Project Spec.

EECS150 Final Project, Fall 1998: Optical Flow Estimator

R.S. Fearing
Dept. of Electrical Engineering and Computer Sciences
Univ. of California, Berkeley

1 Objectives

The final project for this semester is to design an optical flow estimator. The optical flow estimator will capture data from a video camera and calculate velocity vectors at specific points. In the first mode of operation, the video data will be stored in a 64K byte RAM, then transmitted at 115.2 KBaud over a serial line to a PC for display. In the second mode, velocity vectors in the horizontal or vertical direction will be tracked from frame to frame. The velocity vectors will be transmitted over the serial line to be displayed as bar graphs on a PC.

1.1 General Philosophy

This document describes the input/output specification for the optical flow estimator, and gives the schedule for completing requirements for this project. As in the real world, the user/external interface is specified; it is up to you to specify most of what goes into the system. You can consider the course staff as being the customers for your project, and you have contracted to deliver a working system. We have checkpoints and demonstrations along the way to see that our contract will be satisfied. Our contract also has a clause that you won't get paid (i.e. credit) if you don't deliver the working pieces on the specified dates. Exceptions can only be made for serious medical problems. Now you may want to deliver extra features, which is fine, but the basic system needs to be working first. As customers, we aren't going to pay for simulations, we need to see the real thing.

1.2 General Tips

Just because a design works does not make it a good design. Use good design practices such as:

- top-down design
- design, simulate, and debug in modules
- use synchronous design methodology (e.g. use only synchronous reset parts). (One global asynchronous reset for the whole design using the STARTUP block is fine). Use BUFGS for a single global clock. Pay attention to timing for portions of the system running at more than 5 MHz, (use Xilinx timing analyzer tool to check for worst case path).
- divide work up with your partner for individual modules (but make sure you have agreed on the interface!)

Budget plenty of time to get your project done. With this level of complexity, it takes at least three times as long to debug a systems as it does to design and enter it. In fact, however long you think it will take to complete a task, it will take 3 times longer. Plan accordingly. Also, lab space is limited, and workstations will need to be shared. Plan accordingly.

2 Project Description

A top level block diagram of the project is shown in Figure 1. (You are free to rearrange the blocks to optimize or simplify the design. For example, the video A/D never drives the serial transmitter directly.) There are five main blocks in the system:

1. Memory and address block. A 64K byte memory will be used to store video data, an image of size up to 256 by 256. You can think of the memory as holding an array:
`unsigned char image[256][256];` .
The counters are used to sequentially store and retrieve data from the memory. This module is built in Lab 7.

2. The serial transmitter sends 8 data bits with a start and stop bit at 115.2KBaud to the PC serial interface. The effective data rate is about 11 K Bytes per second. This block will be built and debugged for checkpoint 1.

3. The video interface converts the analog data from a video camera into 8 bits of digital data (one byte per pixel) using an analog to digital (A/D) converter. The *sync detector* provides synchronization information, in particular, when the top of the image is (vertical sync) and when the start of a line is (horizontal sync). This block will be built and debugged in checkpoint 2.

4. The optical flow estimator will implement your velocity estimation algorithm on the image data stored in memory. Suggested algorithms are template matching or spatial/temporal difference ratios. This will provide crude velocity estimates. (We will assume that a higher level algorithm, such as running on a PC, would interpret these raw velocity vectors). Part of the project is designing your own algorithm which fits in the XC4005.

5. The controller module is responsible for generating all clock, timing, user input/output, and control signals. It is probably better to have a number of simple, modular FSMs controlling various functions, than one huge FSM controlling everything.

2.1 Operation Mode 1: Video Frame Capture

In mode 1, the system operates as a frame grabber with serial interface. First an image is captured in 1/60 of a second (one *field* time). Then the most significant 7 bits of each image pixel are sent, one at a time, through the serial port to the PC in a little over 4 seconds. The software on the PC will display the image in a window. This mode can be used to verify that the A/D, RAM and serial transmitter are functioning correctly.

2.2 Video Frame Size

To increase the design space of projects, your team's video frame size will depend on student ID numbers. Take the last digit of each partner's student ID number. (Reentry students should

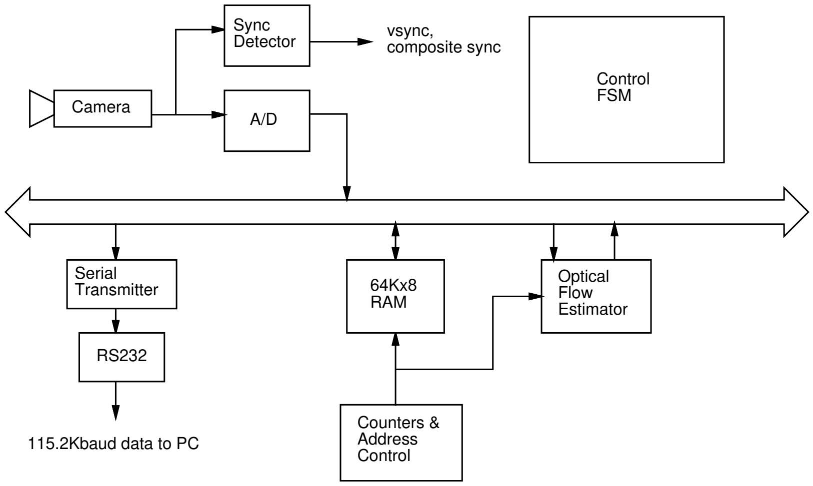

Figure 1: Top Level Block Diagram.

SID Sum	Horiz.	Vert.	vectors	Code	SID Sum	Horiz.	Vert.	vectors	Code
0	190	190	5V	0x0	10	190	210	5H	0x09
1	190	190	6V	0x1	11	190	230	6H	0x0a
2	190	190	7V	0x2	12	210	190	7H	0x0b
3	210	190	5V	0x3	13	210	230	5H	0x0c
4	210	190	6V	0x4	14	210	230	6H	0x0d
5	210	210	7V	0x5	15	230	190	7H	0x0e
6	230	190	5V	0x6	16	230	210	5H	0x0f
7	230	190	6V	0x7	17	190	210	5H	0x09
8	230	210	7V	0x8	18	190	190	5V	0x0
9	190	190	5V	0x0					

Table 1: Frame Size Specification. There are several different video frame sizes, based on student ID number. The code byte tells the PC which frame size you are using and how many velocity vectors. The notation *5 V* means that you will be measuring the vertical velocity component at 5 uniformly spaced spots along a column. Similarly, *7 H* means that you will be measuring the horizontal velocity component at 7 spots along a row.

assume 0). Add the last digits together, resulting in a number from 0 to 18. (Example 11112222 + 22223333 gives 5). This scheme is shown in Table 1. There is not much difference in complexity between the different frame sizes. The purpose of this design variation is to make it a little more difficult for your project design to get ripped off.

2.3 Operation Mode 2: Optical Flow Estimation

In mode 2, the system estimates the optical flow between frames. The optical flow estimate at a particular location is usually given in terms of pixels per frame. Horizontal and vertical velocity estimates at the specified number of locations will be sent to the PC at 60 times per second. (See section 3.1).

2.4 Serial Protocol

The video tracker uses two different packets for mode 1 and mode 2 for communication to the PC. Figure 2 shows the data transmitted in the two modes. The header byte signals the start of a serial transmission packet, by setting its MSB=0 (bit 7 = 0). (All data bytes in the packet have MSB=1). Bit 6 is 1 for mode 1, and 0 for mode 2. In both modes, bit 5 is 0 for the header.

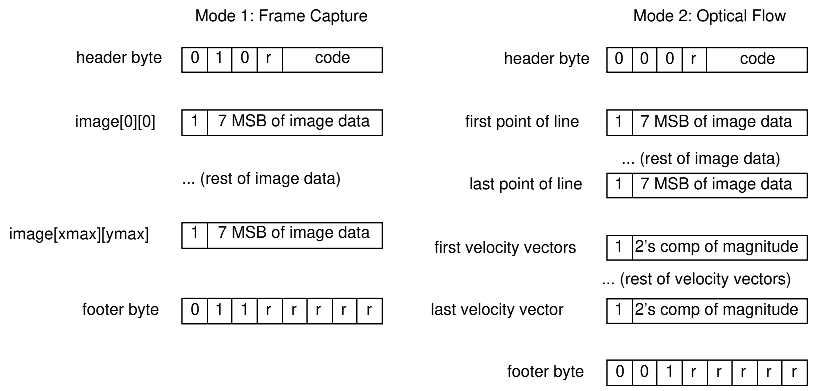

Figure 2: Serial transmission protocols for mode 1 and mode 2. The MSB must be 1 except in a header byte or footer byte. r is reserved for extra features.

Bit 4 is reserved. The last 4 bits are the hex code for the resolution.

In frame dump mode (mode 1), the next 36,100 (for 190×190) to 52,900 (for 230×230) bytes represent the 7 most significant bits of each image data point. The PC knows what size frame to expect by the value of the code byte (Table 1).

In optical flow mode (mode 2), the next set of data to be sent to the PC is a horizontal or vertical line (depending on which direction you are assigned) of the image being processed. If you are measuring vertical velocity (modes 5V, 6V, 7V) you send a single vertical column of data from the center of the image. If you are measuring horizontal velocity, (modes 5H, 6H, 7H) you send a single horizontal row of data.

The bytes are arranged with bit $7 = 1$, and the rest of the bits represent the 7 most significant bits of each point of the line. The next set of data to be sent are the estimated velocity vectors. The velocity vector has bit $7 = 1$, and the next seven bits are the 2's complement representation of the velocity. An invalid velocity, such as from a divide by zero, is represented by $-64 = 1000000_2$.

The PC then expects a footer byte to be sent. A footer byte has bit $7 = 0$, bit $6 = 1$ for mode 1, bit $6 = 0$ for mode 2, bit 5 set to 1 to differentiate this byte from a header byte. Bit 4 is reserved. The last 4 bits are also reserved.

A reserved bit is defined to be zero for design purposes.

3 Video Interface

Basically, a television monitor displays a serial data stream in a parallel format. A television displays data by means of a *raster scan* . The electron beam which fluoresces the phosphor (in the cathode ray tube = CRT), sweeps across and down the picture tube in a zig-zag fashion, starting from the upper left-hand corner. (Fig. 3). (The electron beam is steered by electro-magnets mounted on the picture tube).

In the figure, the heavy lines are the display. Since the vertical sweep is continuous, these lines are not quite horizontal. The light lines are the horizontal and vertical *retrace* . Horizontal and vertical *blanking* are usually applied during the respective retraces to prevent the display of unwanted data. Unlike an oscilloscope, vertical and horizontal sweep can not be triggered independently, instead the television "locks on" to synchronizing information encoded in the video signal.

One full vertical scan of the television screen defines a *field* containing 262.5 lines. (Vertical resolution may be doubled to 525 lines using interlaced fields, but 262.5 horizontal lines is sufficient

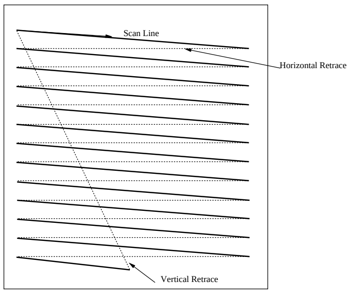

Figure 3: Raster Scan.

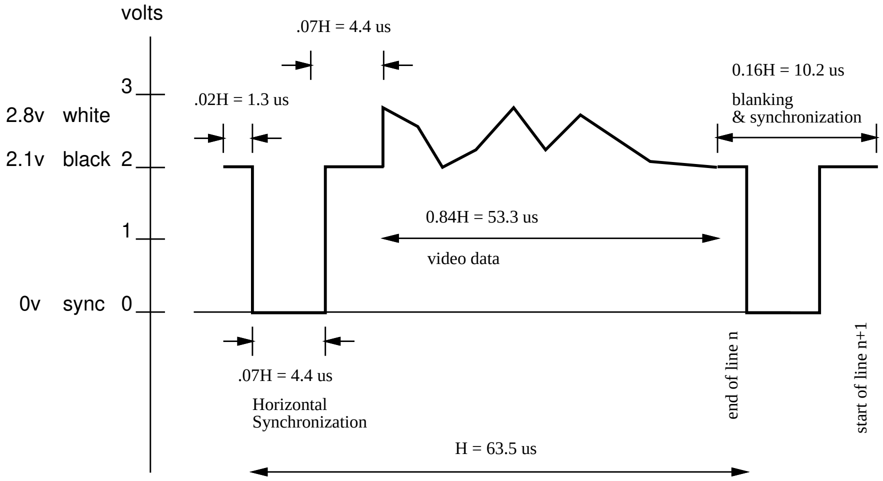

Figure 4: Horizontal Synchronization Timing. Note that only $53.3\mu s$ of the line is visible, the other $10.2\mu s$ is taken up by retrace and blanking time.

for our purposes). The field is refreshed (redisplayed) at 60 Hz.

The video signal generated by the camera is a *composite* video signal. That is, the video data (gray levels), horizontal and vertical synchronization information is all encoded on a single wire. The television locks on to the horizontal and vertical sync information in the video signal. Horizontal sync defines the beginning of a new line, and vertical sync defines the beginning of a new field. The horizontal sweep rate is 60 fields per sec $\times$ 262.5 lines per field or 15.75 KHz. The horizontal sweep interval is divided into three portions as shown in Fig. 4: blanking, synchronization, and data. Each line lasts 63.5 μs (= 1.0/15.75KHz). Retrace occurs during the blanking interval, and the leftmost visible data occurs at the end of the blanking interval. The horizontal synchronization signal occurs approximately in the middle of the blanking interval. Note that the video signal is analog, not binary, with levels from white to black, to synch.

It is slightly more complicated to understand how vertical synchronization is generated. Generation of the vertical synchronization signal is shown in Fig. 5. Since the vertical scan is much slower than the horizontal scan, a longer vertical blanking interval is required to avoid seeing the retrace lines. The example in the figure indicates a blanking time of 15 horizontal periods. How many horizontal lines will be visible? Approximately 247 (= 262 -15) lines. The standard blanking interval is between 18 and 21 horizontal periods. It is ok if your design misses a few lines on the top or bottom of the screen. A wide VSYNC pulse triggers the vertical oscillator in the monitor to start another cycle, however, the horizontal oscillator must be kept synchronized

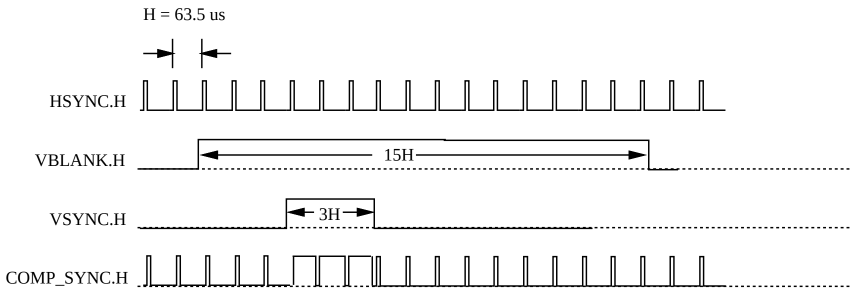

Figure 5: Approximate Vertical Synchronization Timing. (Please check your camera sync output to verify actual values.)

during this time. Hence the serrations in the COMP_SYNCH.H signal.

Figuring out where the top and left side of the image is could be complicated. Fortunately, you will be using the LM1881 sync separator in checkpoint 2. It takes as input the composite video signal, and provides outputs of vertical sync and composite sync. So to find the beginning of the video frame, wait about 17 horizontal periods after the vertical sync signal goes low. Then begin reading in video data about 8.8 μs after the composite sync goes low. Read a line, then wait for the next line until you have read all the lines you need.

3.1 Optical Flow

The design of the analog interface to the camera will be provided in checkpoint 2. Basically, you will obtain a rapid stream (about one byte every 200 ns or so) of 8 bit values which represent the gray level of a pixel in the camera. There will probably be noise, and depending on lighting levels, you will probably only obtain about 6 bits of usable data. For optical flow estimation, you need to examine the image data of each frame (at 60 Hz) and estimate the horizontal or vertical velocity at specified points. These specified points should be approximately evenly spaced on a horizontal or vertical line respectively in the center of the image. For example, with 230H by 210V and 7 vertical velocity components, vertical image velocity would be measured near [row][col] [15][115], [45][115], [75][115], [105][115], [135][115], [165][115], [195][115]. A small change of 2 or 3 pixels in position is acceptable if it simplifies the design.

You will be able to store many images in the RAM because of the symmetry of the images. For horizontal velocity estimates, there will be vertical symmetry. For vertical velocity estimates, there will be horizontal symmetry. A list of suggested references for optical flow estimation will be posted shortly to the class newsgroup. The basic idea of the image velocity detector is shown in Figure 6.

4 Project Deadlines

This section summarizes the functionality you need to demonstrate, as well as the steps in the design you need to follow. The number in parentheses is points out of 100 total for the project. Prelab assignments will only be checked off during your lab section during scheduled meeting times. You will only receive full credit for prelab if it is complete and checked off during your lab section. (Both partners need to show up to get individual prelab credit).

For checkpoints 1-4, there will be a formal design review with your TA during your scheduled lab section. You will need drawings and printouts on paper to hand in as part of a brief presentation

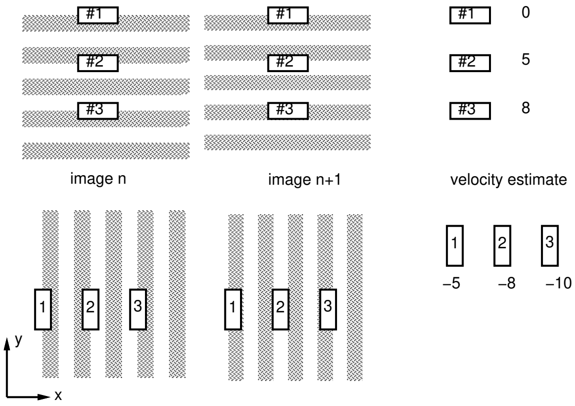

Figure 6: Local optical flow detection cartoon. The top figure shows vertical velocity detection at 3 spots ("3V") and the bottom half of the figure shows horizontal velocity detection at 3 spots ("3H"). The velocity estimates can be fairly crude due to noise in the image data.

on your progress and plans so far. Points are maximum, and will be based on completeness and quality of presentation as well as soundness of reasoning.

4.1 Checkpoint 1. Due in lab week of 19 Oct.

Prelab:

- (2) Detailed block diagram of data path and controller, (to level of Xilinx library components, counters, comparators, registers, etc).
- (2) State diagram for controller (initial attempt)
- (1) Project plan: division of labor, number of CLBs, testing strategy, milestones, etc.

Lab:

- (4) Demonstrate dumping RAM through serial port to PC.

4.2 Checkpoint 2. Due in lab week of 26 Oct.

Prelab:

- (3) State diagram for controller for project, complete.

Lab:

- (5) Demonstrate functioning A/D converter and video waveforms.

4.3 Checkpoint 3. Due in lab week of 2 Nov.

Prelab:

- (2) Schematics for data path

Lab:

- (4) Demonstrate video data storage in RAM.

4.4 Checkpoint 4. Due in lab week of 9 Nov.

Prelab:

- (2) Complete schematics for data path and controller. Simulation output for controller.

Lab:

- (5) Demonstrate capture a video frame in RAM, transmit over serial line to PC, and display on PC window.

4.5 Checkpoint 5. Due in lab week of 16 Nov.

Lab:

- (5) Demonstrate sending a correctly formed mode 2 packet, with vertical or horizontal line data, and correct number of velocity vectors. (For this checkpoint, velocity data vectors can be anything).
- Early Completion Bonus. +5 points if you demo the complete basic project during your scheduled lab section.

4.6 Project Week 11/23, 11/24, 11/25

- (45) Demo completed project. Sign up for demo slots during your scheduled lab section. (Thursday 5-8 lab section only will sign up, and demo by 5 pm Mon. 11/30.

4.7 Project Week 11/30

- (20) Final Report is due Friday 12/4/98 in lab along with your unwrapped Xilinx kit and all components. Project grade will be zero and check cashed if Xilinx kit is not returned by 12/4/98