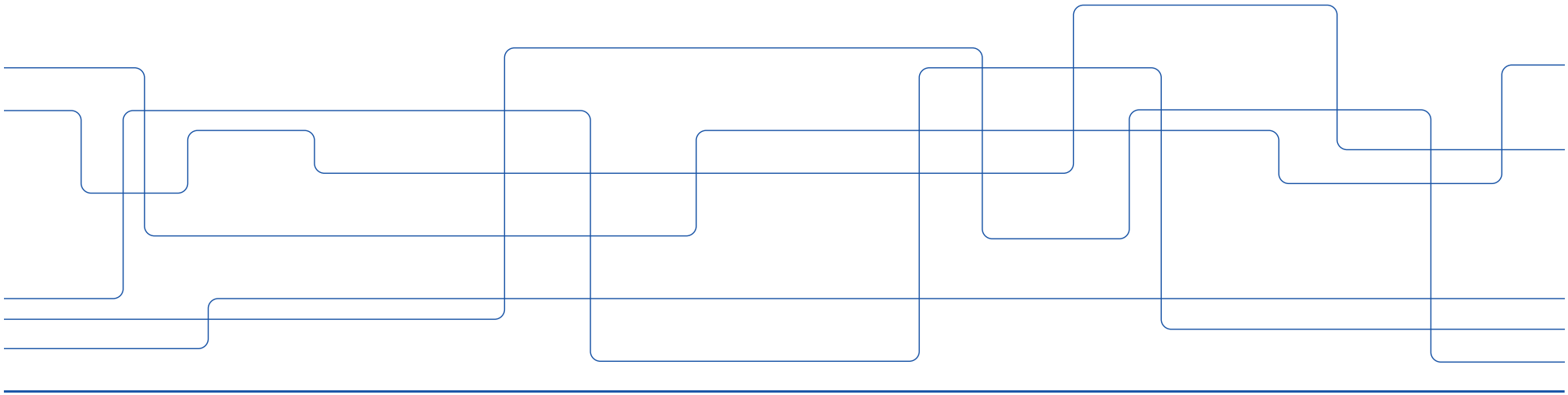
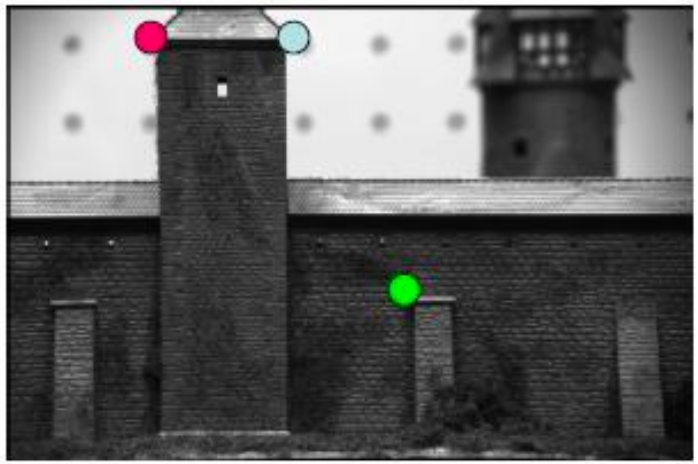

# Stereo geometry

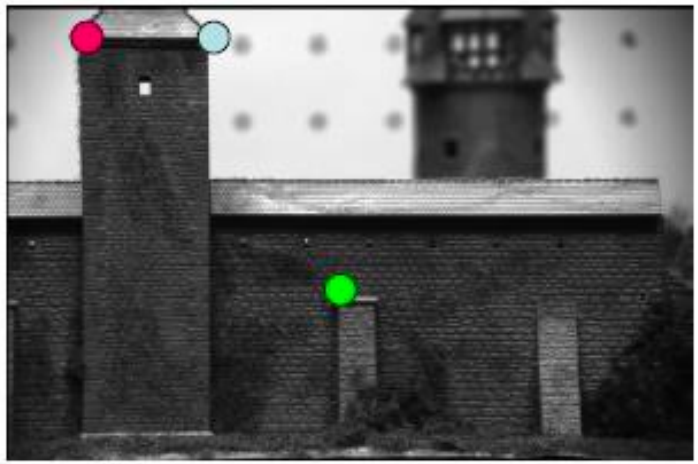
Mårten Björkman



# Depth from stereo

- Inferring depth from images taken at the same time by two or more cameras by using the differences between object's positions.
- If corresponding points can be identified in the left and right images, depth can be computed by triangulation.



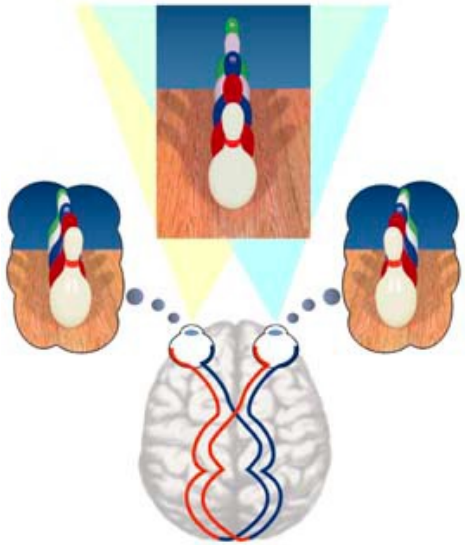


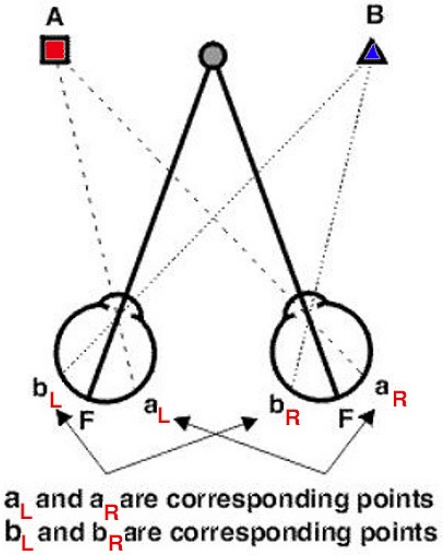
# Simple test of your stereo vision

- Hold a pencil in each hand, at about a relaxed arms length, place the ends toward each other separated about 10cm.
- Try to bring the pencils together:
  1. with one eye shut
  2. using both eyes
- About 5-10 % of human beings are either stereo deficient or have no stereo perception. They still have depth perception though, since depth from focus is a stronger cue.

# Stereopsis in biological vision

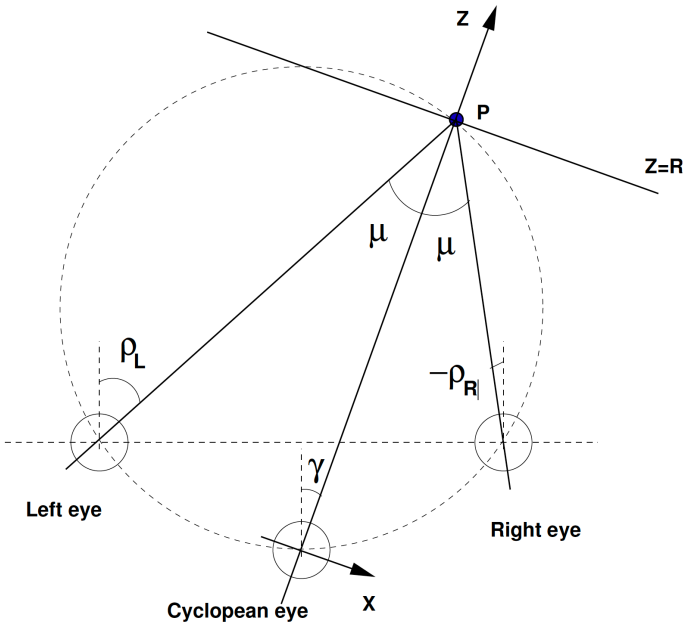
- Fusion: Images from left and right eye are merged into a unified percept, but only corresponding points. Other points give rise to ”double vision”.
- Corresponding retinal points are points stimulated on the retina that give rise to the same visual direction. These are located on a plane in 3D, the horopter.





# Stereo geometry: Verging cameras

- The principal rays of the two cameras converge at a point – the fixation point.
- In the plane defined by the fixation point and the centres of projection of the cameras, we have:




Vergence – angle between the principal directions.

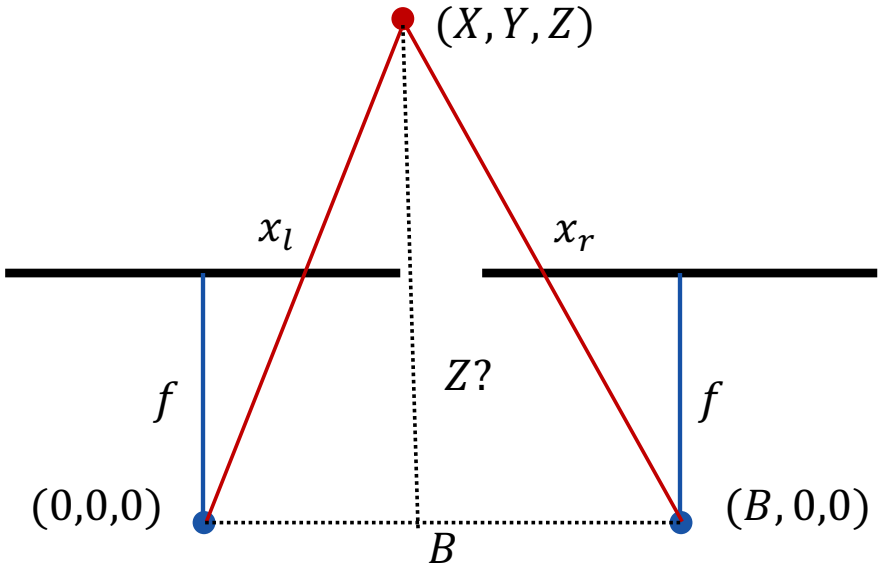
$$2\mu = \rho_L - \rho_R$$

Gaze (version) – angle between primary direction and the ray from the cyclopean eye to the fixation point.

$$\gamma = \frac{1}{2}(\rho_L + \rho_R)$$

# Triangulation – parallel cameras

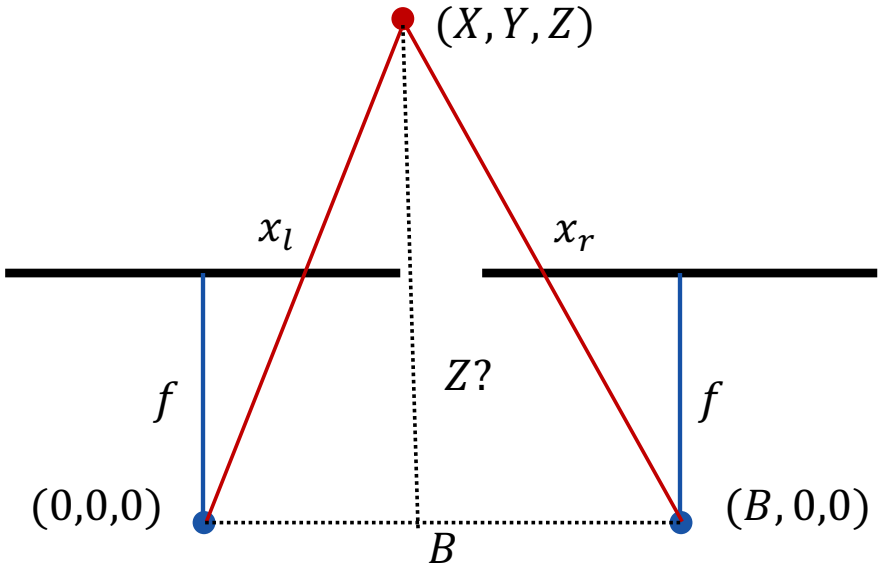
- Assume we have two parallel cameras with <u>baseline</u> $b$ and two matching points $(x_l, y)$ and $(x_r, y)$ in respective cameras?
- What is the depth $Z$ of the point in 3D and what are the coordinates $(X, Y, Z)$?



# Triangulation – parallel cameras

- For the left camera we have the projection:

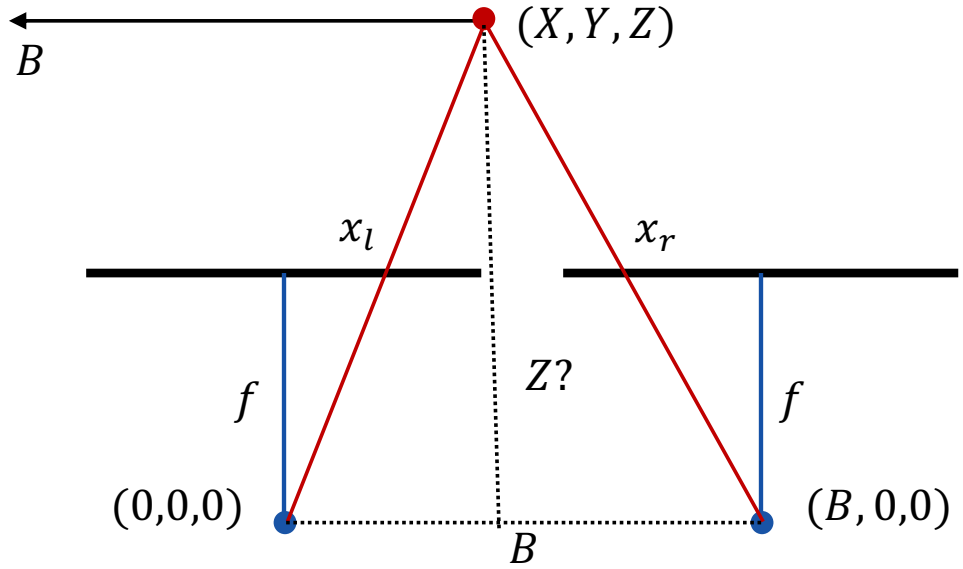
$$x_l = f\frac{X}{Z}, \qquad y_l = f\frac{Y}{Z}$$



# Triangulation – parallel cameras

- Moving the coordinate system to the right is equivalent to moving the 3D point to the left. For the right camera we have the projection:

$$x_r = f\frac{X - B}{Z}, \qquad y_r = f\frac{Y}{Z}$$



# Triangulation – parallel cameras

- We can summarize the two projections:

$$x_l = f\frac{X}{Z}, \qquad y_l = f\frac{Y}{Z}$$
$$x_r = f\frac{X-B}{Z}, \qquad y_r = f\frac{Y}{Z}$$

- The difference in x-coordinates is the horizontal disparity:

$$d = x_l - x_r = f\frac{X}{Z} - f\frac{X-B}{Z} = f\frac{B}{Z}$$

- Assuming that the baseline $B$ and focal length $f$ are known, we can compute the depth from the disparity:

$$Z = \frac{Bf}{d}$$

- After that $X$ and $Y$ coordinates are then trivial to compute.

# Trade-off for wide baselines

- What much do errors in disparity $d$ propagate to errors in depth $Z$?
- Differentiate

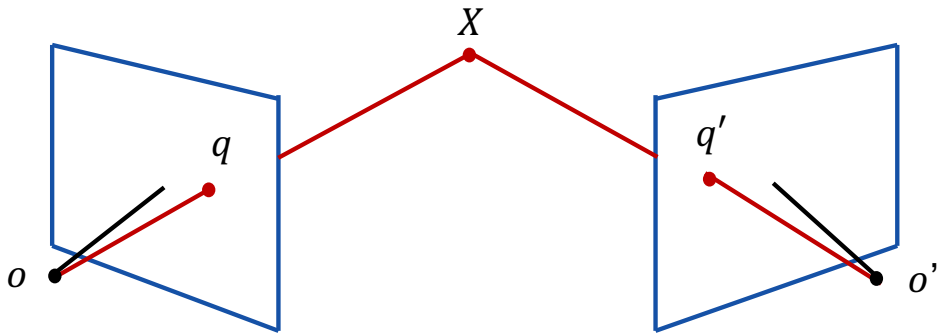
$$Z = \frac{Bf}{d}$$

$$\frac{\delta Z}{\delta d} = -\frac{Bf}{d^2} = -\frac{Z^2}{Bf}$$

- Conclusion: for a given error in disparity, error in depth increases quadratically as the depth increases.
- How to reduce the errors:
  - Increase baseline $B$ – difficult since it might make matching harder.
  - Increase focal length $f$ – then field of view decreases and less is seen of the scene.

# Triangulation – general case

- Assume we have two matching points $\mathrm{q} = (x, y)$ and $\mathrm{q}' = (x', y')^T$ in two different cameras with projection matrices $P$ and $P'$. Where is the point $X$ in 3D space?



# Triangulation – general case

- If we write the projection matrix $P$ in terms of three rows, we get two equations per projection. For the first camera:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \simeq PX = \begin{pmatrix} p_1^T \\ p_2^T \\ p_3^T \end{pmatrix} X \Longrightarrow \begin{cases} x = p_1^T X / p_3^T X \\ y = p_2^T X / p_3^T X \end{cases} \Longrightarrow \begin{cases} x(p_3^T X) - (p_1^T X) = 0 \\ y(p_3^T X) - (p_2^T X) = 0 \end{cases}$$

  and for the second camera:

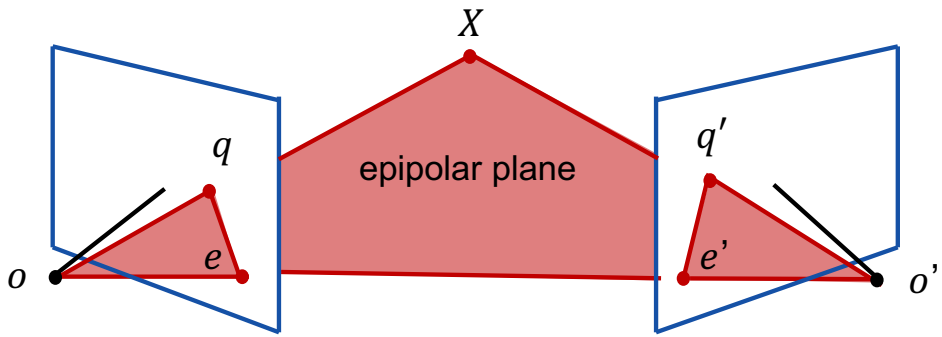
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \simeq P'X = \begin{pmatrix} p'^T_1 \\ p'^T_2 \\ p'^T_3 \end{pmatrix} X \Longrightarrow \begin{cases} x' = p'^T_1 X / p'^T_3 X \\ y' = p'^T_2 X / p'^T_3 X \end{cases} \Longrightarrow \begin{cases} x'(p'^T_3 X) - (p'^T_1 X) = 0 \\ y'(p'^T_3 X) - (p'^T_2 X) = 0 \end{cases}$$

- We can combine these four equations and find the least square estimate of $X$:

$$\hat{X} = \underset{X}{\operatorname{argmin}} \left\| \begin{pmatrix} xp_3^T - p_1^T \\ yp_3^T - p_2^T \\ x'p'^T_3 - p'^T_1 \\ y'p'^T_3 - p'^T_2 \end{pmatrix} X \right\|^2$$

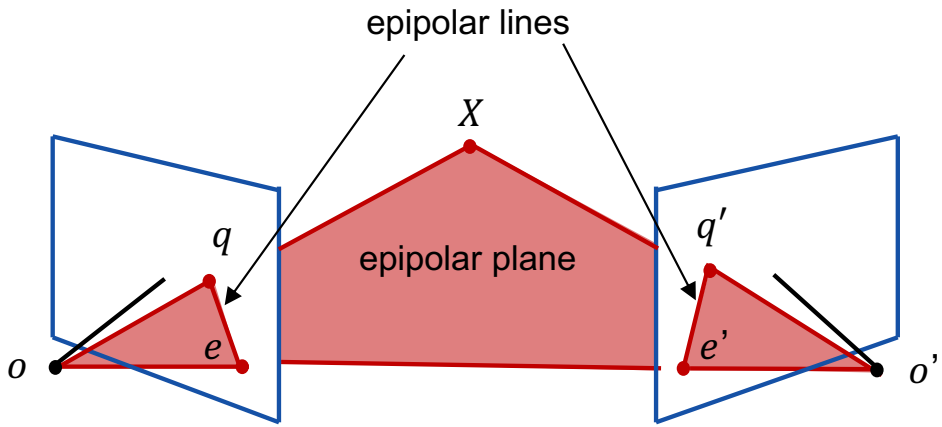
# The epipolar plane

- The optical centres $o$ and $o'$, and 3D point $X$ define an epipolar plane.



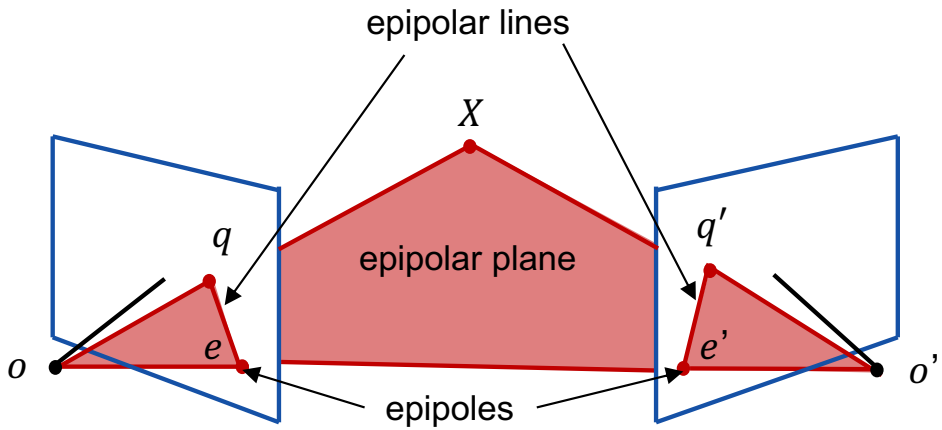
# The epipolar plane

- The optical centres $o$ and $o$', and 3D point $X$ define an epipolar plane.
- This plane intersect the image planes along two lines, the epipolar lines



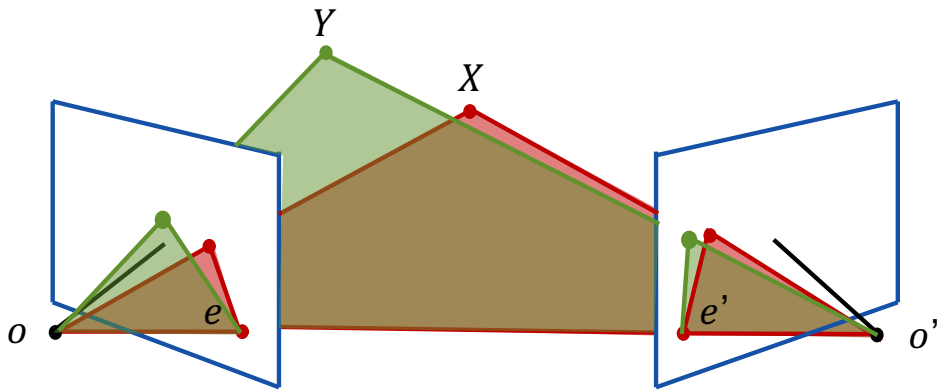
# The epipolar plane

- The optical centres $o$ and $o$', and 3D point $X$ define an epipolar plane.
- This plane intersect the image planes along two lines, the epipolar lines.
- The line between $o$ and $o$' intersect the image planes in two epipoles, $e$ and $e$'.



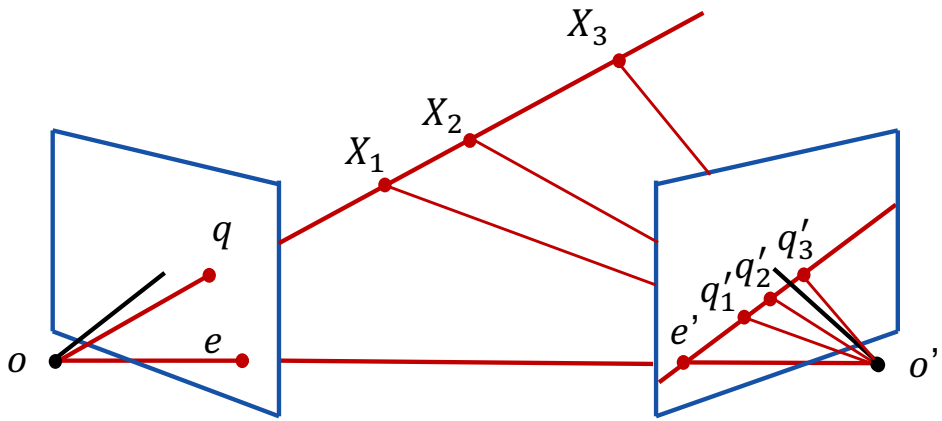
# The epipolar plane

- If you have another 3D point Y, you get a new epipolar plane.
- But the epipoles $e$ and $e'$ remain the same.
- All epipolar lines will intersect the respective epipoles.



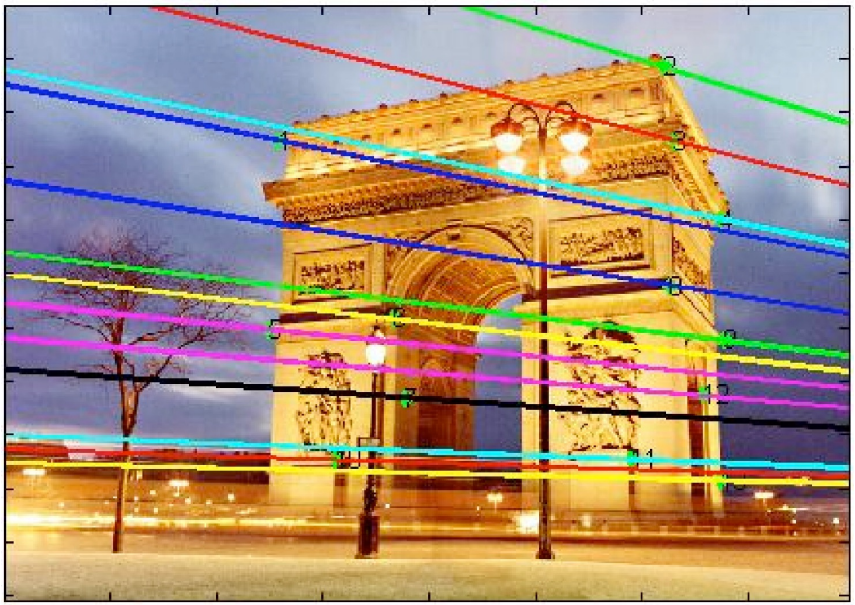
# Epipolar geometry

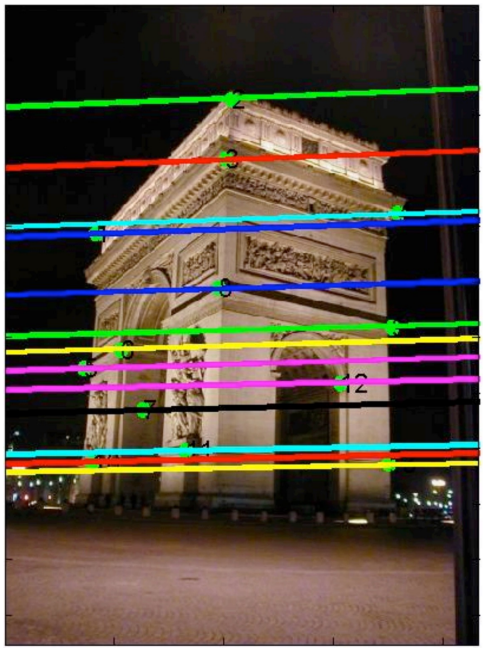
- What if you only have a point $q$ in one image, where is then the match $q$'?
- The match $q$' can be along the epipolar line, but how do we find this line?



# Epipolar lines example

Matching points can be found along epipolar lines





# Remember from linear algebra

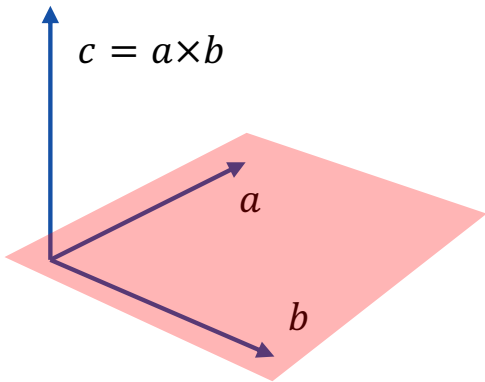
- The cross product of two vectors $a$ and $b$ is a new vector $c = a \times b$, perpendicular to both $a$ and $b$, i.e. $c^T a = c \cdot a = 0$ and $c^T b = c \cdot b = 0$.
- If $a$ and $b$ are two non-parallel lines on a plane, $c$ will be the normal of that plane.

Expressed in terms of coordinates:

$$a \times b = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \times \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = \begin{pmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{pmatrix}$$

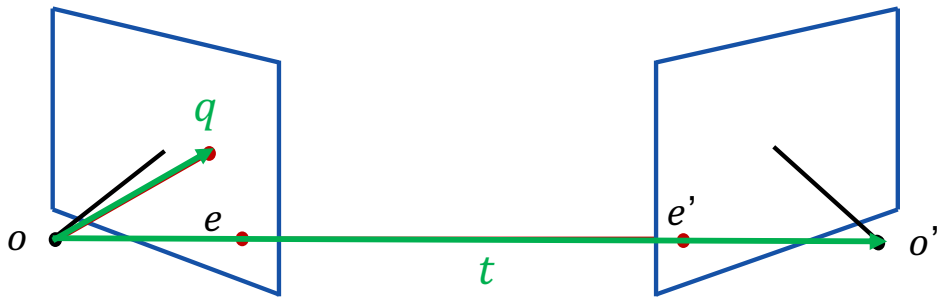
Sometimes a more convenient form:

$$a \times b = [a]_\times b = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix}$$



# Epipolar geometry

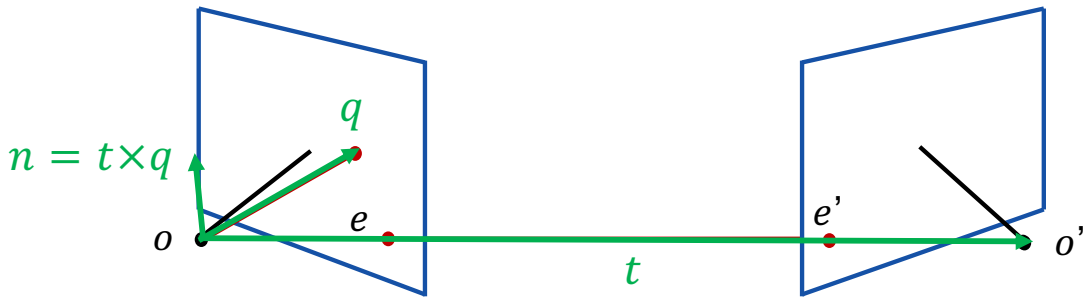
- Assume we know the relative translation $t$ and rotation $R$ between the cameras.
- The translation $t$ is the baseline between the optical centres $o$ and $o$'.
- In the coordinate system of the left camera, $q$ can be viewed as a 3D vector in homogeneous coordinates.
- Note: the two vectors $q$ and $t$ must lie on the same plane, the epipolar plane.



# Epipolar geometry

- The normal of the epipolar plane must then be given by

$$n = t \times q = [t]_{\times} q$$



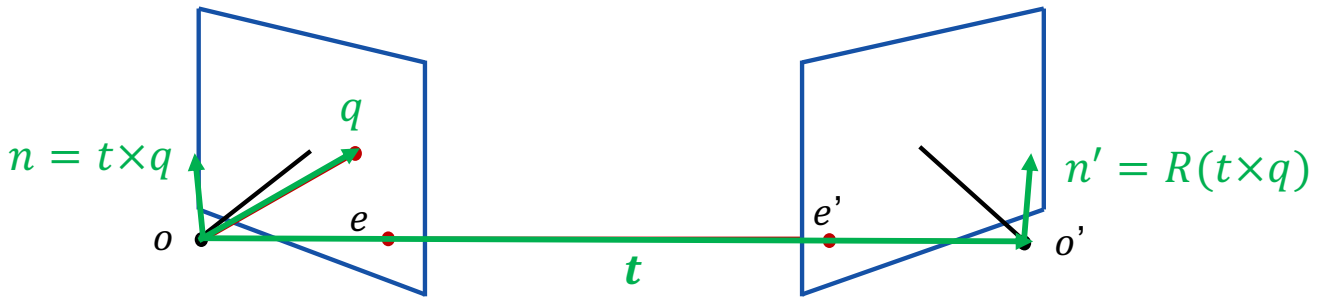
# Epipolar geometry

- The normal of the epipolar plane must then be given by

$$n = t \times q = [t]_\times q$$

- Expressed in the coordinate system of the right camera, the normal must be

$$n' = R(t \times q) = R[t]_\times q$$



# Epipolar geometry

- The normal of the epipolar plane must then be given by

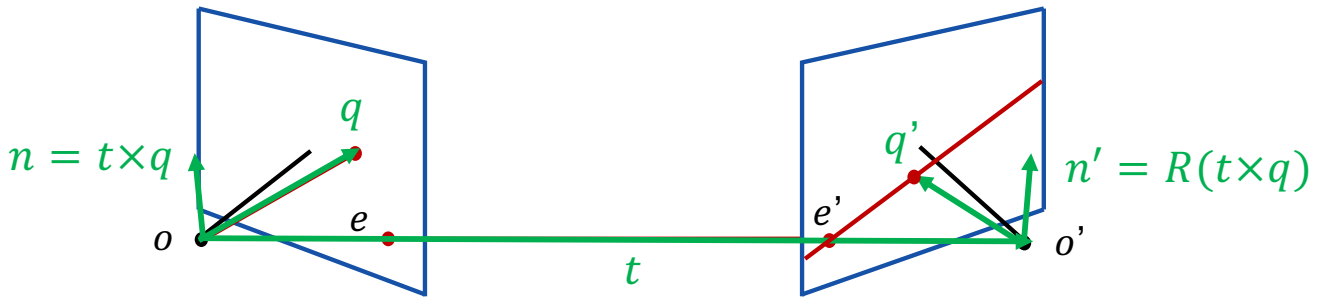
$$n = t \times q = [t]_\times q$$

- Expressed in the coordinate system of the right camera, the normal must be

$$n' = R(t \times q) = R[t]_\times q$$

- The corresponding point $q'$ must also lie on the epipolar plane, thus

$$q'^T n' = q'^T R(t \times q) = q'^T R[t]_\times q = 0$$



# The Essential Matrix

- The rotation and translation can be combined into the essential matrix

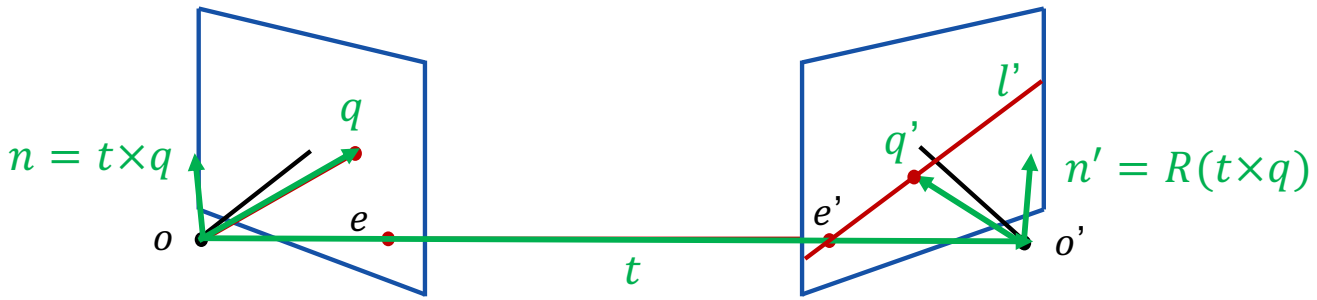
$$E = R[t]_\times$$

- This leads to a constraint called the epipolar constraint

$$q'^T E q = 0$$

- If $q$' lies the right image plane we have the line equation of the epipolar line

$$q'^T l' = 0, \quad \text{where} \quad l' = Eq$$




# The Fundamental Matrix

- So far we have assumed that we have used calibrated image coordinates $q$, with focal length $f = 1$. What if we instead use pixel coordinates $x$ directly?
- We can use the camera matrix $K$ to relate the two kinds of coordinates:

$$x = Kq \Longrightarrow q = K^{-1}x$$

- Then we can write the epipolar constraint as:

$$q'^{T}R[t]_{\times}q = 0 \Longrightarrow \mathrm{x}'^{T}\mathrm{K}'^{-T}R[t]_{\times}K^{-1}x = 0$$

- The fundamental matrix

$$\mathrm{F} = \mathrm{K}'^{-T}R[t]_{\times}K^{-1}$$

can then be used to write the epipolar constraint as:

$$\mathrm{x}'^{T}Fx = 0$$


# Determining the Essential Matrix

- Use features (e.g. SIFT) matched between the two images.
- The essential matrix depends on
  - Rotation $R$ (3 parameters)
  - Translation $t$ (3 parameters)
- The matrix is, however, homogeneous in the components of $t$.
  $\Longrightarrow$ Totally 5 unknowns.
- Each correspondence gives one constraint $\Longrightarrow$ 5 matches needed.

- Common methods:
  - Hartley's Normalized 8-point method – needs 8 matches, but can be solved linearly.
  - Nistér's 5-point method – requires solving a 10 degree polynomial equation.

# Bundle adjustment: multiple cameras

- Assume you have $K$ camera images with unknown projection matrices $\mathrm{P}_k$ and projections $q_{ik}$ of $N$ unknown 3D points $Q_i$.
- Then you can set up a large system of equations and search for $\mathrm{P}_k$ and $Q_i$ by minimizing

$$\min_{\{P_k, Q_i\}} \sum_{k=1}^{K} \sum_{i=1}^{N} d(q_{ik}, f(P_k, Q_i))$$

where

$$d\big(q_{ik}, f(P_k, Q_i)\big) = \left(x_{ik} - \frac{p_k^1 Q_i}{p_k^3 Q_i}\right)^2 + \left(y_{ik} - \frac{p_k^2 Q_i}{p_k^3 Q_i}\right)^2$$

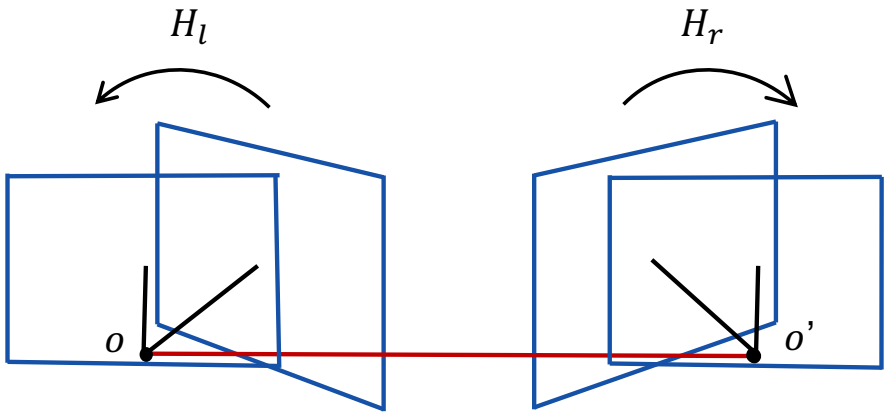
- After initialization by first computing the essential matrices between pairs of camera images, this can be solved iteratively.

# Bundle adjustment: example

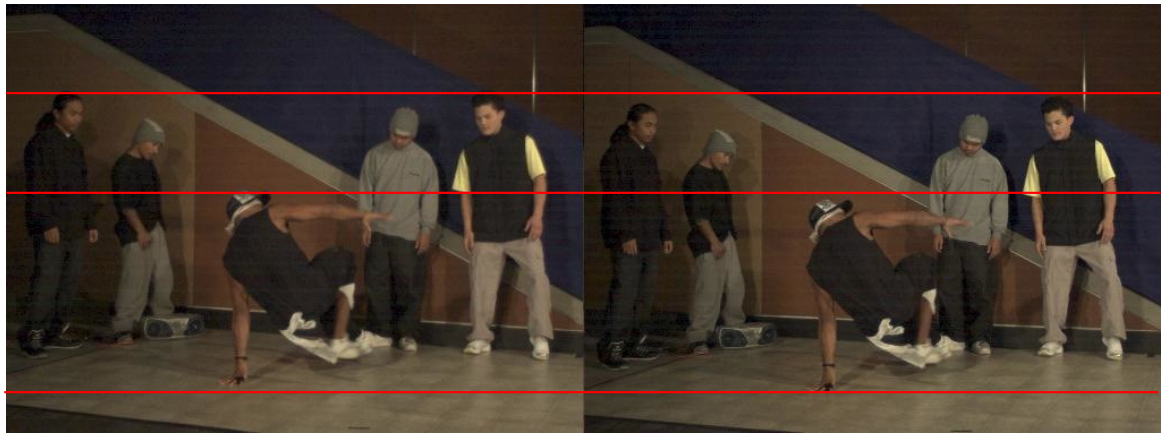
- Feature extraction, matching and bundle adjustment using `ColMap`.

# Stereo rectification

- Matching is much easier if cameras are parallel and the epipolar lines are parallel to the $x$-axis.
- Solution: apply a homography to rotate each image, to get two images, as if the cameras were parallel.



# Stereo rectification example



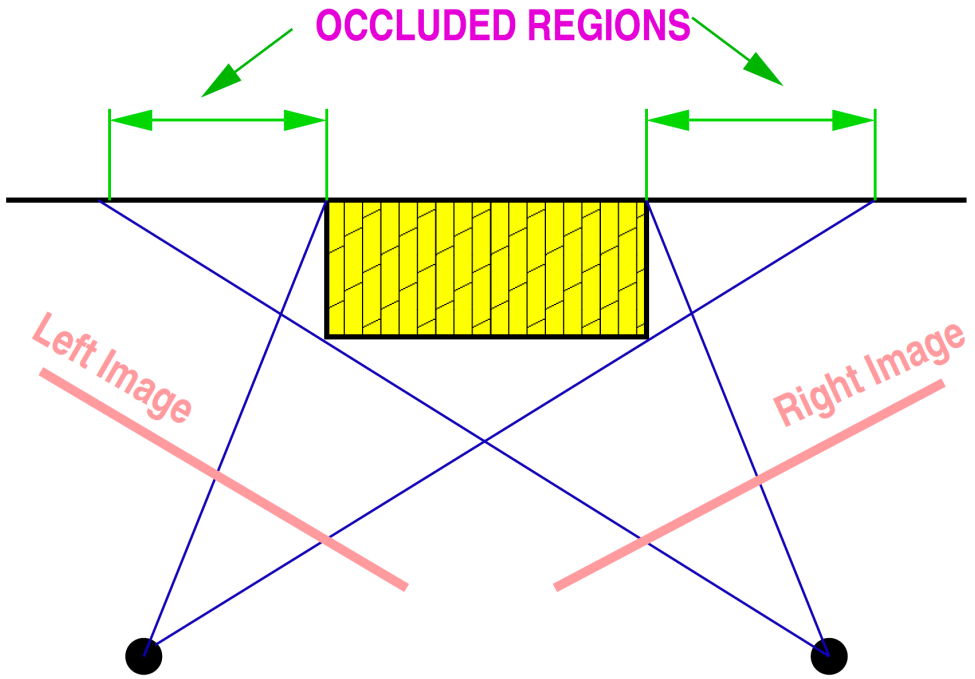
Unrectified: the $y$-positions are not aligned.

Rectified: the $y$-positions are better aligned.

# Establishing correspondence – Problems

Non-trivial problem, because of several reasons.

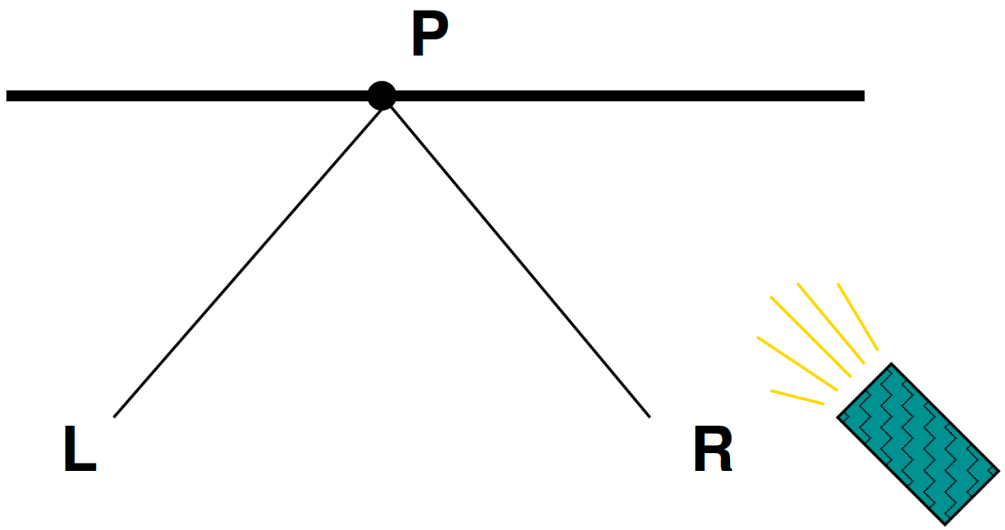
- Occlusions: some parts of scene are only seen by one camera.



# Establishing correspondence – Problems

- Brightness variations: in general, cameras often have different orientations relative to the source of illumination.

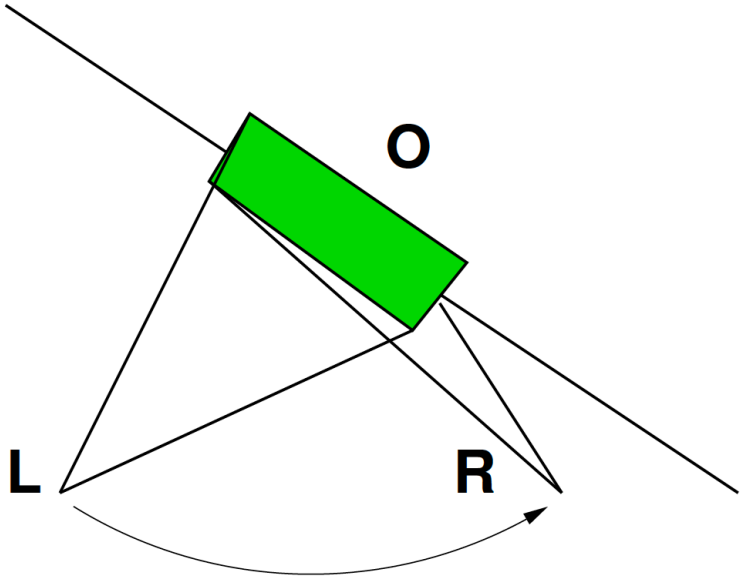
  $\Longrightarrow$ Brightness will not be the same in the two regions.




P appears brighter to L than to R.

# Establishing correspondence – Problems

- Distortions due to perspective effects.

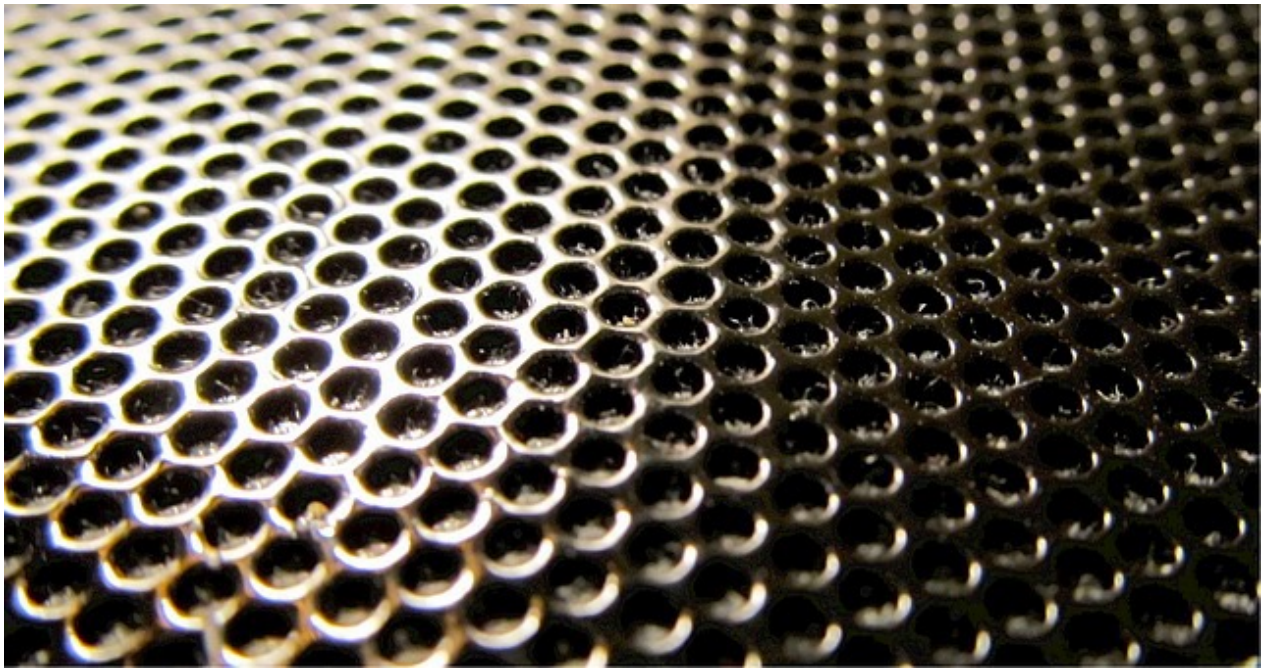



foreshortening

O appears to be larger to L than R.

# Establishing correspondence – Problems

- Repetitive textures ⟹ ambiguous matches.

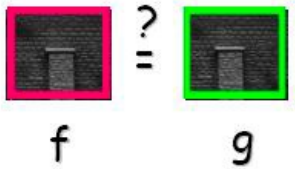


Lack of unique surface markings

# Basic method: Correlation based disparities

- Just use windows around each point in the image. For each window, find the closest window on the epipolar line in the other image.

Comparing Windows:




$SSD = \sum_{i,j}\big(f(i,j) - g(i,j)\big)^2$ most natural, but not very robust

$SAD = \sum_{i,j} |f(i,j) - g(i,j)|$ more common and more robust

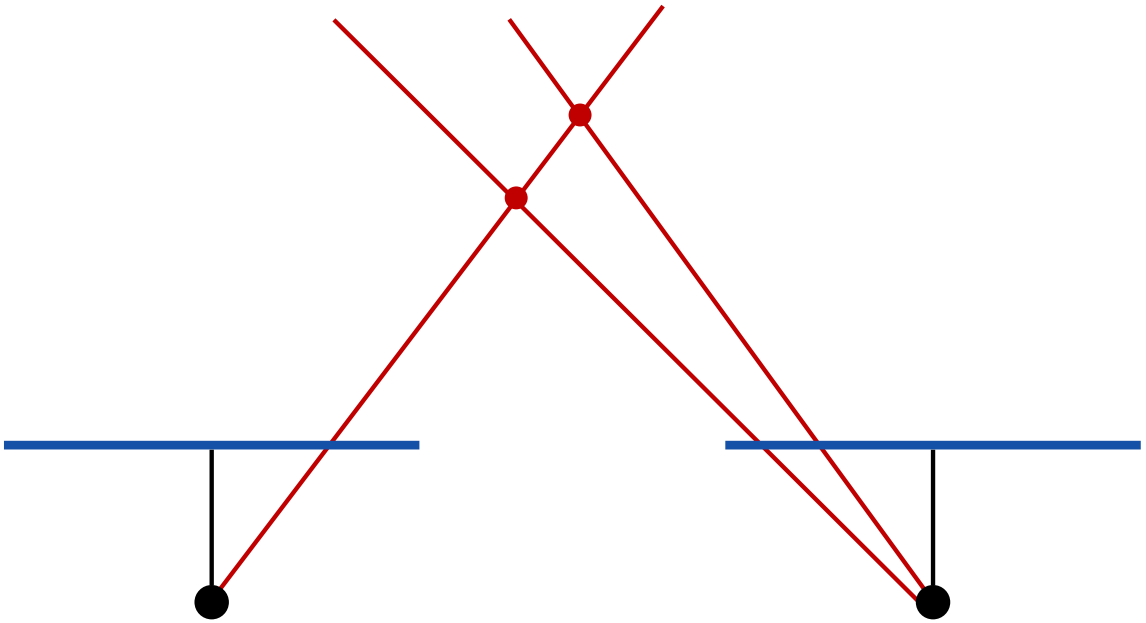
# Results using different window sizes

- If the window is too small, it becomes too sensitive to noise.
- If the window is too large, disparity within the window varies too much.

W = 3 W = 20

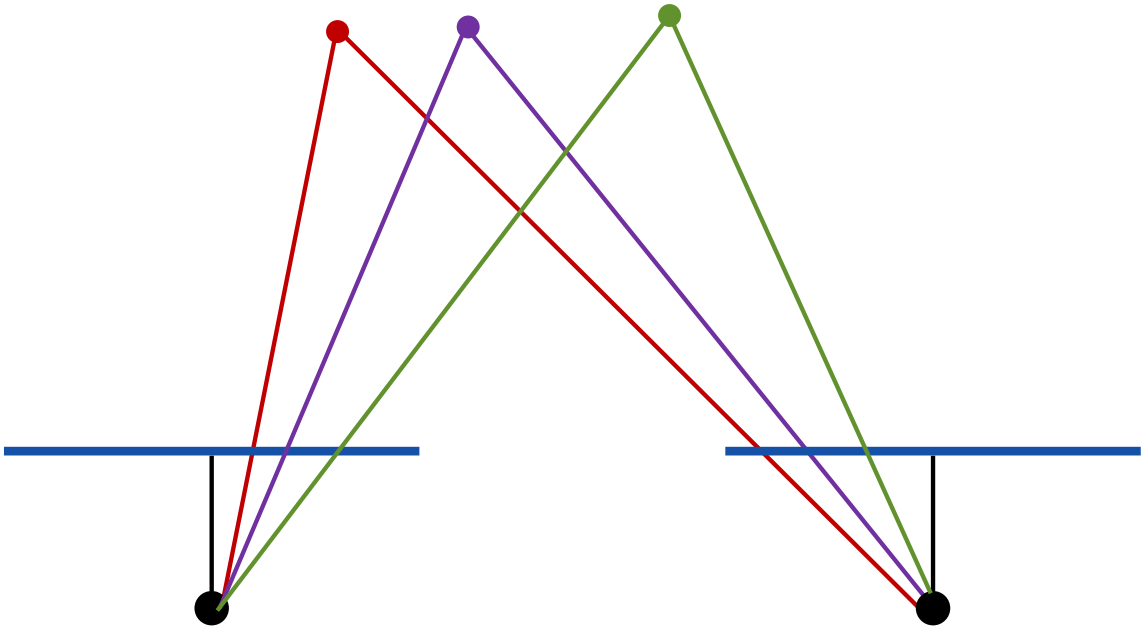
# Constraints – Uniqueness

- Each point should match only one point in the other image.



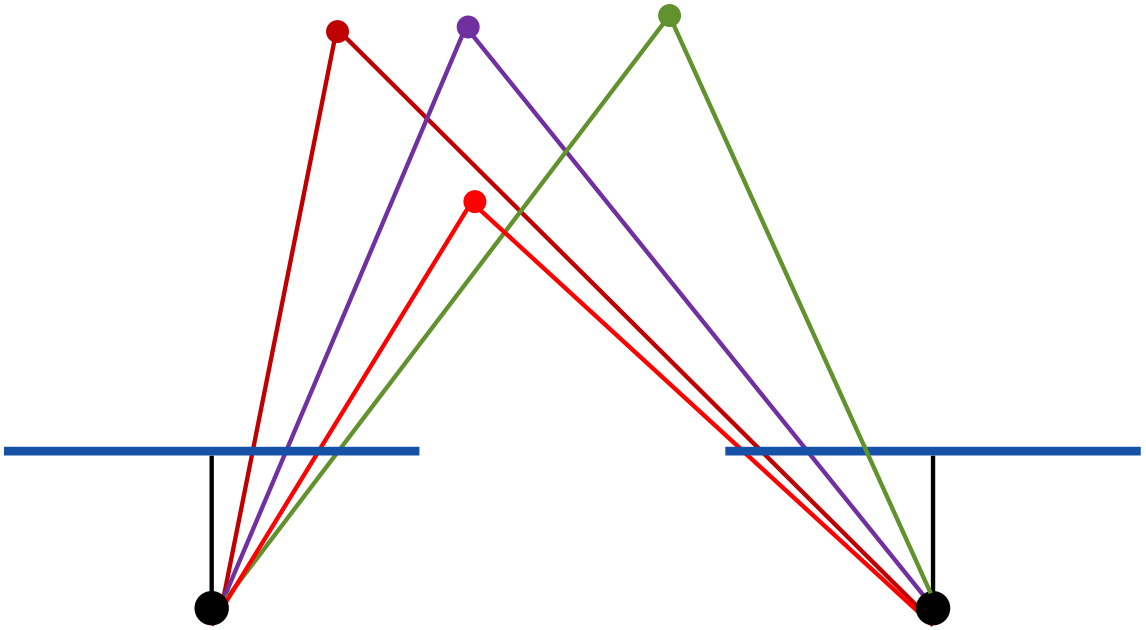
# Constraints – Ordering

- Monotonic ordering constraint: ordering of points is the same in both images.

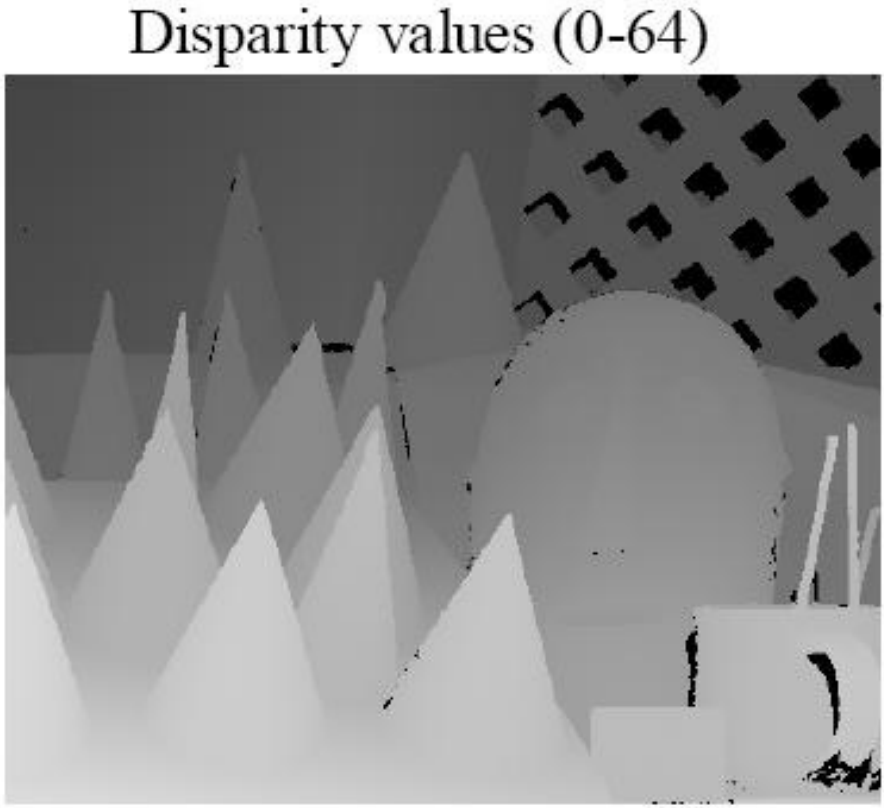


# Constraints – Ordering

- Ordering of points is the same in both images. Not always true!

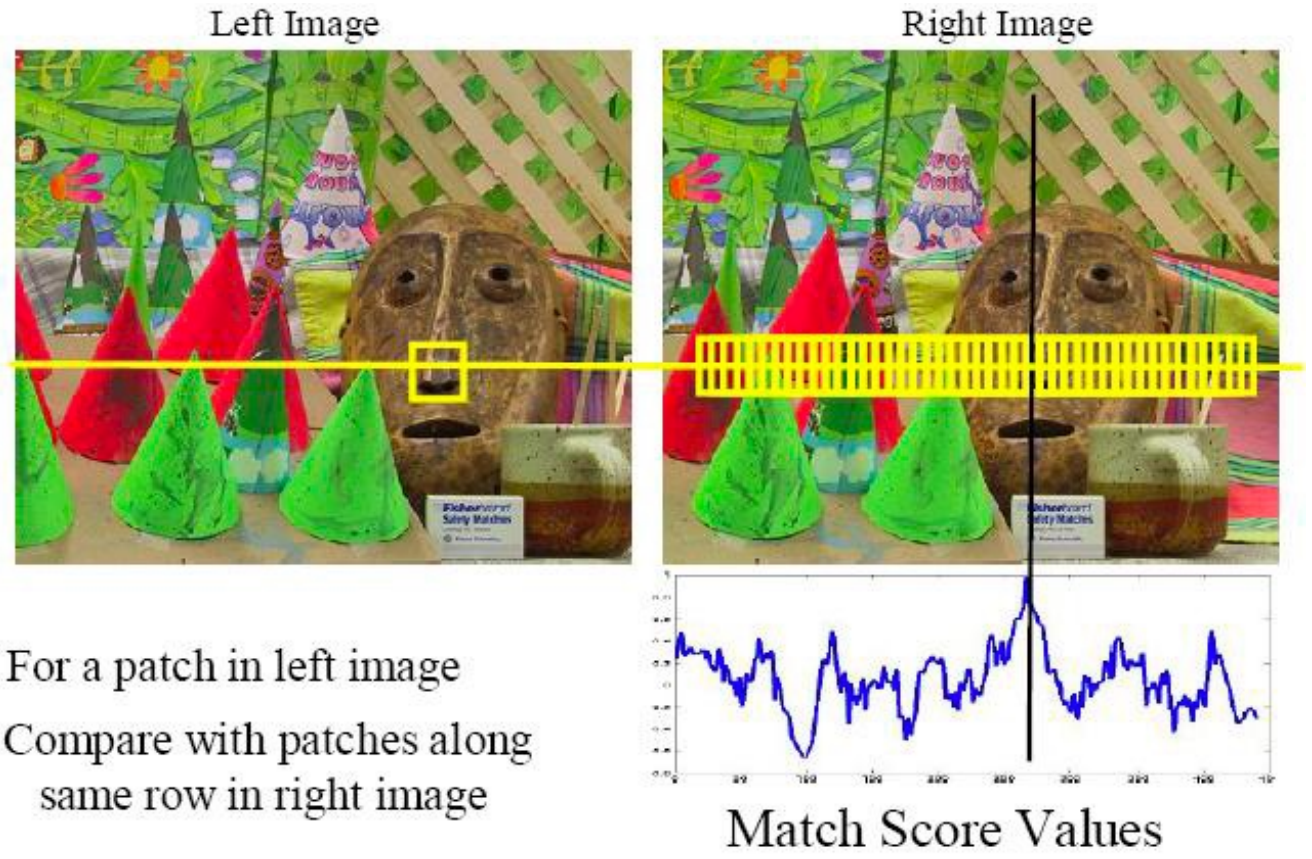


# Example: Disparity map


Common benchmark example – far too easy compared to the real world.

# Example: Disparity map



# Constraints – Smoothness

- Neighbouring pixels usually have the same disparity.
- Better results can be achieved by adding a smoothness cost $E_s(p, d_p)$ to the regular matching cost $E_d(p, d_p)$ for point $p$ with disparity $d_p$:

$$E(p, d_p) = E_d(p, d_p) + \lambda E_s(p, d_p)$$

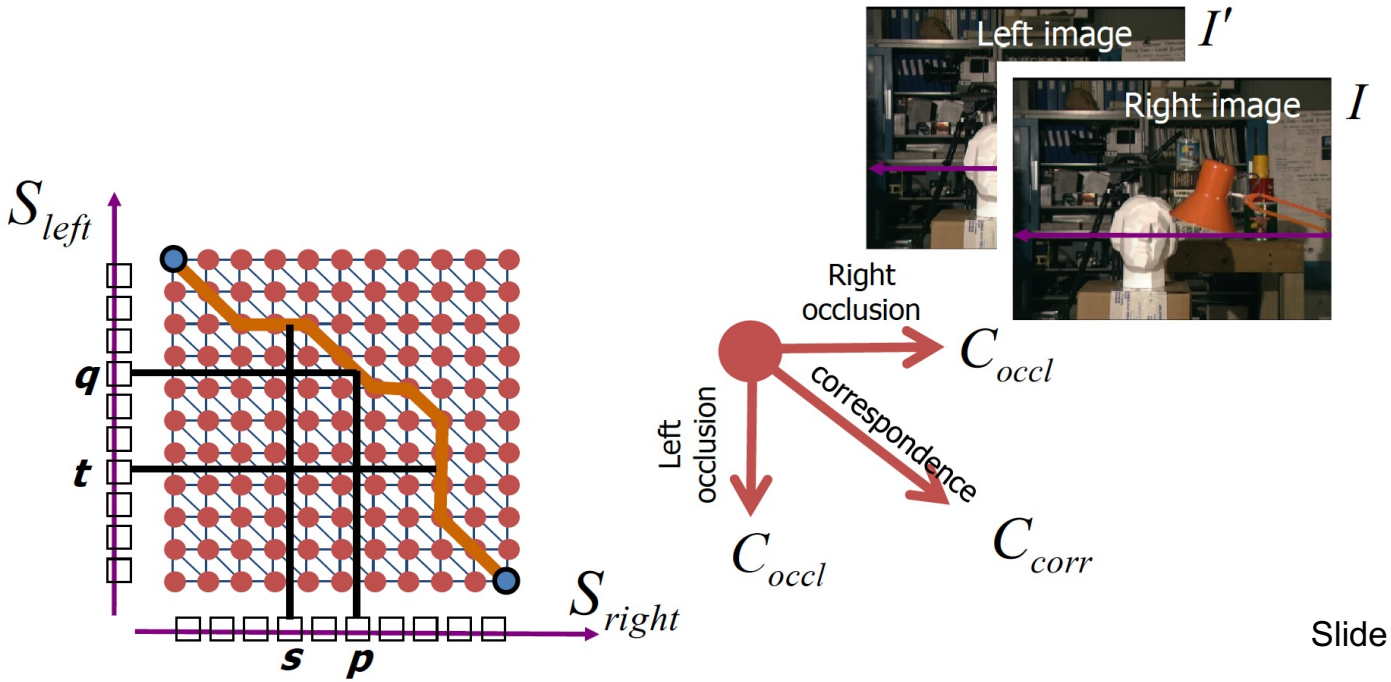
Example: $E_s(p, d_p) = \sum_{q \in N_p} V(d_p, d_q)$

$$V(d_p, d_q) = |\mathrm{d_p} - \mathrm{d_q}| \quad \text{or}$$

$$V(d_p, d_q) = \begin{cases} 0 & \text{if } d_p = d_q \\ 1 & \text{if } d_p \neq d_q \end{cases}$$

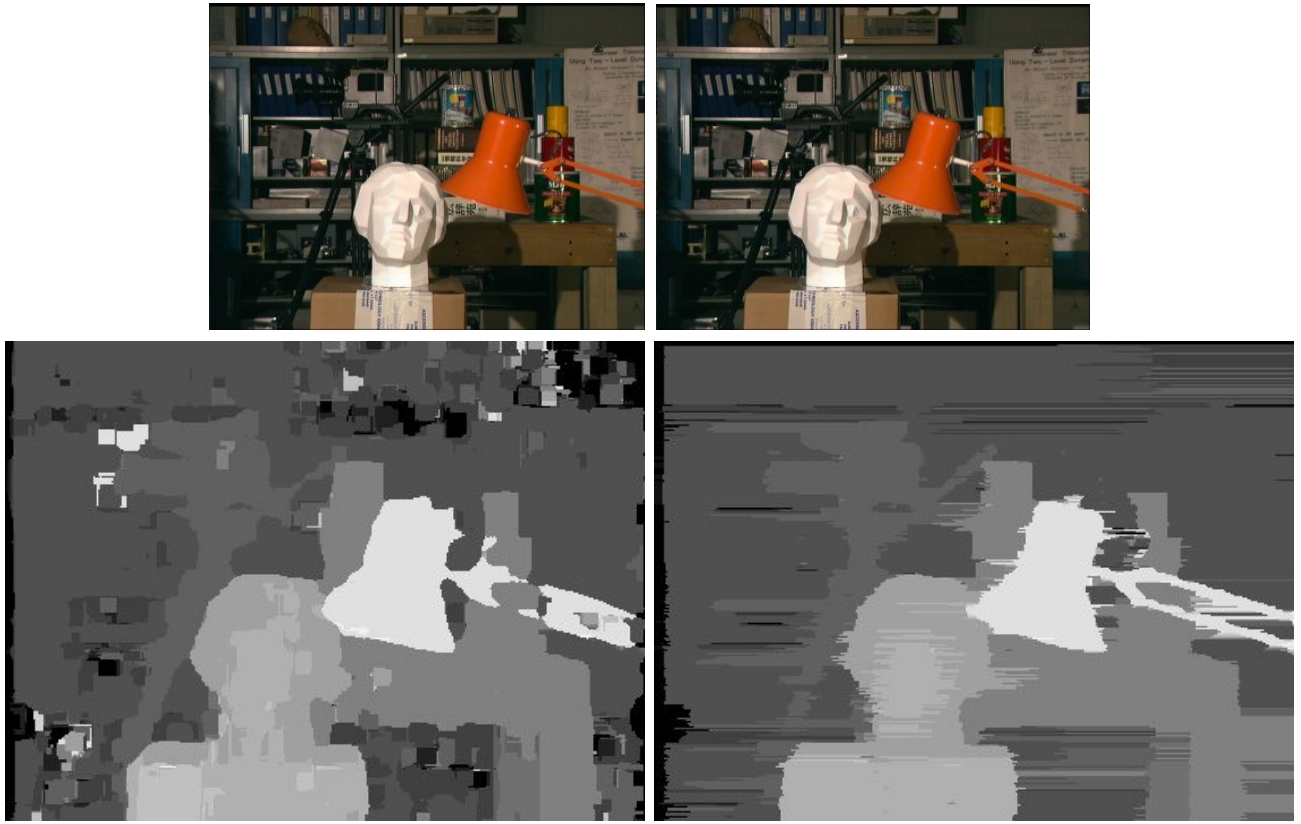
# Shortest path for scanline stereo matching

- Using dynamic programming, the lowest cost combination of disparities can be found for each scanline ($y$-position) by searching for a shortest path.




Slide credit: Y. Boykov

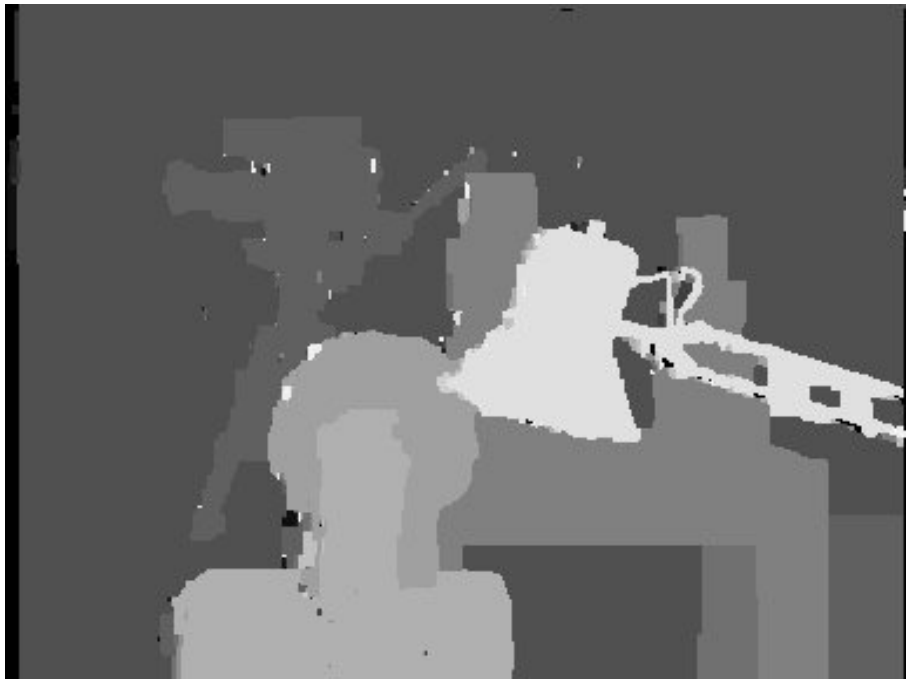
# Stereo results – dynamic programming



Results based on SSD (left) and dynamic programming (right).

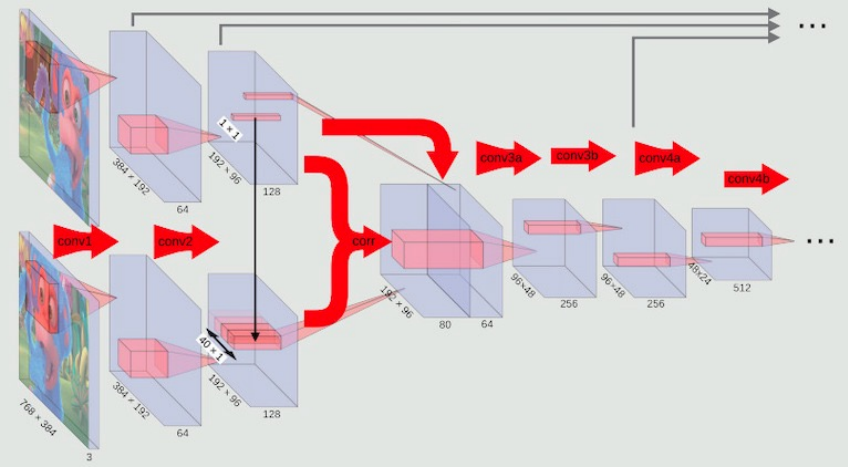
# Stereo results – graph cuts

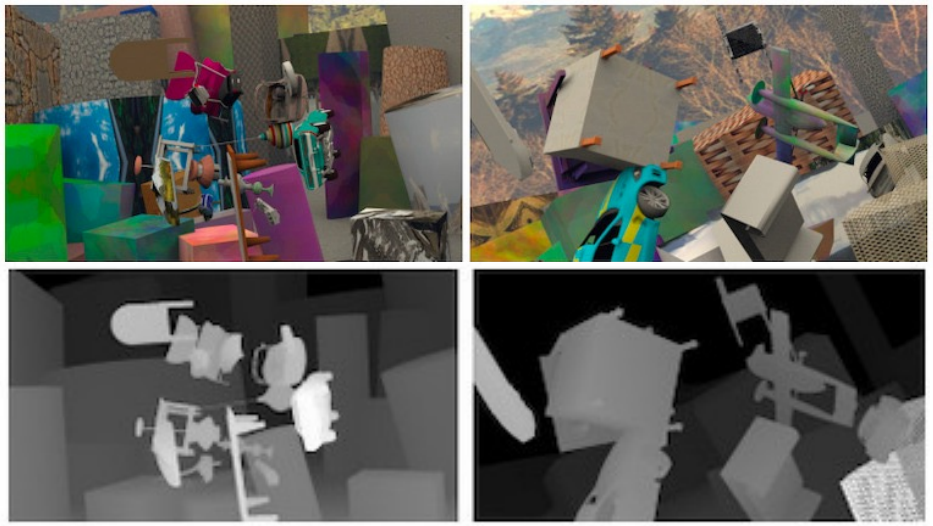
- Better results with a smoothing constraint applied both $x$-wise and $y$-wise.
- Can be done through e.g. energy minimization with graph cuts.



# Stereo matching with deep learning

- Most deep learning methods (e.g. DispNet) also use correlation.
- First a network to extract features, followed by correlation and another network to interpret the correlation scores.

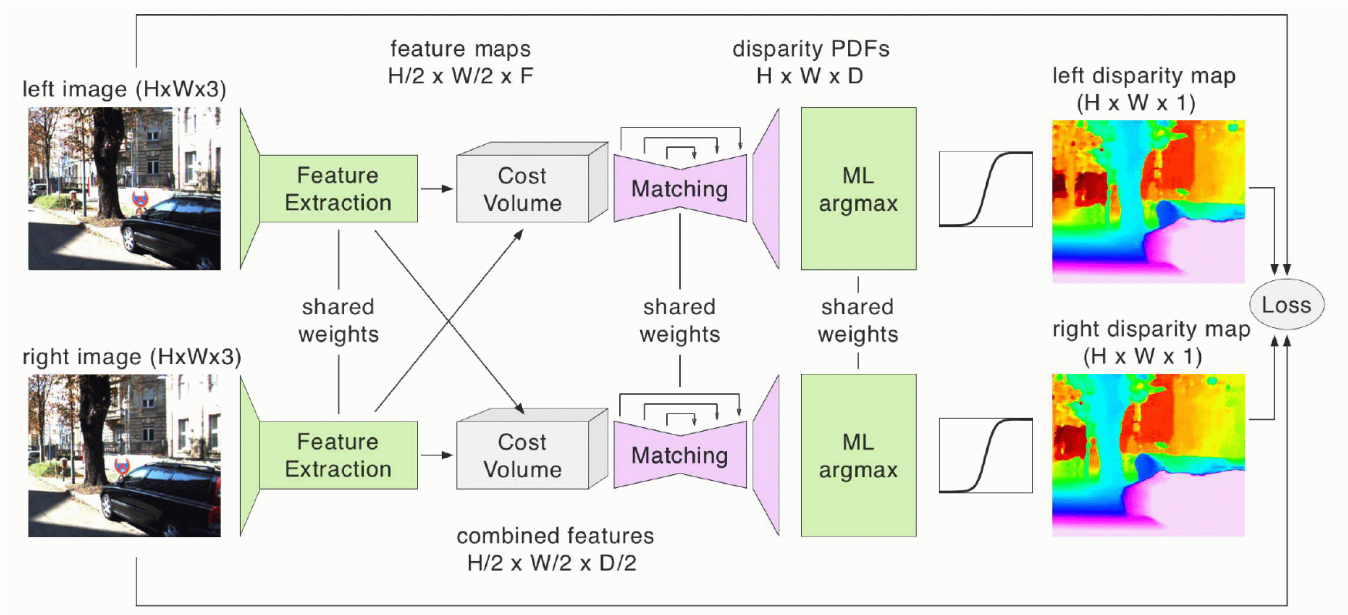


Mayer et al., “A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation”, CVPR, 2016.

# Stereo matching with deep learning

- Rather than supervised training, many novel methods uses self-supervised training that tries to warp one image to the other using the disparities.




Smolyanskiy et al., ”On the Importance of Stereo for Accurate Depth Estimation: An Efficient Semi-Supervised Deep Neural Network Approach”, CVPR 2018.


# Summary of good questions

- How does stereo work in general?
- Why can you get double vision?
- What is triangulation?
- What is the relationship between disparities and depths?
- Why does the error in depth increase for larger distances?
- What are the key concepts of epipolar geometry?
- What is an essential matrix and how is it used?
- What might complicate stereo matching?
- How does a simple stereo matcher work?
- What constraints are often used to improve stereo matchers?
- What parts do a deep network based stereo matcher often contain?


# Recommended reading

- Szeliski: Chapters 12.1 – 12.5