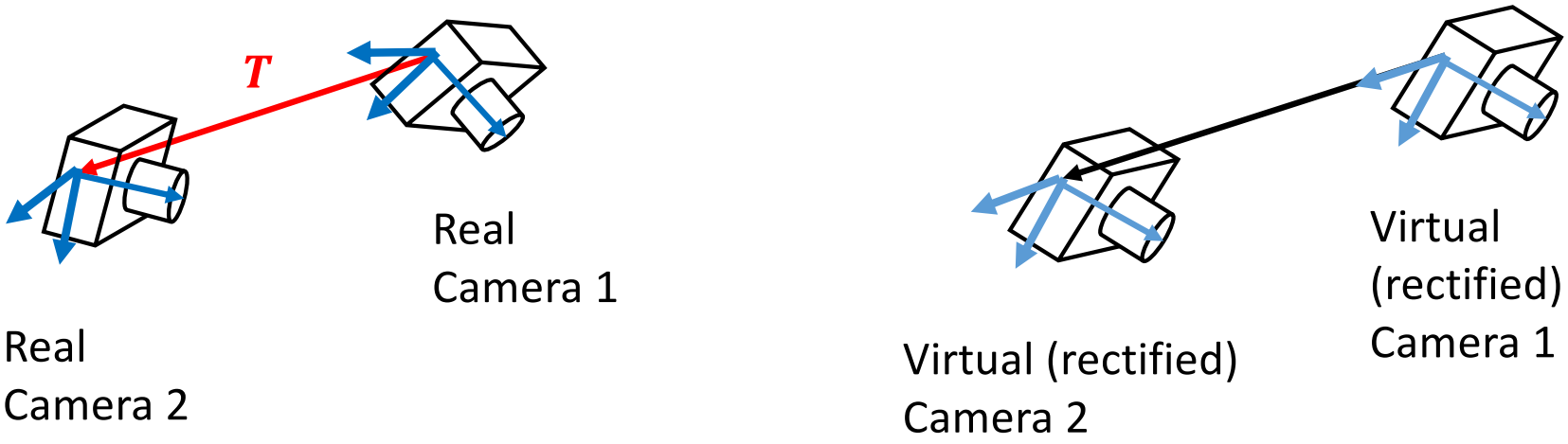
COMP 558

Lecture 22

Binocular stereo disparity estimation

Slides courtesy Mike Langer

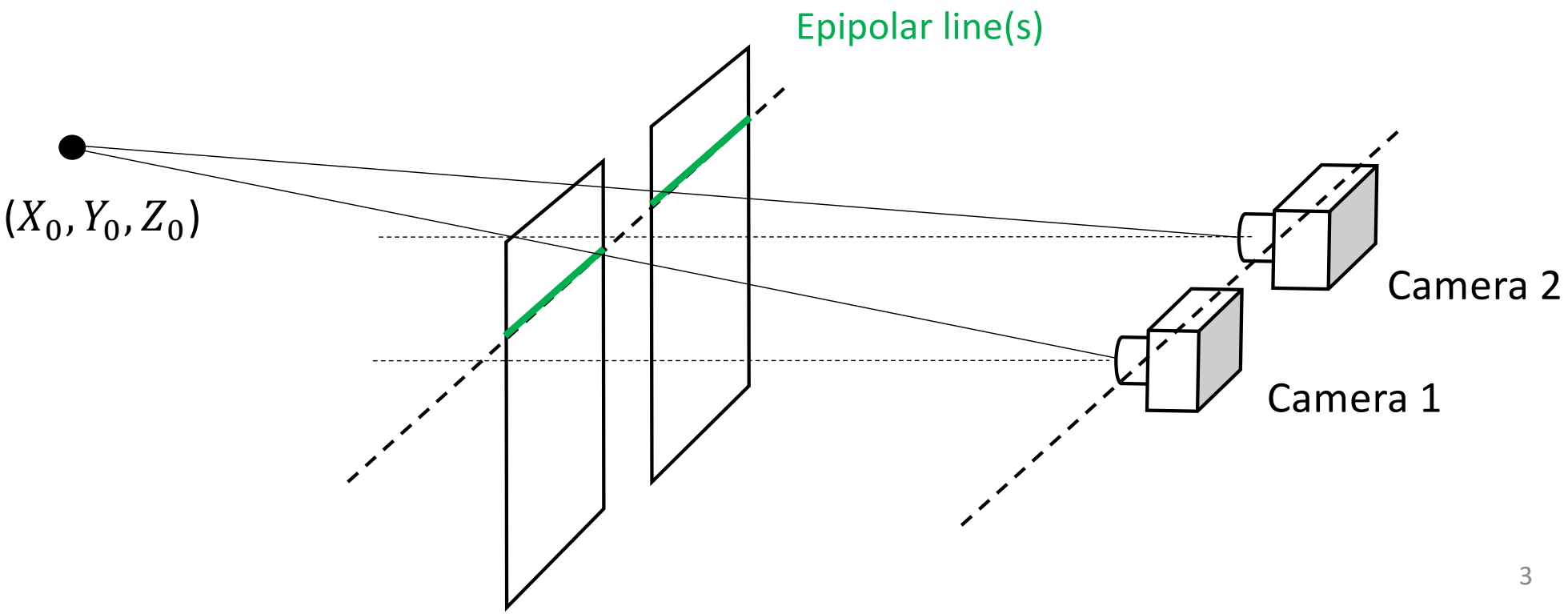
Recall from a past lecture: homographies (case 4)



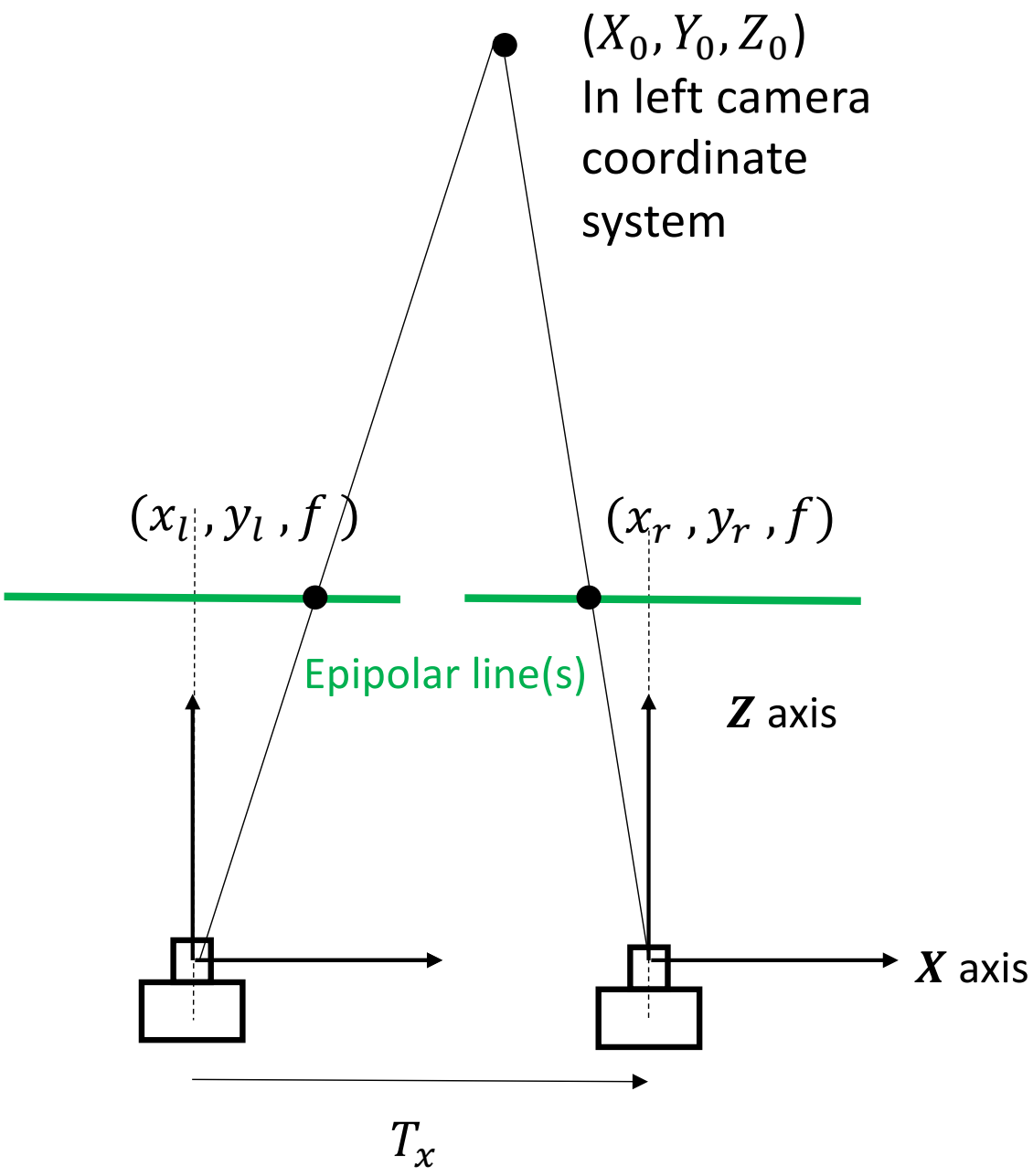
Assume the external and internal camera parameters are known (have been estimated). For each camera, one can compute a homography that rectifies the images.

Lecture 21 (last lecture): Suppose the real cameras are parallel.
(Or suppose the images have been rectified using *known* camera parameters.)

Pairs of matching image points lie along the same row.
To find matching pairs, we only need to search within each row.



View from above



Binocular disparity

$$d \equiv x_l - x_r$$

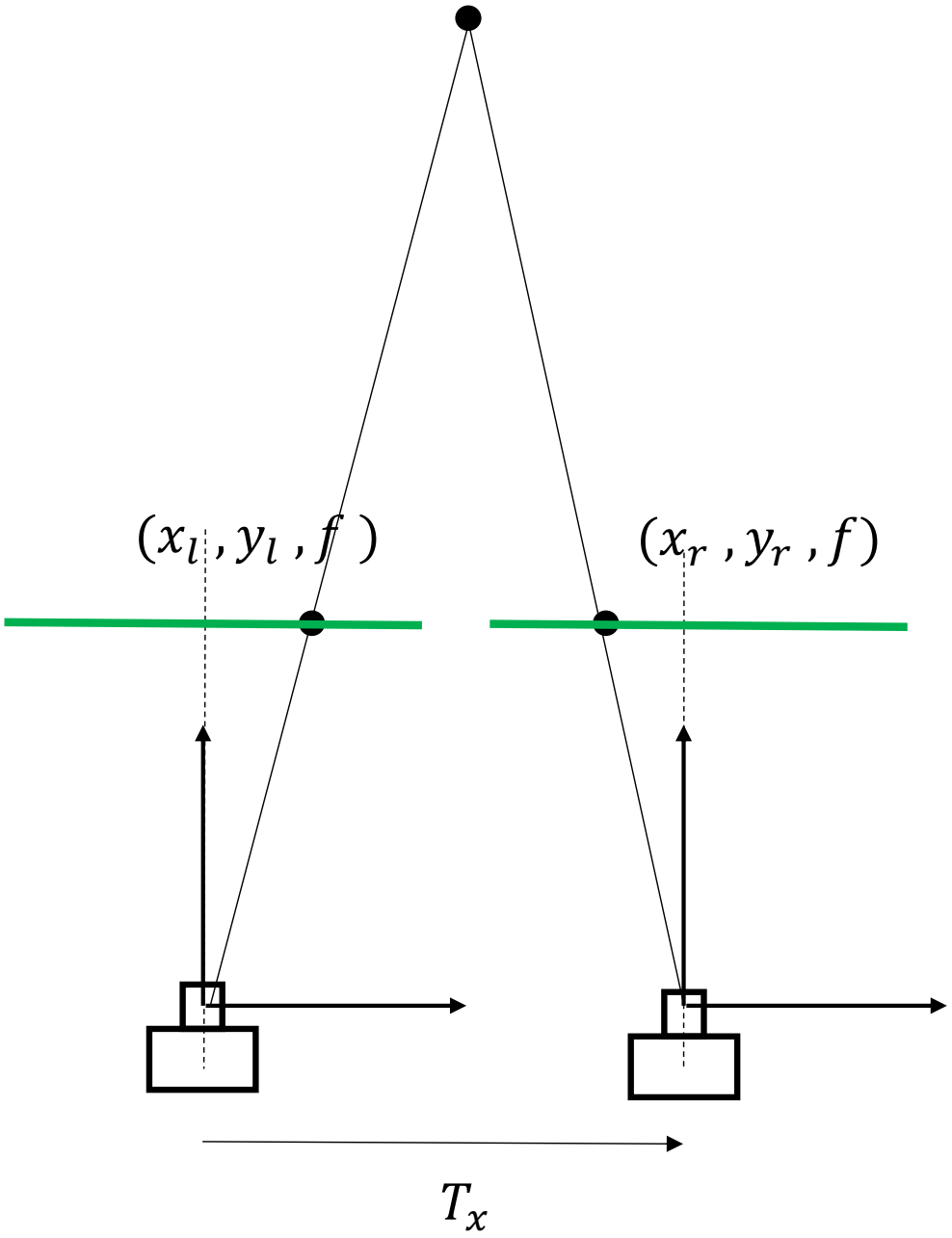
is the difference in image position of a 3D point in the (shared) image projection plane.

We showed last lecture:

$$\boldsymbol{d = \frac{f\, T_x}{Z_0}}$$

Binocular disparity

$$d \equiv x_l - x_r$$



If the camera parameters (e.g. $\boldsymbol{m_x}, \boldsymbol{f}, \boldsymbol{T_x}$) are known, then we can map disparity to depth.

Pixels per mm

$$\boldsymbol{d_{pixels}} = \frac{\boldsymbol{m_x f T_x}}{\boldsymbol{Z_0}}$$

Correspondence (matching) problem

- We can match feature points e.g. SIFT, or SURF. But feature points are sparse. We would like *dense* matches.

- Lucas-Kanade registration performs dense matching, but only for points that are locally distinctive. For stereo matching, we need images only to be locally distinctive *within each epipolar line.* This is still a problem for dense matching.

Example

Image 1

Cones image from the Middlebury Stereo database.

Some surfaces are textured. Others are not.

Example

Image 2

Cones image from the Middlebury Stereo database.

Some surfaces are textured. Others are not.

Overview of today

- Data costs and smoothness costs
- Random dot stereograms
- Occlusions
- Disparity Space
- Constraints on stereo matching

Lucas-Kanade type approach:

For each (x_l, y_l), find the disparity $d_l(x_l, y_l)$ that minimizes:

$$\sum_{(x,y)\in N g d\,(x_l,\, y_l)} \{\, I_l(x - d_l(x_l, y_l)\,, y) - I_r(x\,, y)\,\}^2$$

That is, shift the left image by disparity of the matching point.

$$d \equiv x_l - x_r$$

We only search in x direction, i.e. horizontal disparity only.

Lucas-Kanade type approach:

For each (x_l, y_l), find the disparity $d_l(x_l, y_l)$ that minimizes:

$$\sum_{(x,y)\in N_{gd}\,(x_l,\, y_l)} \{ I_l(x - d_l(x_l, y_l), y) - I_r(x, y) \}^2$$

Lucas-Kanade assumes a unique value of disparity in the local neighborhood of (x_l, y_l) .

Lucas-Kanade assumes that the image intensities (RGB) of matching points are roughly the same, and intensities are locally distinctive.

A typical approach to the first problem is to define a cost for surfaces that are not smooth.

For example, to require that the disparity function $d_l(x_l, y_l)$ doesn't change much from point to point, one could minimizes a "smoothness cost" such as :

$$\sum_{(x_l, y_l)} \{ d_l(x_l + 1, y_l) - d_l(x_l, y_l) \}^2 + \{ d_l(x_l, y_l + 1) - d_l(x_l, y_l) \}^2$$

Note that this doesn't depend on the image intensities.

To find the disparity map that (1) gives the best match of the left and right RGB images intensities *and* (2) gives a surface that is as smooth as possible, one tries to minimize the sum of two costs :

(1) "data cost"

$$\sum_{(x_l,\, y_l)} \quad \sum_{(x,y)\in Ngd\,(x_l,\, y_l)} \{\, I_r(x_l - d_l(x_l, y_l)\,, y_l) - I_l(x_l\,, y_l)\,\}^2$$

$$+ \quad \sum_{(x_l,\, y_l)} \{\, d_l(x_l + 1, y_l) - d_l(x_l, y_l)\,\}^2 + \{\, d_l(x_l, y_l + 1) - d_l(x_l, y_l)\,\}^2$$

(2) "smoothness cost"

One can give more weight to one versus the other, as one chooses. *There is no best way to do this, in general, i.e. that works best for every image pair.*

Data cost + Smoothness cost

Do we take the squared error? Or absolute values only? Or some other error?

What assumptions should we make about scene geometry? How much to we want to penalize a depth edge ?

Once one defines these cost functions, one also need data structures and algorithms for minimizing their sum.

There have been literally thousands of scientific papers written on this problem! I'll say a bit more about this later today.

Data cost + Smoothness cost

Overview of today

- Data costs and smoothness costs
- Random dot stereograms (history)
- Occlusions
- Disparity Space
- Constraints on stereo matching

Random Dot Stereogram

[Julesz, 1964]

left image

right image

left

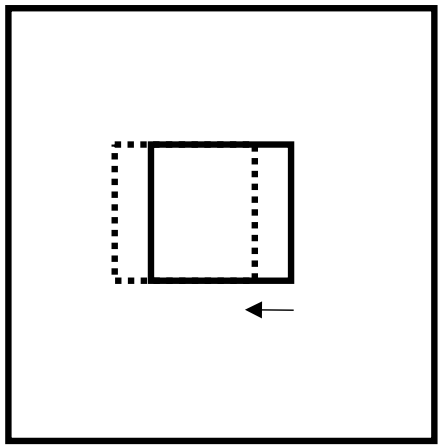
right

How to make a random dot stereogram?

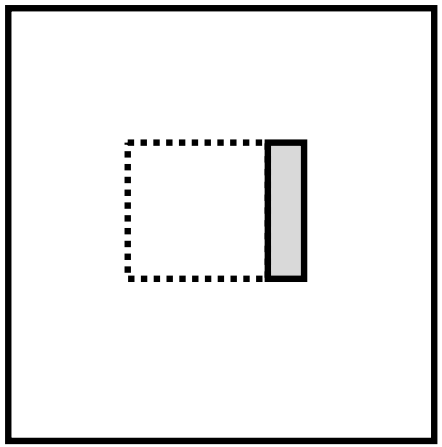
1) Make left image (random binary)

4) The result is the right image

2.) make a copy of left image;
then shift central patch towards left



3.) Fill empty patch (random binary)



$$d = x_l - x_r > 0 \ .$$

For the above example, the central patch has positive disparity and the background has zero disparity. So $Z_{background} = \infty$.

Let's use a slightly different stereogram.

We shift *both* the central patch and the background such that:

$$d_{central} > d_{background} > 0$$

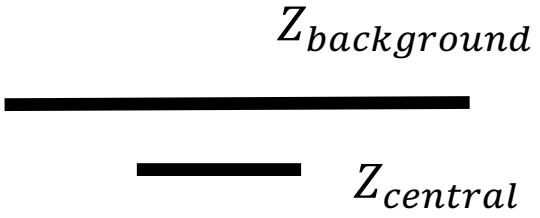
So

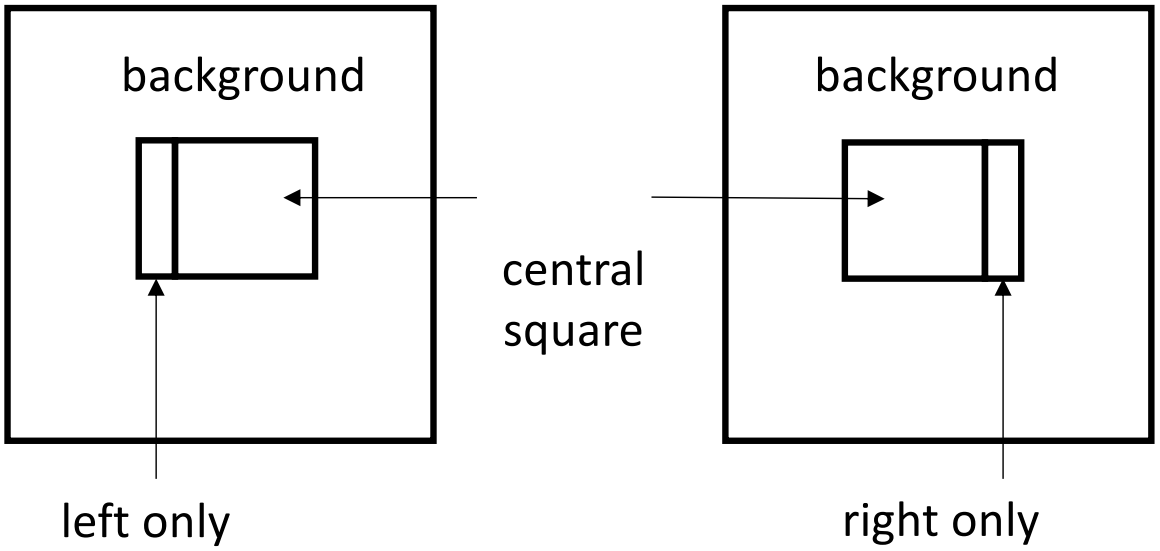
$$Z_{central} < Z_{background} < \infty$$

Left image

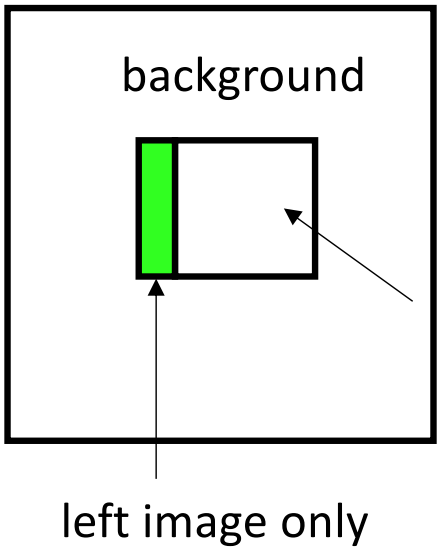
Right image

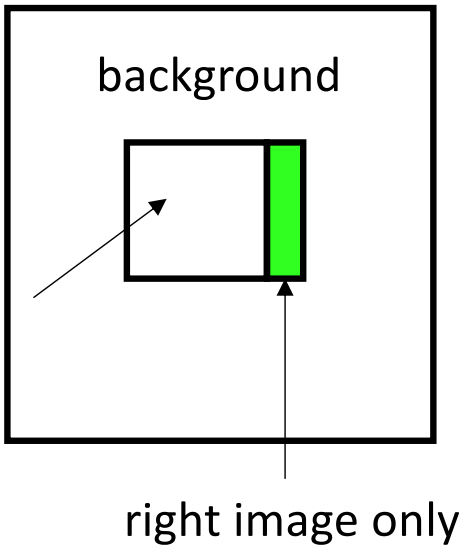
simulated 3D scene





Occlusions

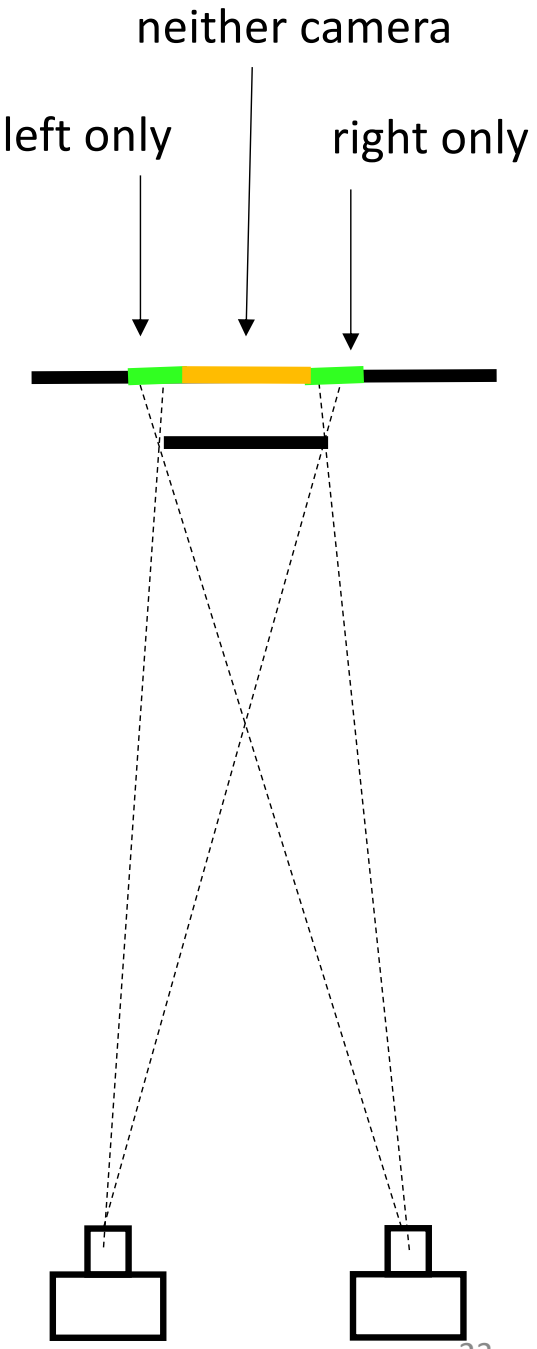




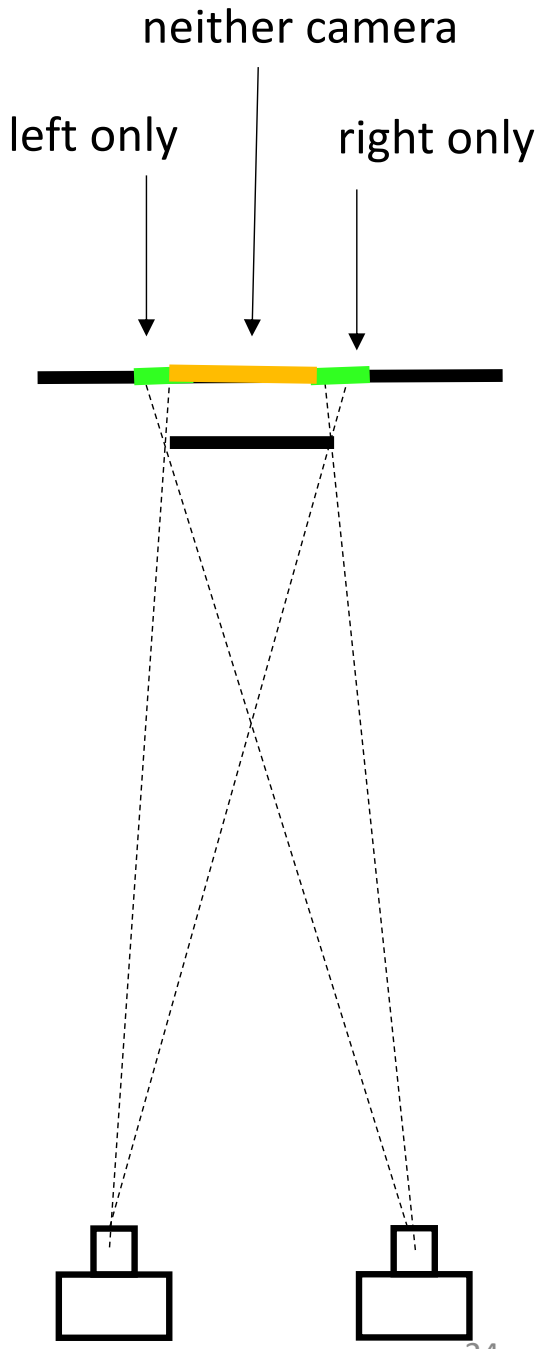
foreground square

left image only

right image only



Occlusions



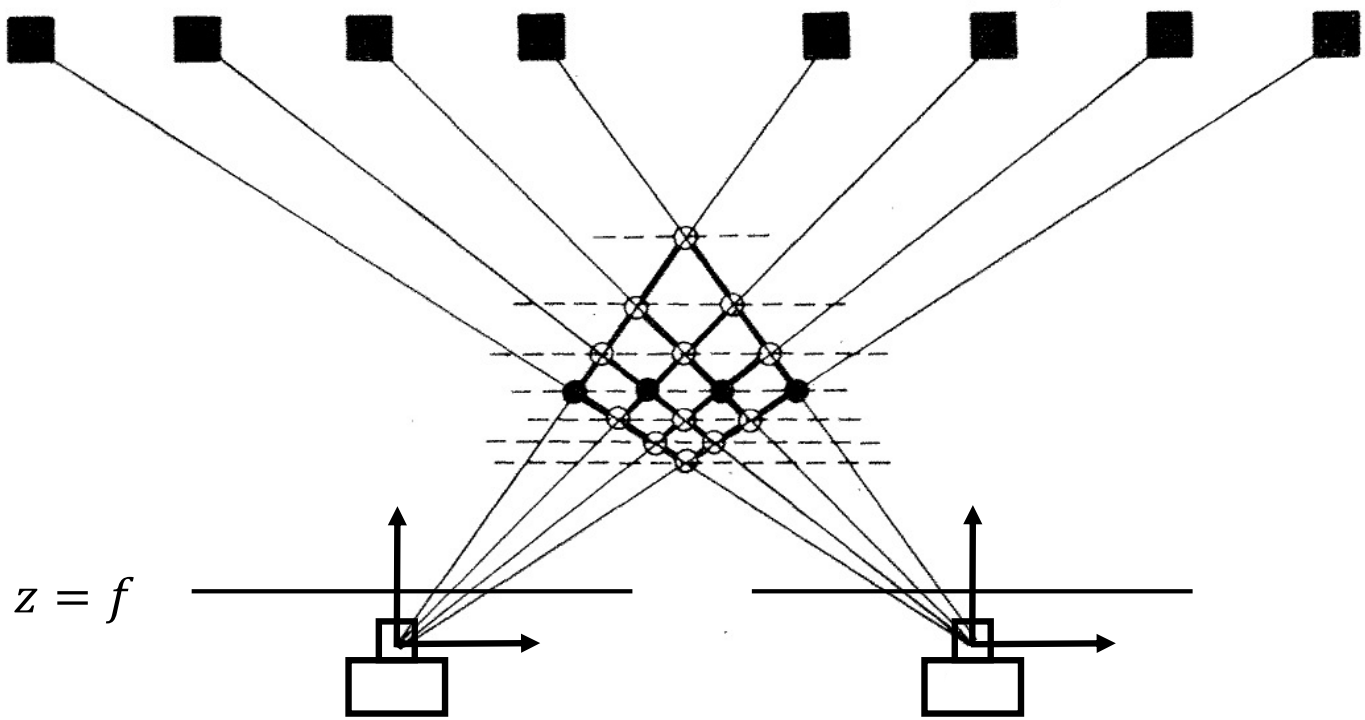
Points that are visible to one camera are called “monocular points” or “half-occluded” points.

There is no correct binocular match for these points.

Overview of today

- Data costs and smoothness costs
- Random dot stereograms (history)
- Half Occlusions
- Disparity Space
- Constraints on stereo matching

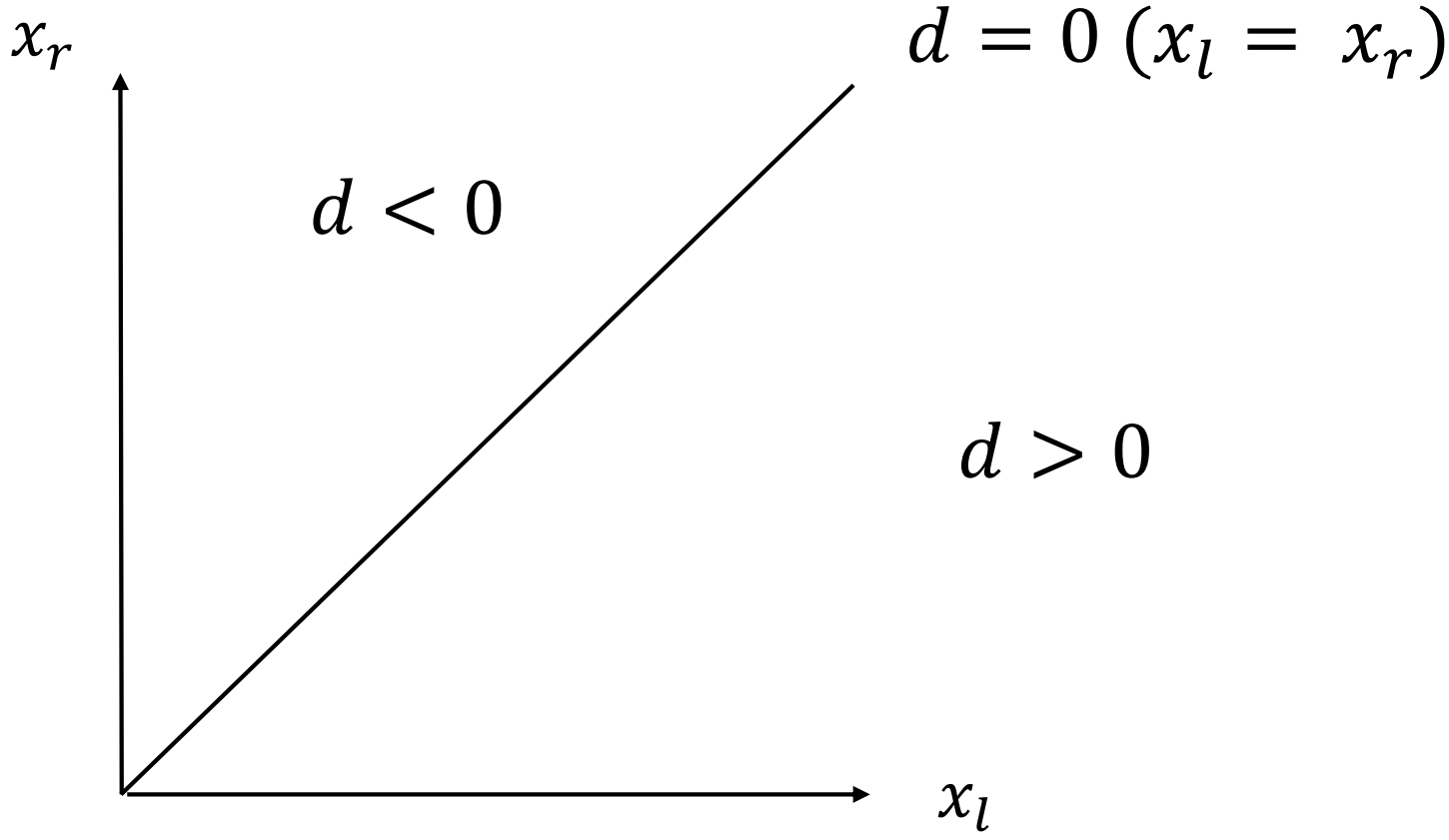
How to formulate the matching problem?



Which points in the left image correspond to which points in the right image ?

[Marr and Poggio, 1976]

Data structure: "2D Disparity Space"



Each pair of values (x_l, x_r) represents a unique point in depth

($d = x_l - x_r$, $d_{pixels} = \frac{m_x f T_x}{Z_0}$)

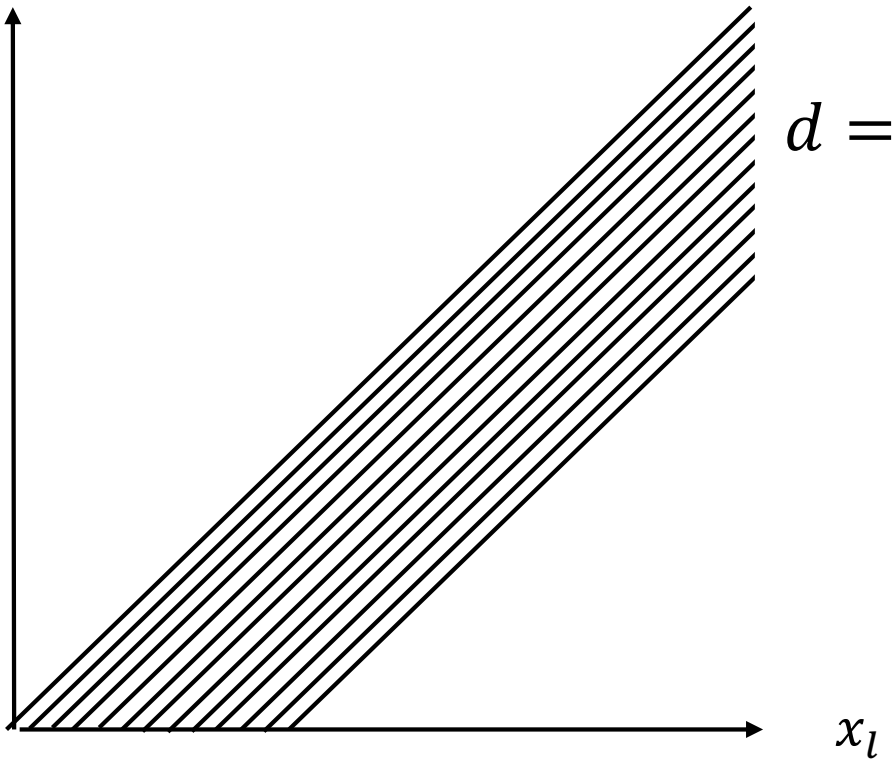
Lines of constant disparity are lines of constant depth.

$$d_{pixels} = \frac{m_x \, f \, T_x}{Z_0}$$

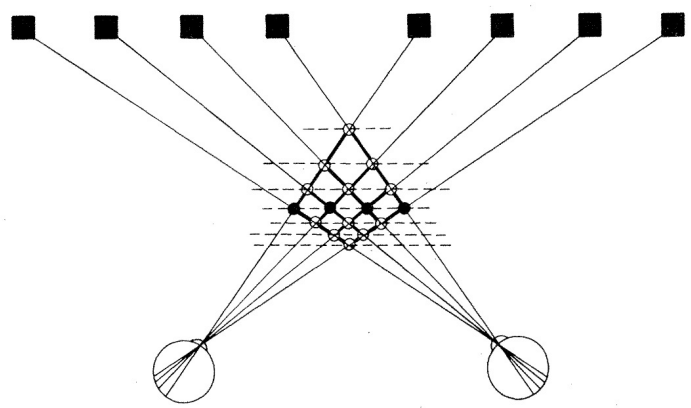
$d = 0$

x_r

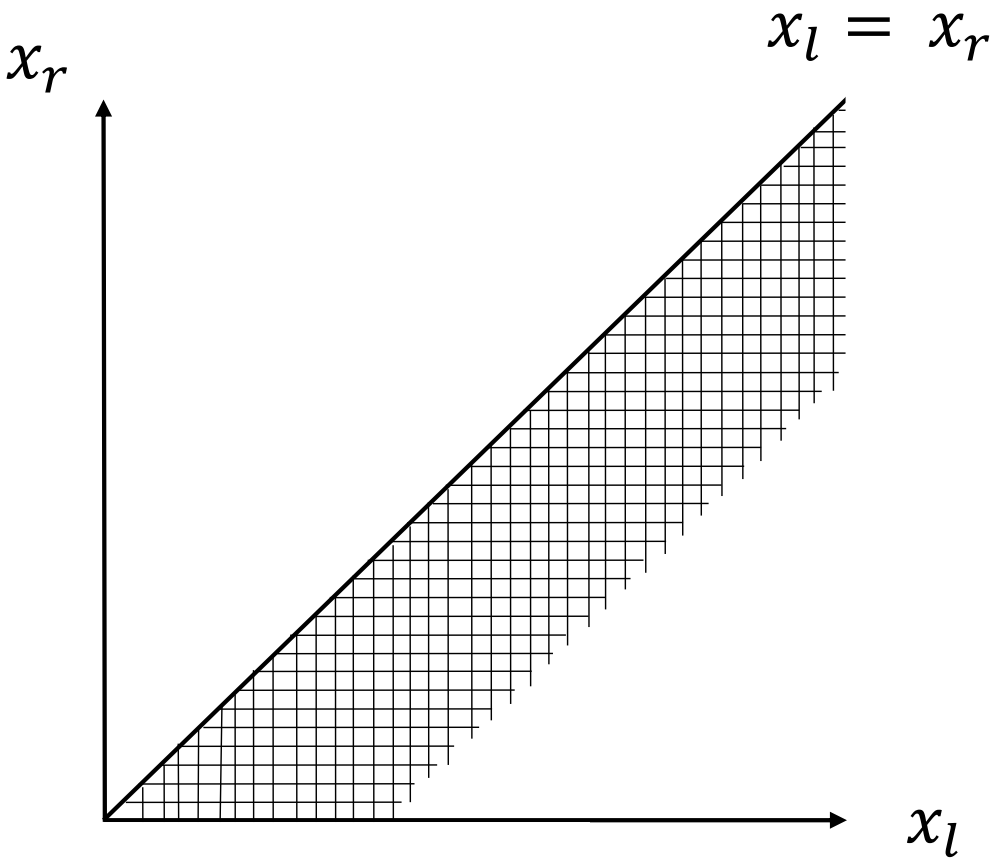
$d = x_l - x_r$

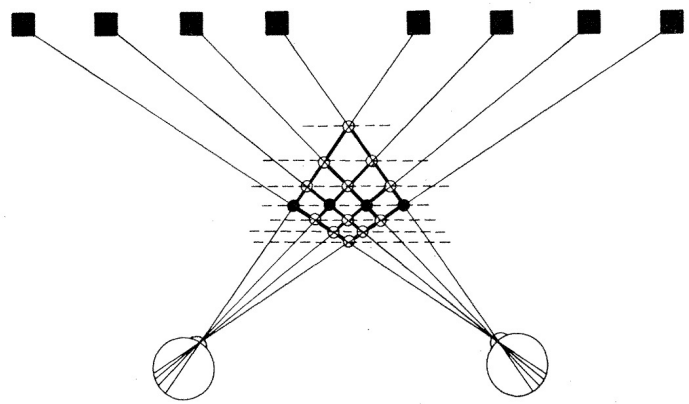


x_l



Lines of constant x_l or x_r are “lines of sight”.
(They should have at most one match.)



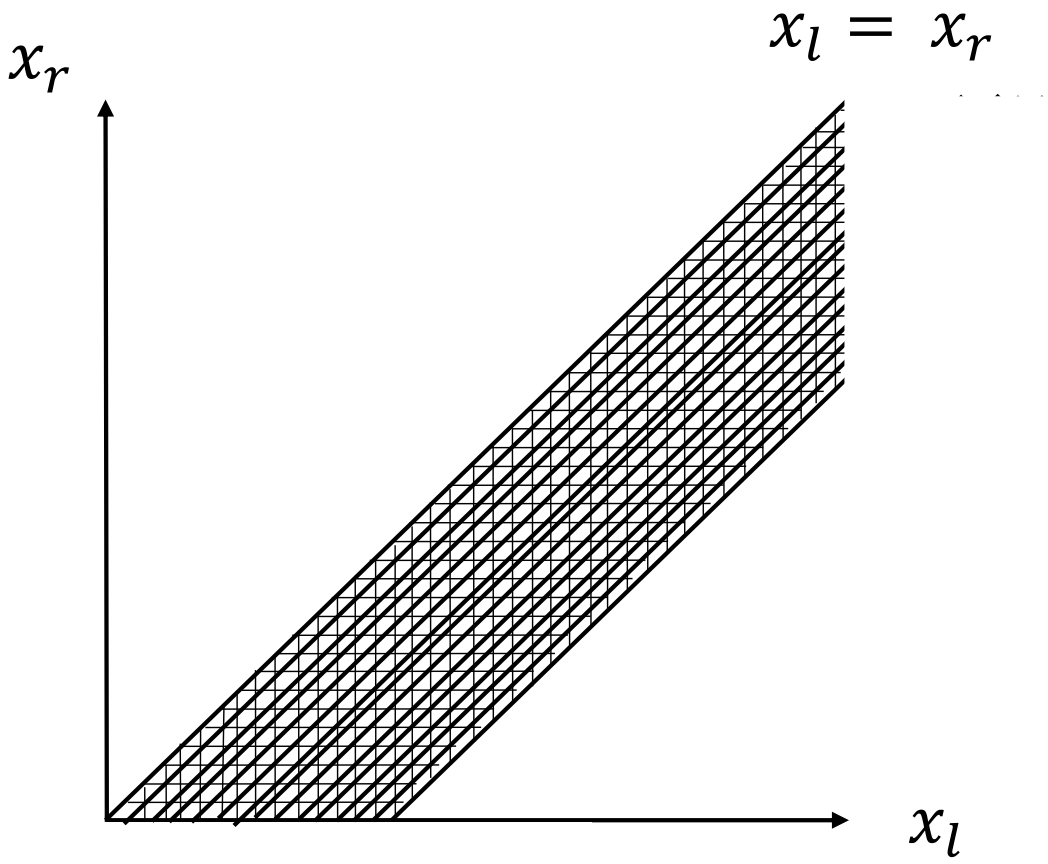


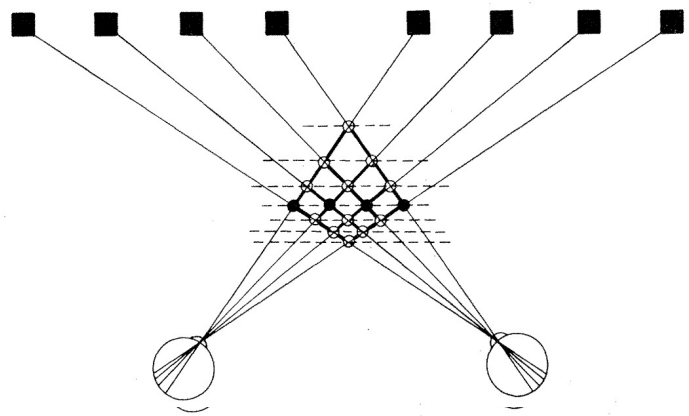
Q: How to pose the matching problem?

A: Use a graph. Vertices are points in space.

Edges join points in space (constant depth, or along lines of sight)

There is also a vertical dimension y (not shown).





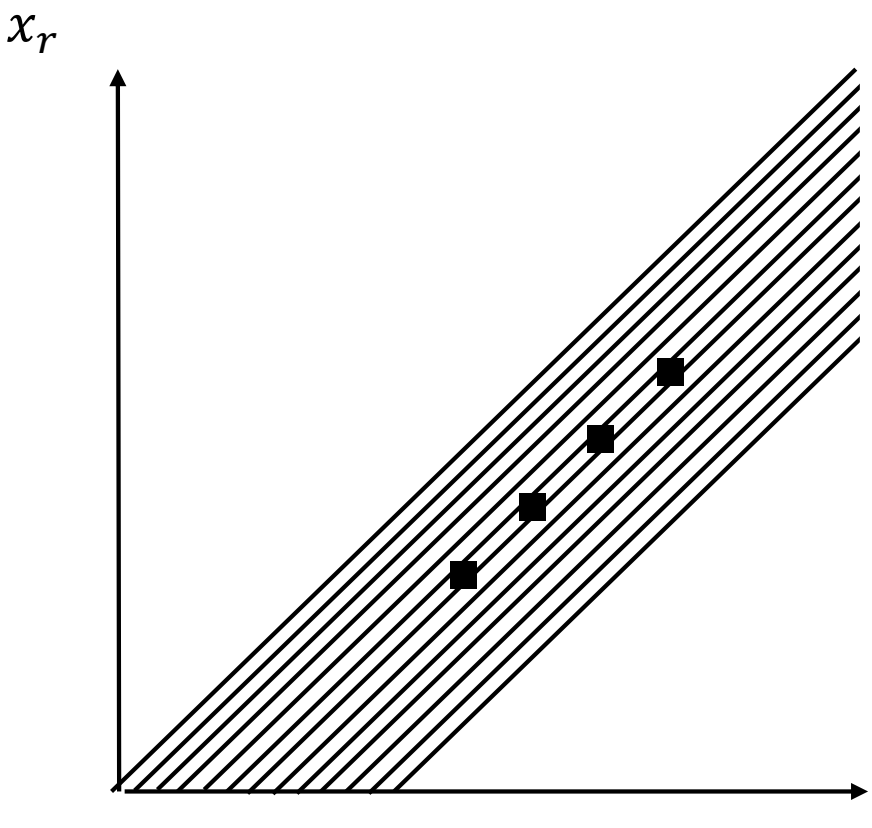
Marr and Poggio (1976) formulated computational *constraints* on the matching problem, aka "correspondence problem":

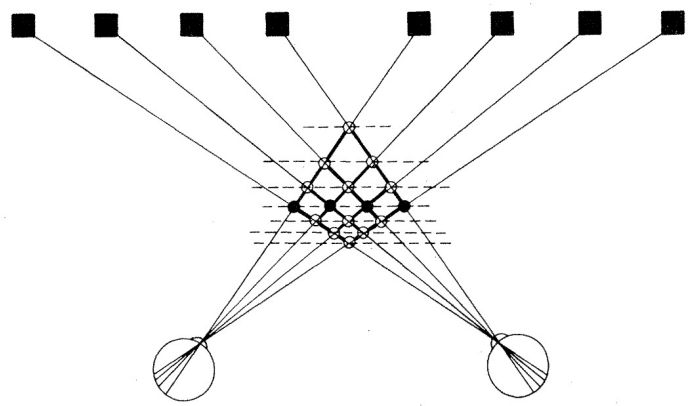
- *Continuity* : surfaces in the world are piecewise continuous; thus, disparities of matching points should be piecewise continuous too

- *Uniqueness* : for each point in left eye, there is typically one matching point in right eye

They also implemented an iterative algorithm based on these constraints. (Details omitted but I will show you an example a few slides from now.)

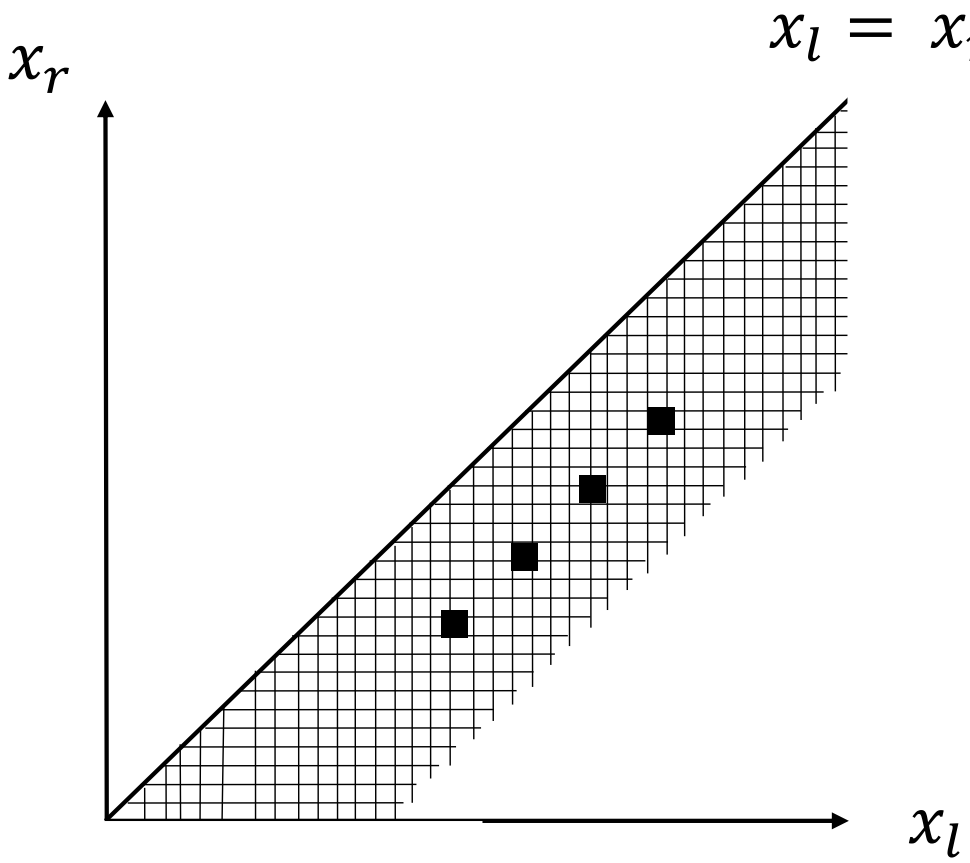
“Continuity constraint”: disparity tends to change slowly with x_l or x_r.

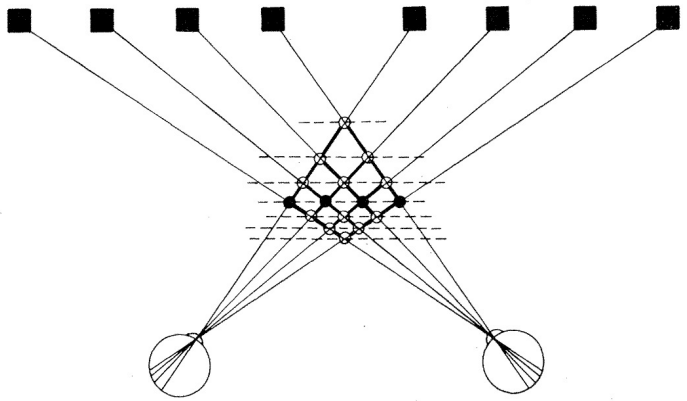




Matching points tend to lie along lines of constant disparity (i.e. lines of constant depth)

"Uniqueness constraint": for each x_l , there is one corresponding x_r and vice-versa.



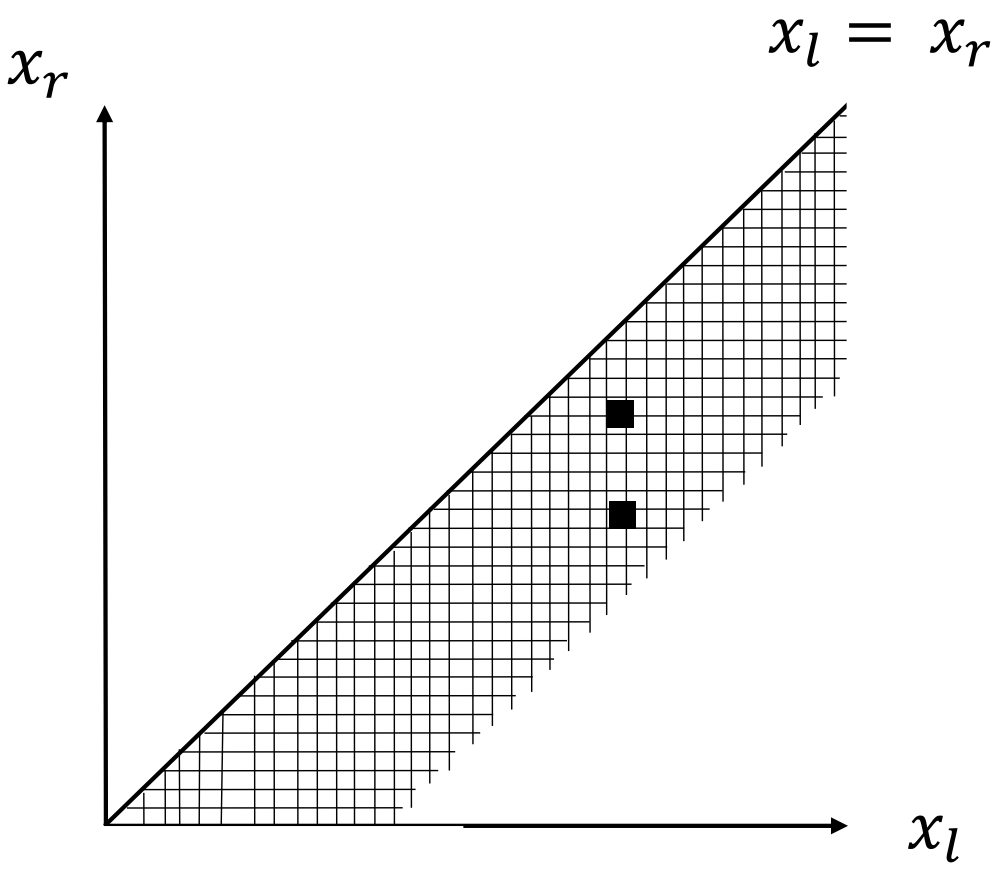


For each position in left image, there is typically exactly one position in right image.

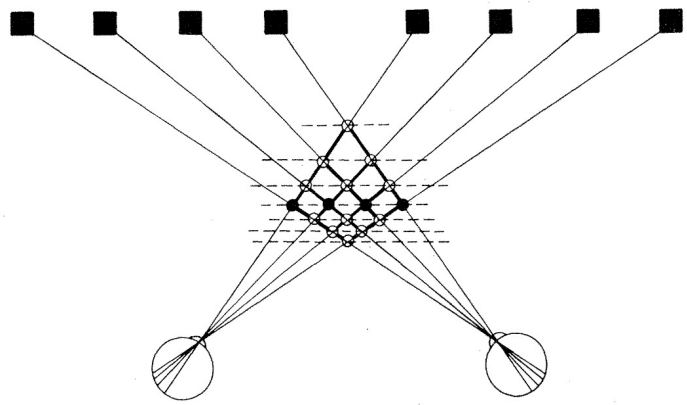
Marr and Poggio's continuity and uniqueness constraints can fail.

(They were well aware of this.)

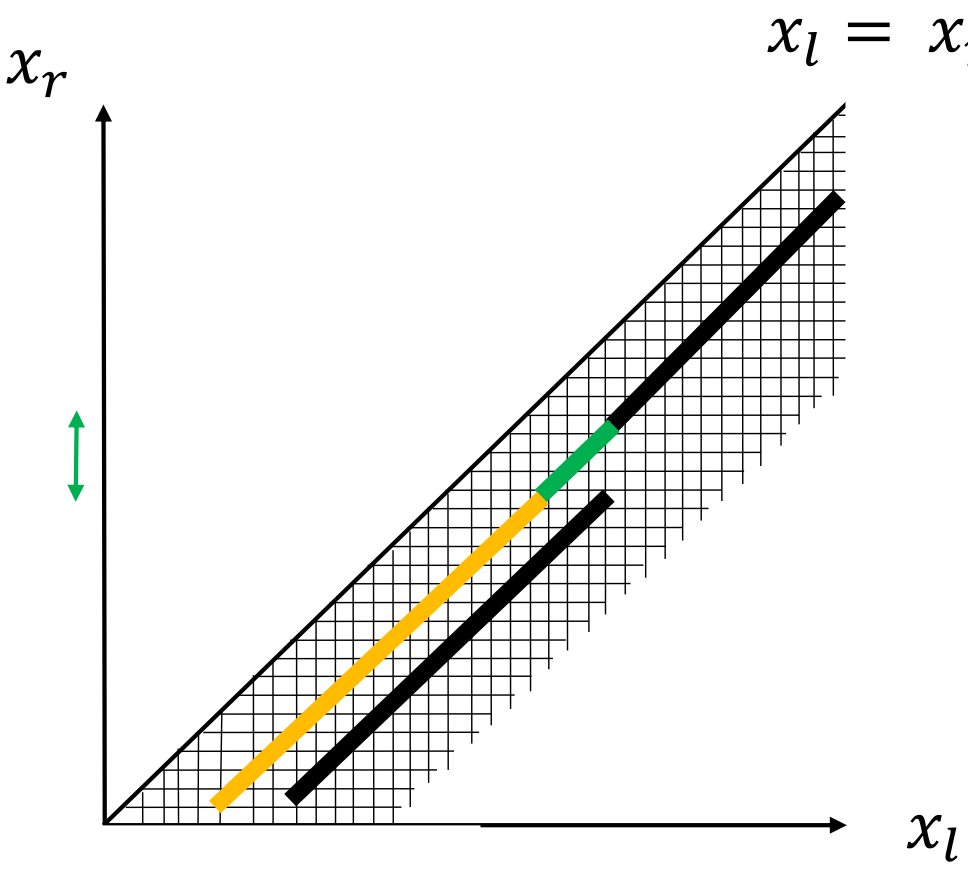
Let's look at a few examples where uniqueness fails, …

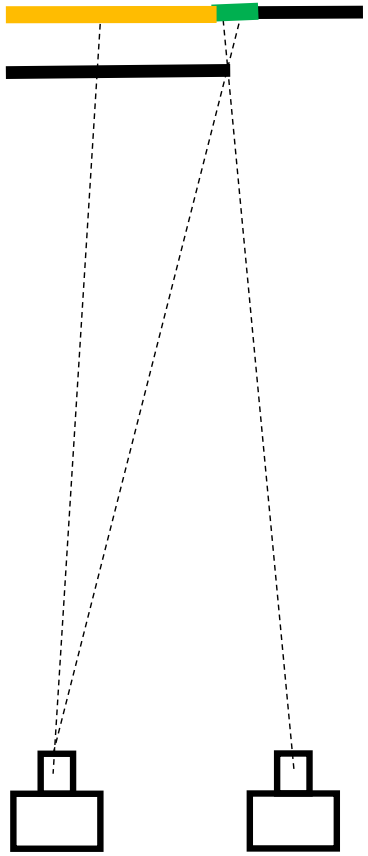


What is the picture below that corresponds to this failure?



The uniqueness constraint fails because there is more than one match for some x_l .

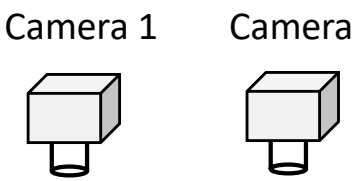


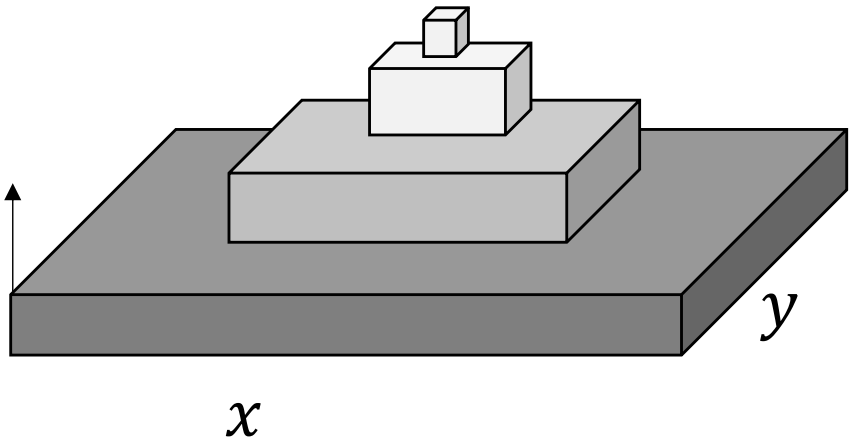


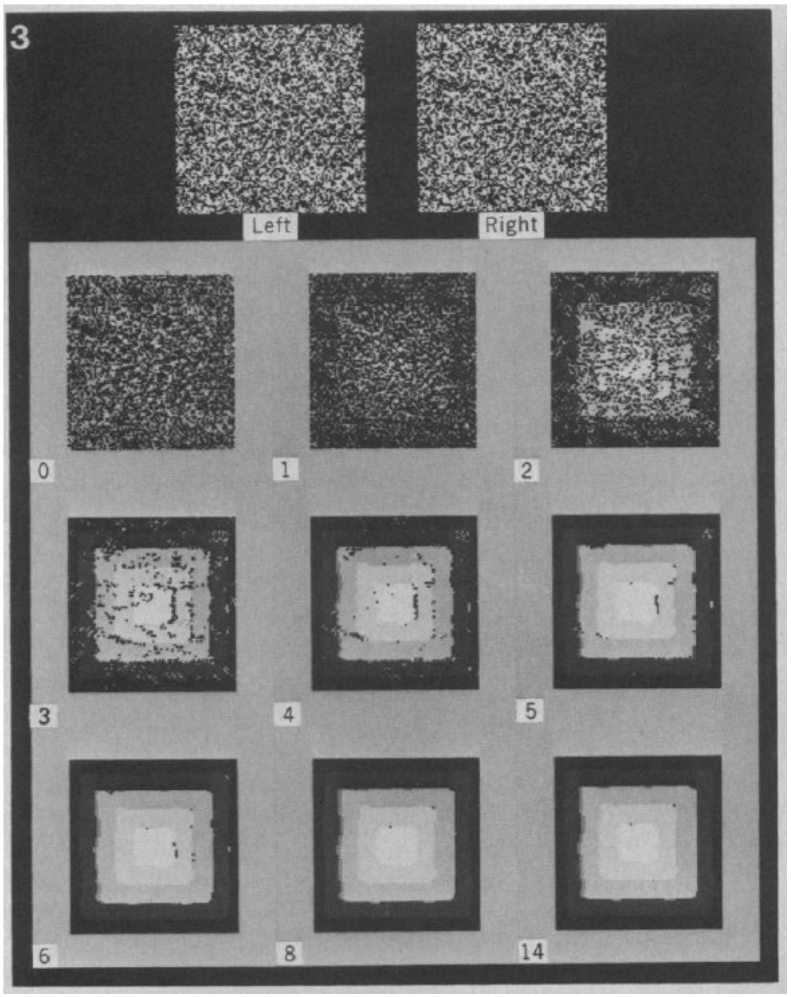
The uniqueness constraint fails at the depth discontinuity, because there are x_r positions that have *no match* x_l *in the left image.*

Note disparity can be defined both in left and right images!

Example from Marr and Poggio (1976). The image pair is a random dot stereogram (top row). The surface depth is like a "wedding cake". The algorithm estimates disparity iteratively. The figure below shows how the estimated disparity (grey value) evolves over time steps.







Apologies for low quality image..

Recall earlier: stereo matching can be formulated as the problem of choosing disparities that minimizes a global cost:

(1) "data cost" (e.g. Lucas Kanade)

$$\sum_{(x_l,\, y_l)} \sum_{(x,y)\in Ngd\,(x_l,\, y_l)} \{\, I_l(x_l - d_l(x_l, y_l)\,, y_l) - I_r(x_l\,, y_l)\,\}^2$$

$$+ \sum_{(x_l,\, y_l)} \{\, d_l(x_l + 1, y_l) - d_l(x_l, y_l)\,\}^2 + \{\, d_l(x_l, y_l + 1) - d_l(x_l, y_l)\,\}^2$$

(2) "smoothness cost" that captures global constraints e.g. in regions of uniform intensity within rows

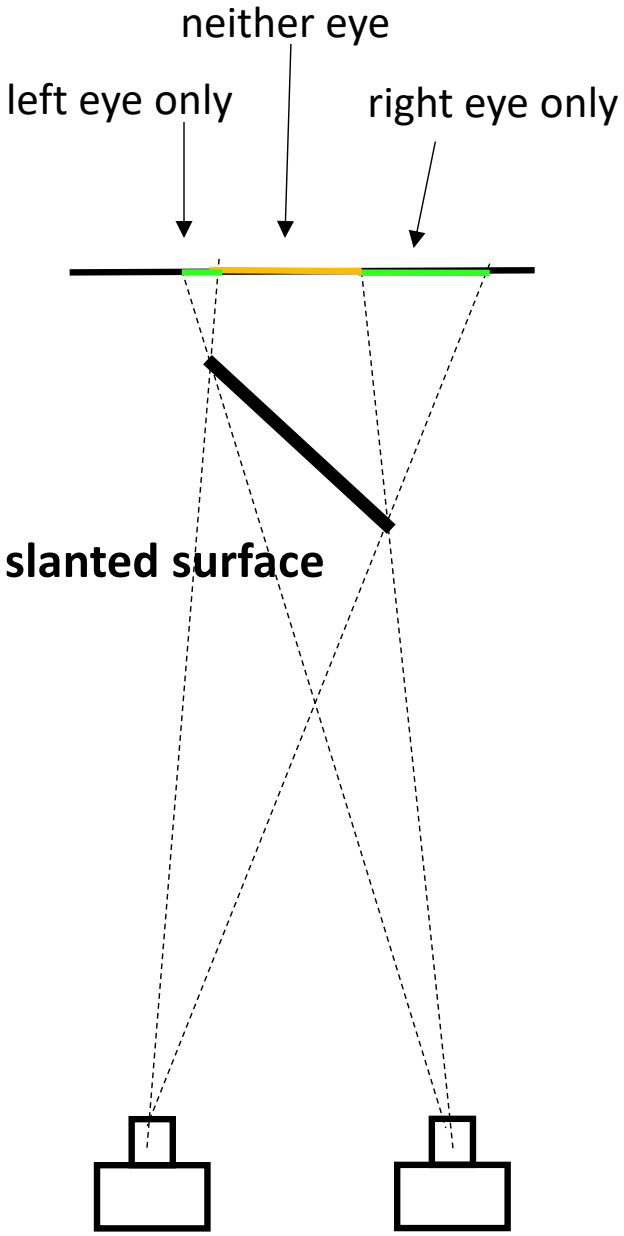
Marr and Poggio also formulated their uniqueness and continuity constraints in terms of a "data" and "smoothness" cost. (Details omitted.)

Modern graph based Stereo Approaches
[e.g. Sun *et al* 2005]

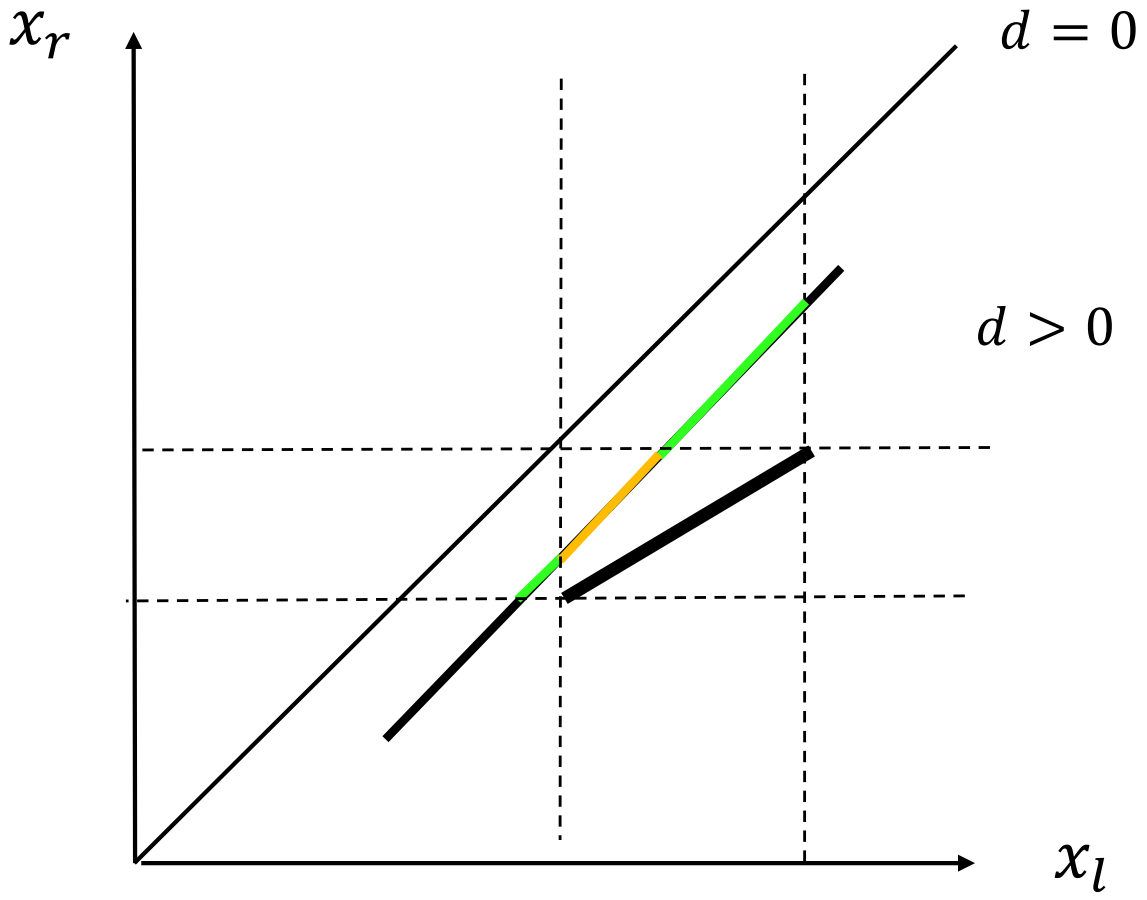
Use more sophisticated costs, for example:

- define disparity $d_l(x_l, y_l)$ in the left image *and* disparity $d_r(x_r, y_r)$ in the right image
- define binary *occlusion functions* $O_l(x_l, y_l)$ and $O_r(x_l, y_l)$ which label a point as matched or not (visible only to that camera) which allows for more precise geometry reasoning
- allow for slanted surfaces (next slide)

Use state-of-the-art graph algorithms for solving combinatorial optimatization problems.



The foreground surface spans more pixels in left image than right image.
The foreground surface is continuous, even though the disparity varies. *So don't penalize!*



There have been thousands of papers written on binocular stereo matching.

In the early 2000's, researchers created databases with test images and ground truth so that algorithms could be compared. See Middlebury Stereo vision page (link).

It became impossible to publish a stereo algorithm in a top conference or journal without showing that it performed well on this data set *and* that it had some advantages (simplicity, speed) over other commonly used algorithms.

Assignment 3

Start with questions 1, 2, 3.

For question 2 you can use SURF features – we have provided sample code.

Whereas you can use Matlab's CV library to check things, you are asked to implement your own code for the several parts of the assignment.

Question 5 is about disparity estimation, and for this you can be creative and come up with solutions and discuss limitations of your approach (we don't expect perfect results).