Project Spec.

EECS150 Final Project, Fall 1997: Video Feature Tracking

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 a video feature tracker. The video feature tracker will acquire data from a black and white video camera, and digitize it. In the first mode of operation, the video data will be stored in a 64K byte RAM, then transmitted at 115.2 KBaud to a PC for display. In the second mode, "interesting" video features, such as a bright spot on a dark background, will be tracked from frame to frame. The coordinates of the video feature will be transmitted over the serial line to be displayed as a block superimposed on the image. It may be possible to use the visual tracker as a mouse replacement, e.g. for those who don't have use of their hands.

1.1 General Philosophy

This document describes the input/output specification for the video tracker, 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).
- 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:
   `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 feature tracking system scans through the entire image, and finds the image feature it is tracking in each frame. The algorithm used in this module is up to you to determine. A suggested simple algorithm is to just find the brightest pixel in each frame. This will provide a quick and dirty (and noisy) estimate of a feature.

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 Operation Mode 2: Video Feature Tracking

In mode 2, the system tracks a feature of interest, and 60 times per second sends the xy coordinates of the feature to the PC. The PC will superimpose a box on the image at the transmitted coordinates.

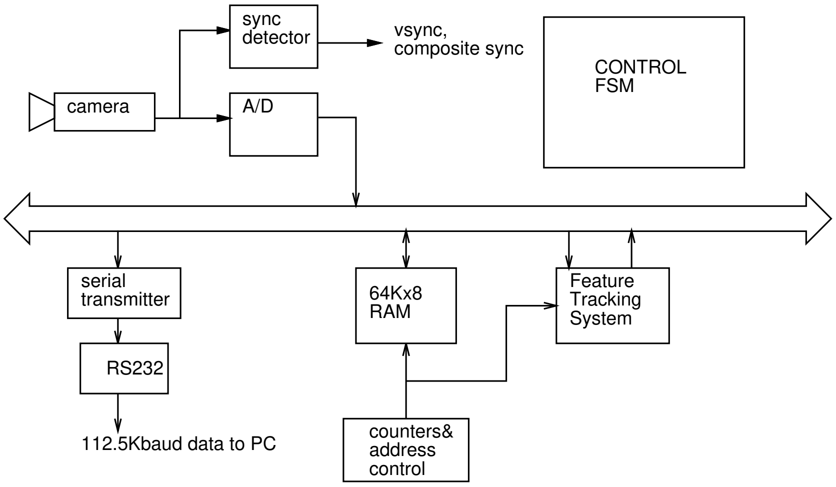

Figure 1: Top Level Block Diagram.

Digit Sum	Horizontal Size	Vertical Size	Code Byte	Digit Sum	Horizontal Size	Vertical Size	Code Byte
0 or 9 or 18	200	200	0x0	10 or 17	220	240	0x0a
1 or 8	200	220	0x01	11 or 16	240	200	0x0b
2 or 7	200	240	0x02	12 or 15	240	220	0x0c
3 or 6	220	200	0x03	13 or 14	240	240	0x0d
4 or 5	220	220	0x04				

Table 1: Frame Size Specification. There are nine different video frame sizes, based on student ID number. The code byte tells the PC which frame size you are using.

2.3 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 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.

2.4 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 other bytes in the packet have MSB=1). Bit 6 is 1 for mode 1, and 0 for mode 2. In frame dump mode (mode 1) the next 40,000 (for 200×200) to 57600 (for 240×240) 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). The PC then expects the next byte sent to be a header byte.

In mode 2, tracking mode, bit 6 of the header byte is 0. The next two bytes give the x and y position of the found feature. The feature position is specified by 6 bits in each dimension. This accuracy is plenty for coarse user interface applications. In tracking mode (mode 2), this completes the packet. The PC then expects the next byte to be a header byte.

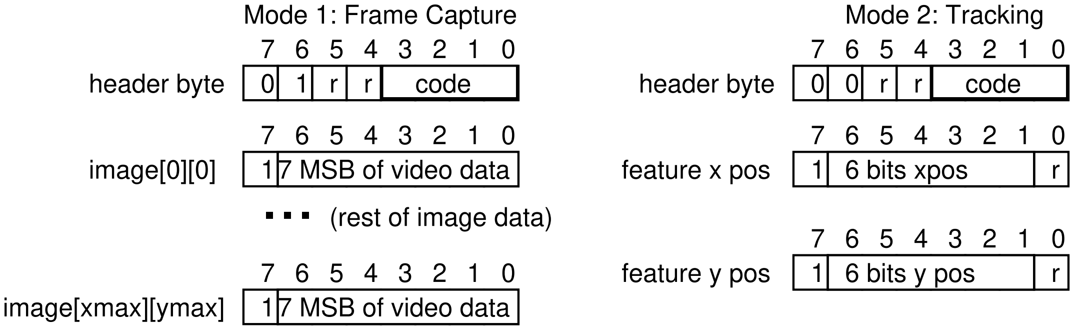

Figure 2: Serial transmission protocols for mode 1 and mode 2. The MSB must be 1 except in a header byte. r is reserved for extra features.

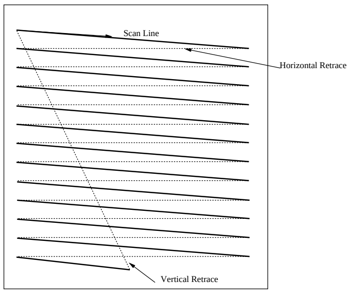

Figure 3: Raster Scan.

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 flouresces 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 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 frame. 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

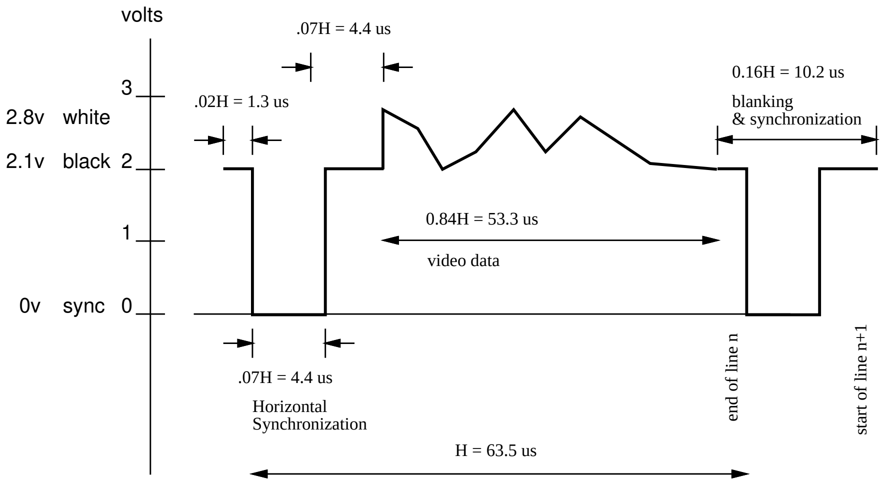

Figure 4: Horizontal Synchronization Timing. Note that only $53.3\mu s$ of the line is visible, the other $10.2\mu s$ is taken up by retrace and blanking time.

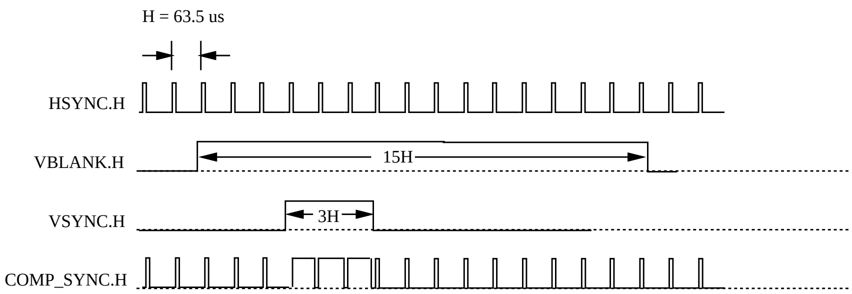

Figure 5: Vertical Synchronization Timing.

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 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 Feature Tracking

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 5 bits of usable data. For feature tracking, you need to examine the image data of each frame (at 60 Hz) and find the feature you are trying to track. The feature tracking may or may not need to use the image values you store in the RAM. For example, you might look at just the current input image and find the maximum. Then you would not need the previous frame. Alternatively, you might subtract the current frame from the previous frame and look for the biggest change. A list of suggested text references to feature tracking will be posted shortly to the class newsgroup. We would like to see a feature tracked to 6 bits of precision in x and y.

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.

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 as part of a brief presentation 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 24 Oct.

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

4.2 Checkpoint 2. Due 31 Oct.

- (3) State diagram for controller for project, complete.
- (5) Demonstrate functioning A/D converter

4.3 Checkpoint 3. Due 7 Nov.

- (2) Schematics for data path
- (4) Demonstrate video data storage in RAM.

4.4 Checkpoint 4. Due 14 Nov.

- (2) Complete schematics for data path and controller. Simulation output for controller.
- (5) Demonstrate capture a video frame in RAM, transmit over serial line to PC, and display on PC window.

4.5 Checkpoint 5. Due 21 Nov.

- (5) Demonstrate finding a feature, such as maximum brightness, in video frame.
- Early Completion Bonus. +5 points if you demo the complete basic project during your scheduled lab section.

4.6 Project Week 11/24, 11/25, 11/26

- (45) Demo completed project. Sign up for demo slots during your scheduled lab section.

4.7 Project Week 12/1 12/2

- Thursday 5-8 lab section demo completed project. Sign up for demo slots.
- (20) Final Report is due Friday 12/5/97 in lab along with your unwrapped Xilinx kit and all components.