INF 5300 - 02.04.2014

Feature-based alignment

Anne Schistad Solberg

- Finding the alignment between features from different images
- Geometrical transforms – short repetition
- RANSAC algorithm for robust transform computation

Curriculum

- Background in geometrical transforms: Read e.g. 2.1.1 and 2.1.2 in Szeliski
- Section 6.1 i Szeliski
- Recommended additional reading:
  - Ransac is not described in detail in the book, you can find more detailes in:
  - *Ransac for Dummies: vision.ece.ucsb.edu/~zuliani/.../**RANSAC**/docs/**RANSAC**4**Dummies**.pdf*
  - Ransac Toolbox for Matlab: git://github.com/RANSAC/RANSAC-Toolbox.git

From last lecture: Image matching

- How do we compute the correspondence between these images?
  - Extract good features for matching (last lecture)
  - Estimation geometrical operation for match (this lecture)

•by Diva Sian

•by swashford

From last lecture: Scale-invariant features (SIFT)

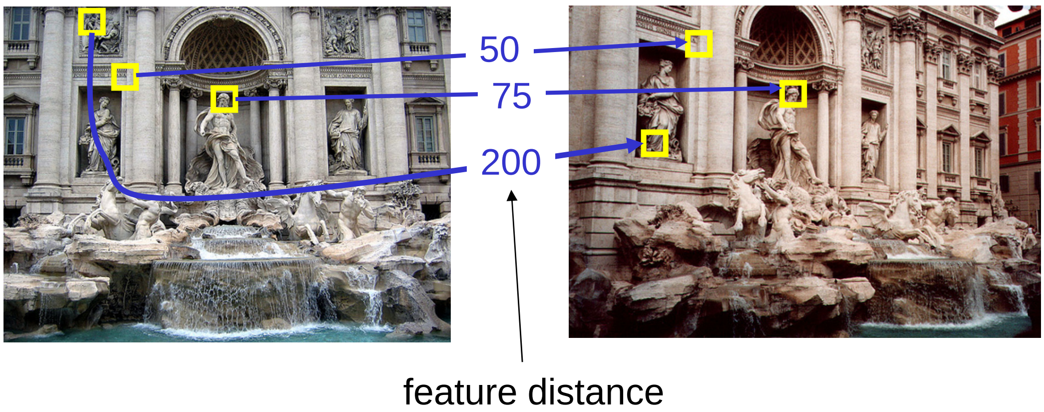
- See Distinctive Image Features from Scale-Invariant Keypoints by D. Lowe, International Journal of Computer Vision, 20,2,pp.91-110, 2004.
- Invariant to scale and rotation, and robust to many affine transforms.
- Main components:
  1. Scale-space extrema detection – search over all scales and locations.
  2. Keypoint localization – including determining the best scale.
  3. Orientation assignment – find dominant directions.
  4. Keypoint descriptor - local image gradients at the selected scale, transformed relative to local orientation.

From last lecture: SIFT: feature matching

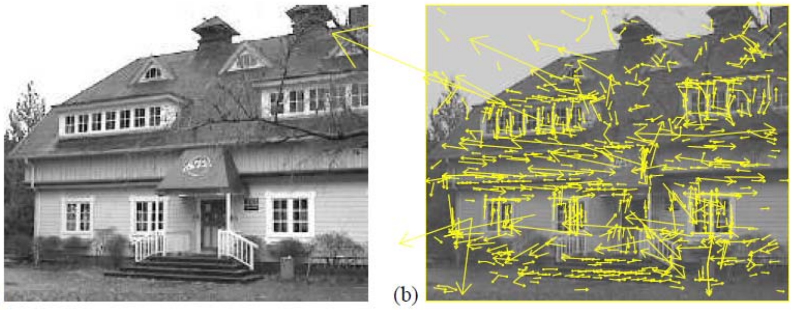
- Compute the distance from each keypoint in image A to the closest neighbor in image B.
- We need to discard matches if they are not good as not all keypoints will be found in both images.
- A good criteria is to compare the distance between the closest neighbor to the distance to the second-closest neighbor.
- A good match will have the closest neighbor should be much closer than the second-closest neighbor.
- Reject a point if closest-neighbor/second-closest-neighbor >0.8.

Results from last lecture – feature detecting and matching

A set of keypoints are detected and matched in two images



Starting point for this lecture

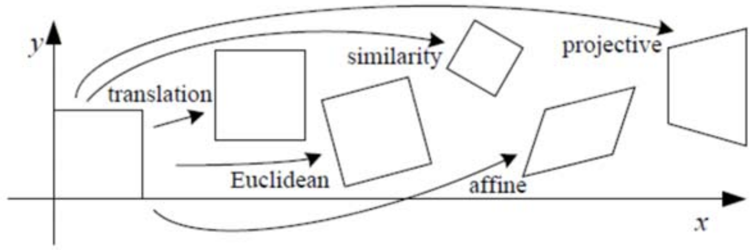

(a) (b)

- A set of corresponding feature points in two images.
- Goal: estimate the geometrical transform that we need to align the two images.
- Problem: movements are noisy and establishing ONE geometric transform for the image is difficult.

Goal of this lecture

- Consider two images containing partly the the same objects but at different times, from different sensors, or from different views.
- Assume that a set of features has been detected and the matching between corresponding features determined.
- Now we need to:
  - Verify that the mathing is geometrically consistent
  - This is the case if we can compute the motion between the features using a simple 2D or 3D geometric transform
  - How do we do this robustly?

2D and 3D feature-based alignment



- We restrict us to parametric transforms such as the ones illustrated above.

Simple operations:

- Translation
- Euclidean = translation + rotation
- Affine transforms
- Similarity = scaled rotation
- Projection

INF 2310 - Geometrical operations

- Transform the pixel coordinates (x,y) to (x',y'):

$$x' = T_x(x,y)$$
$$y' = T_y(x,y)$$

- The transforms T_x og T_y are often given as transforms.

2D coordinate transformations

- translation: $\boldsymbol{x'} = \boldsymbol{x} + \boldsymbol{t}$ $\boldsymbol{x} = (x,y)$
- rotation: $\boldsymbol{x'} = \boldsymbol{R}\,\boldsymbol{x} + \boldsymbol{t}$
- similarity: $\boldsymbol{x'} = s\,\boldsymbol{R}\,\boldsymbol{x} + \boldsymbol{t}$
- affine: $\boldsymbol{x'} = \boldsymbol{A}\,\boldsymbol{x} + \boldsymbol{t}$
- perspective: $\underline{\boldsymbol{x}}' \cong \boldsymbol{H}\,\underline{\boldsymbol{x}}$ $\underline{\boldsymbol{x}} = (x,y,1)$
  ($\underline{\boldsymbol{x}}$ is a *homogeneous* coordinate (expanded for convenient notation)

INF 2310: Affine transforms

- Affine transforms are described by:

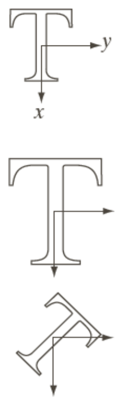
$$x' = a_0x+a_1y+a_2$$
$$y' = b_0x+b_1y+b_2$$

- Matrix form:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \text{eller} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_0 & a_1 \\ b_0 & b_1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_2 \\ b_2 \end{bmatrix}$$

INF 2310 - Examples of simple transforms

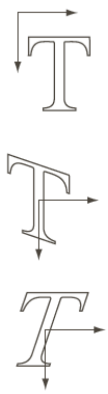
Transform	a_0	a_1	a_2	b_0	b_1	b_2	Expression
Identity	1	0	0	1	0	0	x'=x y'=y
Scalie factor s	s	0	0	0	s	0	x'=sx y'=sy
Rotation by θ	$\cos\theta$	$-\sin\theta$	0	$\sin\theta$	$\cos\theta$	0	x'=$\cos\theta$x-$\sin\theta$y y'= $\sin\theta$y+ $\cos\theta$y



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

INF 2310 – More examples

Transform	a_0	a_1	a_2	b_0	b_1	b_2	Expression
Translation by Δx og Δy	1	0	Δx	0	1	Δy	x'=x+Δx y'=y+Δy
Horisontal "shear" factor s_1	1	s1	0	0	1	0	x'=x+s1y y'=y
Vertical "shear" factor s2	1	0	0	s2	1	0	x'=x y'=s2x+y



Vertikalt

Horisontalt

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

INF 2310 - Combinations of affine transforms

$$\begin{bmatrix} & & \\ & transl. & \\ & & \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \qquad \begin{bmatrix} & & \\ & rot & \\ & & \end{bmatrix}\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} & & \\ & rot & \\ & & \end{bmatrix}\begin{bmatrix} & & \\ & transl. & \\ & & \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} & & \\ & transl.\,\&\,rot & \\ & & \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix}$$

INF 2310 - Higher order transforms

- Bilinear transforms:

  $x' = a_0x+a_1y+a_2+a_3xy$

  $y' = b_0x+b_1y+b_2+b_3xy$

- Quadratic transforms:

  $x' = a_0x+a_1y+a_2+a_3xy+a_4x^2+a_{5y}y^2$

  $y' = b_0x+b_1y+b_2+b_3xy+b_4x^2+b_{5y}y^2$

- Higher order polynomials can also be used

2D Transform equations

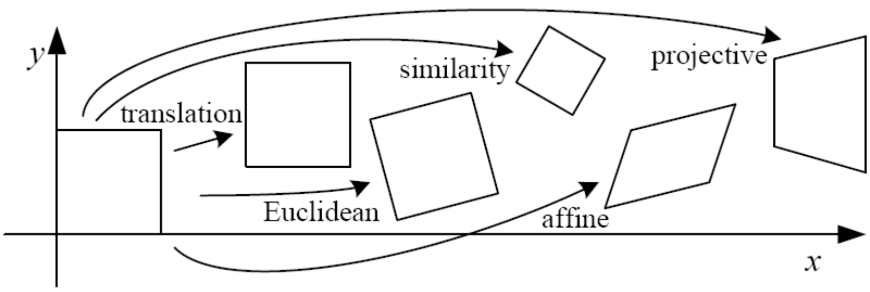
Transform	Matrix	Parameters p	Jacobian J
translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$	(t_x, t_y)	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Euclidean	$\begin{bmatrix} c_\theta & -s_\theta & t_x \\ s_\theta & c_\theta & t_y \end{bmatrix}$	(t_x, t_y, θ)	$\begin{bmatrix} 1 & 0 & -s_\theta x - c_\theta y \\ 0 & 1 & c_\theta x - s_\theta y \end{bmatrix}$
similarity	$\begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}$	(t_x, t_y, a, b)	$\begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$
affine	$\begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}$	$(t_x, t_y, a_{00}, a_{01}, a_{10}, a_{11})$	$\begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix}$
projective	$\begin{bmatrix} 1+h_{00} & h_{01} & h_{02} \\ h_{10} & 1+h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix}$	$(h_{00}, h_{01}, \ldots, h_{21})$	(see Section 6.1.3)

Table 6.1 Jacobians of the 2D coordinate transformations $\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{p})$ shown in Table 2.1, where we have re-parameterized the motions so that they are identity for $\boldsymbol{p} = 0$.

Projective Transformations

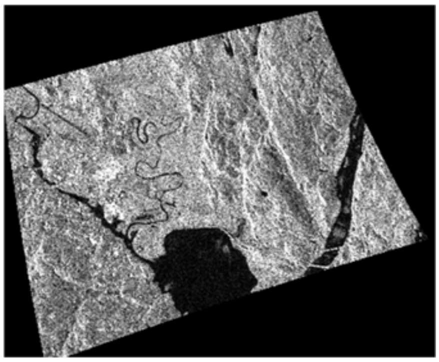
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Projective transformations:
  - Affine transformations, and
  - Projective warps
- Parallel lines do not necessarily remain parallel



From INF 2310: Image co-registration

Original



Transformert

Image to co-register with

INF 2310 - coregistration III

- The root mean square error is used to evaluate how good a match is
- Given M point pairs $(x_i,y_i),(x_i^r,y_i^r)$ (r is the reference image)
- Assume that the transform gives estimated coordinates in the reference image as (x'_i,y'_i)
- (x_i,y_i) --> (x'_i,y'_i)
- The number of point pairs is M >>3 for affine transforms og M>>6 for quadratic
- The coefficients in the transform are computed as the values that minimize the square error between the true coordinates
- (x_i^r,y_i^r) and the transformed coordinates (x_i',y_i')

$$J = \sum_{i=1}^{M} (x_i' - x_i^r)^2 + (y_i' - y_i^r)^2$$

- Simple linear algebra is used to find the solution to this problem.

INF 2310 – Mean square error

$$J = \sum_{i=1}^{M} (x_i' - x_i^r)^2 + (y_i' - y_i^r)^2 = J_x + J_y \qquad J_x = \sum_{i=1}^{M} (x_i' - x_i^r)^2$$

Note that this is based on a linear relationship between the estimated and true coordinates.

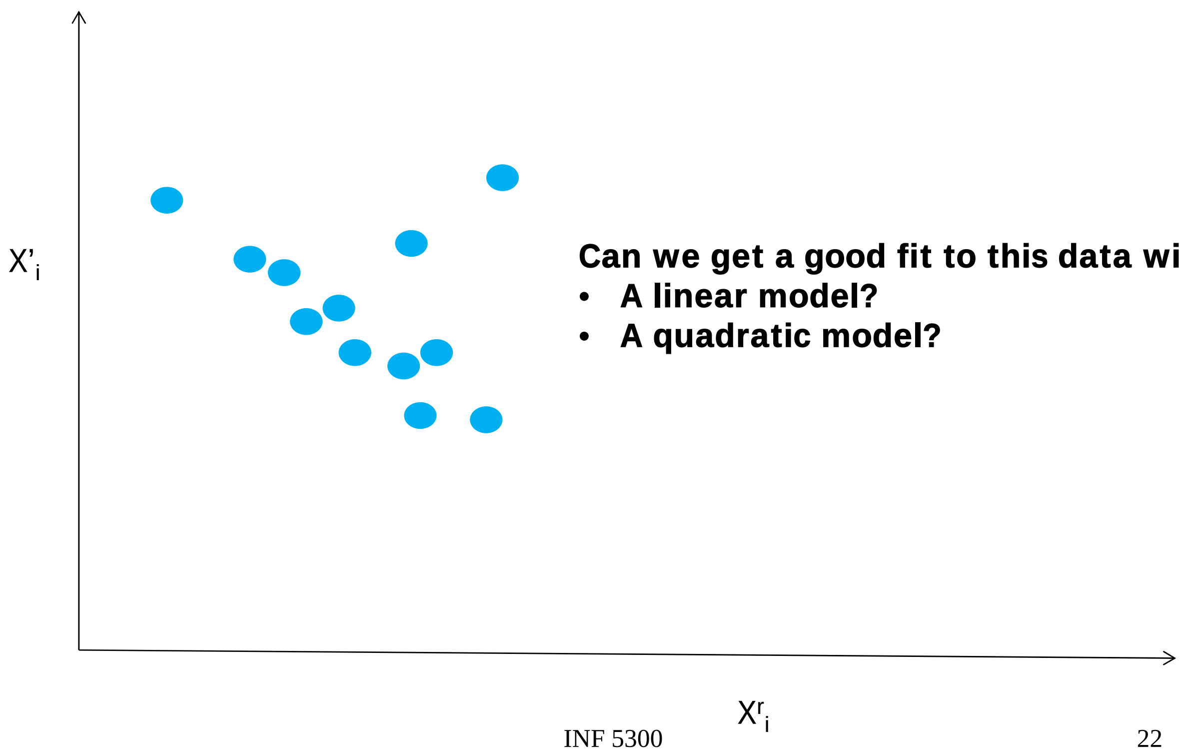
$$\overbrace{\begin{bmatrix} x_1^r \\ x_2^r \\ \vdots \\ x_n^{r'} \end{bmatrix}}^{d} = \overbrace{\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix}}^{G} \overbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}}^{a}$$

Find a that minimize the error

$$J_x = (d - Ga)^T (d - Ga) = d^T d + a^T G^T G a - 2 a^T G^T d$$

$$\frac{\delta J_x}{\delta a^T} = 2G^T G a - 2G^T d = 0 \quad \Rightarrow \quad \boxed{a = (G^T G)^{-1} G^T d}$$

A data example
Estimated vs. true coordinates



Can we get a good fit to this data with
- **A linear model?**
- **A quadratic model?**

Limitations of least squares matching (LSM)

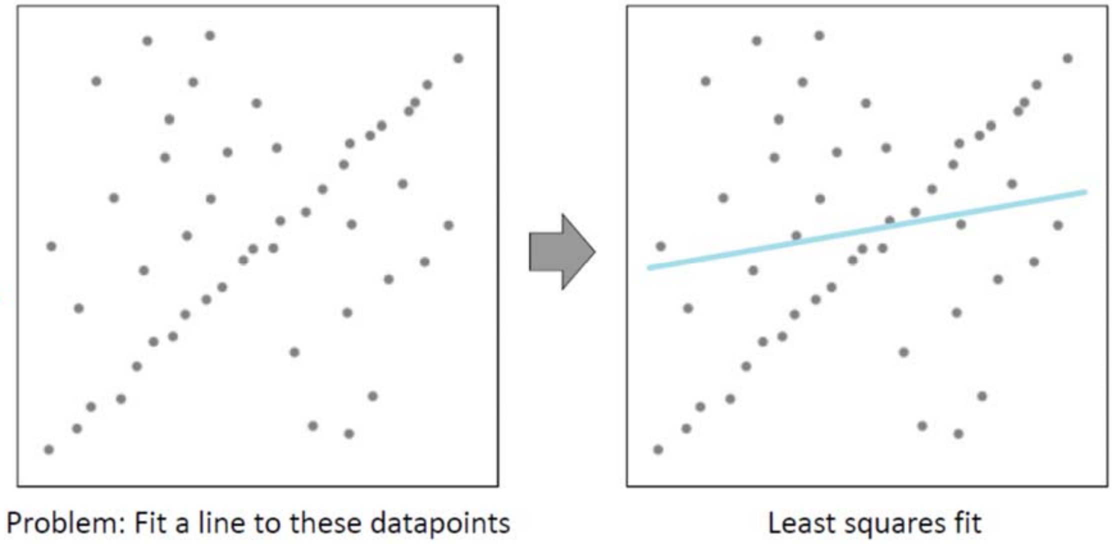
- LSM matching assumes that all feature points are matched with the same accuracy. This is normally not the case.
  - Possible solution: weighted least squares, where each points is weighted by an uncertainty measure:

$$E_{WLS} = \sum_i \sigma_i^2 \left\| x_i - x_i' \right\|^2$$

- LSM assumes a linear relationship between the measurements and the unknowns. This is also often not the case.
  - An alternative is non-linear least squares which uses iterative algorithms (6.1.3). We will not go through this.

Robustness of matching

Robustness in data fitting



Is this a good fit?

Introducing a robust matching algorithm

- The detected features are not perfect, there may be outliers where the match is NOT good.
- If we want to fit a line:
  – Count the number of points that agree with the line.
    - Agree means that the distance between the location of the estimated and the true coordinates is very small.
    - Points which fulfill this criterion are called inliers.
    - Other points are called outliers.
  – For all possible lines, select the one with the larges number of inliers.

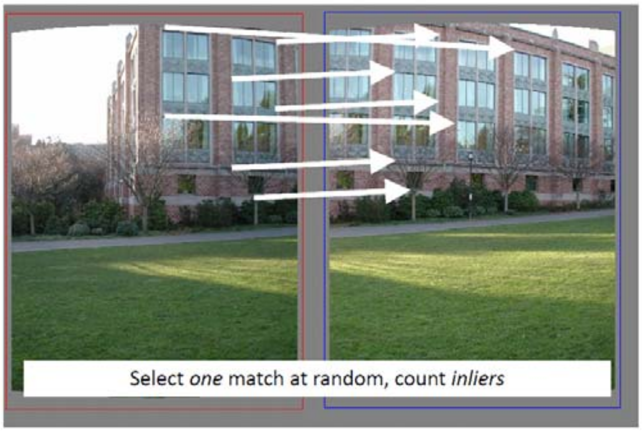
How do we find the best line?

- Unlike least-squares, there is no simple closed-form solution.

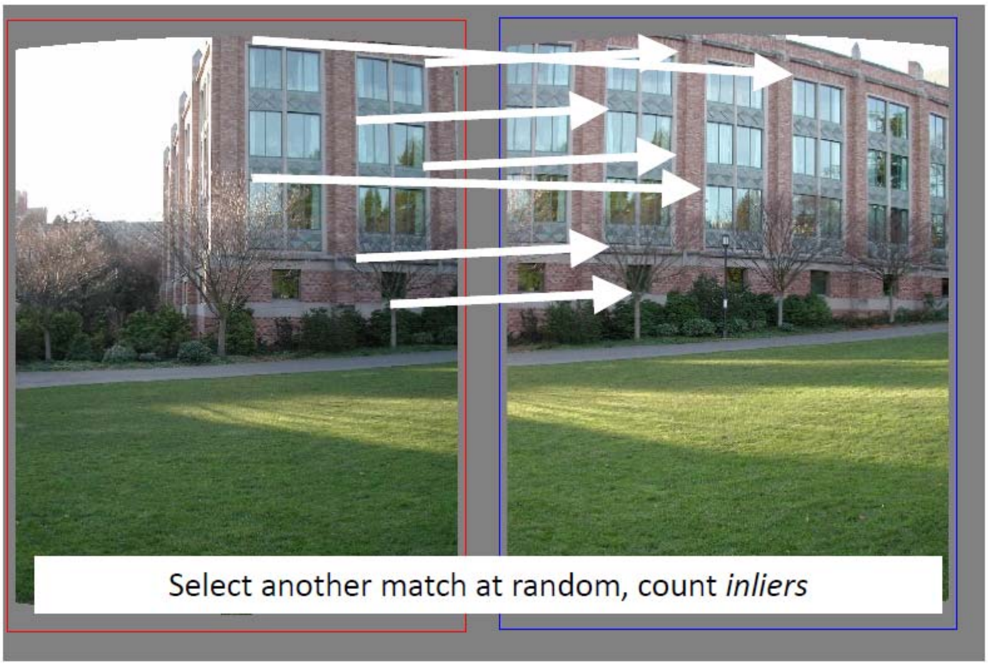
- Trial-and-test:
  - Try out many lines, keep the best one

<u>RAN</u>dom **<u>S</u>**ample **<u>C</u>**onsensus

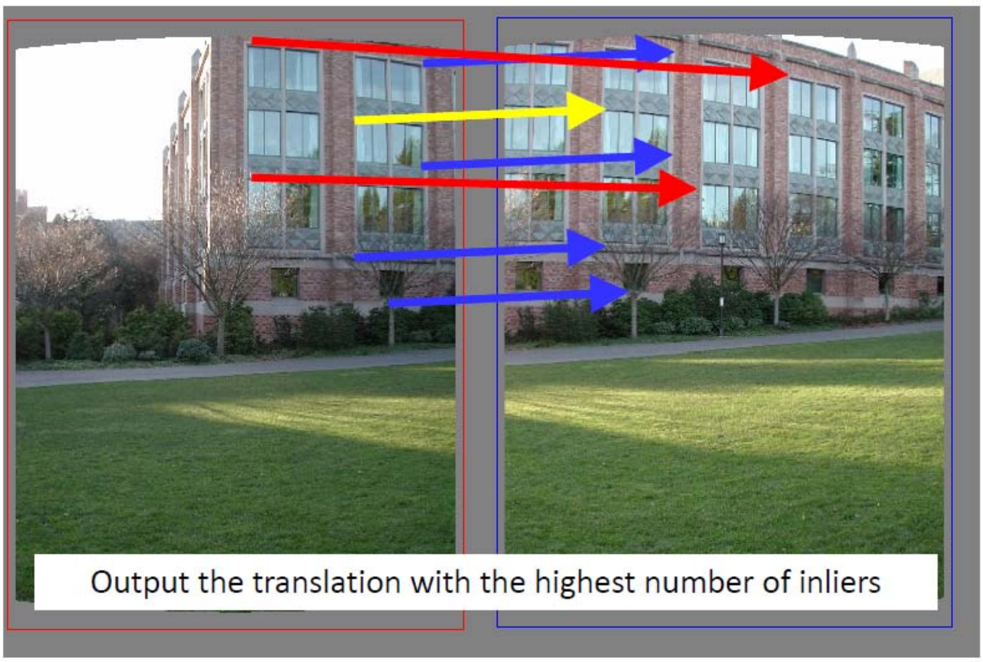
- In this example: Linear model, two points needed to get a fit.
- Select two points at random, compute the transform coefficients.
- Try this model for all other samples and count the number of inliers among the other samples.



RANdom Sample Consensus

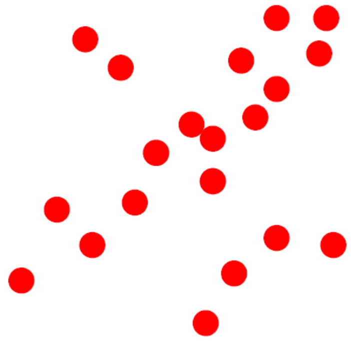


RANdom Sample Consensus



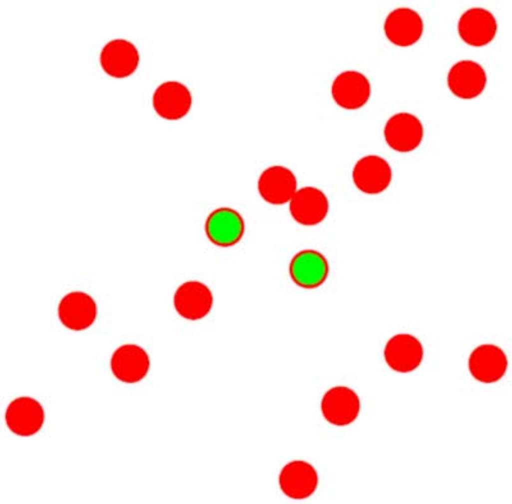
RANSAC

- RANdom Sample Consensus (Fischler and Bolles, 1981)
- Algorithm:
  1. Sample (randomly) exactly the number of points needed to fit the model.
  2. Solve for the model parameters based on the samples.
  3. Score by the fraction of inliers within a preset threshold.
- Repeat 1-3 until the best model is found with high confidence.



RANSAC

Line fitting example



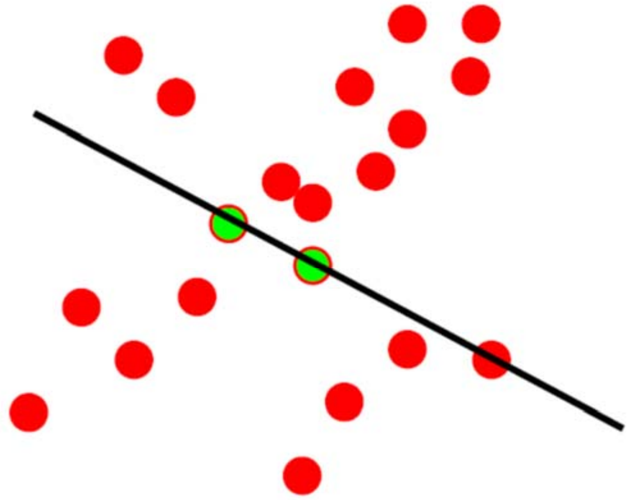
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



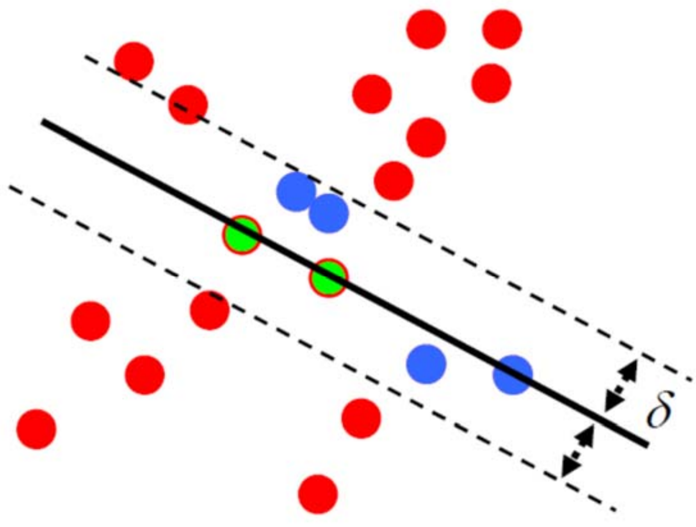
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



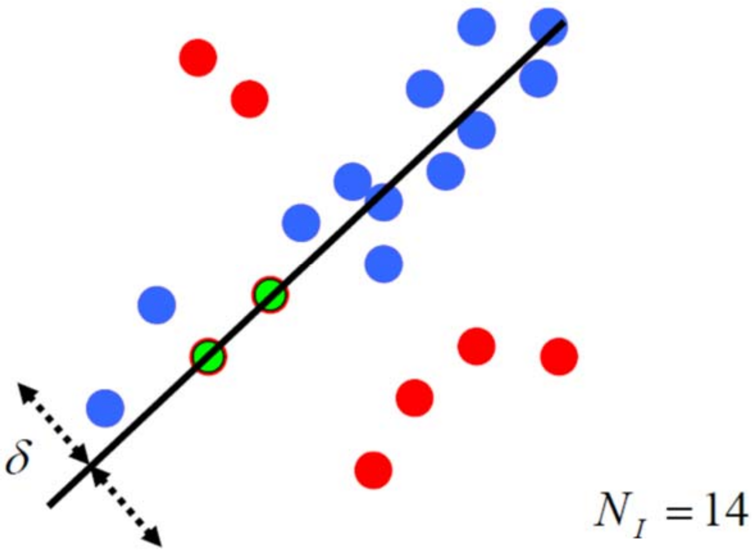
$$N_I = 6$$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

- The inlier threshold is related to the amount of noise we expect in the inliers.
- Assume Gaussian noise with a given standard deviation (usually set in pixels, e.g. 3 pixels)
- The algorithm should terminate when the probability of finding a better consensus set (higher number of inliers) is lower than a certain threshold.
  - More on this shortly

RANSAC algorithm

General version:

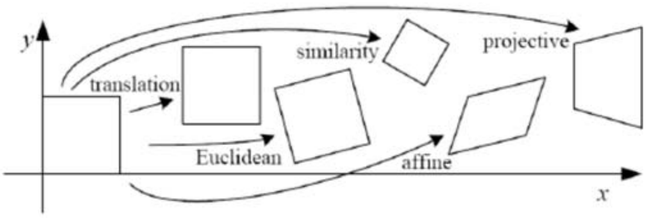
1. Randomly choose s samples

   s=minimum sample size that let you fit a model

2. Fit a model (e.g. line) to those samples
3. Count the number of inliers that approximately fit the model.
4. Repeat N times
5. Choose the model that has the largest set of inliers, and fit this model to all inliers using e.g. least squares.
   – When we have the best set of points, refine the model using all inliers.

Different models and s

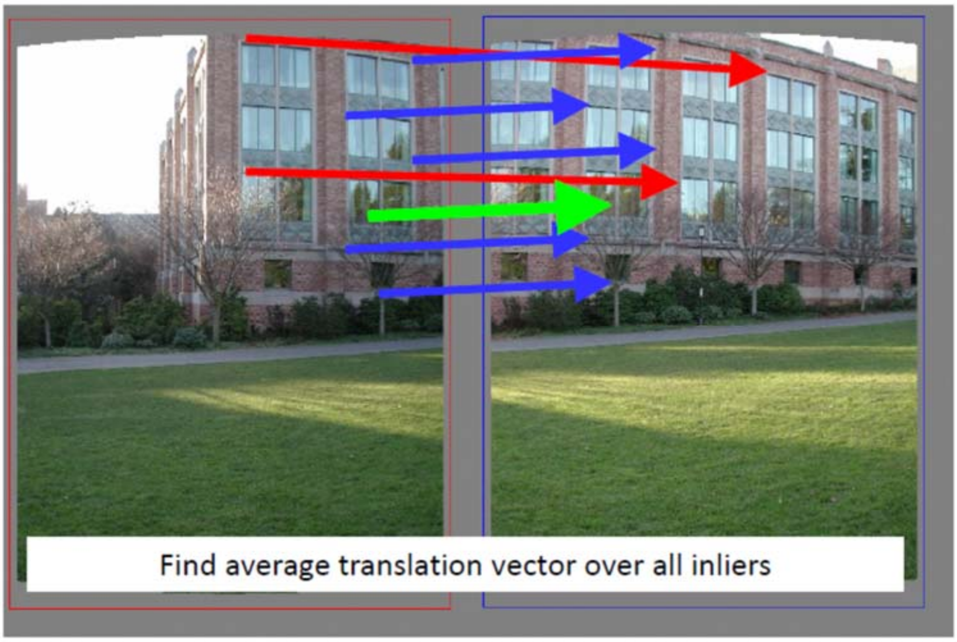
- For alignment, s, the number of points needed, depends on the motion model. Each corresponding point in the image pair is one sample.



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{2\times3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{2\times3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} A \end{bmatrix}_{2\times3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times3}$	8	straight lines	

Final step: refine the best model

- When the model with the highest number of inliers is found, this model is refitted to the set of all samples that are inliers.



Termination of the algorithm

- The criterion for terminating the algorithm is that the probability of finding a better consensus set is lower than a certain threshold.
- Let *q* be the probability for picking a set that does not contain any outliers.
- This depends on the number of points picked as $q = p_i^s$
- The probability of picking as least one outlier will then be *1-q*.
- If this is repeated *h* times, the probability to pick outliers in <u>every random pick</u> is $(1-q)^h$.
- Since we are selecting a small number s out of all corresponding points we will sooner or later make a good pick and this quantity goes to zero as *h* goes to infinity.

Termination of the algorithm

- Goal: pick h large enough so that *$(1-q)^h$* is smaller than a probability threshold ε.

$$(1-q)^h \leq \varepsilon$$

$$h \log(1-q) \leq \log \varepsilon$$

$$h \geq \left\lceil \frac{\log \varepsilon}{\log(1-q)} \right\rceil$$

- The threshold for the iterations will be to stop at iteration

$$\hat{T}_{iter} = \left\lceil \frac{\log \varepsilon}{\log(1-q)} \right\rceil \leftarrow$$

Notation means smallest integer larger than

The number of iterations

- e, the outlier ratio, is unknown. We often pick worst case, e.g. 50% first, then adapt as we find more inliners.
- $N=\log(1-\varepsilon)/\log(1-(1-e)^s))$
- While N>sample_count repeat
  - Choose a sample and count the number of inliers
  - Set e=(1-(number of inliners))/(total number of points) Recompute N from e
  - Increment sample_count

A table for the number of iterations

	proportion of outliers e						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

RANSAC parameters

- Model
  - Choose the simplest model that describes the type of motion involved
  - Possible simple motion models (for equations see RANSAC4Dummies section 4.2)
    - Linear
    - Plane
    - Rotation, scaling and translation
    - Homographic(linear transform to relate two views from the same camera, used for panography)
- Distance threshold t:
  - Choose t such that the probability for inlier is p (e.g. 0.95).
  - Assume zero mean Gaussian noise with std. dev. σ: $t^2=3.84\ \sigma^2$
- Number of iterations: Choose according to the table

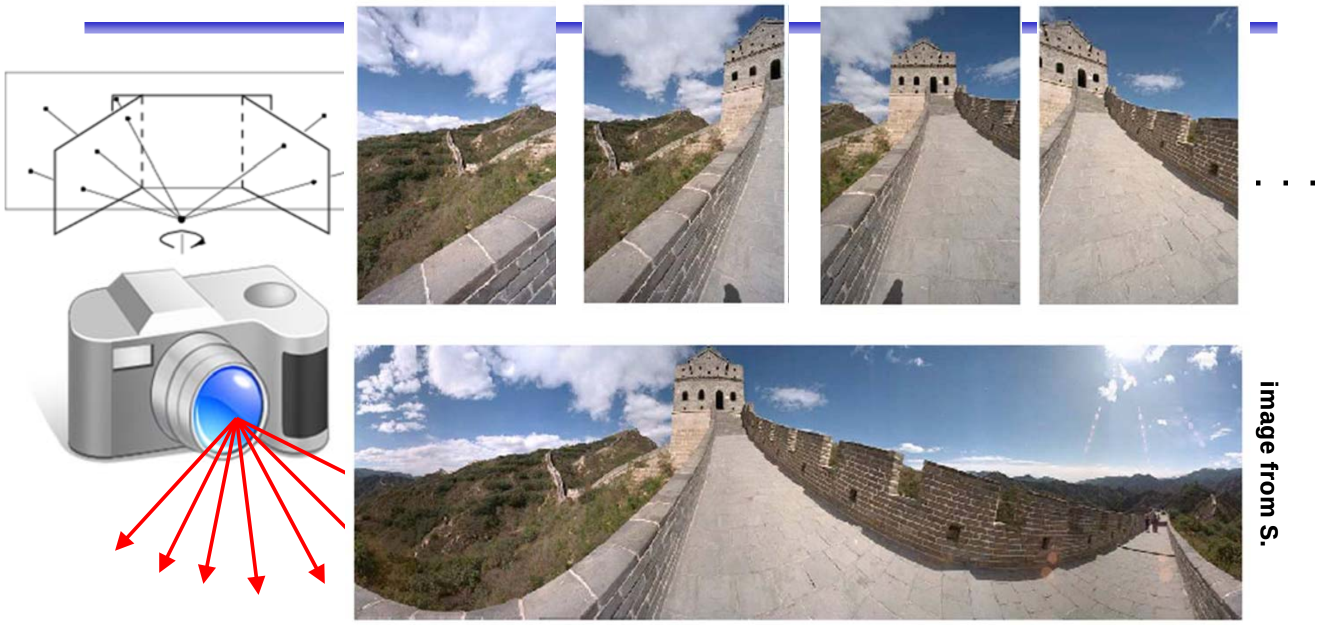
Algorithm 15.4: RANSAC: fitting lines using random sample consensus

```
Determine:
    n — the smallest number of points required
    k — the number of iterations required
    t — the threshold used to identify a point that fits well
    d — the number of nearby points required
      to assert a model fits well
Until k iterations have occurred
    Draw a sample of n points from the data
      uniformly and at random
    Fit to that set of n points
    For each data point outside the sample
      Test the distance from the point to the line
        against t; if the distance from the point to the line
        is less than t, the point is close
    end
    If there are d or more points close to the line
      then there is a good fit. Refit the line using all
      these points.
end
Use the best fit from this collection, using the
  fitting error as a criterion
```


RANSAC conclusions

- Good:
  - Robust to outliers (can handle up to 50% outliers)
  - Applicapable to a larger number of parameters than Hough transform/parameters are easier to choose.
- Bad:
  - Computational time grows quickly with fraction of outliers and number of parameters.
  - Not good for getting multiple fits.
- Common applications:
  - Robust linear regression (and similar)
  - Computing the transform behind image stitching (called homography)
  - Image registration/Estimating the fundamental matrix relating two views.

Panoramas



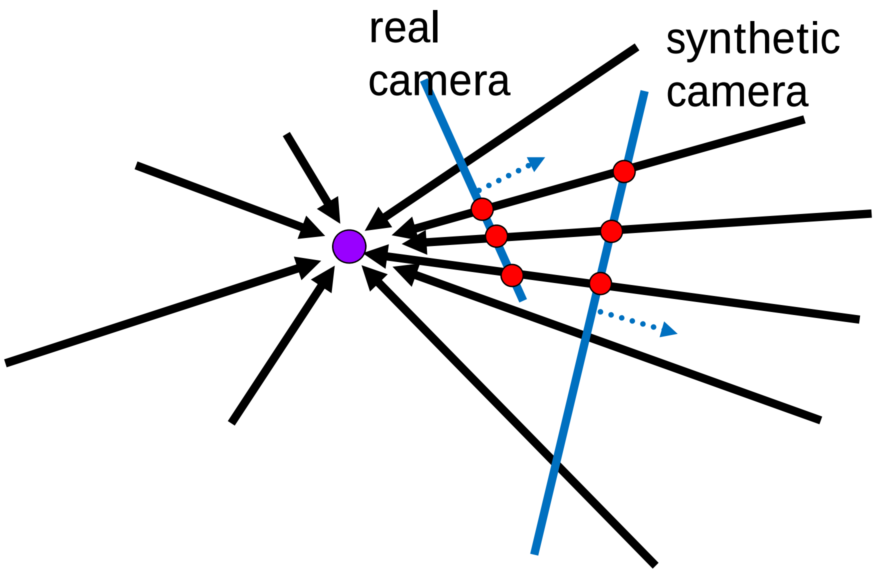
Obtain a wider angle view by combining multiple images.

Grauman

How to stitch together a panorama?

- Basic Procedure
  - Take a sequence of images from the same position
    - Rotate the camera about its optical center
  - Compute transformation between second image and first
  - Transform the second image to overlap with the first
  - Blend the two together to create a mosaic
  - (If there are more images, repeat)

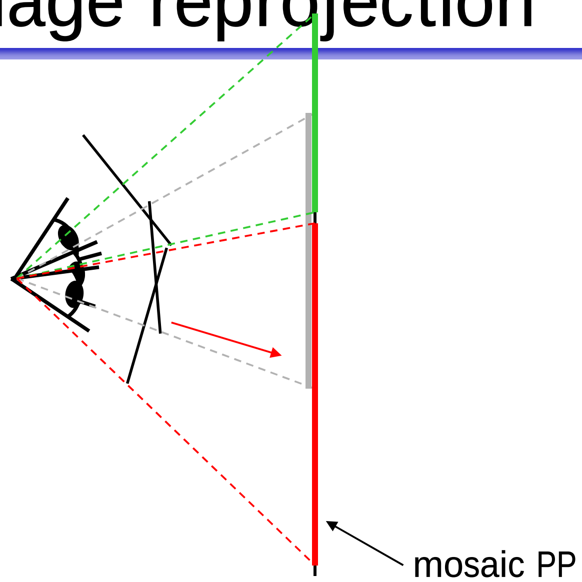
Panoramas: generating synthetic views



Can generate any synthetic camera view
as long as it has **the same center of projection**!

Source: Alyosha Efros

Image reprojection



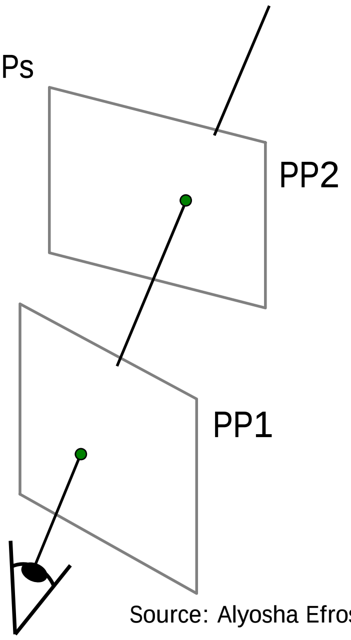
- The mosaic has a natural interpretation in 3D
  - The images are reprojected onto a common plane
  - The mosaic is formed on this plane
  - Mosaic is a *synthetic wide-angle camera*

Source: Steve Seitz

Homography

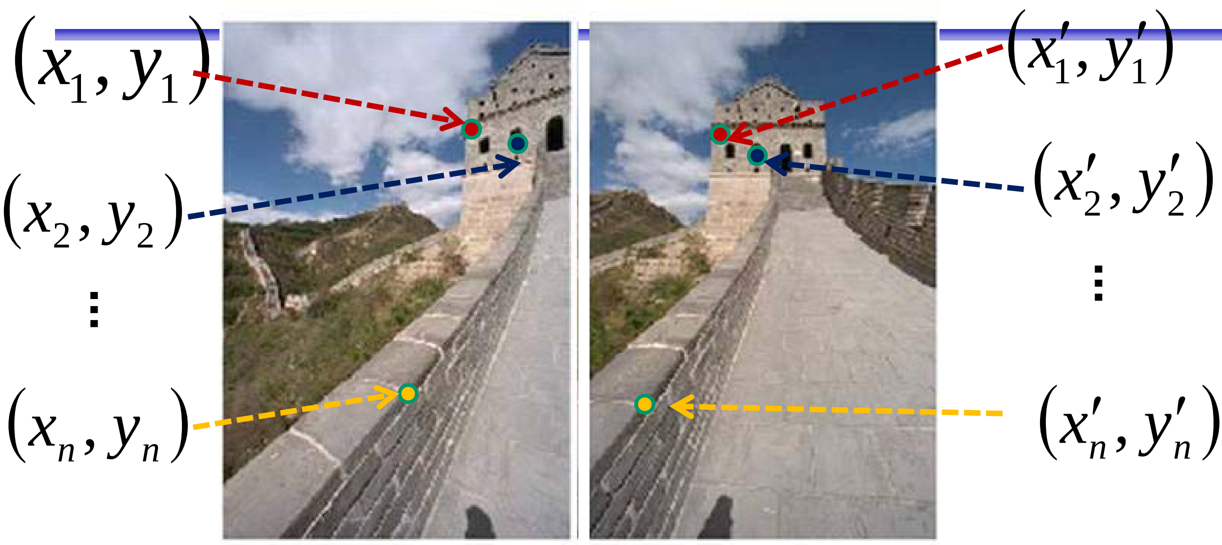
- How to relate two images from the same camera center?
  - how to map a pixel from PP1 to PP2?
- Think of it as a 2D **image warp** from one image to another.
- A projective transform is a mapping between any two PPs with the same center of projection
  - rectangle should map to arbitrary quadrilateral
  - parallel lines aren't
  - but must preserve straight lines
- called **Homography**

$$\underbrace{\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}}_{\mathbf{p'}} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\mathbf{p}}$$



Source: Alyosha Efros

Homography



To **compute** the homography given pairs of corresponding points in the images, we need to set up an equation where the parameters of **H** are the unknowns...

Grauman

Solving for homographies

p' = Hp $$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Can set scale factor i=1. So, there are 8 unknowns.
- Set up a system of linear equations:
  - **Ah = b**
- where vector of unknowns h = $[a,b,c,d,e,f,g,h]^T$
- Need at least 8 eqs, but the more the better…
- Solve for h. If overconstrained, solve using least-squares:

$$\min \|Ah - b\|^2$$

Grauman

Summary: How to stitch together a panorama?

- Basic Procedure
  - Take a sequence of images from the same position
    - Rotate the camera about its optical center
  - Compute transformation between second image and first
  - Transform the second image to overlap with the first
  - Blend the two together to create a mosaic
  - (If there are more images, repeat)

Source: Steve Seitz