MCGILL UNIVERSITY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

ECSE 457 - FINAL REPORT

Research & Development of a Real-Time Object Tracking System

Authors:

Benjamin BROWN
benjamin.brown2@mail.mcgill.ca
260450182

Taylor DOTSIKAS
taylor.dotsikas@mail.mcgill.ca
260457719

Supervisors:

Warren GROSS, PROF.

Arash ARDAKANI

Sponsored by:

ANALOG DEVICES

Abstract

This project aimed to research object tracking algorithms, and implement the algorithm best suited to meet project requirements in both software and hardware. The software implementation used static input videos and the hardware implementation used a real-time input video stream. Phase 1 of the project focused on algorithm research and software implementation, and is documented in [**?**]. Phase 2 of the project focused on hardware implementation, and this report encompasses all of Phase 2 from the design process to the final results. It was found that hardware implementation of the algorithm provides a much faster way of tracking an object when compared to the equivalent software implementation. It was also found that using a background subtraction, delta frame based algorithm to determine object position is not the best choice for real-time situations, due to the inability to cope with lighting and background variations. It was also found that while a Kalman filter does greatly improve object tracking results, it comes at a heavy cost in terms of hardware utilization.

Acknowledgments

We would like to thank Professor Warren Gross and Arash Ardakani for overseeing this project and providing advice, insight, and direction over the course of the year. We would also like to thank Analog Devices for providing us with the Altera DE2 board needed for hardware implementation, and in particular Leah Magaldi for being our main point of contact. Finally, the example TV decoder Verilog code provided by Terasic was a major asset for hardware implementation, we would like to thank them for making it open and available to developers like ourselves.

Contents

1 Abbreviations & Notation

FPGA - Field Programmable Gate Array

VGA - Video Graphics Array

2 Introduction

The ability to determine the position of an object in a scene is a highly relevant challenge for industries like surveillance and robotics. This task represents a challenge from an engineering perspective due to the real-time timing constraints placed on these systems, and the large amount of data that must be processed when working with video data. In order to overcome these challenges, custom hardware implementation has become an increasingly popular solution as demonstrated in [**?**], [**?**], and [**?**].

3 Background

This section contains the prerequisite information regarding video tracking needed to understand the system architecture and design. For background information regarding the basics of video processing, Kalman filtering, fixed-point representation, and optical flow, please see [**?**].

3.1 Moving Average Filter

A moving average filter replaces the current input data sample with a mean of some number of past input data samples. The number is referred to as the moving average filter length, N. This type of filter is particularly useful when data samples are arriving in a time series (i.e. at constant time intervals) as it will remove outliers, creating a smoother trend in data samples. Given that p_i is the current input data sample, the filtered results, p'_i, is given using the following equation,

$$p'_i = \frac{\sum_{k=i}^{i+N} p_{k-N}}{N} \tag{1}$$

3.2 Saturation Filter

A saturation filter, in the context of this project, refers to either flooring a grayscale pixel intensity to a minimum value or ceiling the grayscale pixel intensity to a maximum value, if it is below or above a threshold. The pseudo-code for the saturation filter is given below.

```
function [output] = saturation_filter(input, min, max, thresh)
    if (input > thresh)
        output = max;
    else
        output = min;
    end
end
```

This makes the data set binary, in the sense that all values are one of two values, and makes data processing significantly easier. In the context of object tracking, values above the threshold indicate an object present in the scene, and values below indicate no object present.

3.3 Determining Position

The algorithm implemented in software, and presented in [**?**], for determining the (x, y) position of an object in the delta frame used a rastor scan technique to determine the leftmost, rightmost, top, and bottom pixels. By intersecting two lines formed between these points, the center of the object can be estimated. It was found that despite the success of this algorithm in software, it would not be conducive to hardware implementation. This is mainly due to the fact that in the software implementation, data arrived in discrete frames from MATLAB's `VideoReader` class at constant time steps. In hardware implementation, data is handled at the pixel level, with a constant stream of pixels being placed in and extracted from a FIFO frame buffer storage module (to be discussed later). Thus using an algorithm that requires an entire frame requires more memory and extra logic. A new algorithm was developed to determine the local of the object.

This algorithm assumes that pixel data arrives in binary format; either a string of all zeros representing no object present or a string of all ones representing an object present. Note that this data format is achieved by using the saturation filter just discussed. The algorithm tests if the pixel is an object pixel or not, and if it is the x and y coordinates of the pixel are each added to a rolling summation (x_{sum} and y_{sum}). A counter, n, is also incremented At the end of the frame, the rolling summations are divided by the counter, the position is outputted, and the three values are cleared. This algorithm is essentially just taking an average of the (x, y) coordinates of the object pixels as the center of the object.

$$x = \frac{x_{sum}}{n} \tag{2}$$

$$y = \frac{y_{sum}}{n} \tag{3}$$

Note that this algorithm achieves best results when the object is highly symmetric.

3.4 Video Pipeline

The phrase *video pipeline* refers to a series of image processing modules that exist between the video input device (e.g. the camera) and the video output device (e.g. the display). The pipeline can be implemented in software or hardware, but for the scope of this report the pipeline will refer to hardware implementation, and modules will be referred to in the Verilog sense. Modules in the video pipeline generally consist of decoding and encoding the video data into various formats, and performing video processing (e.g. applying algorithms of interest) in the middle. Implementing a video pipeline is directly coupled with implementing an object tracking algorithm in hardware, as there are no libraries or classes to convert and store the incoming video data when working at this low of a level.

3.5 The VGA Interface

Discuss the timing signals

4 Requirements

5 Design

5.1 Generating VGA Output

Make sure mention the design references for this.

5.2 Simple Video Pipeline

This is where you should cite [**?**] and talk about the video input wrapper module.

5.3 Storing the Base Frame

This is where you should cite [**?**] and talk about the SRAM.

5.4 RGB to Grayscale Conversion

Simple discussion here.

5.5 Modified Video Pipeline

Talk about moving the RGB conversion further upstream prior to the SDRAM such that we can work in RGB.

5.5.1 Delta Frame Generation

Discuss the poor man's absolute value implemented and the saturation filter.

5.5.2 Moving Average Filter

Show the two pictures of the before/after moving average filter here.

5.6 Measuring Object Position

Make sure to mention it's using the VGA module counters.

5.7 Kalman Filter

Lots of meat here.

6 Future Work

The system could be significantly improved by using a measurement algorithm that has fewer requirements and assumptions regarding the environment. Variations in lighting and changes in the background cause the delta frame generation to break, and in a real-time situation we found it difficult to ensure that all the requirements were met.

7 Impact on Society

This topic was discussed in the Phase 1 report, and there were no changes in the project scope during Phase 2 that would change the impact of society. Please see [**?**] for the discussion.

8 Allocation of Work

9 Conclusion

The final hardware implementation was a success, and the system can display a red dot that shows the (x, y) measurement of the object in the frame as well as green dot showing the improved (x', y') position the Kalman filter produces. Due to the underlying principles of the delta frame generation, the system is an object tracking algorithm and not a motion tracking algorithm. However, the Kalman filter theoretical model assumes constant velocity.This is why when the object stops moving, the green dot slowly converges to the red one. This shows the system favoring the measurements over the model to reduce error, and demonstrates the Kalman filter functioning properly. When the object is moving with constant velocity, the Kalman filter provides a smoother, improved output.

References

[1] F. Roth, “Using low cost FPGAs for realtime video processing”, M.S. thesis, Faculty of Informatics, Masaryk University, 2011.

[2] A. Saeed et al., “FPGA based Real-time Target Tracking on a Mobile Platform,” in 2010 International Conference on Computational Intelligence and Communication Networks, 2010, pp. 560-564.

[3] ”Video and Image Processing Design Using FPGAs.” Altera. 2007. January 2014. http://www.altera.com/literature/wp/wp-video0306.pdf

[4] E. Trucco and A. Verri, “Chapter 8 - Motion,” in Introductory Techniques for 3D Computer, pp. 177-219.

[5] S.A. El-Azim et al., “An Efficient Object Tracking Technique Using Block-Matching Algorithm”, in Nineteenth National Radio Science Conference, Alexandria, 2002, pp. 427 - 433.

[6] Caner et al., “An Adaptive Filtering Framework For Image Registration”, IEEE Trans. Acoustics, Speech, and Signal Processing, vol. 2, no. 2, 885-888. March, 2005.

[7] Yin et al. *Performance Evaluation of Object Tracking Algorithm* [Online]. Available: http://dircweb.kingston.ac.uk/

[8] G. Shrikanth, K. Subramanian, “Implementation of FPGA-Based object tracking algorithm,” Electronics and Communication Engineering Sri Venkateswara College of Engineering, 2008.

[9] E. Pizzini, D. Thomas, “FPGA Based Kalman Filter,” Worcester Polytechnic Institute, 2012.

[10] M. Shabany. (2011, December 27). *Floating-point to Fixed-point Conversion* [Online]. Available: http://ee.sharif.edu/ digitalvlsi/Docs/Fixed-Point.pdf

[11] N. Devillard. (1998, July). *Fast median search: an ANSI C implementation* [Online]. Available: http://ndevilla.free.fr/median/median.pdf

[12] D. Kohanbash. (2014, January 30). *Kalman Filtering - A Practical Implementation Guide (with code!)* [Online]. Available: http://robotsforroboticists.com/kalman-filtering/

[13] Recommendation ITU-R BT.656-5 (12/2007). “Interface for digital component video signals in 525-line and 625-line television systems operating at the 4:2:2 level of Recommendation ITU-R BT.601”.

[14] ISSI Datasheet: “1M x 16 High-Speed Asynchronous CMOS Static RAM with 3.3V Supply”.

[15] B. Brown & T. Dotsikas, “A Real-Time Object Tracking System”, ECSE 456 Final Report, unpublished.

Appendices