CSEE 4840 – Embedded systems Lab
Project proposal – Object tracker
[OBTRAK]

Date: 2/19/2004

Project Team Members:
Anuj T Maheshwari (atm2104@columbia.edu)
Marcio O. Buss (marcio@cs.columbia.edu)

Anuj T. Maheshwari
Marcio O. Buss

Table of Contents

Table of Figures

Anuj T. Maheshwari
Marcio O. Buss

1. Introduction

Image and pattern recognition has been an area of much research and development in recent years. An immaculate number of complex software have been written to get the best level of pattern recognition. The applications of this type of software range from security systems to simple forms of targeting in missile launch systems. The basis for this project are primitive forms of pattern recognition and possibly if time permits color distinction.

The initial part of this document (sections 1 – 4) gives you a brief idea about the specifics of the project proposed. The latter part of the document (sections 5 – 7) delves deeper into the actual mechanics of the project giving insights on the various project goals and methodology used to accomplish each one.

2. Project description

The setup would involve the FPGA board connected to a camcorder streaming digital images, the software would grab a frame and attempt to look for either one pattern or a set of patterns or colors. If a match is found, the result is displayed on the screen. Particularly, it is shown what object/pattern was found and its current position

The pattern match is done by storing the search pattern in memory or by having an equation describing a geometric form (in the simple case of searching for circles, squares, etc.). The frame captured from the digital video is loaded into memory, and the real processing begins. More precisely, a sequential search for the best match is performed starting in the initial address of the memory where the image is stored. The final address is used to stop the search. Few optimizations in the search algorithms will be tried aiming to speed up the entire process.

3. Project goals

The didactic goal of the project is to get familiar with the board's video processing capabilities. The operational goal, however, can be stated as:

- Tracking a white square of a fixed size on a black background.

Additional goals depending on resource and timing constraints are listed below. The realization of these goals will be subject to the time constraints as well as the limitations of the hardware being used. The following targets would mean simple trivial extrapolation of the main algorithm that would be coded to obtain the first goal. All efforts will be made to achieve as many of these goals as possible:

- To track other objects such as a circle and ovals
- To track the objects against any non controlled backdrop
- To track multiple objects at a time
- To track objects of different color
- To be able to track multiple objects with colored backgrounds

4. Feasibility approximations

Memory required for 8 bit color: $1024 \times 768 \times 8 = 6\,\mathrm{mb}$ per frame stored in memory
On board SDRAM would more than suffice this memory requirement. The memory needed for swap space too would be accounted for on this chip.

A higher resolution and/or definition is prohibitive for this class project, as the image frame would have to be spread over multiple chips and the system's overall complexity would be substantially increased.

5. Project methodology

This part of the proposal will describe in some detail how we intend to achieve all the above mentioned goals

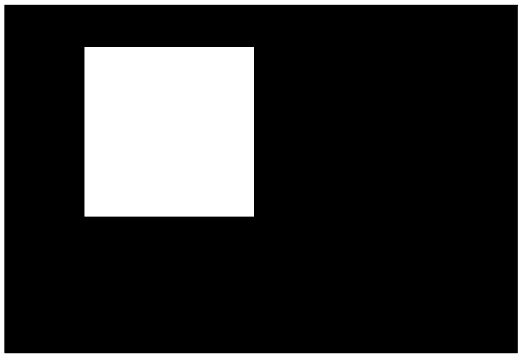


Figure 1 : Tracking a Square, original image input

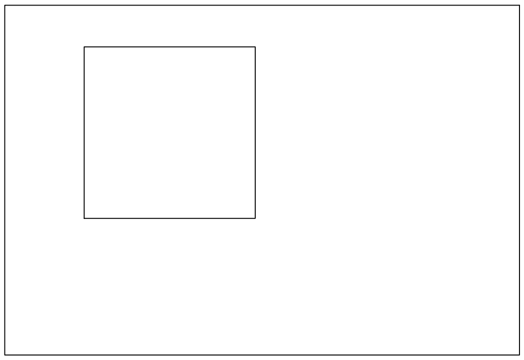


Figure 2 :Image to Monitor

§ **Tracking a while square of fixed size on a black background:** This scenario suggests that each pixel needs only one bit to represent its color. In this case, a match is obtained when a grid of consecutive zeros or ones is found. A continuous addition using an accumulator enables us to verify such condition (the sum remains close to zero when we analyze a series of zeros, for instance). An assumption is made here: the object is not moving too fast and it is also reasonably large. What "not too fast" and "reasonably large" will be determined by practical tests. Alternatively, a basic traversal through an array representing the image, looking for a color consistency, can be also applied (See section 6). Both algorithms have been tested using Matlab and the results were satisfactory.

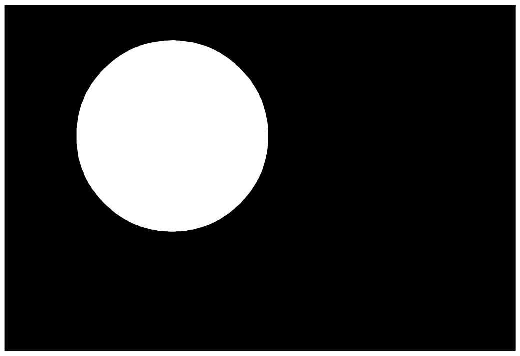


Figure 3: Tracking a circle, input

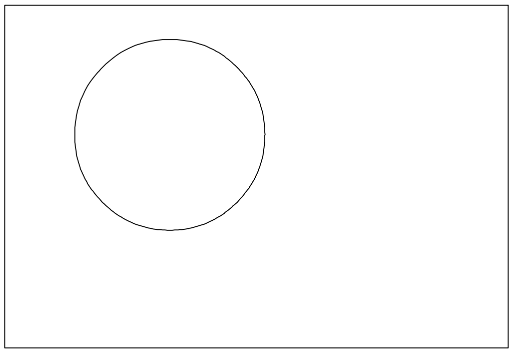


Figure 4: Tracking a circle, output

§ **Tracking other objects like circles and ovals:** Such scenario requires a heavier processing as it needs to recognize different geometric forms. Moreover, it demands several traversals over the image (Figures 3 and 4), which slows down the tracking process significantly. Once the initial search has been matched, and the particular object has been identified, the assumption that the same object is moving relatively slow helps to speed up subsequent matching.

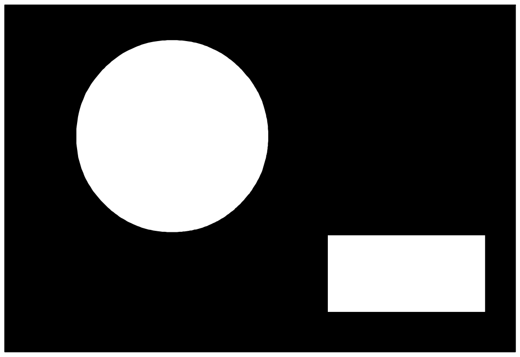


Figure 5: Tracking multiple objects, input

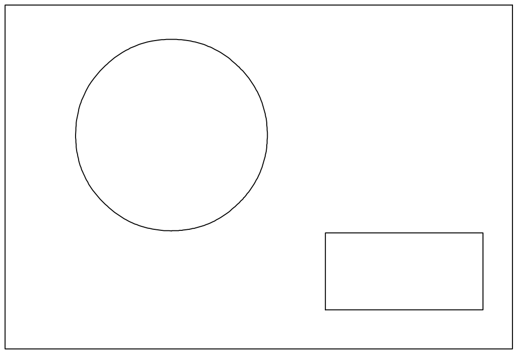


Figure 6: Tracking multiple objects, output

§ **Track multiple objects:** For this goal, the code needs what one can call an "extra sensitivity", as the assumption that only one object is to be matched is not valid. More precisely, the algorithm has to be invoked recursively in a same frame in order to find all matches. Once the matches are found, the last matches and their positions are kept in memory so that future matches can be made faster (Figures 5 and 6).

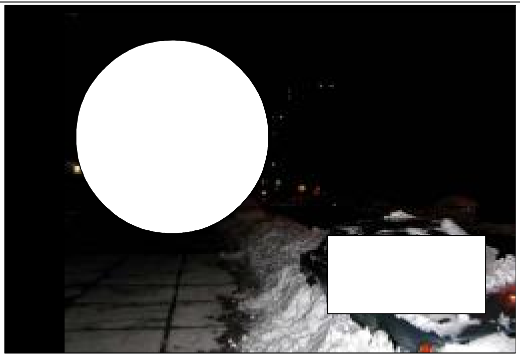


Figure 7 Tracking multiple white objects, input

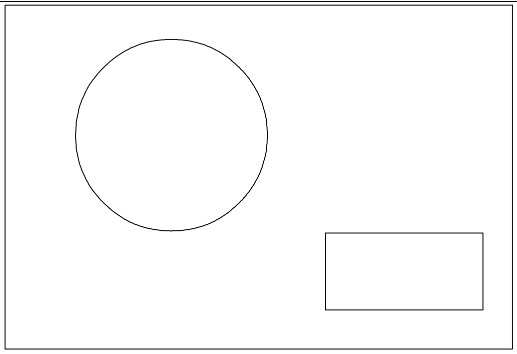
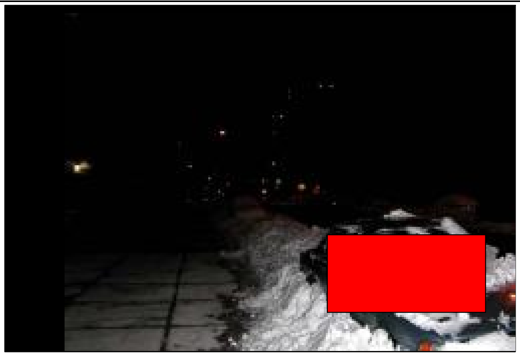
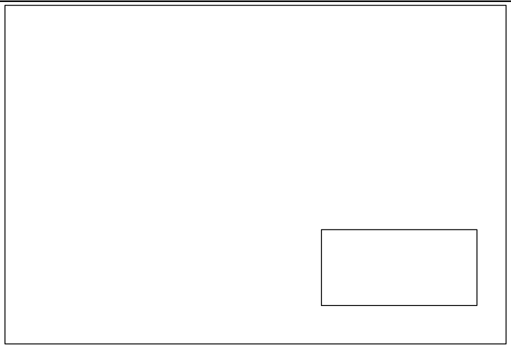
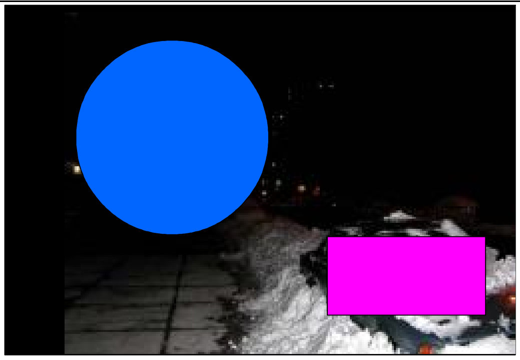
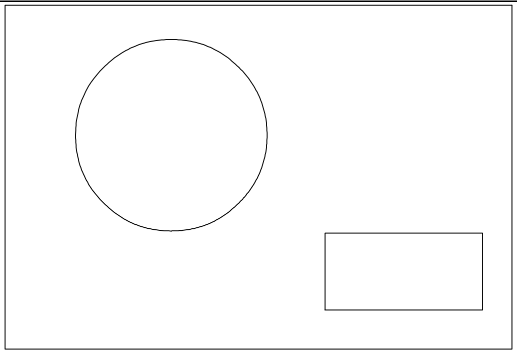


Figure 8 Tracking multiple white objects, output

§ **Track any objects against a non controlled background:** In such a case, a hybrid algorithm is used which optimizes traditional "traversal" pattern matching: selective continuous addition needs to be added to the software. Particularly, since the objects we track are still of a single color, we are still able to exploit the accumulator method described earlier. The difficulty here will rely on the system's performance.

 **Figure 9 Tracking colored object, input**	 **Figure 10 Tracking colored object, output**

§ **Track objects of different color:** The only additional constraint for this goal is memory, as the only change is the number of bits needed to storage color information (instead of black-and-white). Looking for one object with a particular color does not alter the main algorithm. The basics of "shape and color consistency" invariants remains the same. As described in the project feasibility section, the board should be able to deliver good results with 8-bit color.

 **Figure 11 Tracking multiple colored objects, input**	 **Figure 12 Tracking multiple colored objects, output**

§ **Tracking multiple colored objects with non controlled background:** This would be the ultimate (and possibly not realizable) goal of the project. Further details will be added if the possibilities for such goal become concrete.

6. Program pseudo code (Software)

In the following section, the pseudo code for searching for a basic white rectangle (dimension M x N) in a frame is described (image is monochrome, with 1 signifying black, 0 signifying white):

```
Step 01   Grab image from video processor / camcorder, perform resizing / color
          manipulations
Step 02   Store the image to memory (into a A x B matrix)
Step 03   Set counter TOTAL = 0
Step 04   Set counters X = 1, Y = 1, MATCH_X=0, MATCH_Y=0
Step 05   G = X, H = Y
Step 06   TOTAL = TOTAL + CAM_IMAGE[G,H]
Step 07   If G < (X+M),  G = G+1; JUMP to Step 06
Step 08   If H < (Y+N), H = H+1; Jump to Step 06
Step 09   If TOTAL = 0 (all locations 0) MATCH_X = X, MATCH_Y = Y; JUMP to Step 12
Step 10   If X < (A-M) X = X+1; JUMP to Step 05
Step 11   If Y < (B-N) Y = Y+1; JUMP to Step 05
Step 12   If MATCH_X !=0 PRINT “MATCH FOUND AT” MATCH_X , MATCH_Y
```

An alternate implementation of this algorithm where the dimension of the square is unknown is given below. First an assumption for the minimum dimension to be considered a square must be made. Let it be an integer ‘MIN_DIM’. The following algorithm would be used thereafter:

```
Step 01   Grab image from video processor / camcorder, perform resizing / color
          manipulations
Step 02   Store the image to memory (into a A x B matrix, called CAM_IMAGE)
Step 03   Set counter TOTAL = 0 , DIMENSION = 0 , T = MIN_DIM
Step 04   Set counters X = 1, Y = 1, MATCH_X=0, MATCH_Y=0
Step 05   G = X, H = Y, T = MIN_DIM
Step 06   TOTAL = TOTAL + CAM_IMAGE[G,H]
Step 07   If TOTAL = 0 GOTO Step 08; ELSE GOTO Step 13
Step 08   If ((G-X) < T) THEN G = G + 1, GOTO Step 06 ; ELSE GOTO STEP 09
Step 09   If ((H-Y) < T) THEN H = H + 1, GOTO Step 06 ; ELSE GOTO STEP 10
STEP 10   MATCH_X = X, MATCH_Y = Y, DIMEMSION = T
Step 11   T = T + 1, GOTO STEP 08
Step 12   If TOTAL = 0 (all locations 0) MATCH_X = X, MATCH_Y = Y; GOTO Step 15
Step 13   If X < (A-T) THEN X = X+1; GOTO Step 05

Step 14   If Y < (B-T) THEN Y = Y+1; GOTO Step 05

Step 15   If MATCH_X !=0 PRINT “MATCH FOUND AT” MATCH_X , MATCH_Y, DIMENSION
```

The algorithms above could be altered to look for a color by making a change to the test where the `TOTAL` variable is checked to be zero (Step 6). Instead of adding up the values of the pixel colors, it should be repeatedly checked against the appropriate color code.

7. Project timeline

This part of the report provides a rough time schedule for the completion of this project

Task 1: Program coding design
Task 2: Hardware interfacing design
Task 3: System implementation
Task 4: System testing and debugging
Task 5: Project report and demo preparations

Week à	03/07	03/15	03/22	04/01	04/08	04/15	04/22	05/01
Task 1	■	■	■					
Task 2		■	■	■				
Task 3				■	■			
Task 4					■	■	■	
Task 5							■	■

8. Bibliography

2001, Atmel corporation, "4 Megabit (512K x 8) Single 2.7 Volt Battery Voltage™ Flash memory" Revision 0679D, (http://www.atmel.com/dyn/resources/prod_documents/DOC0679.PDF)

2003, XESS corporation, "XSB Board V1.0 Manual" Version 1.0 (https://host18.webserver1010.com/xess/manuals/xsb-manual-v1_0.pdf)

2000, Philips semiconductors, "SAA7114H PAL/NTSC/SECAM video decoder with adaptive PAL/NTSC comb filter, VBI-data slicer and high performance scaler" (http://www-us.semiconductors.philips.com/acrobat/datasheets/SAA7114H_1.pdf)

1998, Toshiba corporation, "Toshiba MOS digital Intergrated Circuit Silicon Gate CMOS 262,144 – word by 18 Bit CMOS Static RAM TC55V16256J 256K X 16 SRAM" (http://www.toshiba.com/taec/components/Datasheet/55v16256.pdf)

2003, Samsung Electronics, "128Mb E-die SDRAM Specification" Revision 1.2 (May 2003) (http://www.samsung.com/Products/Semiconductor/DRAM/SDRAM/SDRAMcomponent/128Mbit/K4S281632E/SDR%20128Mb%20E-die%20x4x8x16%20rev%2012.pdf)