CSE 527:
Introduction to Computer Vision

Week 5 - Class 1: Matching, Stitching, Registration

September 26th, 2017


●●●●○ Verizon LTE
3:02 PM
42%
Albums
sheepdog or mop
Select


●●●○○ Verizon
4:20 PM
76%
Albums
chihuahua or muffin
Select
@teenybiscuit


Verizon
4:20 PM
69%
Albums
shiba or marshmallow
Select
@teenybiscuit

???


Verizon
4:20 PM
69%
Albums
kitten or ice cream
Select
@teenybiscuit

Recap

Gradient Histogram

A very powerful idea that is the basis for many image & feature descriptors.

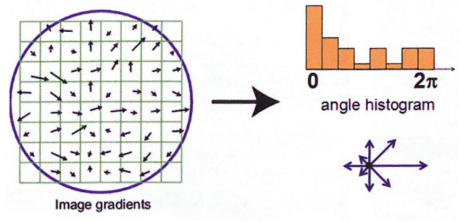
Why? Hint: invariant in more than one way...

Look at a (local) region and calculate local gradients, then bin the gradients in a histogram.

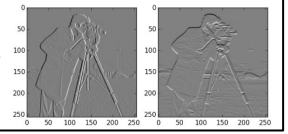
How to get direction and magnitude of edges?

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

Most can be pre-calculated over the entire image with a single convolution:



$$L_x = \frac{\partial L}{\partial x}, L_y = \frac{\partial L}{\partial y}$$
$$m(x,y) = \sqrt{L_x(x,y)^2 + L_y(x,y)^2}$$
$$\theta(x,y) = \tan^{-1}\left(\frac{L_y}{L_x}\right)$$



Feature Matching

Given features of image 1 and of image 2, how do we find the best matching of the two sets?

- Define a distance metric between a pair of feature descriptors
- Find the best matching.

Some metrics are determined by the descriptor (Hamming, L_2 norm).

$$M(f_L, f_R) = \|f_L - f_R\|_{L_2} < t$$

What is an optimal matching?

Naive: Pick the nearest neighbor (NN), set a threshold on distance.

SIFT matching and object finding

Today

Feature Matching

Image Alignment

Panoramas

HW2!

Feature Matches

Feature Matching

Given features of image 1 and of image 2, how do we find the best matching of the two sets?

- Define a distance metric between a pair of feature descriptors
- Find the best matching.

Naive: Pick the nearest neighbor (NN), set a threshold on distance.

$$M(f_L, f_R) = \|f_L - f_R\|_{L_2} < t$$

Apply: Ratio test and Reciprocity

SIFT matching, ratio, recip., and object finding

Feature Matching | Measuring Performance

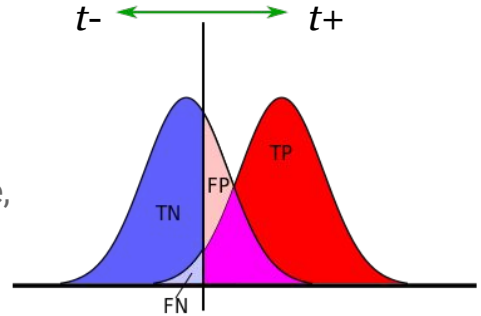
Receiver Operator Characteristic (ROC) Curve

- Vary the threshold *t*
- Calculate precision and recall
- Take a *t* that has highest TP-rate, lowest FP-rate.

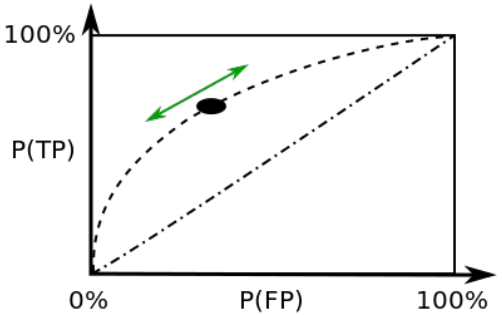
Precision: TP / (TP + FP)

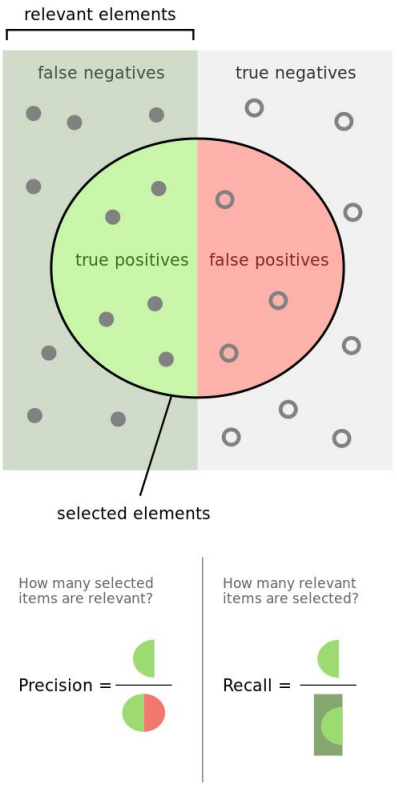
“

Recall: TP / (TP + FN)



TP	FP
FN	TN





Feature Matching | Brute Force

For every feature i in image 1

- Go over all features in image 2
- Take the one (k) that are closest to i

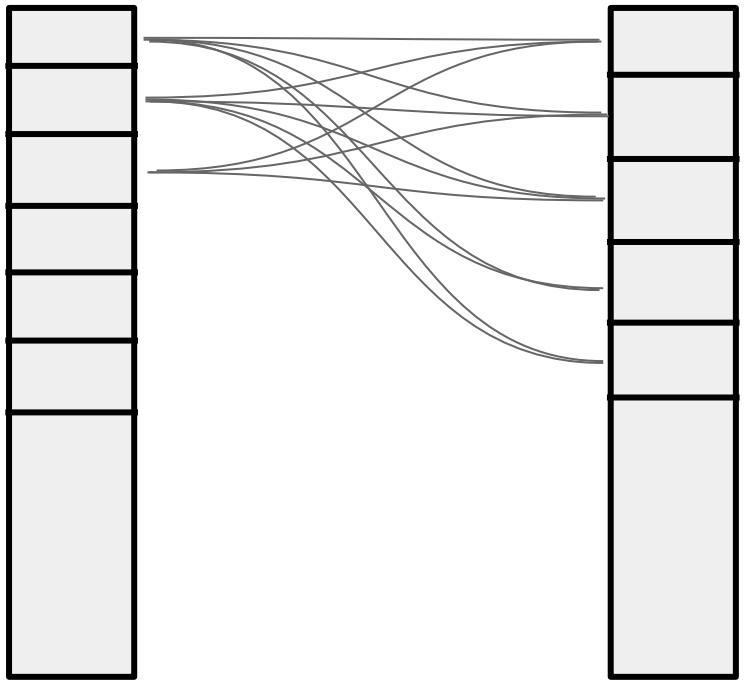
Complexity: $O(n^2)$

Good:

- Simple
- Easily parallelizable
- Exhaustive, deterministic

Bad:

- Slow



Feature Matching | k-d tree

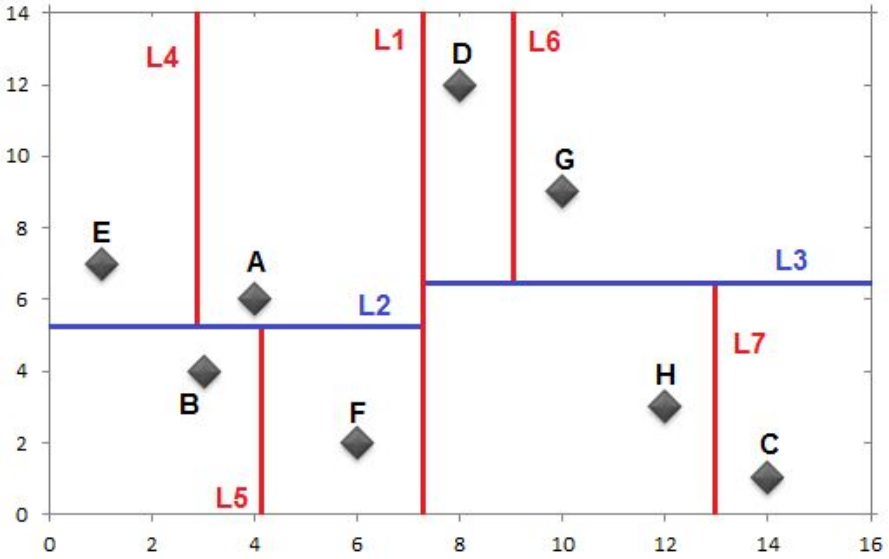
Accelerate the matching process.

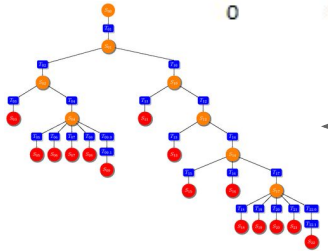
Preprocess the “training” dataset:

- Hierarchically subdivide the k-dimensional space (by sorting)
- If a bin has more than m points - split to smaller spatial bins.

Build the tree: O($k\ n$ log n)

Search the resulting tree: O(log $n + m$)





Feature Matching | k-d tree

Accelerat

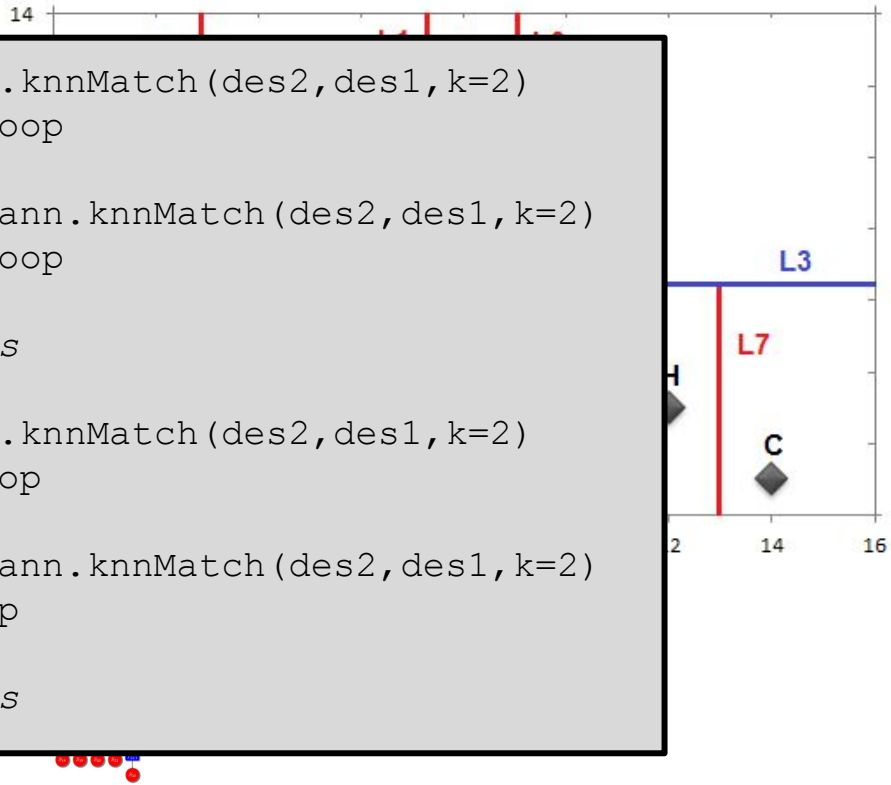
Preproces

- Hie
  spa
- If a
  sma

Build the t

Search th

```
$ %timeit -n50    matches2to1 = bf.knnMatch(des2,des1,k=2)
50 loops, best of 3: 17.1 ms per loop

$ %timeit -n50    matches2to1 = flann.knnMatch(des2,des1,k=2)
50 loops, best of 3: 10.5 ms per loop

Speedup: ~ x1.7 with ~1000 features

$ %timeit -n50    matches2to1 = bf.knnMatch(des2,des1,k=2)
50 loops, best of 3: 148 ms per loop

$ %timeit -n50    matches2to1 = flann.knnMatch(des2,des1,k=2)
50 loops, best of 3: 33 ms per loop

Speedup: ~ x4.4 with ~3300 features
```



Questions?

Image Alignment

Panoramic view, right?

Panoramic view, right?

Panoramic view, right?

Image Alignment | Motivation

Why align, stitch images together?

- **The human eye has ~210°$_h$ x 150°$_v$ FOV**
- Our images/cameras have narrow FOV
- Object is just. too. big.
- We want to deliver the “big picture”

Image Alignment | Applications

Panoramas

Image Alignment | Applications

Google Street View

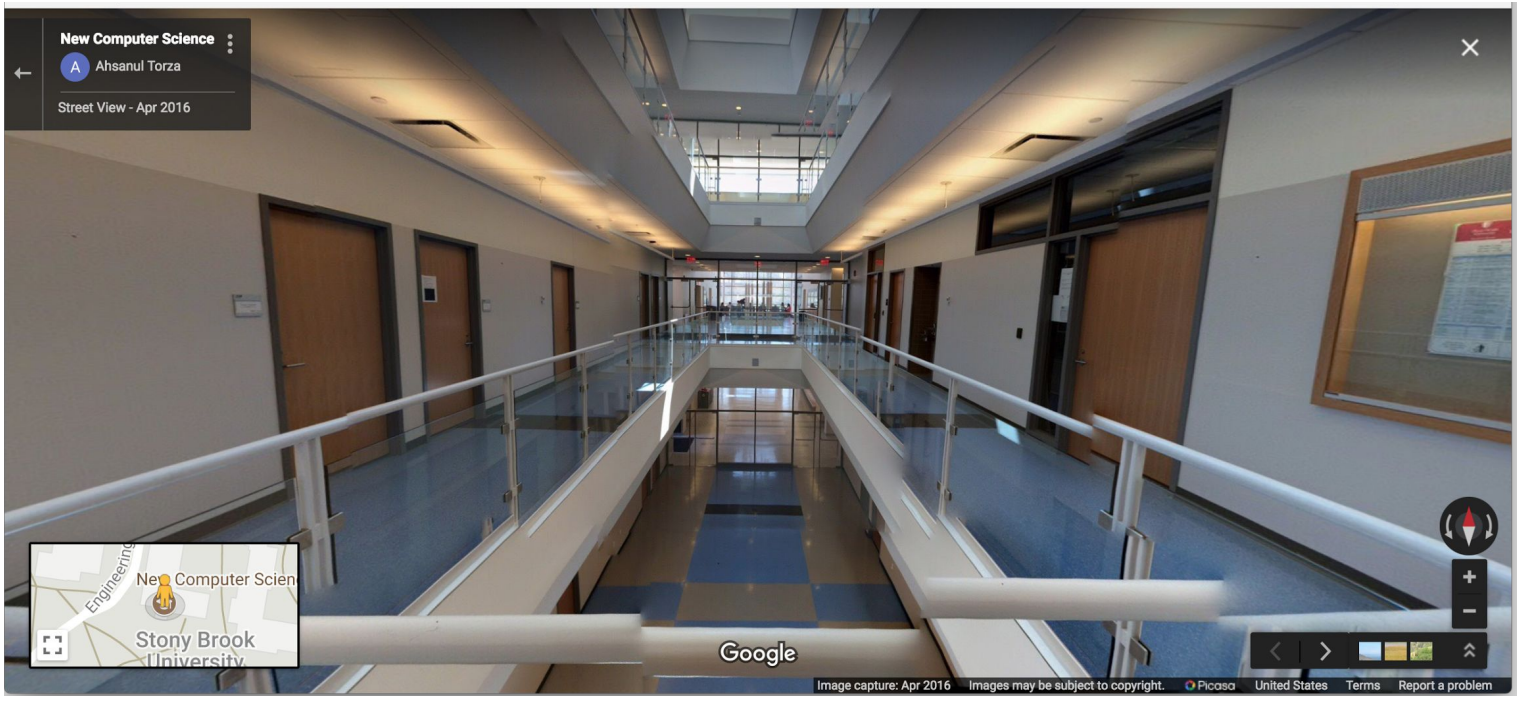


Image Alignment | Applications

360 VR

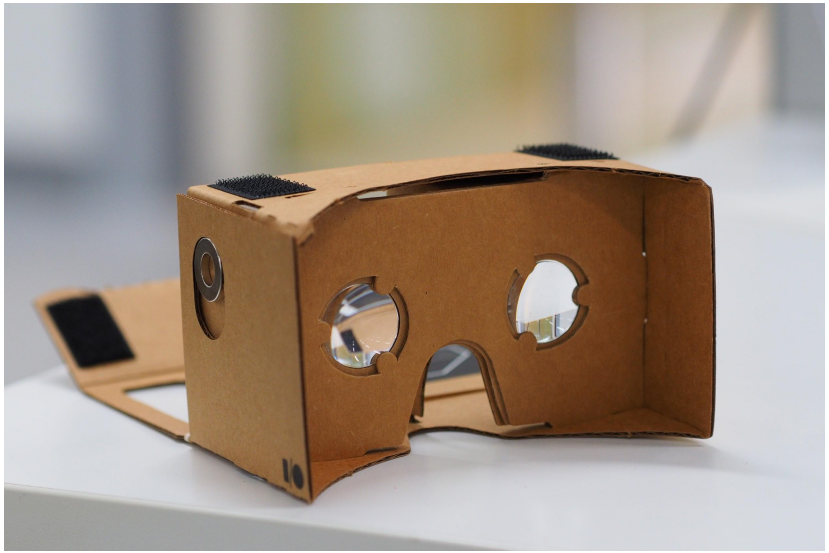


Image Alignment | Applications

Aerial mapping, Surveillance, Drones!

Link

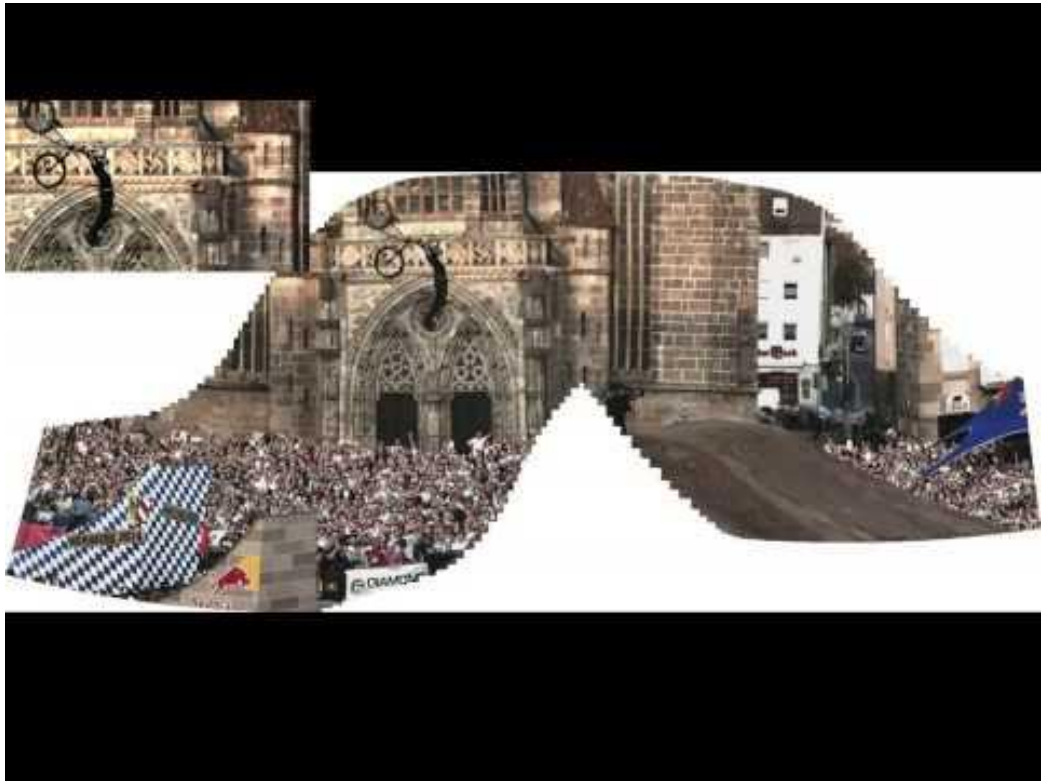
Image Alignment | Applications

Image search, Search in image

[OpenCV]

Image Alignment | Applications

Video stabilization



[youtube]

Image Alignment | Challenges

Is image alignment hard?

Between any two images we may have:

- Little overlap
- Occlusion
- Clutter (confusing features)
- Intensity changes
- Distortions
- Different hardware, optics
- …

→ Alignment should be robust to a lot of variance.

Image Alignment | Panography

What if we were aligning real paper images?

- We can only rotate and translate.

This was done: Panography.

David Hockney, 1986

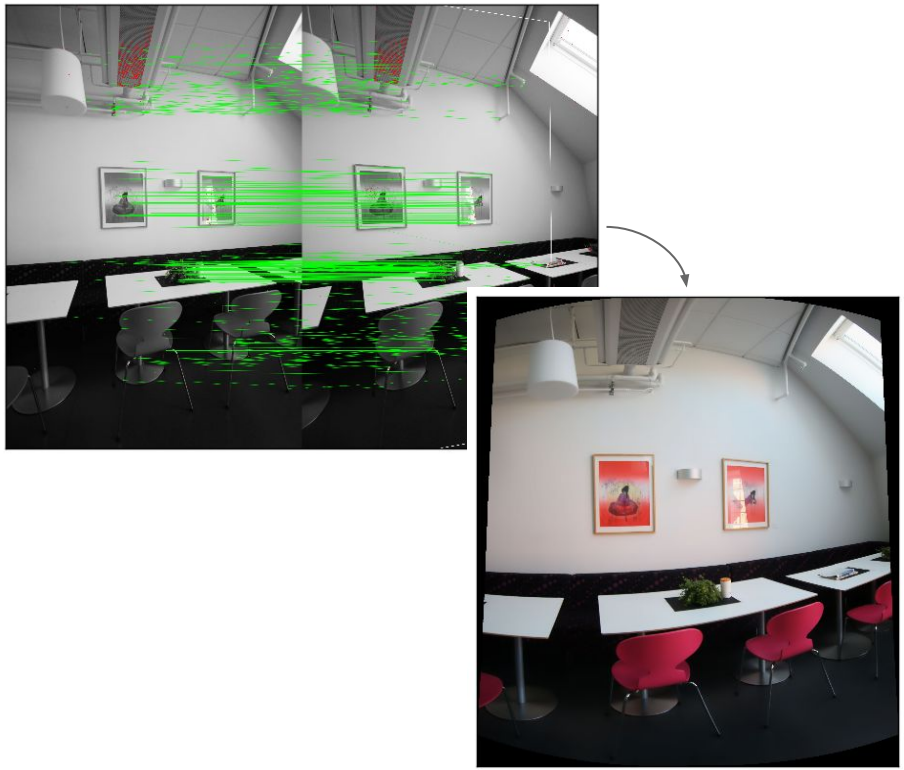
Feature Alignment

Instead of looking at pixels, we look at **interest points**.

- Less data to process
- More potential noise (from false matching)

Method overview:

1. Find key points and feature descriptors (Last time)
2. Match descriptors (we saw today)
3. Apply an overall geometric configuration (Today)
    - Affine, Homography
4. Find optimal parameters (Today)
    - Fitting, LSQ, robust LSQ
5. Blend



Alignment as a Fitting Problem

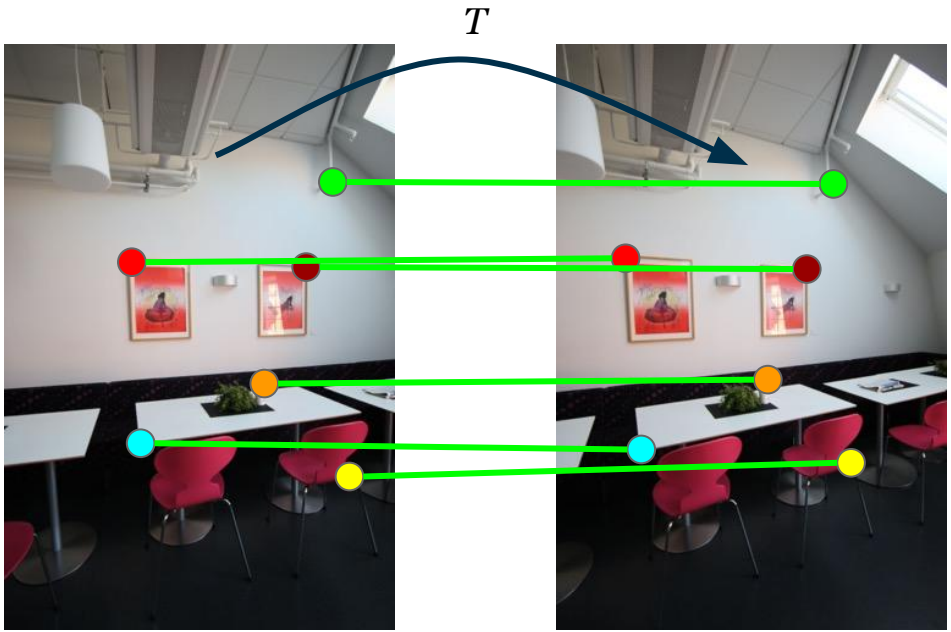
Over the feature matching:

- Design a goodness of fit measure

$$\hat{T} = \text{argmin}_T f(T(X_1), X_2)$$

$$f(T(X_1), X_2) = \sum_i \|T(x_{1_i}) - x_{2_i}\|$$

- Design an optimization scheme
  - Avoid local minima
  - Robust to outliers



Simple Line Fitting: Least Squares

Input: Points $X = [(x_1, y_1), \ldots, (x_n, y_n)]$

Output: (Best) Line through all points $y = \hat{a}x + \hat{b}$

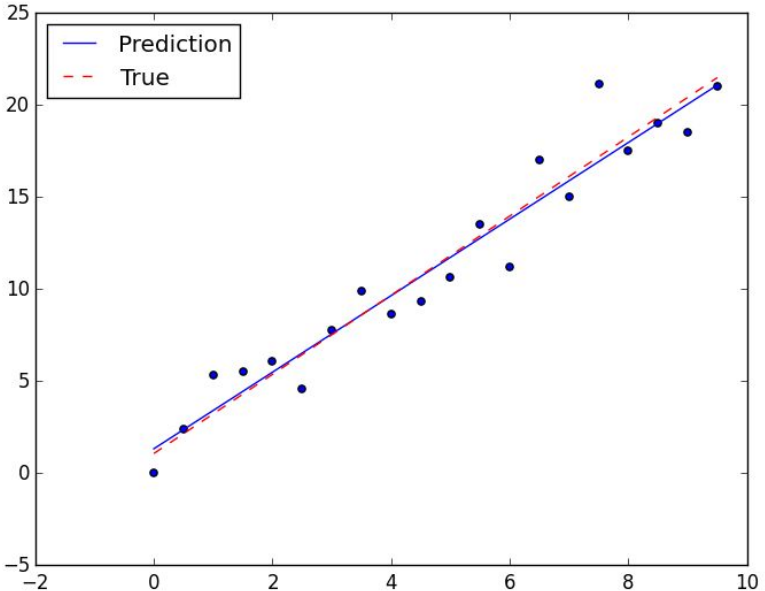
$$a, b = \text{argmin}_{a,b} E(X) = \sum_{i=1}^{n} (y_i - ax_i - b)^2$$

$$E(X) = \sum_{i=0}^{n} \left\| (x_i, 1) \begin{bmatrix} a \\ b \end{bmatrix} - y_i \right\|^2 = \|Ax - y\|^2$$

$$= y^T y - 2(Ax)^T y + (Ax)^T (Ax)$$

$$\frac{\partial E}{\partial x} = -2A^T y + 2A^T Ax = 0$$

$$A^T Ax = A^T y \Rightarrow x = (A^T A)^{-1} A^T y$$



Noise: $\sigma = 2$

Least Squares

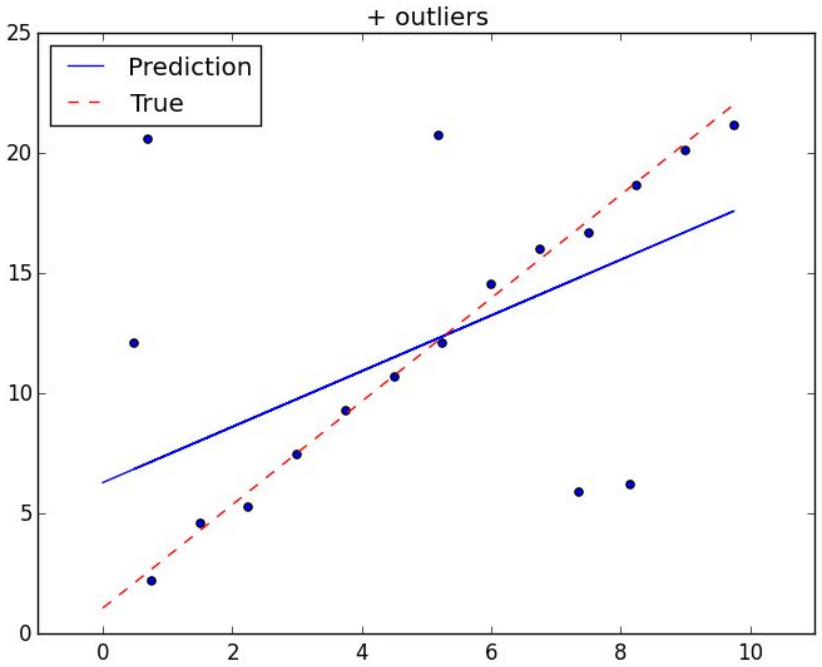
Good:

- Linear (fast, closed form)

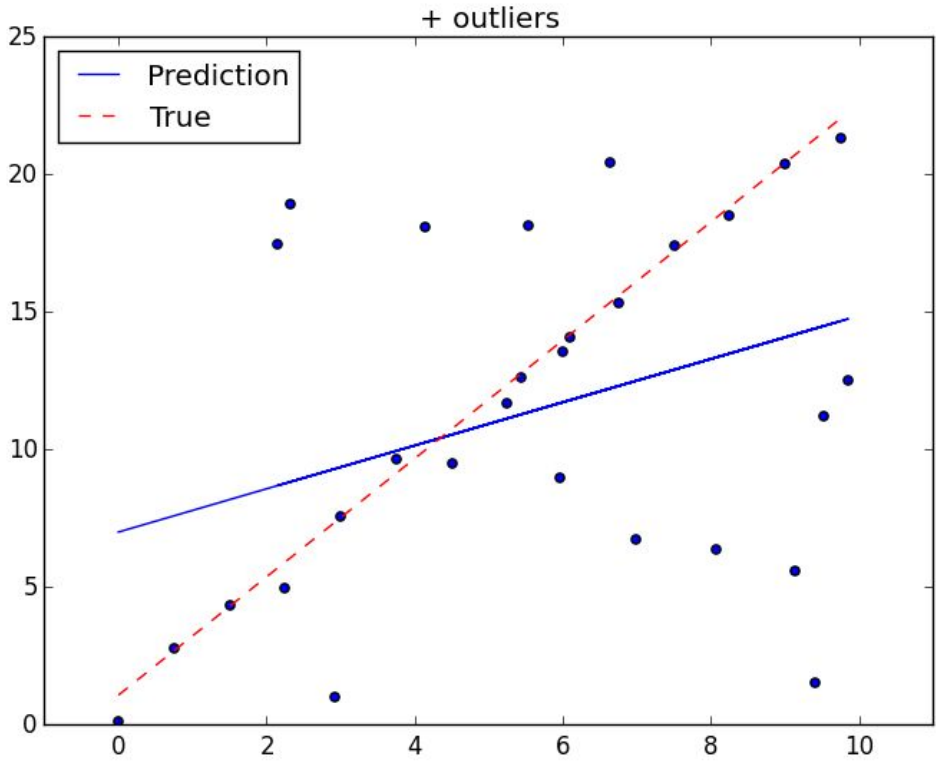
Bad:

- Sensitive to outliers (not robust)
- Only detects a single line

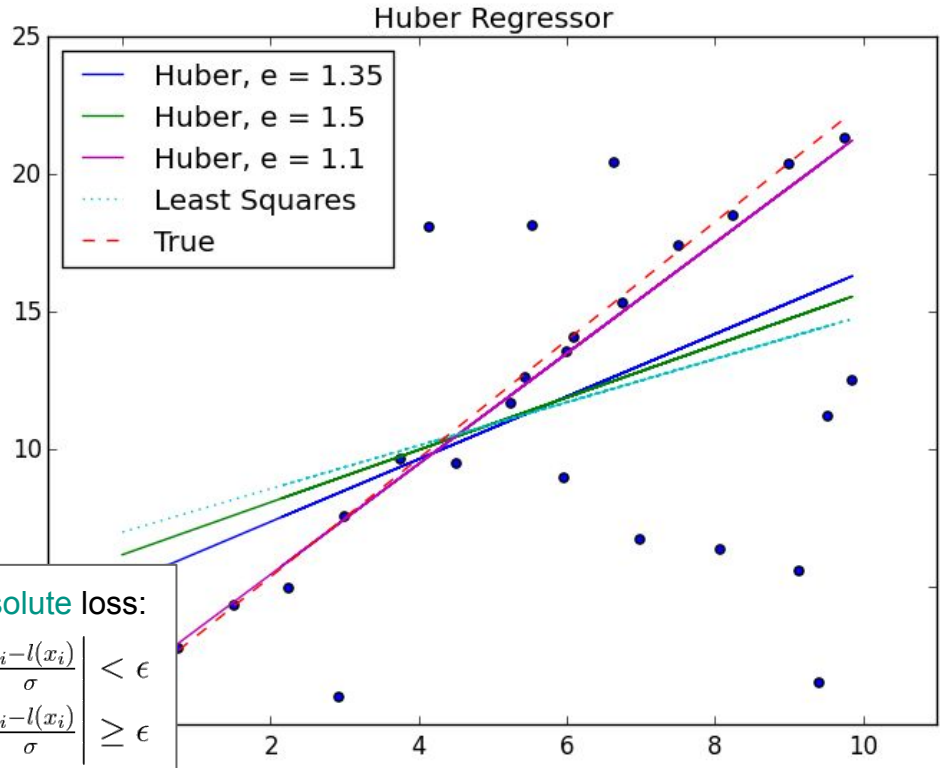
→ Use a robust estimator.



Robust Least Squares



Robust Least Squares



Choose between squared or absolute loss:

$$E(x_i) = \begin{cases} \|y_i - l(x_i)\|^2 & \text{if } \left|\frac{y_i - l(x_i)}{\sigma}\right| < \epsilon \\ |y_i - l(x_i)| & \text{if } \left|\frac{y_i - l(x_i)}{\sigma}\right| \geq \epsilon \end{cases}$$

RANSAC

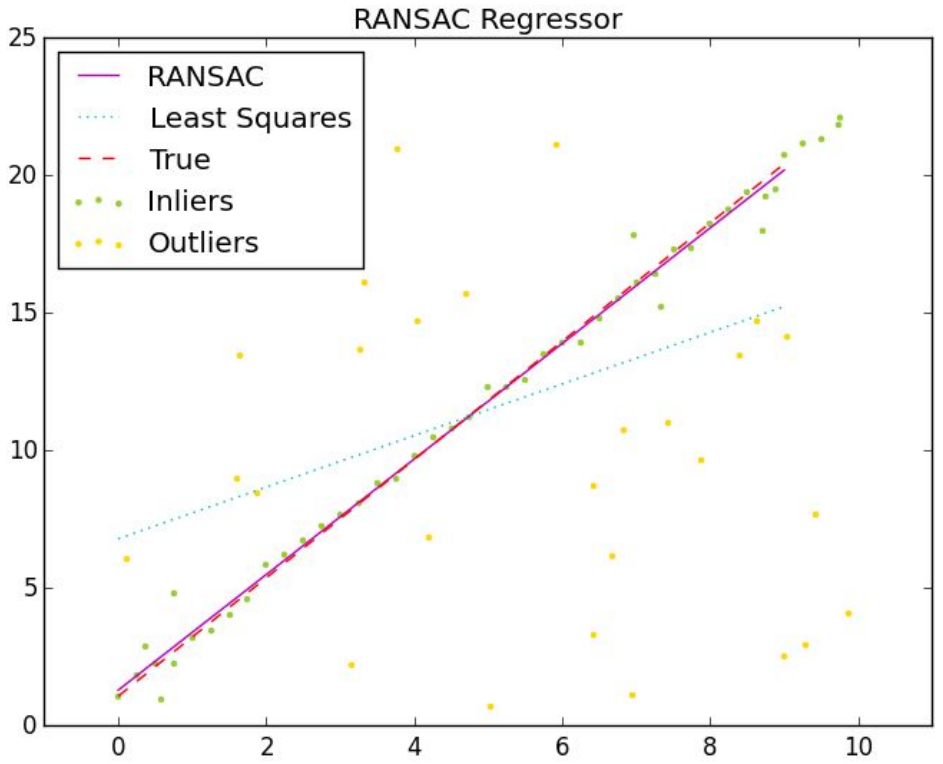
RAndom **SA**mple **C**onsensus

So far - use *all* the points for matching & minimize the effect of outliers (robust estimators)

RANSAC idea: use only a *subset* of the points, discard outliers from the model fitting.

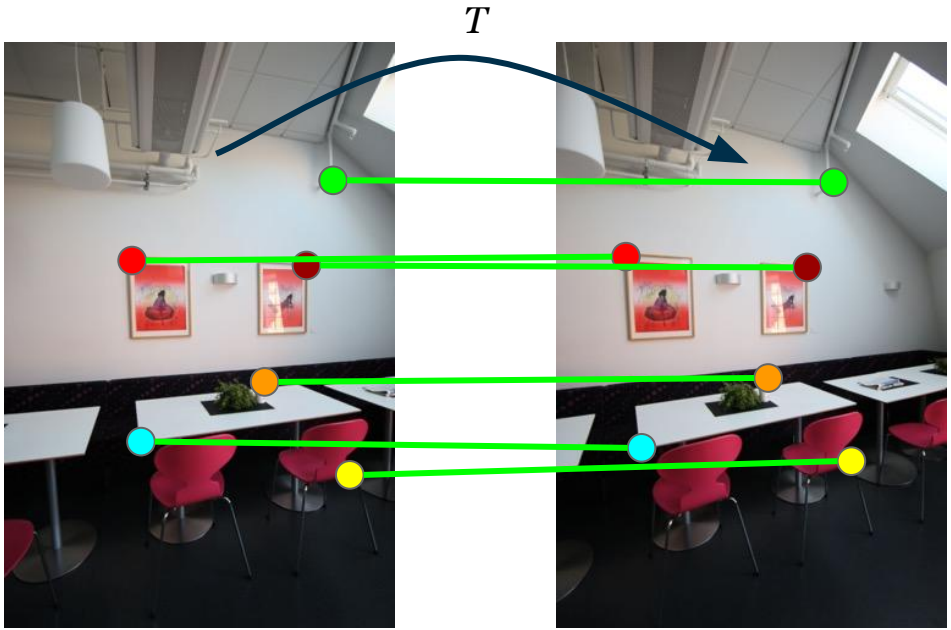
Method:

1. Choose a minimal set of samples
2. Estimate parameters (e.g. with LSq)
3. Count the number of inliers
4. Repeat 1-3 n times, pick best configuration
5. Refine the solution using only inliers



Questions?

Back to Alignment | Affine Transform



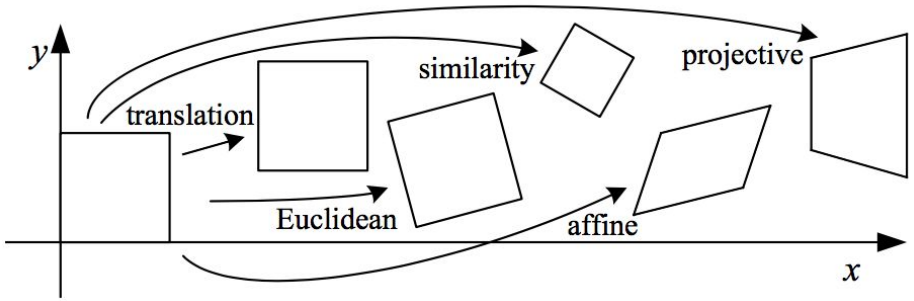
Let's assume T is Affine:

$$\begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = \begin{pmatrix} m_1 & m_2 \\ m_3 & m_4 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

Or in Homogeneous coordinates:

$$\begin{pmatrix} x_i' \\ y_i' \\ w_i' \end{pmatrix} = \begin{pmatrix} m_1 & m_2 & t_x \\ m_3 & m_4 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$$

2D Planar Transformations | Affine



Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

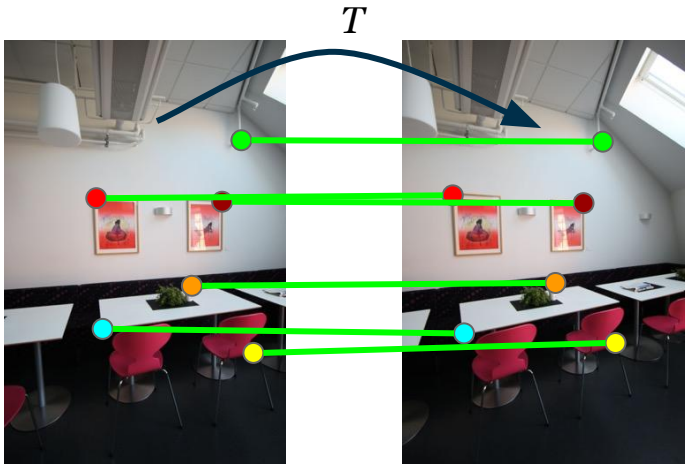
2D Affine transformation:

- Parallel lines and planes remain parallel, but not angles or lengths.
- 6 DoF: 4 for rotation, scale and shear + 2 for translation
- Still linear → we can use Linear Least Squares to solve, and possibly add robustness to outliers

Affine Alignment

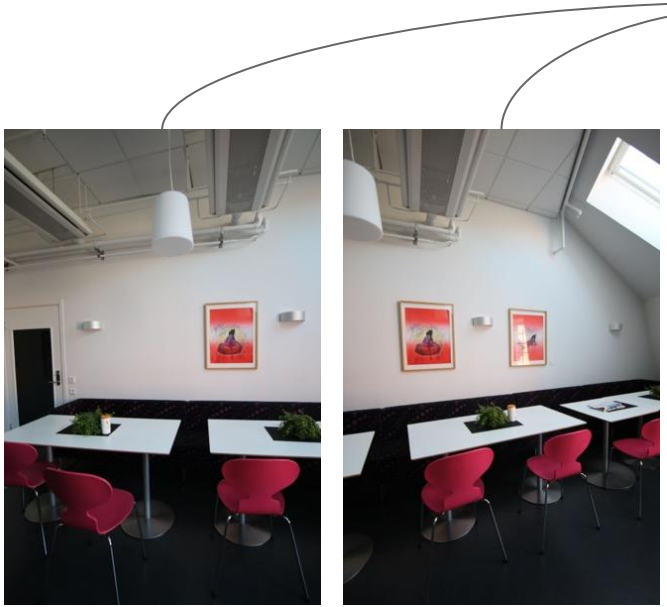
$$E(X) = \sum_{i}^{n} \left\| \begin{pmatrix} x'_i \\ y'_i \end{pmatrix} - \begin{pmatrix} m_1 & m_2 \\ m_3 & m_4 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} t_x \\ t_y \end{pmatrix} \right\|_{L_2}$$

$$= \left\| \begin{pmatrix} \cdots & & & & & \\ x_i & y_i & 0 & 0 & 1 & 1 \\ 0 & 0 & x_i & y_i & 1 & 1 \\ \cdots & & & & & \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{pmatrix} - \begin{pmatrix} \cdots \\ x'_i \\ y'_i \\ \cdots \end{pmatrix} \right\|_{L_2}$$



A non-homogeneous system of linear equations, with 6 DoF.

Affine Alignment | Result



Somewhat aligned,
But it will simply not do.

Homographies

What can we do besides Affine? a Homography.

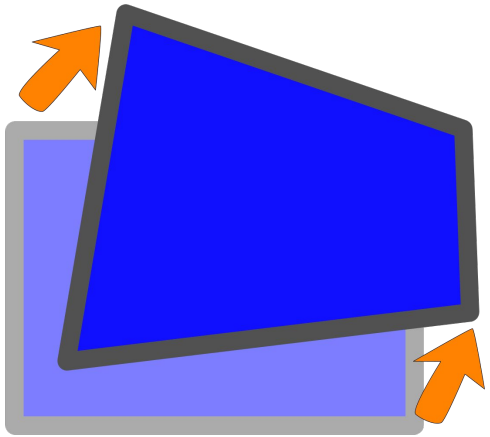
- It does not preserve parallel lines.
- Still linear - we can use the same exact fitting process.

$$\begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} \frac{h_1x+h_2y+h3}{h_7x+h_8y+1} \\ \frac{h_4x+h_5y+h6}{h_7x+h_8y+1} \\ 1 \end{pmatrix}$$

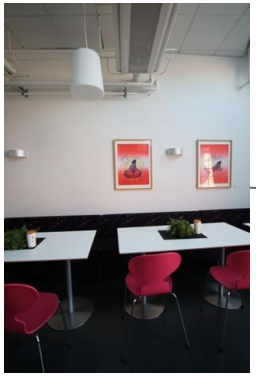
$$H = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{pmatrix}$$

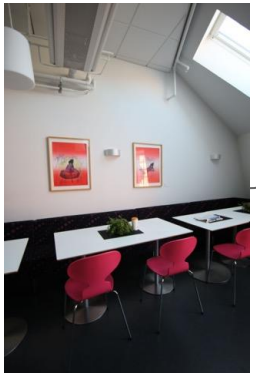
We added two more elements - what do they do?

Allow us to model perspective transformation.

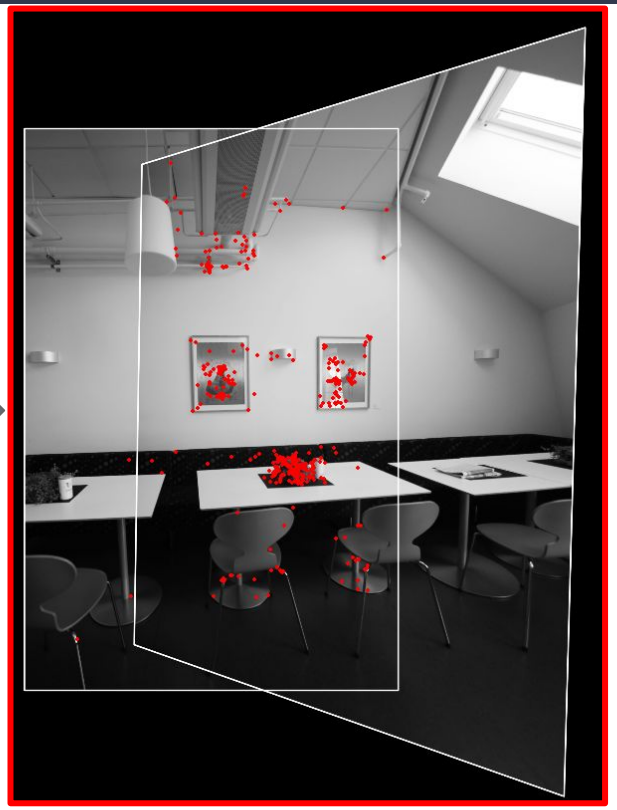


Homography-based Alignment





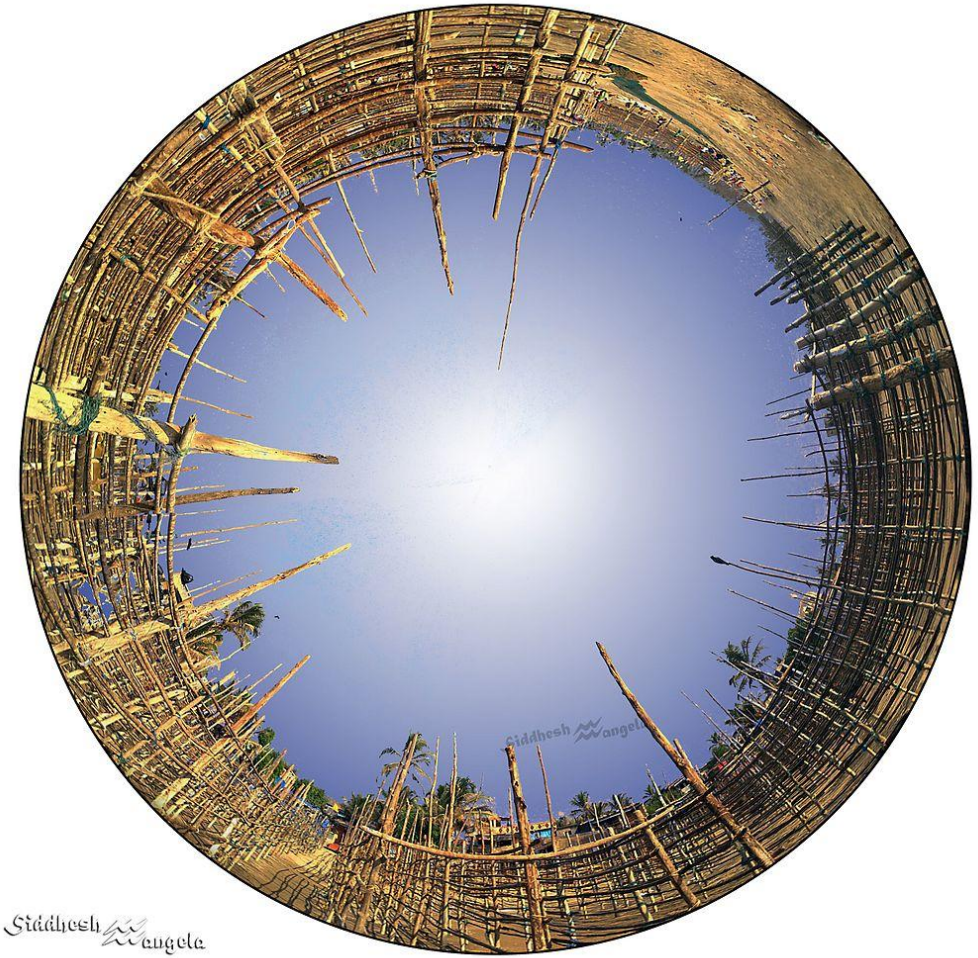
Affine? Meh.

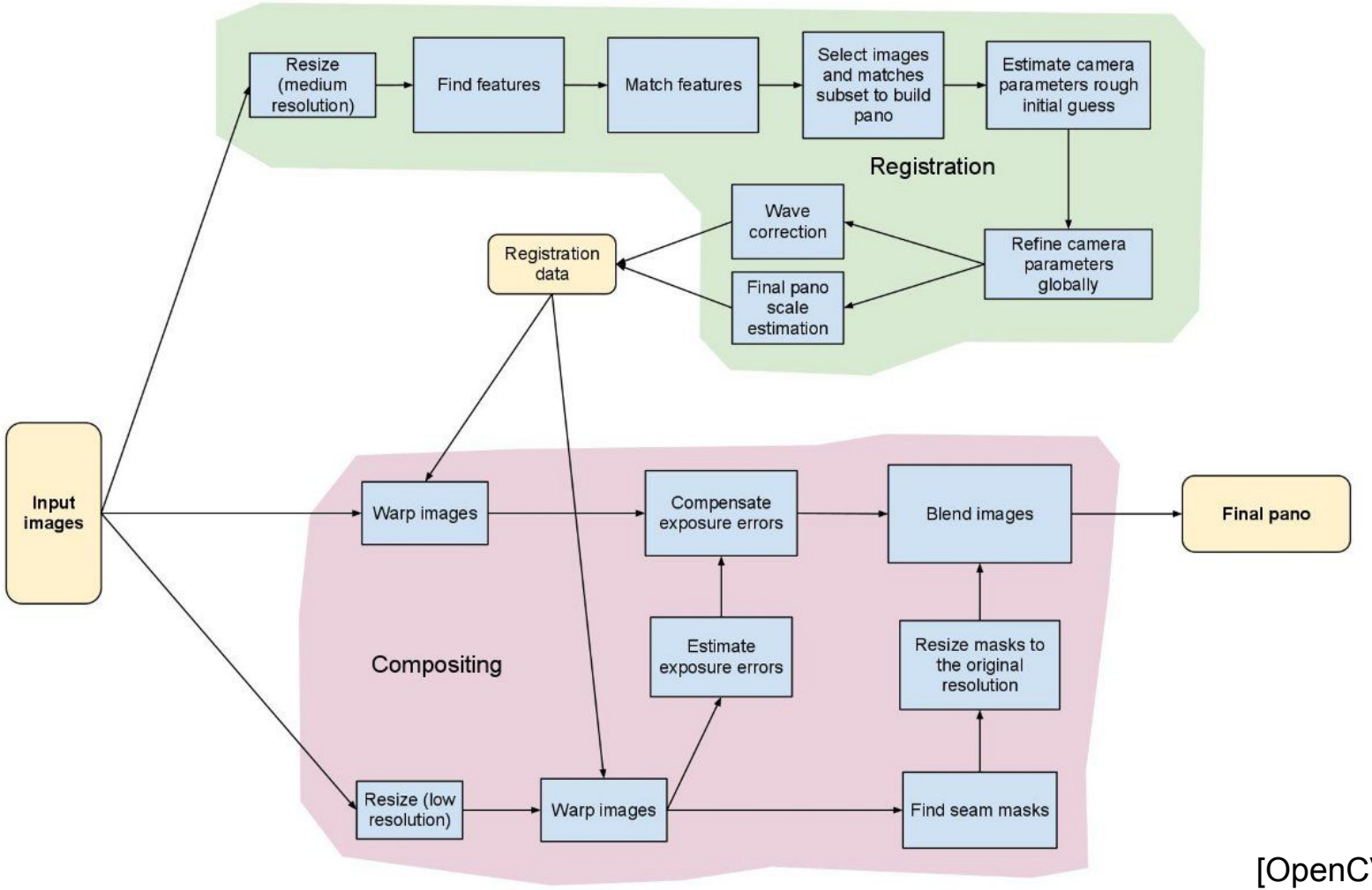


Homography!
Yay!

Questions?

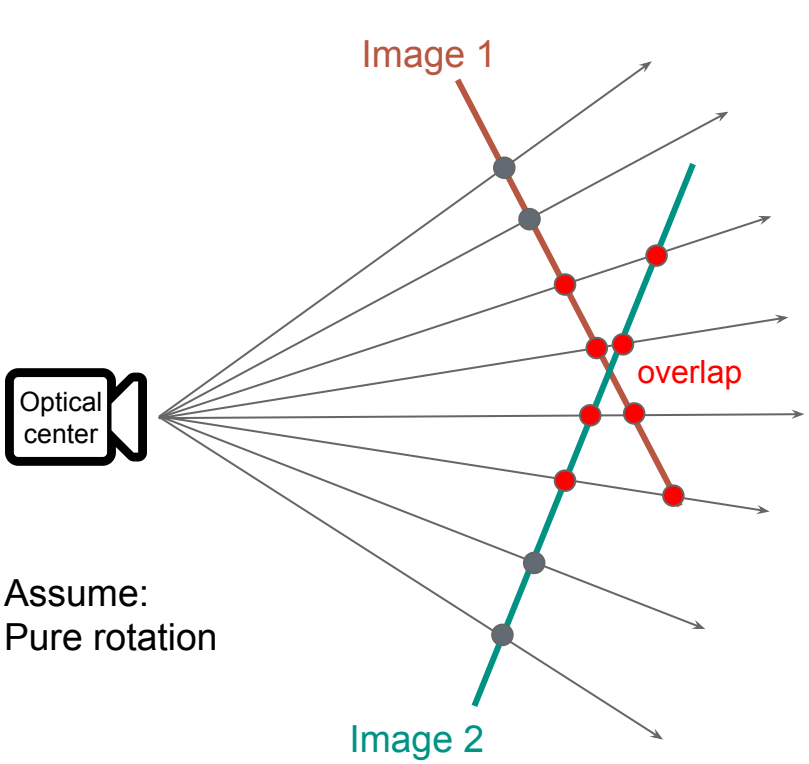
Panoramas




Resize (medium resolution)
Find features
Match features
Select images and matches subset to build pano
Estimate camera parameters rough initial guess
Registration
Wave correction
Refine camera parameters globally
Registration data
Final pano scale estimation
Input images
Warp images
Compensate exposure errors
Blend images
Final pano
Compositing
Estimate exposure errors
Resize masks to the original resolution
Resize (low resolution)
Warp images
Find seam masks

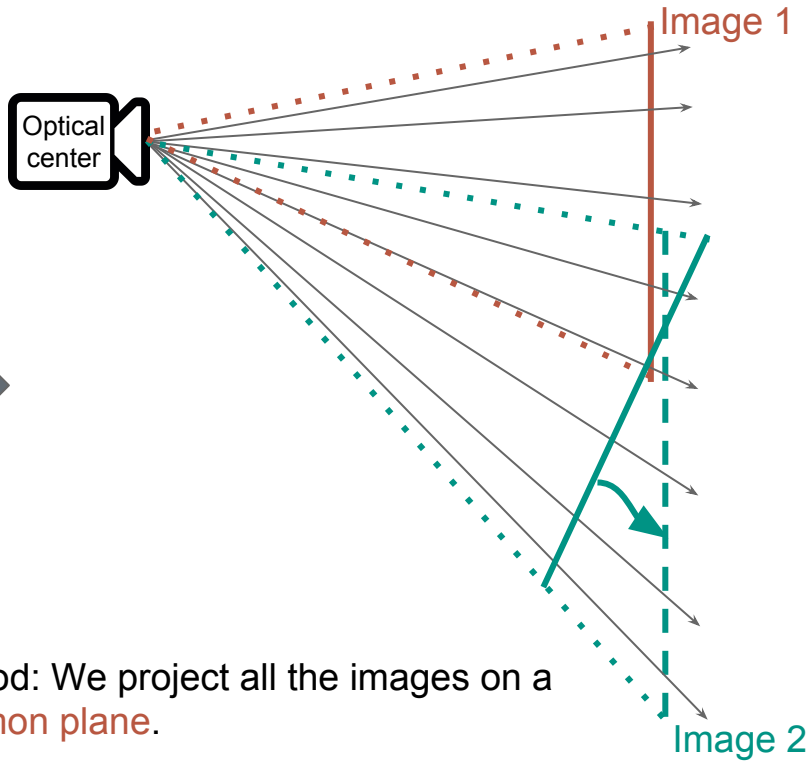
[OpenCV]

Mosaics: Geometry



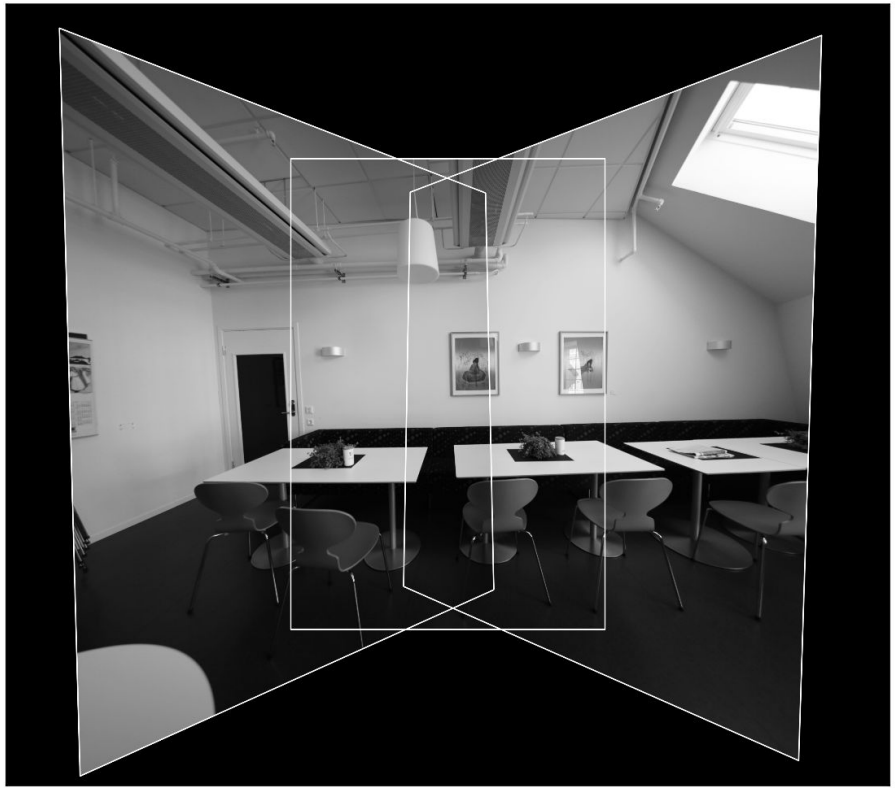
Assume:
Pure rotation

Projection



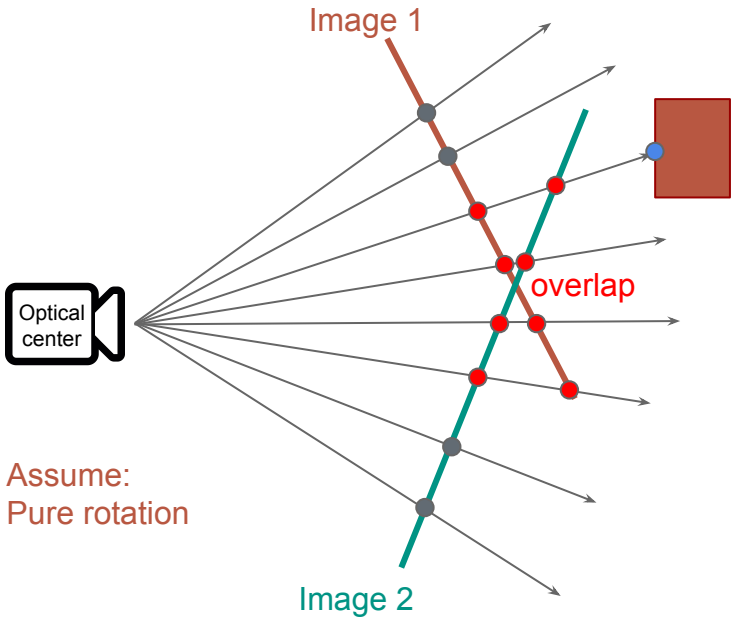
Method: We project all the images on a common plane.

Projective Panorama



Panorama Geometry

What happens to the overlapping points?



Assume:
Pure rotation

In camera 2: $\begin{pmatrix} X_2 \\ Y_2 \\ Z_2 \end{pmatrix} = K_2^{-1} \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix}$

Move to camera 1: $\begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \end{pmatrix} = \underbrace{R_1 R_2^T K_2^{-1}}_{\text{Undo 2, Apply 1}} \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix}$

Project on image 1: $\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} \approx \underbrace{K_1 R_1 R_2^T K_2^{-1}}_{\text{3x3 Homography}} \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix}$

⇒ Homography is:

(1) Undo projection 2,
(2) rotate from cam 2 to cam 1,
(3) apply projection 1

A 360° Panorama With Homographies?

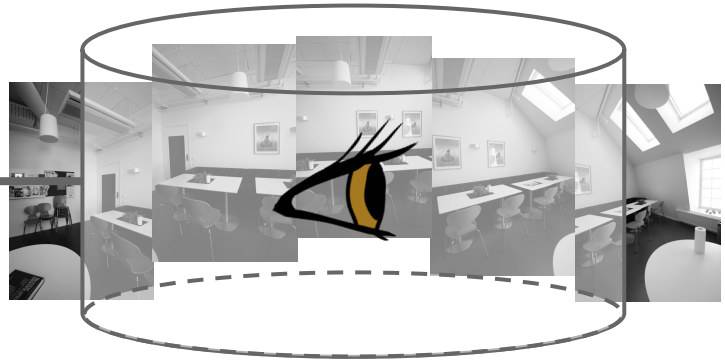
eek.

Not really.

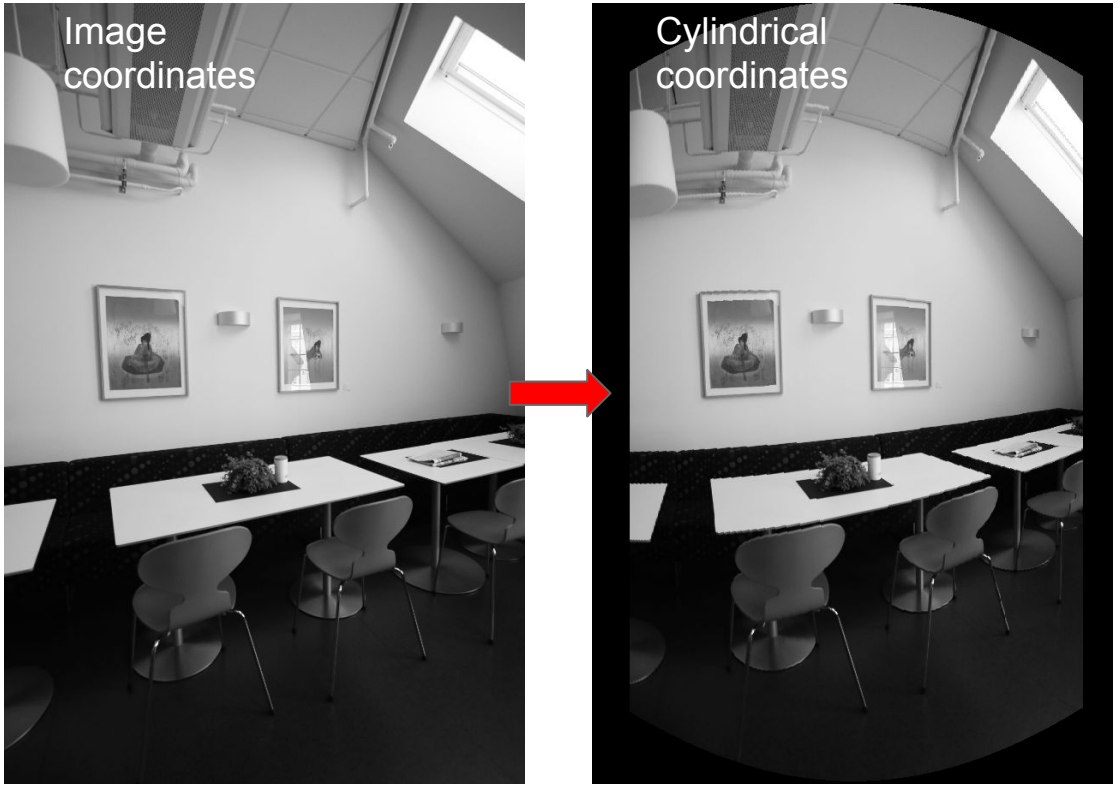
360° Panoramas

View from inside the cylinder.

Imagine the images are on the wall of a cylinder...



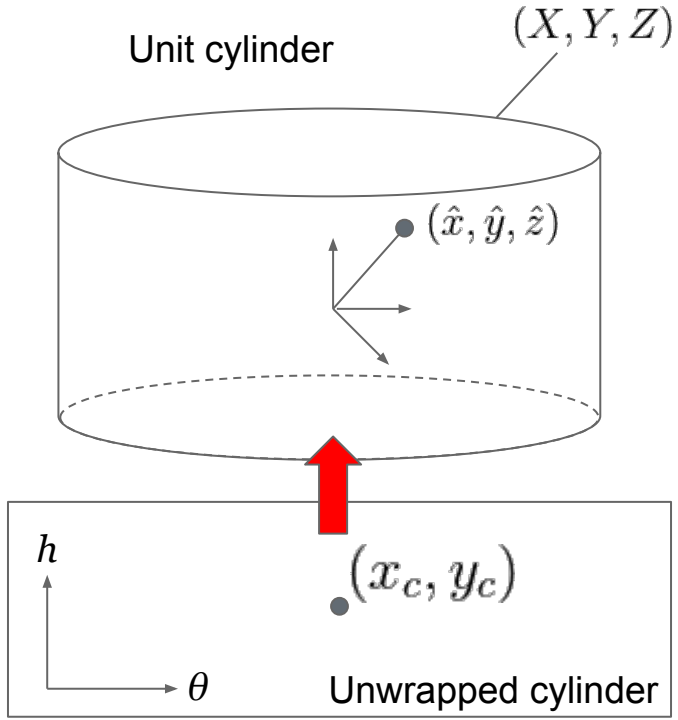
A 360° Panorama | Cylindrical Warping



A 360° Panorama | Cylindrical Warping

Easier to go the inverse, from cylinder to image:

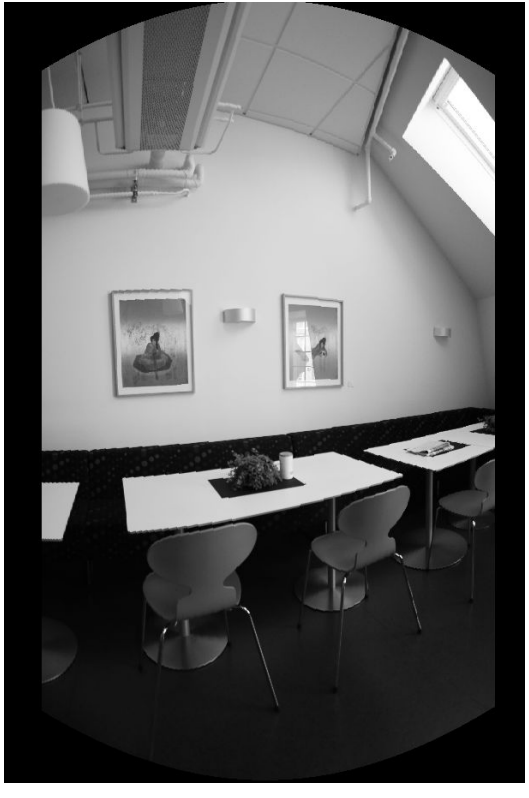
$$\theta = (x_{cyl} - x_c)/f$$
$$h = (y_{cyl} - y_c)/f$$
$$\hat{x} = \sin\theta$$
$$\hat{y} = h$$
$$\hat{z} = \cos\theta$$



Aligning in Cylindrical Coordinates

We rotated the camera by θ.

How does this affect the cylindrical image?



Aligning in Cylindrical Coordinates

We rotated the camera by θ.

How does this affect the cylindrical image?

It's a (simple) translation.

Aligning cylindrical images needs only a translational model of 2 DoF.

How to Cut?

… Use Spherical Coordinates

Questions?

HW2: Image Stitching

Your goal is to create 2 panoramas:

1. Using homographies and perspective warping on a common plane (3 images).
2. Using cylindrical warping (many images).

In both options you should:

- Extract features and descriptors
- Match features
- Calculate the best model parameters
- Bonus (!!): Use your Laplacian Blending code to stitch the images together nicely

You will be given sample code to

- convert an image to cylindrical coordinates
- calculate affine or homography transform
- skeleton panorama maker

The rest is up to you...