NUS National University of Singapore | School of Computing

CS4277 / CS5477
3D Computer Vision

Lecture 11:
Two-view and Multi-view Stereo

Assoc. Prof. Lee Gim Hee

AY 2022/23

Semester 2

Course Schedule

Week	Date	Topic	Assignments
1	11 Jan	2D and 1D projective geometry	**Assignment 0:** Getting started with Python (Ungraded)
2	18 Jan	3D projective geometry, Circular points and Absolute conic	
3	25 Jan	Rigid body motion and Robust homography estimation	
4	01 Feb	Camera models and calibration	**Assignment 1:** Metric rectification and robust homography (10%) **Due:** 2359hrs, 07 Feb
5	08 Feb	Single view metrology	**Assignment 2**: Affine 3D measurement from vanishing line and point (10%) **Due:** 2359hrs, 14 Feb
6	15 Feb	The Fundamental and Essential matrices	
-	22 Feb	Semester Break	No lecture
7	**01 Mar**	**Mid-term Quiz (20%)** Lecture: Generalized cameras	**In-person Quiz (LT 15, 1900hrs – 2000hrs)** Lecture: 2000hrs – 2130hrs
8	08 Mar	Absolute pose estimation from points or lines	
9	15 Mar	Three-view geometry from points and/or lines	
10	22 Mar	Structure-from-Motion (SfM) and bundle adjustment	**Assignment 3:** SfM and Bundle adjustment (10%) **Due:** 2359hrs, 28 Mar
11	29 Mar	Two-view and multi-view stereo	**Assignment 4:** Dense 3D model from multi-view stereo (10%) **Due:** 2359hrs, 04 Apr
12	05 Apr	3D Point Cloud Processing	
13	12 Apr	Neural Field Representations	

Final Exam: 03 MAY 2023

Learning Outcomes

- Students should be able to:

1. Do stereo rectification and correspondence search along scanlines to get the disparity value of each pixel in two-view stereo.

2. Compute depth values from the disparity map.

3. Explain the concepts of scanline optimization and semi-global matching for two-view stereo.

4. Perform multi-view stereo using the plane sweeping algorithm.

Acknowledgements

- A lot of slides and content of this lecture are adopted from:

1. Svetlana Lazebnik, “Computer Vision Lectures”
   http://slazebni.cs.illinois.edu/spring19/lec18_stereo.pdf
   http://slazebni.cs.illinois.edu/spring19/lec19_multiview_stereo.pdf
2. Sanjay Fidler, “Introduction to Image Understanding”
   http://www.cs.toronto.edu/~fidler/slides/2019/CSC420/lecture12.pdf
   http://www.cs.toronto.edu/~fidler/slides/2018/CSC420/lecture13.pdf
3. D. Gallup et. al, “Real-time Plane-sweeping Stereo with Multiple Sweeping Directions”, CVPR 2007.
4. Yasutaka Furukawa, Carlos Hernndez, “Multi-view stereo: A tutorial”, 2015.

Two-View Stereo

- The goal is to get dense points in 3D from two image pairs with known baseline $(\mathrm{R}, \mathbf{t})$.

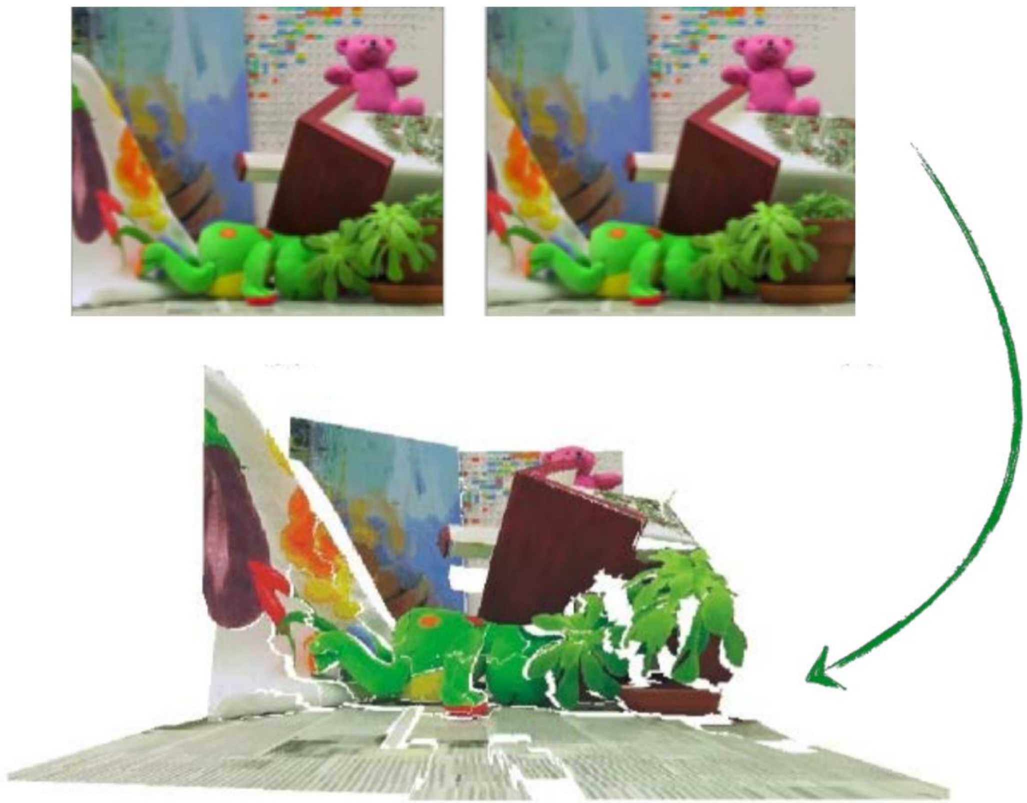


Image source: http://slazebni.cs.illinois.edu/spring19/lec18_stereo.pdf

Human Has Stereo Vision!

- Our pair of eyes gives us the ability to sense depth.

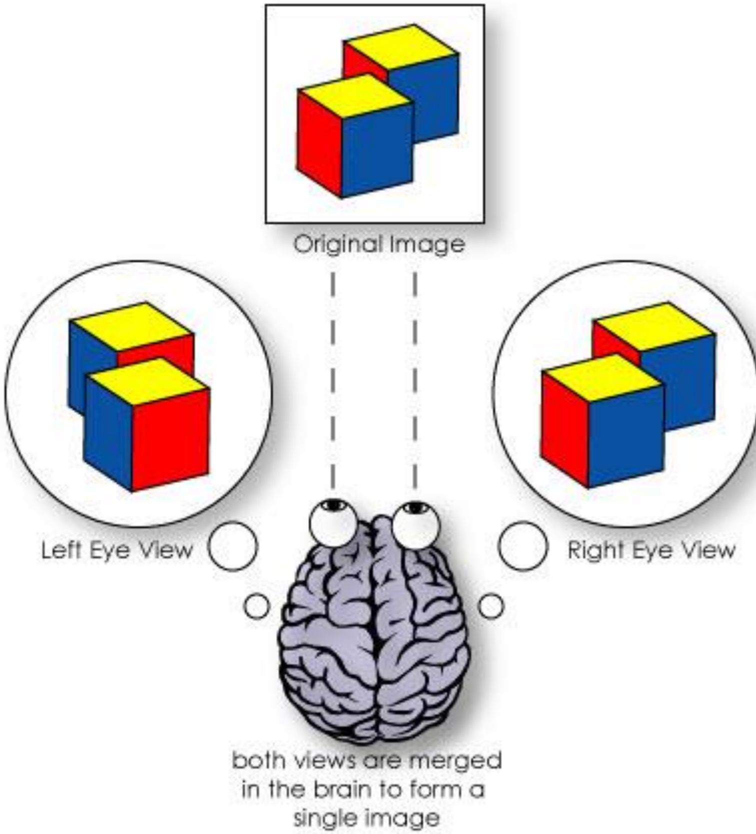
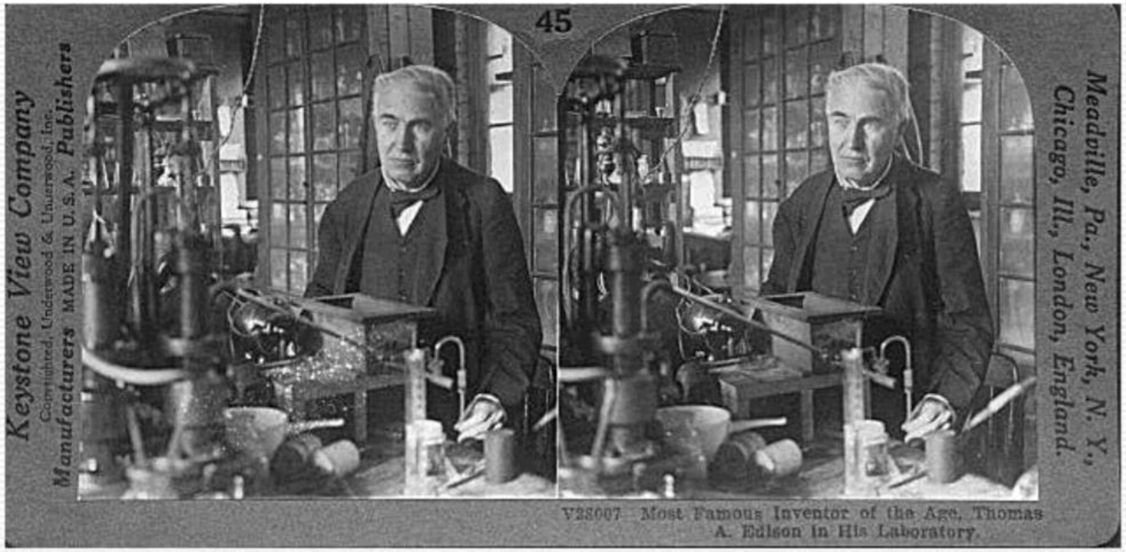


Image source: https://computer.howstuffworks.com/3d-pc-glasses1.htm

Stereograms

- Humans can fuse pairs of images to get a sensation of depth.



Stereograms: Invented by Sir Charles Wheatstone, 1838

Image source: http://slazebni.cs.illinois.edu/spring19/lec18_stereo.pdf

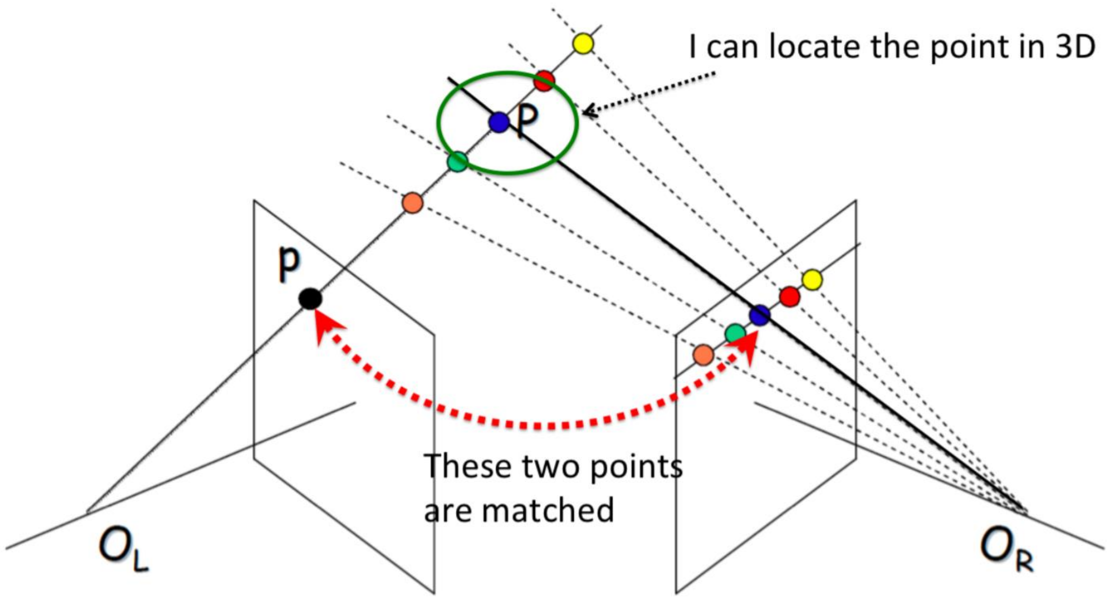
Stereograms

- This is how 3D movies are made!

Image source: http://slazebni.cs.illinois.edu/spring19/lec18_stereo.pdf

Two-View Stereo

- What cues tell us about scene depth?



Epipolar geometry and triangulation from image correspondences give us depth!

Image source: http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf

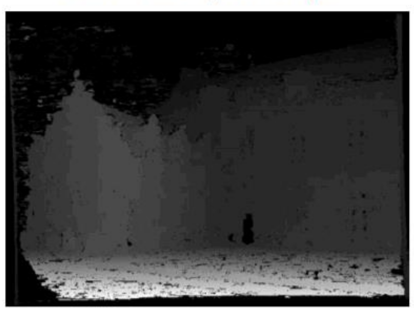
Two-View Stereo: Problem formulation

- **Given:** Two cameras with known baseline $(\mathrm{R}, \mathbf{t})$ and rigidly fixed onto a rig.
- **Find:** The depth map, which gives the dense 3D points of the scene.

image 1

image 2

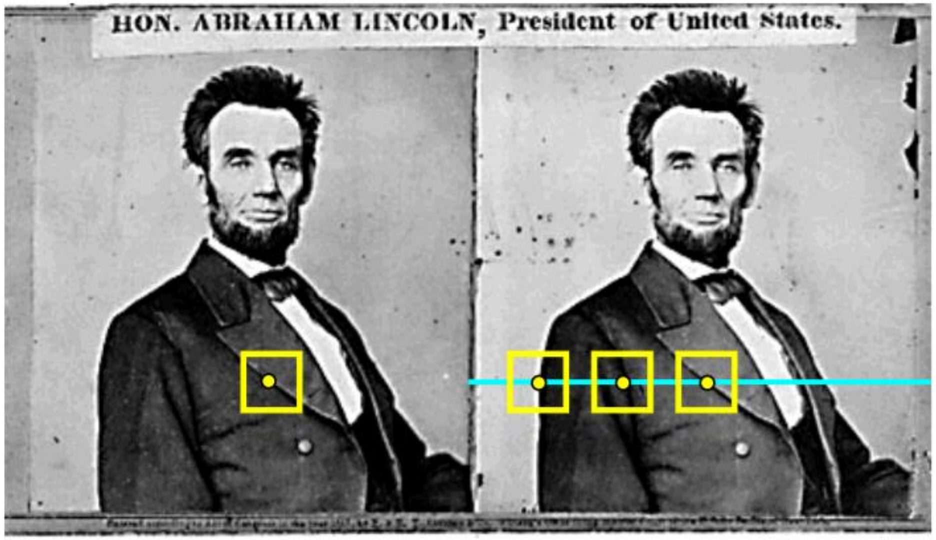
Dense depth map



Bumblebee®2 FireWire

Image source:
https://www.flir.com/support/products/bumblebee2-firewire#Overview

Basic Stereo Matching Algorithm

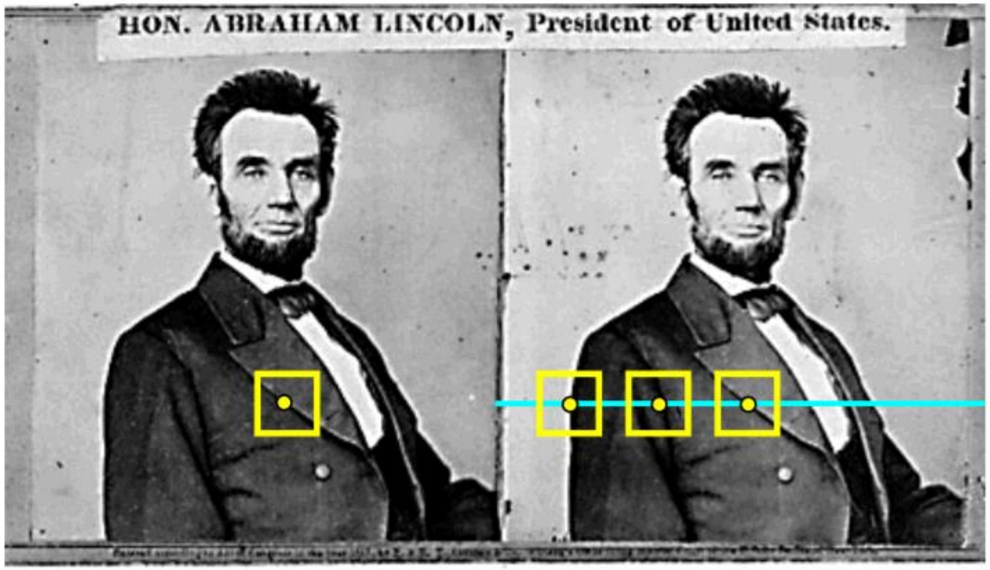


For each pixel in the first image:

1. Find corresponding epipolar line in the right image.
2. Examine all pixels on the epipolar line and pick the best match.
3. Triangulate the matches to get depth information.

Image source: http://slazebni.cs.illinois.edu/spring19/lec18_stereo.pdf

Basic Stereo Matching Algorithm

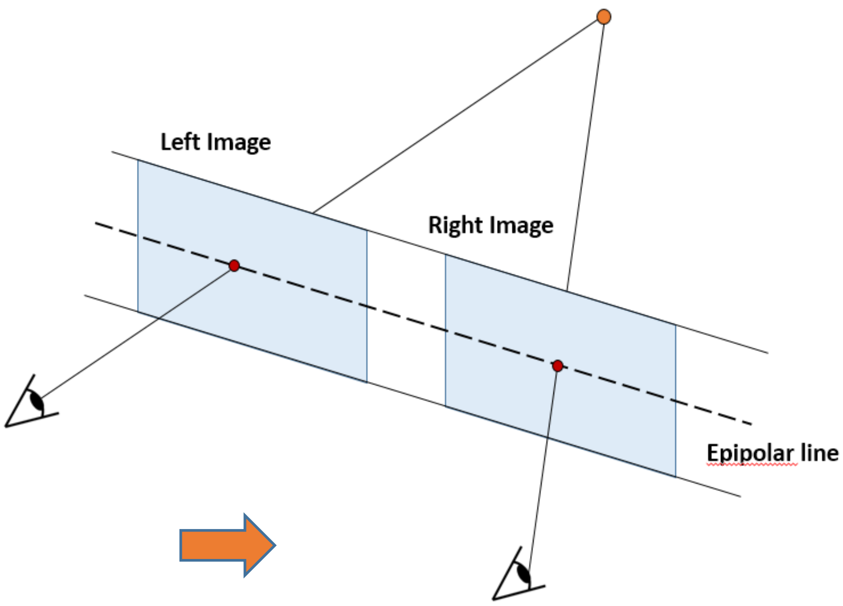


- Simplest case: epipolar lines are corresponding scanlines.
- When does this happen?

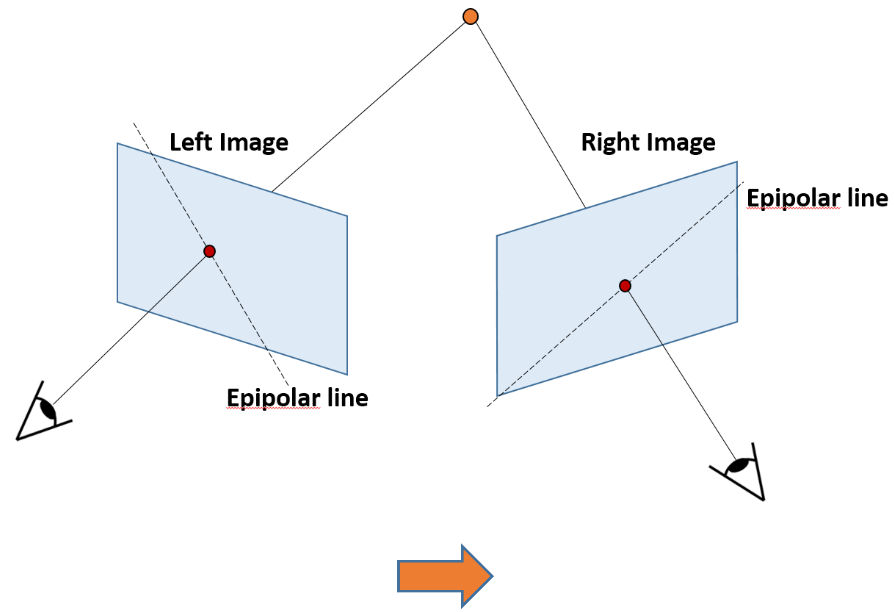
Image source: http://slazebni.cs.illinois.edu/spring19/lec18_stereo.pdf

Two cases of Epipolar Geometry

- Case with two cameras with parallel optical axes
- General case



$\mathrm{R} = \mathrm{I}, \mathbf{t} = (T, 0\ 0)^{\top}$



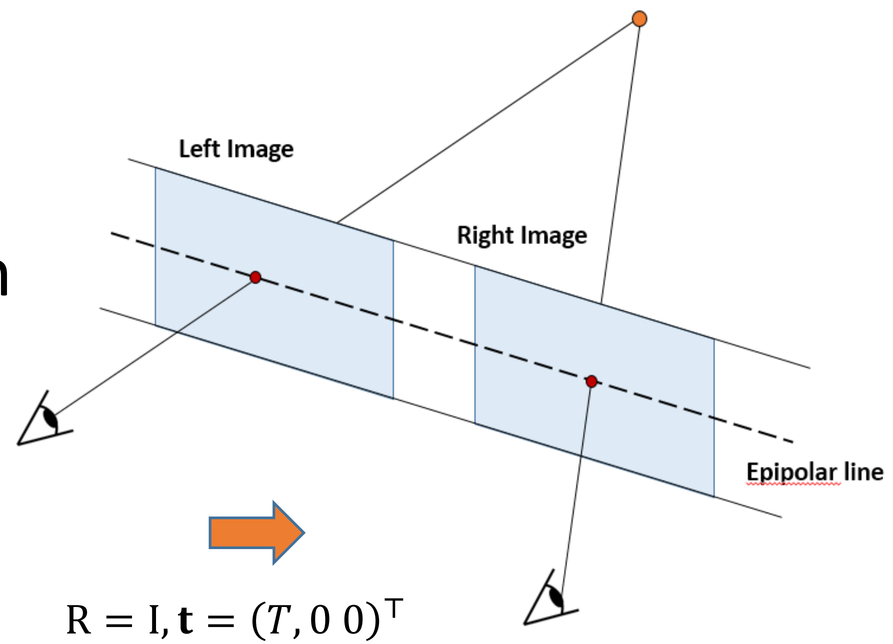
$[\mathrm{R} \mid \mathbf{t}]$

Simplest Case: Parallel Images

- Image planes of cameras are parallel to each other and to the baseline.

- Camera centers are at same height and focal lengths are the same.

- Then epipolar lines fall along the horizontal scan lines of the images.



$\mathrm{R} = \mathrm{I}, \mathbf{t} = (T, 0\ 0)^{\top}$

Essential Matrix for Parallel Images

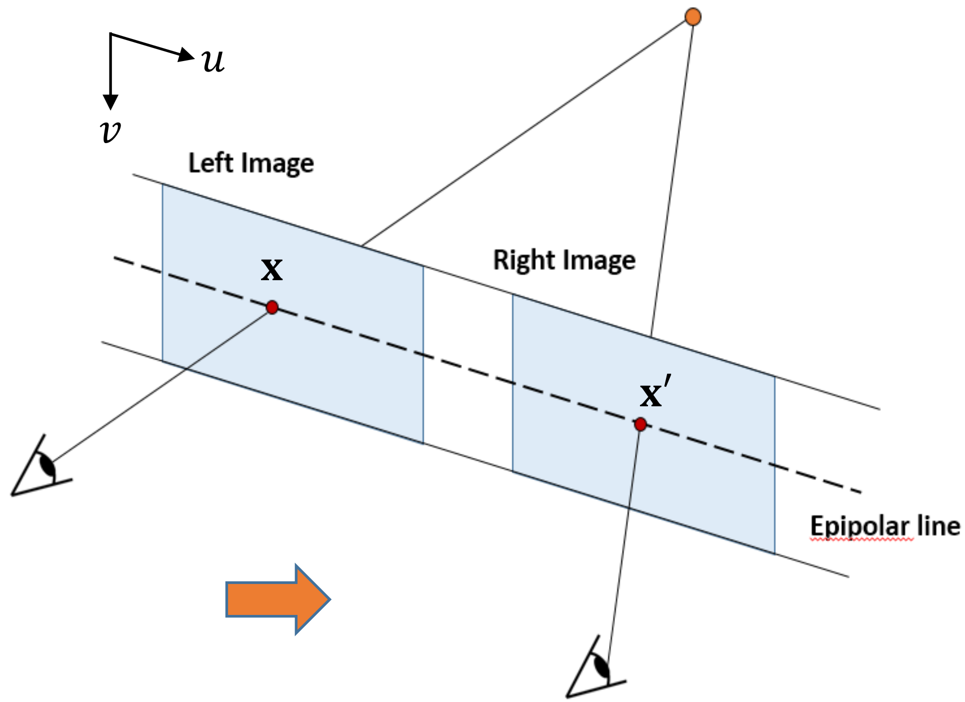
Epipolar constraint:

$$\mathbf{x}'^{\top}\mathrm{E}\mathbf{x} = 0, \qquad \mathrm{E} = [\mathbf{t}_{\times}]\mathrm{R}.$$

Since

$$\mathrm{R} = \mathrm{I} \text{ and } \mathbf{t} = (T, 0\ 0)^{\top},$$

$$\Rightarrow \mathrm{E} = [\mathbf{t}_{\times}]\mathrm{R} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}.$$



Now we have:

$$(u' \quad v' \quad 1)\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = 0 \;\Rightarrow\; (u' \quad v' \quad 1)\begin{pmatrix} 0 \\ -T \\ Tv \end{pmatrix} = 0 \;\Rightarrow Tv' = Tv.$$

The y-coordinates of corresponding points are the same!

Simplest Case: Parallel Images

- So, all points on the projective line span by the left camera center $\mathbf{C}$ and image point $\mathbf{x}$ project to a horizontal line with $v' = v$ on the right image.

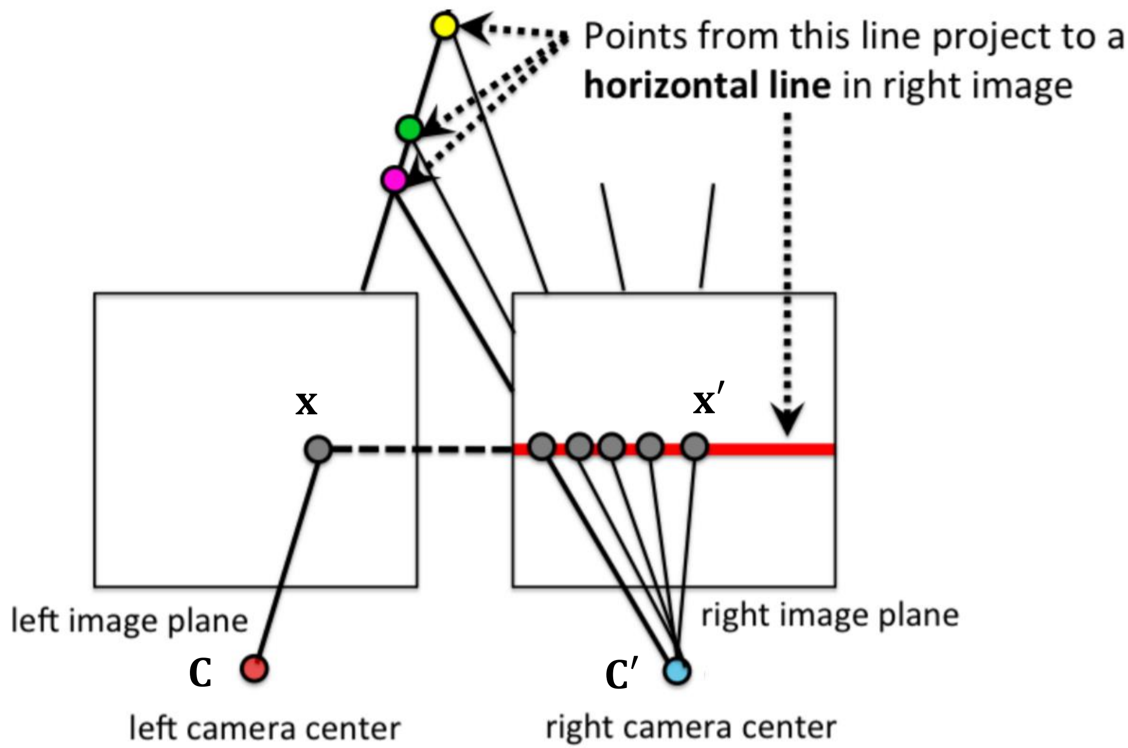


Image source: http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf

Simplest Case: Parallel Images

- **Another observation:** No point from the projective line of the left image can project to right of $\mathbf{x}'$ on the right image.

- That would mean our image can see behind the camera (we will see the detail later).

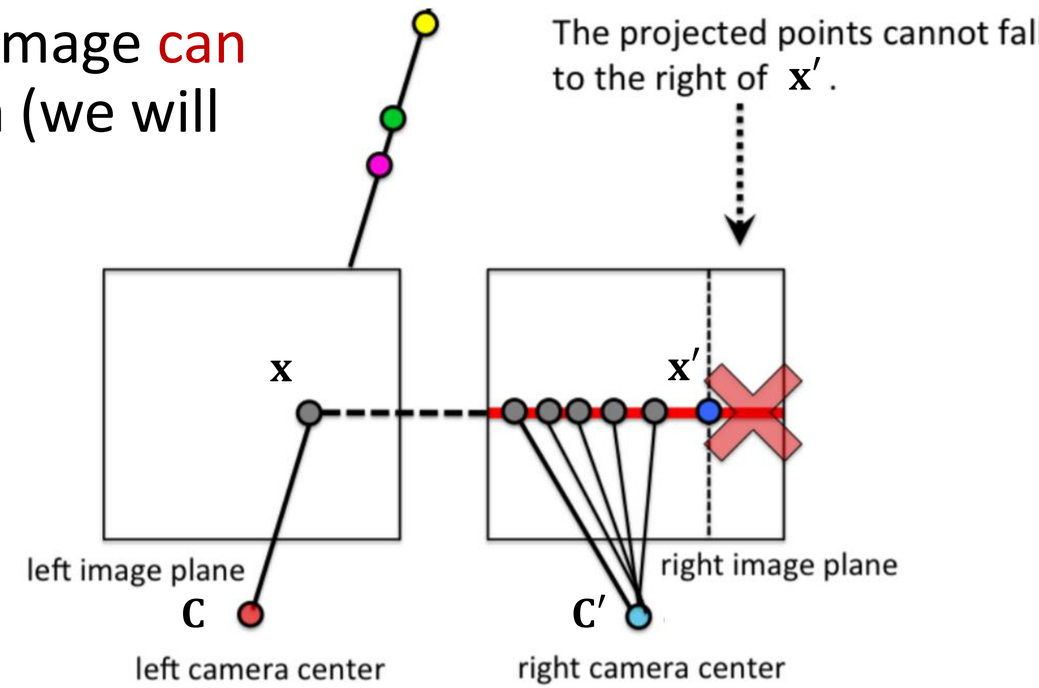


Image source: http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf

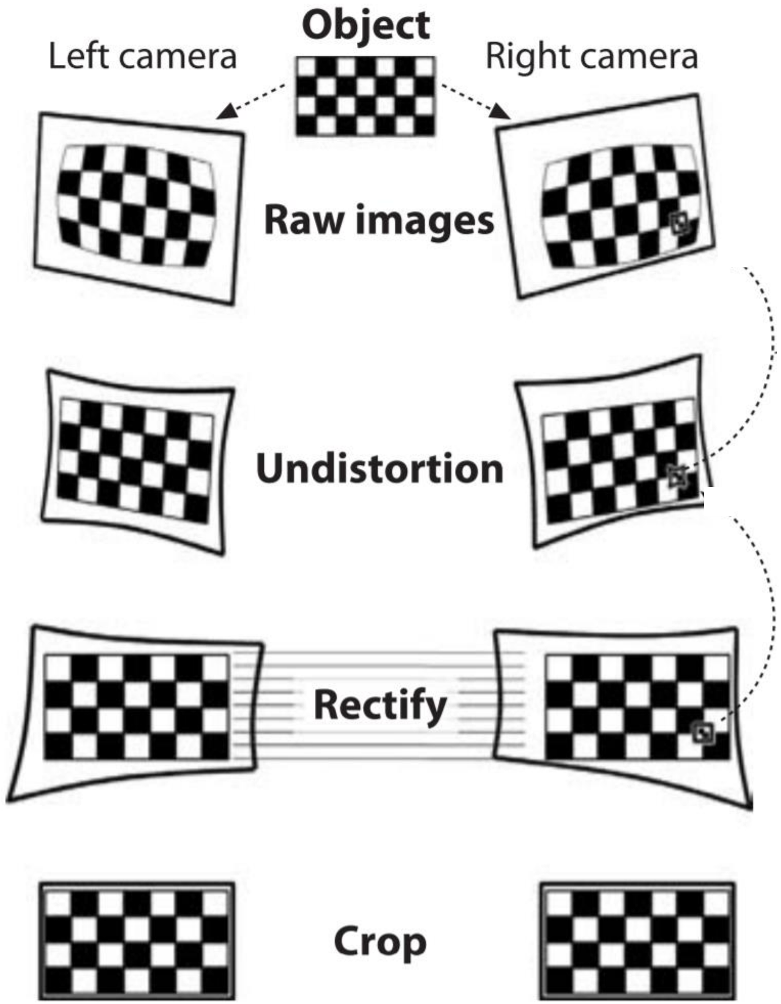
General Case: Non-Parallel Images

This is a two-step process:

1. **Stereo calibration**:
   a. Use the checkerboard pattern to find the intrinsics parameters K and K', and distortion parameters of each camera.
   b. Undistort images and compute the essential matrix E and then decompose to get the relative pose $(\mathrm{R}, \mathbf{t})$ of the cameras.

2. **Stereo rectification**:
   a. Correct the individual images so that they appear as if they had been taken by two cameras with row-aligned image planes.

General Case: Non-Parallel Images



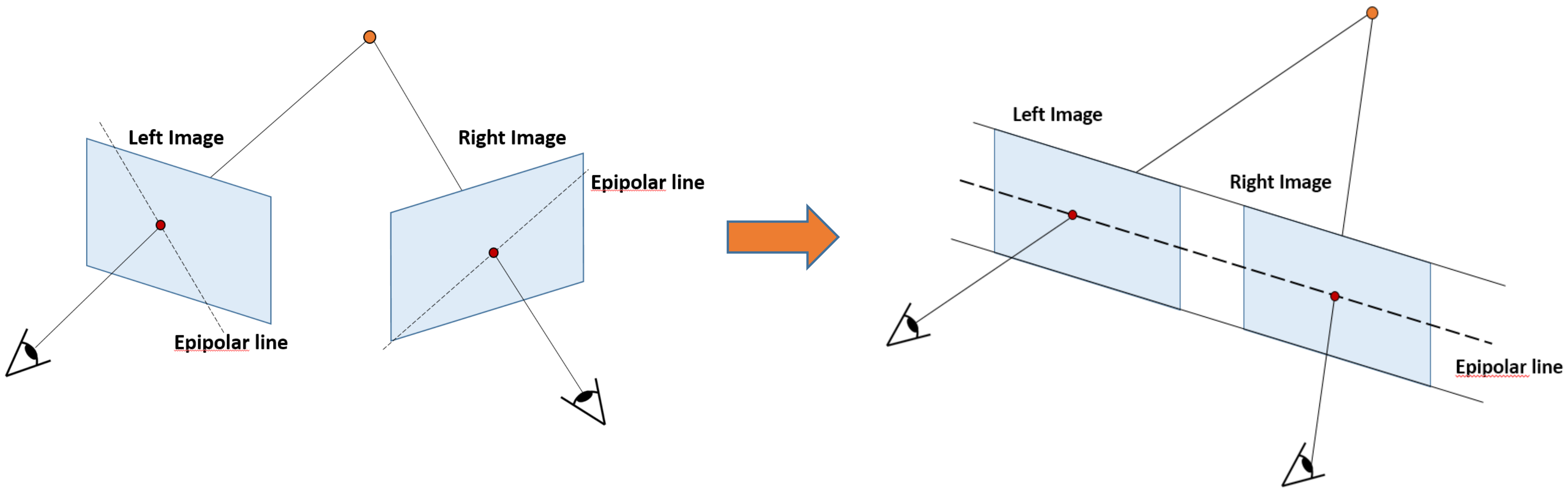
Stereo calibration

Stereo rectification

Source: Gary Bradski and Adrian Kaehler, “Learning OpenCV”, 2008

Stereo Rectification

- Our goal is to mathematically align the two cameras into one viewing plane so that pixel rows between the cameras are exactly aligned with each other.



Stereo Rectification

Stereo Rectification

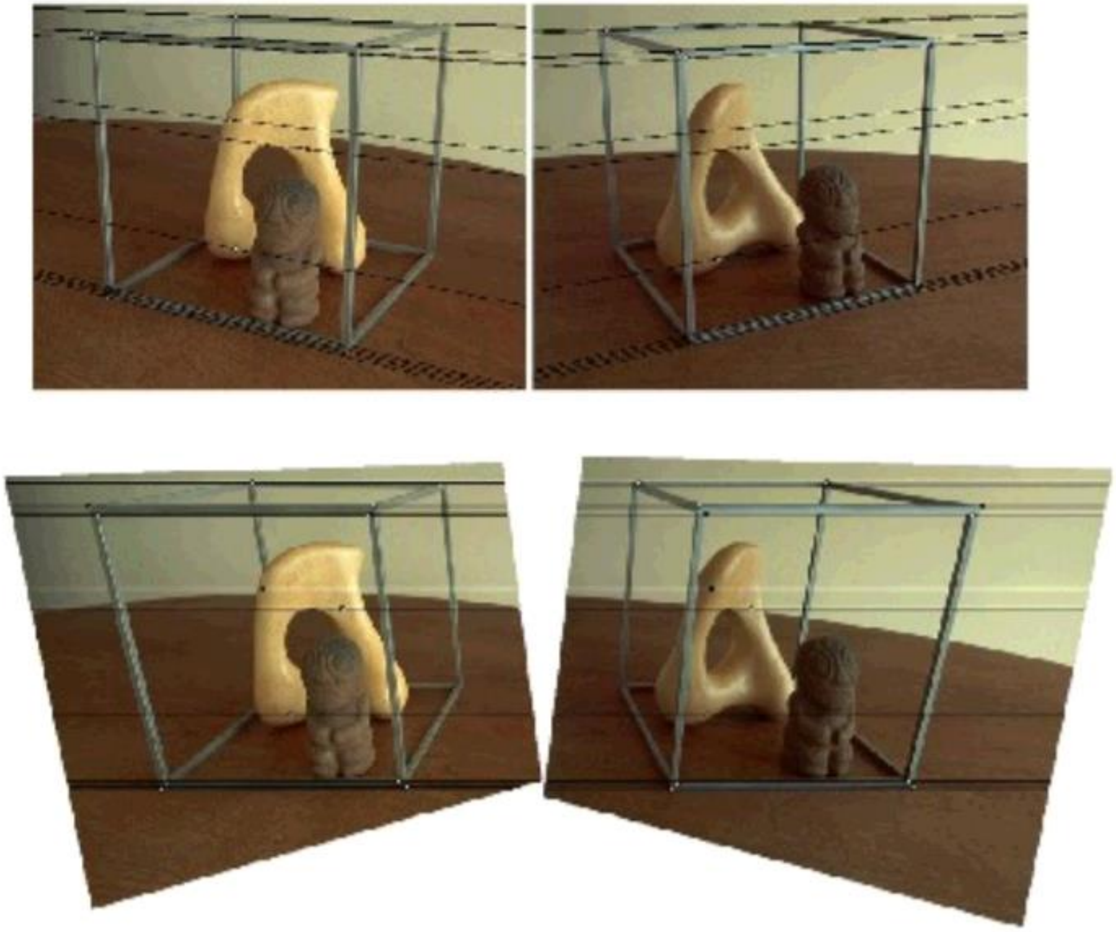


Image Source: J. Hays

Stereo Rectification

- All rectified images satisfy the following two properties:

1. All epipolar lines are parallel to the horizontal axis.
2. Corresponding points have identical vertical coordinates.

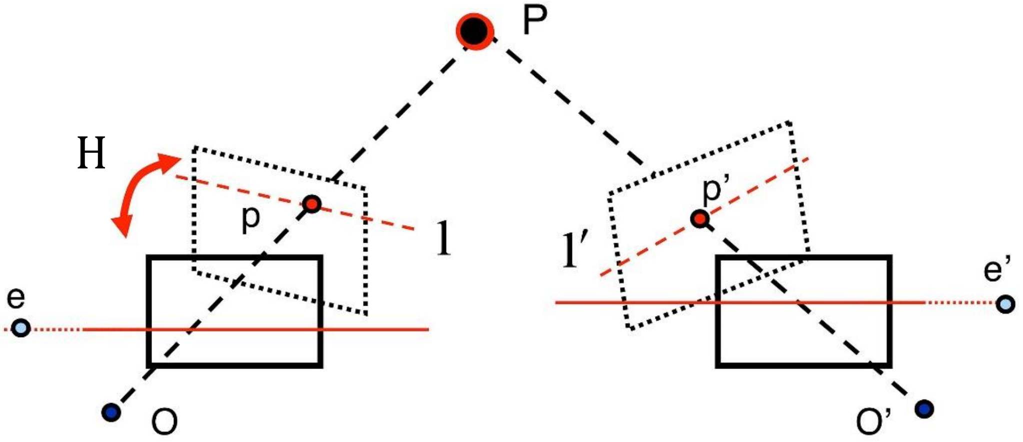


Image source: https://en.wikipedia.org/wiki/Image_rectification

Stereo Rectification

- **Goal:** Find a projective transformation H such that the epipoles $\boldsymbol{e}$ and $\boldsymbol{e}'$ in the two images are mapped to the infinite point $[1,0,0]^{\mathsf{T}}$.

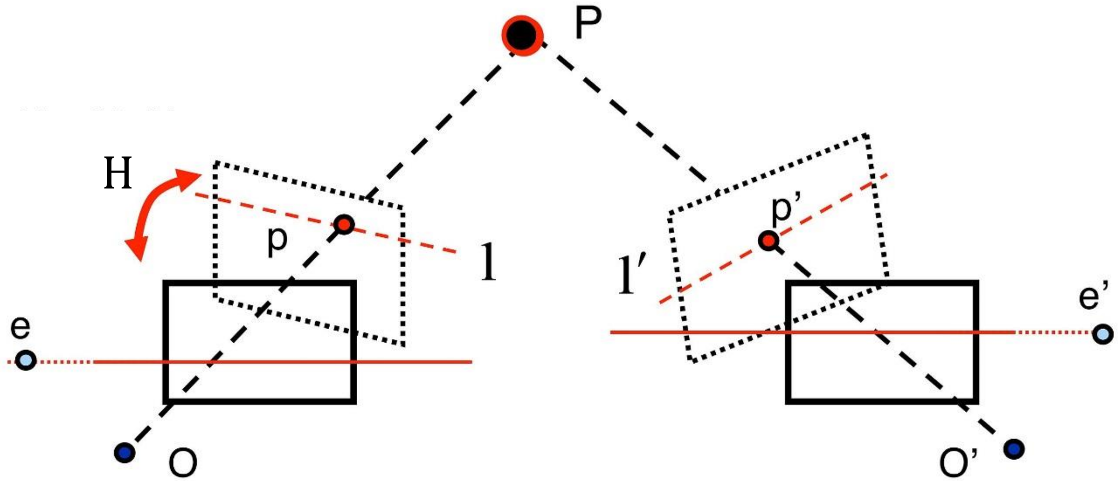


Image source: https://en.wikipedia.org/wiki/Image_rectification

Stereo Rectification

- Our derivation is based on a pair of images that observe a 3D point $\mathbf{P}$, which corresponds to $\mathbf{p}$ and $\mathbf{p}'$ in the pixel coordinates of each image.

- We further let $\mathbf{O}$ and $\mathbf{O}'$ represent the optical centers of each camera, with known camera matrices $\mathrm{M} = \mathrm{K}[\mathrm{I} \mid 0]$ and $\mathrm{M}' = \mathrm{K}[\mathrm{R} \mid \mathbf{t}]$.

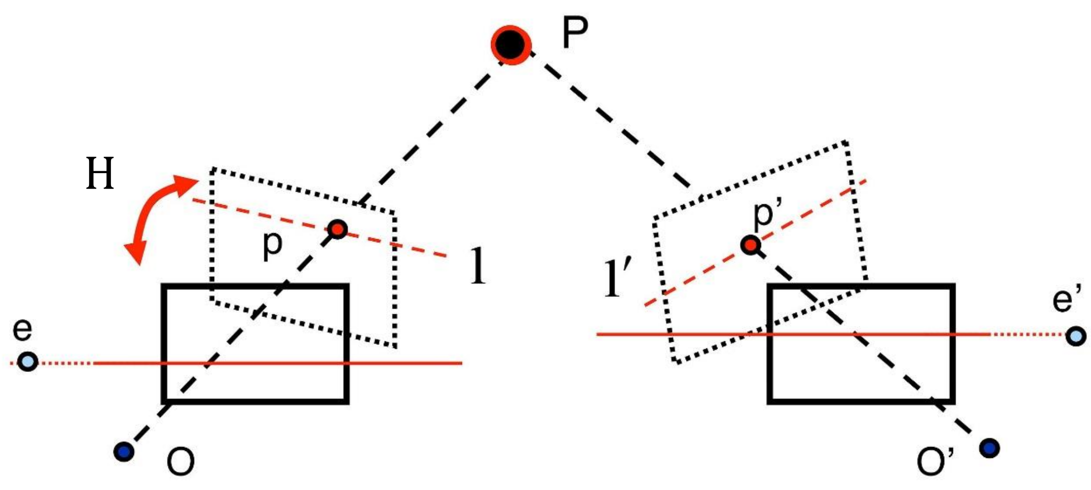


Image source: https://en.wikipedia.org/wiki/Image_rectification

Stereo Rectification

1. We compute the camera normalized epipoles, $\hat{\boldsymbol{e}}$ and $\hat{\boldsymbol{e}}'$ in each image:

$$\boldsymbol{e} = \mathrm{M}\begin{bmatrix}\mathbf{O}'\\1\end{bmatrix} = \mathrm{M}\begin{bmatrix}-\mathrm{R}^\top\mathbf{t}\\1\end{bmatrix} = \mathrm{K}[\,\mathrm{I}\mid 0\,]\begin{bmatrix}-\mathrm{R}^\top\mathbf{t}\\1\end{bmatrix} = -\mathrm{K}\mathrm{R}^\top\mathbf{t} = \mathrm{K}\mathbf{O}' \Rightarrow \hat{\boldsymbol{e}} = \mathbf{O}',$$

$$\boldsymbol{e}' = \mathrm{M}'\begin{bmatrix}\mathbf{O}\\1\end{bmatrix} = \mathrm{M}'\begin{bmatrix}\mathbf{0}\\1\end{bmatrix} = \mathrm{K}'[\,\mathrm{R}\mid\mathbf{t}\,]\begin{bmatrix}\mathbf{0}\\1\end{bmatrix} = \mathrm{K}'\mathbf{t} \Rightarrow \hat{\boldsymbol{e}}' = \mathbf{t}.$$

2. Compute the projective transformation H that maps $\hat{\boldsymbol{e}}$ to $[1,0,0]^\top$, a good choice is the rotation matrix:

$$\mathrm{H} = \mathrm{R}_{\mathrm{rect}} = \begin{bmatrix}\mathbf{R}_1^\top\\\mathbf{R}_2^\top\\\mathbf{R}_3^\top\end{bmatrix}, \text{where} \quad \mathbf{R}_1 = \frac{\mathbf{O}'}{\|\mathbf{O}'\|}, \quad \mathbf{R}_2 = \frac{[-o'_y, o'_x, 0]^\top}{\sqrt{{o'_x}^2 + {o'_y}^2}}, \quad \mathbf{R}_3 = \mathbf{R}_1 \times \mathbf{R}_2.$$

Stereo Rectification

3. Next, we find a projective transformation H' that maps $\hat{\boldsymbol{e}}'$ to $[1,0,0]^\top$, a good choice is: $\mathrm{H}' = \mathrm{HR}^\top$.

   Exercise: verify that this is true!

4. Finally, we apply H on the left image and H' on the right image to get the rectified pair.

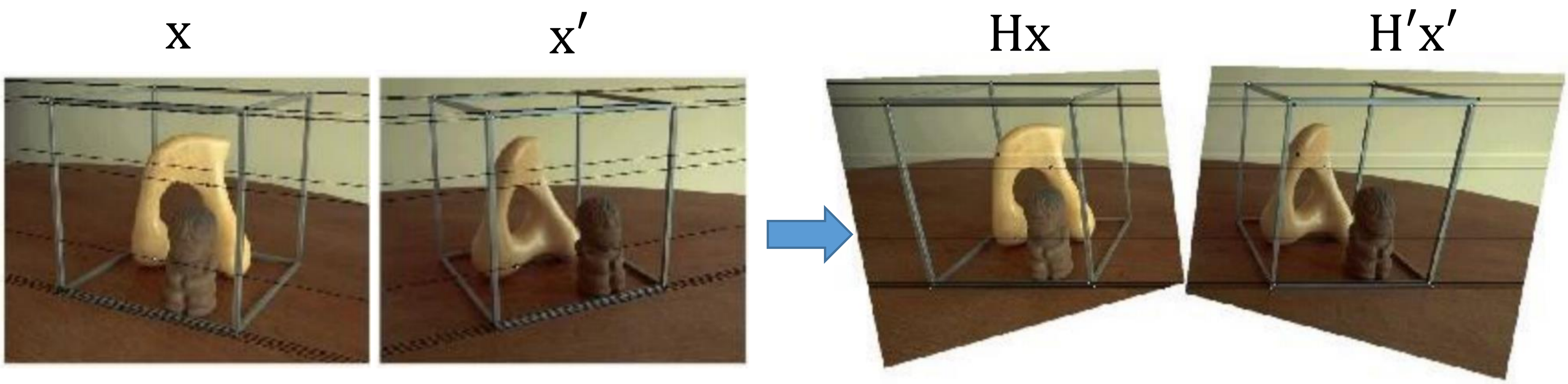


Image Source: J. Hays

Correspondence Search

- Slide a window along the right scanline and compare contents of that window with the reference window in the left image.

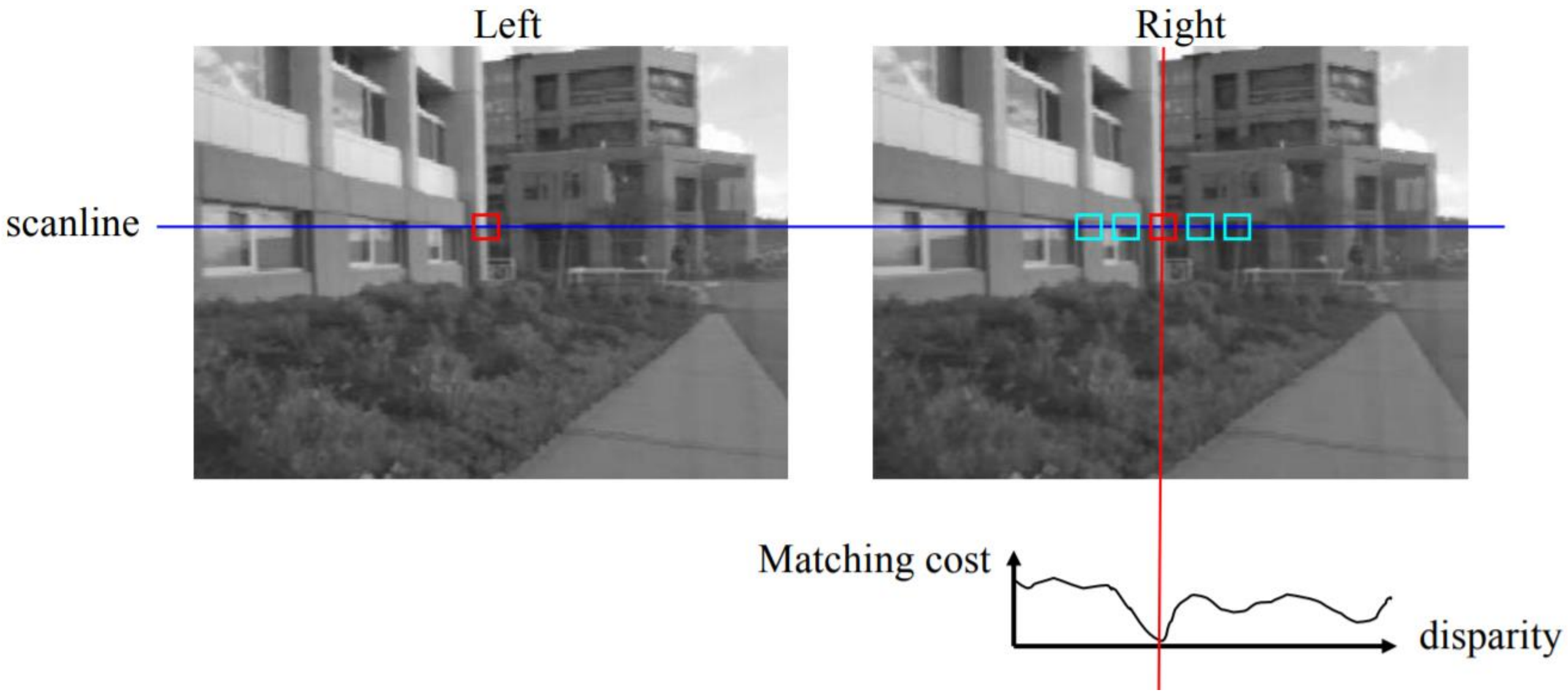


Image source: http://slazebni.cs.illinois.edu/spring19/lec18_stereo.pdf

Correspondence Search

- Several methods are used to compute the matching /photo-consistency cost:

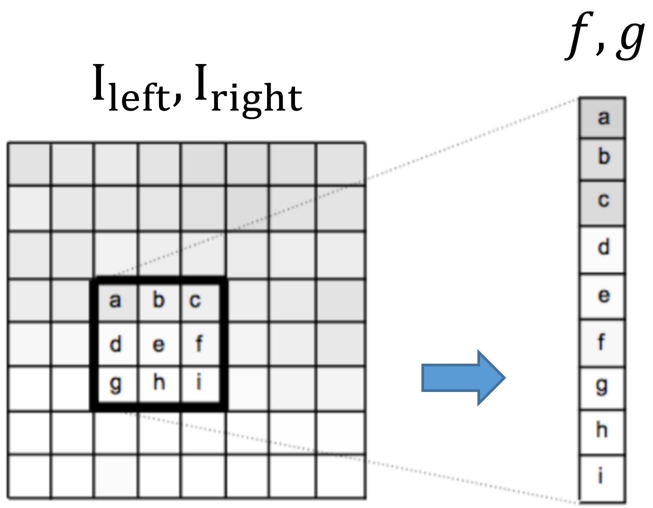
1. Normalized cross correlation
2. Sum-of-squared differences
3. Sum-of-absolute differences
4. Mutual information

Normalized Cross Correlation

- Zero-mean normalized cross correlation (NCC) is defined as:

$$\rho_{NCC}(f,g) = \frac{(f-\bar{f})\cdot(g-\bar{g})}{\sigma_f \sigma_g} \in [-1,1],$$

where



- $\bar{f}, \bar{g}$: means of patch intensities
- σ_f, σ_g: standard deviations of $\bar{f}, \bar{g}$.

Vectorization of 3×3 patch Ω around $\mathbf{x}$ and $\mathbf{x}'$.

Normalized Cross Correlation

- NCC is invariant to changes in gain and bias and it is mainly used when lighting and material invariance is required (this is good for multi-view stereo).

- The main failure modes of NCC are a lack of surface texture and repetitive textures, while the main advantage is its accuracy.

Sum-of-Squared Differences

- The sum-of-squared differences (SSD) is defined as the L2 squared distance between vectors f and g

$$\rho_{SSD}(f,g) = ||f-g||^2.$$

- It is usually mapped through an exponential to the $[0,1]$ range for normalization purposes, i.e.

$$\rho_{SSD}(f,g) = e^{-\frac{||f-g||^2}{\sigma^2}} \in [0,1].$$

- σ is an additive Gaussian noise with standard deviation.

Sum-of-Squared Differences

- The use of the L2 norm makes SSD sensitive to outliers, e.g. visibility outliers or bias and gain perturbations.

- A normalized variant of SSD exists that helps mitigate some of these issues:

$$\rho_{NSSD}(f,g) = \left|\left|\frac{f-\bar{f}}{\sigma_f} - \frac{g-\bar{g}}{\sigma_g}\right|\right|^2,$$

- Where $\bar{f}$ is the mean of f and σ_f is the standard deviation of f.

Sum-of-Squared Differences

- This version is equivalent to NCC:

$$
\begin{aligned}
\rho_{NSSD}(f,g) &= \left|\left|\frac{f-\bar{f}}{\sigma_f} - \frac{g-\bar{g}}{\sigma_g}\right|\right|^2 \\
&= \left|\left|\frac{f-\bar{f}}{\sigma_f}\right|\right|^2 + \left|\left|\frac{g-\bar{g}}{\sigma_g}\right|\right|^2 - 2\frac{f-\bar{f}}{\sigma_f}\cdot\frac{g-\bar{g}}{\sigma_g} \\
&= 2\left(1 - \frac{(f-\bar{f})\cdot(g-\bar{g})}{\sigma_f\sigma_g}\right) \\
&= 2(1 - NCC(f,g))
\end{aligned}
$$

Sum-of-Absolute Differences

- The sum-of-absolute differences (SAD) is very similar to SSD, but uses an L1 norm instead of an L2 norm, which makes it more robust to outliers:

$$\rho_{SAD}(f,g) = ||f - g||_1.$$

- Similarly to SSD it is sensitive to bias and gain, so it is rarely used in algorithms that match images with a wide variability in illumination.

- It is however a very good measure for applications that can guarantee similar capture conditions for the different images

Mutual Information

- In information theory, the mutual information of two random variables X and Y is a measure of how dependent the two variables are:

$$MI(X,Y) = \sum_{x \in X, y \in Y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)},$$

- where $P(x,y)$ is the joint probability of X and Y, and $P(x)$ and $P(y)$ are the marginals.

Mutual Information

- The mutual information between two image patches measures how similar the two patches are, i.e., how well one patch predicts the other.

- The photo-consistency measure is defined as:

$$\rho_{MI}(f, g) = -MI(f, g).$$

Mutual Information

- The joint probability is estimated using a Parzen window method

$$P(x, y) = \frac{1}{|\Omega|} \sum_{q \in \Omega} K(f(q) - x, g(q) - y),$$

- $K(\cdot,\cdot)$ is the particular 2d kernel being used, typically a Gaussian.

- The range of x, y is the range of the images being matched, for instance, $[0, 255]$ for gray scale images.

Mutual Information

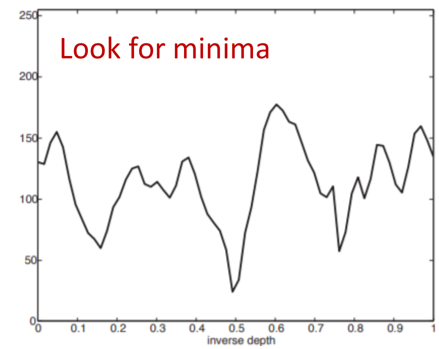
- $P(x)$ and $P(y)$ are obtained by marginalizing the joint probability.

- Note that, compared to other similarity measures, MI usually requires a large domain Ω so that $P(x, y)$ properly models the joint distribution.

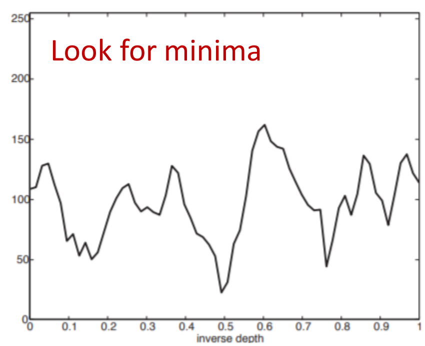
Example of Matching Cost

- Two calibrated images used as input to matching a single pixel (center of the left image) against a second image across its epipolar line.

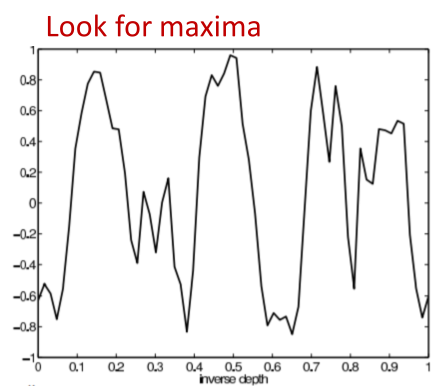
SSD



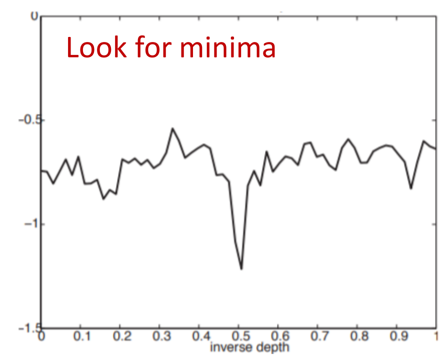
SAD



NCC



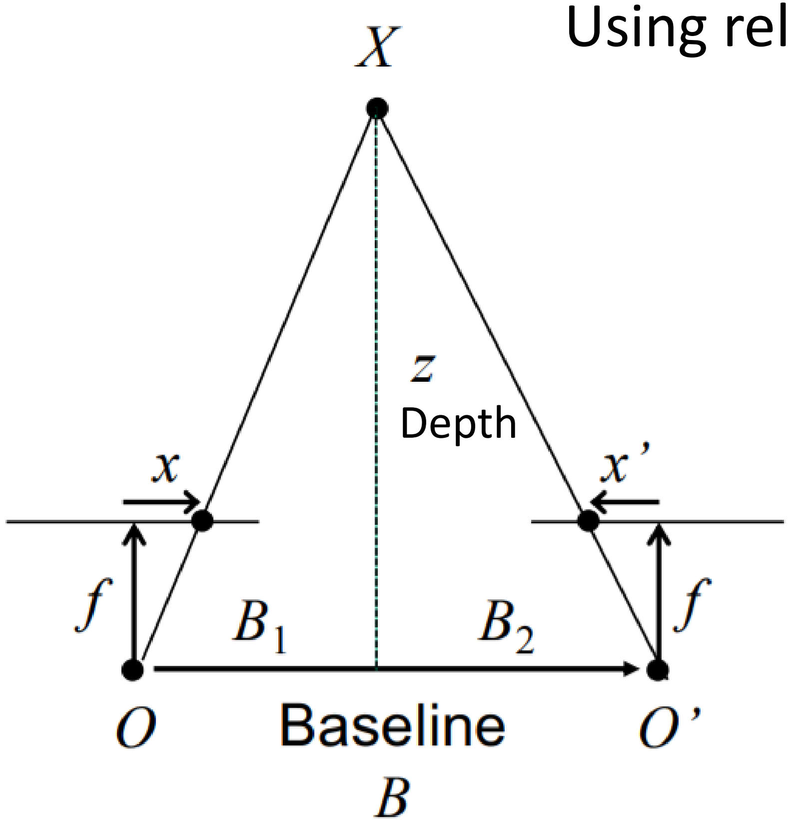
MI



Correct depth is roughly at 0.5 depth

Image source: Y. Furukawa, “Multi-view Stereo: A Tutorial”, 2015

Depth from Disparity



Using relation from similar triangles, we get

$$\frac{x}{f} = \frac{B_1}{z} \quad \text{and} \quad \frac{-x'}{f} = \frac{B_2}{z},$$

$$\Rightarrow \frac{x - x'}{f} = \frac{B_1 + B_2}{z},$$

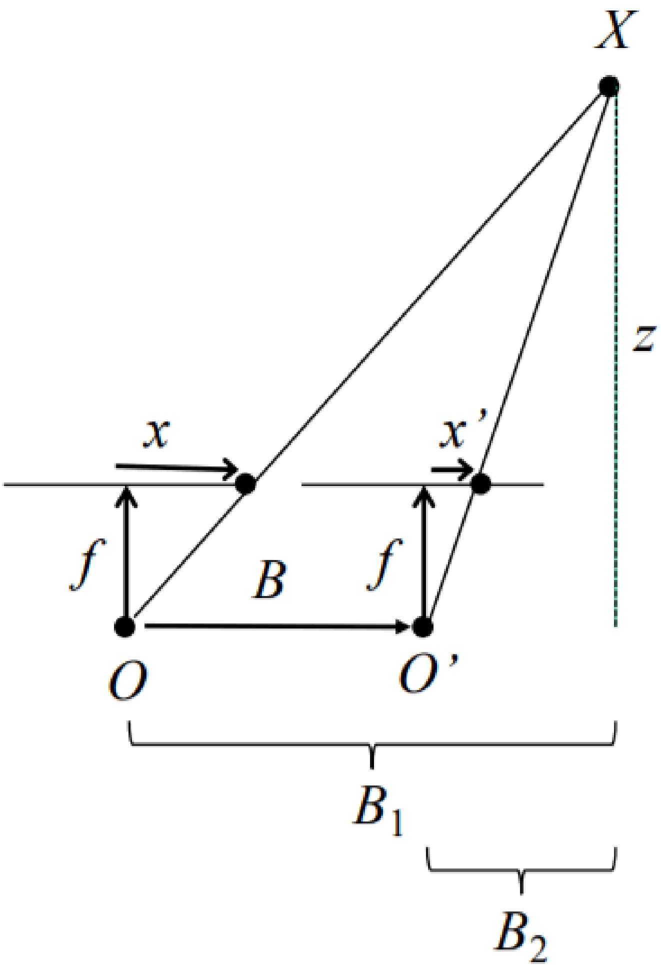
$$\Rightarrow disparity = x - x' = \frac{B \cdot f}{z},$$

$$\Rightarrow depth = z = \frac{B.f}{x - x'}.$$

Disparity is inversely proportional to depth!

Image source: http://slazebni.cs.illinois.edu/spring19/lec18_stereo.pdf

Depth from Disparity



Similarly,

$$\frac{x}{f} = \frac{B_1}{z} \quad \text{and} \quad \frac{x'}{f} = \frac{B_2}{z},$$

$$\Rightarrow \frac{x - x'}{f} = \frac{B_1 - B_2}{z},$$

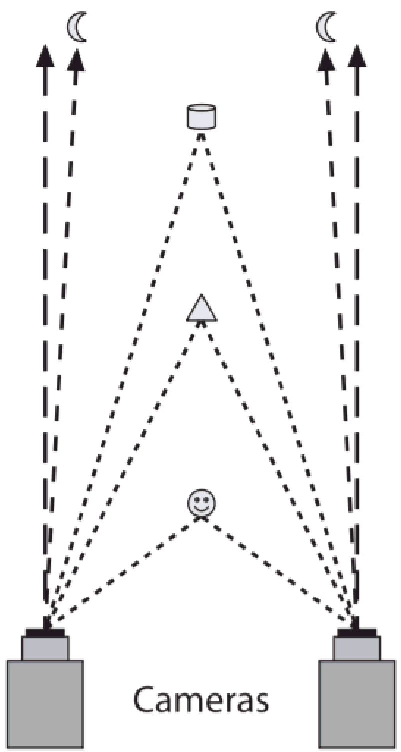
$$\Rightarrow disparity = x - x' = \frac{B \cdot f}{z},$$

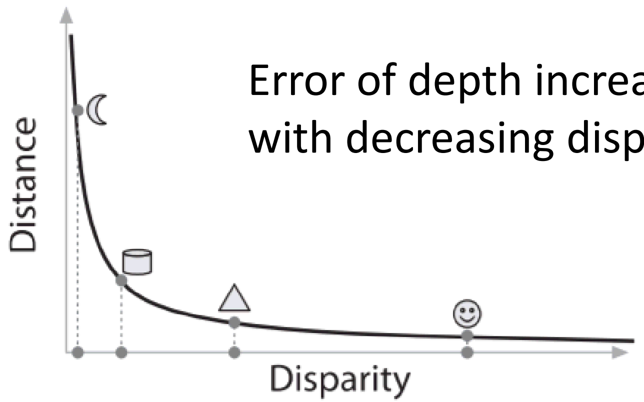
$$\Rightarrow \ depth = z = \frac{B.f}{x - x'}.$$

Image source: http://slazebni.cs.illinois.edu/spring19/lec18_stereo.pdf

Depth from Disparity

- Depth and disparity are inversely related, so fine-grain depth measurement is restricted to nearby objects.





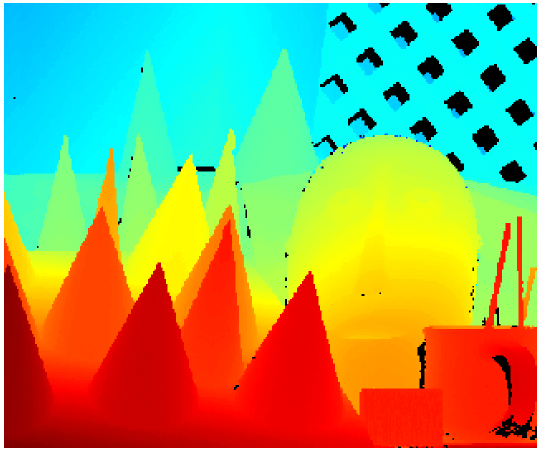
Error of depth increases with decreasing disparity!

Source: Gary Bradski and Adrian Kaehler, “Learning OpenCV”, 2008

Block Matching

- Naïve search along the epipolar line is also known as block matching, which results in "blocky" depth maps.
- This is because each pixel is considered independently.

Image (left)



Ground truth depthmap

3x3 block matching

Image source: N. Einecke, "A multi-block-matching approach for stereo" IV 2015.

Effect of window size

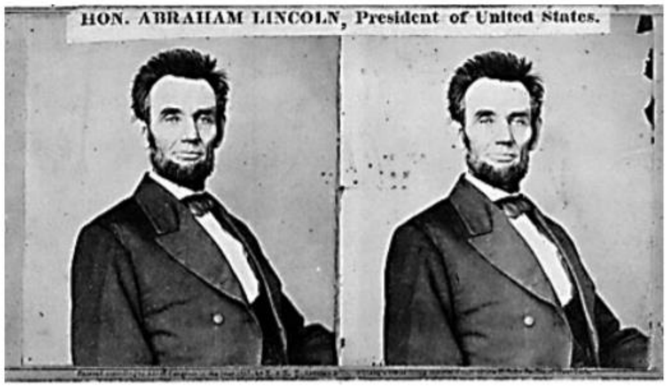
- **Smaller window:** + More detail, – More noise.
- **Larger window:** + Smoother disparity maps, – Less detail.

W = 3

W = 20

Image source: http://slazebni.cs.illinois.edu/spring19/lec18_stereo.pdf

Failures of Correspondence Search



Textureless surfaces

Occlusions, repetition

Non-Lambertian surfaces, specularities

Image source: http://slazebni.cs.illinois.edu/spring19/lec18_stereo.pdf

Scanline Optimization Stereo

- Pixelwise cost calculation is generally ambiguous and wrong matches can easily have a lower cost than correct ones, due to noise.
- Therefore, an additional constraint is added that supports smoothness by penalizing changes of neighboring disparities.
- Scanline optimization adds smoothness constraint along each scanline.

Scanline Optimization Stereo

- The disparity $x - x'$ is a discrete set of numbers since it is the difference between the x index of a two image pixels.

- Therefore, we can cast stereo as a labeling problem, i.e. for every pixel p, find an assignment of a label $d_p \in \{1, \ldots, L\}$.

- We consider all disparities within a bound, i.e. for a pixel with coordinates (x, y), the set of pixels in other image is in the range of $\{(\tilde{x}, y) \mid x - \tilde{x} < \alpha\}$.

Scanline Optimization Stereo

- This can be achieved by minimizing the following cost function:

$$E(d) = \sum_p D(p, d_p) + \sum_{q \in \mathcal{N}(p)} R(d_p, d_q),$$

- $D(p, d_p)$ is the pixel-wise dissimilarity cost (i.e. matching cost) at pixel p with disparity d_p.

- We first consider a simple version where p and q are neighbor on the same scanline.

Scanline Optimization Stereo

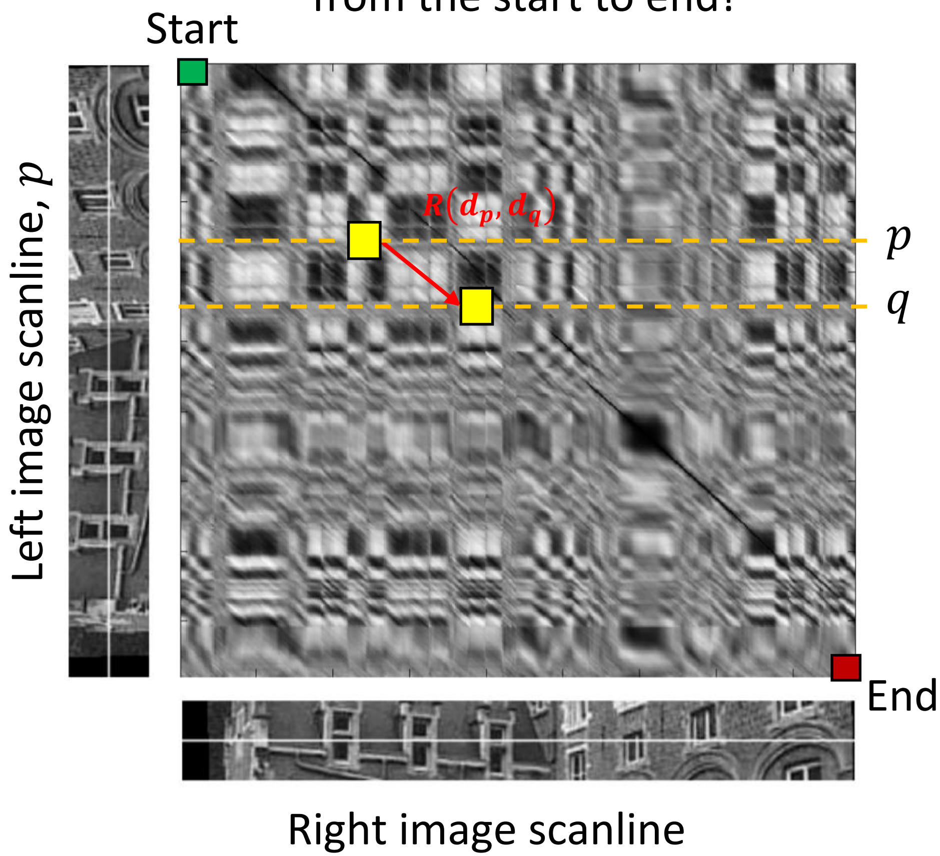
- $R(d_p, d_q)$ is the regularization cost term to ensure smoothness of the disparity values in a neighborhood, i.e.

$$R(d_p, d_q) = \begin{cases} 0 & d_p = d_q \\ P_1 & |d_p - d_q| = 1 \\ P_2 & |d_p - d_q| > 1 \end{cases}$$

- P_1 and P_2 are two constant parameters, with $P_1 < P_2$.

Scanline Optimization Stereo

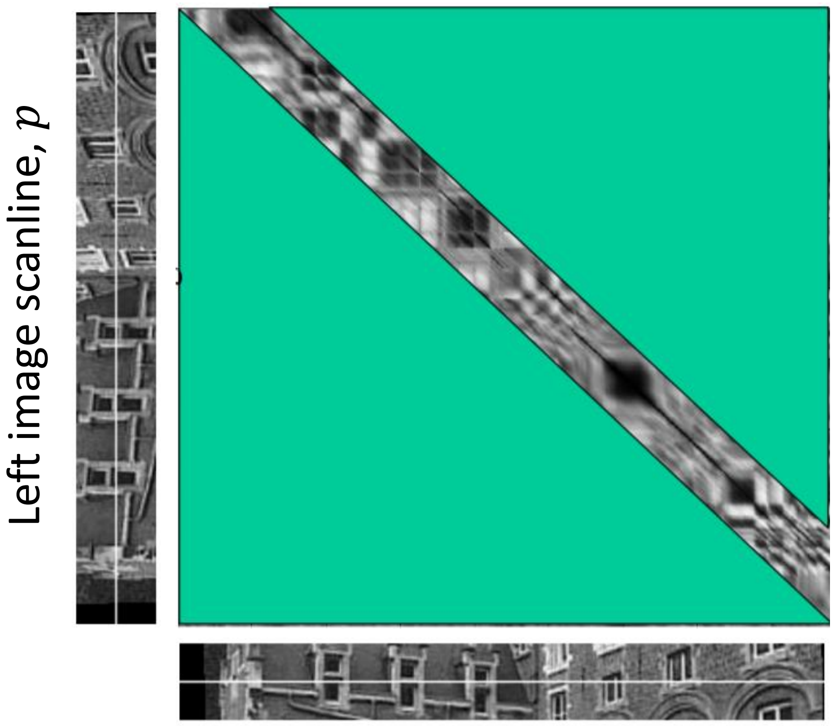
- The optimization is solved using dynamic programming.



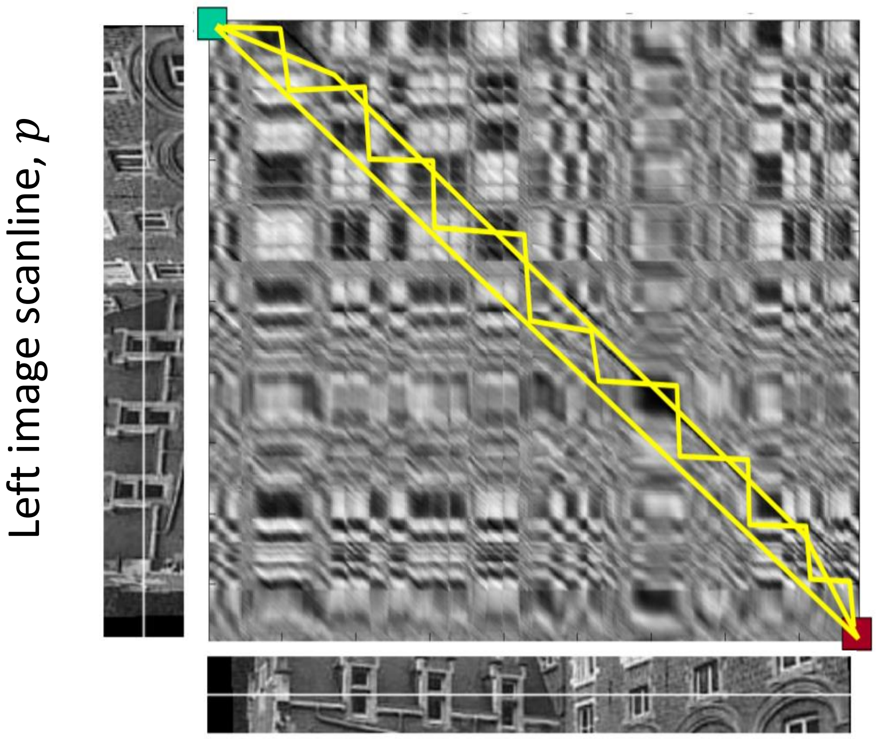
- Figure shows the dissimilarity matrix of left and right scanlines.
- Each row represents the dissimilarity cost $D(p, d_p)$ for a pixel p.
- p and q are neighbor on the same scanline.

Scanline Optimization Stereo

- Applying the bound on the maximum disparity, we get:



Find the shortest path from the start to end!



Scanline Optimization Stereo

- Considering each scanline independently leads to streaking artifacts.

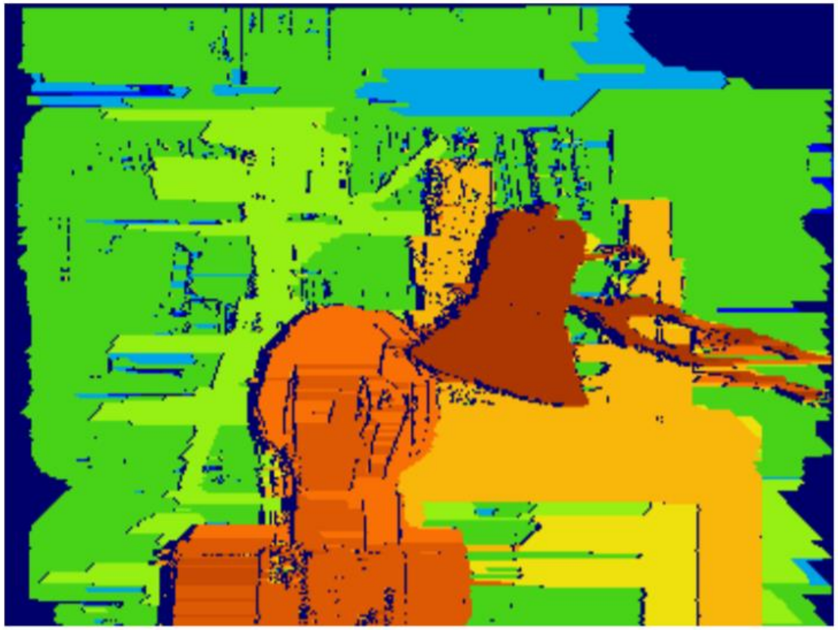


Image source: http://slazebni.cs.illinois.edu/spring19/lec18_stereo.pdf

Semi-Global Matching (SGM)

- The neighboring pixel q can be in any direction:

$$E(d) = \sum_p D(p, d_p) + \sum_{q \in \mathcal{N}(p)} R(d_p, d_q).$$

- In the most extreme case, we can consider $\mathcal{N}(p)$ to be the whole image, i.e. global matching.

- Unfortunately, this is a NP-complete problem which is too difficult to solve (we can use Graph-cut and alpha-expansion, but these are out of scope).

Semi-Global Matching (SGM)

- The computational complexity can be mitigated by semi-global matching (SGM).
- The idea behind SGM is to perform line optimization along multiple directions.
- And computing an aggregated cost $S(p, d)$ by summing the costs to reach pixel p with disparity d from each direction.

Semi-Global Matching (SGM)

- The accumulated cost $S(p,d) = \sum_r L_r(p,d)$ is the sum of all costs $L_r(p,d)$ to reach pixel p with disparity d along direction r.

- Each term can be expressed recursively as:

$$L_r(p,d) = D(p,d) + \min\Big\{L_r(p-r,d), L_r(p-r,d-1) + P_1, L_r(p-r,d+1) + P_1, \min_i L_r(p-r,i) + P_2\Big\} - \min_k L_r(p-r,k)$$

- The minimum cost at the previous pixel $\min_k L_r(p-r,k)$ is subtracted for numerical stability.

Semi-Global Matching (SGM)

- The value of disparity at each pixel is given by

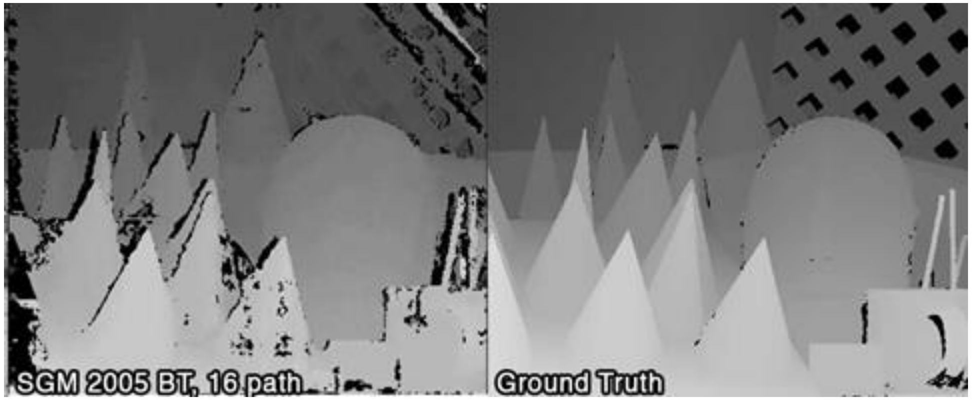
$$d^*(p) = \operatorname{argmin}_d S(p, d),$$

- and sub-pixel accuracy can be achieved by fitting a curve in $d^*(p)$ and its neighbouring costs and taking the minimum along the curve.

Semi-Global Matching (SGM)

Left Image

3x3 block matching

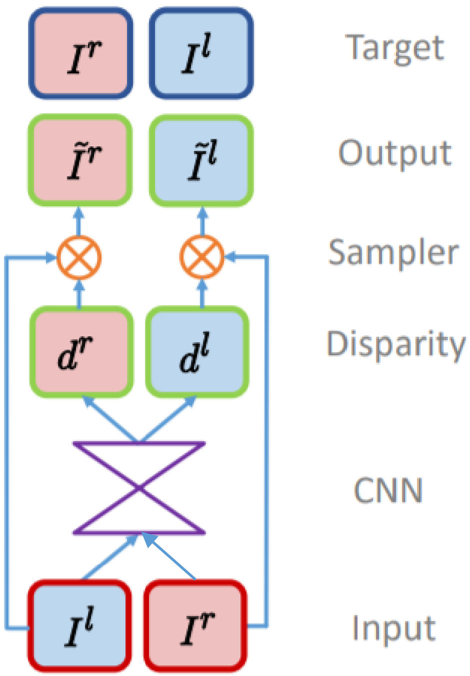


Single-View Stereo – Deep Learning

- Given a single image I at test time, our goal is to learn a function f that can predict the per-pixel scene depth, $\hat{d} = f(I)$.

- Use a Convolution Neural Network (CNN) to model the function f.

- Training is done by enforcing the left-right consistency.



Reference: C Godard et al, "Unsupervised Monocular Depth Estimation with Left-Right Consistency", CVPR 2017.

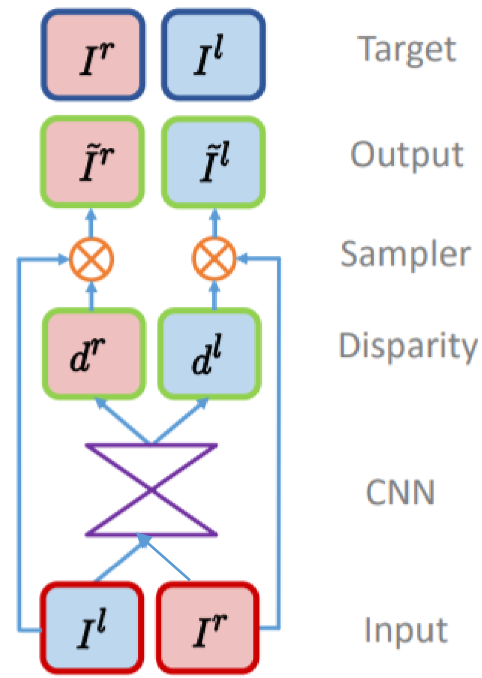
Single-View Stereo - Deep Learning

- At training time, we have the left and right color images I_l and I_r from a calibrated stereo pair.

- Use the right disparity d^r predicted by the CNN and the left image I^l to reconstruct the right image, i.e. $\tilde{I}^r = I^l(d^r)$.

- The reconstruction is just a pixel shift along the scanline since this is a calibrated stereo.

- Similarly, we reconstruct the left image, i.e. $\tilde{I}^l = I^r(d^l)$.



Reference: C Godard et al, "Unsupervised Monocular Depth Estimation with Left-Right Consistency", CVPR 2017.

Single-View Stereo - Deep Learning

- Training loss:

$$C_s = \alpha_{ap}(C^l_{ap} + C^r_{ap}) + \alpha_{ds}(C^l_{ds} + C^r_{ds}) + \alpha_{lr}(C^l_{lr} + C^r_{lr}),$$

- C_{ap} encourages the reconstructed image to appear similar to the corresponding training input.

- C_{ds} enforces smooth disparities, and C_{lr} prefers the predicted left and right disparities to be consistent.

Reference: C Godard et al, “Unsupervised Monocular Depth Estimation with Left-Right Consistency”, CVPR 2017.

Single-View Stereo - Deep Learning

- Appearance Matching Loss:

$$C_{ap}^{l} = \frac{1}{N}\sum_{i,j}\alpha\underbrace{\frac{1-\mathrm{SSIM}(I_{ij}^{l},\tilde{I}_{ij}^{l})}{2}}_{\text{Matching cost}} + (1-\alpha)\underbrace{\left\|I_{ij}^{l}-\tilde{I}_{ij}^{l}\right\|}_{\text{L1 loss between image pixels}}.$$

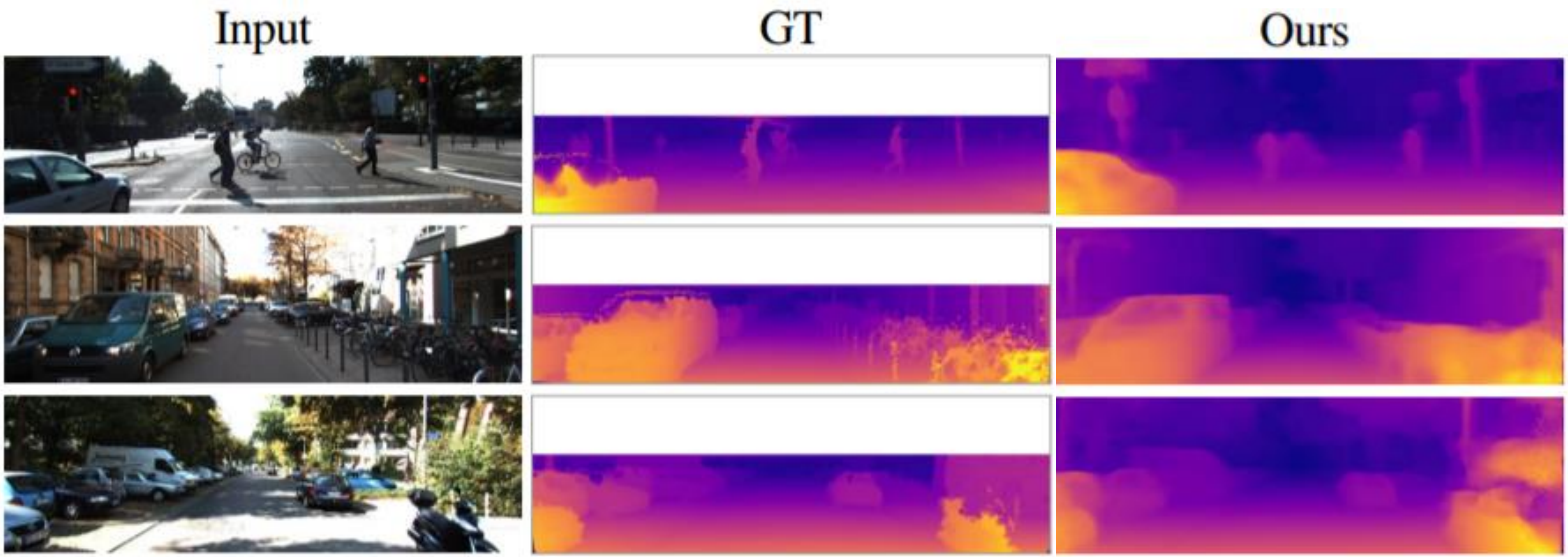
- Disparity Smoothness Loss:

$$C_{ds}^{l} = \frac{1}{N}\sum_{i,j}\left|\underbrace{\partial_x d_{ij}^{l}}_{\text{disparity gradients}}\right| e^{-\left\|\partial_x I_{ij}^{l}\right\|} + \left|\partial_y d_{ij}^{l}\right| e^{-\underbrace{\left\|\partial_y I_{ij}^{l}\right\|}_{\text{image gradients}}}.$$

- Left-Right Disparity Consistency Loss:

$$C_{lr}^{l} = \frac{1}{N}\sum_{i,j}\left|d_{ij}^{l} - d^{r}_{\underbrace{ij+d_{ij}^{l}}_{\text{projected right-view disparity map}}}\right|.$$

Single-View Stereo - Deep Learning



Reference: C Godard et al, “Unsupervised Monocular Depth Estimation with Left-Right Consistency”, CVPR 2017.

Multi-View Stereo

- **Recall:** Multi-view stereo is the last step of large-scale 3D reconstruction.
- Given the multi-view images and the camera poses, the goal is to find the depth maps of all the images.
- We will use the plane sweeping algorithm to achieve this.

Multi-View Stereo

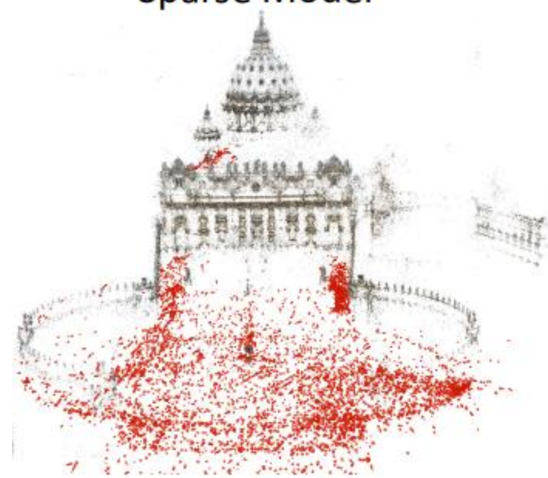
Unstructured Images

Data Association

Scene Graph

Structure-from-Motion

Sparse Model



Multi-view Stereo

Dense Model

Image source: https://demuc.de/tutorials/cvpr2017/sparse-modeling.pdf

Plane Sweeping Algorithm

- The inputs to the algorithm are M 3D-planes for the depth tests, $N + 1$ images at different camera positions.

- We assume images have been corrected for radial distortion and their respective camera projection matrices P_k:

$$P_k = K_k[R_k^\top \ -R_k^\top C_k] \text{ with } k = 1, \ldots, N,$$

- K_k is the camera calibration matrix, and R_k, C_k are the rotation and translation of camera P_k with respect to the reference camera P_{ref}.

Plane Sweeping Algorithm

- The reference camera is assumed to be the origin of the coordinate system.
- Accordingly, its projection matrix is $P_{ref} = K_{ref}[\, I_{3\times3} \mid 0]$.
- The family of depth planes Π_m with $m = 1, \ldots, M$ is defined in the coordinate frame of the reference view by:

$$\Pi_m = [n_m^\top \; -d_m] \text{ for } m = 1, \ldots, M$$

- $n_m^\top$ is the unit length normal of the plane and d_m is the distance of the plane to the origin.

Plane Sweeping Algorithm

- For a fronto-parallel sweep $n_m^\top = [0\ 0\ 1]$.
- The depths d_m of the planes Π_m fall within the interval $[d_{near}, d_{far}]$.
- To test the plane hypothesis Π_m for a given pixel (x, y) in the reference view I_{ref}, the pixel is projected into the other images $k = 1, \ldots, N$.
- The mapping from the image plane of the reference camera P_{ref} to the image plane of the camera P_k is a planar mapping.

Plane Sweeping Algorithm

- This planar mapping is described by a homography H_{Π_m, P_k} induced by the plane Π_m, i.e.

$$H_{\Pi_m, P_k} = K_k \left(R_k^\top - \frac{R_k^\top C_k n_m^\top}{d_m} \right) K_{ref}^{-1}.$$

- The location (x_k, y_k) in image I_k of the mapped pixel (x, y) of the reference view is computed by:

$$\begin{bmatrix} \tilde{x} & \tilde{y} & \tilde{w} \end{bmatrix}^\top = H_{\Pi_m, P_k} \begin{bmatrix} x & y & 1 \end{bmatrix}^\top$$

$$x_k = \tilde{x}/\tilde{w}, \qquad y_k = \tilde{y}/\tilde{w}.$$

Plane Sweeping Algorithm

- We use the absolute difference of intensities (or other matching cost) as the dissimilarity measure at the pixel location (x, y) in the reference view and the plane Π_m:

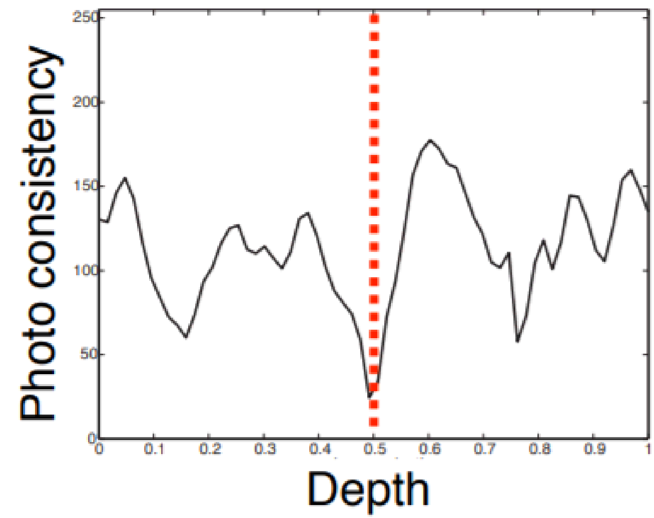
$$
\begin{aligned}
C(x, y, \Pi_k) \; &= \; \sum_{k=0}^{N-1} \sum_{(i,j)\in W} |I_{ref}(x-i, y-j) \\
&- \; \beta_k^{ref} I_k(x_k - i, y_k - j)|,
\end{aligned}
$$

- W is a rectangular window centered at the pixel (x, y), and (x_k, y_k) is obtained by applying the homography H_{Π_m, P_k} as shown earlier.

- β_k^{ref} corresponds to the gain ratio between image k and the reference.

Plane Sweeping Algorithm

$$
\begin{aligned}
C(x, y, \Pi_k) \;=\; & \sum_{k=0}^{N-1} \sum_{(i,j)\in W} |I_{ref}(x-i, y-j) \\
& - \beta_k^{ref} I_k(x_k - i, y_k - j)|,
\end{aligned}
$$



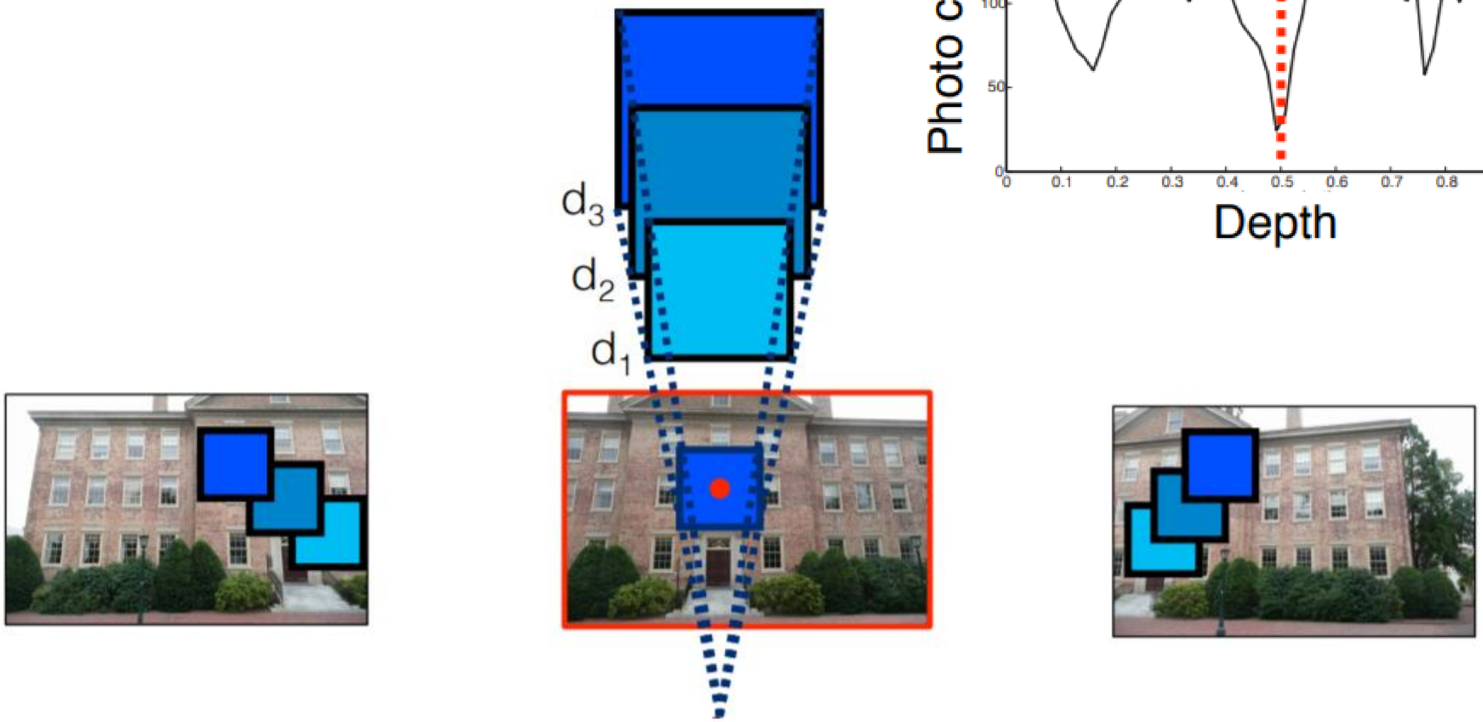


Image source: https://demuc.de/tutorials/cvpr2017/dense-modeling.pdf

Plane Sweeping Algorithm

- Once the cost function for all pixels has been computed the depth map may be extracted.
- The first step is to select the best plane at each pixel in the reference view.
- This may simply be the plane of minimum cost, also called best-cost $\tilde{\Pi}(x, y)$ or winner-takes-all, defined as follows:

$$\tilde{\Pi}(x, y) = \underset{\Pi_m}{\operatorname{argmin}}\, C(x, y, \Pi_m).$$

Plane Sweeping Algorithm

- For a given plane Π_m at pixel (x, y), the depth can be computed by finding the intersection of Π_m and the ray through the pixel's center.

- This is given by:

$$Z_m(x,y) = \frac{d_m}{\begin{bmatrix} x & y & 1 \end{bmatrix} K_{ref}^{-\top} n_m}.$$

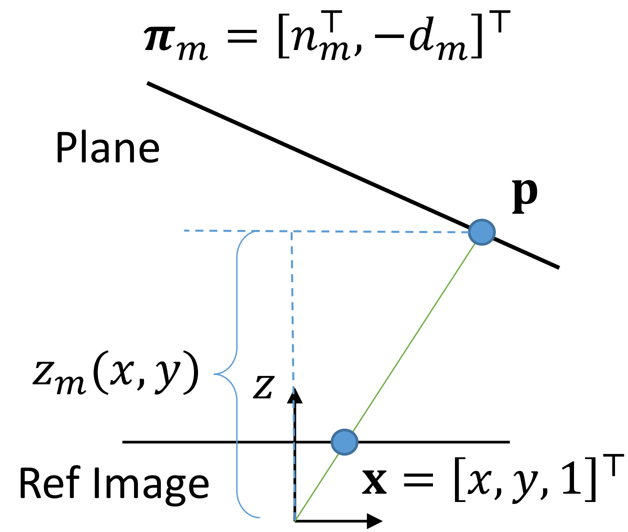
Plane Sweeping Algorithm

Proof:

Camera intrinsic: K_{ref}

The backprojected ray from pixel $\mathbf{x}$ is given by: $K_{ref}^{-1}[x, y, 1]^\top$.

Let $\mathbf{p} = z_m(x, y)K_{ref}^{-1}[x, y, 1]^\top$ be the point where the ray intersects the plane.



Thus, $\mathbf{p}^\top n_m = d_m$

$$\Longrightarrow \left(z_m(x, y)K_{ref}^{-1}[x, y, 1]^\top\right)^\top n_m = d_m$$

$$\Longrightarrow z_m(x, y)[x, y, 1]K_{ref}^{-\top} n_m = d_m \qquad \Longrightarrow z_m(x, y) = \frac{d_m}{[x, y, 1]K_{ref}^{-\top} n_m}.$$

$\square$

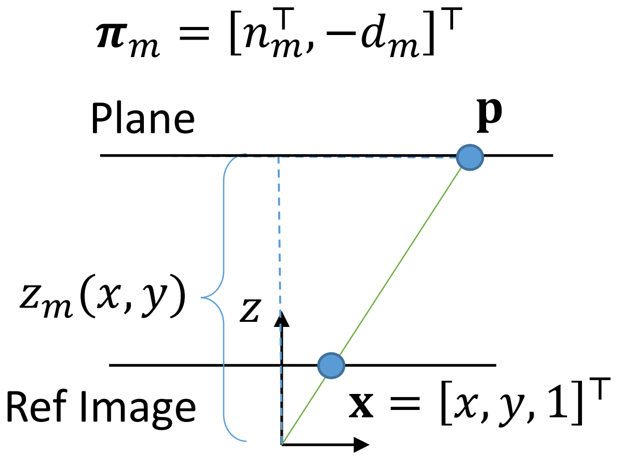
Plane Sweeping Algorithm

For a fronto-parallel plane to the reference image, we have $z_m(x, y) = d_m$.

Proof:

The denominator $[x, y, 1]K_{ref}^{-\top} n_m = 1$ since $n_m = [0, 0, 1]^\top$.

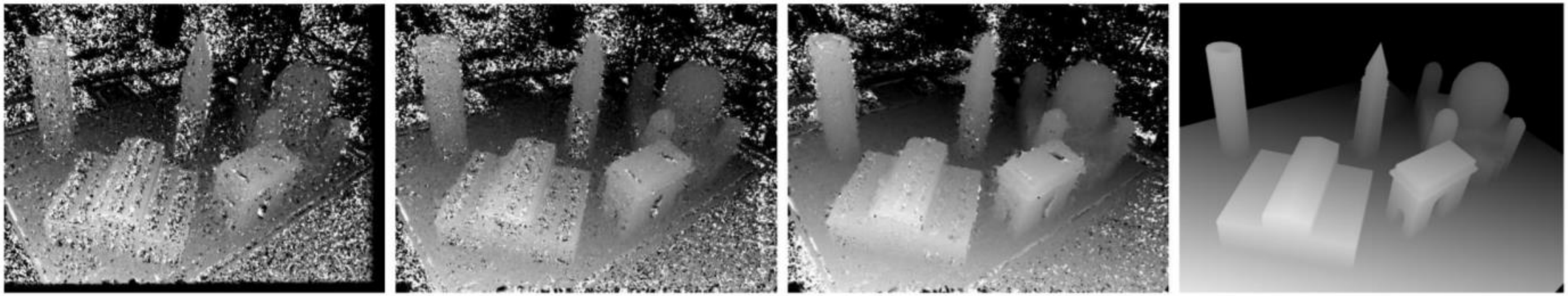
$\Longrightarrow z_m(x, y) = d_m$.



$\square$

Plane Sweeping Algorithm

- The quality of the depth map improves with more views.



(a) 2 views (b) 5 views (c) 20 views (d) Ground Truth

Source: Newcombe, 2013

Summary

- We have looked at how to:

1. Do stereo rectification and correspondence search along scanlines to get the disparity value of each pixel in two-view stereo.

2. Compute depth values from the disparity map.

3. Explain the concepts of scanline optimization and semi-global matching for two-view stereo.

4. Perform multi-view stereo using the plane sweeping algorithm.