CSEE4840: Embedded Systems Design Document

Stereo Depth Extraction (SDE)

March 31, 2004

Department of Electrical Engineering
School of Engineering and Applied Science
Columbia University

Team Members:
Ang Cui (ac2024)
Jeng-Ming Hwang (jh2026)
Yen Yen Ooi (yo2006)
Kashif Siddiqui (kms2010)
Ting-Hsiang Wu (tsw2008)

Table of Contents

Introduction …….. 3

Purpose and Scope of Document

Design Overview

Architecture Design …….. 4

Required Equipment

Explanation of Alias Render

System Architecture …… 5

Block Diagram …… 6

Software Architecture …….. 7

Pseudo code for major components
Light Scanning
3D extraction

Introduction

Purpose and Scope of Document

Our project will utilize an ordinary video camera to extract 3D coordinates from a single light source in a dark background. Using the formulas of stereo-depth extraction and a setup of mirrors augmented to the video camera, we will track a light source, and in real-time output a stream of 3D coordinates into either the onboard CF card, or through MINICOM.

Possible Applications: We can make use of the stereo-depth extractor to construct an alarm system that can detector an object that is within a certain radius from our device. This can be used in security systems or even in defense departments.

Design Overview

We will extract the image of a point of light from a digital video camera. We will process the video signal with the onboard Philips video decoder chip (SAA7114H) and store it in a video buffer (SRAM). We then proceed to locate the position of the light source by scanning through the video buffer and extract the 2D coordinates of the light source from each of the two sections of the video buffer. By using an optimized depth extraction formula coded in C, we will compute a 3D coordinate of the light source. The collected data can then be exported to any 3D modeling program, or our own real-time visualization application. As an extra feature, we will experiment with tracking multiple light sources.

Architecture Design

Required Equipment

- Video Camera
  - with S-Video NTSC output
- Periscope apparatus (self-constructed)
- Camera tripod
- Video Capture card
  - to import test data into our own PC to ensure that our video calculations are accurate by using points that we manually find using the pixel locator of a graphics program

Explanation of Alias render

Please refer to attachment

This rendering demonstrates the process by which stereo input will be taken in. The video camera is mounted on a regular commercial tripod. Attached to the lens of the camera is a self-constructed periscope. The periscope consists of 2 pairs of mirrors mounted on a 45 degree angle to simulate a stereo input. The periscope itself is mounted via braces that attach to the tripod. This setup enables a full range of motion for the light source, thereby making our experiment more realistic.

System Architecture

Refer to Block Diagram on page 6

1. Philips S447114H decoder

This is the decoder that is built into our Xilinx board. It will handle the S-video input into the board from the Digital 8 camcorder. The input signal will be in NTSC at a resolution of (427 x 242).

2. Video Buffer in black/white (427 x 242 x 4)

The Video Buffer that we plan on using will be coded in hardware to enable fast access to the memory locations. We decided to go with black and white to cut down the amount of processing and minimize the input bandwidth. The SRAM is suitable for our purpose.

3. VHDL Light Recognizer

We decided to code the Light Recognizer in VHDL to enable fast computation of the presence of light.

4. Tangent Function Lookup Table (91 x 10 x 10)

The Tangent Function lookup table will be used to minimize the number of floating point computations done in our algorithm. This is an important consideration because it will enable the possibility of real time modeling.

We plan on populating the 900 lines of this table by writing a program that will calculate the tangent values from 0 to 90 degrees at 0.1 degree intervals. These values will then be stored in the lookup table. Intermediate values will be interpolated using the closest 0.1 degree values. A weighting mechanism might be employed to ensure that our estimated tangent values is a feasible value. The SDRAM will be used to store the trigonometric values.

Software Architecture

Pseudo code for major components

- Light Scanning

1) Search for light

```
Input: None
Output: 1 x 2D coordinate

for(w) {
  for{h) {
    if(has_light_around_point) {
    return coordinate
    }
  }
}
```

2) Track light

```
Input: 1 x 2D coordinate for last known light position.
Output: 1 x 2D coordinate for current light position.

x_old, y_old = 2D coordinate of known.

from x_old - scan_radius to x_old + scan_radius
from y_old - scan_radius to y_old + scan_radius

if(has light around current position)
   return current position
```

3) Save light coordinate

```
Input: 2 x 2D coordinate
Output: None

Write_Data(cord1, loc1)
Write_Data(cord2, loc2)
```

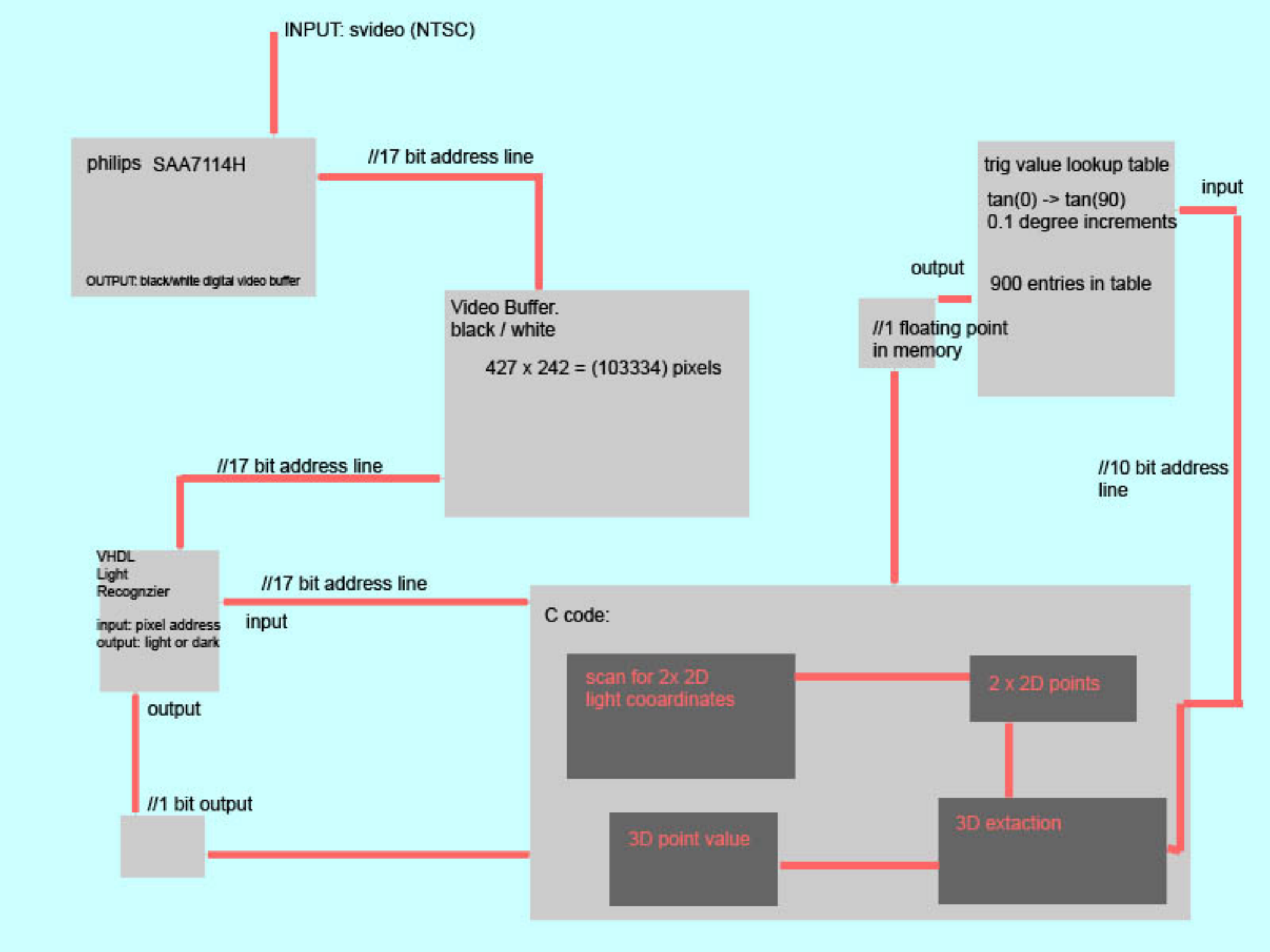
- 3D extraction

```
Input: 2 x 2D Coordinate
Output: 1 x 3D Coordinate

cord1 = Read_Data(loc1);
cord2 = Read_Data(loc2);

depth = k / tan(theta1) + tan(theta2);

return 3D coordinate
```


INPUT: svideo (NTSC)
philips SAA7114H
OUTPUT: black/white digital video buffer
//17 bit address line
Video Buffer.
black / white
427 x 242 = (103334) pixels
trig value lookup table
tan(0) -> tan(90)
0.1 degree increments
900 entries in table
output
//1 floating point
in memory
input
//10 bit address
line
//17 bit address line
VHDL
Light
Recognzier
input: pixel address
output: light or dark
//17 bit address line
input
output
//1 bit output
C code:
scan for 2x 2D
light cooardinates
2 x 2D points
3D point value
3D extaction