ESE 461: Design Automation for Integrated Circuit Systems

Final Project

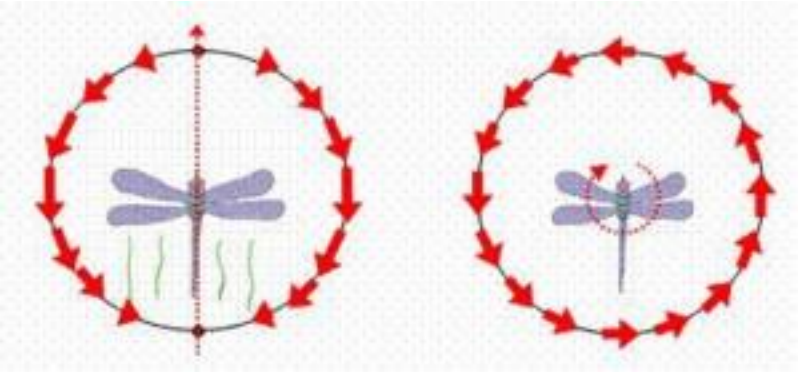
Option 1: Optical Flow

---

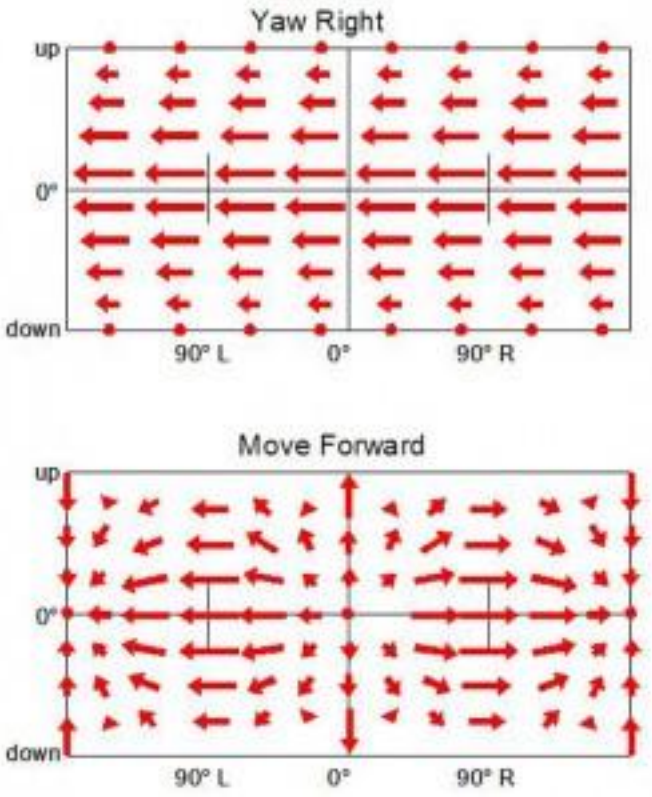
1. Optical Flow Introduction

The term "optical flow" refers the apparent visual motion that you experience as you move through the world. Suppose you sitting in a car or a train, and are looking out the window. You see trees, the ground, buildings, etc., appear to move backwards. This motion is optical flow. This motion can also tell you how close you are to the different objects you see. Distant objects like clouds, and mountains move so slowly they appear still. The objects that are closer, such as buildings and trees, appear to move backwards, with the closer objects moving faster than the distant objects. Very close objects, such as grass or small signs by the road, move so fast they whiz right by you.

In nature, insects are believed to be using optical flow for motion estimation and navigation. For example, consider the overhead view of a dragonfly in the figure below. The figure on the left shows the dragon fly traveling forward. The optic flow travels from the forward to backward direction, and is generally faster on the left and right than in the front or back. The figure on the left shows a dragon fly rotating to the right in one spot. Here the optic flow is to the left in all directions. If the dragonfly were flying a curved path, the optic flow patterns would be a combination of these two patterns. What we see here is that in addition to detecting obstacles, the optic flow can be used to measure or estimate one's own motion.



Now if we evaluate from the point of view of the dragonfly, and with the 360-degree field of view flattened onto the screen, what we get is the next figure below. When the dragonfly is rotating, or yawing, to the right, the optical flow everywhere will be to the left. However in the directions along the axis of rotation, the optic flow will be zero. When traveling forward, the optic flow will diverge from the forward direction, flow backwards, and converge in the rear.



There are many algorithms that may be used to measure optical flow. The most famous one is the Lucas-Kanade method [2], which analyzes spatial and temporal gradients over multiple frames to estimate optical flow. In this project, you are asked to implement the LK method.

2. Lucas–Kanade method Description

The Lucas-Kanade method is a widely used differential method for optical flow estimation developed by Bruce D. Lucas and Takeo Kanade. It has 3 assumptions:

1) **brightness constancy: the value of pixel is same for a point I(x,y,t) and I(x',y',t+1)**
2) **small motion: the motion vector for a point is small,**
3) **small region move together: the pixel's neighbors have the same (u,v).**

Algorithm steps:

Input: I_1, I_2, assuming image size = 128×128

(I_1 is the image at time t, I_2 is the image at time t+1)

Output: motion vector V = (V_x, V_y)

1) **Calculate the partial derivatives of the image I_1 (I_x, I_y, I_t) with respect to position x, y and time t. At the edge of the image I_1, I_x, I_y and It can be set as 0, because I_1(i-1, j), I_1(i+1, j), I_1(i, j-1), I_2(i, j+1) might not exis.**

	I(i,j-1)	
I(i-1,j)	I(i,j)	I(i+1,j)
	I(i,j+1)	

2) $I_x(i, j) = I_1(i+1, j) – I_1(i-1, j)$
3) $I_y(i, j) = I_1(i, j+1) – I_2(i, j-1)$
4) $I_t(i, j) = I_2(i, j) – I_1(i, j)$
5) For every pixel in I1, take location (i, j) as an example:
   Within a window centered at $I_1(i, j)$, assuming the window size is 5×5

		I(i,j)		

We have the corresponding 5×5 windows in I_x, I_y, It matrixes.

		$I_x(i,j)$		

		$I_y(i,j)$		

		$I_t(i,j)$		

For pixel $q_1=I_1(i-2,j-2)$, $q2= I_1(i-2,j-1)$, …, $q_{25}= I_1(i+2,j+2)$ in the 5×5 window, we have the equation:

$$I_x(q_1)V_x + I_y(q_1)V_y = -I_t(q_1)$$
$$I_x(q_2)V_x + I_y(q_2)V_y = -I_t(q_2)$$
$$\vdots$$
$$I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n)$$

These equations can be written in matrix form Av=b, where

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix}, \qquad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, \quad \text{and} \quad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}$$

6) The motion vector $V = (V_x, V_y) = (A^TA)^{-1}A^Tb$

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i)I_y(q_i) \\ \sum_i I_y(q_i)I_x(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i)I_t(q_i) \\ -\sum_i I_y(q_i)I_t(q_i) \end{bmatrix}$$

The inverse of a matrix

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1} = \frac{1}{A_{11}A_{22} - A_{12}A_{21}} \begin{bmatrix} A_{22} & -A_{12} \\ -A_{21} & A_{11} \end{bmatrix}$$

3. Project Requirement

For this project, we will focus on implementing the basic optical flow algorithm -- Lucas–Kanade method in a custom ASIC chip to speed up its performance.

In the testbench you create, please load the test images we provide consecutively to process these images with the optical flow accelerator module you have designed. The outputs generated by your optical flow hardware accelerator should equal the "ground truth" generated by the example C code that also implement the LK method in software.

4. Design Specifications

Your custom-designed optical flow accelerator should be able to correctly perform the LK method, and achieve the minimum performance listed below using the 180nm standard cell library we provide:

Clock frequency: higher than 160 MHz

Power consumption: lower than 70 mW

Total chip area: smaller than 3 mm^2

Reference and useful links

[1] http://www.centeye.com/technology/optical-flow/
[2] https://en.wikipedia.org/wiki/Lucas%E2%80%93Kanade_method
[3] http://docs.opencv.org/trunk/d7/d8b/tutorial_py_lucas_kanade.html