Final Project
Depth Perception on FPGA

Jeff Au-Yeung, Rahul Malavalli, Naing Min
Fall 2017

Introduction

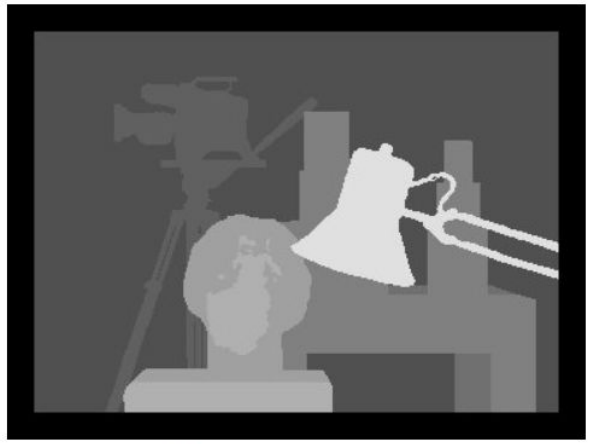
What? – Create a disparity map from 2 images captured by a camera.

Why? – We can calculate depth, just like the human brain does with the images it captures from the eyes.

Purpose? – This can be implemented in robotics and autonomous systems to improve their ability to interact with the environment.

Left Image

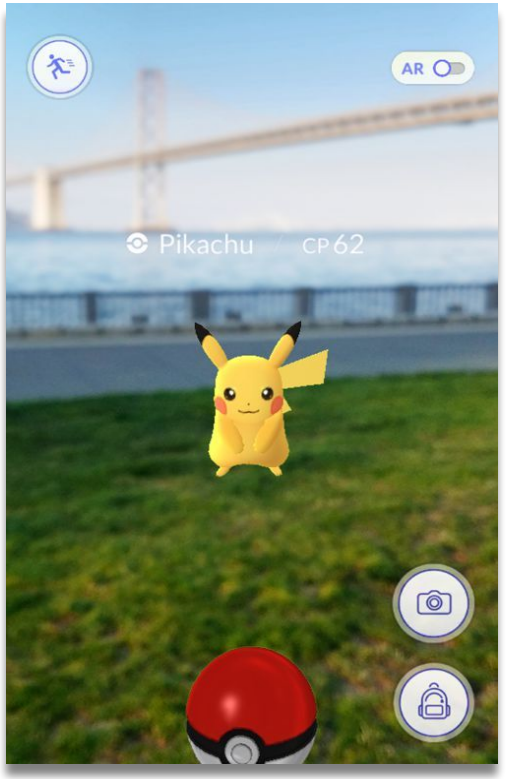
Right Image



Disparity map

Real-World Viability

- Some potential uses:
  - Commercial Drones
    - Minimize collisions
    - Better autonomous control in crowded environments
  - Augmented Reality
    - Calculate accurate distances to provide smoother experiences
    - ex. placing objects accurately in Pokemon Go
- Drawbacks
  - Higher quality cameras and higher resolution images needed for more accurate disparity maps.
  - Requires higher computing power to maintain same compute time.
- Thankfully, devices like smartphones are getting more powerful by the day; already much faster than FPGA.



Industry Standards

- VmodCAM falls far below visual standards of mobile devices
  - Resolution (640x480) fails to capture detail of higher resolution cameras
  - FPS (15) far below refresh rate of smartphone cameras
  - Each red, green, and blue value only has 4 bit depth; results in discolorations and other issues.
  - Poor picture quality from distortion, noisy pixels, inconsistency, and more.
- Computer-Human Interaction
  - Comparison algorithms used take too long (~2 seconds)
  - Camera initialization time is way too long (~5 minutes)
  - These problems may only be due to the FPGA

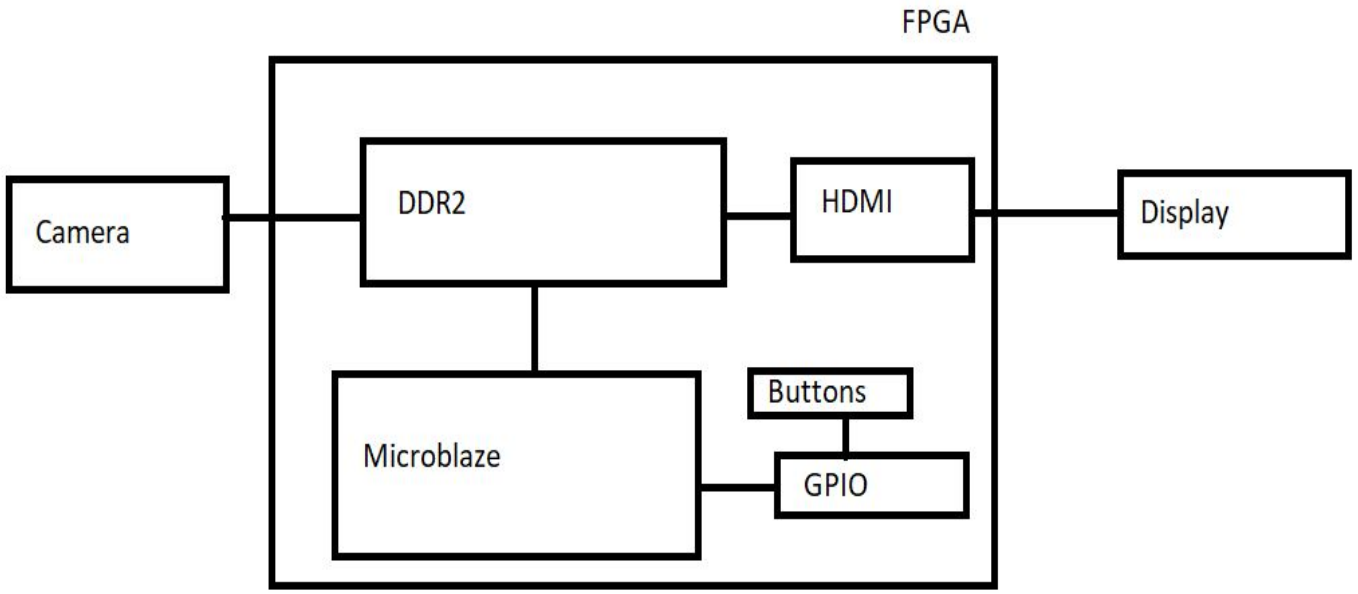
Image of color gradient taken with smartphone (left) is much sharper and accurate than image of same color gradient taken with VmodCAM (right).

Cost of Disparity Mapping

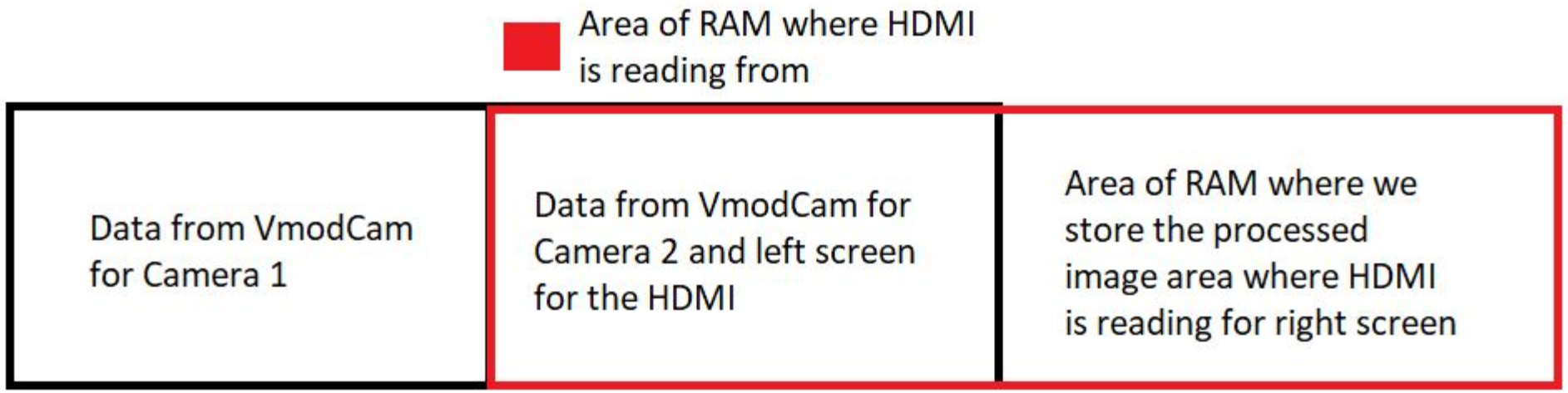
- On small FPGA with low resolution camera, takes ~2 minutes to produce a single low-accuracy disparity map.
  - Camera is cheap
  - FPGA is expensive, could replace with ASIC for better performance and price
- A higher resolution and higher quality camera with less noisy pixels, etc., can provide more accurate disparity maps
  - More expensive
  - Takes more time on the same FPGA because of increase in pixel count
- Alternatives exist
  - Some smartphones have dual cameras
  - Most smartphones have increasingly powerful CPUs and GPUs to compute disparity map
  - Some smartphones can already perform depth perception with one camera and sensing motion

Project Description - Block Design

- Camera writes directly to specific locations in DDR2 RAM
- HDMI reads directly from other locations specified in Direct Video Memory Access (DVMA).
- Push button connected to Microblaze through General Purpose Input Output (GPIO).
- Microblaze connected directly to DDR2 RAM and GPIO
  - Computes disparity map
  - Manages everything



DDR2 RAM Mapping



Example of DDR2 RAM segmenting of camera data input, disparity map calculations, and HDMI data output.

Disparity Map Algorithm

Performs Sum of Absolute (SAD) algorithm to compute disparity map.

Steps to obtain disparity map given a left and right image:

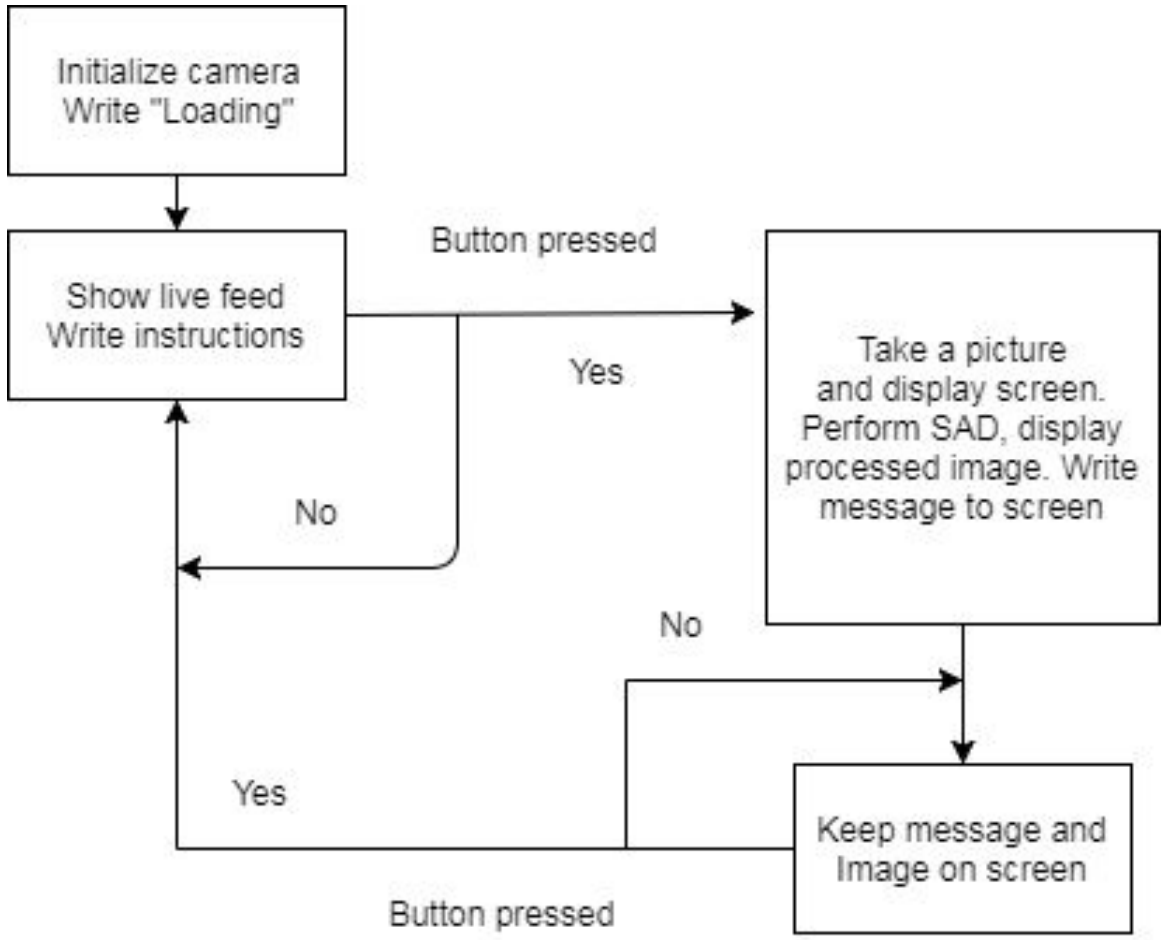
1. Convert both images to grayscale for easier comparison
    a. Assumes ratio of (0.3*R + 0.6*G + 0.1*B), for red (R), green (G), and blue (B) values
2. For each row:
    a. compare every block of pixels in the left image to every block of pixels in the left side of the same row in the right image
        i. Sum the absolute differences (SAD) between each corresponding pixel for each block
    b. Find the block with the minimum SAD and find the corresponding horizontal distance
    c. Write disparity (horizontal distance) to the corresponding pixel in the disparity map for each block
    d. Repeat for every distinct block in the left image

Operational Algorithm

- Camera stores the image as pixel values within the DDR2 RAM
- Take snapshot of images before performing SAD by copying to other location in DDR2 RAM.
- Compute disparity map by performing SAD on the snapshotted images.
  - In Microblaze, read and write appropriate pixels from DDR2 RAM
  - Highest overhead seems to be from this type of I/O operation
- Write disparity map to DDR2 RAM
- Set HDMI to read from disparity map location to display the snapshotted image and the disparity map

Basic Finite State Machine (FSM)

1. **Loading state**
   a. Configure camera for ~5 minutes
   b. Display “loading” message
2. **Initial state**
   a. Show live stream from right camera
   b. Display push button prompt text
   c. Wait for button press
3. **Calculation state**
   a. Take snapshot of current image
   b. Calculate disparity map (SAD)
4. **Viewing state**
   a. Display disparity map
   b. Display push button prompt text
   c. Wait for button press, then return to **Initial State** to repeat



Hardware

- Xilinx Virtex5 FPGA board
  - DDR2 RAM for storing all data
  - Push button as detailed below
  - Runs Microblaze to control entire system
- VmodCAM
  - Used to capture an image and store it into RAM for later use.
- Push button
  - Used to “take picture” for disparity map computation
  - Used to progress through states of program
- HDMI
  - The protocol used to send the data to the monitor via reading off the RAM

Pictured from top to bottom:

- Dell monitor connected via HDMI
- Digilent VmodCAM stereo camera
- Xilinx Virtex5 FPGA board

Software Overview

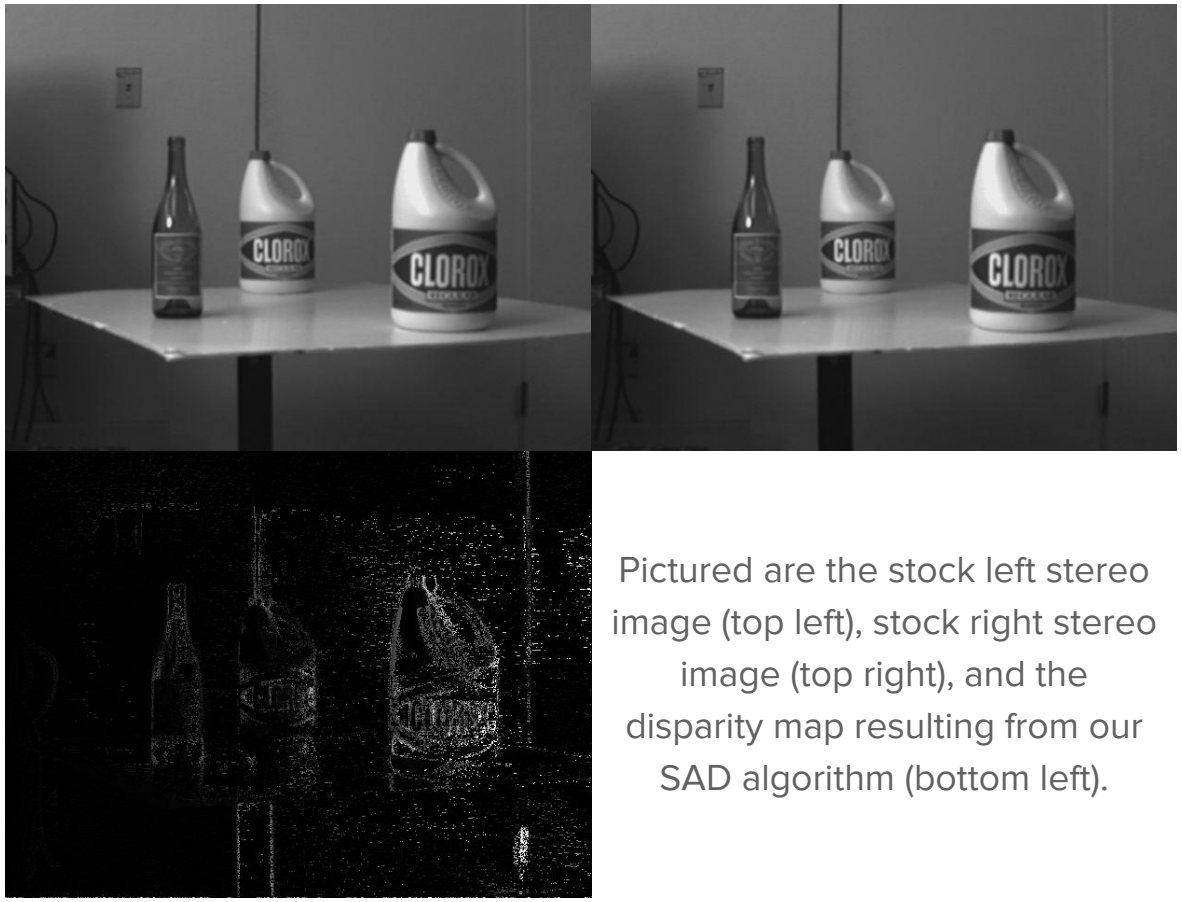
- VmodCAM
  - Requires configuration and initialization driver code, provided in tutorial.
- HDMI
  - Requires control code provided in DVMA module
- Push button
  - Set up in hardware as a GPIO interface, read from Microblaze through GPIO
- Depth perception
  - Disparity map calculated with Sum of Absolute Differences (SAD) algorithm
  - DDR2 RAM reading and writing is heavily optimized
  - Strategic caching of pixel rows and pixel blocks used in algorithm
- Text display on HDMI
  - 16 segment control developed to draw all 26 letters anywhere on screen
- Basic Finite State Machine (FSM)
  - Different state control flow through FSM based on disparity map completion and button presses

SAD Algorithm Baseline

Used our SAD algorithm on stock stereo photos, as seen to the right.

Disparity map produced cleanly and accurately.

Proves that the SAD algorithm works; limited by quality of camera.



Pictured are the stock left stereo image (top left), stock right stereo image (top right), and the disparity map resulting from our SAD algorithm (bottom left).

Conclusions

Need better quality camera.

Need to optimize camera configuration.

Need to further optimize RAM usage.