Computer Vision: Lecture 12

Carl Olsson

2020-02-26

Todays Lecture

Stereo

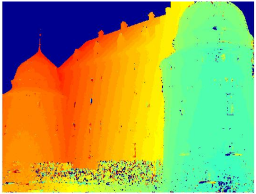
- Stereo cameras
- Disparity and depth
- Dense matching
- Normalized cross correlation
- The plane sweep approach
- Regularization
- Silhouettes, Visual hull.

Dense Stereo

Goal

- Estimate the depth in every pixel. Dense depth map.
- Requires every point in the image to be matched!





Dense Stereo

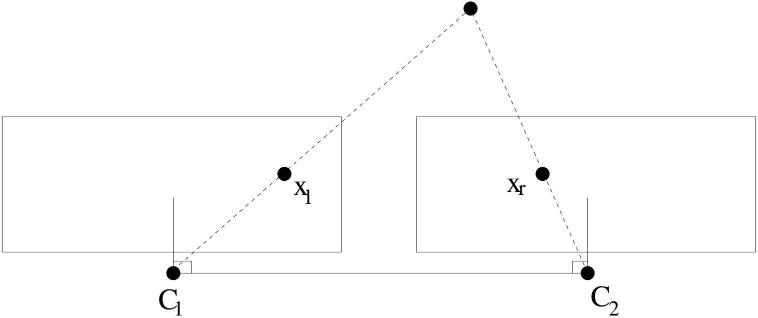
Rectified images

Assumptions:

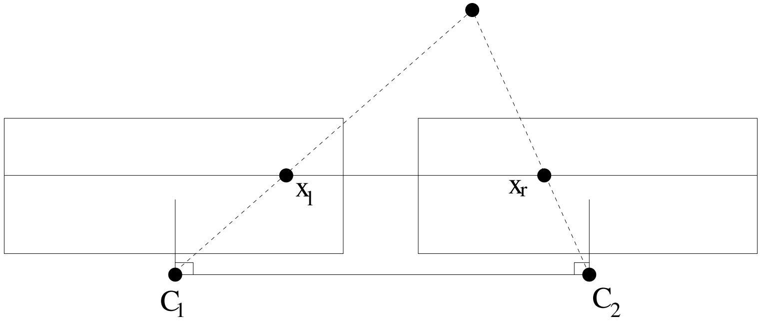
- The image-planes are parallel, and the second camera center is translated in the x-direction of the first. (Can always be achieved by transforming the images.)
- The cameras have the same inner parameters.

$$P_1 = K[I \; 0] \text{ and } P_2 = K\left[I \; \begin{pmatrix} b \\ 0 \\ 0 \end{pmatrix}\right]$$

Rectified Cameras

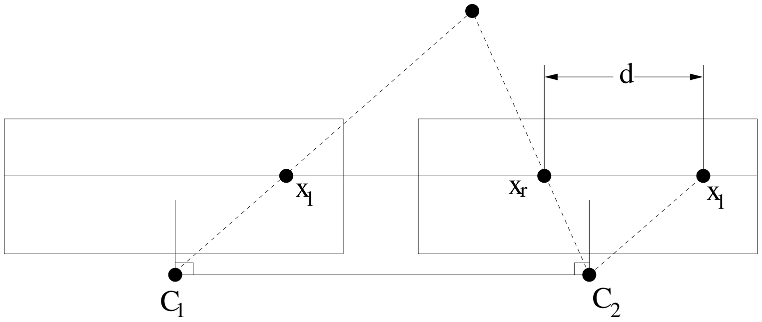


Rectified Cameras



Epipolar lines are parallel to the x-axis.

Rectified Cameras

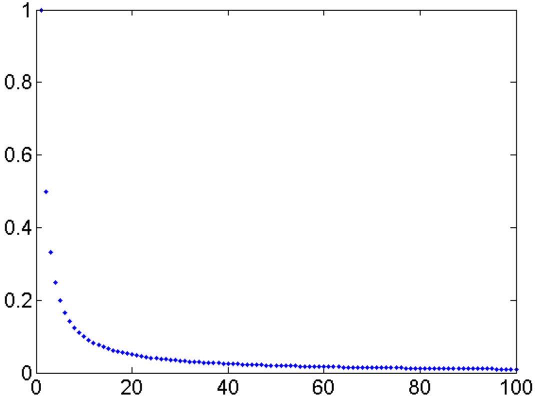


Difference between the x-coordinates of x_l and x_r is called the disparity.

Disparity and Depth

See lecture notes.

Disparity vs. Depth



Disparity (in pixels) vs. depth when $f_x b = 1$. Higher resolution when the depth is small/the disparity is large.

Normalized Cross Correlation

Why not SIFT?

- Need measurements everywhere.
- Cameras known $\Rightarrow$ don't need scale, rotation invariance.

NCC

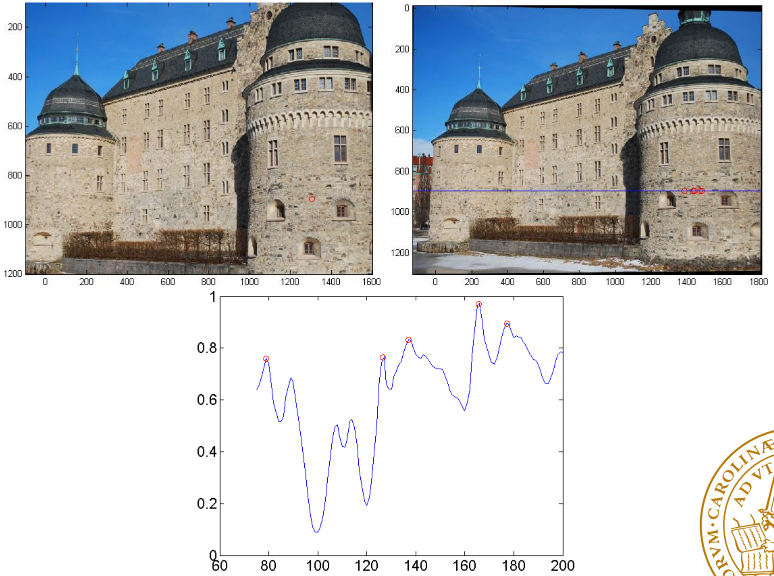
If I_1 and I_2 are gray levels of two patches,

$$NCC(I_1, I_2) = \frac{1}{n-1} \sum_{i=1}^{n} \frac{(I_1(x_i) - \bar{I}_1)(I_2(x_i) - \bar{I}_2)}{\sigma(I_1)\sigma(I_2)},$$

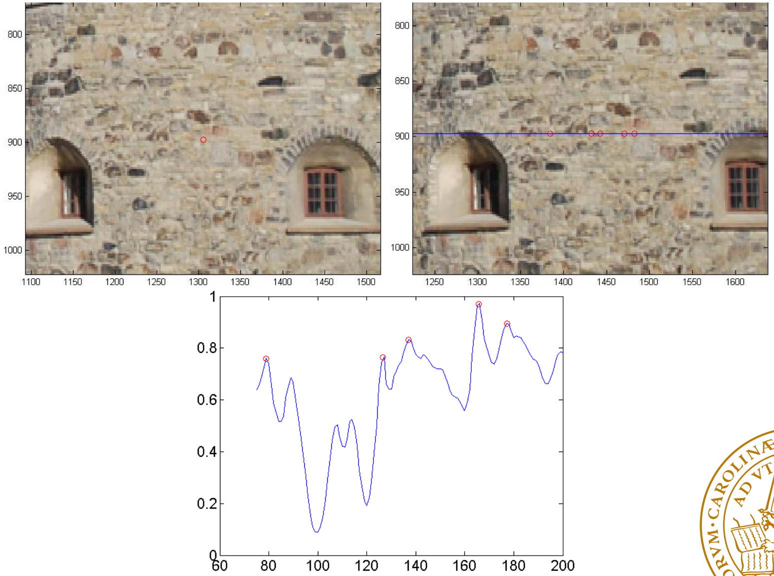
$\bar{I}_1$,$\bar{I}_2$ - mean values of each patch.
$\sigma(I_1)$,$\sigma(I_2)$ - standard deviations of each patch.

- Invariant to translation and rescaling of the grayvalues. (Good for handeling different lighting conditions.)

Normalized Cross Correlation



Normalized Cross Correlation

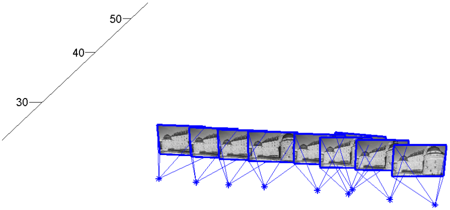


Normalized Cross Correlation

Demonstration.

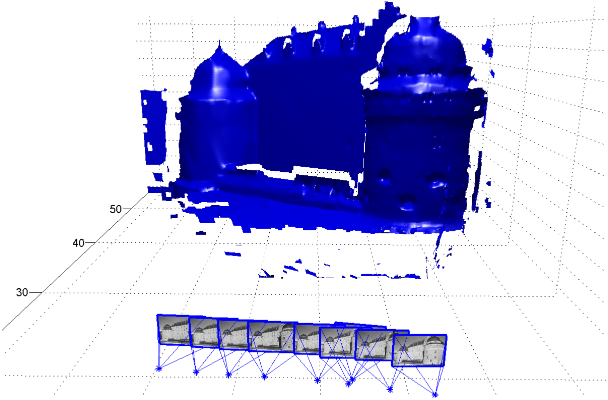
Depth Map Estimation

Given images and Cameras, how do we compute a dense surface estimate? (Need to find matches for all the pixels in the image.)



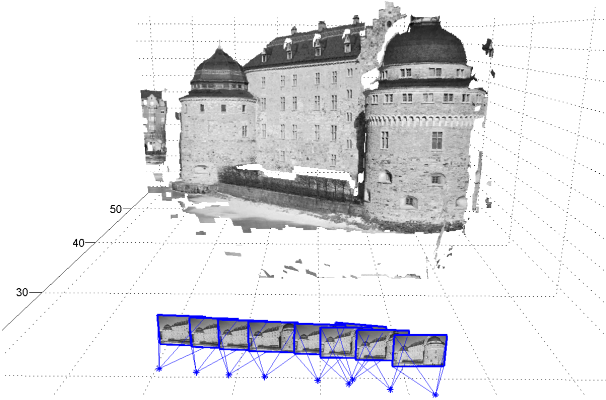
Depth Map Estimation

Given images and Cameras, how do we compute a dense surface estimate? (Need to find matches for all the pixels in the image.)

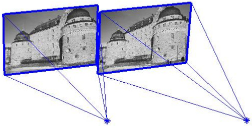


Depth Map Estimation

Given images and Cameras, how do we compute a dense surface estimate?
(Need to find matches for all the pixels in the image.)

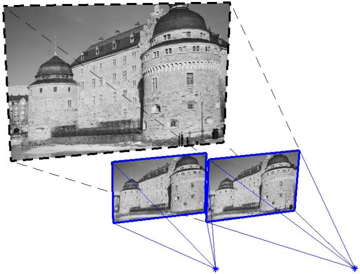


Plane Sweep Algorithm



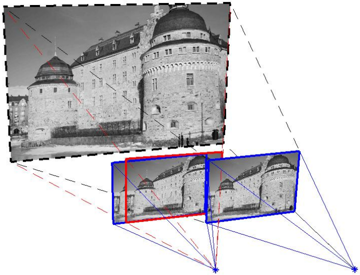
Given 2 cameras. How do we find the depths of all the pixels?

Plane Sweep Algorithm



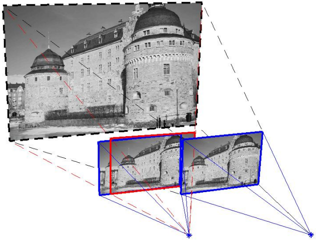
Try giving all the pixels the same depth.

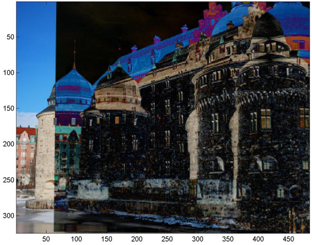
Plane Sweep Algorithm

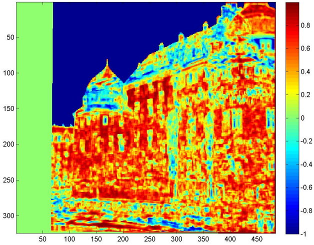


Project into the second image and compare.

Plane Sweep Algorithm

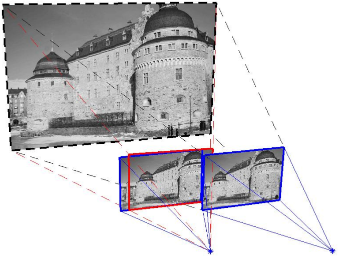


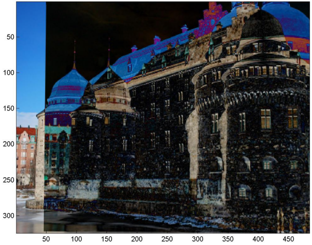


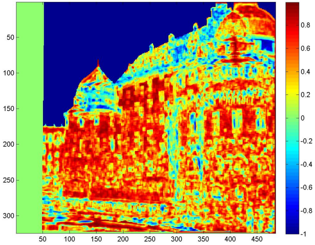


Difference between the original image and the projection.

Plane Sweep Algorithm

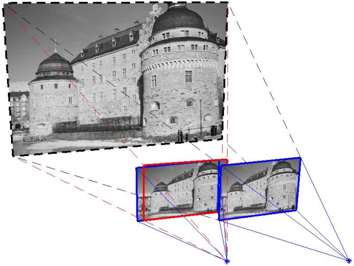


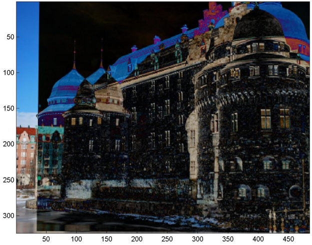


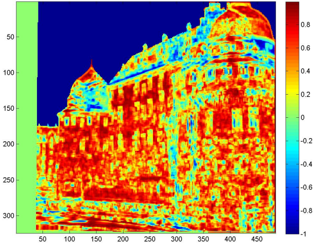


Try several different depths.

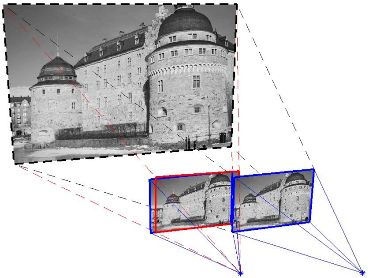
Plane Sweep Algorithm

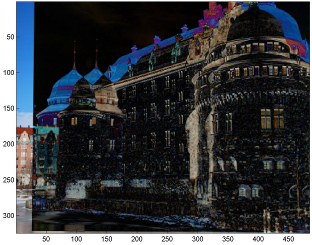


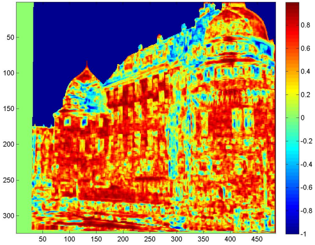




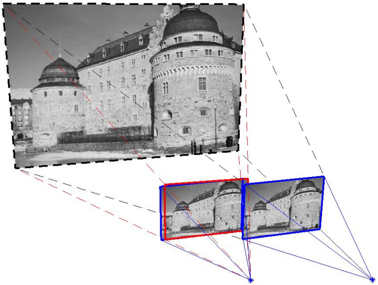
Plane Sweep Algorithm

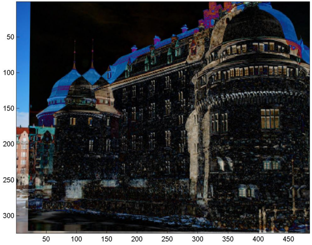


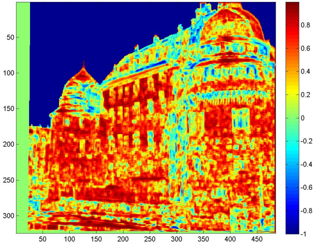




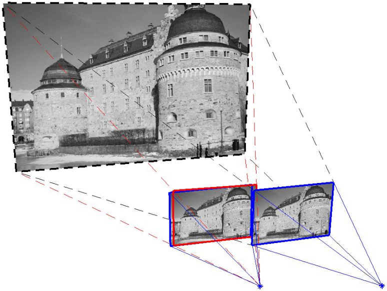
Plane Sweep Algorithm

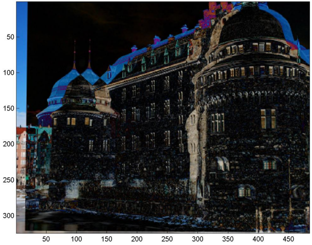


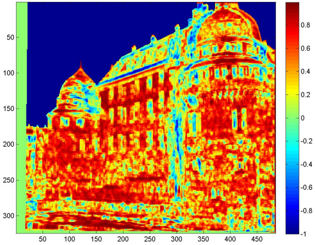




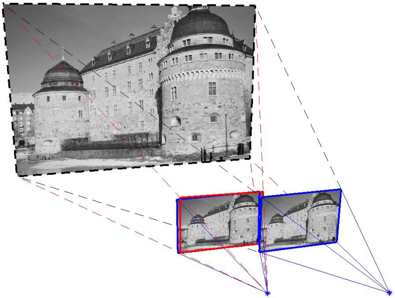
Plane Sweep Algorithm

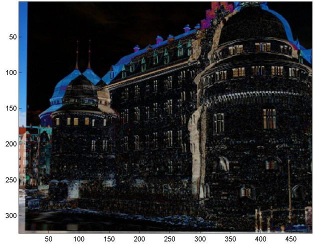


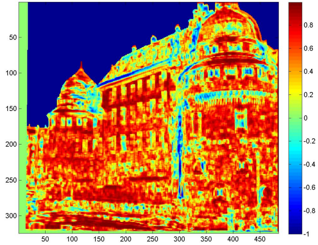




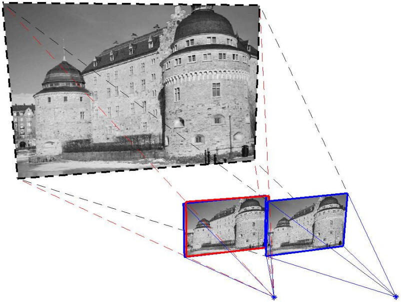
Plane Sweep Algorithm

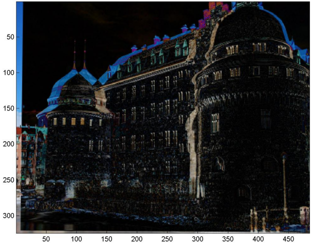


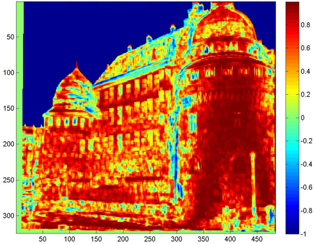




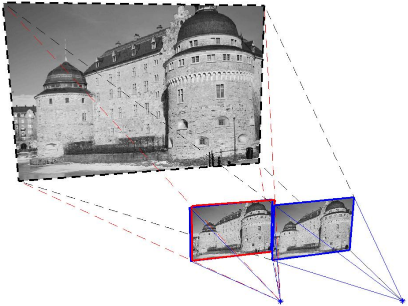
Plane Sweep Algorithm

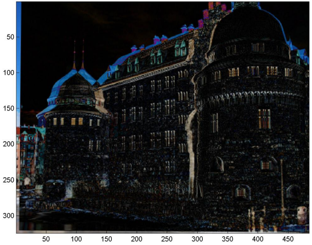


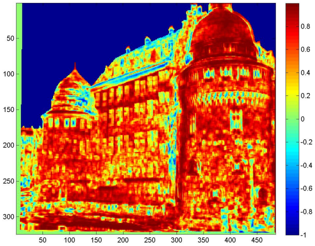




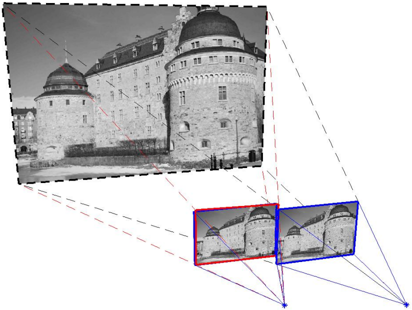
Plane Sweep Algorithm

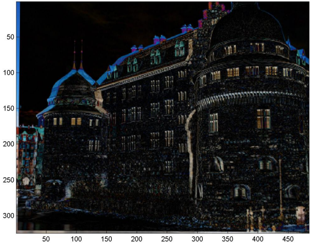


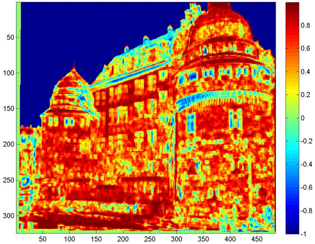




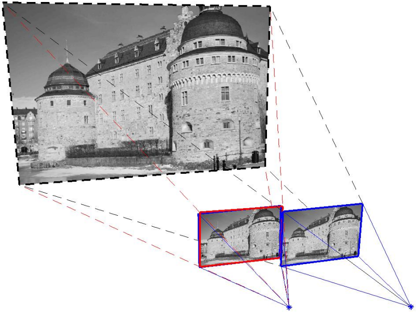
Plane Sweep Algorithm

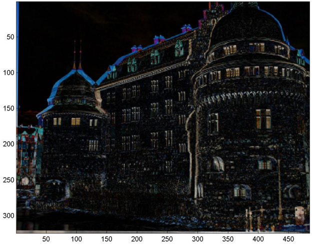


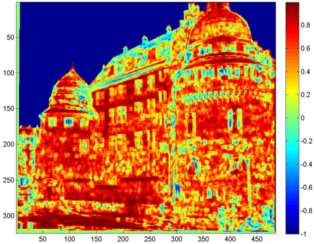




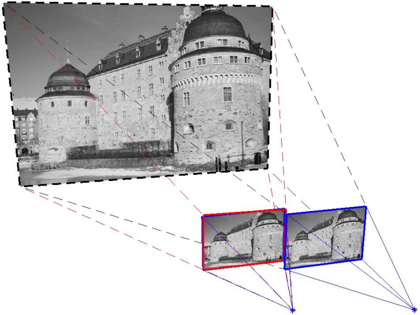
Plane Sweep Algorithm

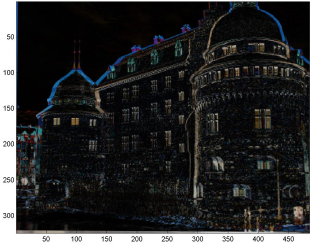


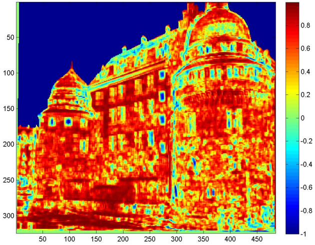




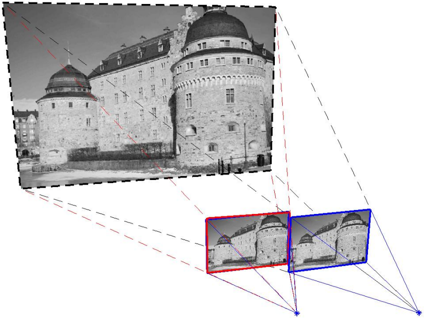
Plane Sweep Algorithm

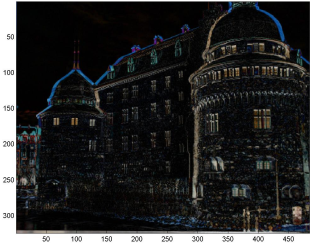


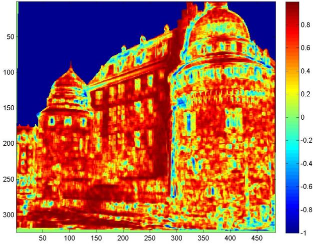




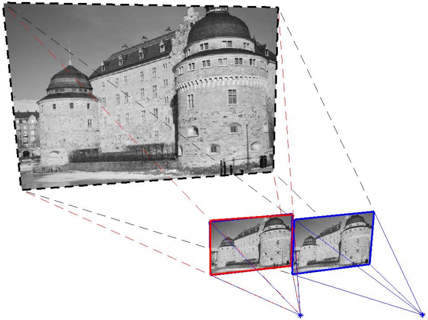
Plane Sweep Algorithm

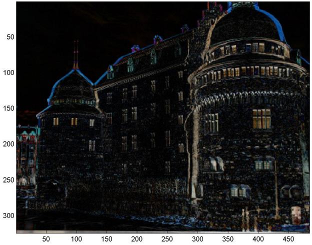


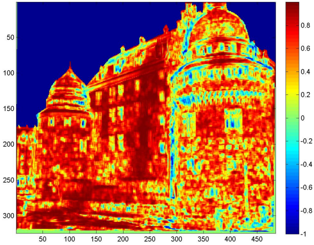




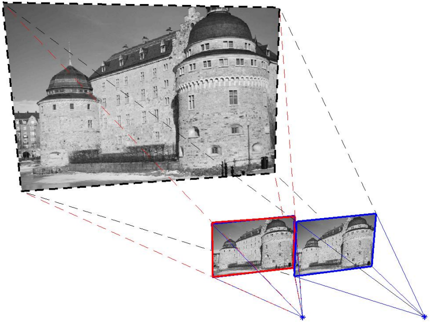
Plane Sweep Algorithm

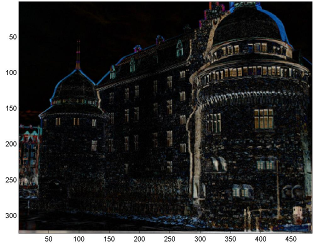


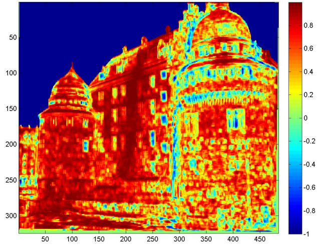




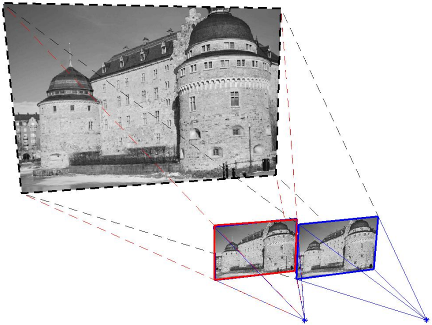
Plane Sweep Algorithm

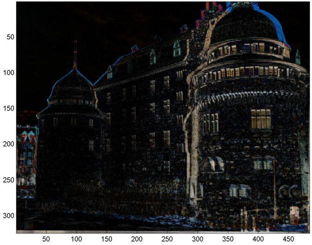


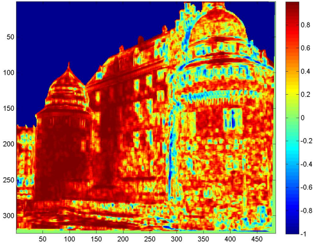




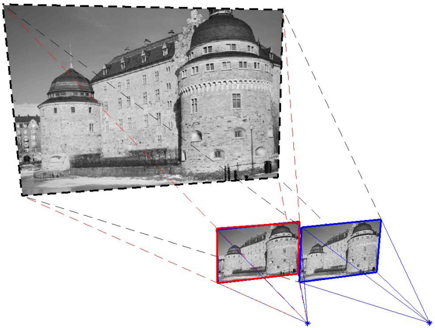
Plane Sweep Algorithm

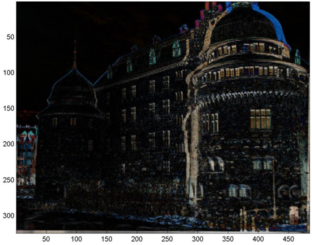


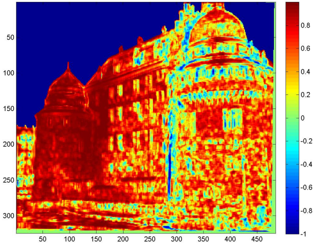




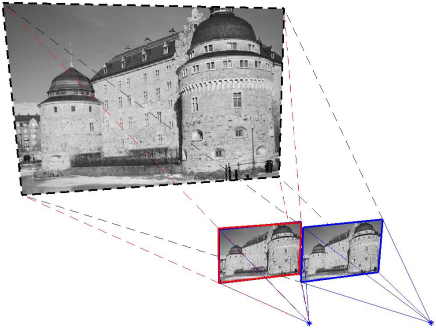
Plane Sweep Algorithm

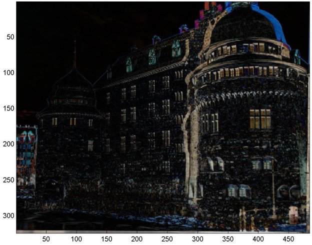


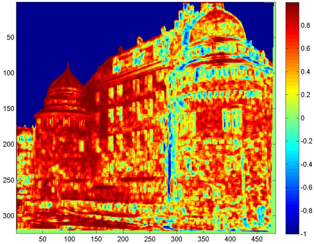




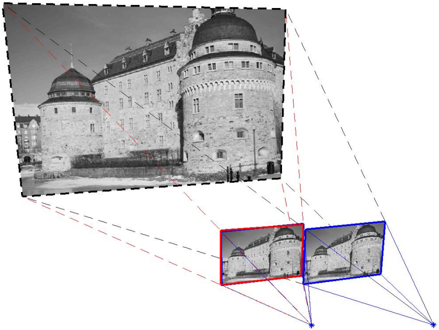
Plane Sweep Algorithm

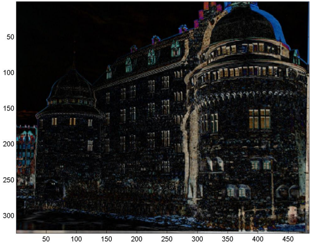


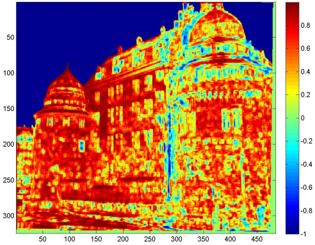




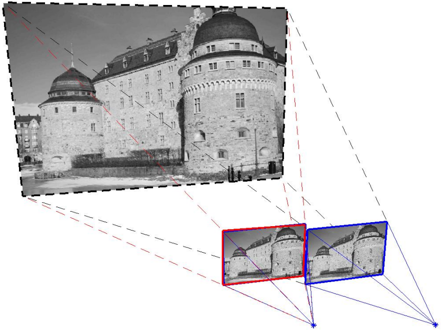
Plane Sweep Algorithm

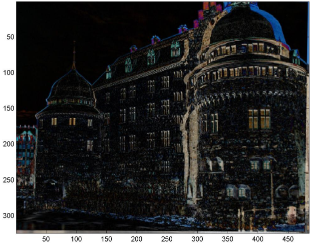


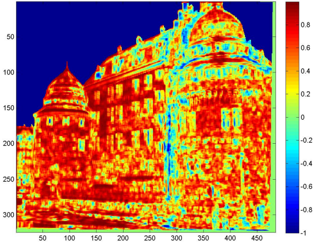




Plane Sweep Algorithm

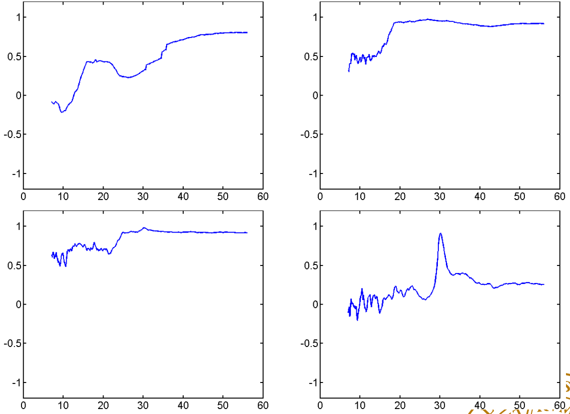






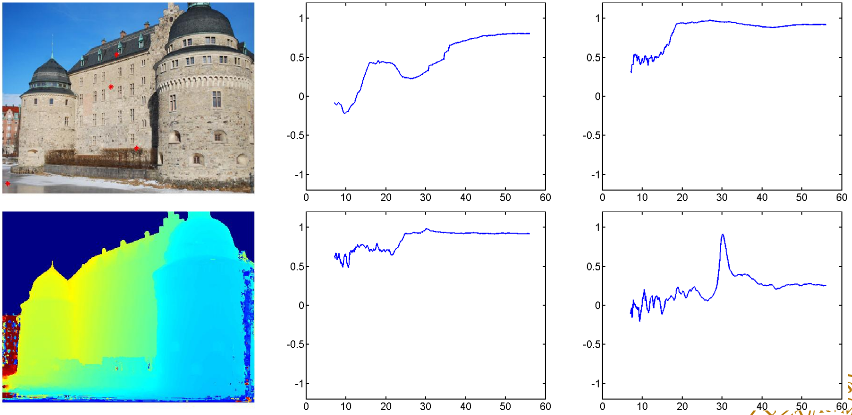
Plane Sweep Algorithm

Gives a function of depth for each pixel:

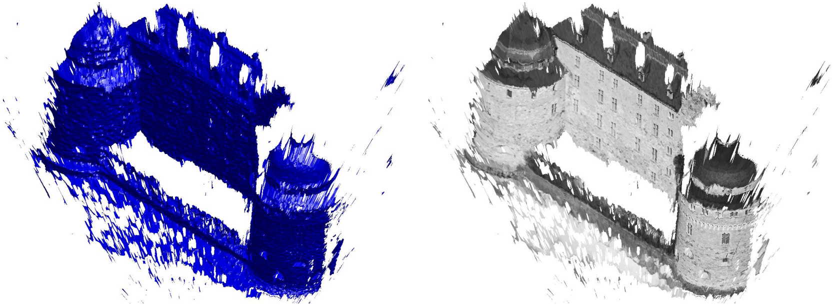


Plane Sweep Algorithm

Gives a function of depth for each pixel:



Energy Minimization & Regularization



Select the best value for each pixel

$$\min_{d} \sum_{i} E_i(d_i)$$

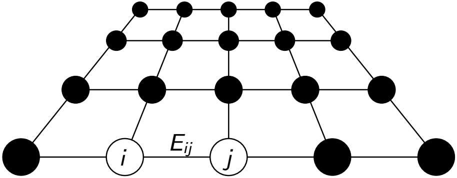
independently of its neighbors.

Energy Minimization & Regularization

Neighboring pixels tend to have similar depth. Add a regularization term

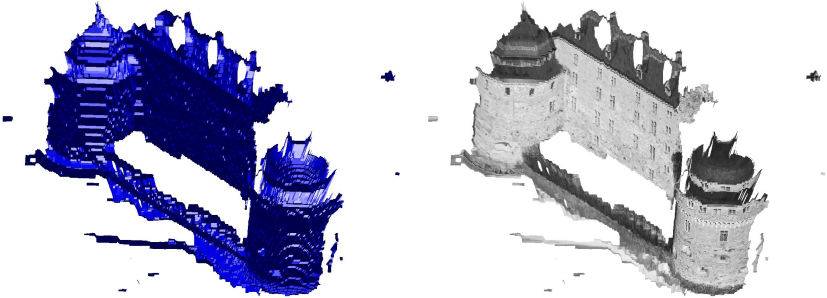
$$\min_d \sum_i \sum_{j \in \mathcal{N}(i)} E_{ij}(d_i, d_j) + \sum_i E_i(d_i)$$

Each pixel can be seen as a node in a graph. E_{ij} can be seen as an edge cost for the edge between nodes i and j.



Can minimize the energy using graph algorithms, e.g. graph cuts, message passing.

Energy Minimization & Regularization



Penalize neighbors that have different depth

$$\min_{d} \sum_{i} \sum_{j \in \mathcal{N}(i)} \min(|d_i - d_j|, tr) + \sum_{i} E_i(d_i)$$

Energy Minimization & Regularization

Taylor:

$$d_j \approx d_i + \nabla d_i^T (x_j - x_i)$$

x_i, x_j are the coordinates of the pixels i, j. Therefore

$$|d_i - d_j|$$

penalizes tilted planes.

Use

$$|d_i + \nabla d_i^T (x_j - x_i) - d_j| \approx |\frac{1}{2}(x_j - x_i)^T \nabla^2 d_i (x_j - x_i)|$$

instead. 2nd derivative is zero for affine functions.

Energy Minimization & Regularization

Taylor:

$$d_j \approx d_i + \nabla d_i^T (x_j - x_i)$$

x_i, x_j are the coordinates of the pixels i, j. Therefore

$$|d_i - d_j|$$

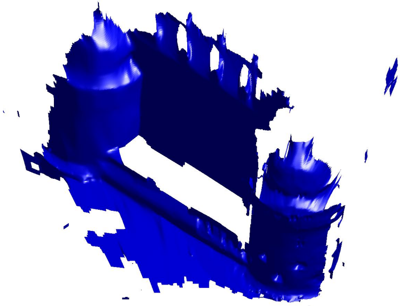
penalizes tilted planes.

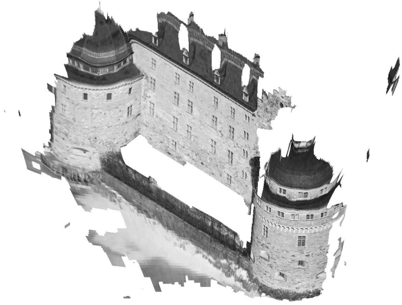
Use

$$|d_i + \nabla d_i^T (x_j - x_i) - d_j| \approx |\frac{1}{2}(x_j - x_i)^T \nabla^2 d_i (x_j - x_i)|$$

instead. 2nd derivative is zero for affine functions.

Energy Minimization & Regularization



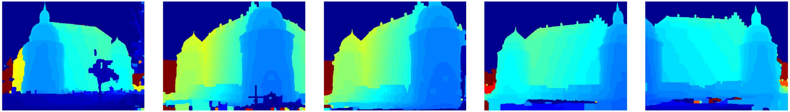


$$\min_d \sum_i \sum_{j \in \mathcal{N}(i)} \min(|d_i + \nabla d_i^T (x_j - x_i) - d_j|, tr) + \sum_i E_i(d_i)$$

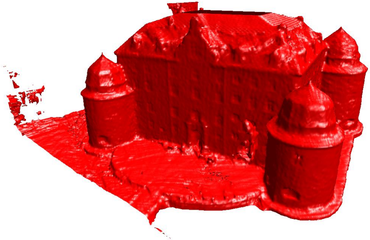
Movie

Dense Surface Reconstruction

Combine all the depth maps into one surface. (Voxel carving algorithm.)



...



More Movies

Silhouettes and Visual Hull

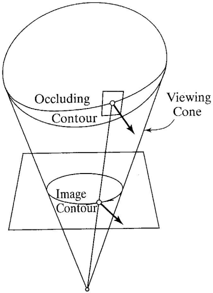


Figure 19.1 The occlusion boundaries of a smooth

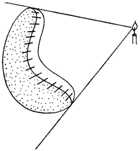


Figure 19.3 Shadow boundaries and occluding contours. *Reprinted from "Solid Shape," by J.J. Koenderink, MIT Press, (1990). © 1990 by The Massachusetts Institute of Technology.*

The viewed object can be constrained to lie within the viewing cone. (Viewing cone = all rays through the camera center and the interior of the projection.)

Visual hull = intersection of all viewing cones.

Largest volume contained in the intersection.

Voxel Carving

Problem

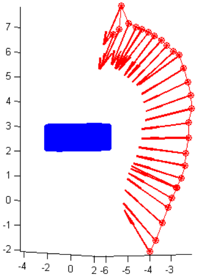
Given cameras P and segmented silhouettes compute a volumetric representation of the viewed object.

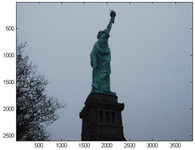
Algorithm

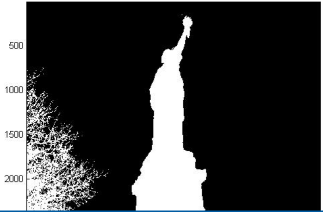
- For each image, project the (center of the) voxels into the image and determine which ones belong to the exterior of the object.
- Remove the voxels that belongs to the exterior of the object in at least one image.

Voxel Carving

Example.

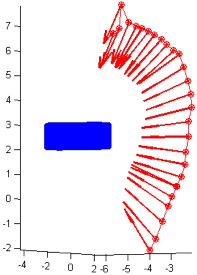


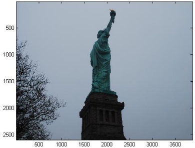


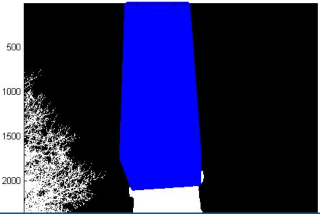


Voxel Carving

Example.

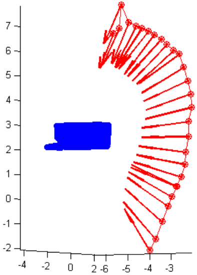


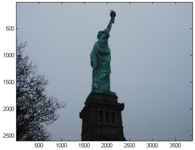


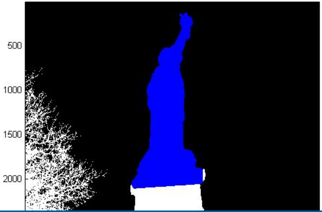


Voxel Carving

Example.

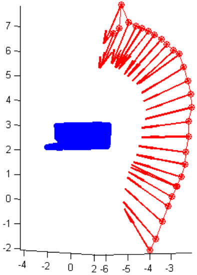


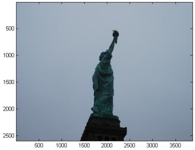




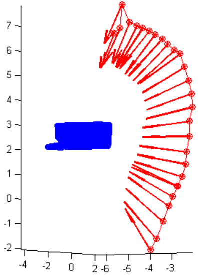
Voxel Carving

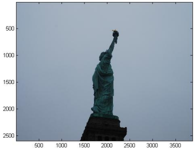
Example.





Example.

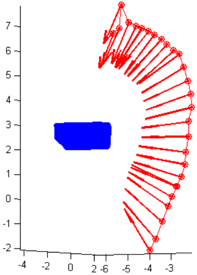

7
6
5
4
3
2
1
0
-1
-2
-4
-2
0
2
-6
-5
-4
-3

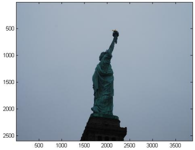

500
1000
1500
2000
2500
500
1000
1500
2000
2500
3000
3500

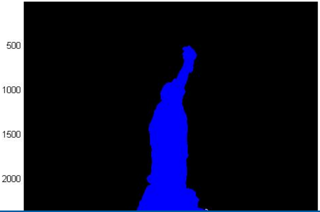

500
1000
1500
2000

Voxel Carving

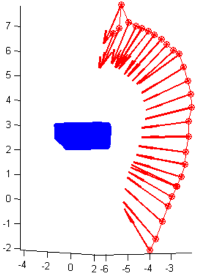
Example.

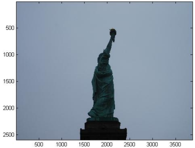






Example.

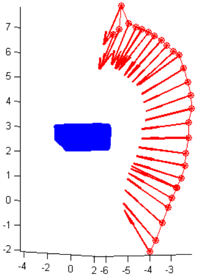

7
6
5
4
3
2
1
0
-1
-2
-4
-2
0
2
-6
-5
-4
-3

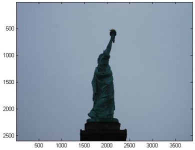

500
1000
1500
2000
2500
500
1000
1500
2000
2500
3000
3500

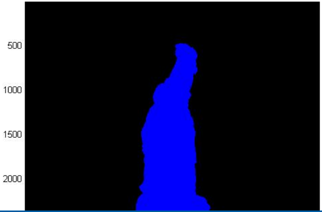

500
1000
1500
2000

Voxel Carving

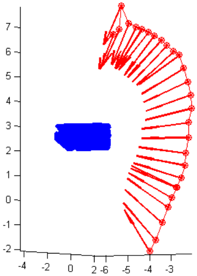
Example.

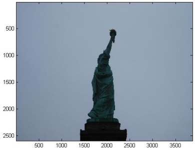


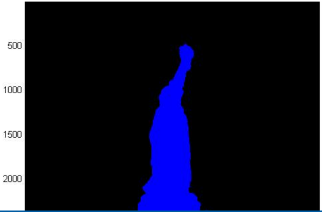




Example.

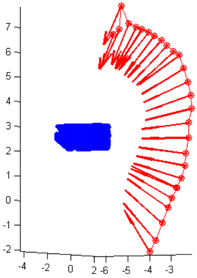

7
6
5
4
3
2
1
0
-1
-2
-4
-2
0
2 -6
-5
-4
-3

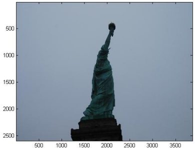

500
1000
1500
2000
2500
500
1000
1500
2000
2500
3000
3500

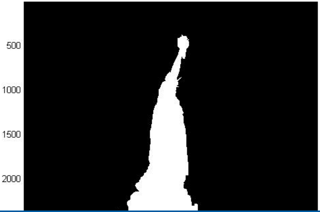

500
1000
1500
2000

Voxel Carving

Example.

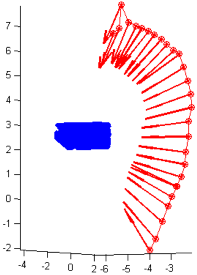


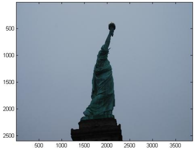


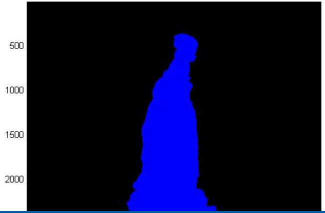


Voxel Carving

Example.

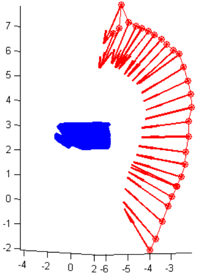


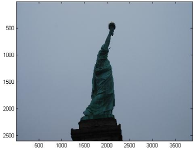


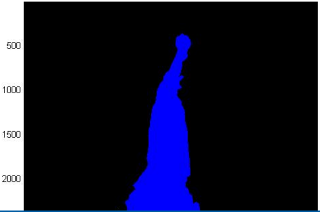


Voxel Carving

Example.

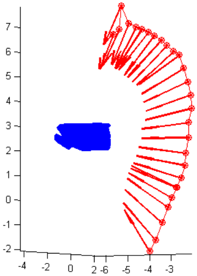


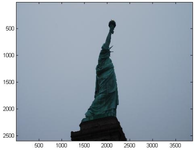


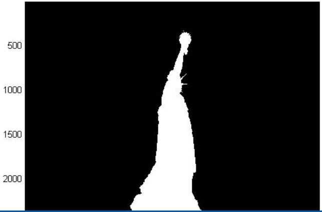


Voxel Carving

Example.

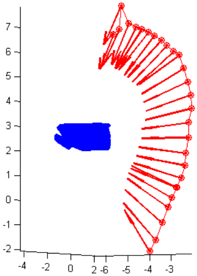


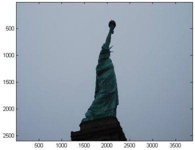


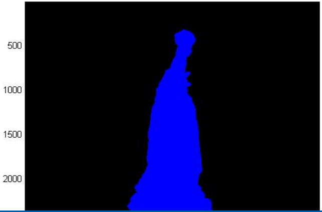


Voxel Carving

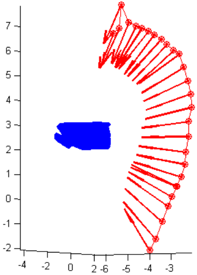
Example.

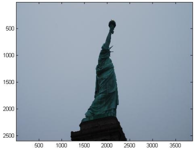


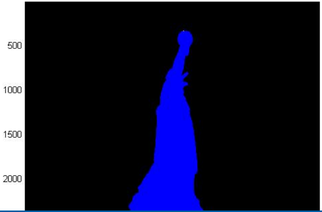




Example.

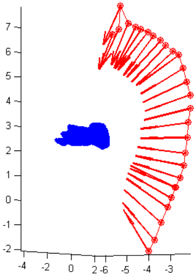

7
6
5
4
3
2
1
0
-1
-2
-4
-2
0
2
-6
-5
-4
-3

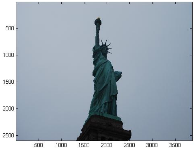

500
1000
1500
2000
2500
500
1000
1500
2000
2500
3000
3500


500
1000
1500
2000

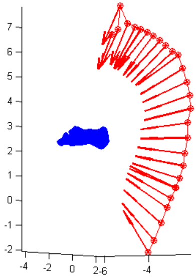
Voxel Carving

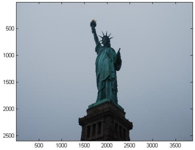
Example.

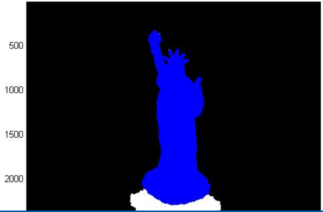




Example.


7
6
5
4
3
2
1
0
-1
-2
-4
-2
0
2-6
-4


500
1000
1500
2000
2500
500
1000
1500
2000
2500
3000
3500


500
1000
1500
2000

Voxel Carving

Project texture on it.