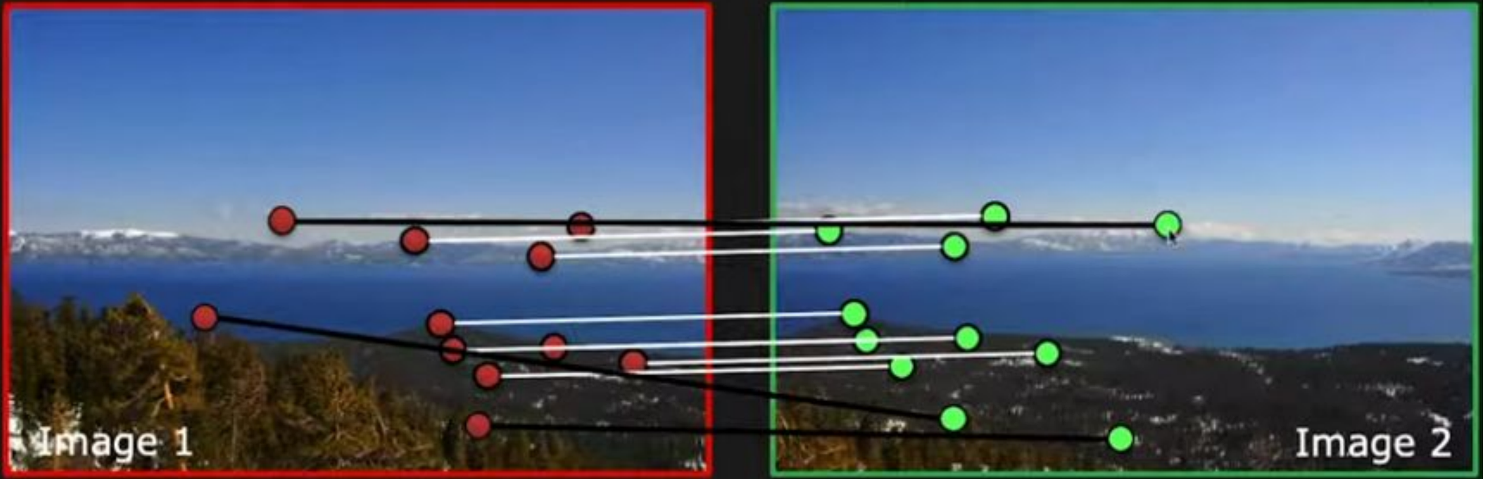
EE655: Computer Vision & Deep Learning

Lecture 18

Koteswar Rao Jerripothula, PhD
Department of Electrical Engineering
IIT Kanpur

What Could Go Wrong?



Outliers!

We need to robustly compute transformation in the presence of wrong matches.

With Homography

RANdom SAmple Consensus

General RANSAC Algorithm:

1. Randomly choose s samples. Typically s is the minimum samples to fit a model.
2. Fit the model to the randomly chosen samples.
3. Count the number M of data points (inliers) that fit the model within a measure of error ε.
4. Repeat Steps 1-3 N times
5. Choose the model that has the largest number M of inliers.

For homography:

$s = 4$ points. ε is acceptable alignment error in pixels.

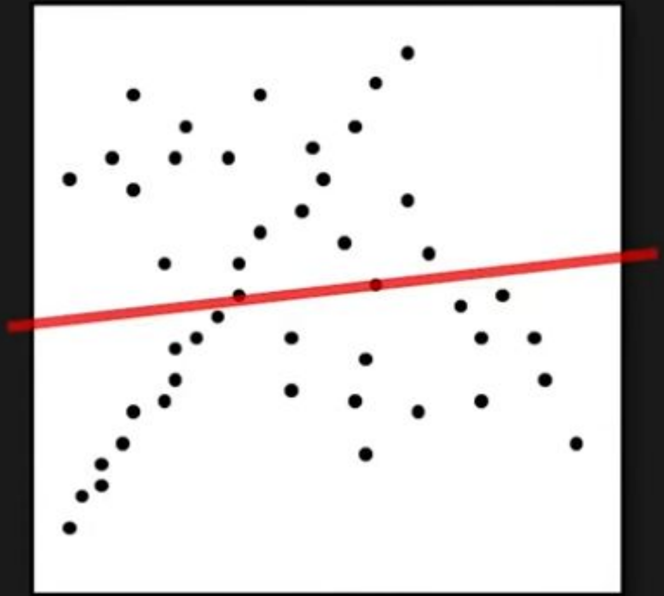
Model in the case of Homography

$$
\begin{bmatrix}
x_s^{(1)} & y_s^{(1)} & 1 & 0 & 0 & 0 & -x_d^{(1)}x_s^{(1)} & -x_d^{(1)}y_s^{(1)} & -x_d^{(1)} \\
0 & 0 & 0 & x_s^{(1)} & y_s^{(1)} & 1 & -y_d^{(1)}x_s^{(1)} & -y_d^{(1)}y_s^{(1)} & -y_d^{(1)} \\
 & & & & \vdots & & & & \\
x_s^{(i)} & y_s^{(i)} & 1 & 0 & 0 & 0 & -x_d^{(i)}x_s^{(i)} & -x_d^{(i)}y_s^{(i)} & -x_d^{(i)} \\
0 & 0 & 0 & x_s^{(i)} & y_s^{(i)} & 1 & -y_d^{(i)}x_s^{(i)} & -y_d^{(i)}y_s^{(i)} & -y_d^{(i)} \\
 & & & & \vdots & & & & \\
x_s^{(n)} & y_s^{(n)} & 1 & 0 & 0 & 0 & -x_d^{(n)}x_s^{(n)} & -x_d^{(n)}y_s^{(n)} & -x_d^{(n)} \\
0 & 0 & 0 & x_s^{(n)} & y_s^{(n)} & 1 & -y_d^{(n)}x_s^{(n)} & -y_d^{(n)}y_s^{(n)} & -y_d^{(n)}
\end{bmatrix}
\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}
$$

Absolute difference between LHS and RHS helps measure the error

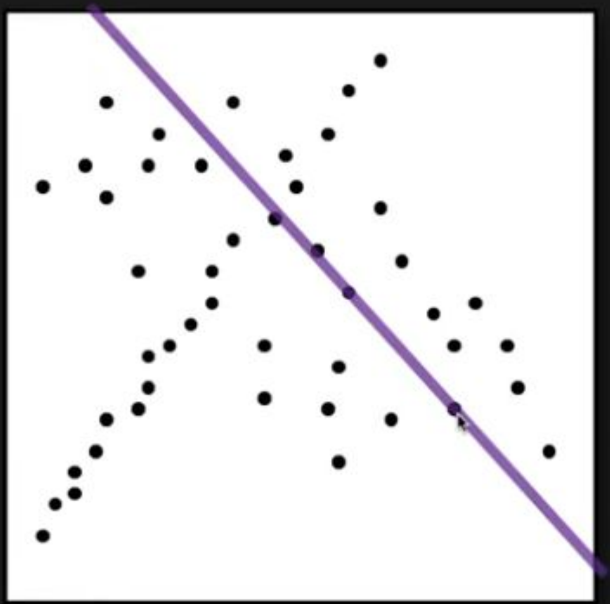
RANSAC Example: Line Fitting

Robust line fitting:



Least Squares Fitting

Inliers: 2

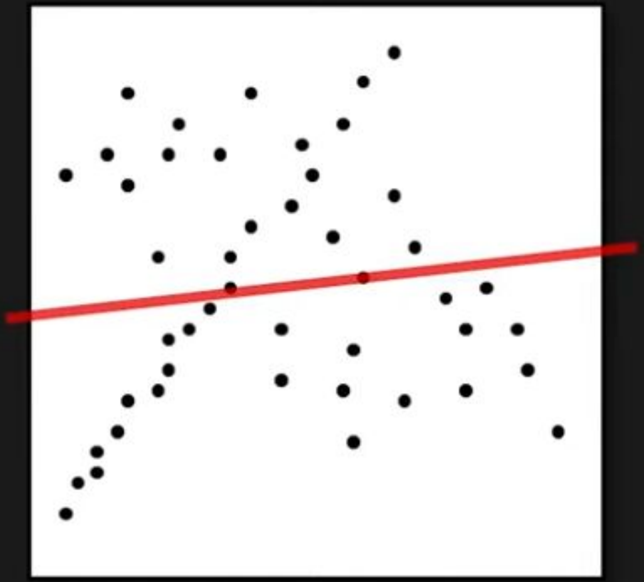


RANSAC Iteration 1

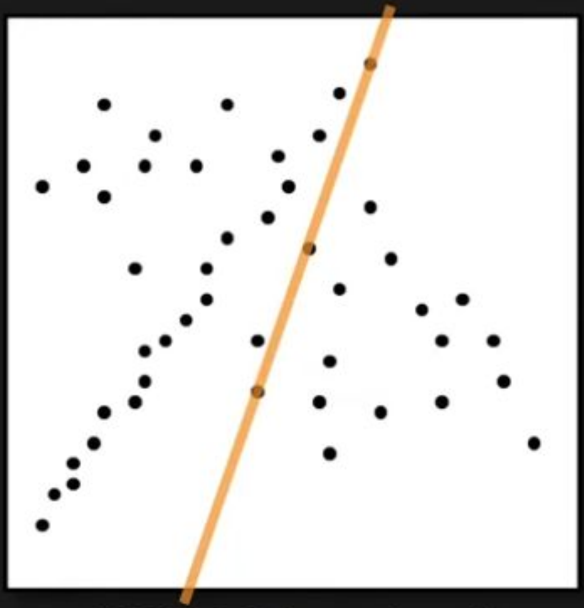
Inliers: 4

RANSAC Example: Line Fitting

Robust line fitting:



Least Squares Fitting

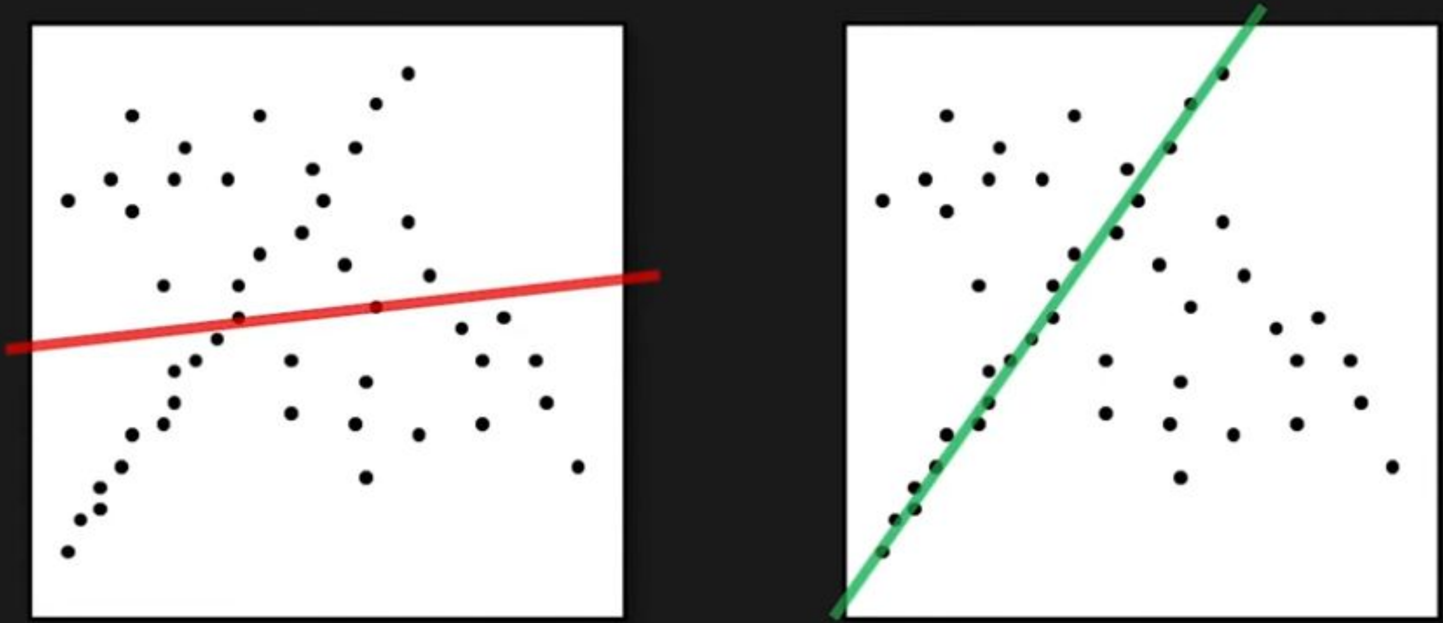


RANSAC Iteration 2

Inliers: 3

RANSAC Example: Line Fitting

Robust line fitting:



Least Squares Fitting

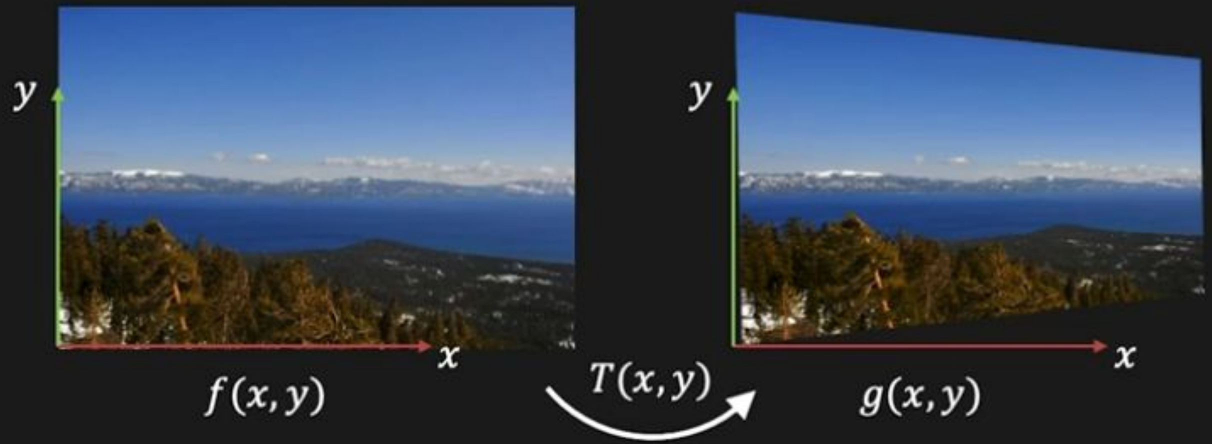
RANSAC Iteration i

Inliers: 20

Warping Images

Given a transformation T and a image $f(x,y)$, compute the transformed image $g(x,y)$

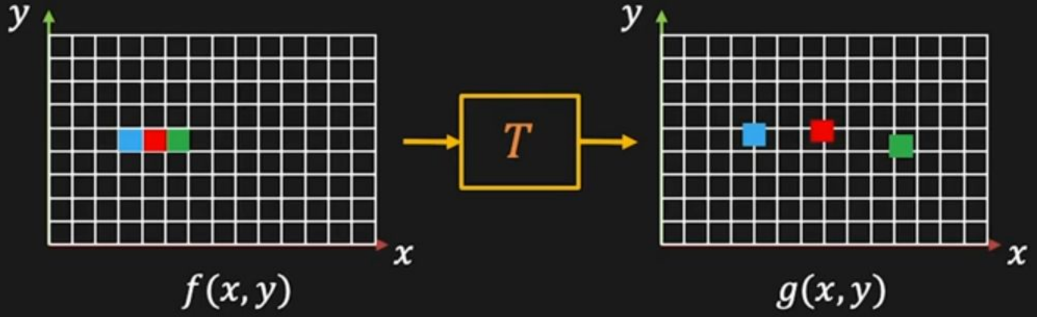
$$\boxed{g(x,y) = f\big(T(x,y)\big)}$$



Forward Warping

Send each pixel (x, y) in $f(x, y)$ to its corresponding location $T(x, y)$ in $g(x, y)$

$$\boxed{g(x,y) = f\big(T(x,y)\big)}$$



What if pixel lands in between pixels?

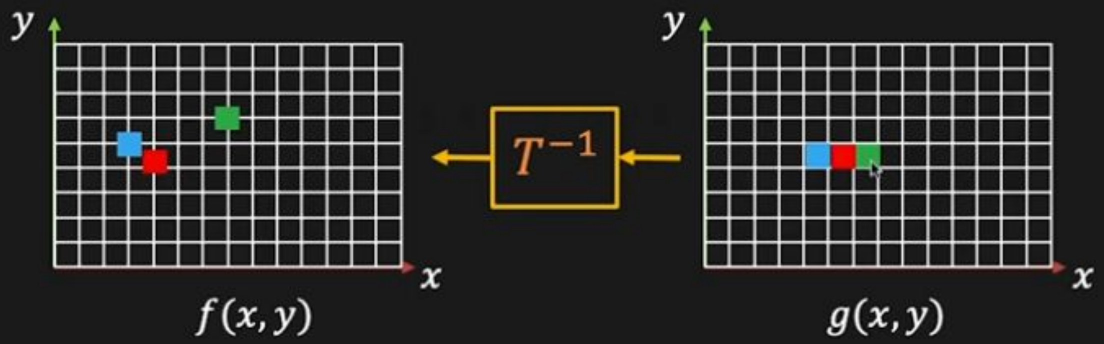
What if not all pixels in $g(x, y)$ are filled?

Can result in holes!

Backward Warping

Get each pixel (x, y) in $g(x, y)$ from its corresponding location $T^{-1}(x, y)$ in $f(x, y)$

$$\boxed{g(x, y) = f(T(x, y))}$$



What if pixel lands between pixels?

Use Nearest Neighbor or Interpolate

NOTE:

1) Before backward warping, forward warping of corner points has to be done.
2) All the pixels within the warped corner points are back warped.

Image Alignment Process

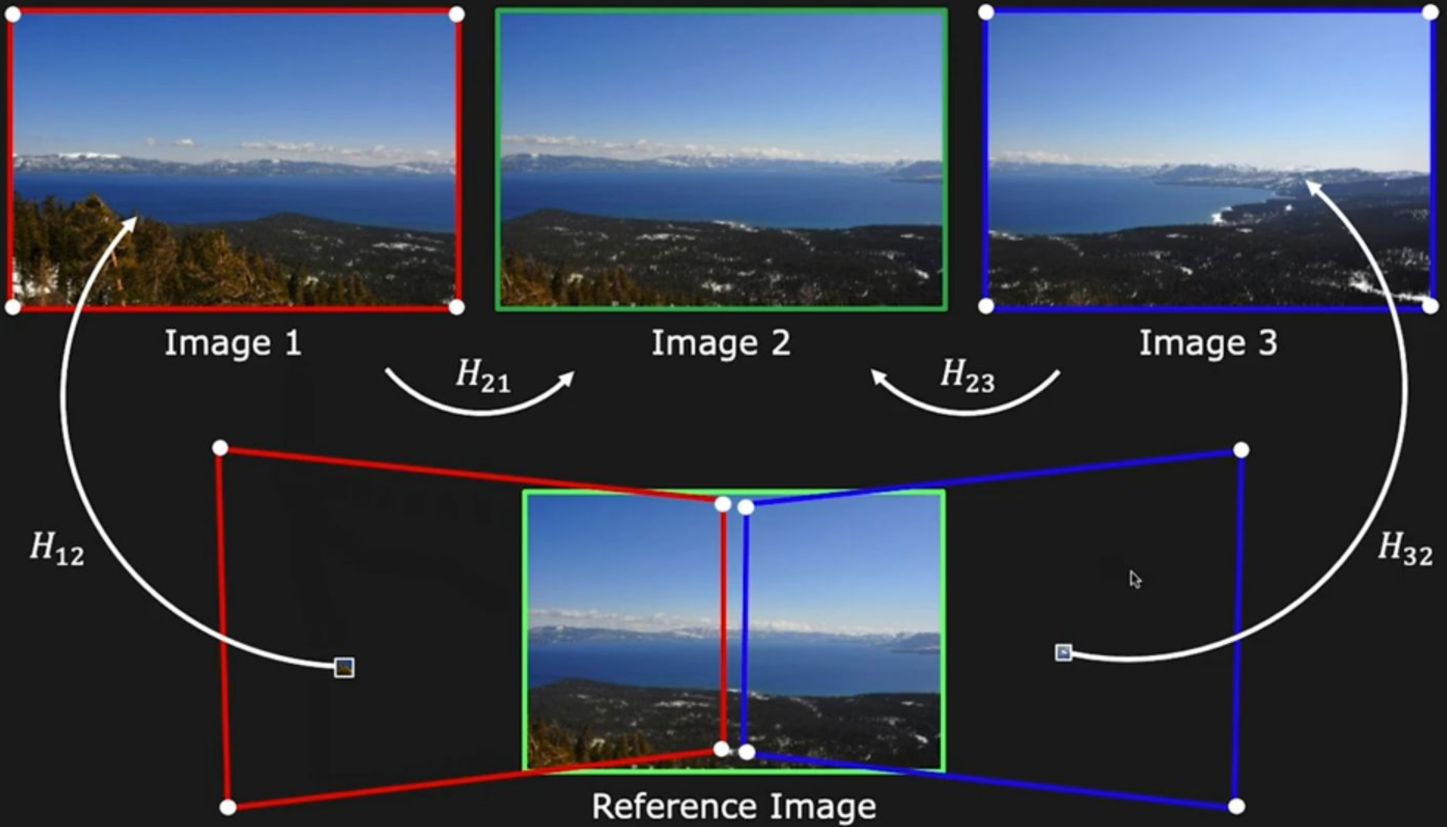
Image 1

Image 2

Image 3

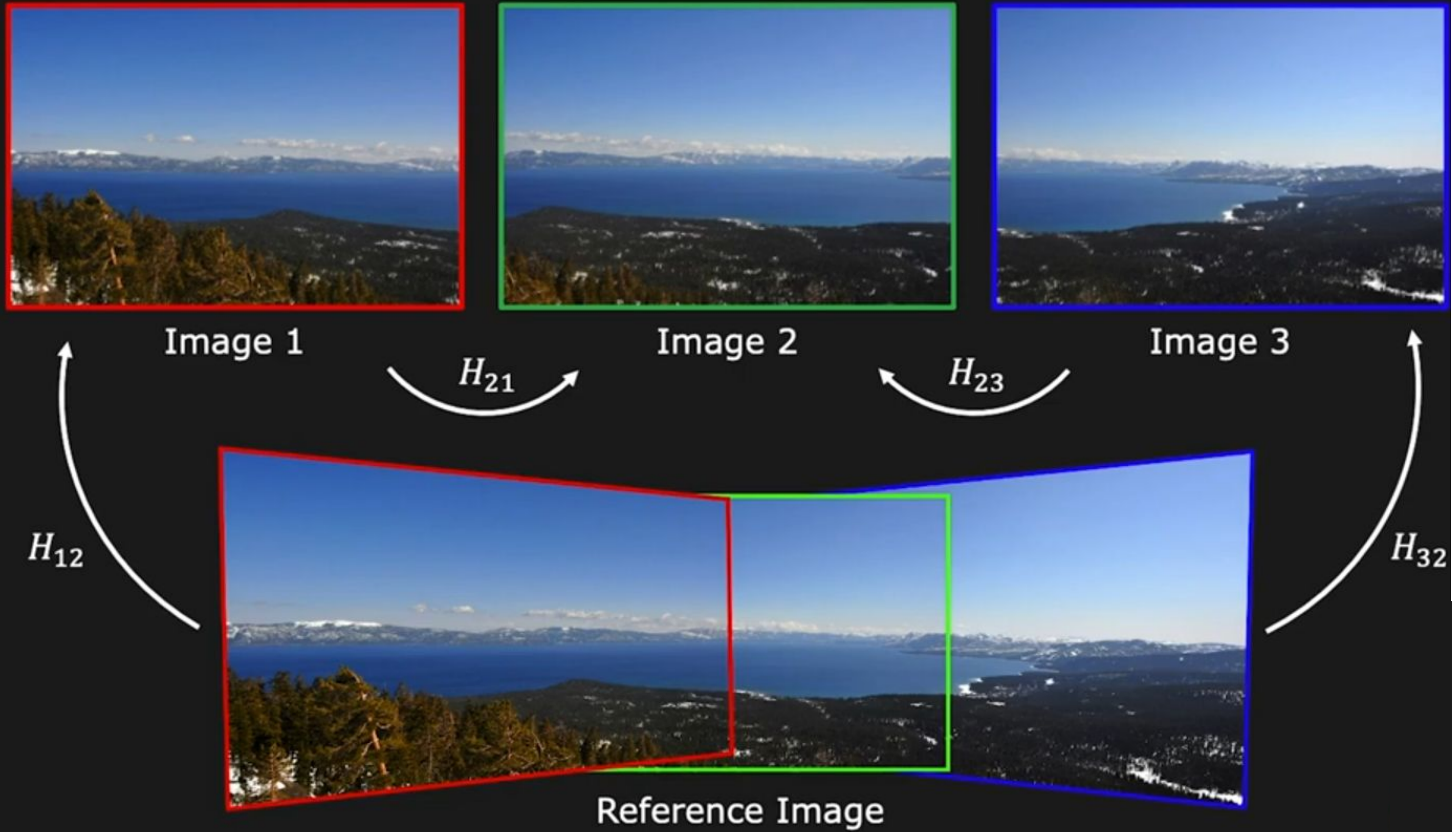
Reference Image
(Image 2)

Image Alignment Process



For each pixel within bounds, compute its location in captured image

Image Alignment Process

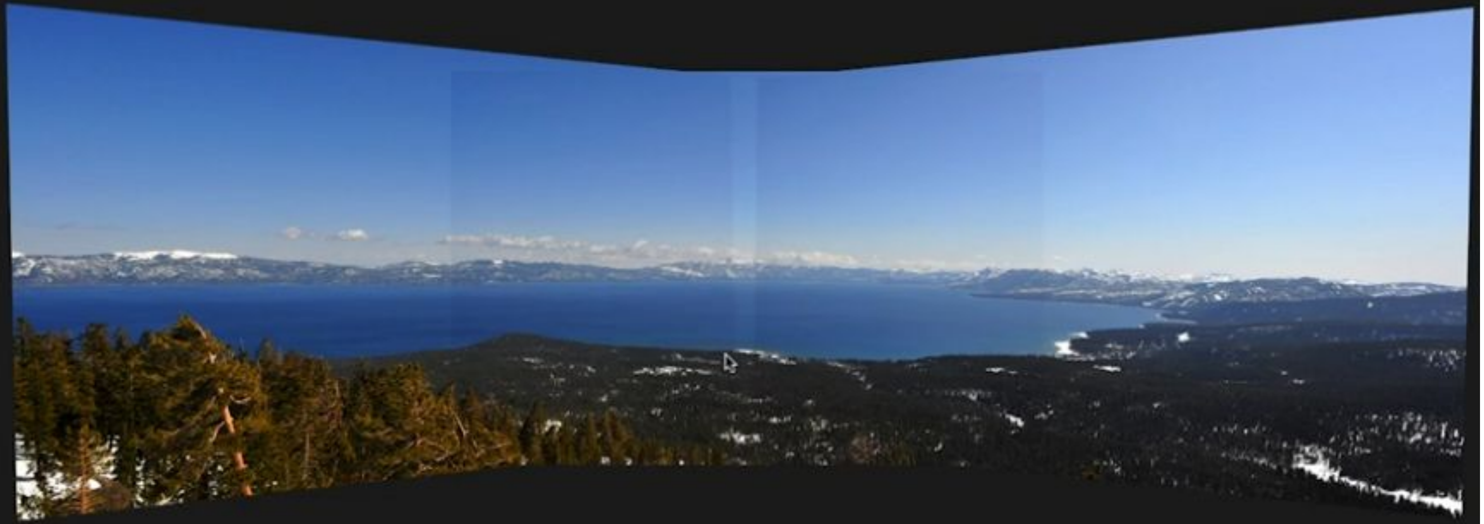


For each pixel within bounds, compute its location in captured image

Blending Images

Overlaid Aligned Images

Blending Images: Averaging



Averaged Images

Seams still visible

Blending Images

Say we want to blend images I_1 and I_2 at the center

Image I_1

+

Image I_2

=

Hard overlay

Weight w_1

Weight w_2

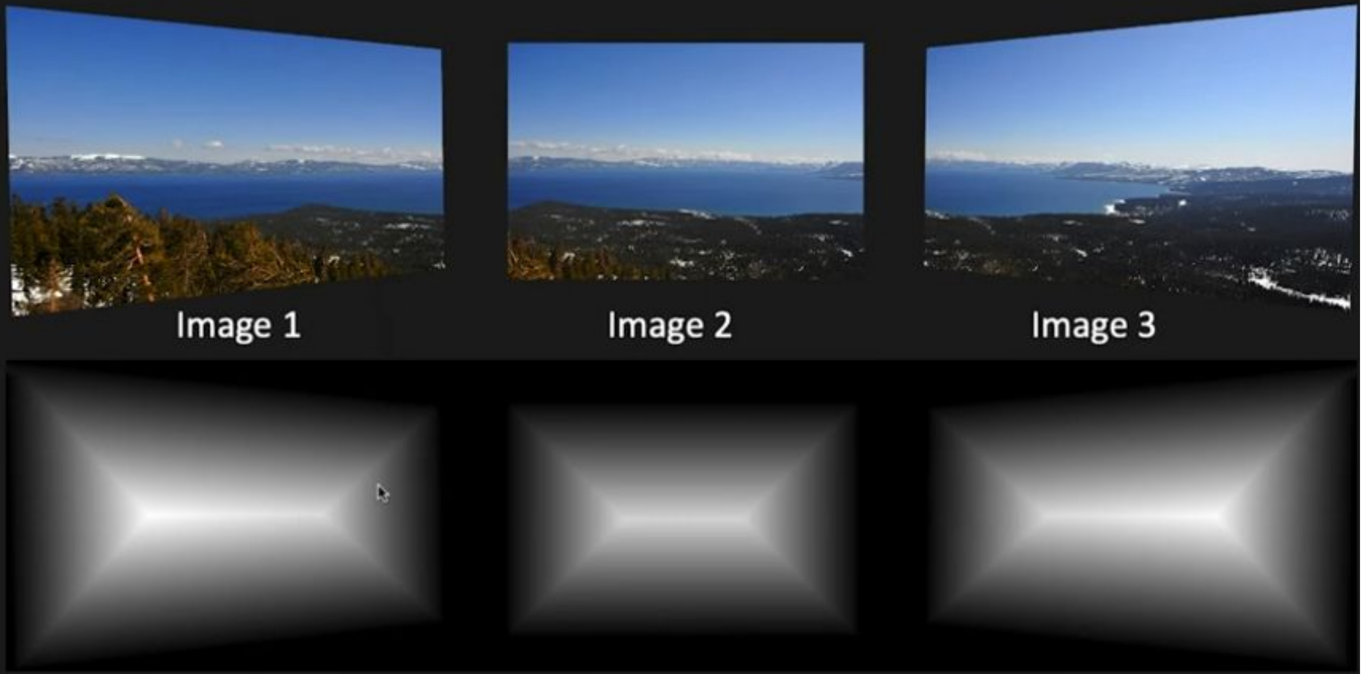
Image I_1 | Image I_2 | Blended Image I_{blend}

Weight w_1

Weight w_2

$$I_{blend} = \frac{w_1 I_1 + w_2 I_2}{w_1 + w_2}$$

Computing Weighting Functions



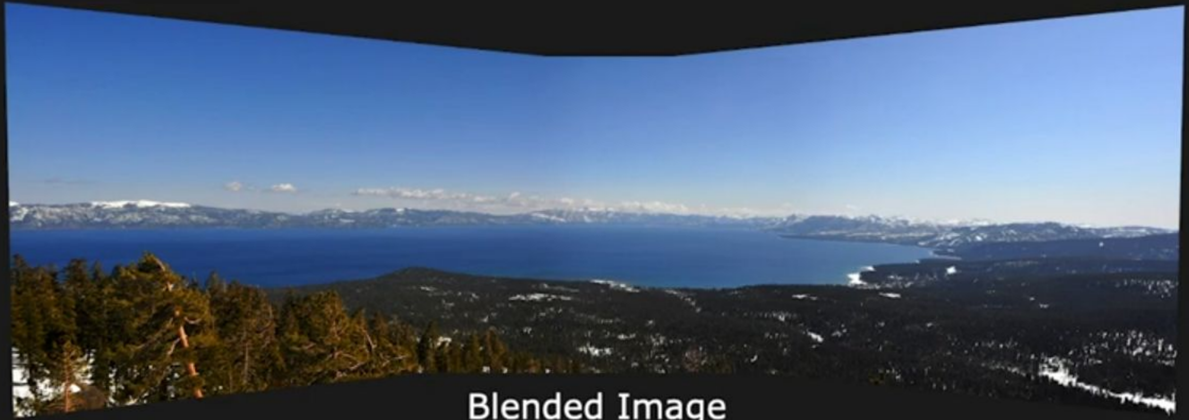
Weight w_1 Weight w_2 Weight w_3

Pixels closer to the edge get a lower weight.

Ex: Distance Transform (`bwdist` in MATLAB).

Weighted Blending

Overlaid Aligned Images



Blended Image

Image Stitching Example

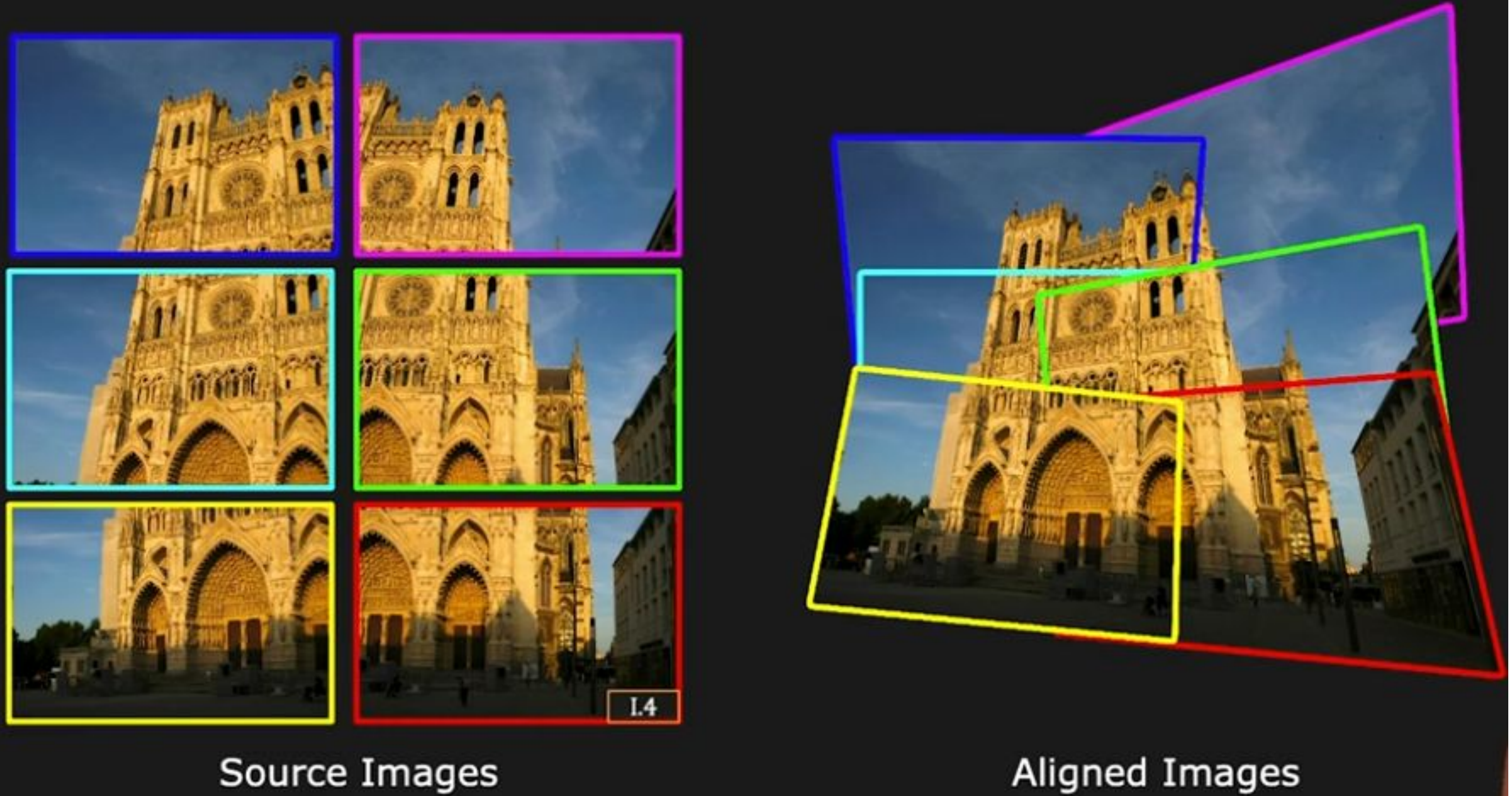
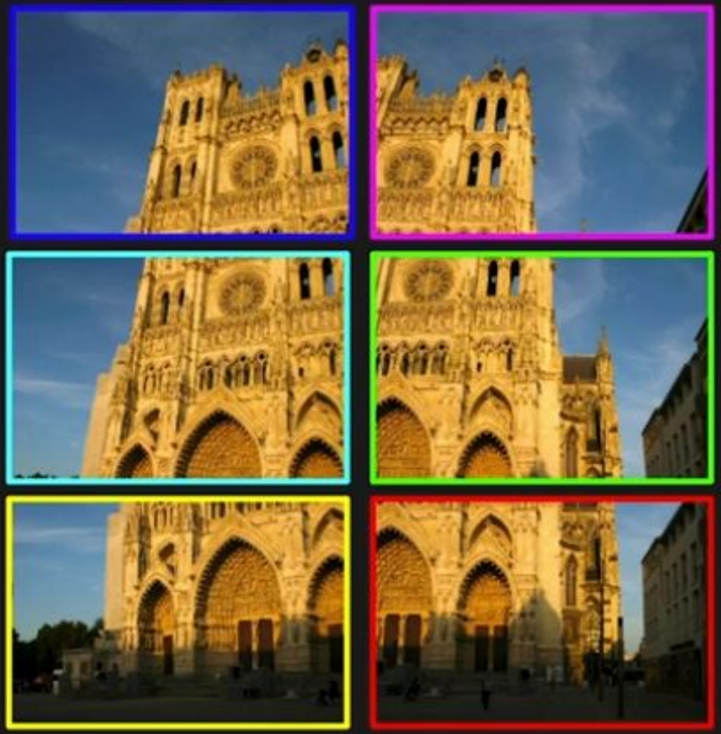


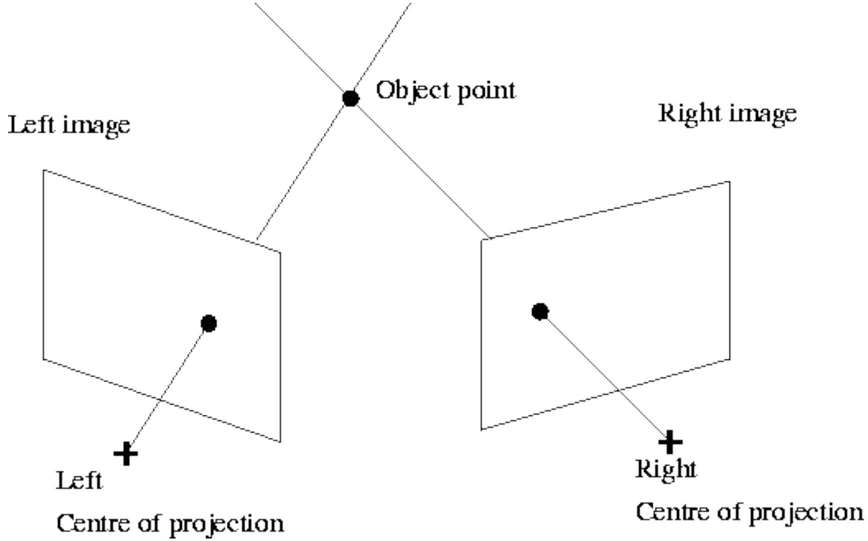
Image Stitching Example



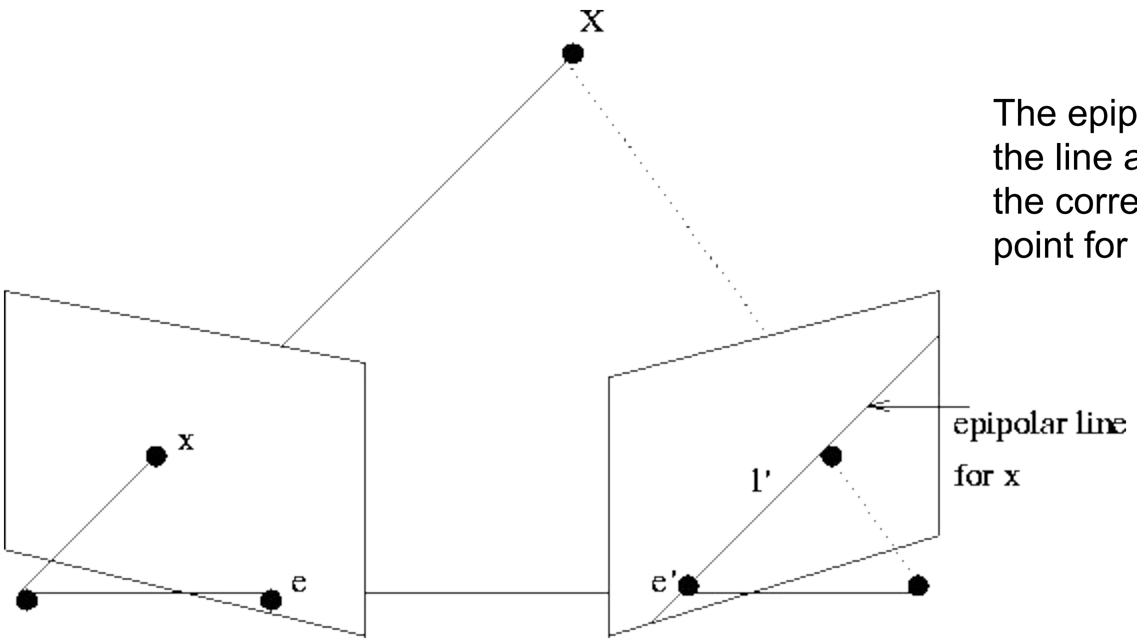
Source Images

Blended Image

Triangulation



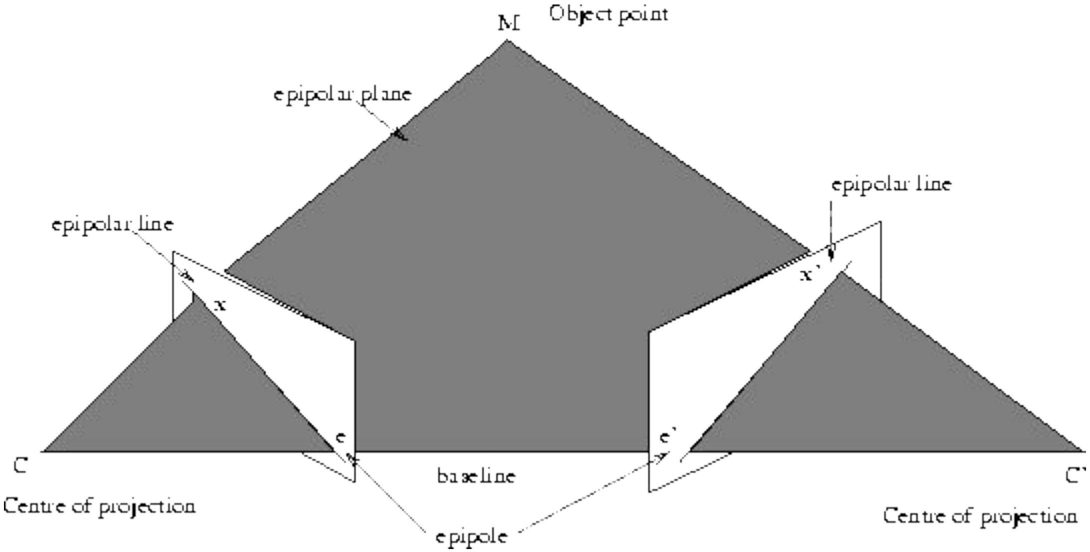
Given a single image, the three-dimensional location of any visible object point must lie on the straight line that passes through the centre of projection and the image of the object point. The determination of the intersection of two such lines generated from two independent images is called triangulation.



The epipolar line is the line along which the corresponding point for x must lie.

The two lines result in a plane, intersection of which with image plane gives the epipolar line of point x of one image on the other image

Epilpolar Geometry



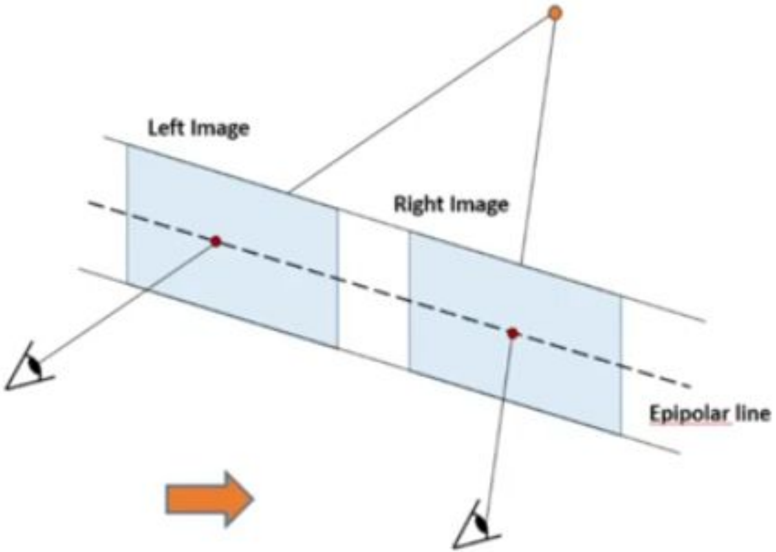
The *epipole* is the point of intersection of the line joining the optical centres, that is the *baseline*, with the image plane. Thus the epipole is the image, in one camera, of the optical centre of the other camera.

The *epipolar plane* is the plane defined by a 3D point *M* and the optical centres *C* and *C*'.

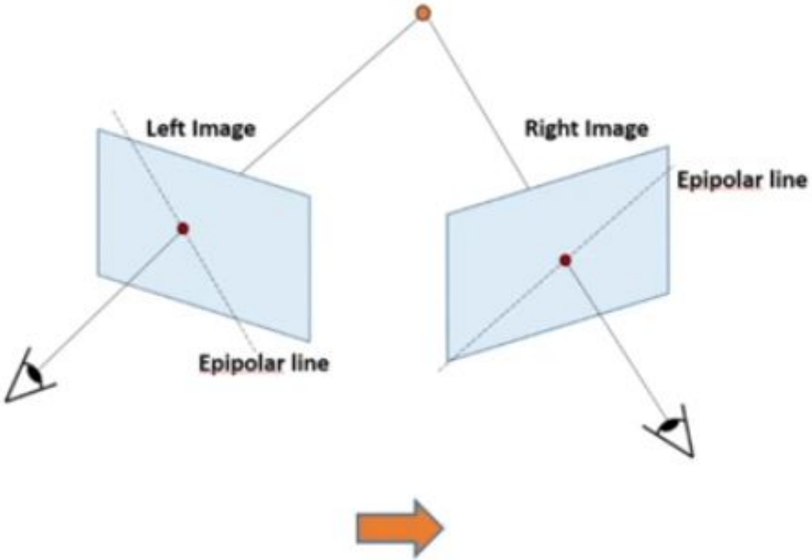
The *epipolar line* is the straight line of intersection of the epipolar plane with the image plane. It is the image in one camera of a ray through the optical centre and image point in the other camera. All epipolar lines intersect at the epipole.

Two cases of Epipolar Geometry

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



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



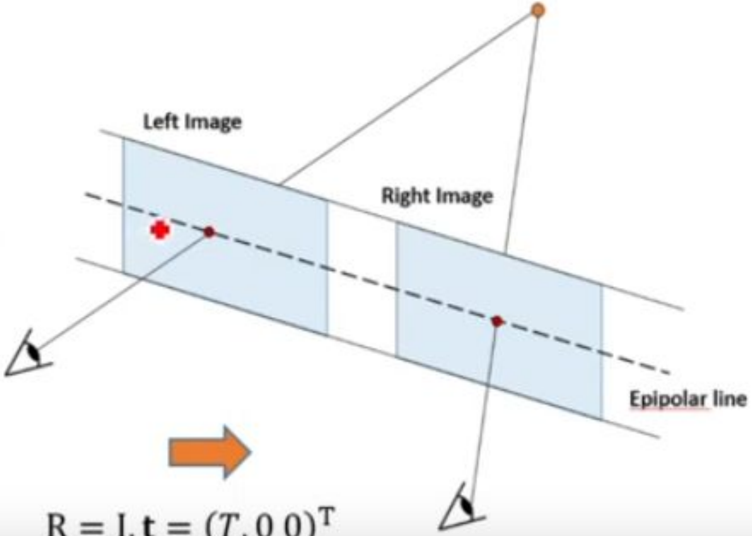
$[\mathrm{R} \mid \mathbf{t}]$

Simplest Case: Parallel Images

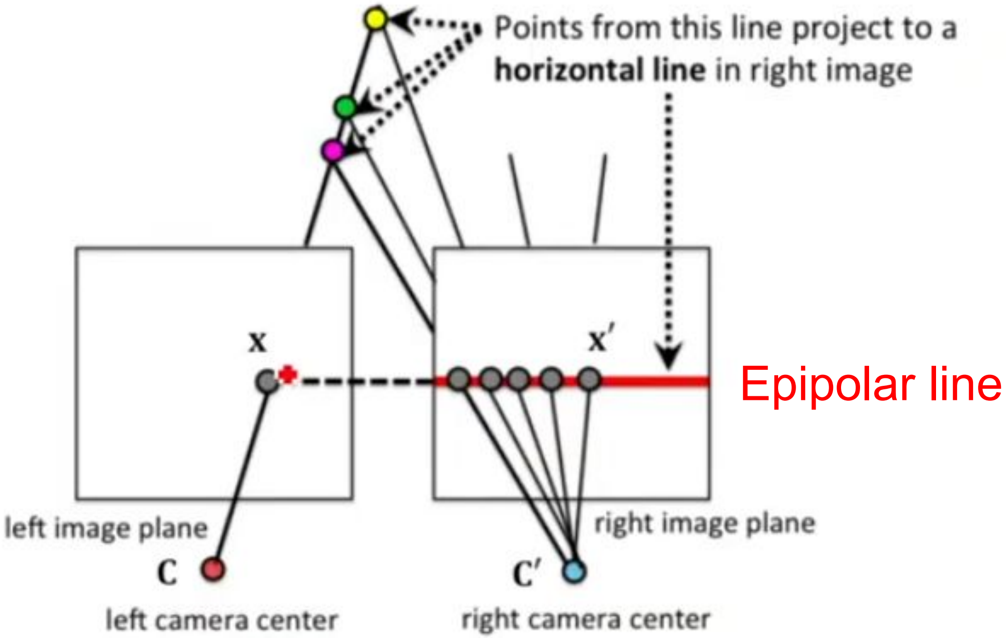
- Image planes of cameras are parallel to each other and to the baseline.

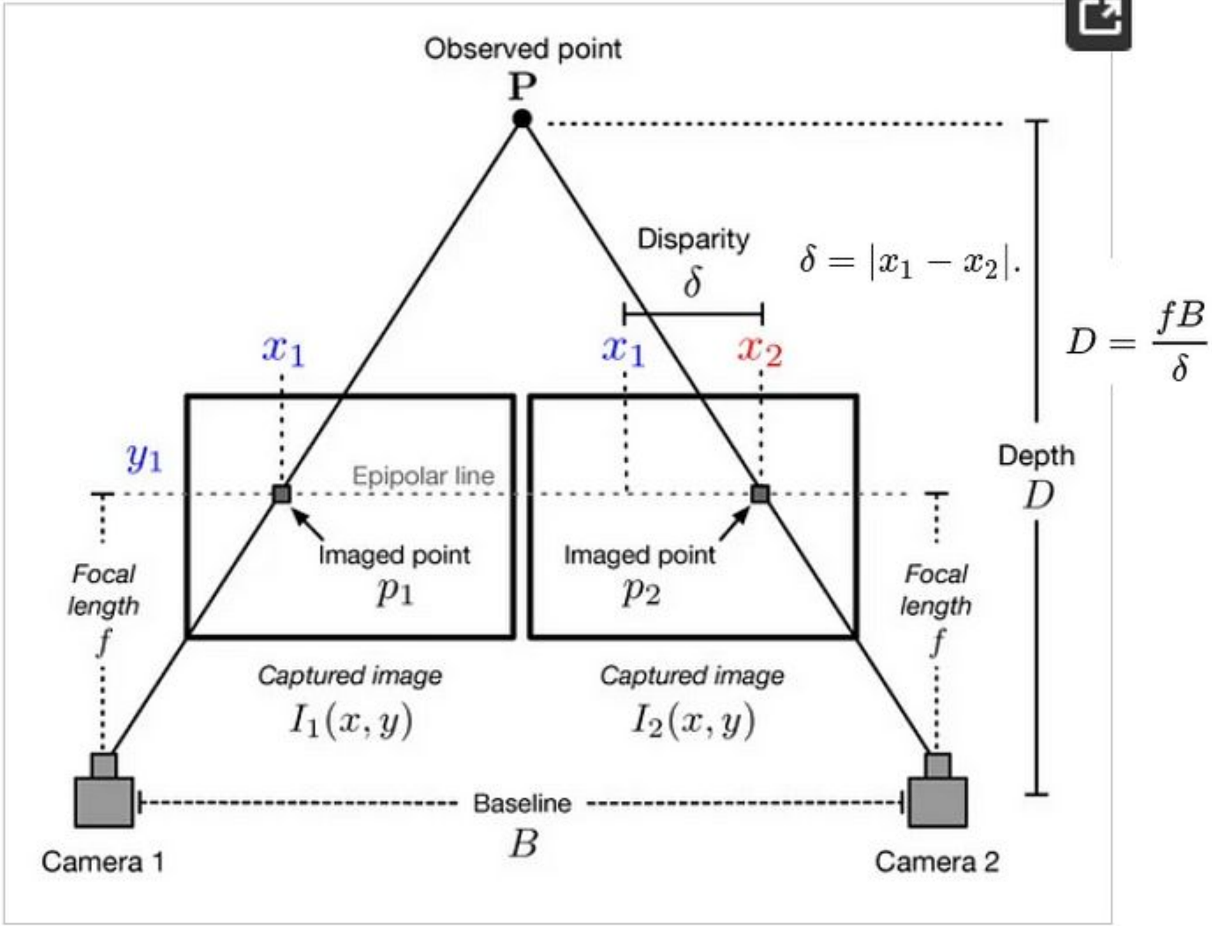
- Camera centers are at same height and focal lengths are the same.

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

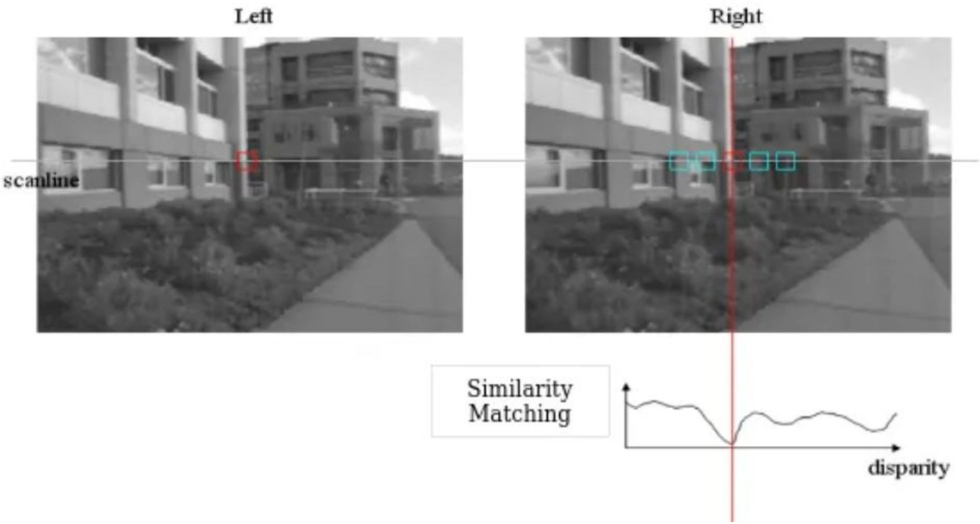


$\mathrm{R} = \mathrm{I}, \mathbf{t} = (T, 0\ 0)^{\mathrm{T}}$


Points from this line project to a
horizontal line in right image
x
x′
Epipolar line
left image plane
right image plane
C
C′
left camera center
right camera center



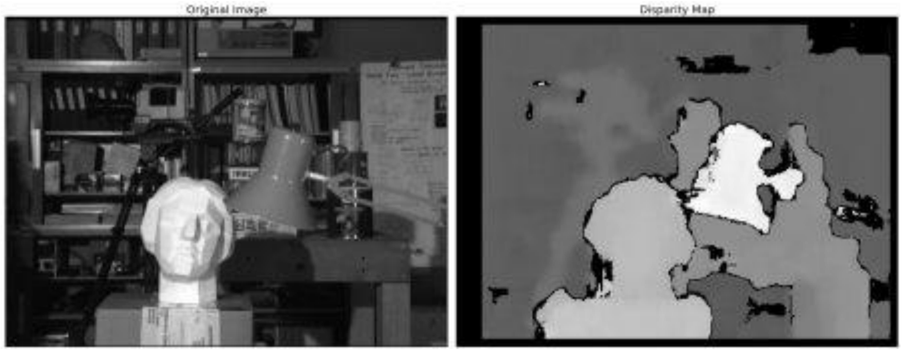
Relationship between Depth Estimation and Disparity in case of Stereo Image setup



Similarity Measure	Formula
Sum of Absolute Differences (SAD)	$\sum_{(i,j)\in W} \lvert I_1(i,j) - I_2(x+i, y+j)\rvert$
Sum of Squared Differences (SSD)	$\sum_{(i,j)\in W} \left(I_1(i,j) - I_2(x+i, y+j)\right)^2$

Disparity Map

The farther the object, the lesser the disparity.



References

https://www.youtube.com/watch?v=EkYXjmiolBg&list=PL2zRqk16wsdp8KbDfHKvPYNGF2L-zQASc&index=5

https://www.youtube.com/watch?v=D9rAOAL12SY&list=PL2zRqk16wsdp8KbDfHKvPYNGF2L-zQASc&index=6