Homography

CS 4243 Computer Vision & Pattern Recognition

Angela Yao

Recap & Outline

Last Week

- Definition and characteristics of (ideal) image descriptors
- MOPS, GIST, SIFT
- SIFT as a keypoint detector and descriptor
- Feature matching and evaluation

Today's Lecture

- Overview of 2D transformations
- Definition of image homographies
- Direct linear transform (DLT) as a solution for solving homographies
- Random Sample Consensus (RANSAC) for robust solutions

Before Panorama Apps on Smartphones…

San Francisco, US

35mm film panorama from year 2000

https://www.flickr.com/photos/parkstreetparrot/27691676929/

How do we stitch images from different viewpoints?

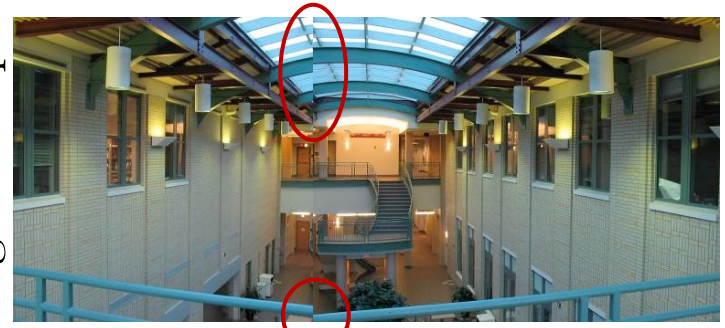
Shift the images on top of each other?

Translation-only stitching is not enough to mosaic these images.

We need to use image homographies to project images onto the same plane.

left on top

right on top



Slide credit: K. Kitani

How can we combine two images?

1. Match parts which are the same on both images
2. Align images based on the matches

rectified

a. Compute homography between two images
b. Project images onto a common plane

Why do we get these black areas?

Adapted from K. Grauman

Image Projections

(a crash course in) Homographies, Warping, DLT & RANSAC

What types of image transformations can we do?

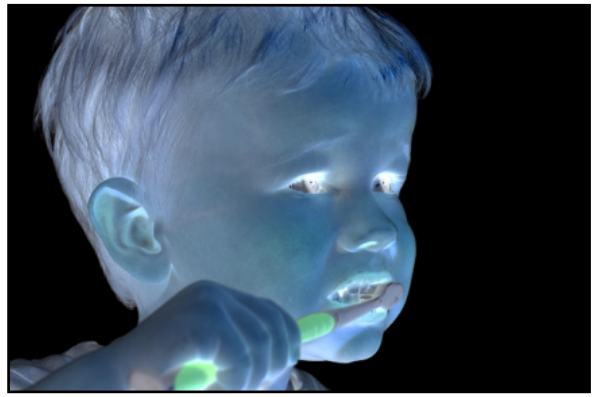
F

Filtering

$$G(\boldsymbol{x}) = h\{F(\boldsymbol{x})\}$$

Changes range of image function

G



Changes pixel *values*

F

Warping

$$G(\boldsymbol{x}) = F(h\{\boldsymbol{x}\})$$

Changes domain of image function

G

Changes pixel *locations*

Slide credit: K. Kitani

2D Transformations

- object recognition
- 3D reconstruction
- augmented reality
- image stitching

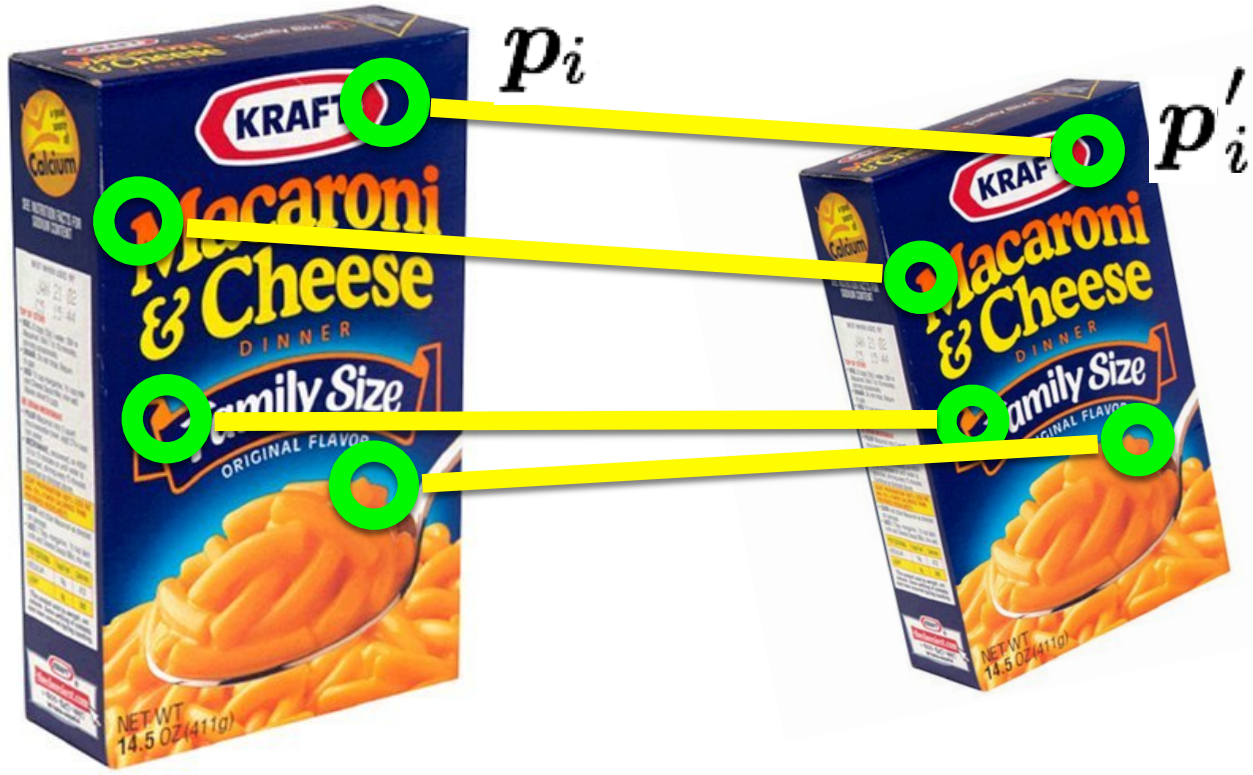
Given a set of matched key points:

$$\{\boldsymbol{p}_i, \boldsymbol{p}'_i\}$$

$i = \{1 \ldots n\}$ for N matched keypoint pairs

point in one image

point in the other image



Can we compute the transformation:

$$\boldsymbol{p}' = \boldsymbol{f}(\boldsymbol{p}; H)$$

transformation function

parameters

What kind of transformation functions $\boldsymbol{f}$ are there?

What are good estimates of parameters $\boldsymbol{H}$?

Slide credit: K. Kitani

Classification of 2D transformations

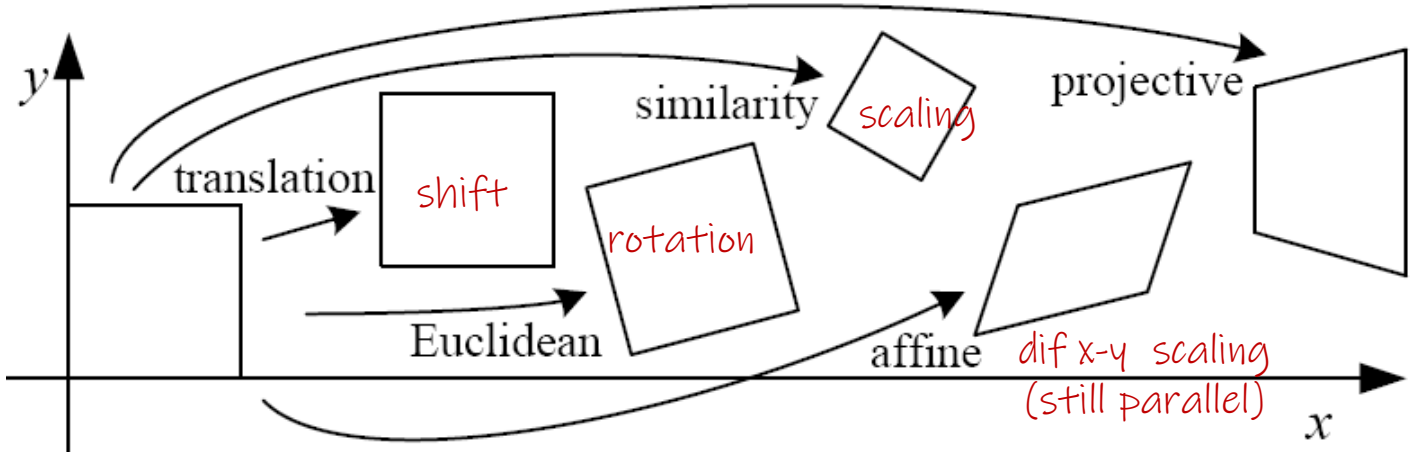
translation

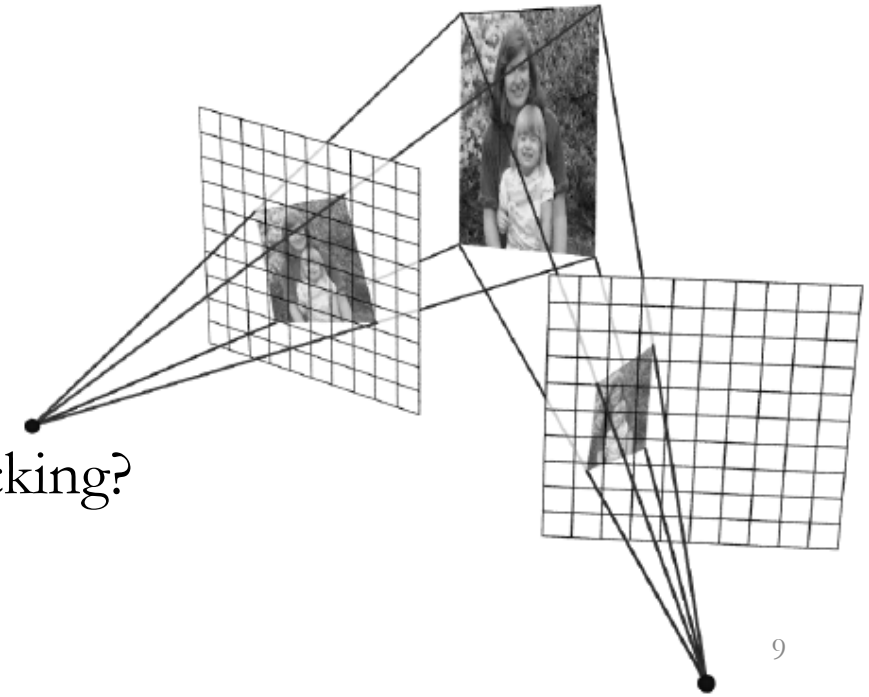
rotation

scaling

affine

projective





Which kind transformation is needed for image mosaicking?
A projective transformation (a.k.a. a homography).

Applying a homography

heterogeneous coordinates

homogeneous coordinates

$$p = \begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Add a 1 here

1. Convert to homogeneous coordinates:

   Represent 2D point with a 3D vector, so that we can conveniently represent translations with matrix multiplication.

2. Multiply by the homography matrix:

What is the size of the homography matrix?

[3x1] = [3x3][3x1]

$$P' = H \cdot P$$

3. Convert back to heterogeneous coordinates:

$$P' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \Rightarrow p' = \begin{bmatrix} x'/w' \\ y'/w' \end{bmatrix}$$

How to solve for H? One way is via the Direct Linear Transform (DLT).

Slide credit: K. Kitani

Determining the homography matrix w/ DLT

Given a set of matched key points $\{p_i, p_i'\}$ find the best estimate of H s.t. $P' = H \cdot P$

Write out linear equation for each correspondence:

$$P' = H \cdot P \qquad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Expand matrix multiplication:

$$\begin{aligned} x' &= \alpha(h_1 x + h_2 y + h_3) \\ y' &= \alpha(h_4 x + h_5 y + h_6) \\ 1 &= \alpha(h_7 x + h_8 y + h_9) \end{aligned}$$

Divide out unknown scale factor:

$$\begin{aligned} x'(h_7 x + h_8 y + h_9) &= (h_1 x + h_2 y + h_3) \\ y'(h_7 x + h_8 y + h_9) &= (h_4 x + h_5 y + h_6) \end{aligned}$$

Slide credit: K. Kitani

Determining the homography matrix w/ DLT

Re-arrange terms:

$$h_7 x x' + h_8 y x' + h_9 x' - h_1 x - h_2 y - h_3 = 0$$
$$h_7 x y' + h_8 y y' + h_9 y' - h_4 x - h_5 y - h_6 = 0$$

Re-write in matrix form:

$$\mathbf{A}_i \boldsymbol{h} = \mathbf{0}$$

$$\mathbf{A}_i = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\boldsymbol{h} = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{bmatrix}^\top$$

Slide credit: K. Kitani

Determining the homography matrix w/ DLT

Stack together constraints from multiple point correspondences:

$$\mathbf{A}\boldsymbol{h} = \mathbf{0}$$

Homogeneous linear least squares problem.
We want to avoid the trivial solution h = 0.

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

Use singular value decomposition (SVD) on A. Set ***h*** equal to the (right) singular vector corresponding to smallest singular value. (detailed proof)

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$$

Singular values found on diagonal elements of Σ.

Columns of V form (right) singular vectors.

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Slide credit: K. Kitani

Solving for H using DLT

Given $\{p_i, p_i'\}$ solve for H such that $P_i' = H \cdot P_i \quad \forall i$

1. For each correspondence, create 2x9 matrix $\mathbf{A}_i$
2. Concatenate into single 2n x 9 matrix $\mathbf{A}$
3. Compute SVD of $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$
4. Store singular vector of smallest singular value $\boldsymbol{h} = \boldsymbol{v}_{\hat{i}}$
5. Reshape to get $\mathbf{H}$

Slide credit: K. Kitani

Drawbacks of DLT

- DLT (direct linear transform) as the name suggests, is a form of **linear** least squares estimation.
  - Applicable only when actual transformation between two images is linear. We assume a projective model but the underlying source images may not actually be related linearly...
- Sensitive to scaling in pixel space
  - add a normalization step
- DLT does not deal well with outliers, i.e. bad matches
  - make more robust with RANSAC

When are Homographies Applicable?

The scene is planar;

… or approximately planar i.e. scene is very far away or has small (relative) depth variation.

The scene is captured under camera rotation only (no translation or pose change)

*Algorithm box taken from different source, {x_i, x_i'} correspond to our {p_i, p_i'}

Normalized DLT

Objective

Given $n \geq 4$ 2D to 2D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}_i'\}$*, determine the 2D homography matrix H such that $\mathbf{x}_i' = \mathrm{H}\mathbf{x}_i$.

Algorithm

(i) **Normalization of x:** Compute a similarity transformation T, consisting of a translation and scaling, that takes points $\mathbf{x}_i$ to a new set of points $\tilde{\mathbf{x}}_i$ such that the centroid of the points $\tilde{\mathbf{x}}_i$ is the coordinate origin $(0,0)^\mathsf{T}$, and their average distance from the origin is $\sqrt{2}$.

(ii) **Normalization of x′:** Compute a similar transformation T′ for the points in the second image, transforming points $\mathbf{x}_i'$ to $\tilde{\mathbf{x}}_i'$.

(iii) **DLT:** Apply standard DLT to the correspondences $\tilde{\mathbf{x}}_i \leftrightarrow \tilde{\mathbf{x}}_i'$ to obtain a homography $\tilde{\mathrm{H}}$.

(iv) **Denormalization:** Set $\mathrm{H} = \mathrm{T}'^{-1}\tilde{\mathrm{H}}\mathrm{T}$.

Algorithm 4.2 from Hartley & Zisserman

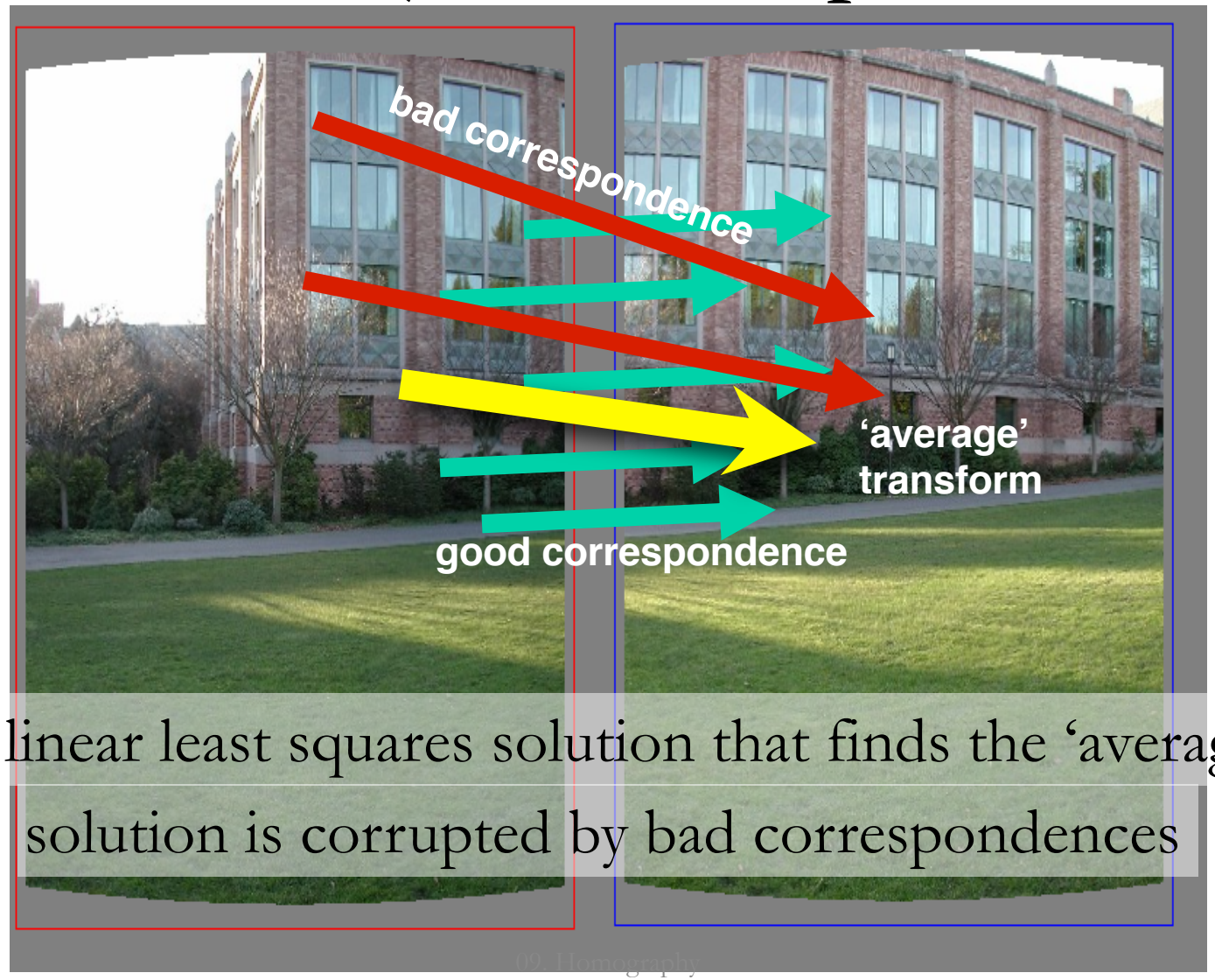
A nice post detailing how to solve for the normalization matrix T.

Multiple View Geometry in Computer Vision. Richard Hartley and Andrew Zisserman, Cambridge University Press, 2004.

RANSAC

Robust method for estimating homographies

Effect of outliers (bad correspondences)



The DLT is a linear least squares solution that finds the 'average' transform

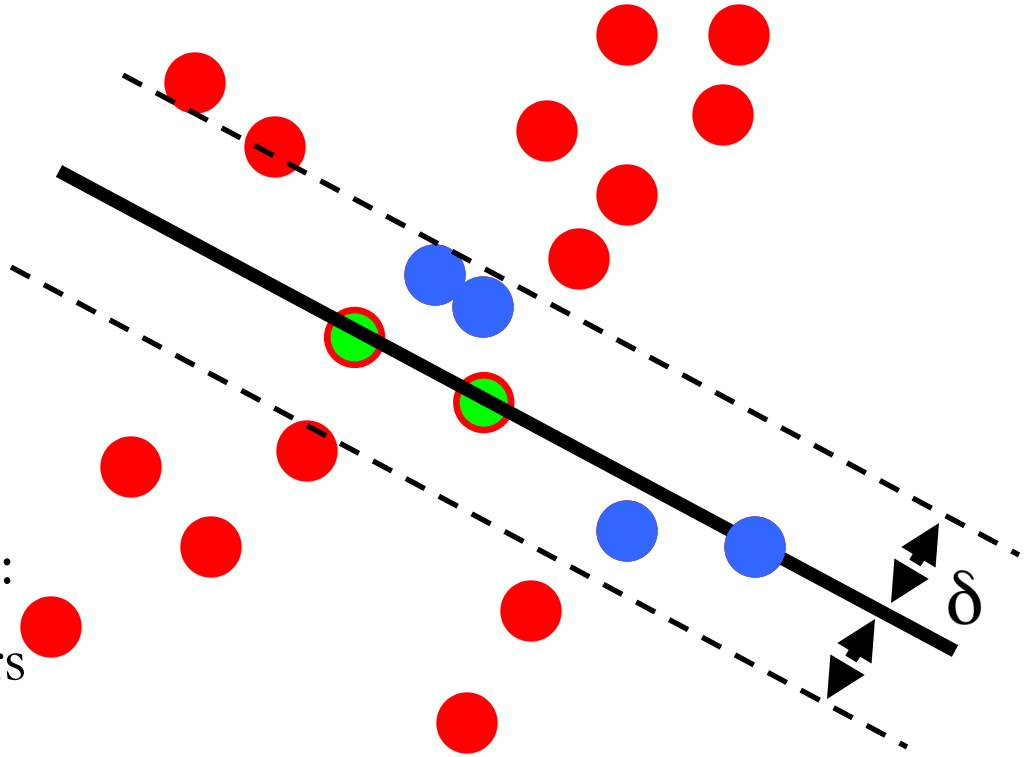
solution is corrupted by bad correspondences

Slide credit: K. Kitani?

RANSAC

Random sample consensus (**RANSAC**) estimates model parameters from a set of noisy observations as a 2-stage process:

- Classify data as inlier vs. outliers
- Fits model to only inliers



Fitting lines (with outliers)

$N_I = 6$

Algorithm:

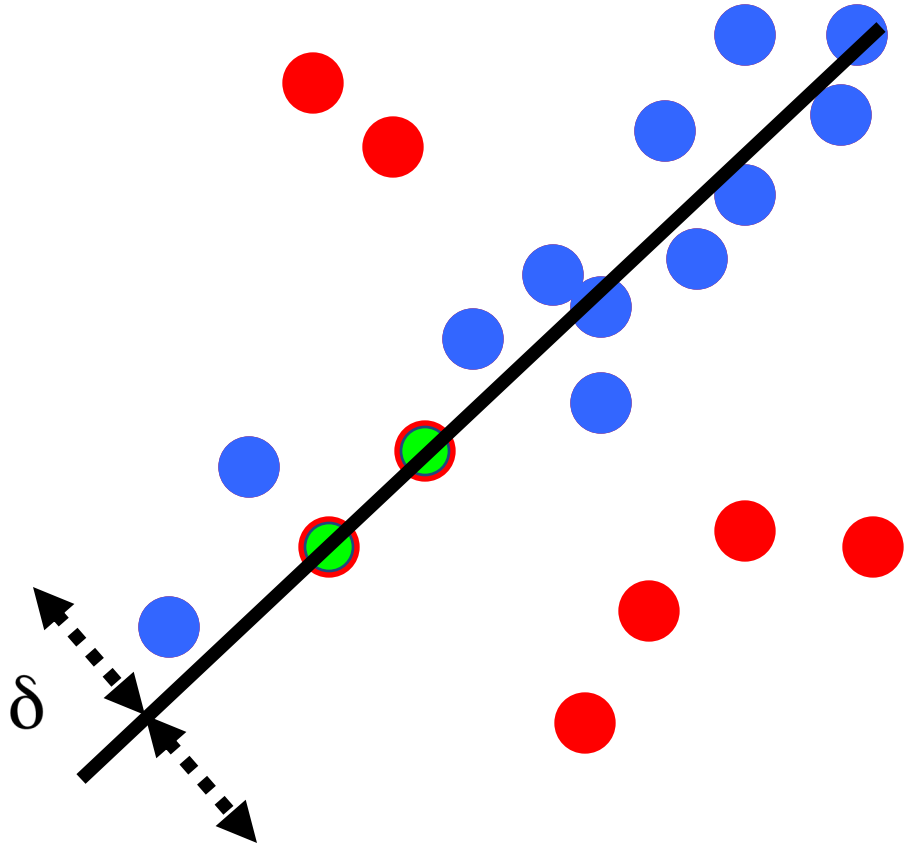
Loop

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

Slide credit: K. Kitani?

RANSAC

Fitting lines
(with outliers)



$N_I = 14$

Algorithm:

Loop

1. Sample (randomly) the number of points required to fit the model
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 the most inliers.

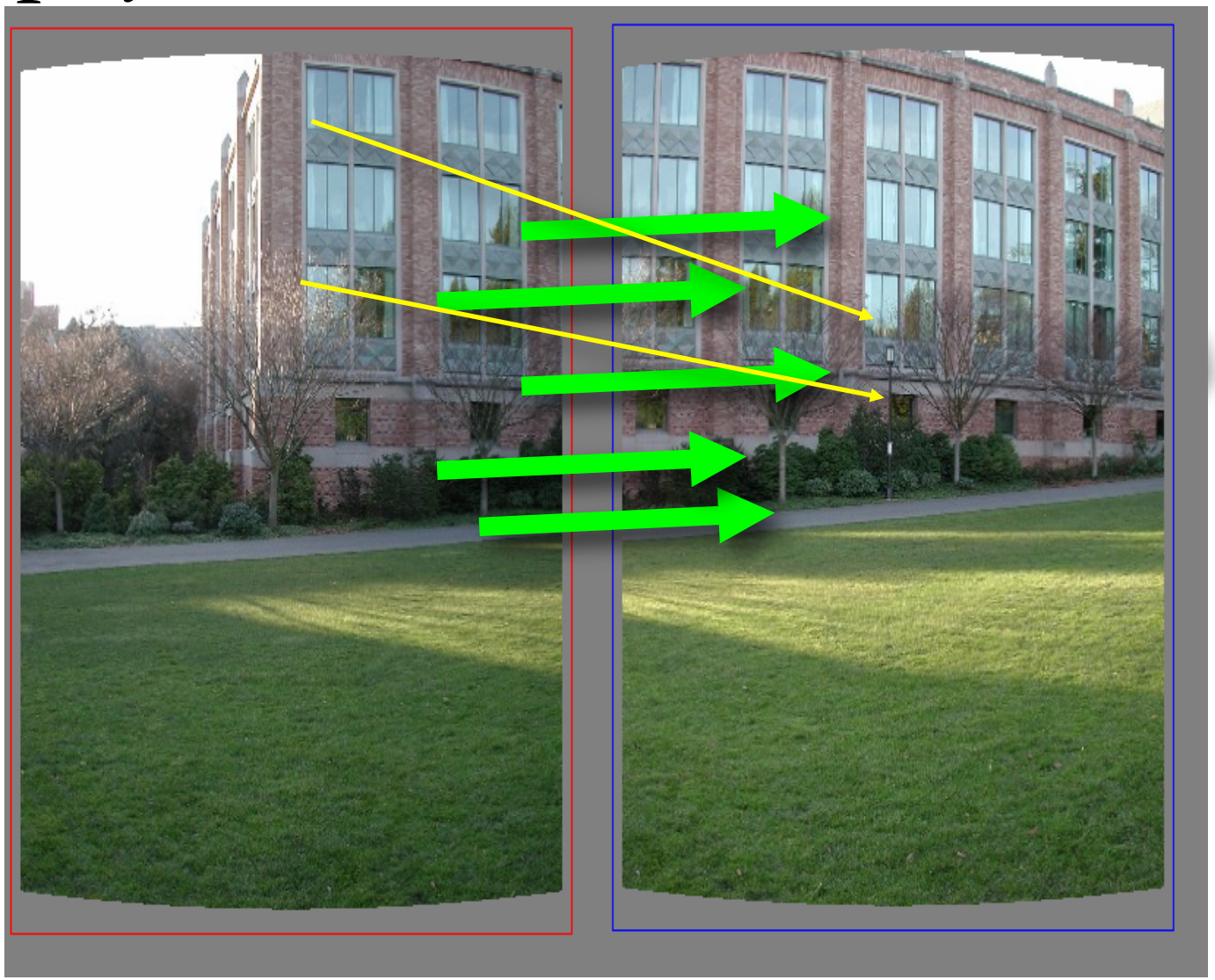
Slide credit: K. Kitani?

Estimating homography with RANSAC

RANSAC loop

1. Randomly get (min 4) correspondences.
2. Compute H using DLT
3. Count inliers
4. Keep H if largest number of inliers

For best H with most inliers, recompute using all inliers.



Slide credit: K. Kitani

RANSAC Parameters

- Two parameters to choose:
  - δ – distance threshold to consider as an inlier
  - N – number of iterations to run RANSAC loop
- Choose δ so that (noisy) inlier points will still lie within the threshold
  - Usually, this is done empirically via trial and error
- An upper bound on N, however, can be solved if we have a rough idea of the proportion of outliers vs. inliers

How Many Iterations N?

- e: probability that a point is an outlier
- s: number of points in we draw per iteration for fitting parameters
- N: is the number of iterations in loop
- p: probability that we want to ensure that at least one set of points which we sampled does not contain any outliers
- How to solve for N?
  This depends on p we want, e.g. 0.99

Probability of having at least one outlier in each of the N iterations

$$(1-(1-e)^s)^N = \underbrace{1-p}$$

probability that at least one point sampled was an outlier

probability of choosing an inlier when sampling one point

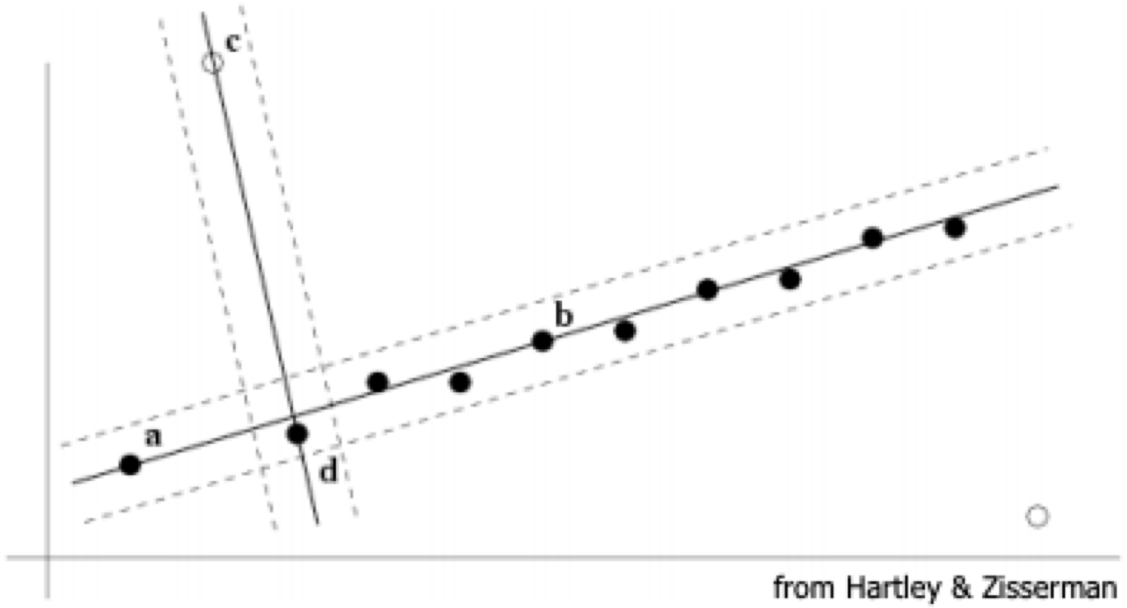
probability of choosing s inliers

1 - probability of at least one iteration having no outliers

$$N = \frac{\log(1-p)}{\log\left(1-(1-e)^s\right)}$$

Example: N for line-fitting

- 12 data samples, 2 of which are outliers (e = 1/6 = 0.167)
- We wish to fit a straight line, i.e. minimum sample size s=2
- For a p of 0.99 (0.99 probability that we sample 2 points which are not outliers), how many times do we need to sample?



from Hartley & Zisserman

$$N = \frac{\log(1 - 0.99)}{\log(1 - (1 - 0.167)^2)} \approx 4$$

Do we really need N samples?

For homography, outliers are wrong keypoint matches

	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

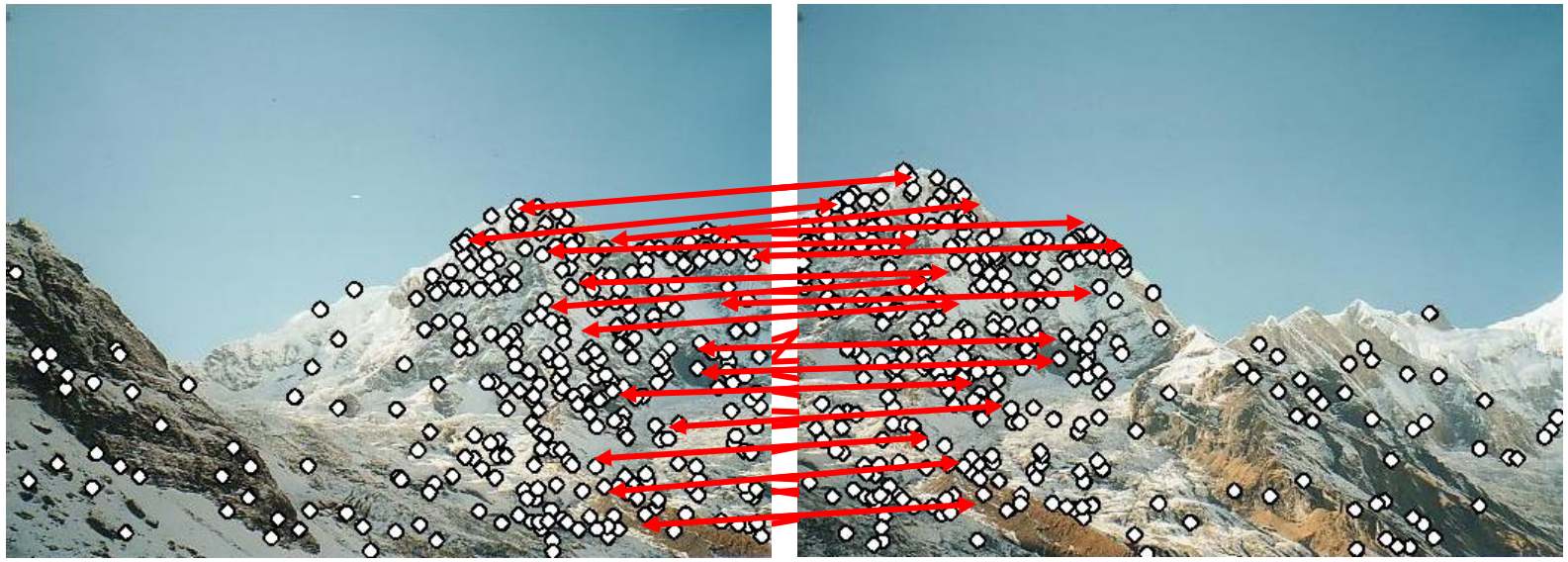
for solving homography

Chart of N for minimum number of points for model (s) and proportion of outliers (e). Note the rapid increase when outlier proportion increases!

Early stopping criteria: when inlier ratio for a given model reaches the expected proportion of inliers.

Both stopping conditions requires us to have an idea of the proportion of outliers (e). This may not always be feasibly.

Robust feature-based alignment



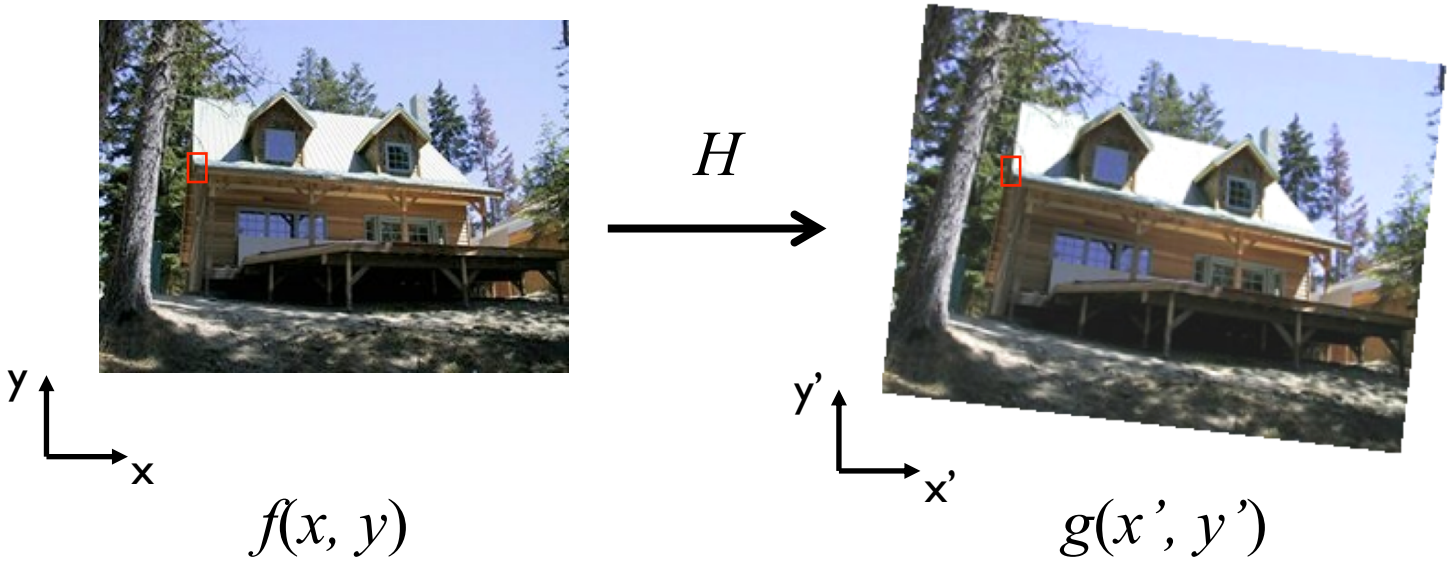
- Extract features, compute matches
- RANSAC Loop:
  - Hypothesize transformation H based on a small group of putative matches
  - Verify transformation by searching for other matches consistent with H

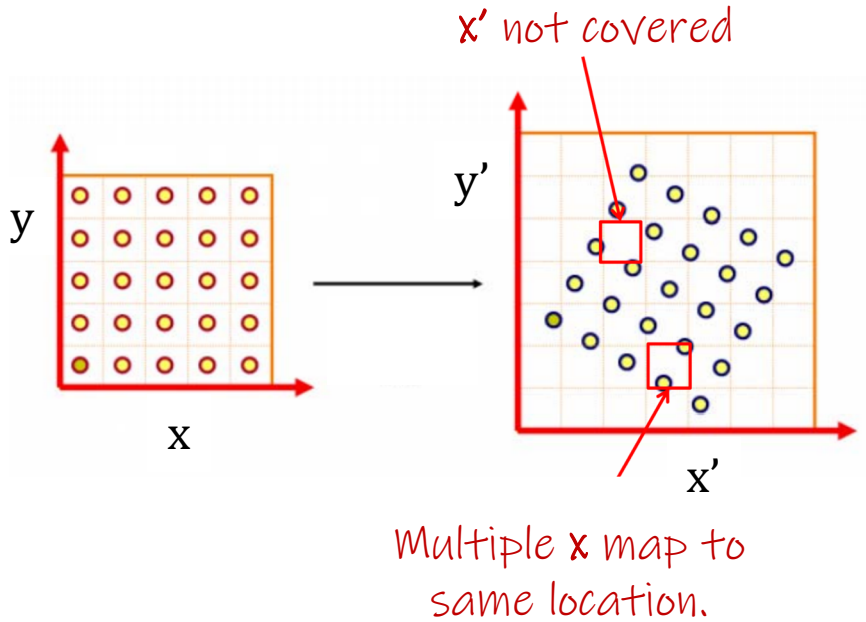
Adapted from K. Grauman, L. Lazebnik

Image Warping

Stitching together the images

Image Warping:





Based on homography H, we can warp the image into a new plane by sending intensities *f(**x**)* in to corresponding location ***x'*** in the new plane.

Discretization onto image grid causes problems:

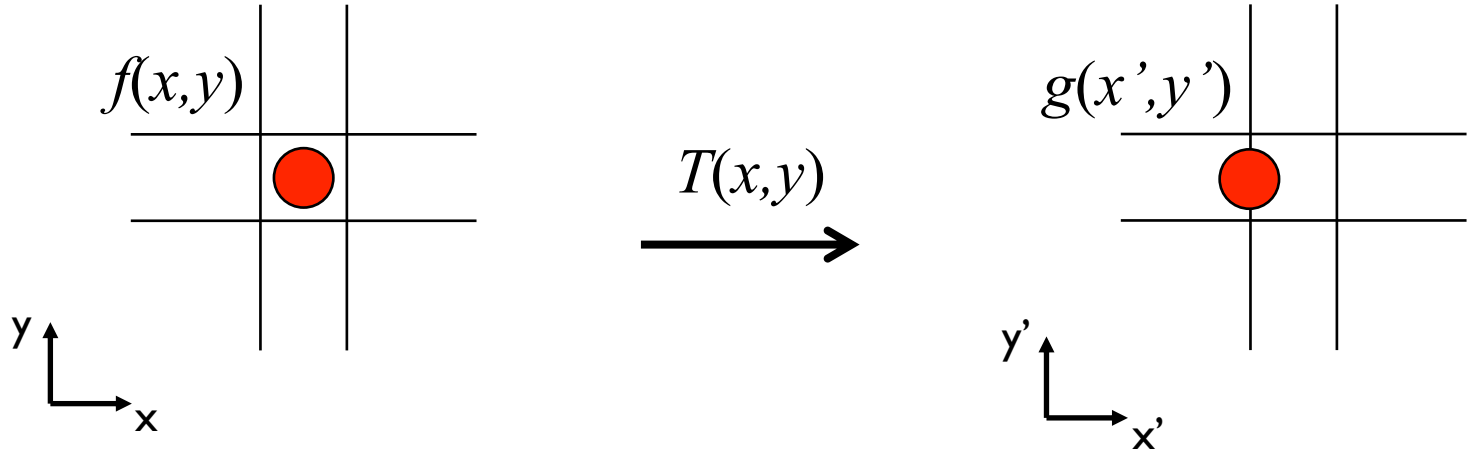
- certain destination pixels are not covered.
- many pixels might map to the same destination.

Slide credit: K. Kitani

Warping

Exactly details of warping and also the physical blending or stitching of the two images are beyond the scope of this lecture. You can call built-in API functions for this purpose.

A source pixels may end up between multiple target pixel locations



- distribute intensity among neighboring pixels (x',y') ("splatting")
- target pixels (x',y') receiving intensity from more than one source (x,y) gets an average

Slide credit: K. Kitani

Summary

- Project two images onto a common plane via a homography
- Homography is solved from matching locations in two images and using the direct linear transform (DLT)
- DLT gives a linear least squares solution, i.e. an "average" solution
  - prone to corruption by outliers, make robust via RANSAC
- Random Sample Consensus (RANSAC)
  - iteratively samples a subset of points to solve for hypothesis model parameters
  - counting inliers based on hypothesized model
  - Integrate RANSAC directly into feature matching and solving homography

Optional Reading on Why Normalization is Necessary for DLT

version of the algorithm, incorporating data normalization, should be used in preference to the basic DLT of algorithm 4.1(p91). Note that normalization is also called *pre-conditioning* in the numerical literature.

The DLT method of algorithm 4.1 uses the SVD of $\mathtt{A} = \mathtt{UDV}^\mathsf{T}$ to obtain a solution to the overdetermined set of equations $\mathbf{Ah} = \mathbf{0}$. These equations do not have an exact solution (since the $2n \times 9$ matrix $\mathtt{A}$ will not have rank 8 for noisy data), but the vector $\mathbf{h}$, given by the last column of $\mathtt{V}$, provides a solution which minimizes $\|\mathbf{Ah}\|$ (subject to $\|\mathbf{h}\| = 1$). This is equivalent to finding the rank 8 matrix $\hat{\mathtt{A}}$ which is closest to $\mathtt{A}$ in Frobenius norm and obtaining $\mathbf{h}$ as the exact solution of $\hat{\mathbf{A}}\mathbf{h} = \mathbf{0}$. The matrix $\hat{\mathtt{A}}$ is given by $\hat{\mathtt{A}} = \mathtt{U}\hat{\mathtt{D}}\mathtt{V}^\mathsf{T}$ where $\hat{\mathtt{D}}$ is $\mathtt{D}$ with the smallest singular value set to zero. The matrix $\hat{\mathtt{A}}$ has rank 8 and minimizes the difference to $\mathtt{A}$ in Frobenius norm because

$$\|\mathtt{A} - \hat{\mathtt{A}}\|_\mathtt{F} = \|\mathtt{UDV}^\mathsf{T} - \mathtt{U}\hat{\mathtt{D}}\mathtt{V}^\mathsf{T}\|_\mathtt{F} = \|\mathtt{D} - \hat{\mathtt{D}}\|_\mathtt{F}.$$

where $\|.\|_\mathtt{F}$ is the Frobenius norm, i.e. the square root of the sum of squares of all entries.

Without normalization typical image points $\mathbf{x}_i, \mathbf{x}_i'$ are of the order $(x, y, w)^\mathsf{T} = (100, 100, 1)^\mathsf{T}$, i.e., x, y are much larger than w. In $\mathtt{A}$ the entries xx', xy', yx', yy' will be of order 10^4, entries xw', yw' etc. of order 10^2, and entries ww' will be unity. Replacing $\mathtt{A}$ by $\hat{\mathtt{A}}$ means that some entries are increased and others decreased such that the square sum of differences of these changes is minimal (and the resulting matrix has rank 8). However, and this is the key point, increasing the term ww' by 100 means a huge change in the image points, whereas increasing the term xx' by 100 means only a slight change. This is the reason why all entries in $\mathtt{A}$ must have similar magnitude and why normalization is essential.

Multiple View Geometry in Computer Vision. Richard Hartley and Andrew Zisserman, Cambridge University Press, 2004.

Optional Reading on Why Normalization is Necessary for DLT

The effect of normalization is related to the condition number of the set of DLT equations, or more precisely the ratio d_1/d_{n-1} of the first to the second-last singular value of the equation matrix A. This point is investigated in more detail in [Hartley-97c]. For the present it is sufficient to say that for exact data and infinite precision arithmetic the results will be independent of the normalizing transformation. However, in the presence of noise the solution will diverge from the correct result. The effect of a large condition number is to amplify this divergence. This is true even for infinite-precision arithmetic – this is not a round-off error effect.

The effect that this data normalization has on the results of the DLT algorithm is shown graphically in figure 4.4. The conclusion to be drawn here is that data normalization gives dramatically better results. The examples shown in the figure are chosen to make the effect easily visible. However, a marked advantage remains even in cases of computation from larger numbers of point correspondences, with points more widely distributed. To emphasize this point we remark:

- *Data normalization is an* essential *step in the DLT algorithm. It must* not *be considered optional.*

Data normalization becomes even more important for less well conditioned problems, such as the DLT computation of the fundamental matrix or the trifocal tensor, which will be considered in later chapters.

Multiple View Geometry in Computer Vision. Richard Hartley and Andrew Zisserman, Cambridge University Press, 2004.