STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

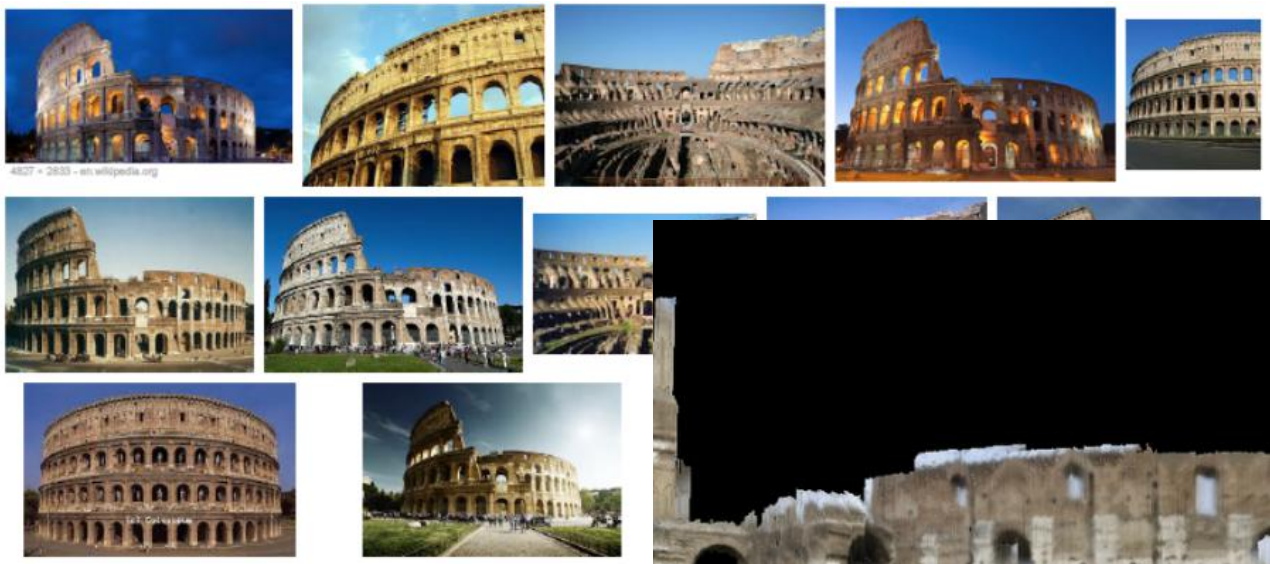
Depth Map From Multi-View Using Plane Sweep Algorithm

Course Project of CPE810 GPU and Multicore Programming

Name: Haixu Song
CWID: 10446032

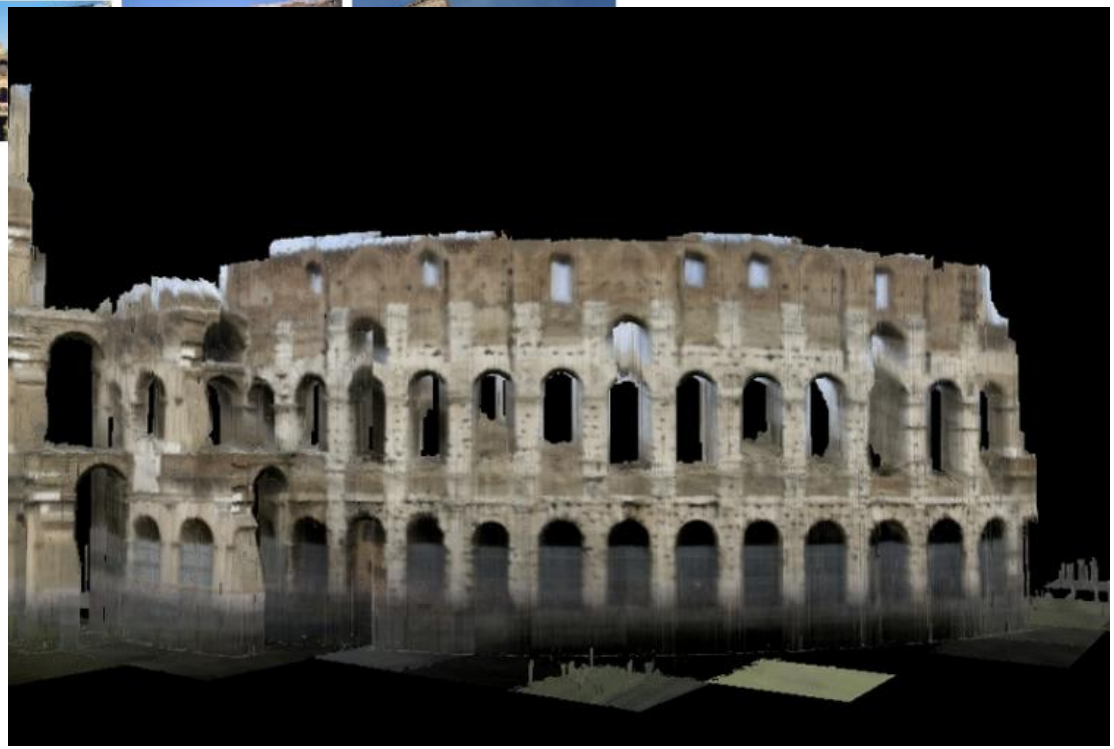
OBJECTIVE

Modeling Depth Map or Point Cloud Using Multi-veiw Pictures.



Colosseum

3D Model



BACKGROUND

Nowadays Popular Techniques No.1

LiDAR

LiDAR

LiDAR + Camera

Fast, Easy, Accurate

BACKGROUND

LiDAR

Nowadays Popular Techniques No.1

$ 70K

15 FPS

120m 0.3~0.5m

BACKGROUND

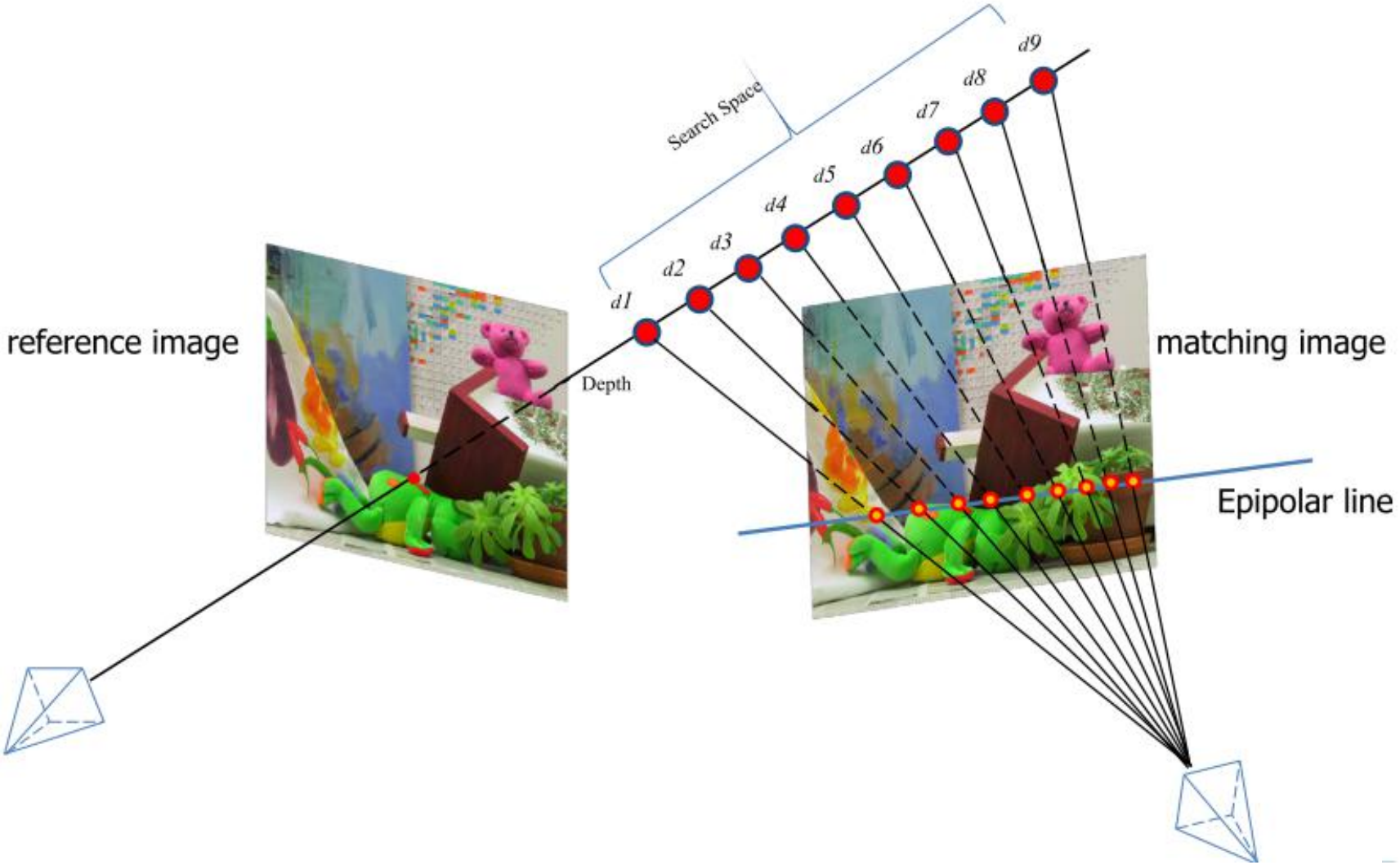
Nowadays Popular Techniques No.2

Camera Stereo

BACKGROUND

Nowadays Popular Techniques No.2

Camera Stereo



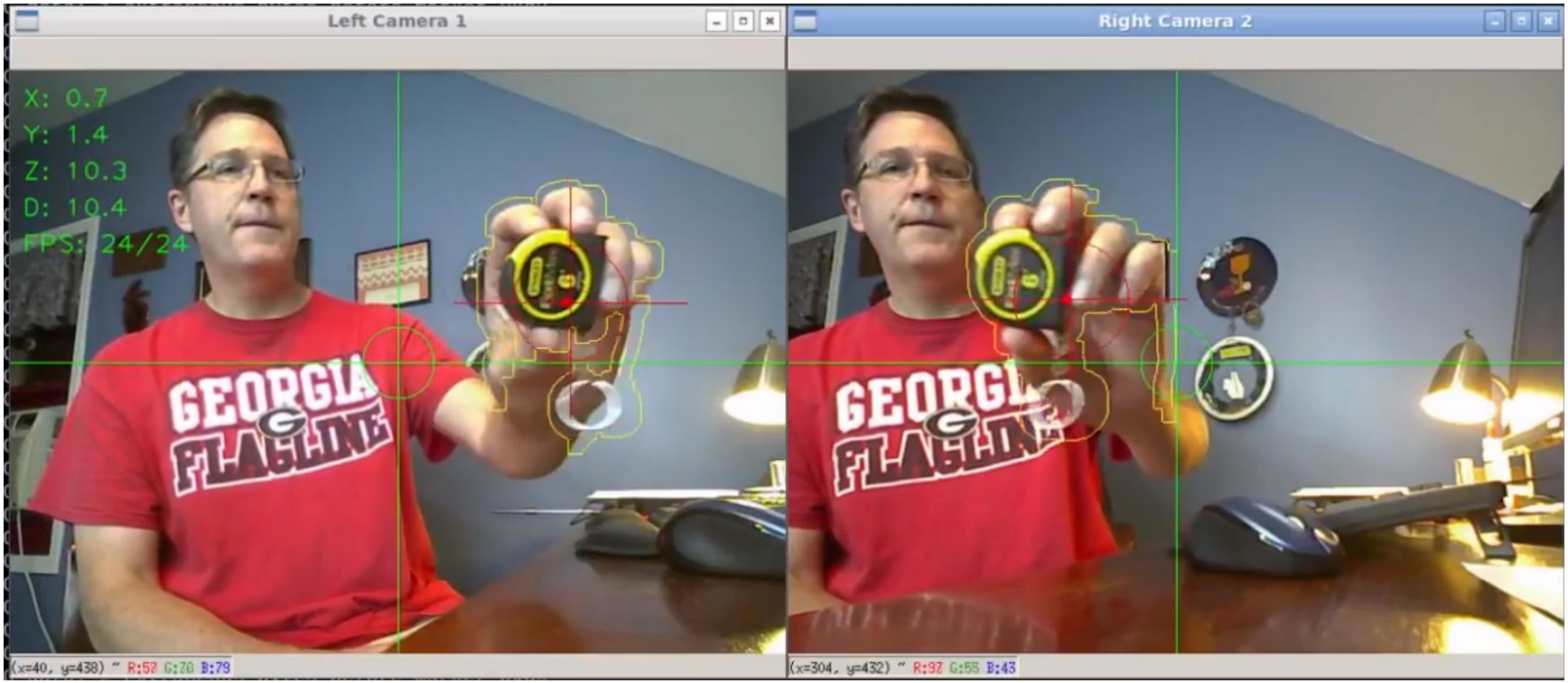
BACKGROUND

Nowadays Popular Techniques No.2

Camera Stereo

+

AI



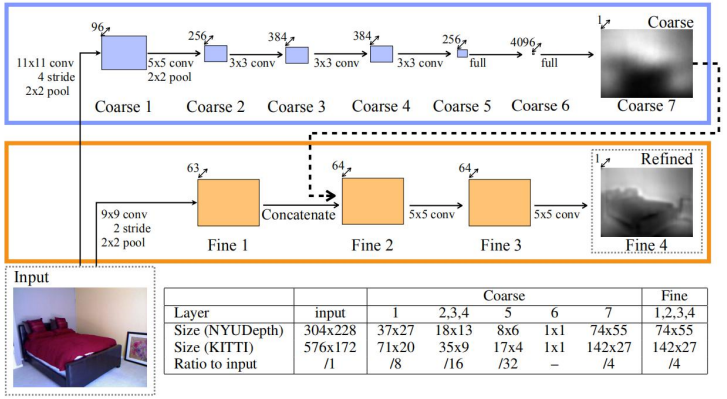
BACKGROUND

Nowadays Popular Techniques No.3

Single Picture

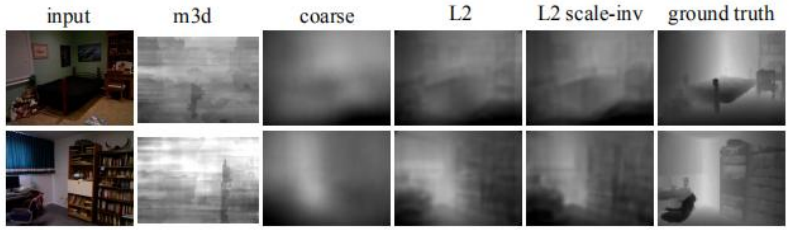
Vedio Based

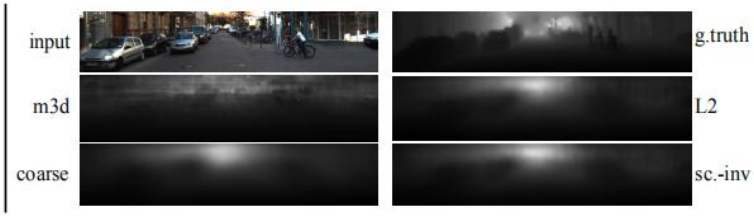
Focus Based



		Coarse					Fine
Layer	input	1	2,3,4	5	6	7	1,2,3,4
Size (NYUDepth)	304x228	37x27	18x13	8x6	1x1	74x55	74x55
Size (KITTI)	576x172	71x20	35x9	17x4	1x1	142x27	142x27
Ratio to input	/1	/8	/16	/32	–	/4	/4

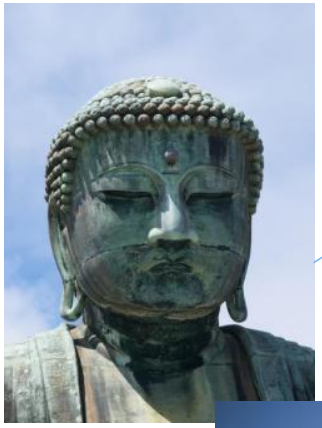
AI Based





Plane Sweep Algorithm

What it used for?



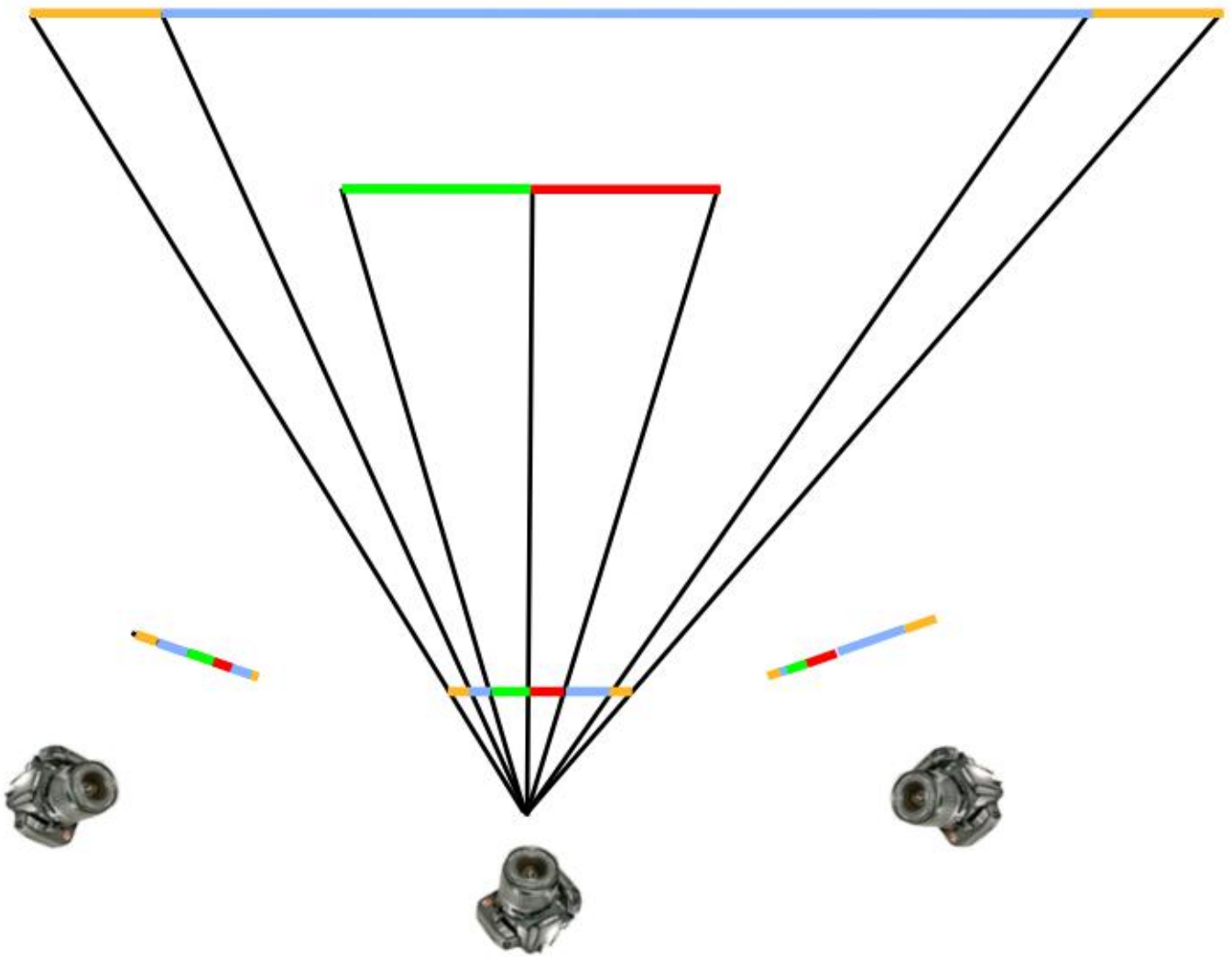
Plane Sweep Algorithm

How it works?

Objects

Images

Angles

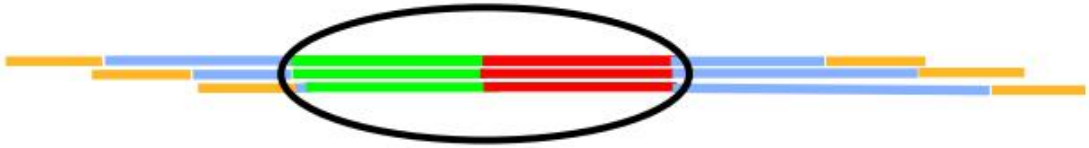


Plane Sweep Algorithm

How it works?

Objects

Plane



Images

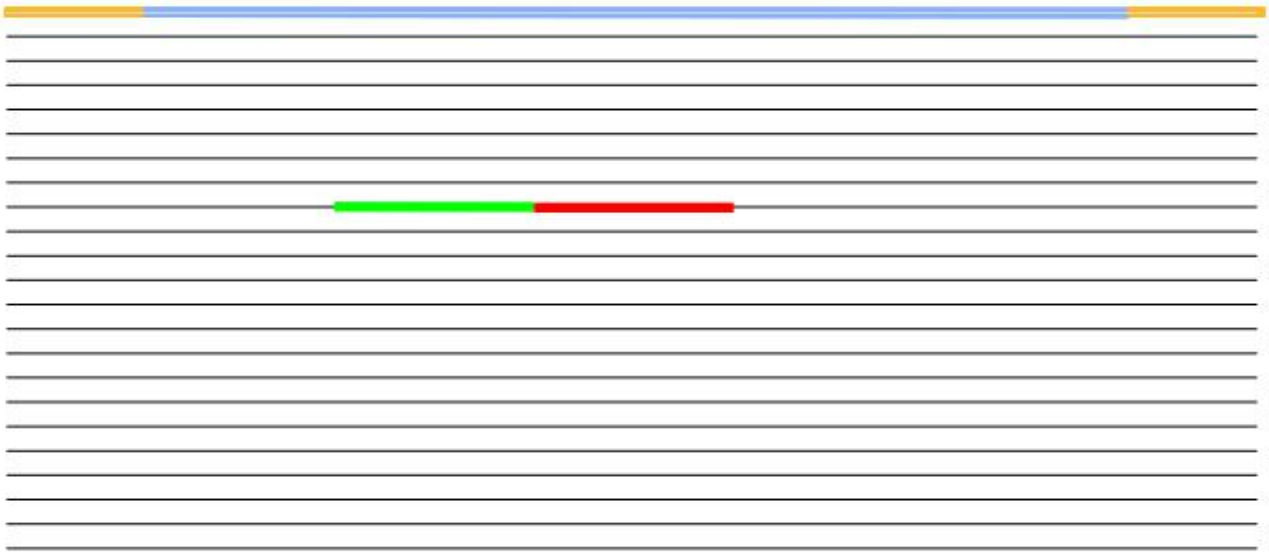
Angles

Plane Sweep Algorithm

How it works?

Objects

Hypothese Space / Planes



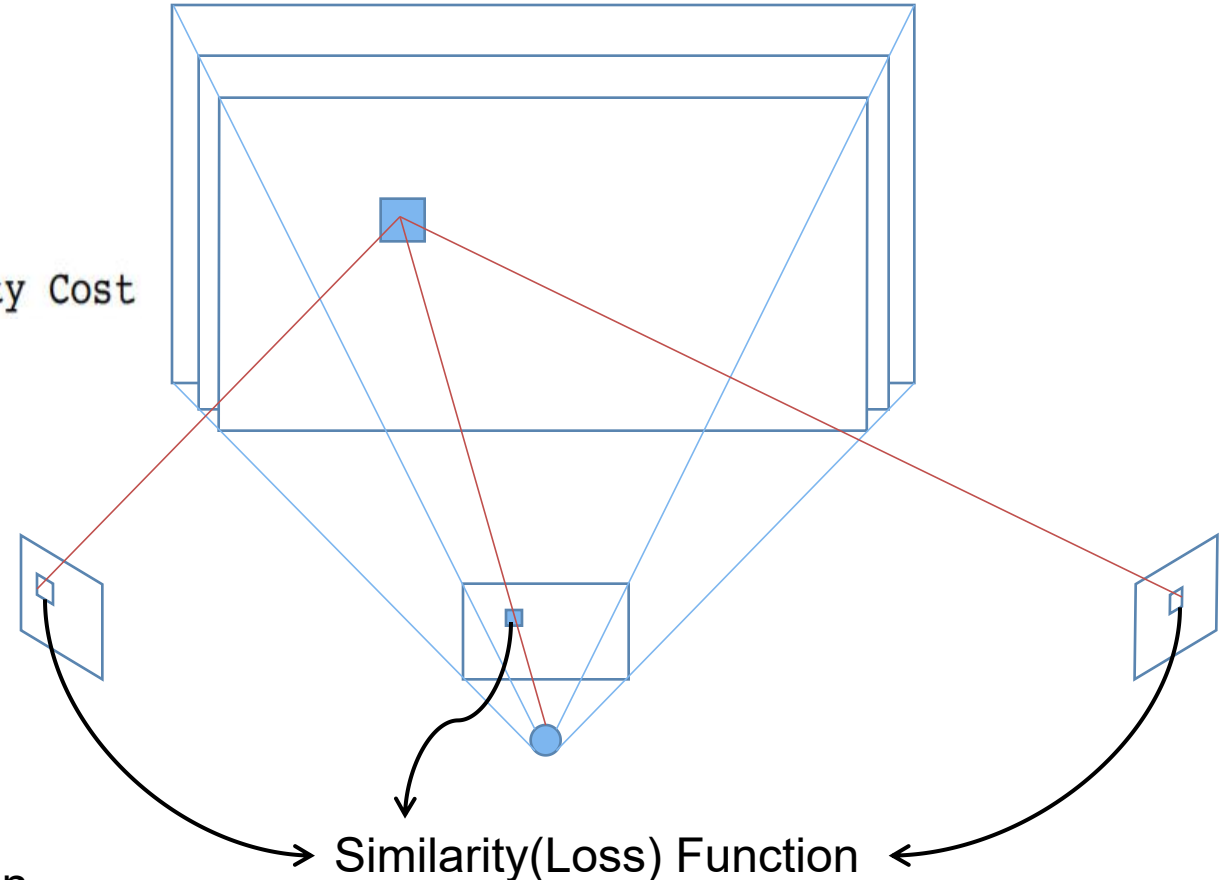
Images

Angles

Plane Sweep Algorithm

Coding (Sequential)

```
Initialize Similarity Cost
For Each Pixel
   For Each Depth
      For Each Source Image
         Accumulate Similarity Cost
```



SSD:
Sum of Squared Difference

SAD:
Sum of Absolute Difference

NCC:
Normalized Cross Correlation

Plane Sweep Algorithm

Equations: How to do projection? Some Concepts

Internal Camera Parameters (ip):

Camera Coordinate -> Pixel Postion

External Camera Parameters (ep):

World Coordinate -> Camera Coordinate

Camera Position (cp):

Camera's World Coordinate

Plane Sweep Algorithm

Equations: How to do projection? Some Concepts

World Coordinate (wld) -> Pixel Position (pix):

$$pix = ip * ep * (wld - cp)$$

Pixel Position (pix) -> World Coordinate (wld):

$$wld = ep^{-1} * ip^{-1} * pix + cp$$

Plane Sweep Algorithm

Python Code for Testing

from psalgo import *

1. Load Input Source

```
test06Image = InputImage("./test/test06.png",
                         ep=ep_06, cp=cmpos_06,
                         bound=bound_06, ip=ip_test)
```

2. Load Target Params

```
targetImage = TargetImage(
    ep=ep_05, cp=cmpos_05, bound=bound_05, ip=ip_test,
    path="./test/test05.png")
```

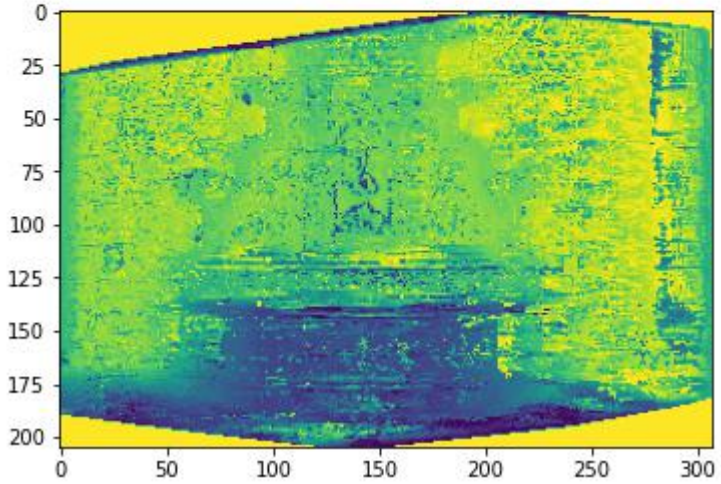
3. Load Input Source to Target

```
source = (test05Image, test06Image, test04Image,
          test03Image, test07Image)
targetImage.load(source)
```

4. Fit

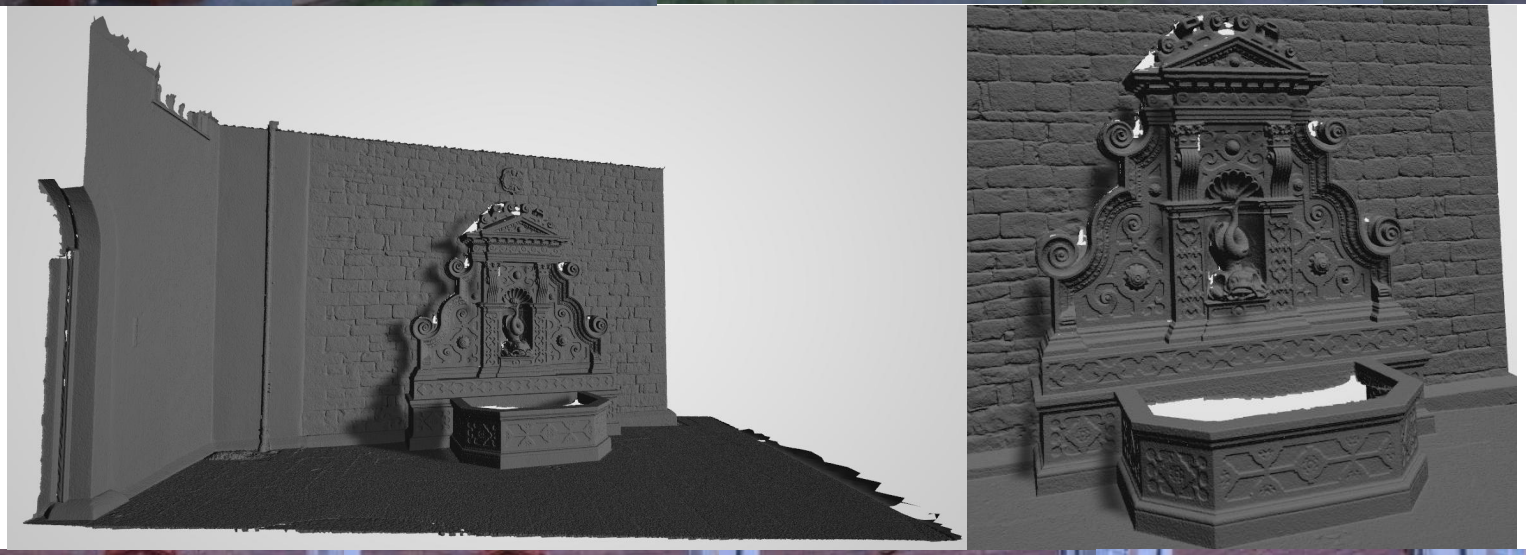
```
targetImage.fit(similarityFunction = "SAD",
                depthRange=(5, 9),
                planes=60,
                windowSize=1)
```

```
Fitting:        100%    60/60
row:204
```

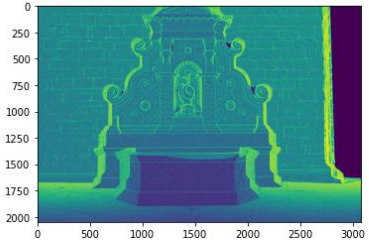


Plane Sweep Algorithm

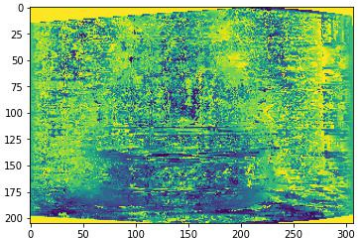
Data: fountain-P11 Resolution 3072 x 2048



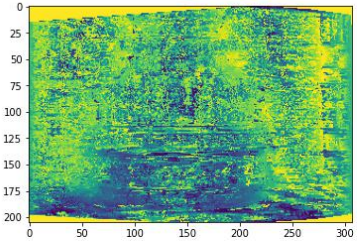
Resolution 3072 x 2048 -> 307 x 205 (10 minutes)



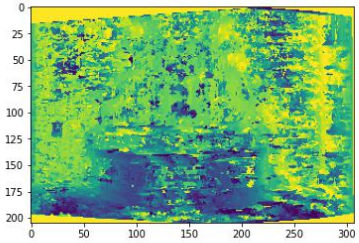
Ground Truth



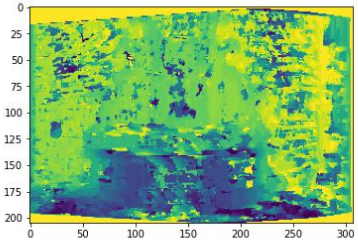
SSD 100planes, 1window



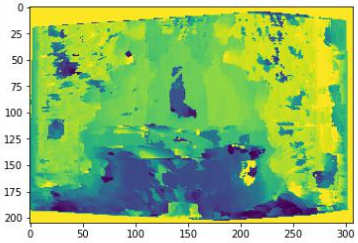
SSD 60planes, 1window



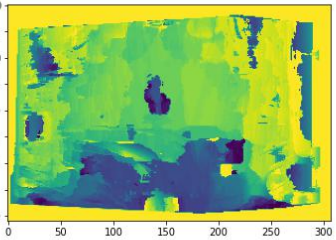
SSD 60planes, 3window



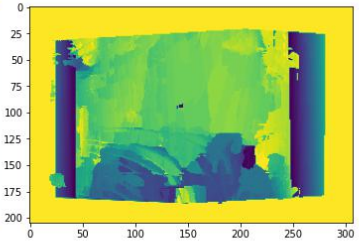
SSD 60planes, 5window



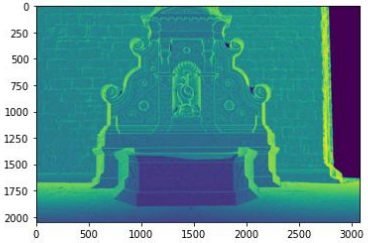
SSD 60planes, 11window



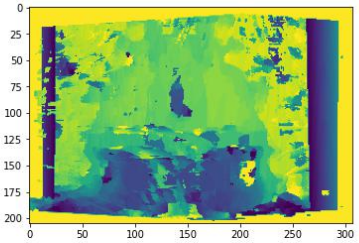
SSD 60planes, 21window



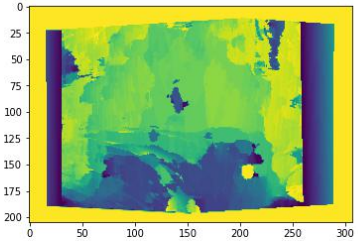
SSD 60planes, 41window



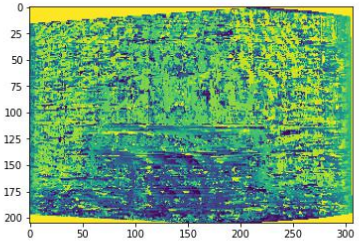
Ground Truth



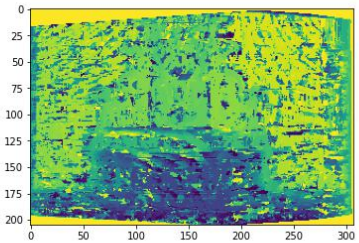
SAD 60planes, 11window



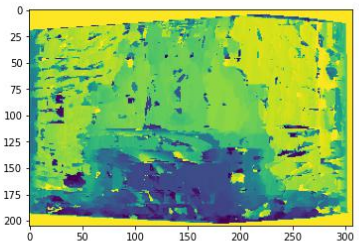
SAD 60planes, 21window



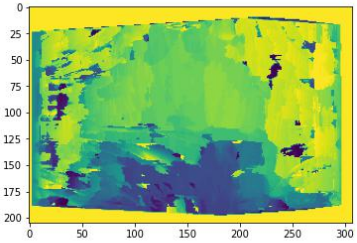
NCC 60planes, 3 window



NCC 60planes, 5window



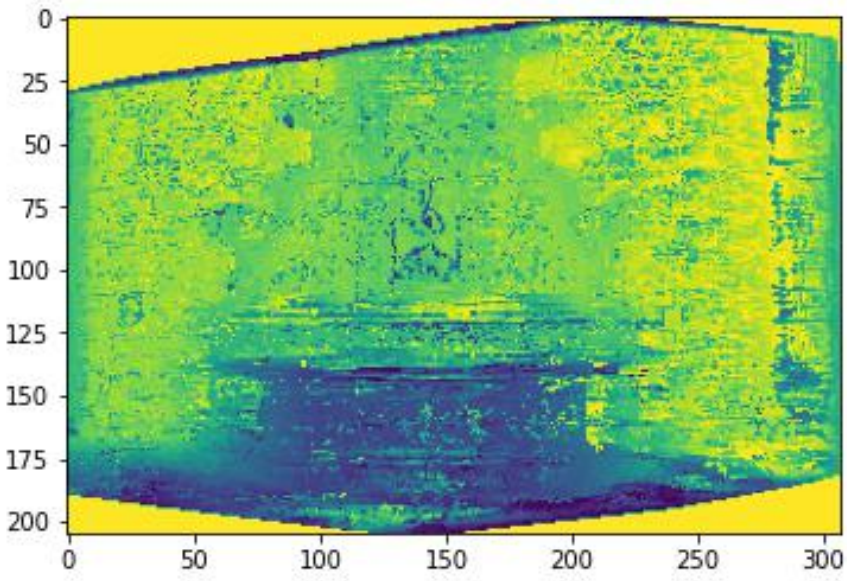
NCC 60planes, 11window



NCC 60planes, 21window

Plane Sweep Algorithm

Data: fountain-P11 Resolution 307 x 205



60 Planes ranged from 5 to 9

SAD

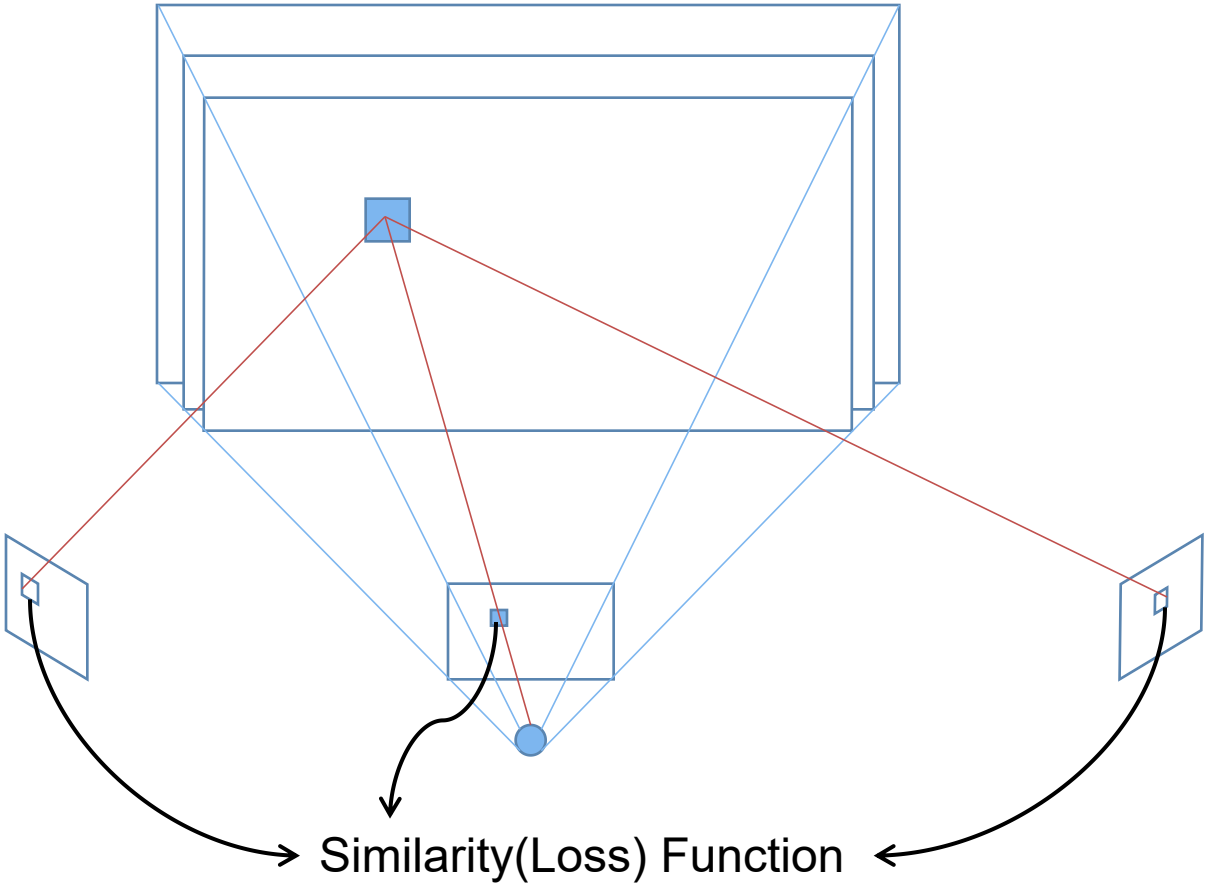
window-size: 1

Plane Sweep Algorithm

Code: GPU using CUDA Resolution 3072 x 2048

__constant__ parameters

each thread:
for each depth:
    find wld
    find pix on each source
    calculate loss
    update depth



Plane Sweep Algorithm

Code: GPU using CUDA Resolution 3072 x 2048

Optmizing equations for each thread:

$pix = ip * ep * (wld - cp)$ -> $P = ip * [\, ep \mid -ep*cp]$

$pix = Cartesian(P * homo(wld))$

$wld = ep^{-1} * ip^{-1} * pix + cp$ -> $Pi = [ep^{-1} * ip^{-1} \mid cp]$

$wld = Pi * homo(homo(pix) * depth)$

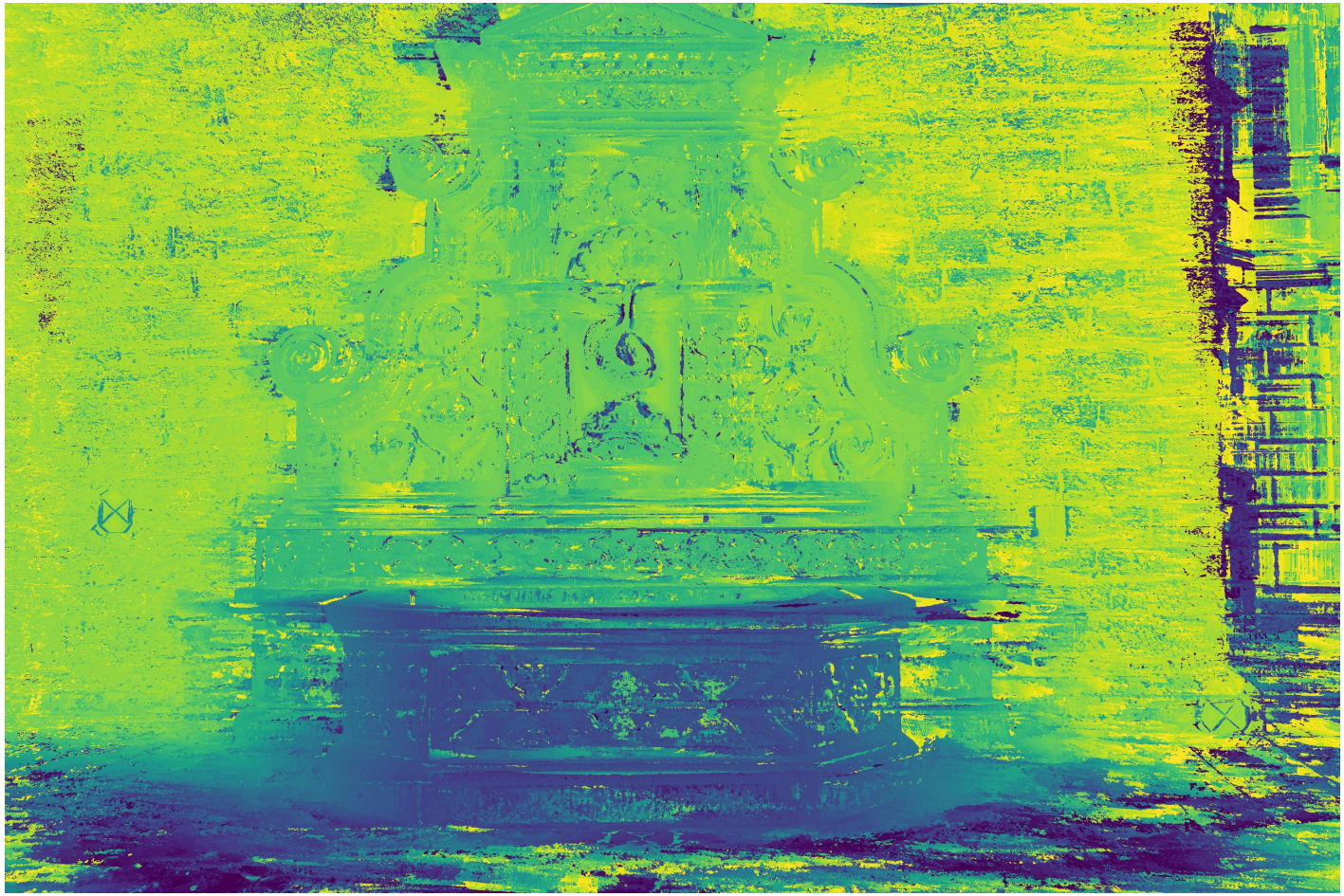
P and Pi can be pre-calculated in sequential code.

Then put them in constant memory.

Plane Sweep Algorithm

Code: GPU using CUDA Resolution 3072 x 2048

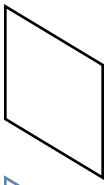
Kernel runtime: about 50 ms. (which means about 20FPS on 4K resolution)



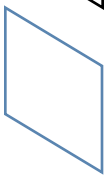
Plane Sweep Algorithm

Future Work: Further Optimizing

- Optimized Similarity Function
- Structured and Scalable C Code Using Libs.
- Use shared memory for window projecting.



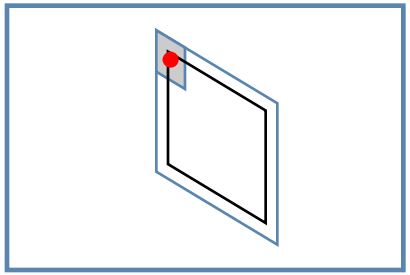
Projected thread block

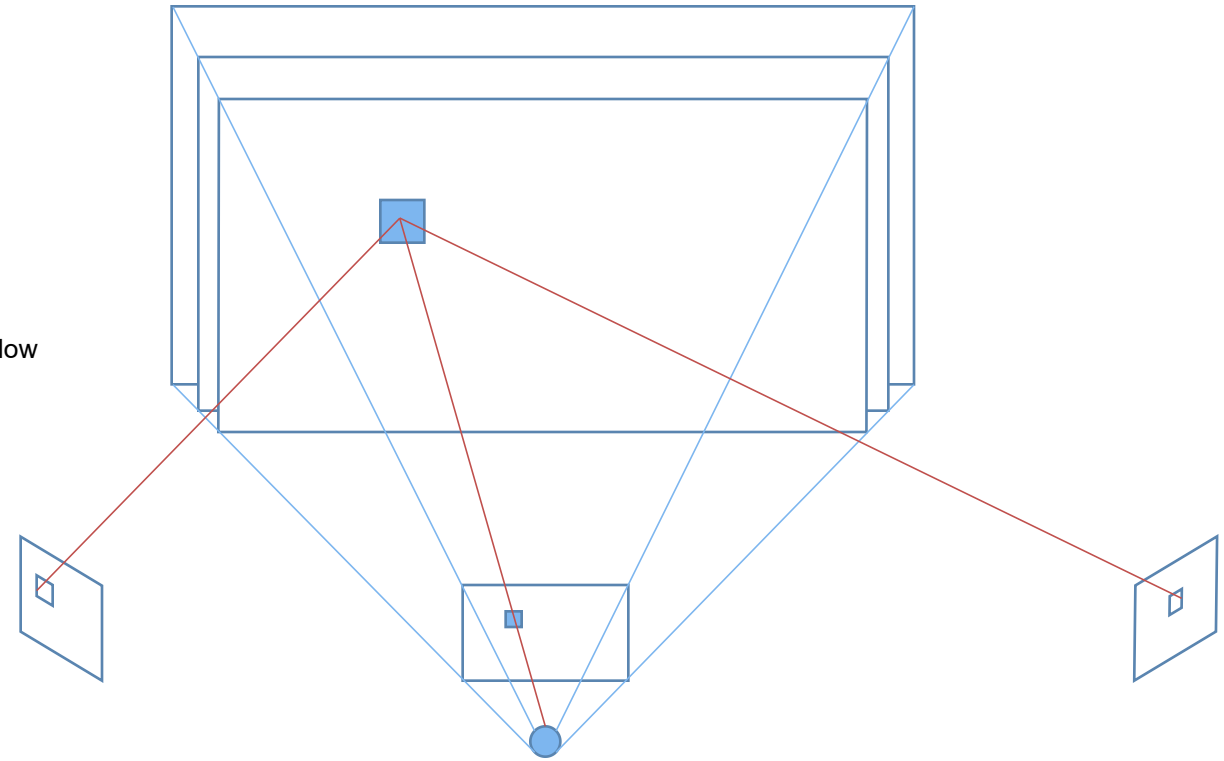


Shared in blocks

Projected pixel

Projected window





THANKS FOR LISTENING