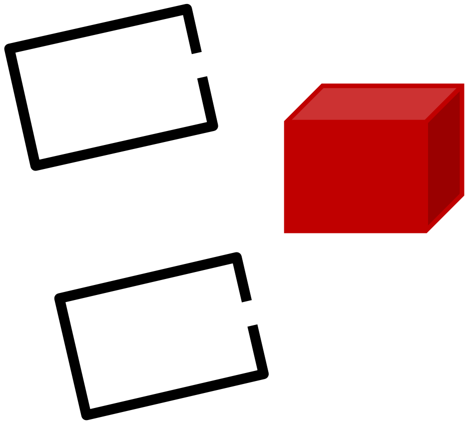
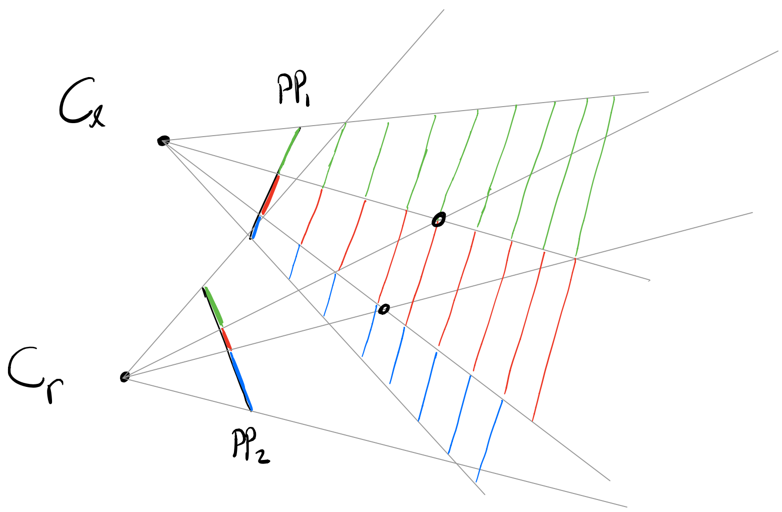
CSCI 497P/597P: Computer Vision

Scott Wehrwein

Stereo: Metrics, Rectification
Planesweep Stereo





Announcements

- Reminder: Exam
  - out this morning
  - due Tuesday night
- P[redacted]2
  - code due tonight
  - artifact due Tuesday night

Goals

- Understand the basic metrics used to compare patches (SSD, SAD, NCC)
- Understand how to rectify a pair of stereo images given their intrinsics and extrinsics.
- Understand the plane sweep stereo algorithm.

A Stereo Algorithm

1. For every pixel (x, y)
    1. For every disparity d
        1. Get patch from image 1 at (x, y)
        2. Get patch from image 2 at (x + d, y)
        3. Compute cost using your metric of choice

```
C = np.array(h,w,D)
for r in range(0,h):
  for c in range(0,w):
    for d in range(min_d, max_d):
      C[r,c,d] = metric(get_patch(im1,r,c), get_patch(im2,r,c+d)

disp = np.argmin(C, axis=2)
depth = f * b / disp
```

Metrics for Stereo Matching

iml patch = w_1 img2 patch = w_2

- SSD = sum of squared differences

np.sum((w2 - w1)**2)

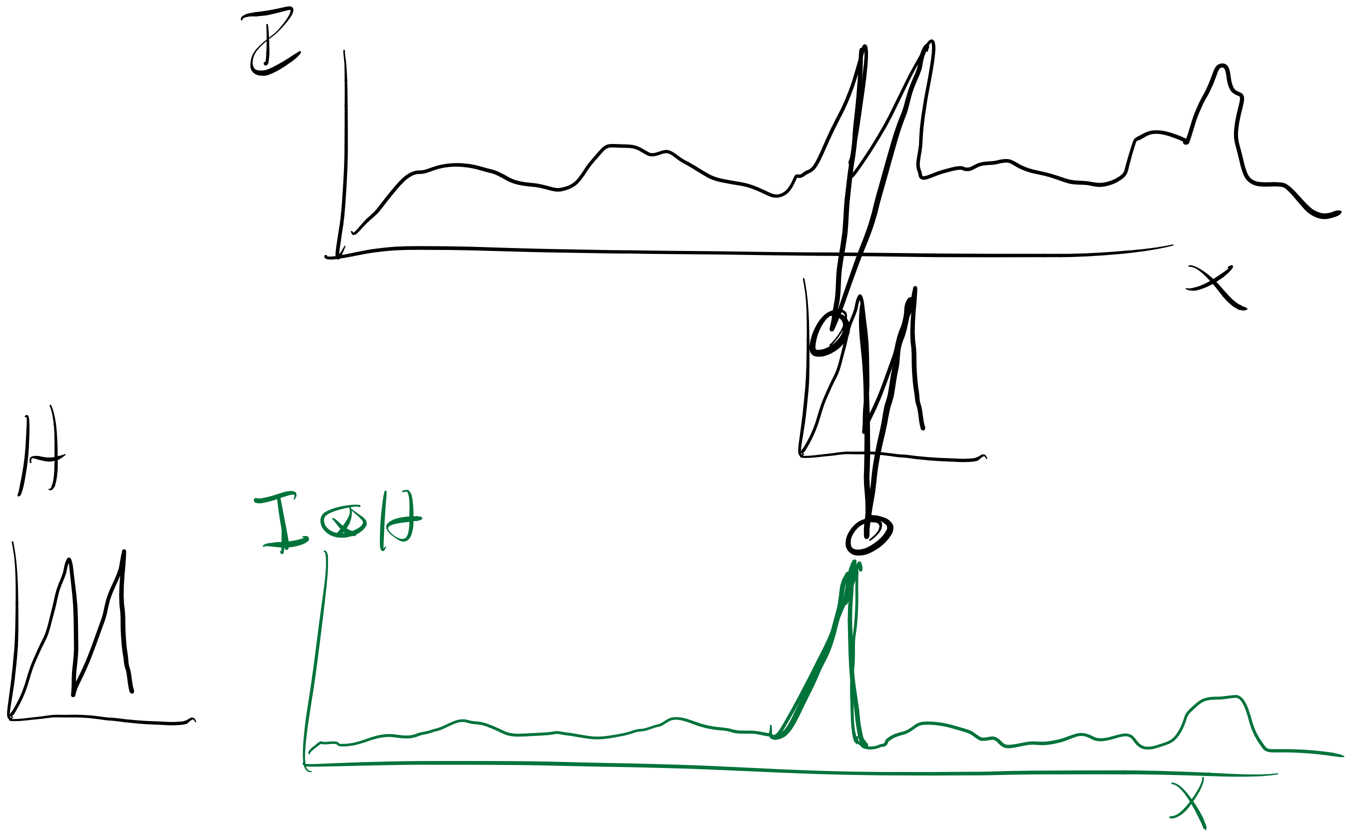
- SAD = sum of absolute differences

np.sum(np.abs(w2 - w1))

- NCC = normalized cross-correlation
  - (more ~~convolution~~cross correlation!)

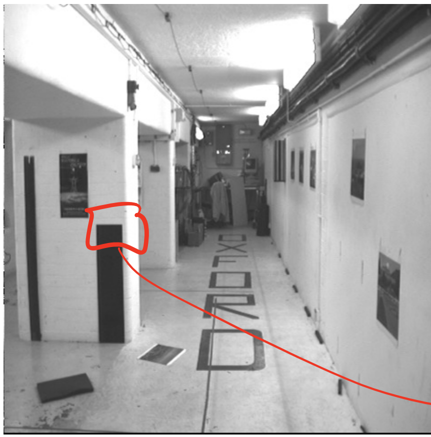
Un-Normalized Cross Correlation

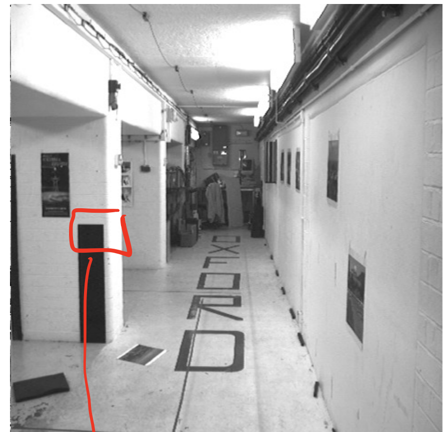
Insight: a cross-correlation filter is good at finding patches that **look like itself**.



Normalized Cross Correlation

Approach: apply a **patch** from one image as a **filter** across the other.
Trick: normalize patches before computing product to add invariance.





regions $\mathtt{A}, \mathtt{B}$, write as vectors $\mathbf{a}, \mathbf{b}$

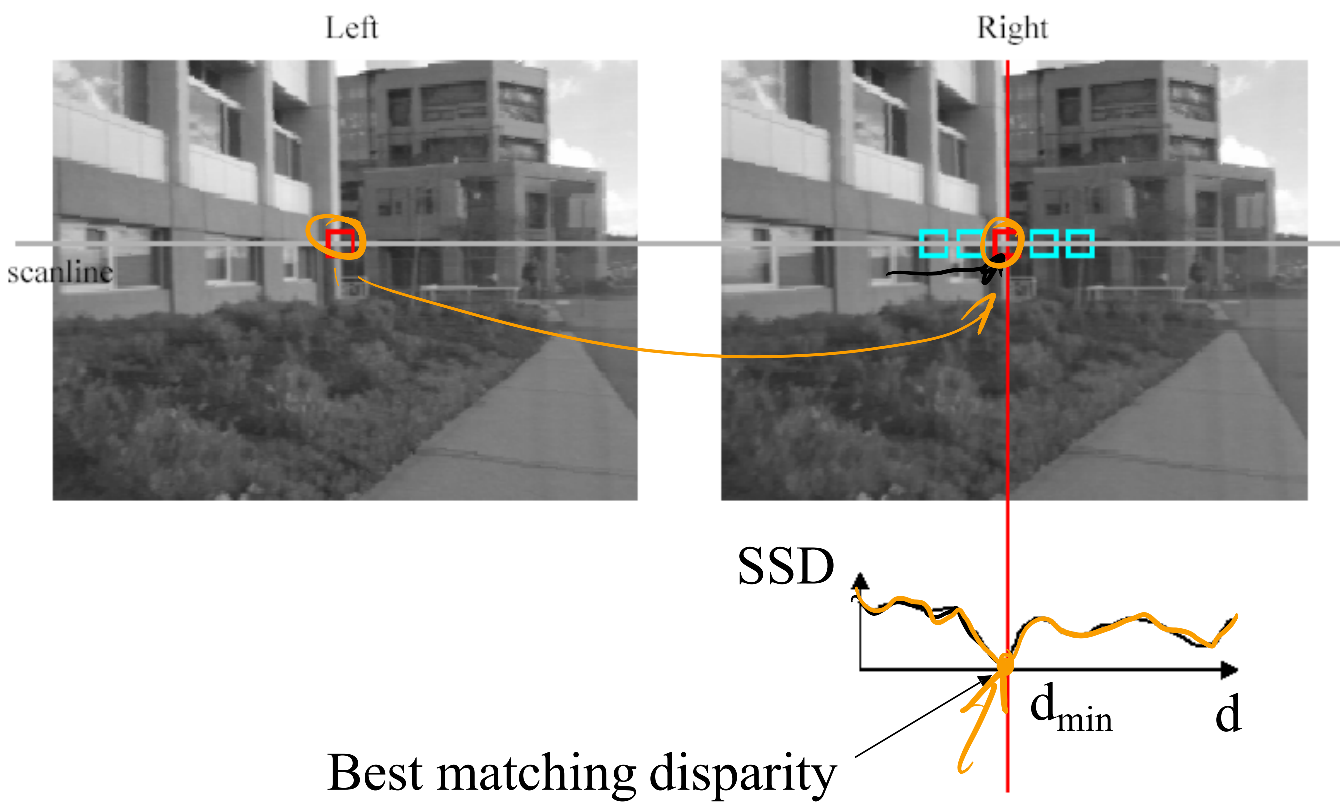
subtract the mean of each vector:

$$\mathtt{a} \rightarrow \mathtt{a} - \langle \mathbf{a} \rangle, \quad \mathtt{b} \rightarrow \mathtt{b} - \langle \mathbf{b} \rangle$$

$$\text{cross correlation} = \frac{\mathbf{a}.\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}$$

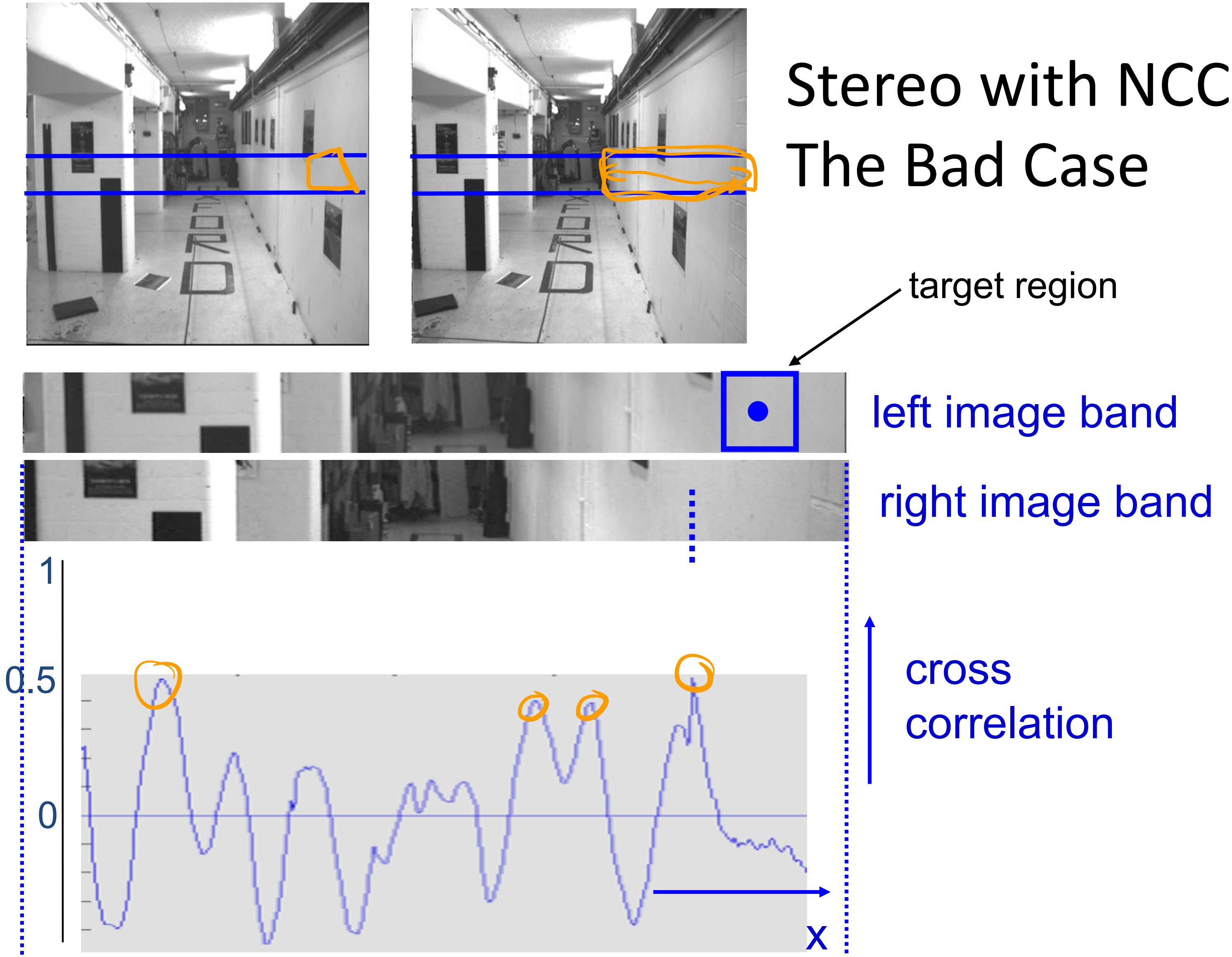
Invariant to $I \rightarrow \alpha I + \beta$

Stereo matching based on SSD



Stereo with NCC: The Good Case

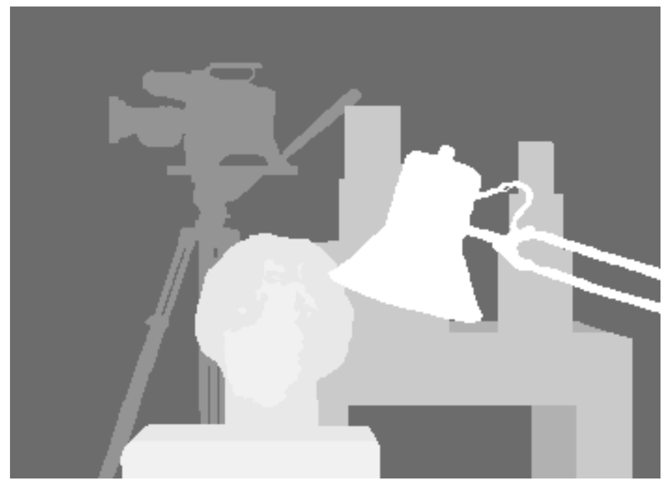
Stereo with NCC: The Bad Case



Stereo results

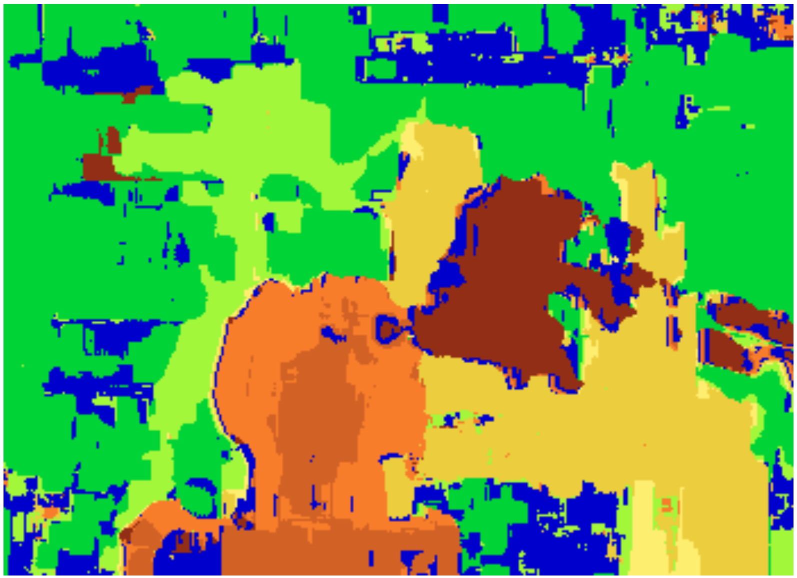
– Data from University of Tsukuba
– Similar results on other images without ground truth

Scene



Ground truth

Results with window search



Window-based matching
(best window size)

Ground truth

Better methods exist...

Fancier method

Ground truth

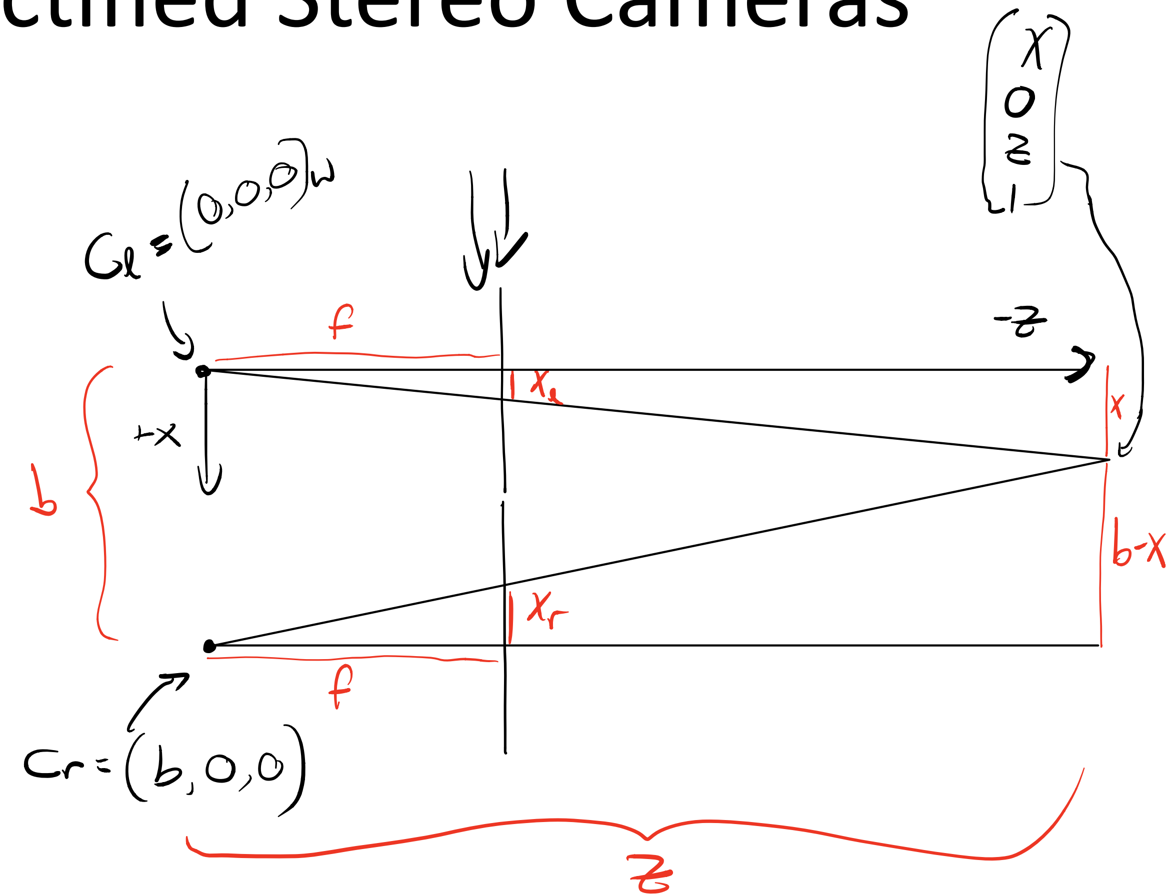
Boykov et al., Fast Approximate Energy Minimization via Graph Cuts, International Conference on Computer Vision, September 1999.

For the latest and greatest: http://www.middlebury.edu/stereo/

Rectified Stereo Cameras

- 2 cameras
- same f
- same pp
- cop off by b in x

↑ known



What if the cameras **aren't** rectified?

If cameras are **calibrated**, i.e., we know:

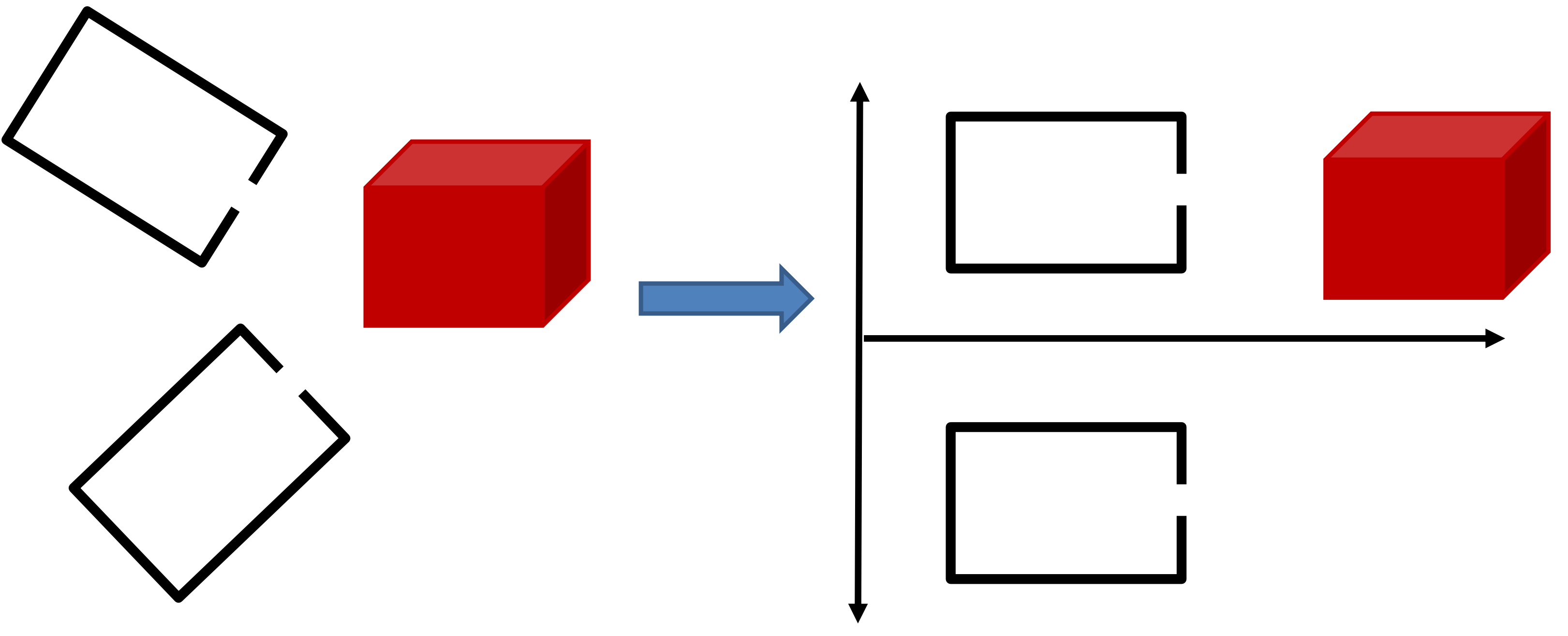
K_l K_r

R_l R_r

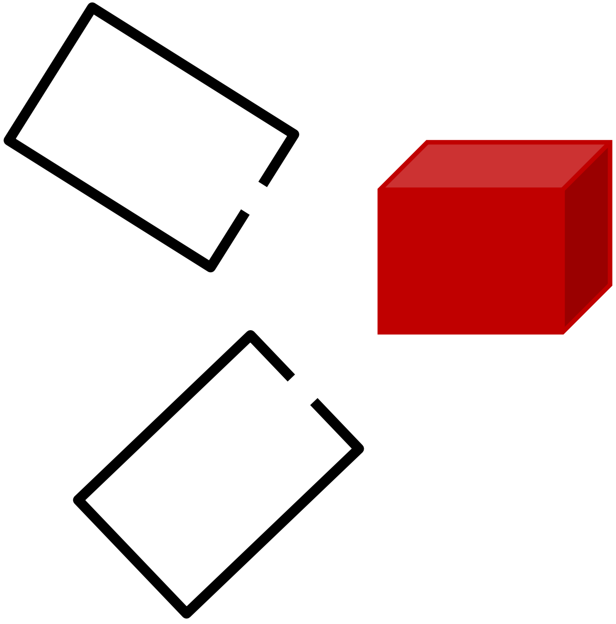
(aka c_l) t_l t_r

...then we can **rectify** the stereo pair
...or we can use **plane sweep stereo**

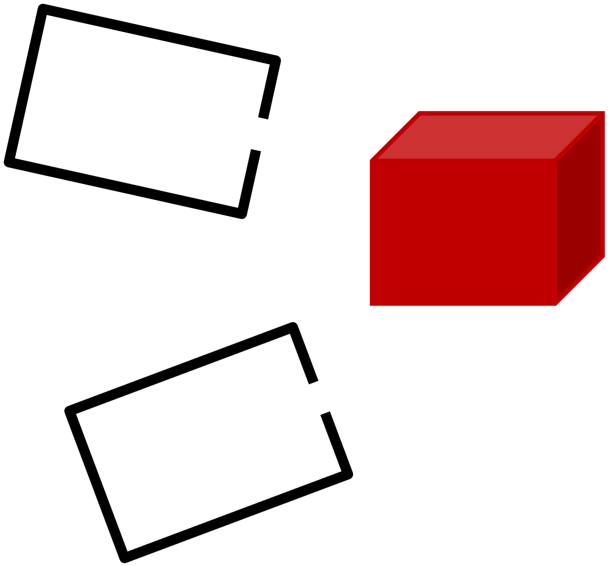
Rectifying cameras



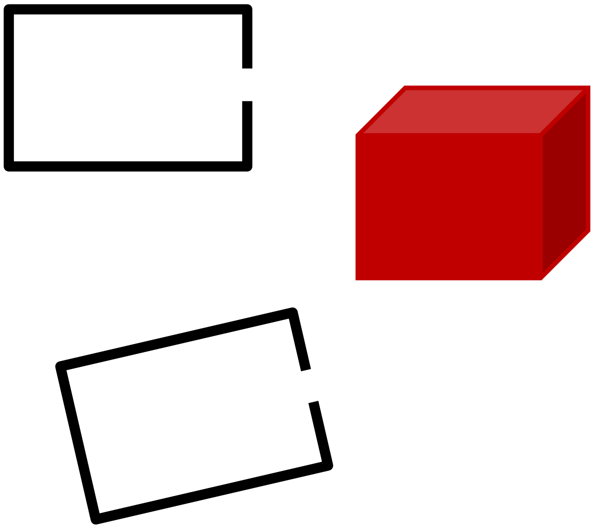
Rectifying cameras



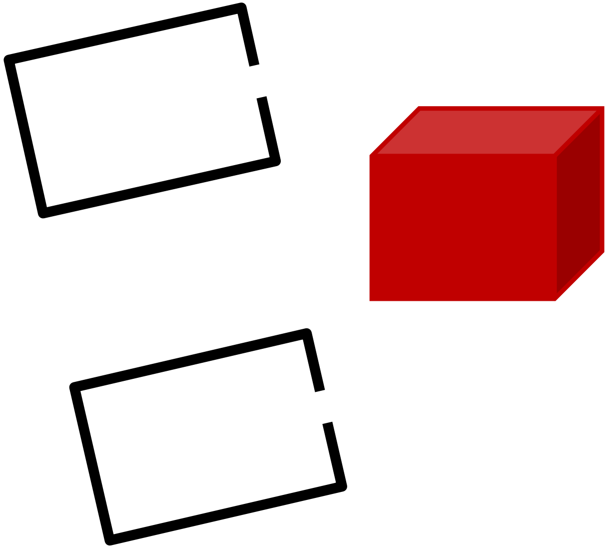
Rectifying cameras



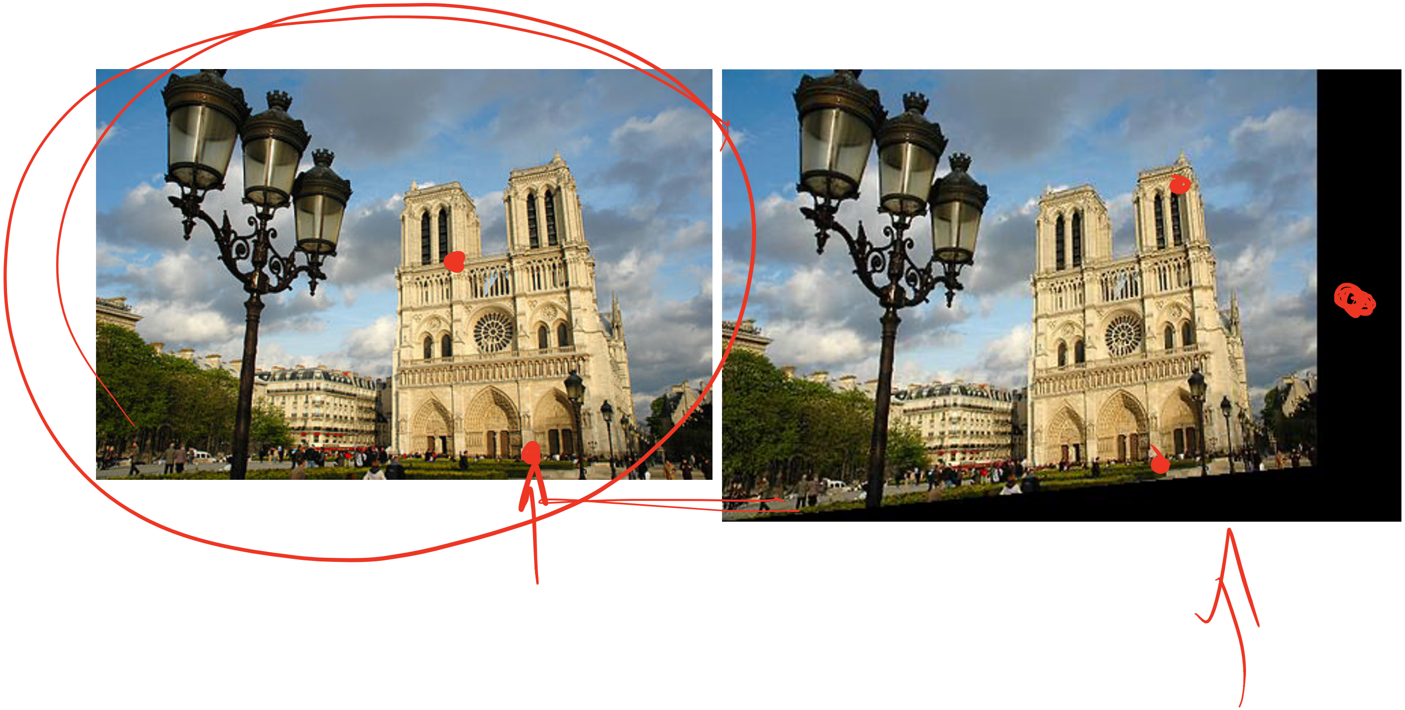
Rectifying cameras



Rectifying cameras

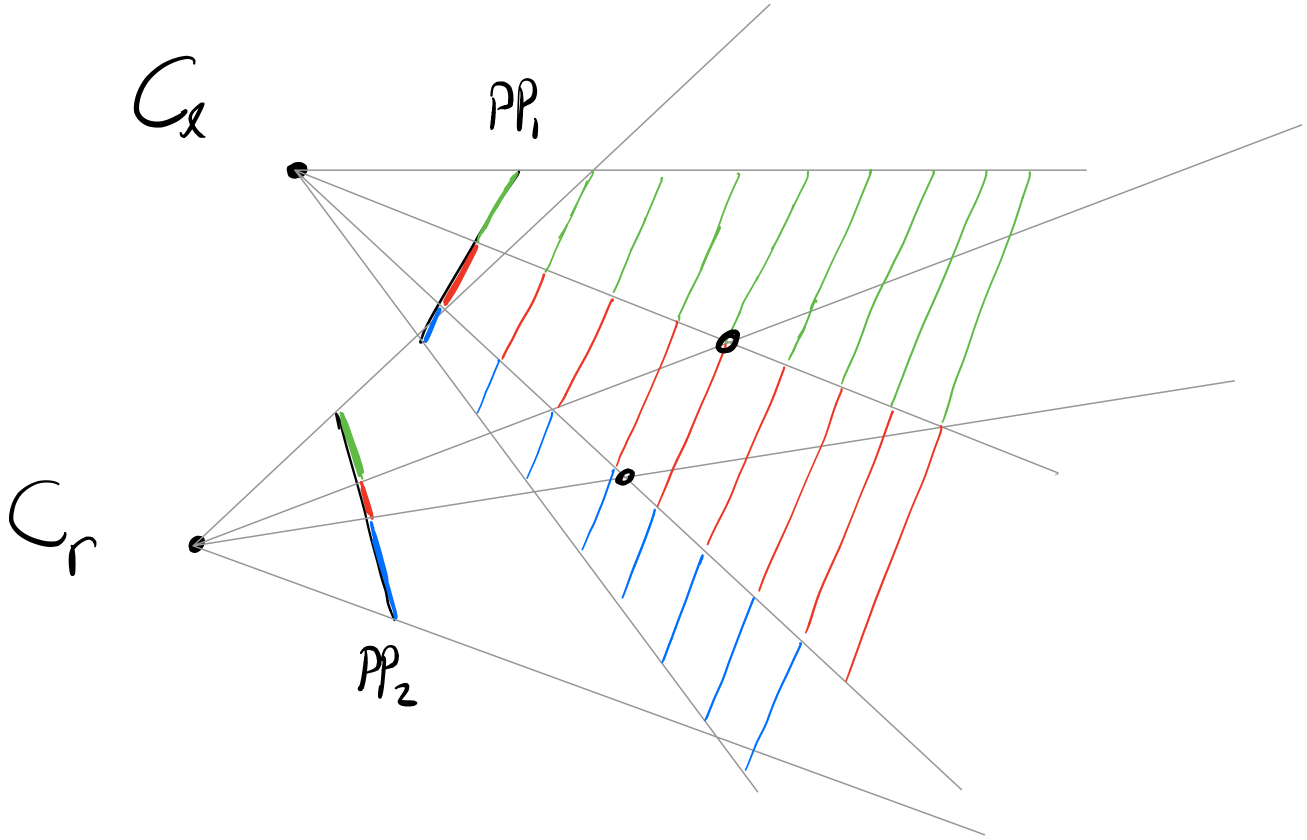


Example: A rectified stereo pair



GOTO notes

Plane Sweep Stereo



A Stereo Algorithm

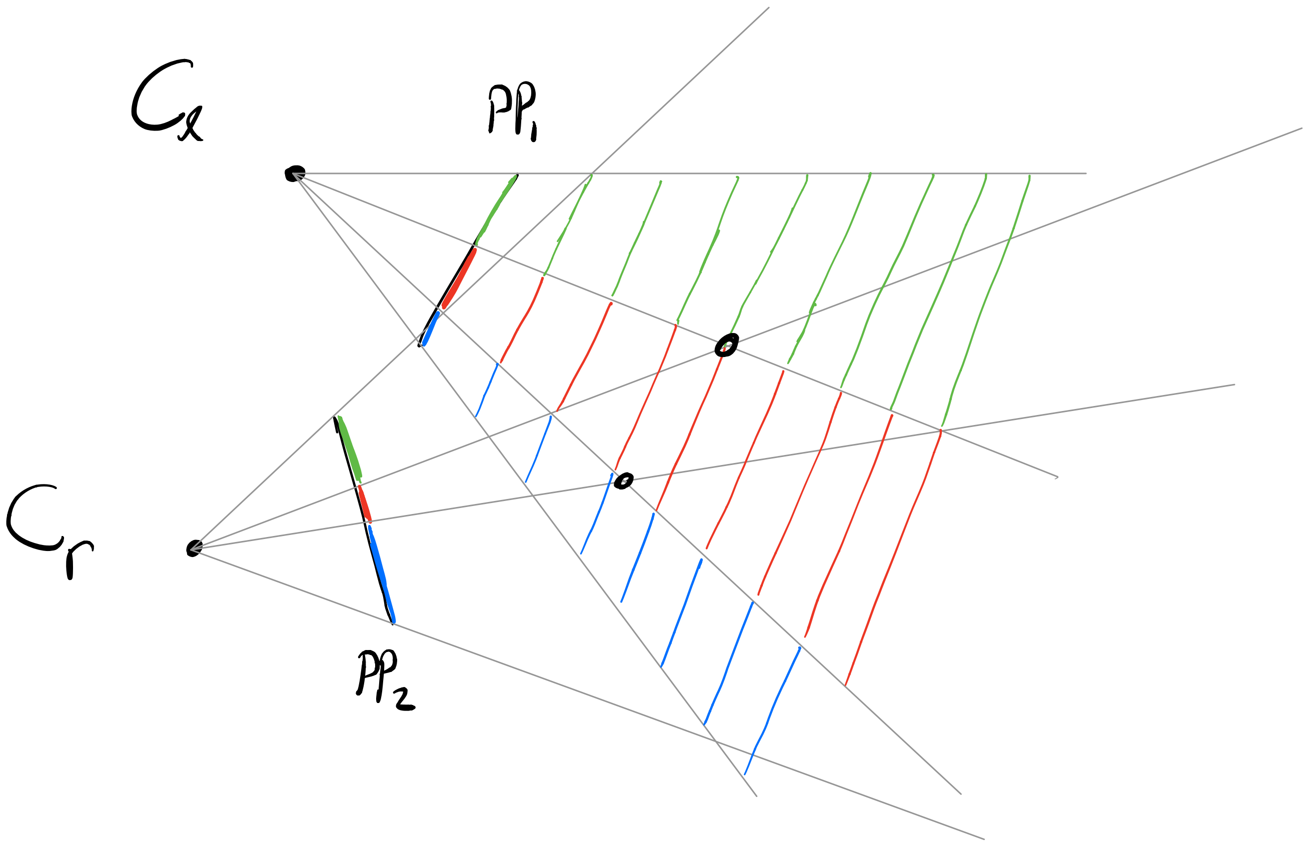
1. For every pixel (x, y)
    1. For every disparity d
        1. Get patch from image 1 at (x, y)
        2. Get patch from image 2 at (x + d, y)
        3. Compute cost using your metric of choice

```
C = np.array(h,w,d)
for r in range(0,h):
  for c in range(0,w):
    for d in range(-maxd, maxd):
      C[r,c,d] = metric(get_patch(im1,r,c), get_patch(im2,r,c+d)

disp = np.max(C, axis=2)
depth = f * b / disp
```

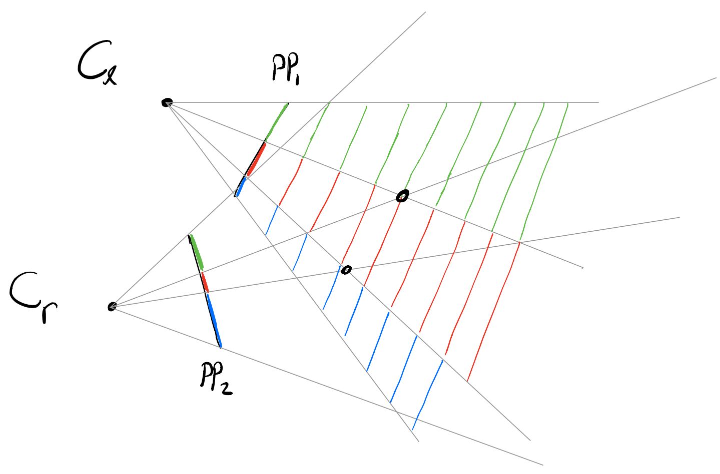
Plane Sweep Stereo Algorithm

1. For every disparity d
    1. For every pixel (x, y)
        1. Get patch from image 1 at (x, y)
        2. Get patch from image 2 at (x + d, y)
        3. Compute cost using your metric of choice

```
C = np.array(h,w,d)
for d in range(-maxd, maxd):
  for r in range(0,h):
    for c in range(0,w):
      C[r,c,d] = metric(get_patch(im1,r,c), get_patch(im2,r,c+d)

disp = np.max(C, axis=2)
depth = f * b / disp
```

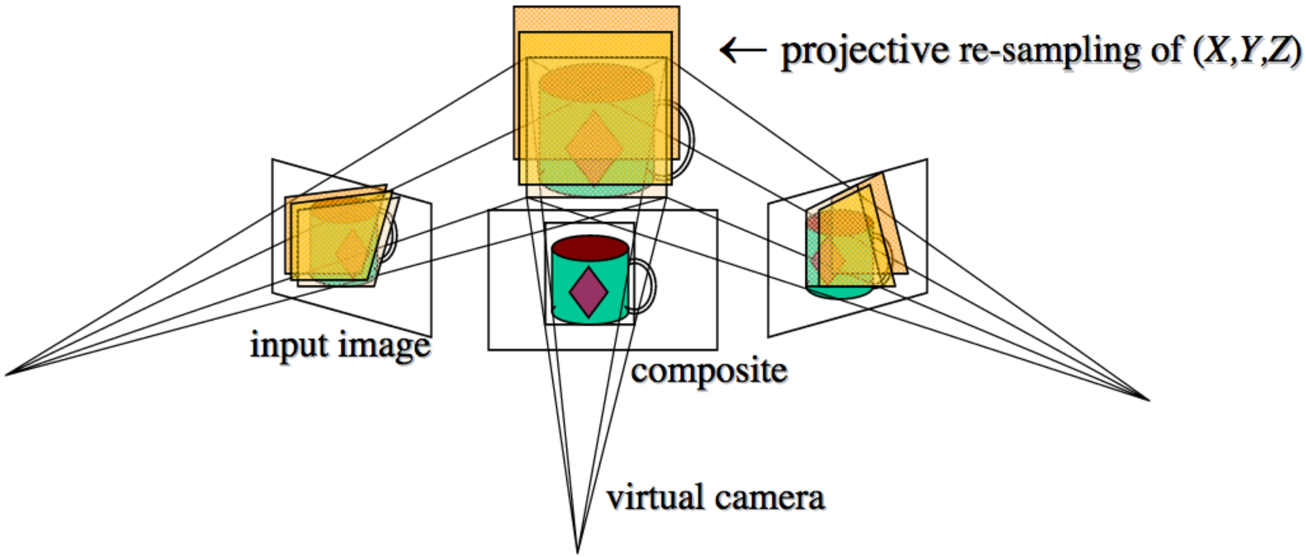


C_ℓ
PP1
C_r
PP2



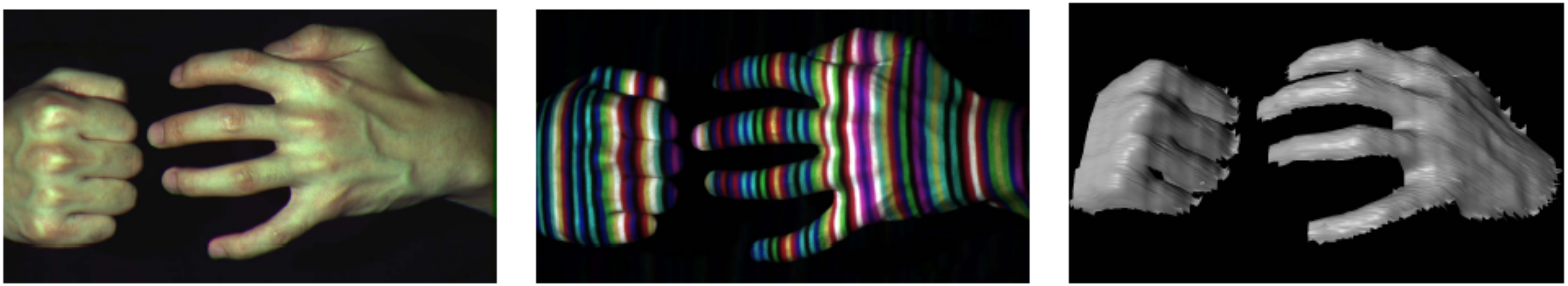
C_ℓ
PP1
C_r
PP2

Plane Sweep Stereo

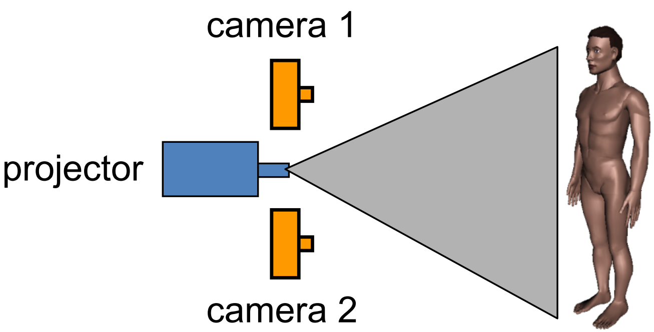


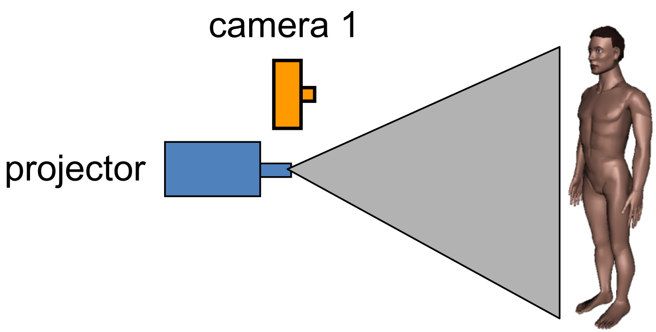
– each plane defines an image ⇒ composite homography

Active stereo with structured light



Li Zhang's one-shot stereo

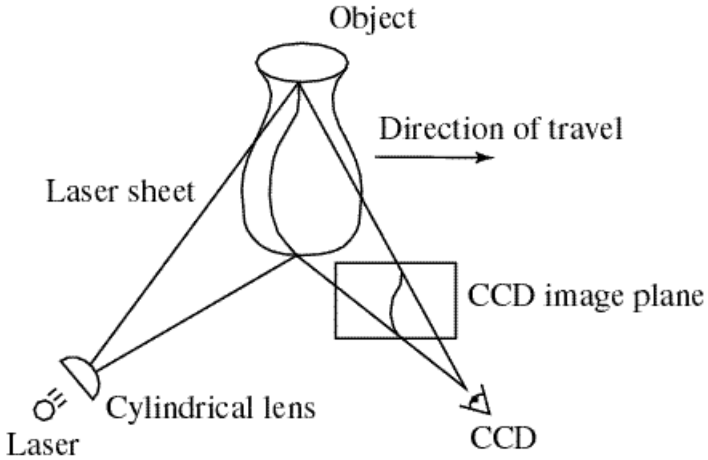


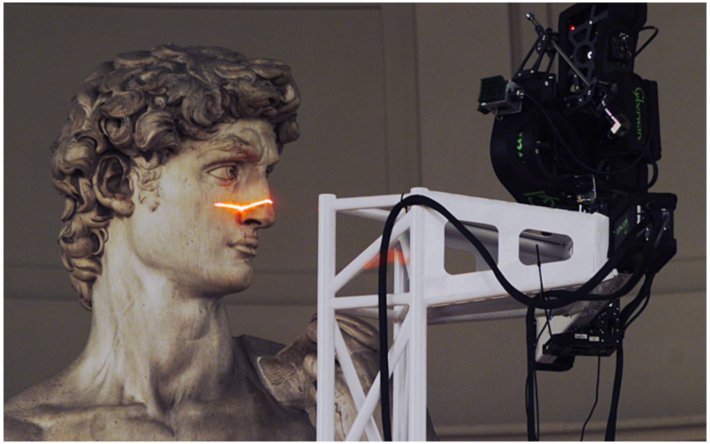


- Project "structured" light patterns onto the object
  - simplifies the correspondence problem
  - basis for active depth sensors, such as Kinect and iPhone X (using IR)

Other methods for getting depth

Laser scanning





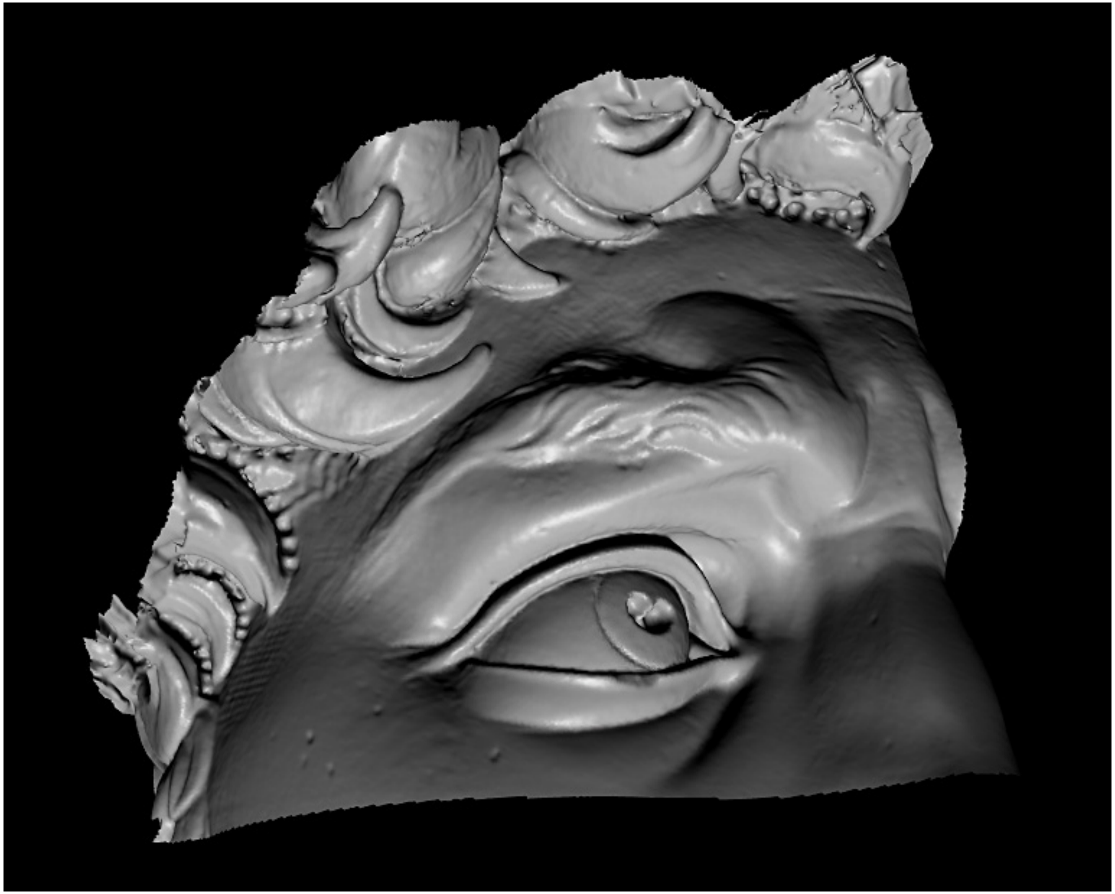
Digital Michelangelo Project
http://graphics.stanford.edu/projects/mich/

- Optical triangulation
  - Project a single stripe of laser light
  - Scan it across the surface of the object
  - This is a very precise version of structured light scanning

Laser scanned models

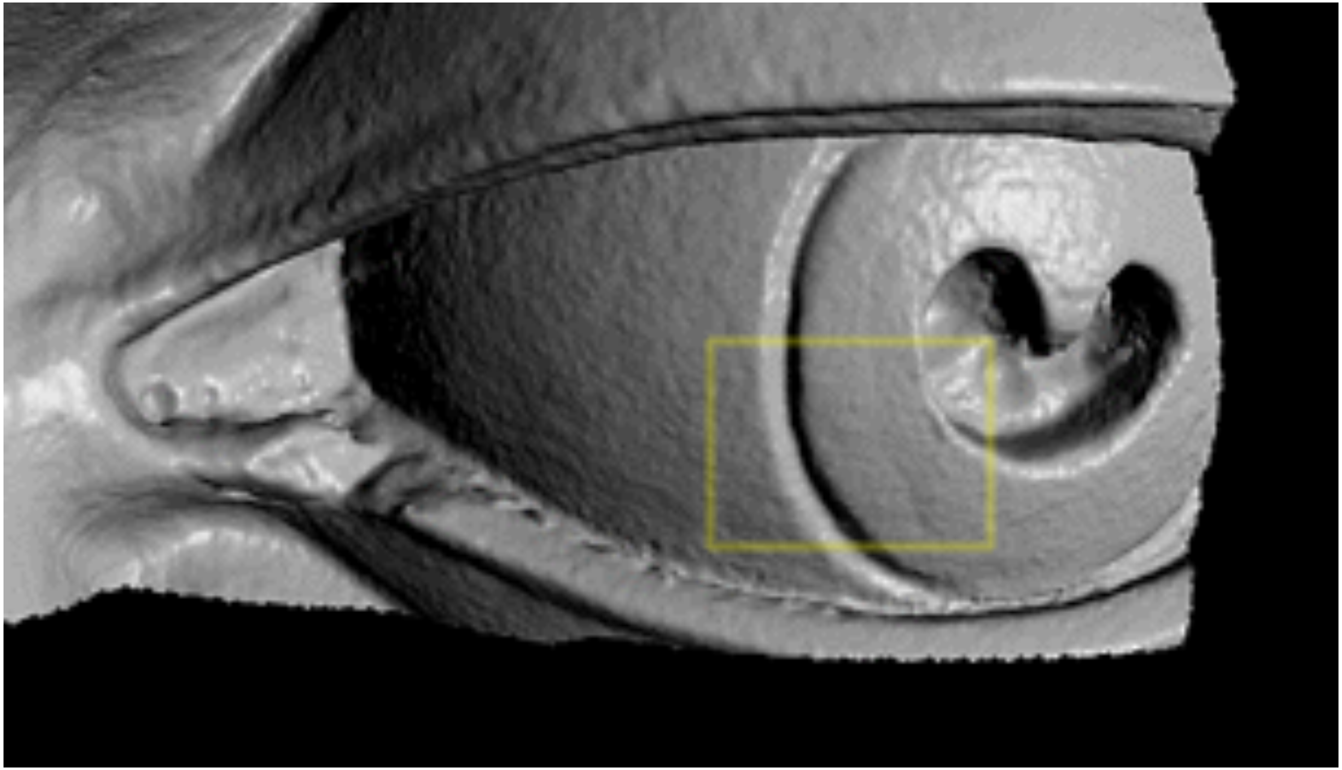
The Digital Michelangelo Project, Levoy et al.

Laser scanned models



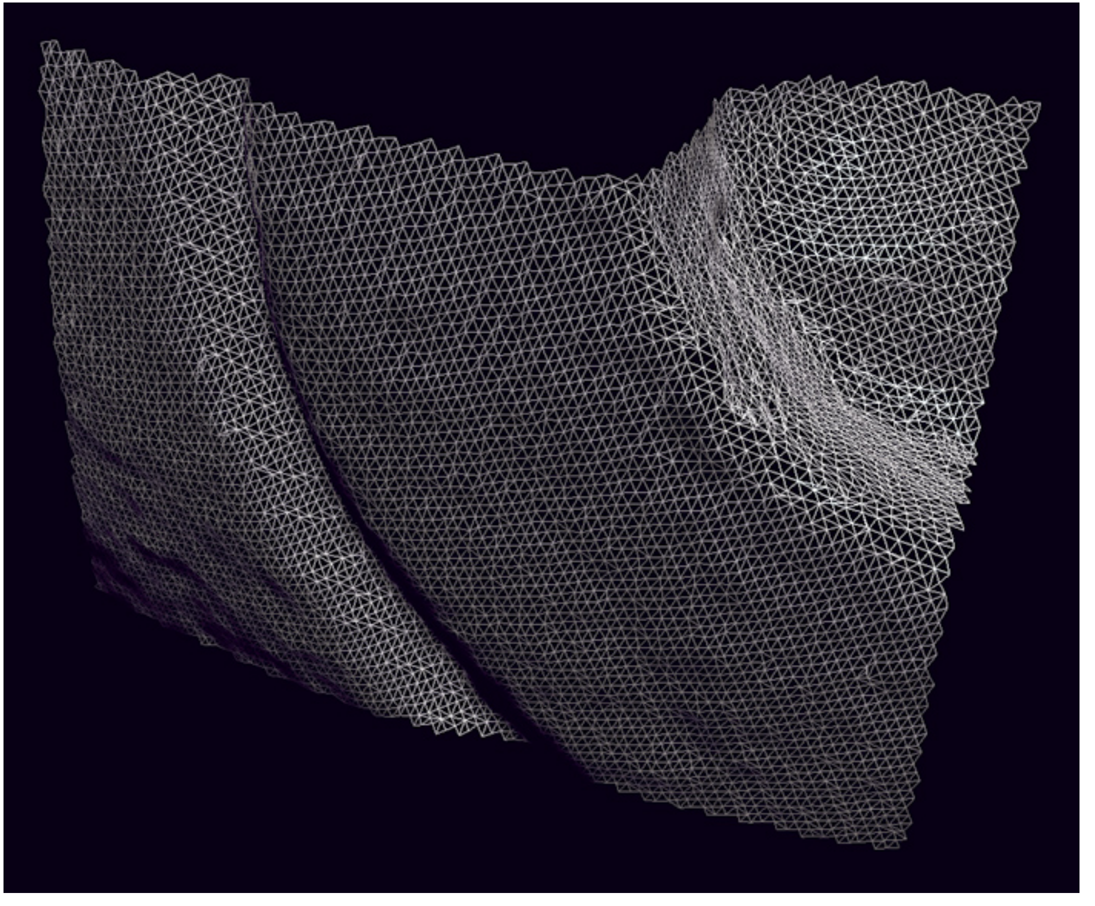
The Digital Michelangelo Project, Levoy et al.

Laser scanned models



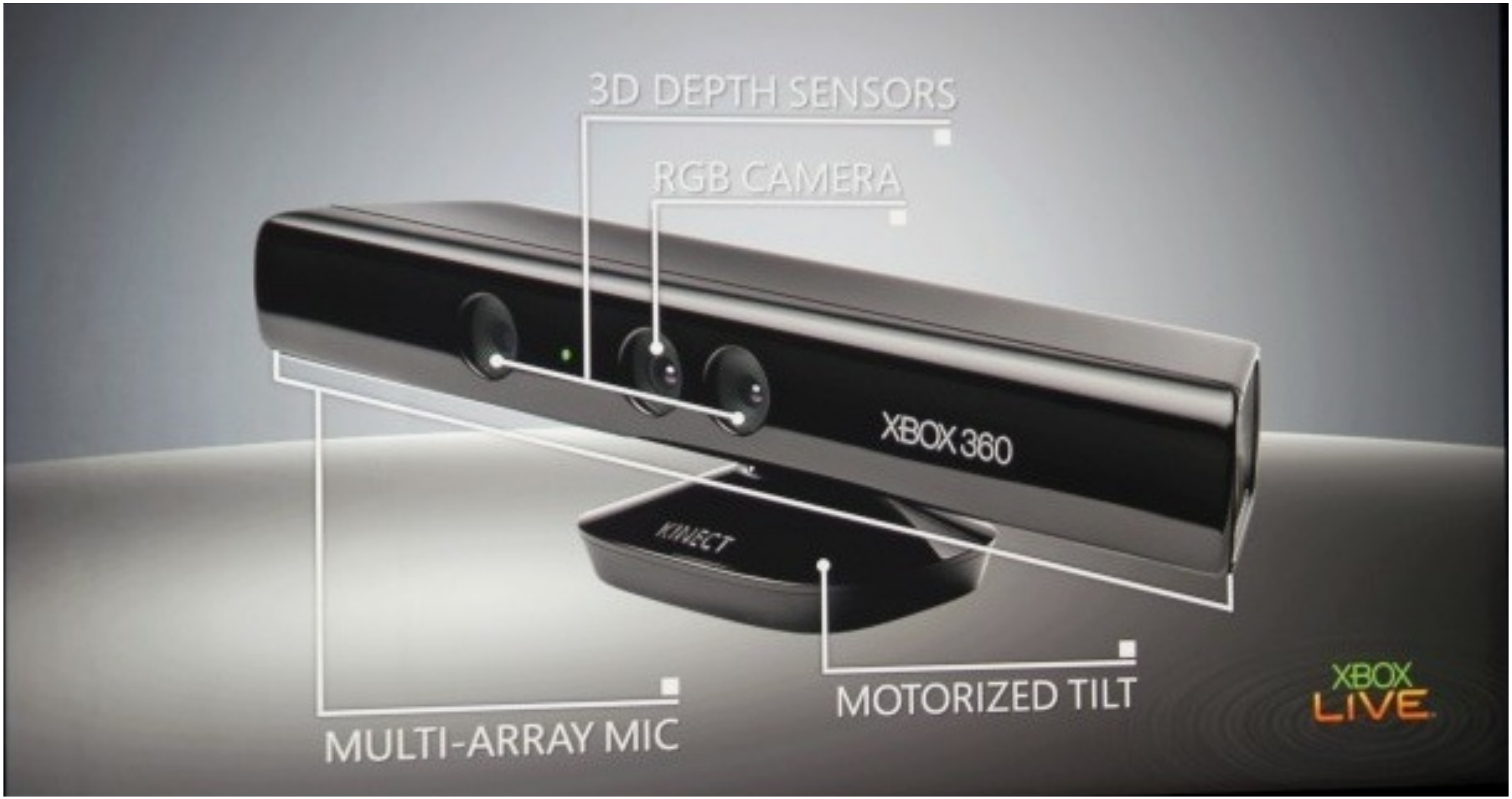
The Digital Michelangelo Project, Levoy et al.

Laser scanned models



The Digital Michelangelo Project, Levoy et al.

Microsoft Kinect



Active stereo with structured light

https://ios.gadgethacks.com/news/watch-iphone-xs-30k-ir-dots-scan-your-face-0180944/