Worksheets

3D matching engine for FPGA

By Tomas Brandt Trillingsgaard
April 7, 2016

Contents

1 Introduction **5**

2 Algorithms **7**

2.1 Efficient Edge-Preserving Stereo Matching 7

2.2 Fast Cost Volume 9

3 Literature **11**

Chapter 1

Introduction

This document is the worksheets which supports the report written for master thesis on AAU.

It will contain a different chapter for each subject (approximate 1 subject per week). When this document is sent to the supervisor it will only contain the newest chapter/subject. All other chapters will be commented in LaTeX.

Chapter 2

Algorithms

2.1 Efficient Edge-Preserving Stereo Matching

This algorithm uses a combination of Sum of Absolute Differences and Census transform as cost for each pixel and a corresponding disparity. Then these costs are aggregated using successive weighted sum and permeability.
Steps:

1. Calculate cost for each pixel and each disparity
2. Calculate permeability for each pixel
3. Calculate left scan order (right to left direction) for each pixel and disparity
4. Calculate right scan order (left to right direction) and combine it and left scan into horizontal aggregation for each pixel and disparity.
5. Calculate top scan order using horizontal data for each pixel and disparity
6. Calculate bottom scan order using horizontal data and combine it and to scan into total aggregation for each pixel and disparity.
7. Find minimum among disparity values for each pixel

complexity

N = number of pixels and d = max disparity value. 0 = multiplications and division and Θ = additions and subtractions
The cost calculation requires: $(3 \times subtraction + 2 \times additions)$ for each SAD cost.
$((Cen_{size} \times Cen_{size} - 1) \times comparisons)$ for each census transform.
$(Cen_{size} \times XOR + 1 \times sum)$ for each hamming distance calculation.
$2 \times multiplication + 1 \times addition$ for combining costs.

step 1 total: $(3 \times subtraction + 2 \times addition) \times (d + 1) \times N + ((Cen_{size} \times Cen_{size} - 1) \times comparisons) \times (d+2) \times N + (Cen_{size} \times XOR + 1 \times sum) \times (d+1) \times N + (2 \times multiplication + 1 \times addition) \times (d+1) \times N$
$= O(2N \times d) + \Theta(6N \times d)$

Permeability requires: $4 \times (1 \times minimization + 3 \times (exp + subtraction + div)$ for each pixel

Step 2 total: $4 \times (1 \times minimization + 3 \times (exp + subtraction + div) \times N$
$= O(12N) + \Theta(12N) + 12 \times exp \times N$

Left scan order requires: $1 \times add + 1 \times mult$ for each pixel and disparity

Step 3 total: $(1 \times add + 1 \times mult) \times N \times (d + 1)$
$= O(N \times d) + \Theta(N \times d)$

Horizontal + right scan order requires: $2 \times add + 1 \times mult$ for each pixel and disparity

Step 4 total: $(2 \times add + 1 \times mult) \times N \times (d + 1)$
$= O(N \times d) + \Theta(2N \times d)$

Top scan order requires: $1 \times add + 1 \times mult$ for each pixel and disparity

Step 3 total: $(1 \times add + 1 \times mult) \times N \times (d + 1)$
$= O(N \times d) + \Theta(N \times d)$

Vertical + bottom scan order requires: $2 \times add + 1 \times mult$ for each pixel and disparity

Step 4 total: $(2 \times add + 1 \times mult) \times N \times (d + 1)$
$= O(N \times d) + \Theta(2N \times d)$

Total cost: $= O(6N \times d + 12N) + \Theta(12N \times d + 12N) + 12 \times exp \times N$

Memory

input requires: $2 \times N \times d \times 3 \times 8bit + 2 \times N \times d \times 8bit = N \times d \times 64bit$
step 1 requires: $N \times \times d32bitfloat = N \times d \times 32bit$
step 2 requires: $N \times 4 \times 32bitfloat = N \times 128bit$
step 3 requires: $N \times d \times 32bitfloat = N \times d \times 32bit$
step 4 requires: $(N \times d + 1) \times 32bitfloat = (N \times d + 1) \times 32bit$
step 5 requires: $N \times d \times 32bitfloat = N \times d \times 32bit$
step 6 requires: $(N \times d + 1) \times 32bitfloat = (N \times d + 1) \times 32bit$
output requires: $N \times \lceil log_2(d) \rceil bits$

simulation:

I have coded a simulation of this algorithm in python. Runtime: 184 secs for tsukuba test set (384 x 288 pixels, max disparity value of 30)

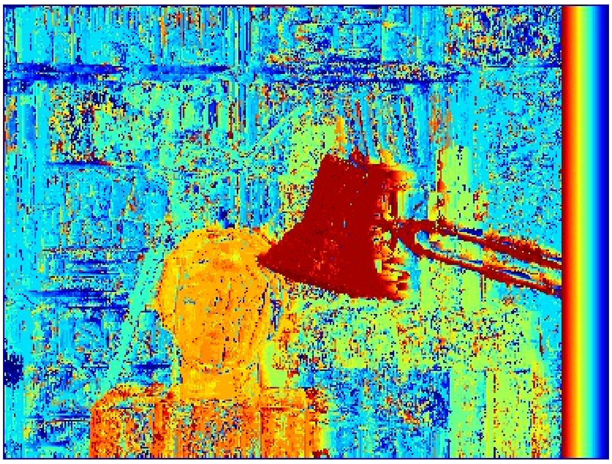

Figure 2.1: Tsukuba result from EEPSM algorithm

2.2 Fast Cost Volume

This algorithm uses a combination of Sum of Absolute Differences and Gradient as cost for each pixel and a corresponding disparity. Then these costs are weighted by a guided image filter.

complexity

guided filter is $O(N)$ according to articles.

memory

mem for
input requires: $2 \times N \times d \times 3 \times 8bit + 2 \times N \times d \times 8bit = N \times d \times 64bit$

simulation

My own python implementation: runtime: 62 for tsukuba test set (384x288 pixels, max disparity value of 30).
Matlab implementation by authors: runtime: 51 seconds for tsukuba test set (384x288 pixels, max disparity value of 15) and 93 seconds for tsukuba test set (384x288 pixels, max disparity value of 30)

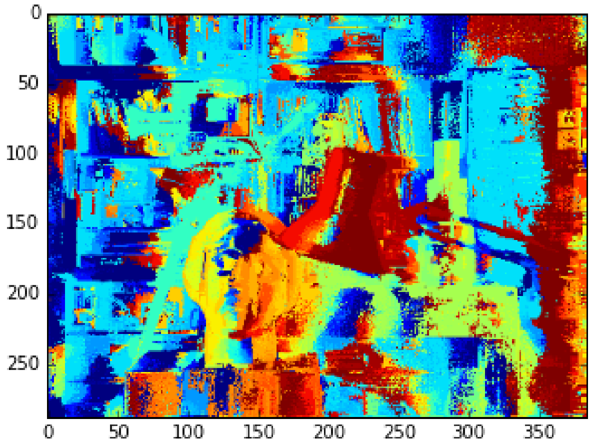

Figure 2.2: pyt test 30

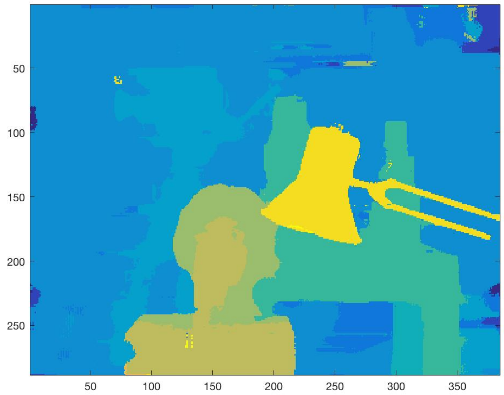

Figure 2.3: mat test 15 disp

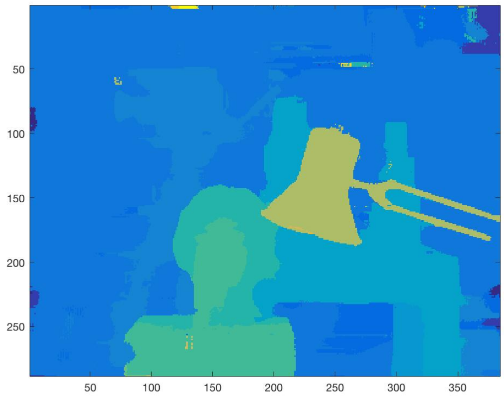

Figure 2.4: mat test 30 disp

Chapter 3

Literature

This chapter contains a list of literature I have read or expect to read during the master thesis:

- **A Fast Binocular Vision Stereo Matching Algorithm**, 2012, *Hui Zhang, LingTao Zhang, Ming Zhao ,and Jian Liu*, article
- **Stereo Vision in Structured Environments by Consistent Semi-Global Matching**, 2006, *Heiko Hirschmüller*, article
- **Structured-light stereo: Comparative analysis and integration of structured-light and active stereo for measuring dynamic shape**, 2013, *Wonkwi Jang, Changsoo Je, Yongduek Seo, Sang Wook Lee*, article
- **Active Animate Stereo Vision**, 1993, *C.W. Urquhart, J.P. Siebert, J.P. McDonald and R.J. Fryer*, article