# Some notes on Optical Flow

gmoy@robotics.eecs.berkeley.edu

Optical flow is defined as the estimation of the apparent motion of the image brightness pattern which can be computed from sequences of images. In our case, we are only dealing with a 1-D problem, so all the equations simplify, but the theory is the same.

We start with the Image Constancy Equation which can be found in modern books on computer vision. The image, $I$, is a function of $x$, $y$, and $t$, and is written as $I(x, y, t)$ (Figure 1). The variables $x$ and $y$ are also functions of time. The apparent brightness of moving objects remains constant, which leads to the equation $\frac{dI}{dt} = 0$.

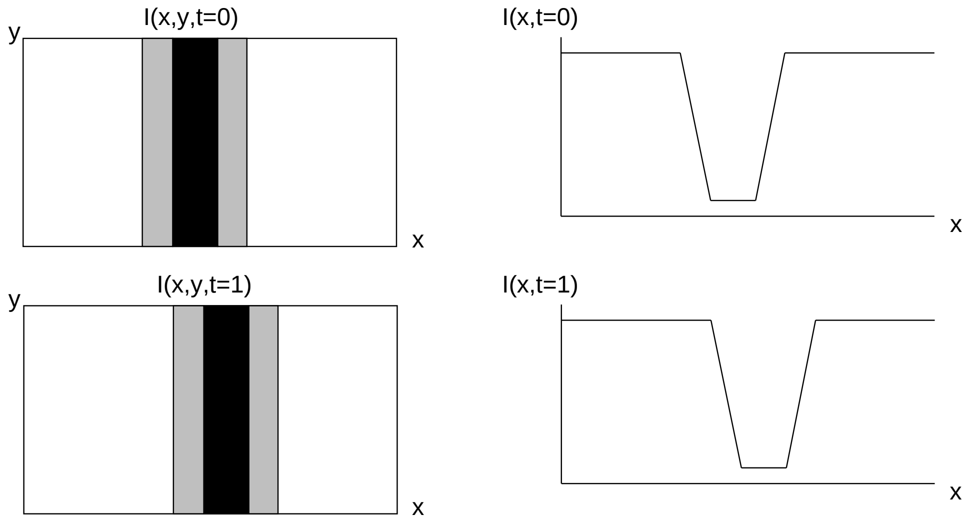


Figure 1: The image, $I$, at two different times.

Using the chain rule, we obtain:

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

I'll assume that we are looking for horizontal velocities, so $\frac{dy}{dt} = 0$. This simplifies the equation to

$$\begin{aligned} \frac{dI(x(t), y_0, t)}{dt} &= \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial t} = 0, \\ \frac{\partial I}{\partial x}\frac{dx}{dt} &= -\frac{\partial I}{\partial t}, \\ \frac{dx}{dt} &= -\frac{\frac{\partial I}{\partial t}}{\frac{\partial I}{\partial x}} \end{aligned}$$

We are estimating the velocity, which is $\frac{dx}{dt}$.

There are three commonly used methods to solve this problem. They are the gradient methods, feature tracking, and template matching.

## Gradient Methods

We want to calculate $\frac{dx}{dt}$ at certain points of interest. If we consider a neighborhood of 5 points (two to the left, the point of interest, and two to the right), we can get 5 numbers for $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial t}$. Using a least squares approach, the *optimal* velocity, in the sense of least squares, can be calculated.

This method might be too complex to implement on the Xilinx chip. The basic idea is to solve for $\frac{dx}{dt}$ as accurately as possible, but still be able to implement it on the Xilinx chip. Simpler approximations and avoiding division is a good start. I will leave you to your creativity to find alternate solutions.

## Feature Tracking

The basic idea of feature tracking is to find features which can be tracked from image to image. Some possible trackable features are edges, brightest spots, darkest spots, local maxima, and local minima. There are some more possibilities so you can be creative in what to track. With the locations of the features, you estimate how much the feature moved from frame to frame, which is $dx$ and $dt$ is known. Other common techniques for determining the movement are cross-correlation and minimizing sum-of-squared error.

## Template Matching

For template matching, you create a table of all the possibilities you will encounter and then given a reading of the image, you look up the result in the table. For example, look at the point of interest and its neighboring point at t = 0, $t_0$, and $2t_0$. From the three time readings, you can get two intensity changes. Having two points, two changes per point, and three possibilities per intensity change (increase, decrease, no change) gives $3^4 = 81$ possible combinations of inputs. To implement this algorithm you would create a ROM with 81 entries corresponding to the possible inputs. See the second reference under Notes on Final Project on the webpage.

## Testing your Algorithm

Whichever algorithm you pursue, you should test it on some basic cases such as blurred stripes moving right or left, blurred stripes moving in and out, and a constant intensity moving right or left (you should see a zero velocity vector, since it is indistinguishable from not moving at all). If you are not getting what you expect, ask yourself where your algorithm breaks down and how to fix it. Prototype your algorithm using a high level language like C, C++, or Java. There is test data on the webpage.