Senior Project – Computer Engineering – 2020

Sign Detection System for Real Time Applications

Brock Harris, Will Christensen

Advisor – Prof. Cherrice Traver

Introduction

Cameras are an integral sensor to cars and need to be able to gather useful data quickly and efficiently in order to make split second decisions on the road. The objective of this project is to use hardware software (HW/SW) co-design to efficiently detect road signs from real time camera video input.

Current Design

Algorithm: We chose to use the Histogram of Oriented Gradients (HOG) feature detector, implemented on an FPGA, and a Support Vector Machine (SVM) implemented on an ARM hard processor.

We will be using a HW/SW System on a Chip (SoC). This board allows us to implement software on the onboard ARM chip and also hardware on the FPGA.

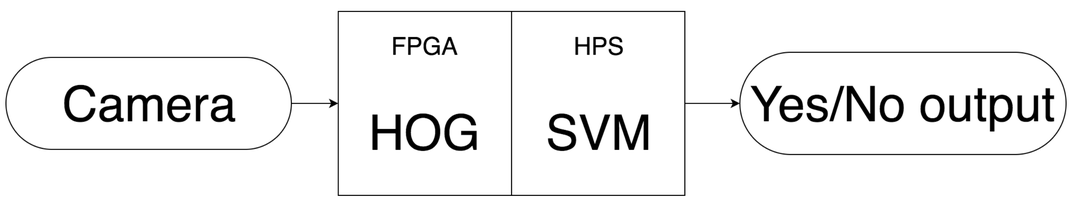


Figure 1: Top level design overview

Algorithm Overview

Preprocessing:
- Get the dominate color vector for each pixel
- Get magnitude and orientation for gradients over the image

HOG:
- Collect the gradients together for 8x8 pixel cells
- Vector normalization
- Compress the vectors to save space

SVM:
- Classification of images based on training

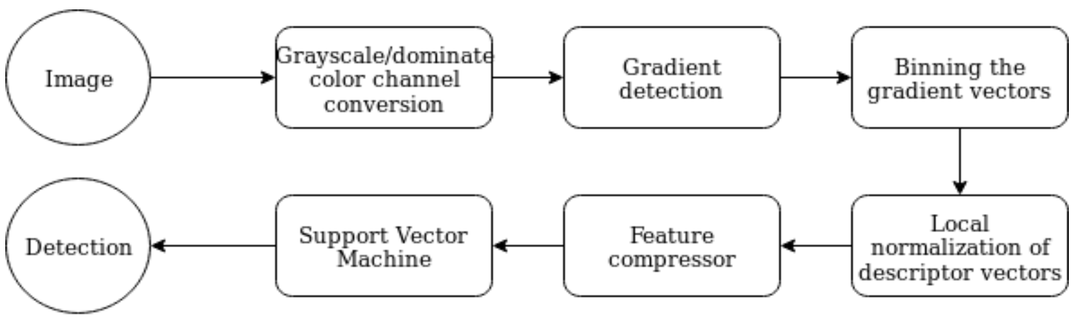


Figure 2: Flow diagram for proposed algorithm

Design Requirements

- Classify standard stop signs in 720x1280 (HD) images
  - Output a true if image contains a stop sign, false otherwise.
  - Must work on all types of American stop signs (with graffiti, discoloration)
- Run detection in under 33.33ms (Real time for 30fps camera)
  - From camera to output.
  - HOG must run in under 20ms
- Must have an accuracy of at least 80%
  - Minimum false negatives (High Recall)
  - False positive are not as important

Figure 3: Example image used for training/detection

Design Alternatives

Histogram of Oriented Gradient (HOG)/Scale Invariant Feature Transform (SIFT):
- HOG: Cons: Used mostly for people detections
  Pros: relatively easily implemented in hardware
- SIFT: Pros: is used mostly for feature tracking
  Cons: Harder to implement in Hardware

Support Vector Machine (SVM)/K-Nearest Neighbors (KNN):
- SVM: Cons: A little harder to implement
  Pros: Very fast and effect at detecting HOG features
- KNN: Pros: A little simpler and easier to implement
  Cons: We predict that it won't be as accurate

RGB/ LAB/YUV:
- RGB: Pros: Most common color scheme
- LAB: Cons: Does not really improve detection rate
- YUV: Cons: Does not really improve detection rate

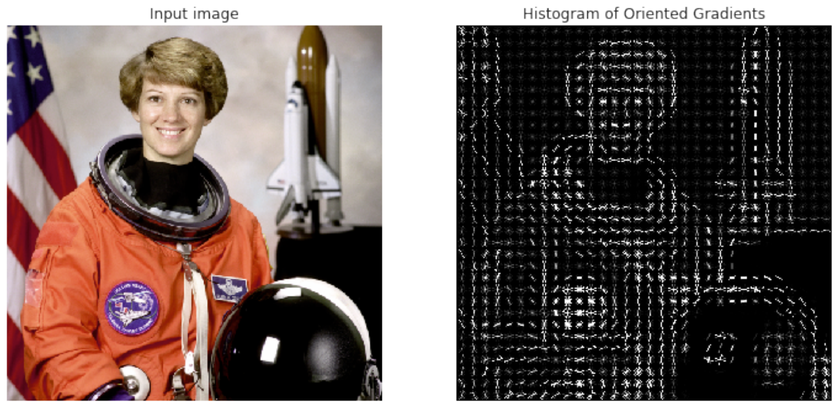


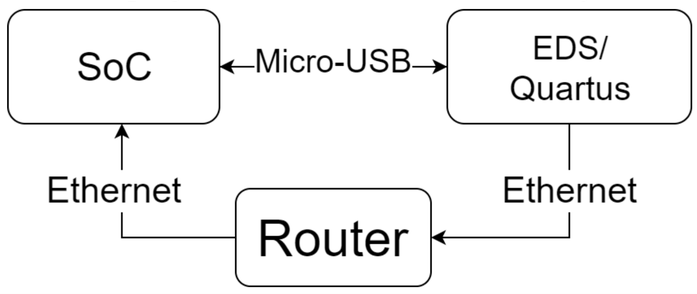
Figure 4: Example of HOG algorithm input/output

Hardware-Software Co-design

Quartus: Software used to design programs for the FPGA section of the SoC. We have implemented a simple project on the FPGA to prove that we can use the FPGA.

EDS: Design environment that allows us to compile C code that can be run on the Arm processor on the SoC board. We have implemented a simple project to prove that we can use the Hard Processor System (HPS)

Qsys: Software to allow for communication between the FPGA and the HPS. This allows us to assign data to buses and transfer the data across clock domains. We plan to implement a test project to show the connection by the end of the term.



Next Steps

- Simple Qsys project: To show how we will connect the two aspect of the project.
- Input/output block diagram: A diagram that shows in more detail how each of the component will be connected and what data will be transferred between blocks
- Hardware implementation of the HOG algorithm: We will implement camera communication, and the full HOG algorithm in hardware, which will increase the efficiency of the algorithm.
- Implement and train the SVM: We have a dataset to train on.
- Connect the hardware and the software: Use Qsys to connect the two systems and test on real time video.

Acknowledgments

Prof. Shane Cotter[+] Prof. Matt Anderson[*] Prof. John Rieffel[*] Prof. John Spinelli[+]

[*] Computer Science Department [+] ECBE Department