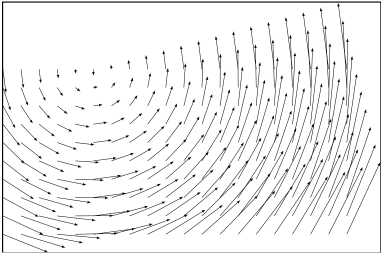
Rotating barber's pole: sketch of 2D motion (without scaling vectors) and sketch of optic flow, an optical illusion

Lambertian sphere: rotation but not visible

Moving light source: "textured" static object and a moving light source (e.g., the sun) generate optic flow

Vector Fields

Rotating rectangle: around a fixpoint, parallel to the image plane:



Motion maps: vectors start at time t and end at time $t + 1$

To be visualized by using a color key

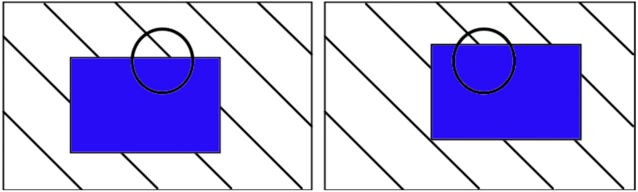
Dense if vectors at (nearly) all pixel locations; otherwise *sparse*

Difficult to infer the shape of a polyhedron from a motion map

Agenda

Aperture Defined by Available Window

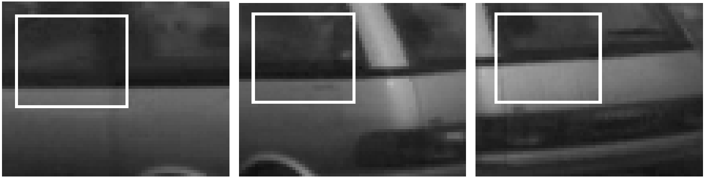
Sitting in a waiting train and assuming to move because the train on the next track started to move



A program only "sees" both circular windows at time t (*left*) and time $t+1$ (*right*); it concludes an upward shift and misses the shift diagonally towards the upper right corner

Camera Aperture

Visible motion defined by the aperture of the camera



Images taken at times t, $t+1$, and $t+2$

Inner rectangles: we conclude an upward translation with minor rotation

Three images: indicate a motion of this car to the left

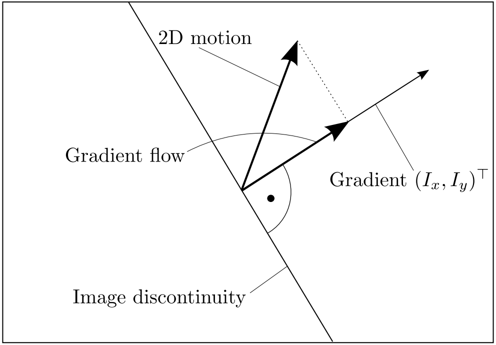
Ground truth: car is actually driving around a roundabout

Gradient Flow

Due to aperture problem: local optic flow detects *gradient flow*

$$\text{2D gradient} \qquad \nabla_{x,y} I = (I_x(x,y,t), I_y(x,y,t))^\top$$

I_x and I_y are partial derivatives of $I(.,.,t)$ w.r.t. x and y



True 2D motion **d**: diagonally up

Identified motion: projection of **d** onto gradient vector

Agenda

Frames in a Video Sequence and Optic Flow

We consider a sequence of scalar images, also called *frames*

Time difference δt between two subsequent time slots

$I(.,.,t)$ is the frame at time slot t with values $I(x,y,t)$

Example: $\delta t = 1/30$ s means 30 Hz (read: "hertz") or

30 fps (read: "frames per second") or 30 pps (read: "pictures per second")

The *optic flow* $\mathbf{u}(x,y) = (u(x,y), v(x,y))$

is the visible motion of a pixel at (x,y) into a pixel at $(x+u(x,y), y+v(x,y))$ between two subsequent frames

The Horn-Schunck Algorithm

Taylor expansion for frame sequence:

$$I(x + \delta x, y + \delta y, t + \delta t)$$

$$= I\left(x, y, t\right) + \delta x \cdot \frac{\partial I}{\partial x}\left(x, y, t\right) + \delta y \cdot \frac{\partial I}{\partial y}\left(x, y, t\right)$$

$$+ \delta t \cdot \frac{\partial I}{\partial t}\left(x, y, t\right) + e$$

Assumption 1.
Let $e = 0$, i.e. $I(.,.,.)$ linear for *small* values of δx, δy, and δt

Assumption 2.
δx and δy model the motion u and v of one pixel between t and $t + 1$

Assumption 3.
Intensity constancy assumption (ICA)

$I\left(x + \delta x, y + \delta y, t + \delta t\right) = I\left(x, y, t\right)$

Horn-Schunck Constraint or Optic Flow Equation

$$0 = \frac{\delta x}{\delta t} \cdot \frac{\partial I}{\partial x}(x, y, t) + \frac{\delta y}{\delta t} \cdot \frac{\partial I}{\partial y}(x, y, t) + \frac{\partial I}{\partial t}(x, y, t)$$

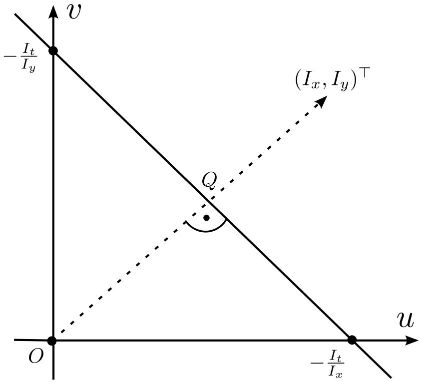
Changes in x- and y-coordinate during δt as optic flow

$$0 = u(x, y, t) \cdot \frac{\partial I}{\partial x}(x, y, t) + v(x, y, t) \cdot \frac{\partial I}{\partial y}(x, y, t) + \frac{\partial I}{\partial t}(x, y, t)$$

Short form:

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

The *uv* Velocity Space



Straight line

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

in *uv* velocity space, with optic flow vector $\mathbf{u} = [u, v]^\top$

Agenda

Labeling Model, Constraints, and an MRF

Labeling function f assigns *label* (u, v) to $p \in \Omega$ in $I(.,.,t)$

Possible set of vectors $(u, v) \in \mathbb{R}^2$ defines the set of labels

Data error or *data energy*

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

Smoothness error or *smoothness energy*

$$E_{smooth}(f) = \sum_{\Omega}\ u_x^2 + u_y^2 + v_x^2 + v_y^2$$

where u_x is the 1^{st} order derivative of u with respect to x, and so forth

Derivatives define dependencies between adjacent pixels: MRF again

The Optimization Problem

Task: Calculate labelling function f which minimizes

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

where $\lambda > 0$ is a weight, e.g. $\lambda = 0.1$

Characterization: *Total variation* (TV)

Search for an optimum f in the set of all possible labelings
We apply L_2-penalties for error terms, thus TVL_2 optimization

Applied solution strategy: *least-square error* (LSE) *optimization*

1. Define an error or energy function. – DONE
2. Calculate derivatives of this function with respect to all the unknown parameters. – NEXT ON OUR LIST
3. Set derivatives equal to zero and solve equational system with respect to the unknowns. Result defines minimum of the error function.

E_{smooth} Defines Underlying MRF Graph: 4-adjacency

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

We exclude t from the formulas, $u_{x,y} = u(x,y)$ and $v_{x,y} = v(x,y)$

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

Derivatives $= 0$, with $\bar{u}_{xy}$ or $\bar{v}_{xy}$ for mean value of 4-adjacent pixels:

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

We solve for $2 N_{cols} N_{rows}$ unknowns u_{xy} and v_{xy}

Iterative Solution Scheme

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

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

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

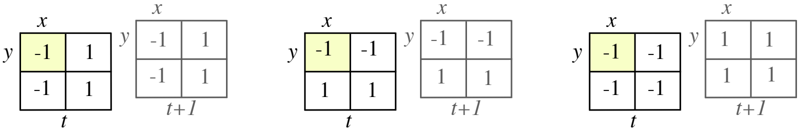
1. *Initialization step:* Values u_{xy}^0 and v_{xy}^0
2. *Iteration Step 0:* Calculate means $\bar{u}_{xy}^0$ and $\bar{v}_{xy}^0$ and values u_{xy}^1 and v_{xy}^1
3. *Iteration Step n:* Use values u_{xy}^n and v_{xy}^n to compute means $\bar{u}_{xy}^n$ and $\bar{v}_{xy}^n$; use those to calculate values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

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

Horn-Schunck Algorithm

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

2. Masks for Approximations for I_x, I_y, and I_t



3. Pyramidal Horn-Schunck algorithm

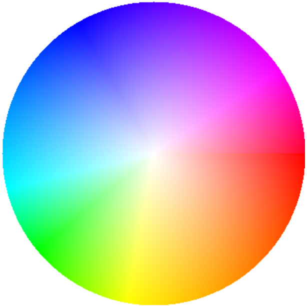
Use a regular image pyramid for input frames $I(.,.,t)$

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

Color Key for Visualising Optic Flow



Colors represent direction of vector $\mathbf{u}$ (start at the center of the disk)

Saturation represents magnitude of the vector, with White for “no motion”

Example 1 for Horn-Schunck-Algorithm

Subsequent frames taken at 25 fps

Color-coded motion field calculated with basic Horn-Schunck algorithm

Shown sparse (magnified) vectors are redundant information

Example 2 for Horn-Schunck-Algorithm

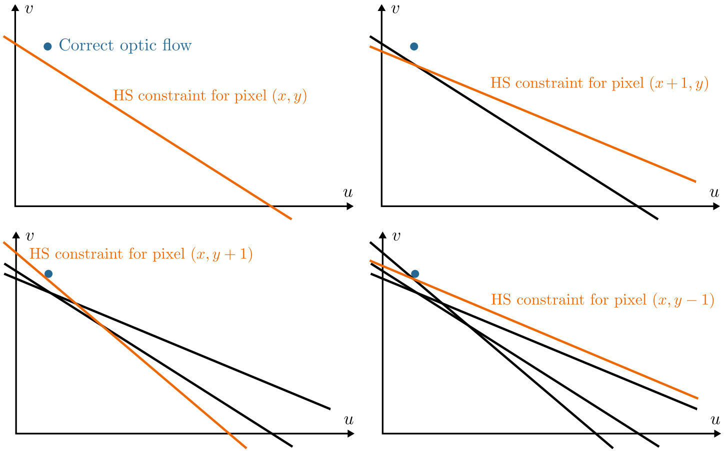
Two frames of video sequence

Color-coded motion field calculated with basic Horn-Schunck algorithm

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*.

Optical flow equation (as, e.g., on Page 15) can also be written as

$$I_t = \mathbf{u}^\top \cdot \nabla_{x,y} I \quad \text{or} \quad I_t = \mathbf{u}^\top \cdot \mathbf{g}$$

where $\nabla_{x,y} I = \mathbf{g}$ are common notations for the gradient $[I_x, I_y]^\top$

Vector notation: $\mathbf{g} = ||\mathbf{g}||_2 \cdot \mathbf{g}^\circ$, where $\mathbf{g}^\circ$ is the unit vector

Simple Case: Two Lines

Assume identical optic flow $\mathbf{u}$ at adjacent pixel locations p_1 and p_2

Optic flow equations $\mathbf{u}^\top \cdot \mathbf{g}_i^\circ = -\frac{I_t(p_i)}{||\mathbf{g}_i||_2}$ at both pixels

Unit gradients $\mathbf{g}_1^\circ = [g_{x1}, g_{y1}]^\top$ and $\mathbf{g}_2^\circ = [g_{x2}, g_{y2}]^\top$ at p_1 and p_2

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

Using b_i for the right-hand side:

$$\begin{aligned} ug_{x1} + vg_{y1} &= b_1 \\ ug_{x2} + vg_{y2} &= b_2 \end{aligned}$$

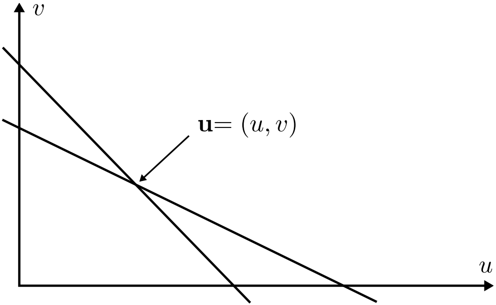
with $0 \leq |g_{xi}| \leq 1$ and $0 \leq |g_{yi}| \leq 1$ for numerical normalisation

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

$\mathbf{W} = \mathrm{diag}[w_1, \ldots, w_k]$ 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.

R. Klette. Concise Computer Vision.
©Springer-Verlag, London, 2014.

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