Stereo vision and Depth map Generation

July 15, 2024

Dongho Kim

opticho

Contents

Introduction

- **Need for making it fast**

1. Efficiency in Parallel Processing

Depth map generation involves complex calculations for numerous pixels.
CUDA leverages the parallel processing capabilities of GPUs, allowing these calculations to be performed simultaneously.

2. High Performance Computing

Algorithms for depth map generation, especially in real-time applications, demand high performance. Using CUDA enables high-performance computing, which is particularly beneficial when generating depth maps for high-resolution RGB images or videos.

Algorithm

1. Compute Cost

$$\mathrm{SAD}(x,y,d) = \sum_{c=0}^{C-1} |I_L(x,y,c) - I_R(x-d,y,c)|$$

If $x - d < 0$: SAD + 255

When it excludes the image range

I_L : Left Image I_R : Right Image c : Index of Channels

2. Aggregate Costs

$$A_H(x,y,d) = C(x,y,d) + \min_{pd}\left[A_H(x-1,y,pd) + P(d,pd)\right]$$

$$A_V(x,y,d) = C(x,y,d) + \min_{pd}\left[A_V(x,y-1,pd) + P(d,pd)\right]$$

$A_H\,(x,y,d), A_V\,(x,y,d)$

Aggregated Costs
Horizontally or Vertically

$\mathrm{C}(x,y,d)$: Initial Costs $\mathrm{P}(d,pd)$: penalty between disparity d and pd

3. Compute Disparity Map

$$D(x,y) = \arg\min_d A(x,y,d)$$

$\mathrm{D}(x,y)$: Optimal disparity value of pixel (x,y)

$\mathrm{A}(x,y,d)$: Aggregated costs

4. Apply Color Map

$$R = \max(0, 1 - |4 \cdot \mathrm{ratio} - 3|) \cdot 255$$
$$G = \max(0, 1 - |4 \cdot \mathrm{ratio} - 2|) \cdot 255$$
$$B = \max(0, 1 - |4 \cdot \mathrm{ratio} - 1|) \cdot 255$$

$ratio = \frac{D(x,y)}{D_{max}}$ Regulated ratio of disparity value

Implementation

- **Cuda Architecture**

Compute_cost_kernel
- Blocks ((imge_width + 15) / 15, (imge_height + 15) 15)
- Threads(16,16)

Aggregate_cost_horizontal_kernel
- Blocks(1, img_height + 15 / 15)
- Threads(16, 16)

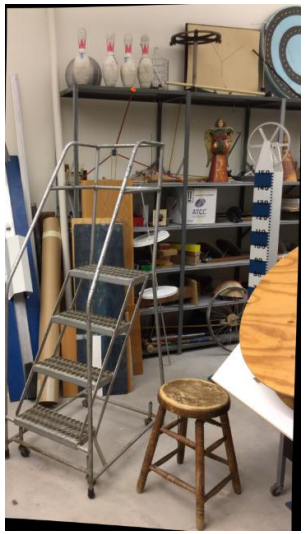
Aggregate_cost_vertical_kernel
- Blocks(img_width + 15 / 15, 1)
- Threads(16, 16)

Compute_disparity_map_kernel
- Blocks ((imge_width + 15) / 15, (imge_height + 15) 15)
- Threads(16,16)

- I couldn’t use shared memory or constant memory etc.
- If I success other way of implementation, I would submit the code together.

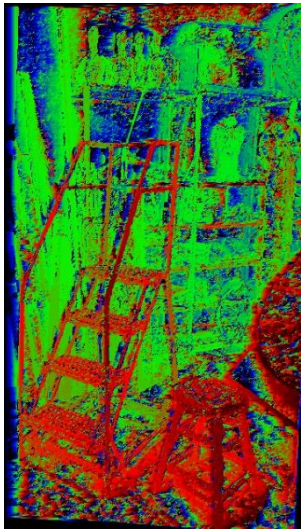
Results

Left Image

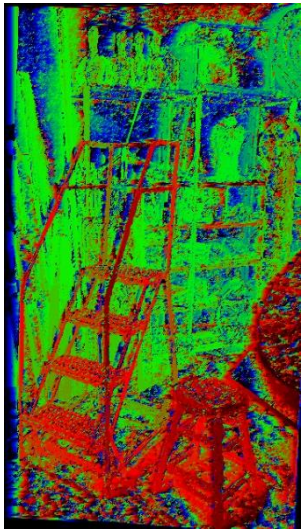


Right Image

After Depth Map generation with C



After Depth Map generation with Cuda



- Each Image is a png file and size is 360 x 640

Results

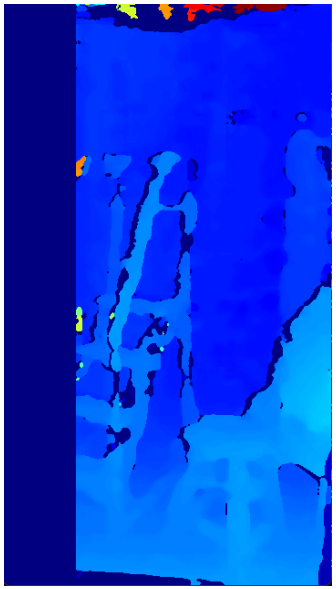
```
stereovision Depth Map C Performance(ms) = 3352.919434
```

About 6.5 Times Faster !

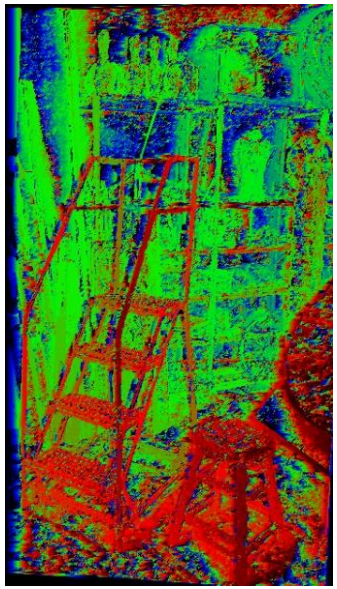
```
StereoVision Depth Map(Cuda C) - Time for execution = 511.8 ms
```

Conclusion and future work

Using SGBM
from OpenCV library



Cuda Implementation



Achievements

- Enhanced quality comparing with OpenCV Image.
- High performance rather than C++ programmed image.

Optimizations

- Using other memory types such as Shared Memory, Constant Memory, Pinned Memory, etc.
- Investigate other algorithms or open-source algorithms for better quality of depth map.

Using SGBM
from OpenCV library

```
StereoVision Depth Map(OpenCV) Time for Excution: 857.484 ms
```

Cuda Implementation

```
StereoVision Depth Map(Cuda C) - Time for execution = 511.8 ms
```