Tutorial 13: Optical Flow

Computer Vision 236873, Winter 2018

adapted from: Kevin Chavez, Ben Cohen-Wang, Garrick Fernandez, Noah Jackson, Will Lauer

In this class so far we have learned a variety of tools that enable us to detect key points, recognize objects, and use segmentation in images. However, in many cases we want to be able to perform similar operations on video. Specifically, we are often interested not only in the location of certain objects, but also the movement of these objects over time. This tutorial focuses on how we can apply previously covered techniques along with new methods to effectively track the motion of pixels across many images, with applications in areas such as self-driving cars, robots, and security systems to name a few.

1 Definitions and Key Assumptions

Optical flow is the movement of pixels over time. The goal of optical flow is to generate a motion vector for each pixel in an image between \( t_0 \) and \( t_1 \) by looking at two images \( I_0 \) and \( I_1 \). By computing a motion vector field between each successive frame in a video, we can track the flow of objects.

Given a two dimensional image, optical flow can only represent the apparent motion of brightness patterns, meaning that the movement vectors of optical flow can be the result of a variety of actions. For instance, variable lighting can cause strong motion vectors on static objects, and movement into or out of the frame cannot be captured by the 2D motion vectors of optical flow. One example of an issue poorly dealt with by optical flow is the aperture problem.

1.1 Brightness Consistency

As optical flow can only represent apparent motion, to correctly track the motion of points on an image we must assume that these points remain at the same brightness between frames. The equation for this brightness consistency equation is as follows

\[
I(x, y, t) = I(x + dx; y + dy, t + dt)
\]

where \( dx, dy \) are the horizontal and vertical motion of a point.
1.2 Small Motion

Optical flow assumes that points do not move very far between consecutive images. This is often a safe assumption, as videos are typically comprised of 20+ frames per second, so motion between individual frames is small. However, in cases where the object is very fast or close to the camera this assumption can still prove to be untrue. To understand why this assumption is necessary, we must consider the Brightness Consistency equation defined above. When trying to solve this equation, it is useful to linearize the right side using a Taylor expansion. This yields:

\[ I(x + dx, y + dx, t + dt) \approx I(x, y, t) + I_x dx + I_y dy + I_t dt \]

Linearizing in this way allows us to solve for the \( dx \) and \( dy \) motion vectors we want.

\[ I(x + dx, y + dy, t + dt) \approx I(x, y, t) + I_x dx + I_y dy + I_t \]

\[ I(x + dx, y + dy, t + dt) - I(x, y, t) \approx I_x dx + I_y dy + I_t = 0 \]

Divide by \( dt \) and denote

\[ u = \frac{dx}{dt}, v = \frac{dy}{dt} \]

Giving us:

\[ I_x u + I_y v = -I_t \]

\[ \nabla I[u, v]^T = -I_t \]

Ignoring the meaning of this derivation for the moment, it is clear that we do not have enough equations to find both \( u \) and \( v \) at every single pixel. This ambiguity can be seen, for example, in the aperture problem.

1.3 Spatial Coherence

Spatial coherence is the assumption that nearby pixels will move together, typically because they are part of the same object. Assuming that pixels move together allows us to use many more equations with the same \([u, v]\), making it possible to solve for the motion of all pixels.

2 Lucas-Kanade

Recovering image motion given by \((u, v)\) in the above equation requires at least two equations per pixel. To achieve this, the Lucas-Kanade technique for image tracking relies on an additional constraint spatial coherence. The spatial coherence constraint is applied to a pixel using a window of size \(k \times k\). The assumption is that the neighboring pixels in this window will have the same \((u, v)\). For example, in a \(5 \times 5\) window the following equations apply for each of the pixels \(p_i\) in that window:

\[ \nabla I(p_i)[u(p_i), v(p_i)]^T + I_t(p_i) = 0 \]
This produces an overly-constrained system of linear equations of the form $Ad = b$:

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

Using a least squares method for solving over-constrained systems, we reduce the problem to solving $d = -(A^TA)^{-1}A^Tb$:

\[
\begin{bmatrix}
u \\
v
\end{bmatrix} = \left(\frac{\sum_i I_x I_x}{\sum_i I_y I_x} \frac{\sum_i I_x I_y}{\sum_i I_y I_y}\right)^{-1} \left(\frac{\sum_i I_x I_t}{\sum_i I_y I_t}\right)
\]

In order to solve the system following conditions should hold:

1. $A^TA$ should be invertible (i.e. Eigenvalues $\lambda_1, \lambda_2 \neq 0$)

2. $A^TA$ should not be too small due to noise. (i.e. Eigenvalues $\lambda_1$ and $\lambda_2$ of $A^TA$ should not be too small)

3. $A^TA$ should be well-conditioned (i.e. $\lambda_1 \approx \lambda_2$)

### 2.1 Geometric Interpretation

It should be evident that the least squares system of equations above produce a second moment matrix $H = ATA$. In fact, this is the Harris matrix for corner detection.

\[
A^TA = \left[\frac{\sum_i I_x I_x}{\sum_i I_y I_x} \frac{\sum_i I_x I_y}{\sum_i I_y I_y}\right] = \sum_i \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} = \nabla I(\nabla I)^T = H
\]

We can relate the conditions above for solving the motion field $[u, v]$ to tracking corners detected by the Harris matrix $H$. In particular, the eigenvectors and eigenvalues of $H = A^TA$ relate to the direction and magnitude of a possible edge in a region. Using this interpretation, it is apparent that an ideal region for Lucas-Kanade optical flow estimation is a corner. Visually, if $\lambda_1$ and $\lambda_2$ are too small this means the region is too flat. If $\lambda_1 >> \lambda_2$, the method suffers from the aperture problem, and may fail to solve for correct optical flow.

### 2.2 Assumptions break

Errors may still arise when:

1. The motion is not small.

   (a) solve iteratively.
2. Brightness constancy is not satisfied, meaning that a pixel may change intensity from different time steps.
   **Solution:** use descriptor which not sensitive to illuminations, for example.

3. Spatial coherence is not satisfied, meaning neighboring pixels do not move alike. This may arise due to an inappropriately sized window (choosing bad k).
   **Solution:** Regularization based methods.