Optical Flow

Image sequence (single camera) → Image tracking → Tracked sequence → 3D computation → 3D structure + 3D trajectory
What is Optical Flow?

Optical flow is the 2D projection of the physical movement of points relative to the observer.

A common assumption is brightness constancy:

\[ I(p_i, t) = I(p_i + \vec{v}_i, t + 1) \]
When does Brightness Assumption Break down?

• TV is based on illusory motion
  – the set is stationary yet things seem to move

• A uniform rotating sphere
  – nothing seems to move, yet it is rotating

• Changing directions or intensities of lighting can make things seem to move
  – for example, if the specular highlight on a rotating sphere moves

• Muscle movement can make some spots on a cheetah move opposite direction of motion
Optical Flow Assumptions: Brightness Constancy

Assumption

Image measurements (e.g. brightness) in a small region remain the same although their location may change.

\[ I(x + u, y + v, t + 1) = I(x, y, t) \]

(assumption)

* Slide from Michael Black, CS143 2003
Optical Flow Assumptions:

Neighboring pixels tend to have similar motions

When does this break down?
Optical Flow Assumptions:

• The image motion of a surface path changes gradually over time
1D Optical Flow
Optical Flow: 1D Case

Brightness Constancy Assumption:

\[ f(t) \equiv I(x(t), t) = I(x(t + dt), t + dt) \]

\[ \frac{\partial f(x)}{\partial t} = 0 \quad \text{Because no change in brightness with time} \]

\[ \frac{\partial I}{\partial x} \bigg|_{t} \left( \frac{\partial x}{\partial t} \right) + \frac{\partial I}{\partial t} \bigg|_{x(t)} = 0 \]

\[ l_x \quad v \quad l_t \]

\[ \Rightarrow v = \frac{I_t}{I_x} \]
2D Optical Flow
From 1D to 2D tracking

1D: \[ \frac{\partial I}{\partial x} \left|_t \right. \left( \frac{\partial x}{\partial t} \right) + \frac{\partial I}{\partial t} \left|_{x(t)} \right. = 0 \]

2D: \[ \frac{\partial I}{\partial x} \left|_t \right. \left( \frac{\partial x}{\partial t} \right) + \frac{\partial I}{\partial t} \left|_{x(t)} \right. + \frac{\partial I}{\partial y} \left|_t \right. \left( \frac{\partial y}{\partial t} \right) + \frac{\partial I}{\partial t} \left|_{x(t)} \right. = 0 \]

One equation but two velocity \((u, v)\) unknowns...
How does this show up visually? Known as the “Aperture Problem”
Aperture Problem in Real Life

Barber pole illusion

Barber’s pole  Motion field  Optical flow
From 1D to 2D tracking

The Math is very similar:

\[ \vec{v} \approx -\frac{I_t}{I_x} \]

\[ G = \sum_{\text{window around } p} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \]

\[ b = \sum_{\text{window around } p} \begin{bmatrix} I_x I_t \\ I_y I_t \end{bmatrix} \]

Aperture problem

Window size here \~ 11x11
More Detail: Solving the aperture problem

• How to get more equations for a pixel? -- impose additional constraints
• most common is to assume that the flow field is smooth locally
• one method: pretend the pixel’s neighbors have the same \((u,v)\)

\[
0 = I_t(p_i) + \nabla I(p_i) \cdot [u \ v]
\]

\[
\begin{bmatrix}
I_x(p_1) & I_y(p_1) \\
I_x(p_2) & I_y(p_2) \\
\vdots & \vdots \\
I_x(p_{25}) & I_y(p_{25})
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix}
= -
\begin{bmatrix}
I_t(p_1) \\
I_t(p_2) \\
\vdots \\
I_t(p_{25})
\end{bmatrix}
\]

\[
A_{25\times2}
\]

\[
d_{2\times1}
\]

\[
b_{25\times1}
\]

Suppose a 5x5 window

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
Lukas-Kanade flow

- Prob: we have more equations than unknowns
  \[ A \begin{pmatrix} d \\ 25x2 \end{pmatrix} = b \begin{pmatrix} 2x1 \end{pmatrix} \quad \rightarrow \quad \text{minimize } \| Ad - b \|^2 \begin{pmatrix} 25x1 \end{pmatrix} \]

- Solution: solve least squares problem
  - minimum least squares solution given by solution (in d) of:
    \[ (A^T A) \begin{pmatrix} d \end{pmatrix} = A^T b \begin{pmatrix} 2x2 \end{pmatrix} \]

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

- \[
A^T A
\]
- \[
A^T b
\]

- The summations are over all pixels in the K x K window
- This technique was first proposed by Lukas & Kanade (1981)
  - described in Trucco & Verri reading

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
Conditions for solvability

– Optimal $(u, v)$ satisfies Lucas-Kanade equation

\[
\begin{bmatrix}
\sum I_x I_x & \sum I_x I_y \\
\sum I_x I_y & \sum I_y I_y
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix} =
\begin{bmatrix}
\sum I_x I_t \\
\sum I_y I_t
\end{bmatrix}
\]

\[A^T A = A^T b\]

When is This Solvable?

• $A^T A$ should be invertible
• $A^T A$ should not be too small due to noise
  – eigenvalues $\lambda_1$ and $\lambda_2$ of $A^T A$ should not be too small
• $A^T A$ should be well-conditioned
  – $\lambda_1 / \lambda_2$ should not be too large ($\lambda_1 =$ larger eigenvalue)

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
Eigenvectors of $A^T A$

$$A^T A = \left[ \begin{array}{cc} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{array} \right] = \sum \left[ \begin{array}{c} I_x \\ I_y \end{array} \right] [I_x \ I_y] = \sum \nabla I (\nabla I)^T$$

- Suppose $(x,y)$ is on an edge. What is $A^T A$?
  - gradients along edge all point the same direction
  - gradients away from edge have small magnitude
  $$(\sum \nabla I (\nabla I)^T) \approx k \nabla I \nabla I^T$$
  $$(\sum \nabla I (\nabla I)^T) \nabla I = k \| \nabla I \| \nabla I$$
  - $\nabla I$ is an eigenvector with eigenvalue $k \| \nabla I \|$
  - What’s the other eigenvector of $A^T A$?
    - let $N$ be perpendicular to $\nabla I$
    $$(\sum \nabla I (\nabla I)^T) N = 0$$
    - $N$ is the second eigenvector with eigenvalue 0
- The eigenvectors of $A^T A$ relate to edge direction and magnitude

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
$\sum \nabla I(\nabla I)^T$

- large gradients, all the same
- large $\lambda_1$, small $\lambda_2$
Low texture region

\[
\sum \nabla I (\nabla I)^T
\]

- gradients have small magnitude
- small \(\lambda_1\), small \(\lambda_2\)

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003*
High textured region

\[ \sum \nabla I (\nabla I)^T \]

- gradients are different, large magnitudes
- large \( \lambda_1 \), large \( \lambda_2 \)

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
Observation

- This is a two image problem BUT
  - Can measure sensitivity by just looking at one of the images!
  - This tells us which pixels are easy to track, which are hard
    - very useful later on when we do feature tracking...
Errors in Lukas-Kanade

What are the potential causes of errors in this procedure?

– Suppose $A^T A$ is easily invertible
– Suppose there is not much noise in the image

• When our assumptions are violated
  – Brightness constancy is not satisfied
  – The motion is not small
  – A point does not move like its neighbors
    • window size is too large
    • what is the ideal window size?

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
Improving accuracy

- Recall our small motion assumption
  \[0 = I(x + u, y + v) - I_{t-1}(x,y)\]
  \[\approx I(x, y) + I_xu + I_yv - I_{t-1}(x,y)\]

- This is not exact
  - To do better, we need to add higher order terms back in:
  \[= I(x, y) + I_xu + I_yv + \text{higher order terms} - I_{t-1}(x,y)\]

- This is a polynomial root finding problem
  - Can solve using **Newton’s method**
    - Also known as **Newton-Raphson** method

  - Lukas-Kanade method does one iteration of Newton’s method
    - Better results are obtained via more iterations

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003*
Iterative Refinement

• Iterative Lukas-Kanade Algorithm
  1. Estimate velocity at each pixel by solving Lucas-Kanade equations
  2. Warp $I(t-1)$ towards $I(t)$ using the estimated flow field
     - use image warping techniques
  3. Repeat until convergence

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
Revisiting the small motion assumption

- Is this motion small enough?
  - Probably not—it’s much larger than one pixel (2\text{nd} order terms dominate)
  - How might we solve this problem?

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
Reduce the resolution!

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
Optical Flow Results

Lucas-Kanade
without pyramids

Fails in areas of large motion

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
Optical Flow Results

Lucas-Kanade with Pyramids

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003
What about other types of motion?
Generalization

- Transformations/warping of image

\[
E(A, h) = \sum_{x \in \mathcal{R}} \left[ I(Ax + h) - I_0(x) \right]^2
\]

Affine: \( A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \) \( h = \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} \)
Affine Flow
Linear Basis

You can think of this as just another set of linear basis functions!

\[ u(x; c) = c_1 \begin{array}{c}
\end{array} + c_2 \begin{array}{c}
\end{array} + c_3 \begin{array}{c}
\end{array} + c_4 \begin{array}{c}
\end{array} + c_5 \begin{array}{c}
\end{array} + c_6 \begin{array}{c}
\end{array} \]

\[ u(x; c) = \sum_{j=1}^{n} a_j b_j(x) \]
Horn & Schunck algorithm

Additional smoothness constraint:

\[ e_s = \iint ((u_x^2 + u_y^2) + (v_x^2 + v_y^2)) \, dx \, dy, \]

besides Opt. Flow constraint equation term

\[ e_c = \iint (I_x u + I_y v + I_t)^2 \, dx \, dy, \]

minimize \( e_s + \alpha e_c \)
Horn & Schunck algorithm

In simpler terms: If we want dense flow, we need to regularize what happens in ill conditioned (rank deficient) areas of the image. We take the old cost function:

\[
d = \arg \min_d \sum_{x \in N} (I(x, t) - I(x + d, t + 1))^2
\]

And add a regularization term to the cost:

\[
d = \arg \min_d \sum_{x \in N} (I(x, t) - I(x + d, t + 1))^2 + \alpha \|d\|
\]

Convex Program!

We will see a lot of such formulations in in robust regression!
Discussion: What are the other methods to improve optical flows?