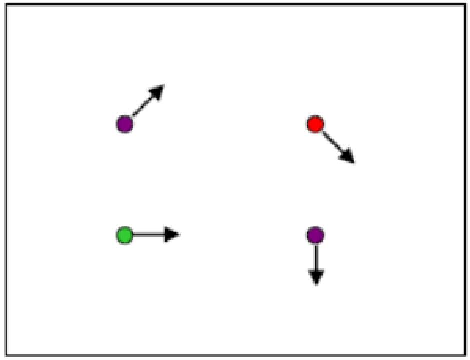
Motion

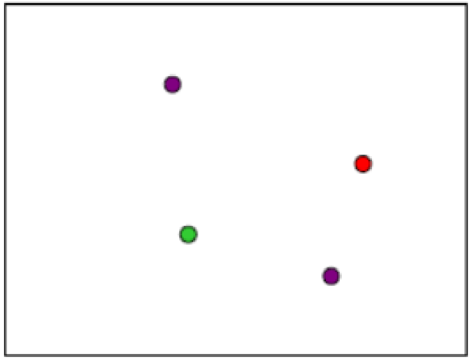
Motion Estimation

- Given a sequence of images we might ask
  - What are the moving objects in the scene?
  - What sort of motion are they undergoing?
  - Where will they be in the future?
- To answer these questions we need to measure the motion

- There are many problems in motion estimation
- Often the motion is ambiguous
- Image sequences contain a lot of data - efficiency is a concern
- Many interesting tasks involve complex motion - e.g. facial expression analysis



$I(x,y,t)$



$I(x,y,t+1)$

Motion

- Motion and stereo are closely related
  - "Correspondence problem", or "visual correspondence": what went where?
  - Can be solved sparsely or densely
    - Except for wide-baseline stereo, in both cases sparse solutions are now rare
- Comparison:
  - Stereo uses a 1D label set, motion has 2D
  - Stereo involves bigger changes in appearance
  - If motion is small, tracking is an easy way
  - Motion has an elegant formulation as a continuous problem

Challenges

- Figure out which features can be tracked
- Efficiently track across frames
- Some points may change appearance over time (e.g., due to rotation, moving into shadows, etc.)
- Drift: small errors can accumulate as appearance model is updated
- Points may appear or disappear: need to be able to add/delete tracked points

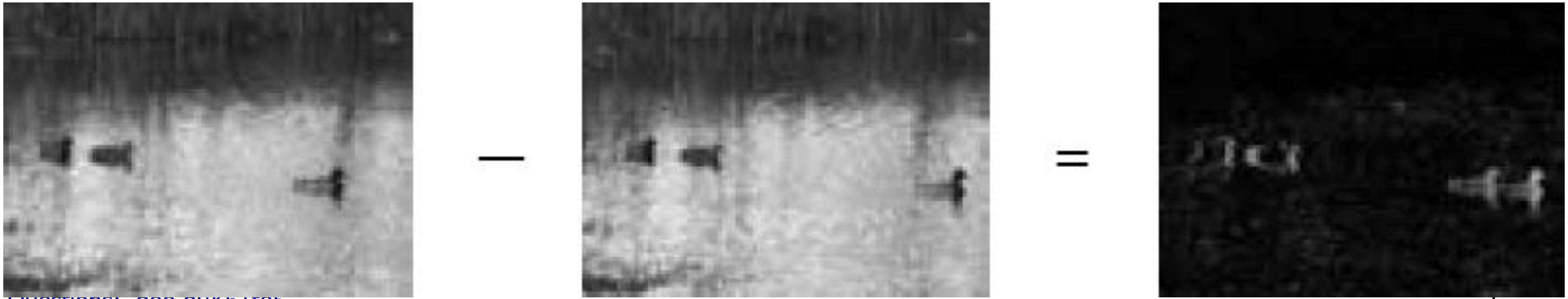
Simple Techniques

Motion Difference

- Take two images from a sequence
- Compute the change in brightness at each pixel in the image
- Threshold

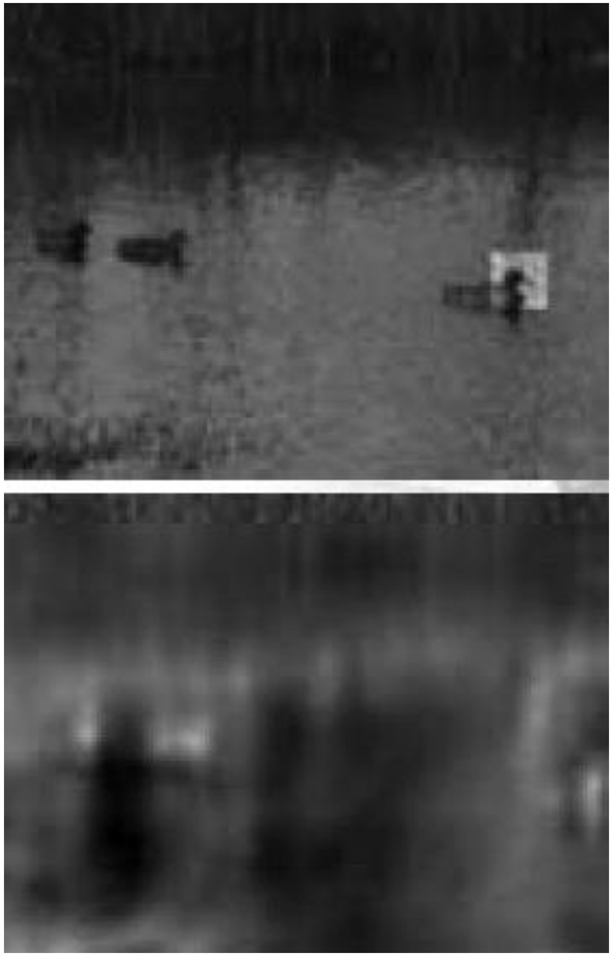
Background Models

- Find the average brightness at each pixel over a sequence
- Use the difference between the current frame and the average to find moving objects



Simple Techniques

- Area-based matching can also be used
- We take a template from the first image
- This is then compared to points in the second image to find corresponding regions
- This uses a 'distance measure' to compare patches



Motion Field and Optical Flow

The Motion Field

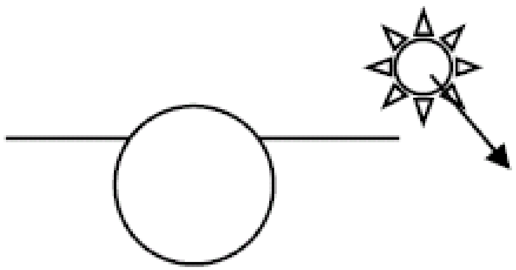
- "assigns a velocity vector to each point in the image"
- Tells us how the position of the *image* of the corresponding *scene point* changes over time
- Can be computed from the *scene* to tell us about the *image*

Optical Flow

- The "apparent motion of the brightness pattern" in an image
- Ideally it will be the same as the motion field, but this is not always the case
- Can be computed from the *image*, to tell us about the *scene*

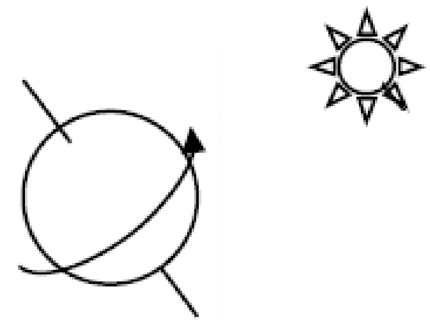
Optical Flow ≠ Motion Field

A Moving light



- The *image* changes so there is optical flow
- The *scene* objects do not move so there is no motion field

A Rotating Sphere



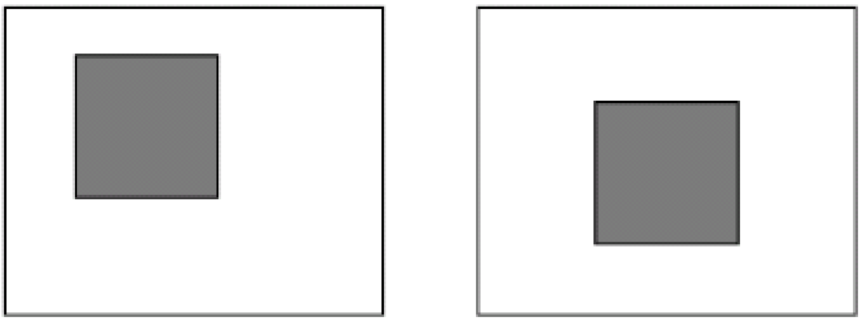
- The *scene* object moves, so there is motion field
- The *image* does not change, so there is no optic flow

Optical flow

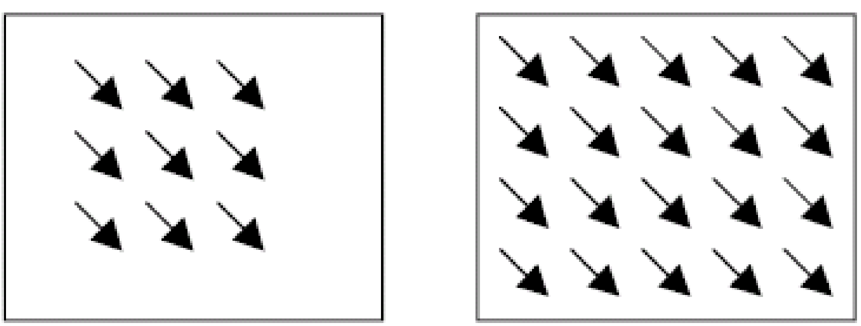
- Our “data cost” will be the difference between the old pixel intensity and the new pixel intensity
  - New and old pixel are related by the motion
- Instead of search, we can directly solve for a data cost
  - We consider a series of images as samples of a function $I(x,y,t)$
    - What are we assuming?
- No explicit search over (many) labels in contrast to stereo

Optical Flow is Ambiguous

- Consider the two images below:



- Two possible fields (of



So, optical flow

- Is not always what we want to compute
- Cannot be determined without ambiguity
- But it is all that we can compute from the images
- This means we need to make assumptions to find a *reasonable* flow field estimate

Brightness Constancy

Brightness constancy assumption:

$$I(x,y,t) = I(x+u, y+v, t+1)$$

- $I(x,y,t)$ is the brightness of the image at location (x,y) and time t
- (u,v) is the motion field at location (x,y) and time t
- This assumption is true apart from the effects of lighting (including shadows, reflections, and highlights)

Brightness Constancy

Another way to express brightness constancy is that

$$\frac{dI(x,y,t)}{dt} = 0$$

- This says that the image doesn't change over time - it just moves about

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

$$\boxed{\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0}$$

Brightness Constancy

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

Image derivative in x direction

Image derivative in y direction

Image derivative in t direction

$$\boxed{\nabla I[u,v]=-\frac{\partial I}{\partial t}}$$

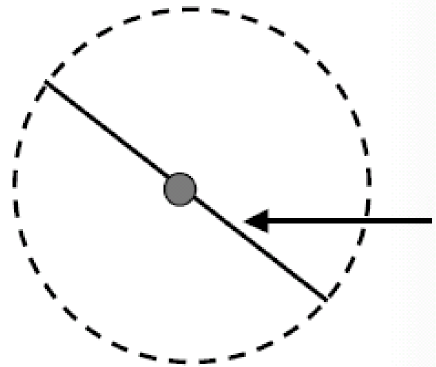
The Aperture Problem

There is no solution to the equation

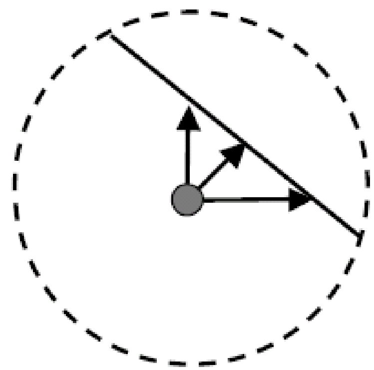
We only measure the projection of the true motion (u,v) on the intensity gradient, which makes it ambiguous

- We can determine the component of flow in the same direction as the image intensity gradient
- We cannot determine the component of flow perpendicular to it

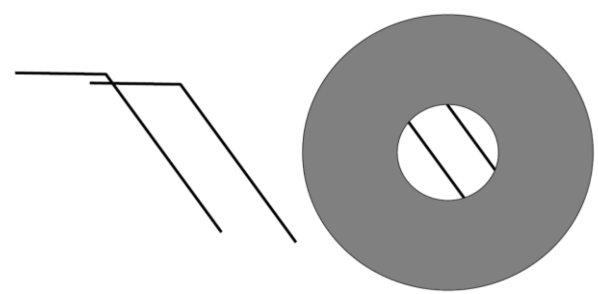
- This is the *aperture problem*



Line of constant brightness



We know we have to move to a point on the line, but not which one



Flow Smoothness

We need another constraint to find a unique solution

- This is the constraint that the flow field is smooth
- Neighbouring pixels in the image should have similar optical flow

We want u and v to have low variation

- We can do this by trying to set

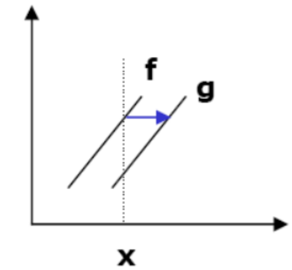
$$(u-\bar{u})=0, \ (v-\bar{v})=0$$

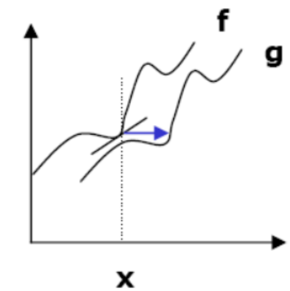
- So u and v are equal to the average of their neighbouring values

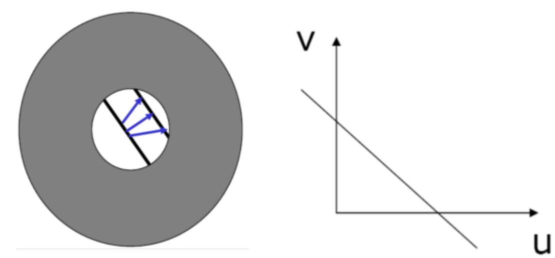
Smoothness assumption

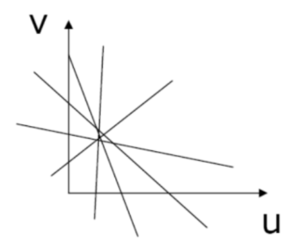
- One-dimensional example –linearization
  - Estimate displacement u using derivative
    - Two functions *f(x)* and *g(x)=f(x-u)*
  - Taylor series expansion
    $f(x-u) = f(x) - u\, f'(x) + E$
    f' denotes derivative
  - write difference as
    $f(x)-g(x) = u\, f'(x) + E$
  - Discarding higher order terms
    $\delta = (f(x)-g(x))/f'(x)$
  - works only for small u

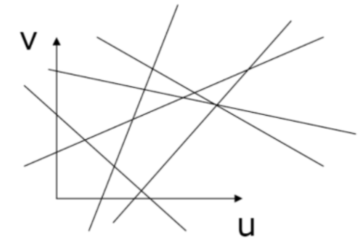
  - We need more than 1 pixel
  - Each pixel defines linear constraint on possible (u,v) displacement
    - For set of pixels with same displacement combine constraints to get estimate
    - For pixels with different displacements, somehow identify that case











Squared Errors

We now have three error terms

- If we square them then the error is always positive, and we can look for a minimum
- A weighting term, λ, balances the influence of the brightness and smoothness errors

The squared error term is

- To minimise: take derivatives with respect to *u* and *v*, set to 0, then solve

$$\lambda\left(\frac{\partial I}{\partial x}u+\frac{\partial I}{\partial y}v+\frac{\partial I}{\partial t}\right)^2+(u-\overline{u})^2+(v-\overline{v})^2$$

Minimisation

$$e = \lambda \left(\frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \frac{\partial I}{\partial t} \right)^2 + (u - \overline{u})^2 + (v - \overline{v})^2$$

$$\frac{\partial e}{\partial u} = 2\lambda \frac{\partial I}{\partial x} \left(\frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \frac{\partial I}{\partial t} \right) + 2(u - \overline{u}) \quad = 0$$

$$\frac{\partial e}{\partial v} = 2\lambda \frac{\partial I}{\partial y} \left(\frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \frac{\partial I}{\partial t} \right) + 2(v - \overline{v}) \quad = 0$$

A Solution

Solving the two equations gives

$$u = \overline{u} - \lambda \frac{\partial I}{\partial x} \frac{\overline{u}\frac{\partial I}{\partial x} + \overline{v}\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t}}{1 + \lambda\left(\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2\right)}$$

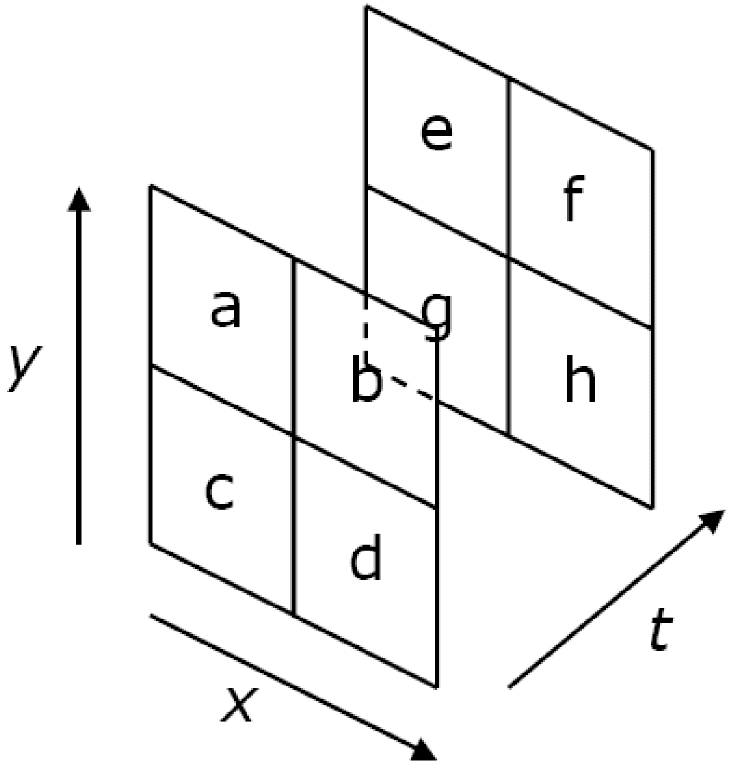
$$v = \overline{v} - \lambda \frac{\partial I}{\partial y} \frac{\overline{u}\frac{\partial I}{\partial x} + \overline{v}\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t}}{1 + \lambda\left(\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2\right)}$$

But we need to know ~*u* and ~*v* to compute *u* and *v*

Iterative solution:

- Estimate *u* and *v*
- Then compute the averages, ~*u* and ~*v*
- Then make a new estimate of *u* and *v*
- Then make a new estimate of ~*u* and ~*v*
- etc…

Computing the Optical Flow



Gradients:

dI/dx = (b+d+f+h) – (a+c+e+g)

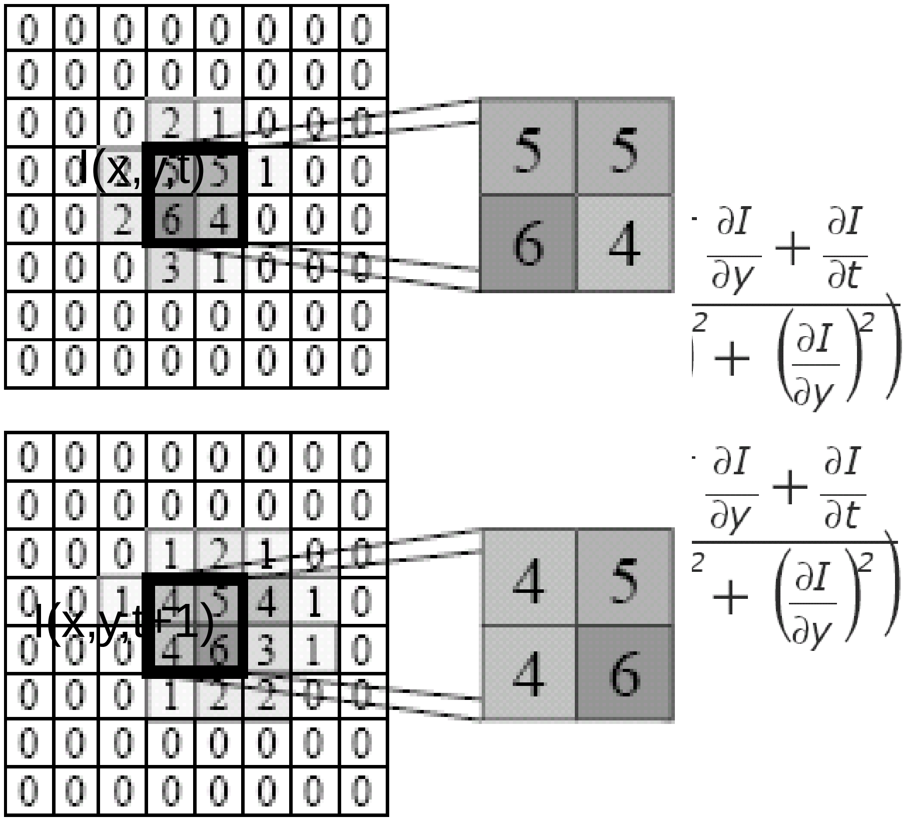
dI/dy = (a+b+e+f) – (c+d+g+h)

dI/dt = (e+f+g+h) – (a+b+c+d)

$$u = \overline{u} - \lambda \frac{\partial I}{\partial x} \frac{\overline{u}\frac{\partial I}{\partial x} + \overline{v}\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t}}{1 + \lambda\left(\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2\right)}$$

$$v = \overline{v} - \lambda \frac{\partial I}{\partial y} \frac{\overline{u}\frac{\partial I}{\partial x} + \overline{v}\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t}}{1 + \lambda\left(\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2\right)}$$

Computing the Optical Flow



dI/dx = (5+4+5+6)-(5+6+4+4)

dI/dy = (5+5+4+5)-(6+4+4+6)

dI/dt = (5+5+6+4)-(4+5+4+6)

Lukas-Kanade

- If there is a single translational motion (u,v)
  - In a window, or over the entire image
- At each pixel, the OFCE (optical flow constraint equation) says:
  - the spatial and temporal gradient $\nabla I[u, v] = -\frac{\partial I}{\partial t}$
  - These are the observations $I_x(x_i, y_i) \cdot u + I_y(x_i, y_i) \cdot v = -I_t(x_i, y_i)$

$$\begin{bmatrix} I_x(x_1, y_1) & I_y(x_1, y_1) \\ I_x(x_2, y_2) & I_y(x_2, y_2) \\ \vdots & \vdots \\ I_x(x_n, y_n) & I_y(x_n, y_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_t(x_1, y_1) \\ -I_t(x_2, y_2) \\ \vdots \\ -I_t(x_n, y_n) \end{bmatrix} \qquad S \cdot \begin{bmatrix} u \\ v \end{bmatrix} = -T$$

- We can use least squares to solve this

Lukas-Kanade

- Least square solution to

$$\begin{bmatrix} I_x(x_1,y_1) & I_y(x_1,y_1) \\ I_x(x_2,y_2) & I_y(x_2,y_2) \\ \vdots & \vdots \\ I_x(x_n,y_n) & I_y(x_n,y_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_t(x_1,y_1) \\ -I_t(x_2,y_2) \\ \vdots \\ -I_t(x_n,y_n) \end{bmatrix} \qquad S \cdot \begin{bmatrix} u \\ v \end{bmatrix} = -T$$

$$S^t S \cdot \begin{bmatrix} u \\ v \end{bmatrix} = -S^t T$$

second order matrix

A

$$S^t S = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \qquad S^t T = \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$$Ad = b \ \overset{LS}{\Rightarrow} \ \min_d \|Ad - b\|^2$$

Lukas-Kanade

$$S^t S \cdot \begin{bmatrix} u \\ v \end{bmatrix} = -S^t T \qquad \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

- When is this solvable? i.e., what are good points to track?
  - $S^T S$ should be invertible
  - $S^T S$ should not be too small due to noise
  - eigenvalues e_1, e_2 of $S^T S$ should not be too small
  - $S^T S$ should be well-conditioned
  - e_1 / e_2 should not be too large (e_1 = larger eigenvalue)
- Inverting 2x2 matrix has a closed form solution
- Unless spatial gradients are parallel
  - Multiple copies of the same gradient give singular matrix
  - We need textured regions
- Best if gradients are orthogonal e.g. corners

Computing the Optical Flow

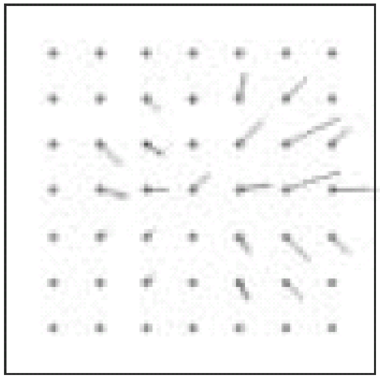
The algorithm is iterative

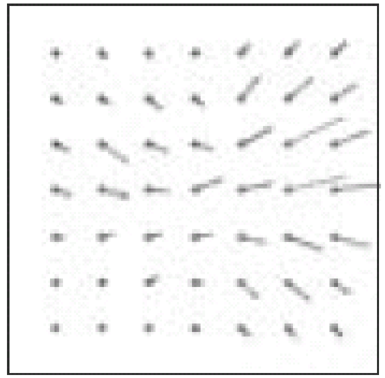
- We start with an initial estimate
- We refine it over a series of cycles
- We need an initial estimate
- We also need to know when to stop

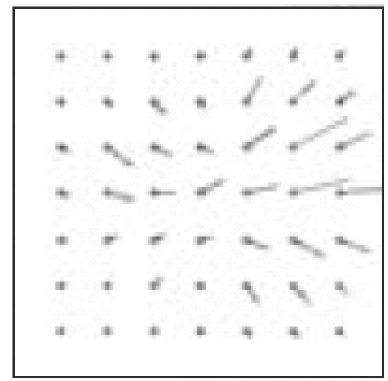
Initialisation

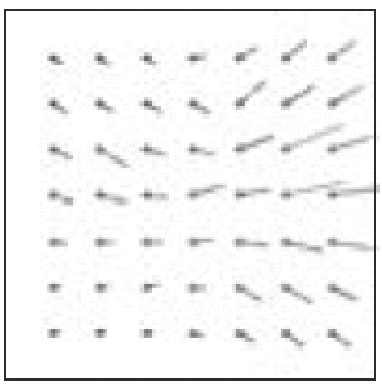
- We can start with an estimate of u and *v* of 0 everywhere
  - From coarse to fine
- Stop when the results at iteration *n* and *n+1* are very similar
  - This is when the algorithm converges
  - Can we be sure it will?

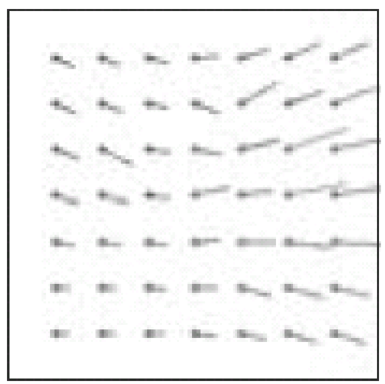
Computing the Optical Flow

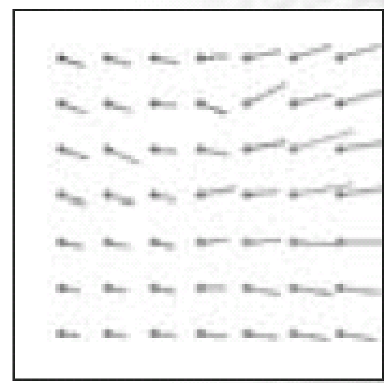

1 Iteration


2 Iterations


3 Iterations


5 Iterations


10 Iterations


15 Iterations

Template tracking

$$E(u,v) = \sum \underline{[I(x+u, y+v)} - T(x,y)]^2$$

template
(model)

current frame

u,v = hypothesized location of
template in current frame

Template tracking

$$E(u,v) = \sum [I(x+u, y+v) - T(x,y)]^2$$

$$\approx \sum [I(x,y) + uI_x(x,y) + vI_y(x,y) - T(x,y)]^2 \quad \text{First order approx}$$

$$= \sum [uI_x(x,y) + vI_y(x,y) + D(x,y)]^2$$

Take partial derivs and set to zero

$$\frac{\delta E}{du} = \sum [uI_x(x,y) + vI_y(x,y) + D(x,y)]\, I_x(x,y) = 0$$

$$\frac{\delta E}{dv} = \sum [uI_x(x,y) + vI_y(x,y) + D(x,y)]\, I_y(x,y) = 0$$

Form matrix equation

$$\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\sum \begin{bmatrix} I_x D \\ I_y D \end{bmatrix} \Rightarrow \boxed{\text{solve via least-squares}}$$

Template tracking

- Assumption of constant flow (pure translation) for all pixels in a larger window is unreasonable for long periods of time

- we can generalize Lucas-Kanade approach to other 2D parametric motion models (like affine or projective) by introducing a "warp" function W

$$E(u,v) = \sum [I(x+u, y+v) - T(x,y)]^2 \xrightarrow{\text{generalize}} \sum [\, I(W([x,y]; P)) - T([x,y])]^2$$

Affine motion

- An affine motion maps between two arbitrary triangles

$$u(x,y) = a_1 + a_2x + a_3y$$
$$v(x,y) = a_4 + a_5x + a_6y$$

  - Generalisation of translational motion where $a_2 = a_3 = a_5 = a_6 = 0$

- Using OFCE we get:

$$I_x(x_i, y_i) \cdot (a_1 + a_2x + a_3y) + \\ I_y(x_i, y_i) \cdot (a_4 + a_5x + a_6y) = -I_t(x_i, y_i)$$

- More complex optimization
  - 6 parameters to find instead of 2
  - Still can use least squares

Motion estimates

- From motion estimates you can compute the focus of expansion
  - Direction of heading, relative to camera by intersecting motion rays
  - clustering of moving pixels e.g. Hough transform voting scheme
- Frequently use for registering images
- Object tracking
- Useful for robotic applications i.e. efficient