Image Alignment and Stitching

COS 429: Computer Vision

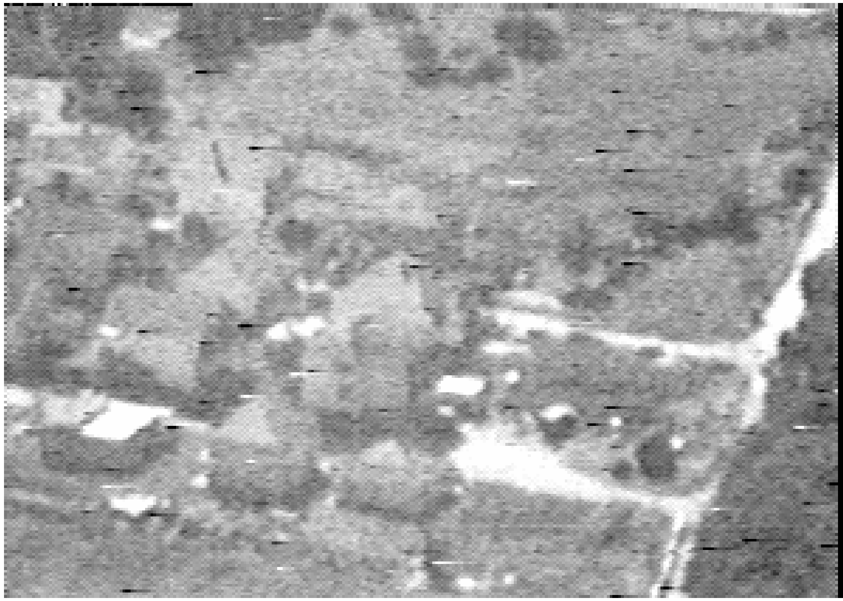
PRINCETON UNIVERSITY

Figure credits: R. Szeliski, S. Lazebnik, H. Sawhney

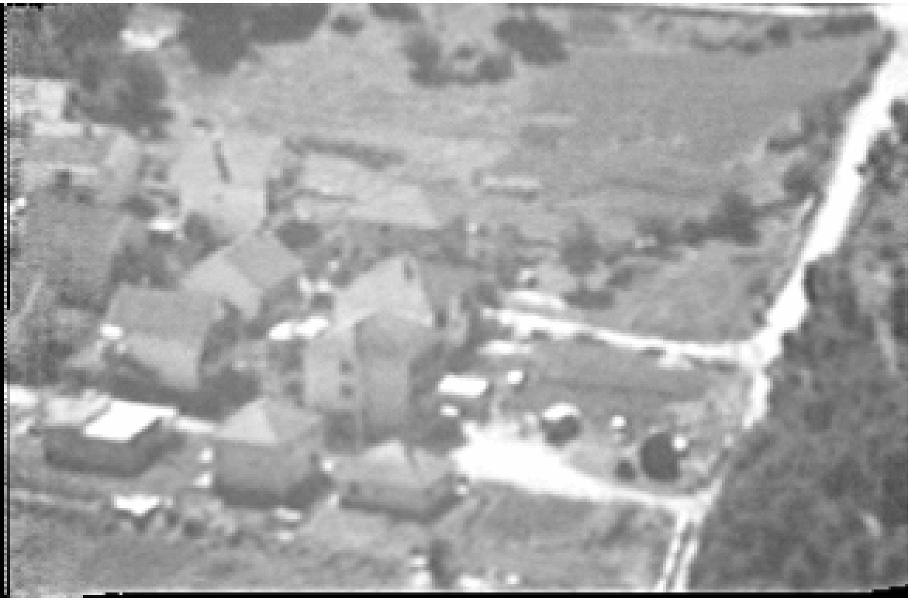
Image Alignment Applications

- Local alignment:
  - Tracking
  - Stereo
- Global alignment:
  - Camera jitter elimination
  - Image enhancement
  - Panoramic mosaicing

Image Enhancement



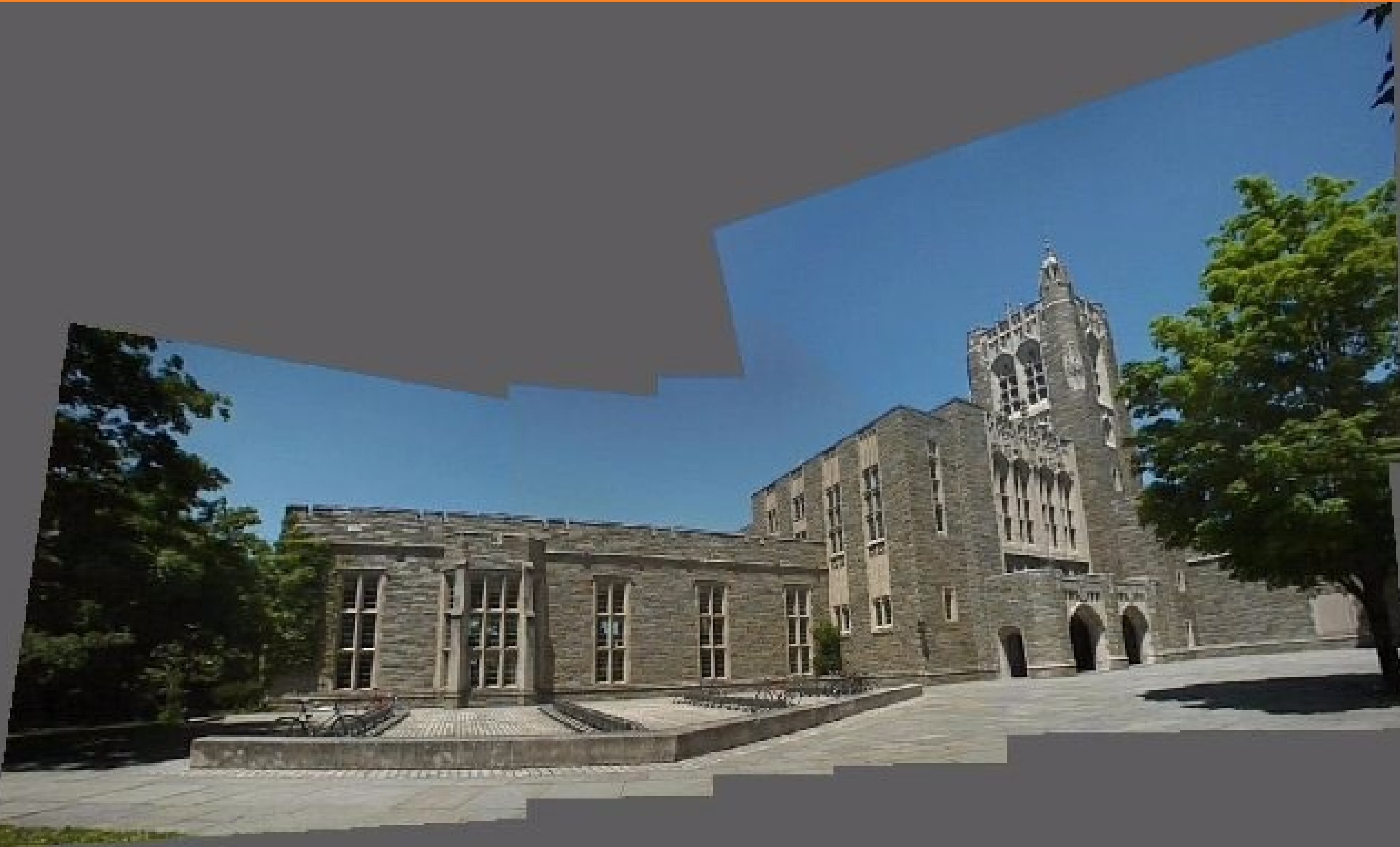
Original



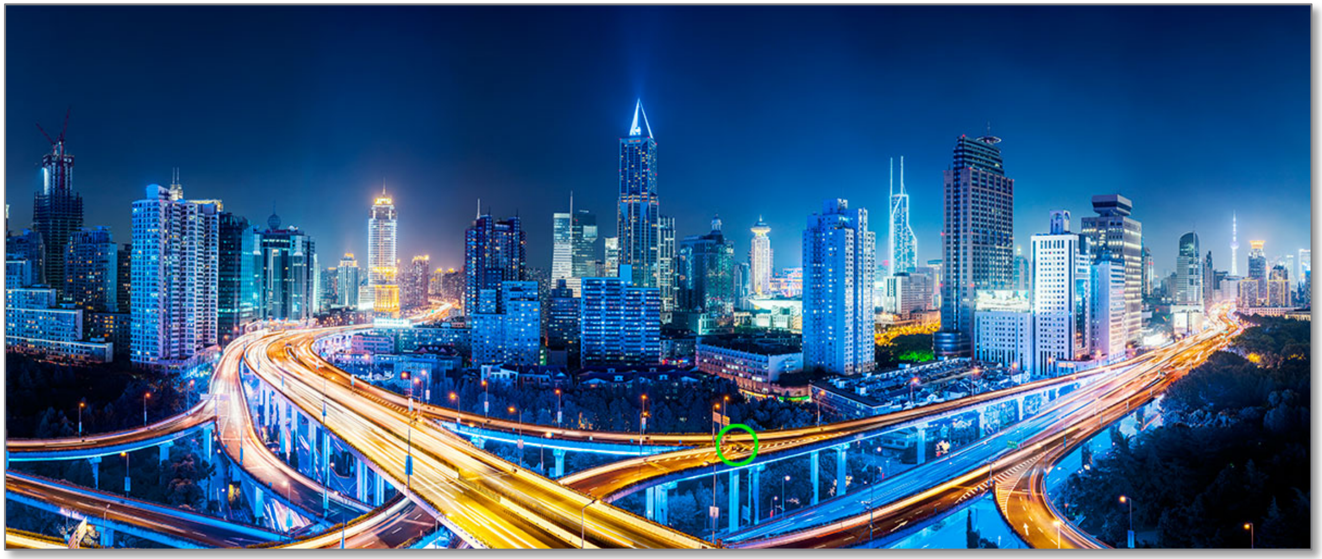
Denoised

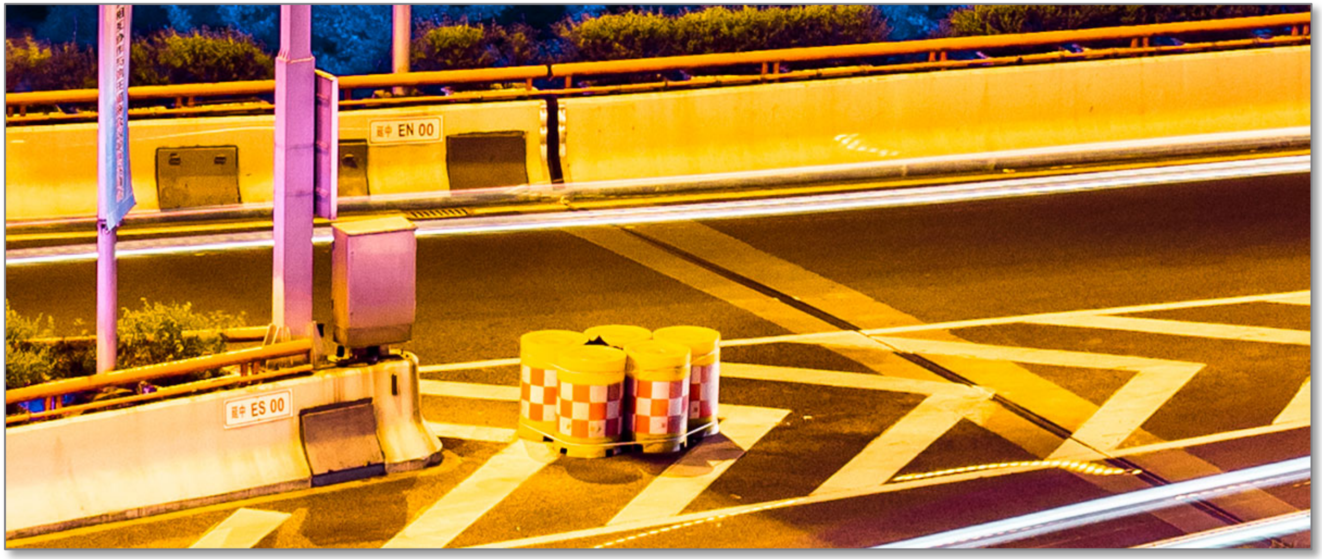
Anandan

Panoramic Mosaics

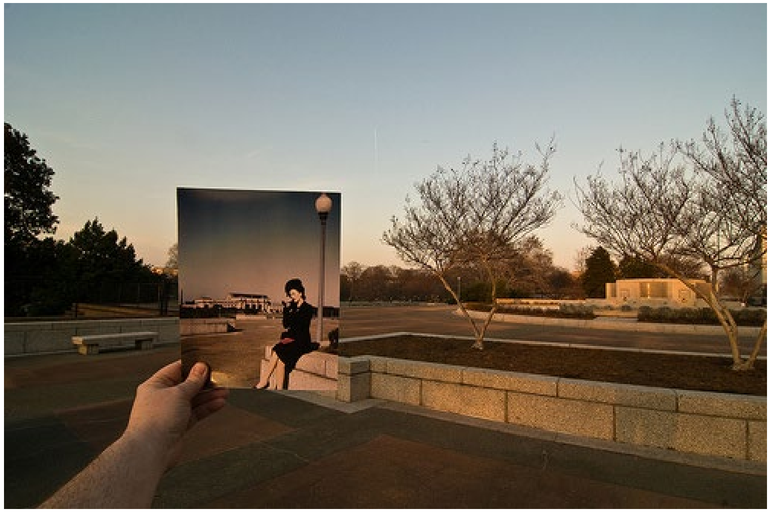


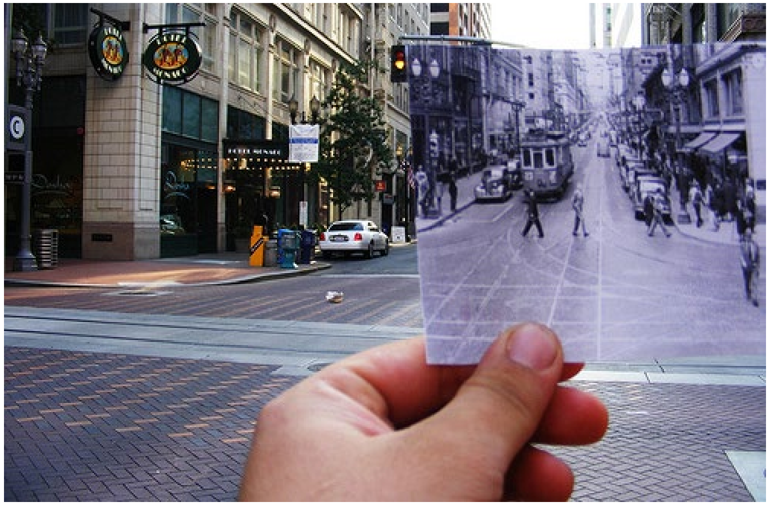
Gigapixel Images

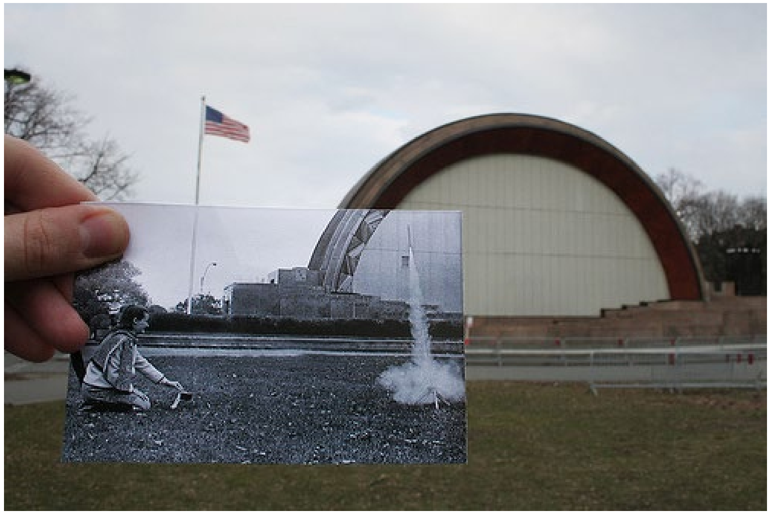




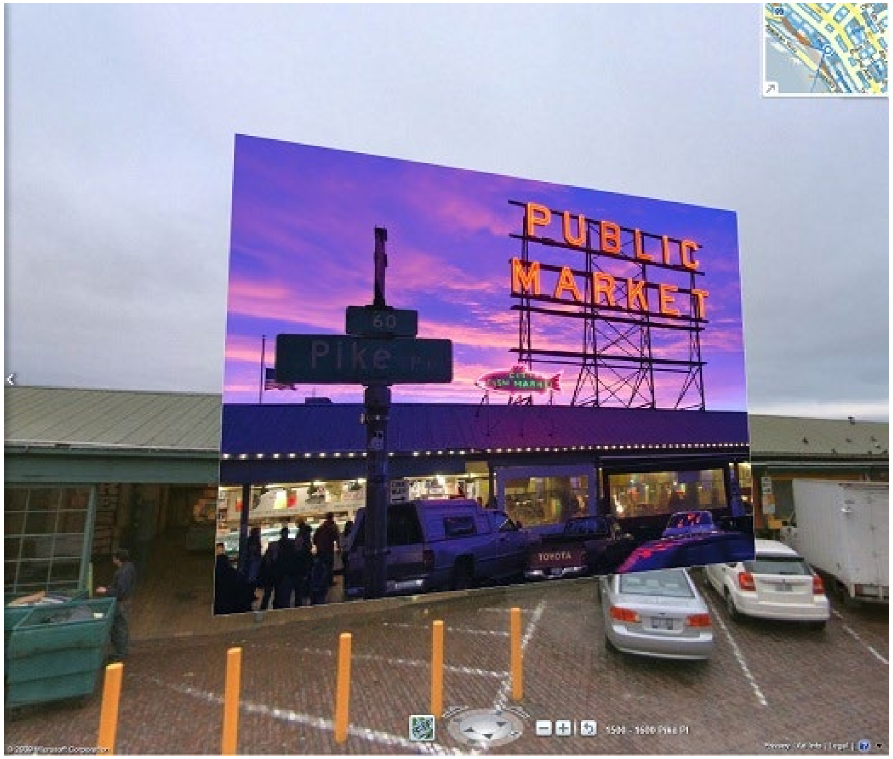
Applications – Look into the Past







Applications – Streetside Images



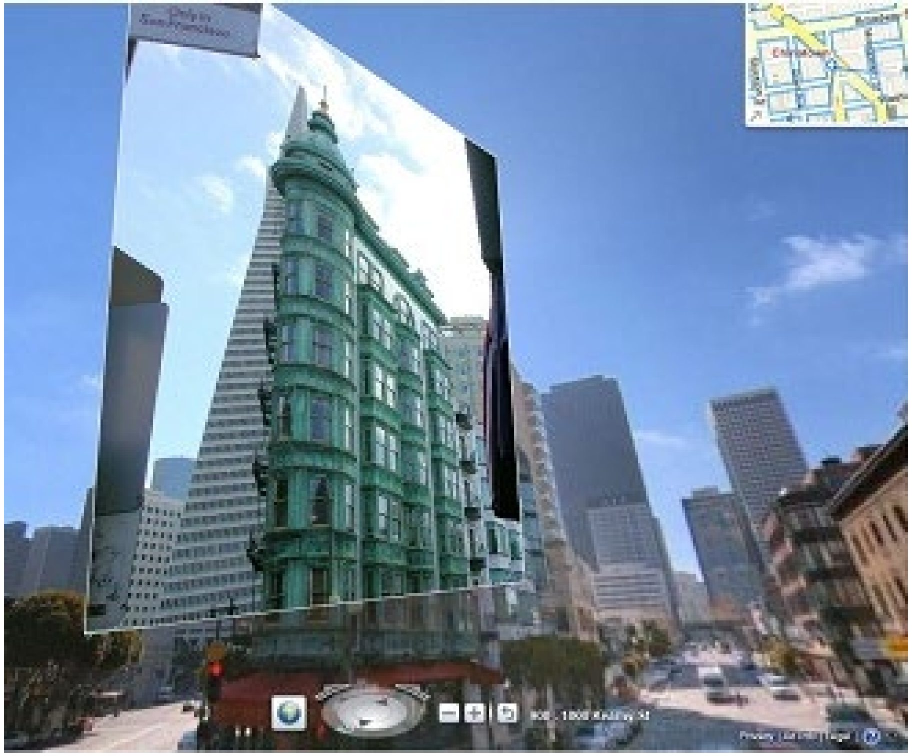
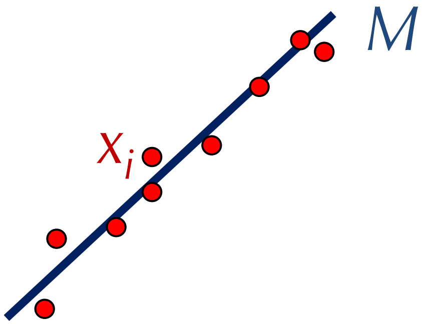


Image Alignment Approaches

- Direct alignment: see which image transformation maximizes similarity in overlap region
  - Often performed coarse-to-fine
- Feature-based alignment: find image transformation that matches keypoint locations

Alignment as Fitting

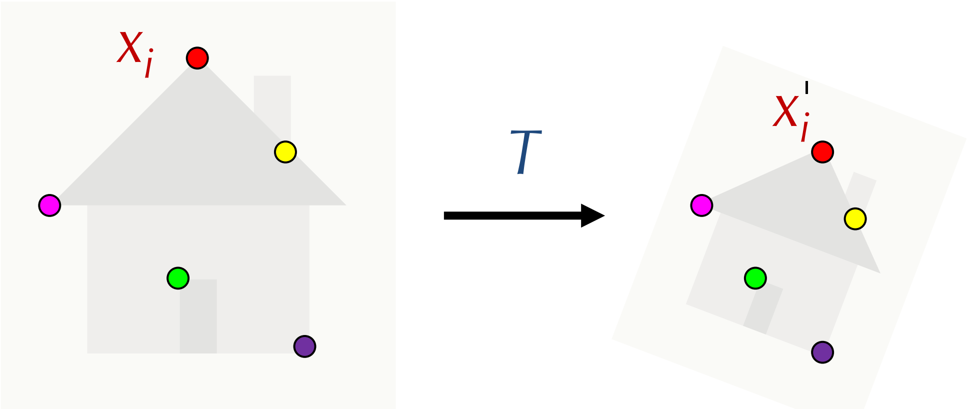
- Previously: fitting a model to features in one image



Find model *M* that minimizes

$$\sum_i \mathrm{L}(x_i; M)$$

- Alignment: fitting a model to a transformation between pairs of features (matches) in two images



Find transformation *T* that minimizes

$$\sum_i \mathrm{L}(T(x_i); x_i')$$

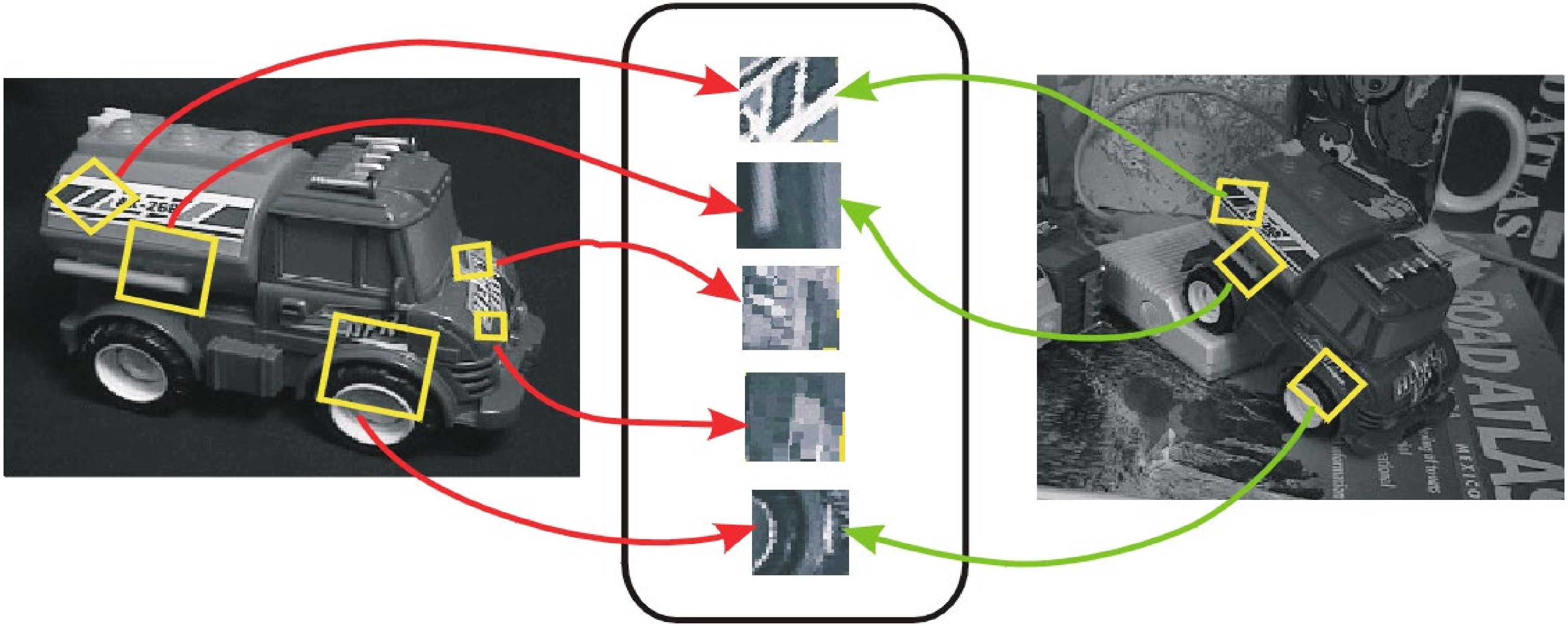
Feature-Based Alignment

- Find keypoints; compute SIFT descriptors
- Generate candidate keypoint matches
- Use RANSAC to select a subset of matches
- Fit to find best image transformation
- Warp images according to transformation
- Blend images in overlapping regions

Feature-Based Alignment

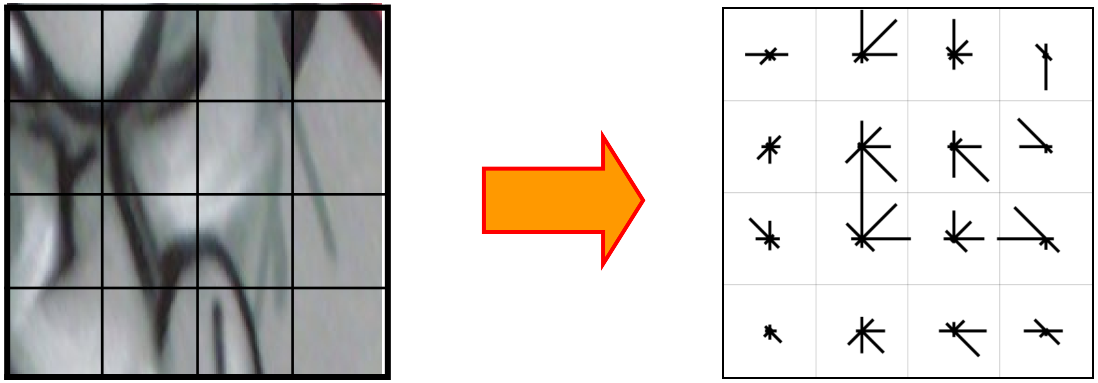
- Find keypoints; compute SIFT descriptors
- Generate candidate keypoint matches
- Use RANSAC to select a subset of matches
- Fit to find best image transformation
- Warp images according to transformation
- Blend images in overlapping regions

Review: Feature Detection and Description



Review: SIFT Descriptors

- Descriptor computation:
  - Divide patch into 4x4 sub-patches
  - Compute histogram of gradient orientations (8 angles) inside each sub-patch
  - Resulting descriptor: 4x4x8 = 128 dimensions



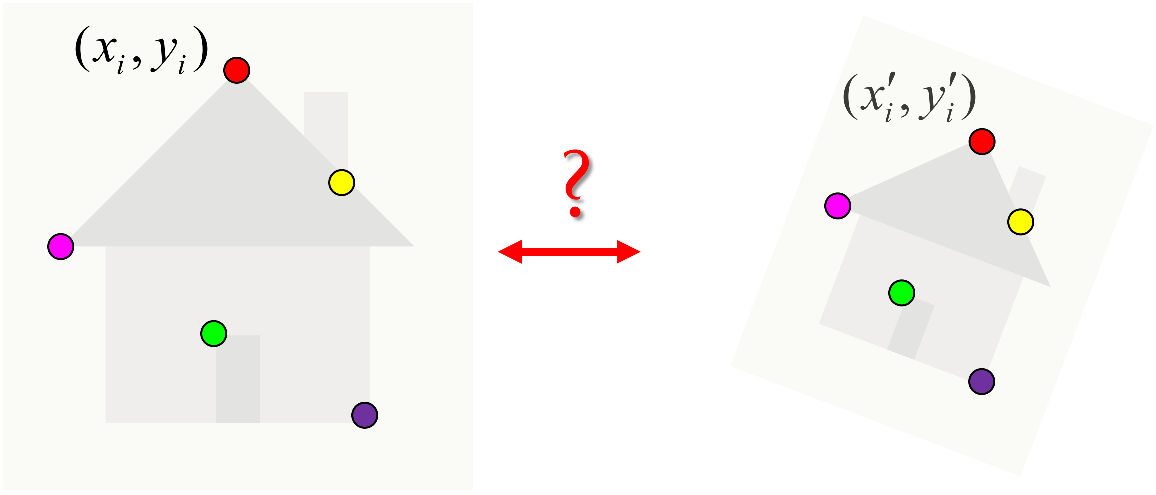
David G. Lowe. "Distinctive image features from scale-invariant keypoints."
IJCV 60 (2), pp. 91-110, 2004.

Feature-Based Alignment

- Find keypoints; compute SIFT descriptors
- Generate candidate keypoint matches
- Use RANSAC to select a subset of matches
- Fit to find best image transformation
- Warp images according to transformation
- Blend images in overlapping regions

Candidate Matches

- For a given keypoint in image A, how to find candidate match in image B?



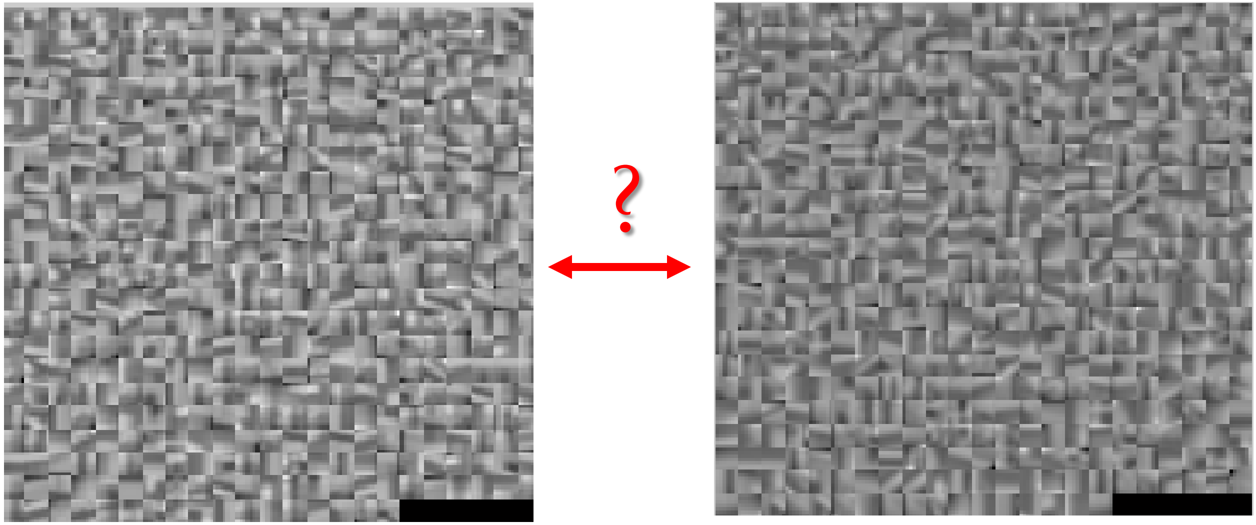
Candidate Matches

- For each SIFT descriptor in image A, find closest (according to Euclidean distance) in image B

$$best\_match(x) = \arg\min_{x_i'} \|x - x_i'\|^2$$

Candidate Matches

- For a given keypoint in image A, how to find candidate match in image B?
  - What if there are a lot of keypoints?



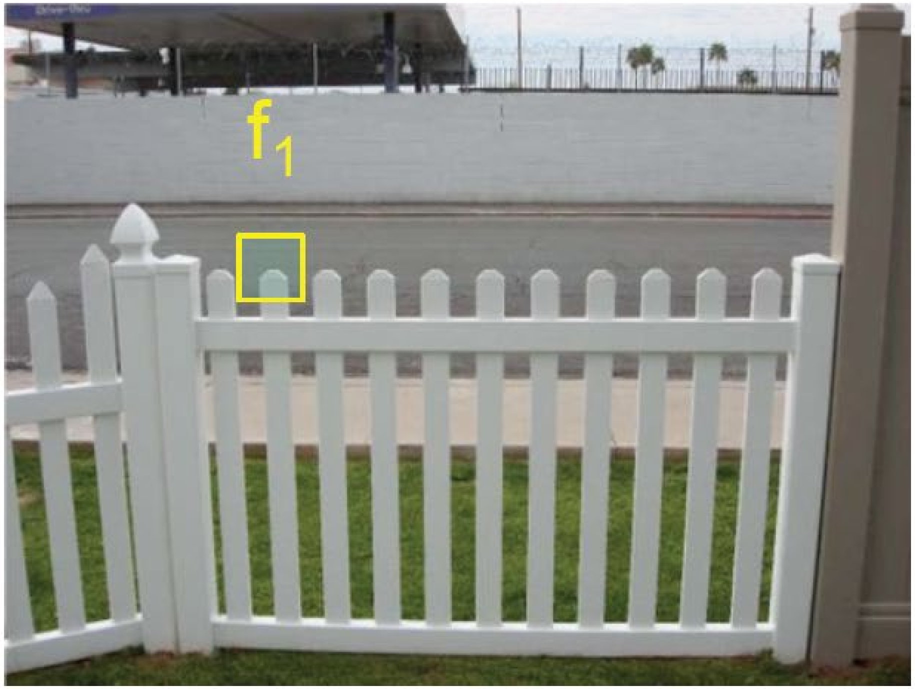
Candidate Matches

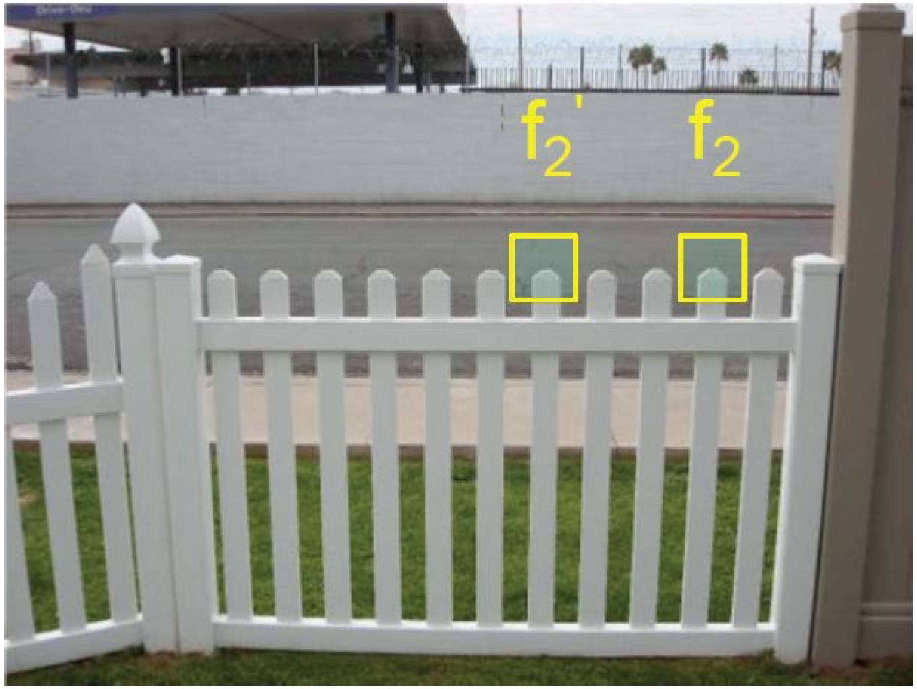
- For each SIFT descriptor in image A, find closest (according to Euclidean distance) in image B

$$best\_match(x) = \arg\min_{x_i{}'} \|x - x_i{}'\|^2$$

- Accelerate using k-d trees

Problem: Ambiguous Correspondences





Source: Y. Furukawa

Candidate Matches

- For each SIFT descriptor in image A, find closest (according to Euclidean distance) in image B

$$best\_match(x) = \arg\min_{x_i'} \|x - x_i'\|^2$$

- Accelerate using k-d trees
- Refinement: mutual best match
  - x' is most similar to x and x is most similar to x'

Candidate Matches

- For each SIFT descriptor in image A, find closest (according to Euclidean distance) in image B

$$best\_match(x) = \arg\min_{x_i'} \|x - x_i'\|^2$$

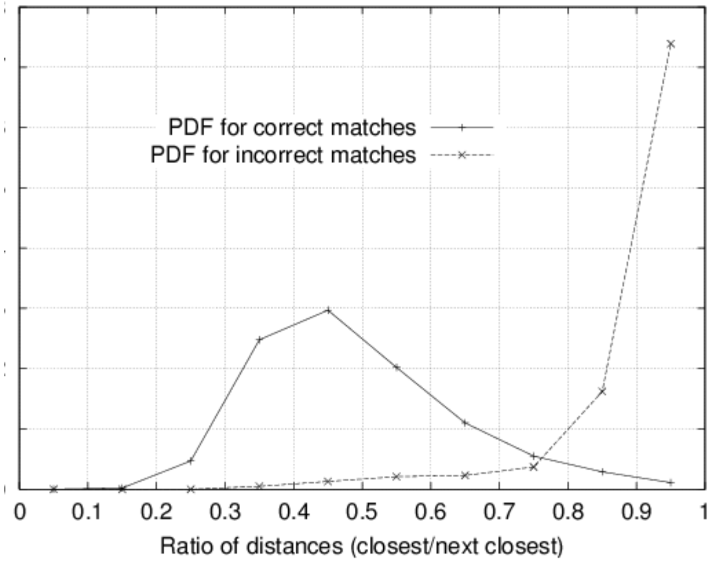
- Accelerate using k-d trees
- Refinement: mutual best match
  - x' is most similar to x and x is most similar to x'

Candidate Matches

- For each SIFT descriptor in image A, find closest (according to Euclidean distance) in image B

$$best\_match(x) = \arg\min_{x_i'} \|x - x_i'\|^2$$

- Accelerate using k-d trees
- Refinement: mutual best match
- Refinement: best match is much better than second-best
  - Ratio of second-closest to closest is high for non-distinctive features
  - Threshold ratio of e.g. 0.8



[Lowe]

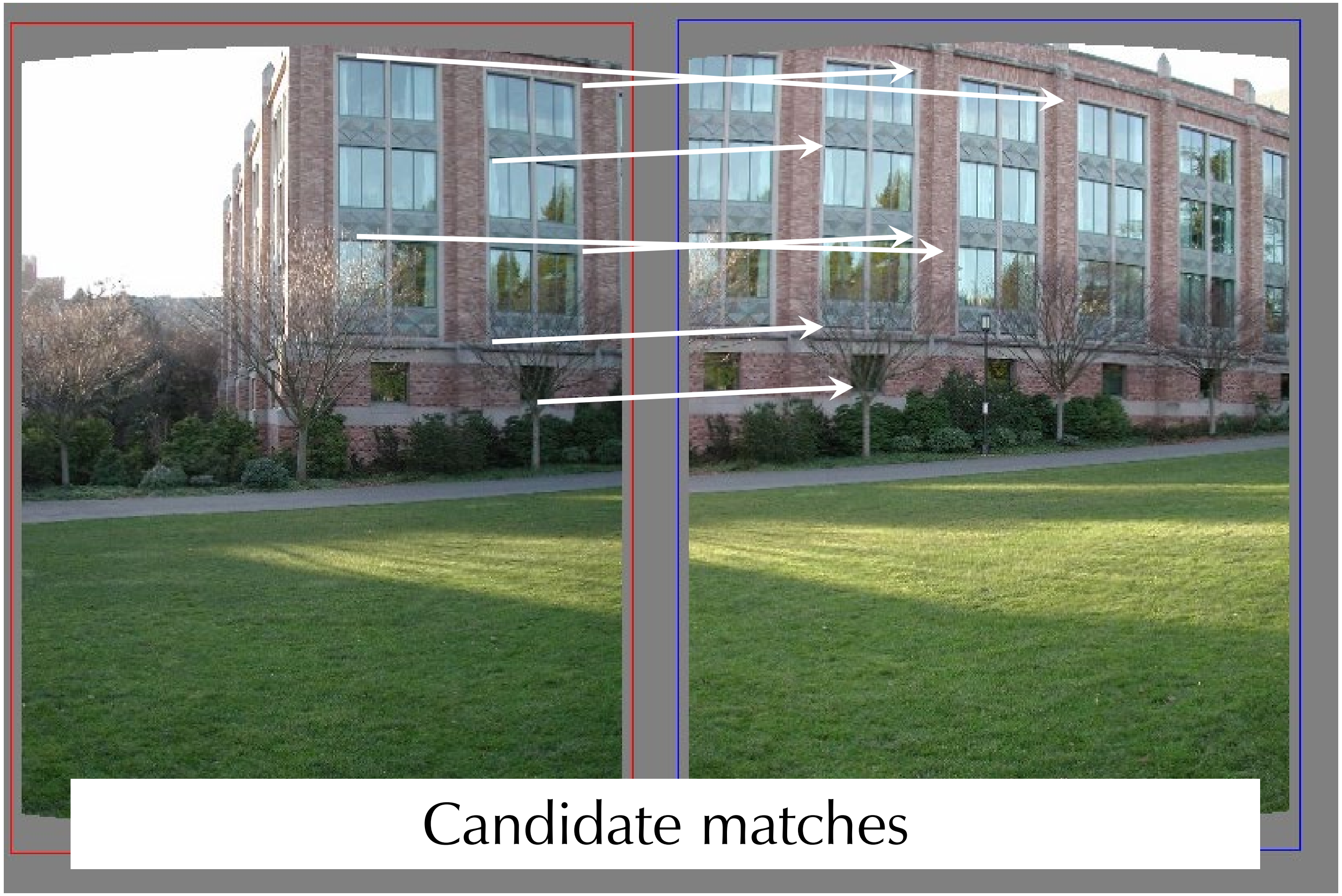
Feature-Based Alignment

- Find keypoints; compute SIFT descriptors
- Generate candidate keypoint matches
- Use RANSAC to select a subset of matches
- Fit to find best image transformation
- Warp images according to transformation
- Blend images in overlapping regions

Review: RANSAC

- Set of candidate matches contains many outliers
- RANSAC loop:
  - Randomly select a minimal set of matches
  - Compute transformation from seed group
  - Find inliers to this transformation
  - Keep the transformation with the largest number of inliers
- At end, re-estimate best transform using all inliers

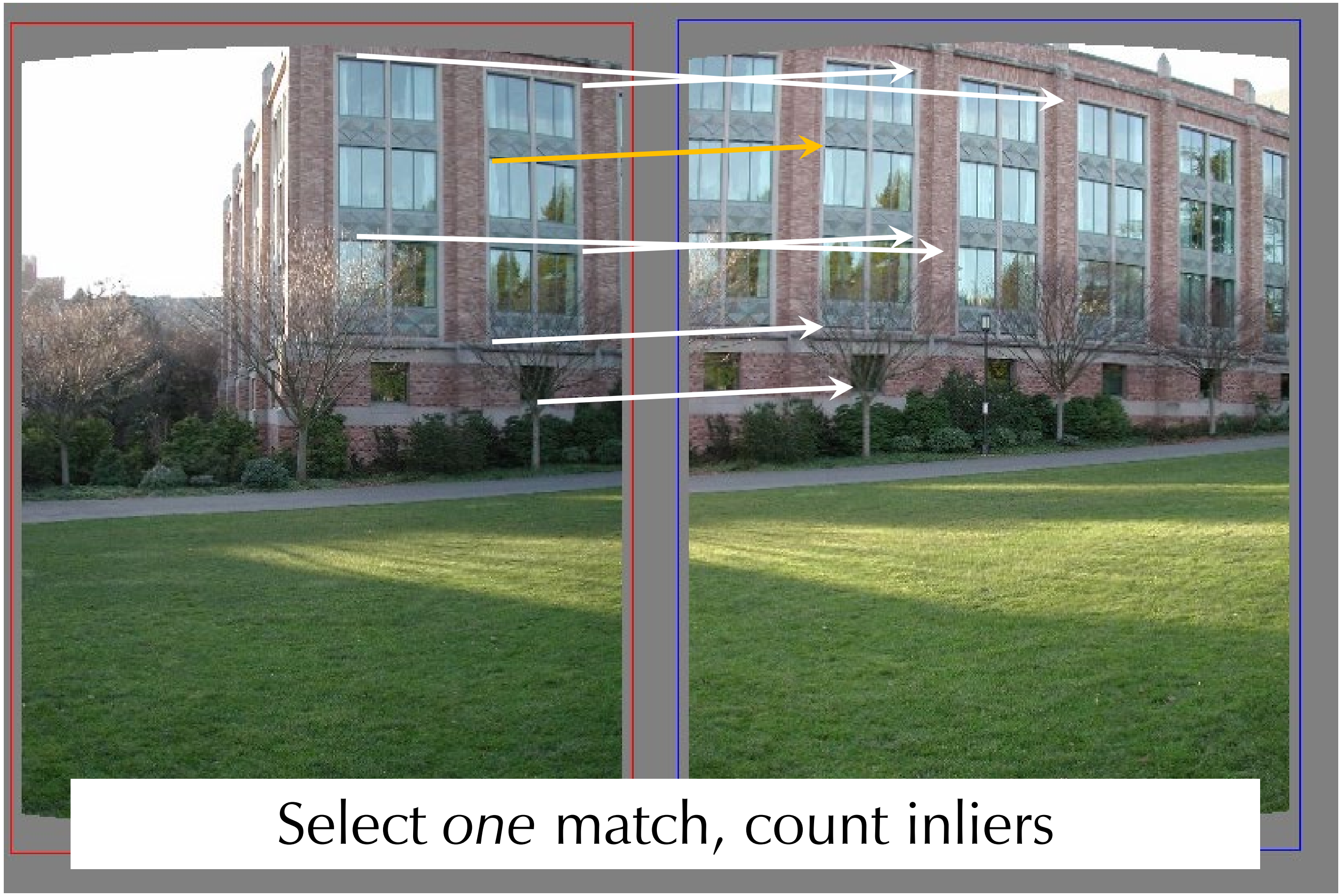
RANSAC: Translation Only



Candidate matches

[Szeliski]

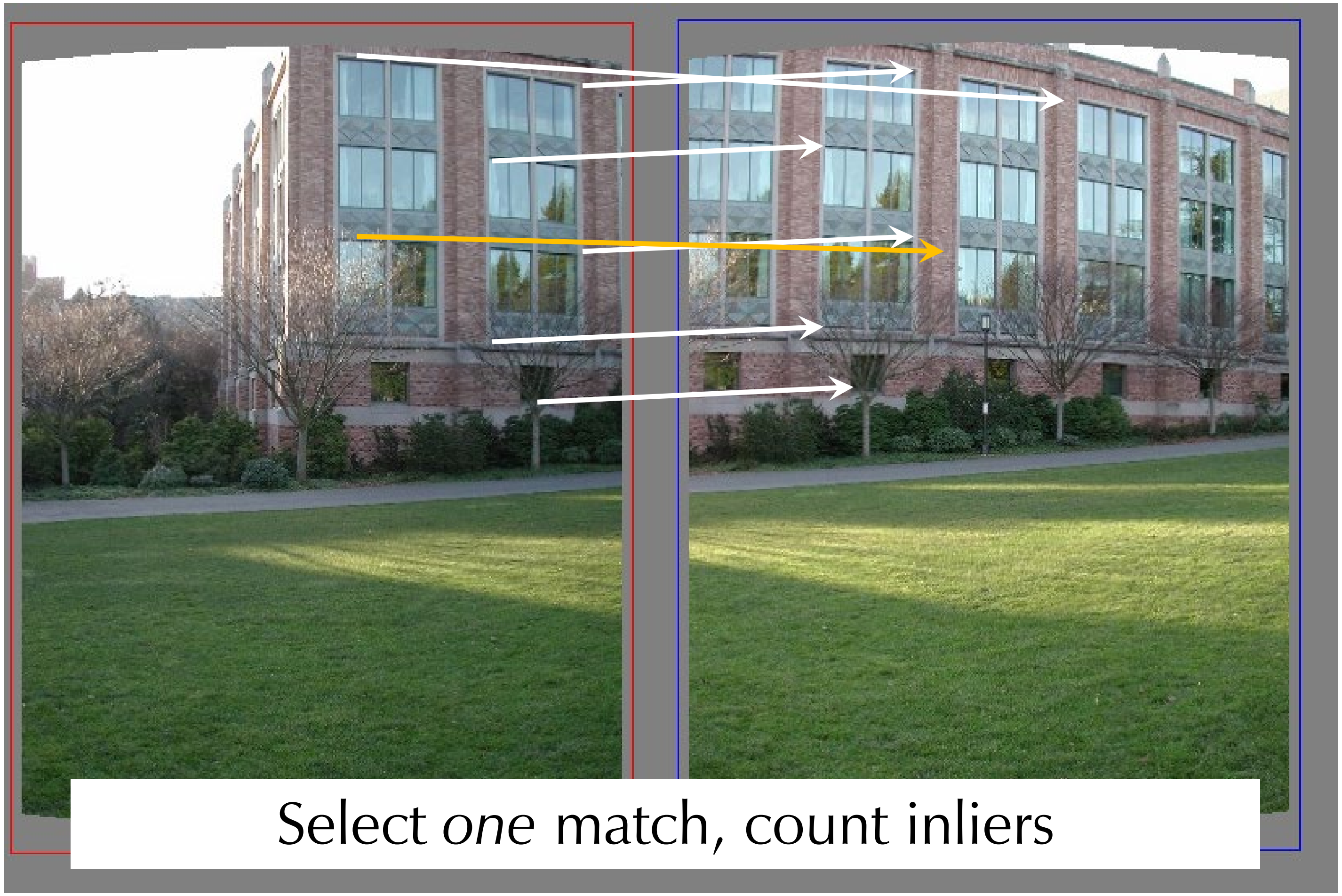
RANSAC: Translation Only



Select *one* match, count inliers

[Szeliski]

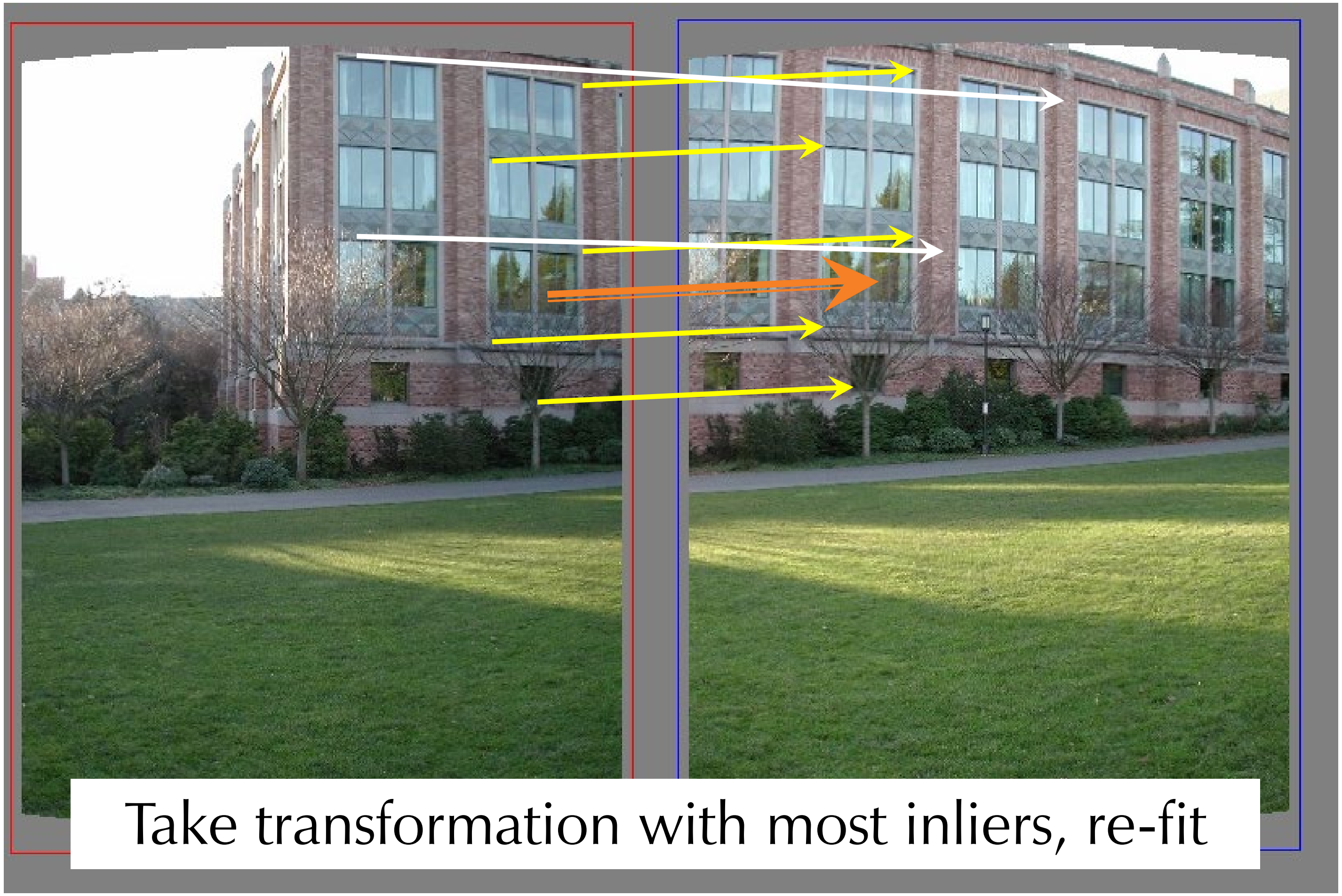
RANSAC: Translation Only



Select *one* match, count inliers

[Szeliski]

RANSAC: Translation Only



Take transformation with most inliers, re-fit

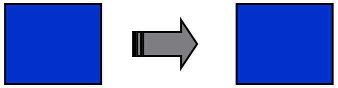
[Szeliski]

Feature-Based Alignment

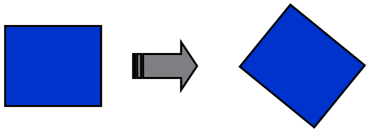
- Find keypoints; compute SIFT descriptors
- Generate candidate keypoint matches
- Use RANSAC to select a subset of matches
- Fit to find best image transformation
- Warp images according to transformation
- Blend images in overlapping regions

2D Transformation Models

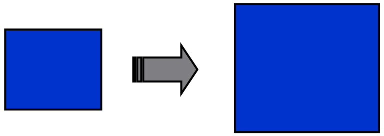
- Translation only



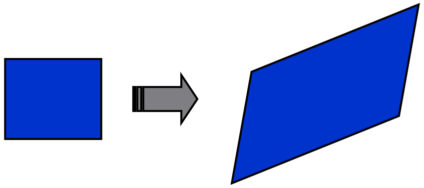
- Rigid body (translation+rotation)



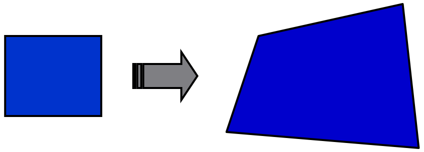
- Similarity (translation+rotation+scale)



- Affine



- Homography (projective)

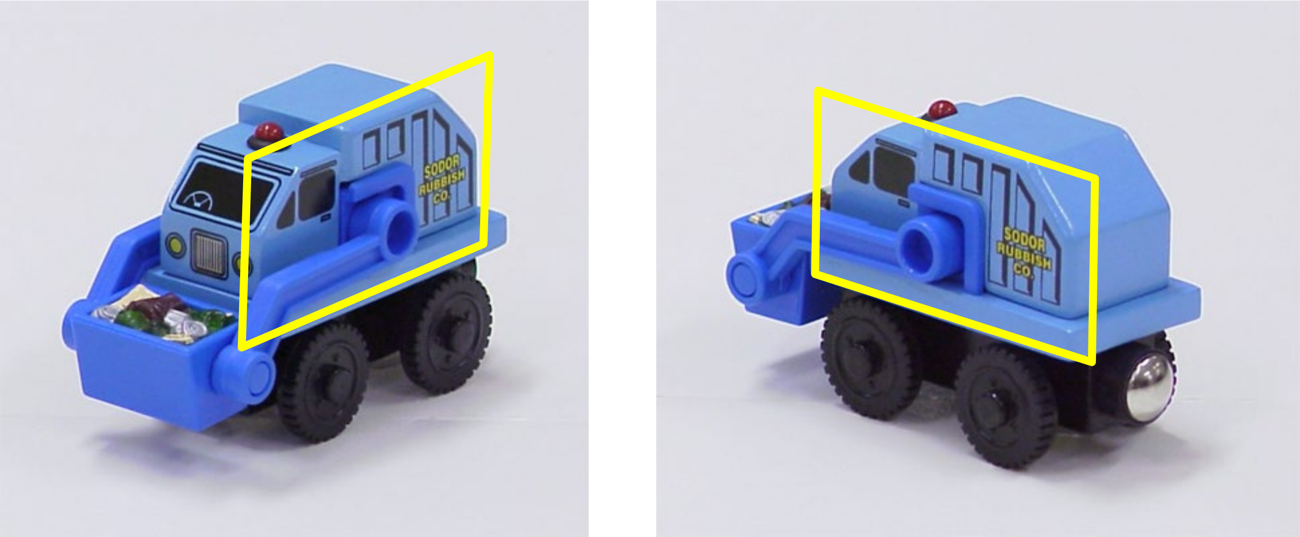


2D Transformation Models

Translation	$x' = x + t_x$ $y' = y + t_y$	2 unknowns	1 point
Rigid body	$x' = x\cos\theta - y\sin\theta + t_x$ $y' = x\sin\theta + y\cos\theta + t_y$	3 unknowns	"1.5" points
Similarity	$x' = Sx\cos\theta - Sy\sin\theta + t_x$ $y' = Sx\sin\theta + Sy\cos\theta + t_y$	4 unknowns	2 points
Affine	$x' = ax + by + t_x$ $y' = cx + dy + t_y$	6 unknowns	3 points
Homography	$x' = \frac{ax + by + c}{gx + hy + i}$ $y' = \frac{dx + ey + f}{gx + hy + i}$	8 unknowns	4 points

Fitting: Affine

- Simple fitting procedure (linear least squares)
- Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras
- Initialize fitting for more complex models



Lazebnik

Fitting: Affine

$$x' = ax + by + t_x$$
$$y' = cx + dy + t_y$$

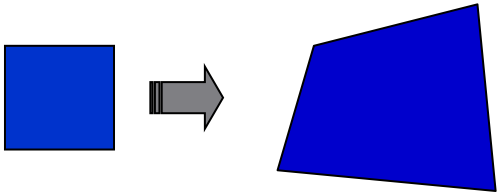
$$\begin{bmatrix} x_1 & y_1 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & y_1 & 0 & 1 \\ x_2 & y_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_2 & y_2 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ \vdots \end{bmatrix}$$

- Linear system with six unknowns
- Each match gives us two linearly independent equations: need at least three to solve for parameters
- Overconstrained if more than 3 points

$$Ax = b$$
$$x = (A^T A)^{-1} A^T b$$

Fitting: Homography

- Projective transformation:
  takes any quad to any other quad

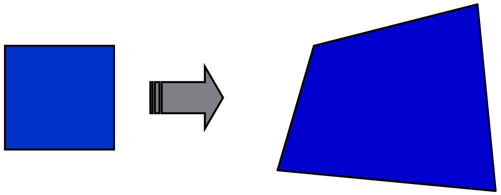


  – Transformation between two views of a planar surface

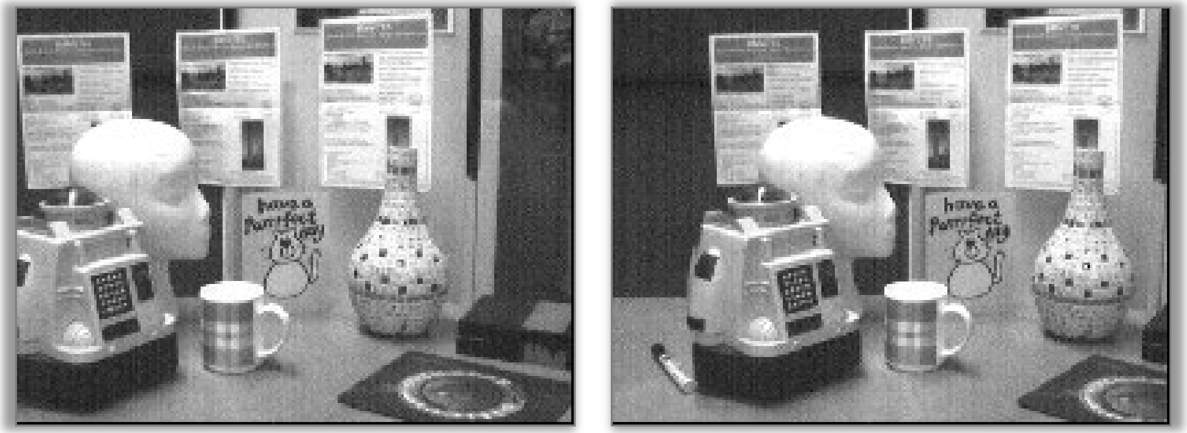
Lazebnik

Fitting: Homography

- Projective transformation:
  takes any quad to any other quad

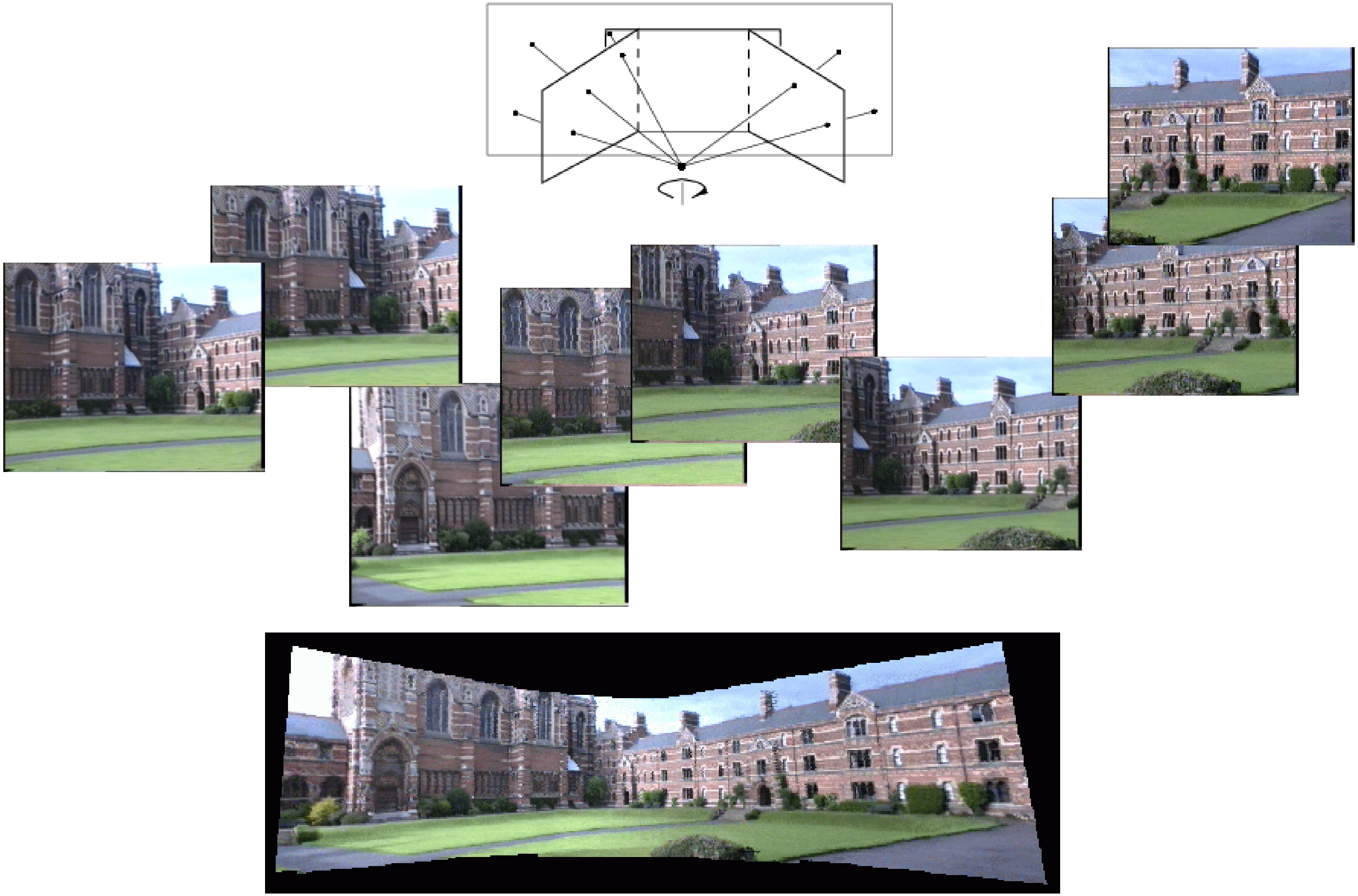


  - Transformation between images from two cameras that share the same center



Lazebnik

Application: Panorama Stitching



Source: Hartley & Zisserman

Fitting: Homography

$$x' = \frac{ax + by + c}{gx + hy + i}$$

$$y' = \frac{dx + ey + f}{gx + hy + i}$$

$$\begin{aligned} gxx' + hyx' + ix' &= ax + by + c \\ gxy' + hyy' + iy' &= dx + ey + f \end{aligned}$$

$$\begin{bmatrix} -x_1 & -y_1 & 1 & 0 & 0 & 0 & x_1x_1' & y_1x_1' & x_1' \\ 0 & 0 & 0 & -x_1 & -y_1 & 1 & x_1y_1' & y_1y_1' & y_1' \\ -x_2 & -y_2 & 1 & 0 & 0 & 0 & x_2x_2' & y_2x_2' & x_2' \\ 0 & 0 & 0 & -x_2 & -y_2 & 1 & x_2y_2' & y_2y_2' & y_2' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

Fitting: Homography

$$\begin{bmatrix} -x_1 & -y_1 & 1 & 0 & 0 & 0 & x_1x_1' & y_1x_1' & x_1' \\ 0 & 0 & 0 & -x_1 & -y_1 & 1 & x_1y_1' & y_1y_1' & y_1' \\ -x_2 & -y_2 & 1 & 0 & 0 & 0 & x_2x_2' & y_2x_2' & x_2' \\ 0 & 0 & 0 & -x_2 & -y_2 & 1 & x_2y_2' & y_2y_2' & y_2' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

- Underconstrained! For $\mathrm{A}x = 0$, $x = 0$ is a valid solution!
- Add constraint $\|x\| = 1$
- Solution (left as an exercise for the student J): x is the eigenvector corresponding to smallest eigenvalue of $\mathrm{A}^{\mathrm{T}}\mathrm{A}$

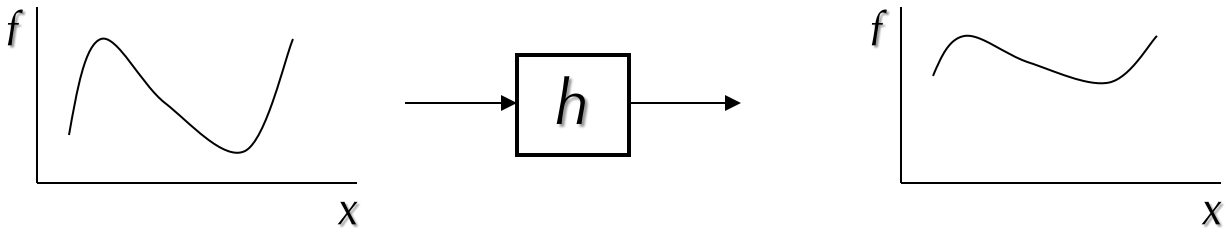
Feature-Based Alignment

- Find keypoints; compute SIFT descriptors
- Generate candidate keypoint matches
- Use RANSAC to select a subset of matches
- Fit to find best image transformation
- Warp images according to transformation
- Blend images in overlapping regions

Image Warping

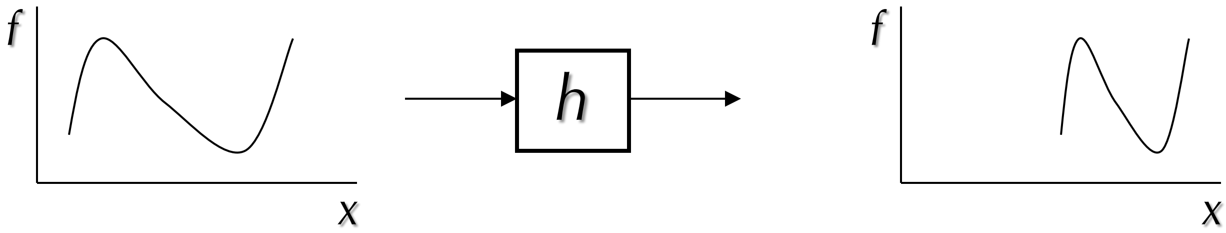
- Image filtering: change *range* of image

$$g(x) = h(f(x))$$



- Image warping: change *domain* of image

$$g(x) = f(h(x))$$

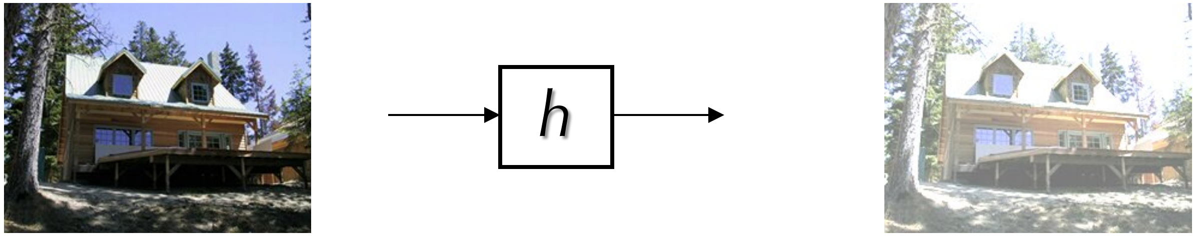


Richard Szeliski

Image Warping

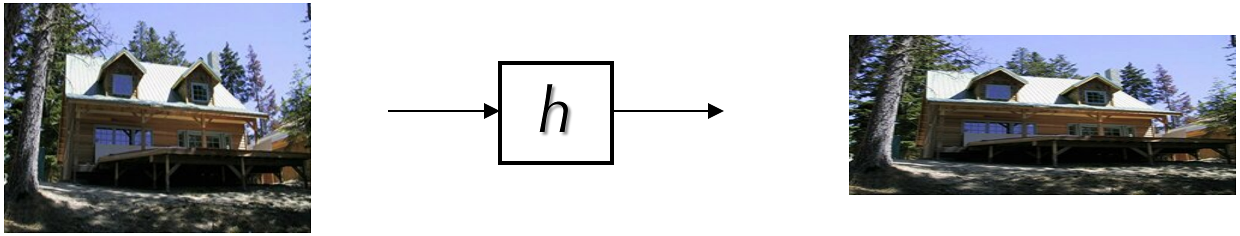
- Image filtering: change *range* of image

$$g(x) = h(f(x))$$



- Image warping: change *domain* of image

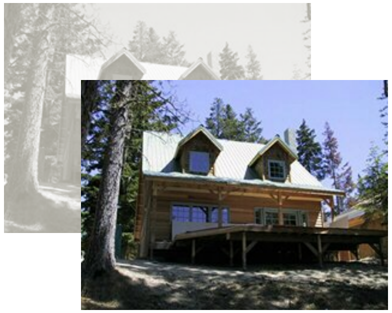
$$g(x) = f(h(x))$$



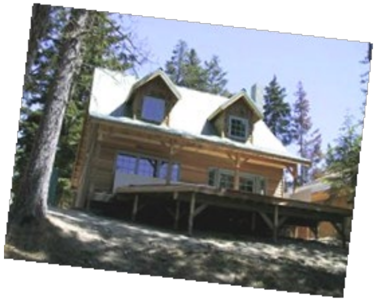
Richard Szeliski

Parametric (Global) Warping

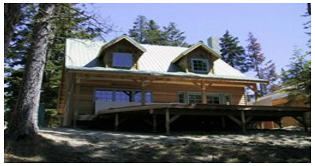
- Examples of parametric warps:



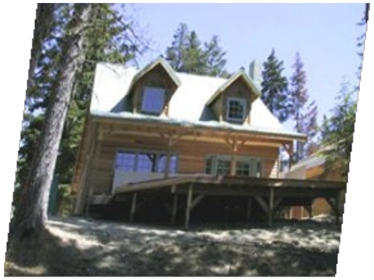
translation



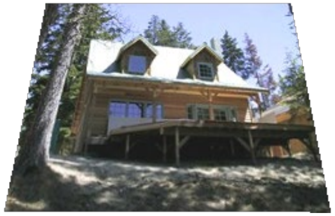
rotation



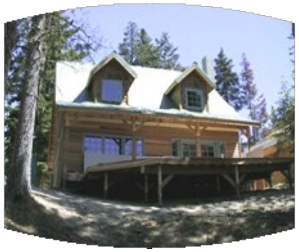
aspect



affine



perspective

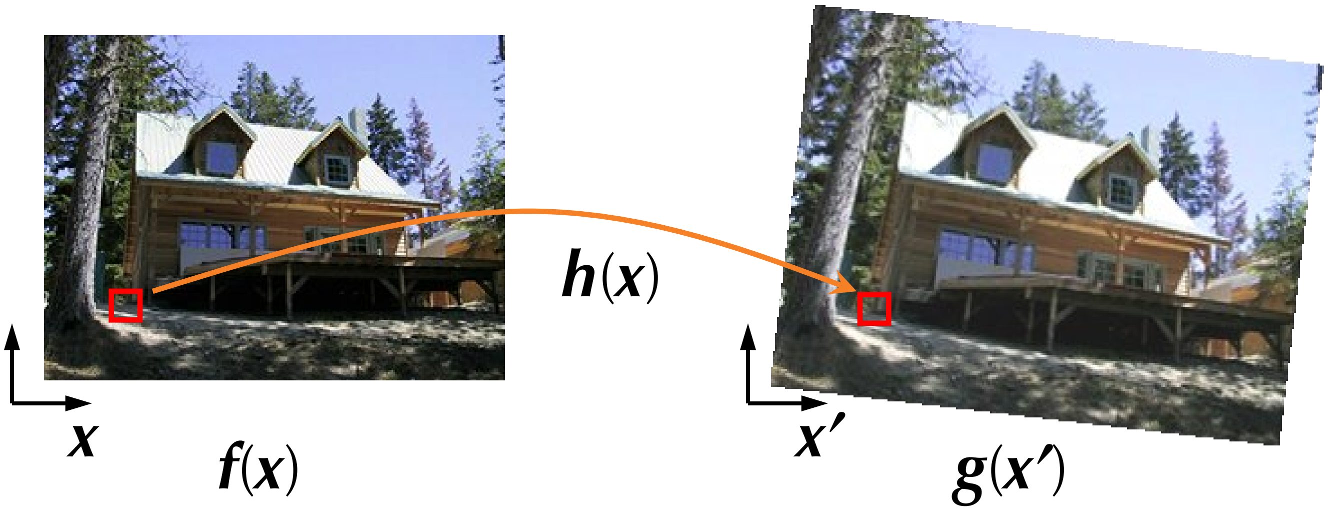


cylindrical

Richard Szeliski

Image Warping

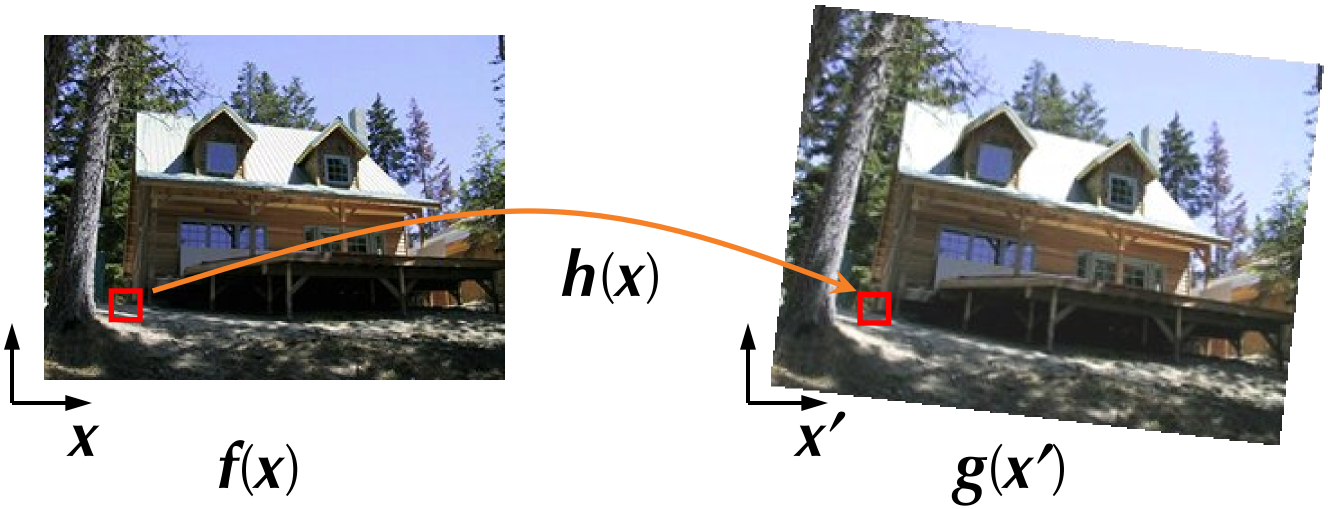
- Given a coordinate transform $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$ and a source image $\boldsymbol{f}(\boldsymbol{x})$, how do we compute a transformed image $\boldsymbol{g}(\boldsymbol{x}') = \boldsymbol{f}(\boldsymbol{h}(\boldsymbol{x}))$?


$\boldsymbol{h}(\boldsymbol{x})$
$\boldsymbol{x}$
$\boldsymbol{f}(\boldsymbol{x})$
$\boldsymbol{x}'$
$\boldsymbol{g}(\boldsymbol{x}')$

Richard Szeliski

Forward Warping

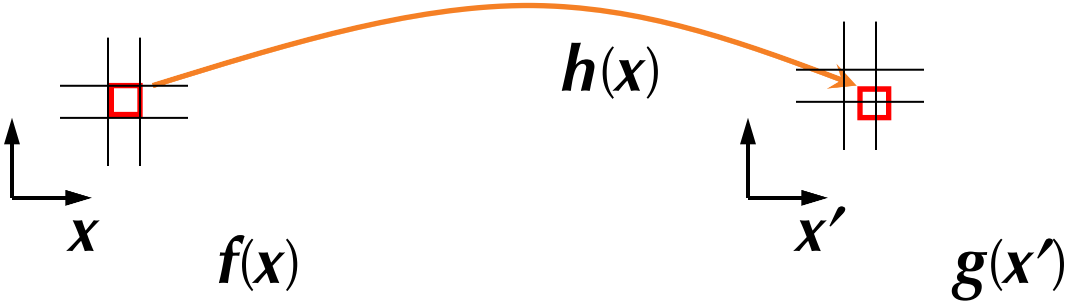
- Send each pixel $\boldsymbol{f}(\mathbf{x})$ to its corresponding location $\mathbf{x}' = \boldsymbol{h}(\mathbf{x})$ in $\boldsymbol{g}(\mathbf{x}')$
    - What if pixel lands "between" two pixels?



Richard Szeliski

Forward Warping

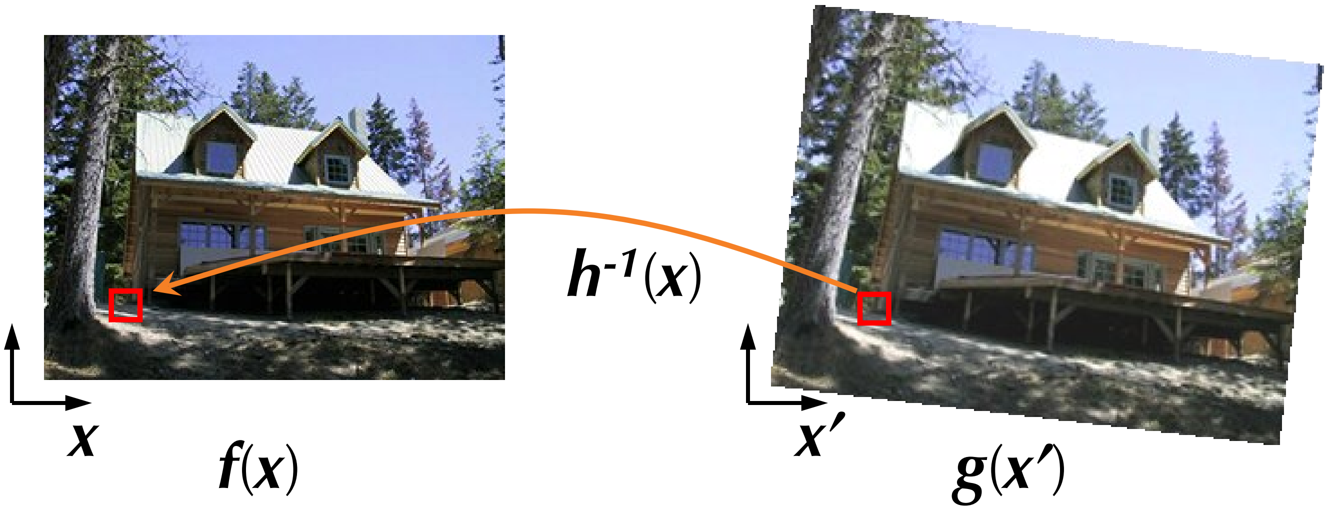
- Send each pixel $\boldsymbol{f}(\boldsymbol{x})$ to its corresponding location $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$ in $\boldsymbol{g}(\boldsymbol{x}')$
  - What if pixel lands "between" two pixels?
  - Answer: add "contribution" to several pixels, normalize later (*splatting*)



Richard Szeliski

Inverse Warping

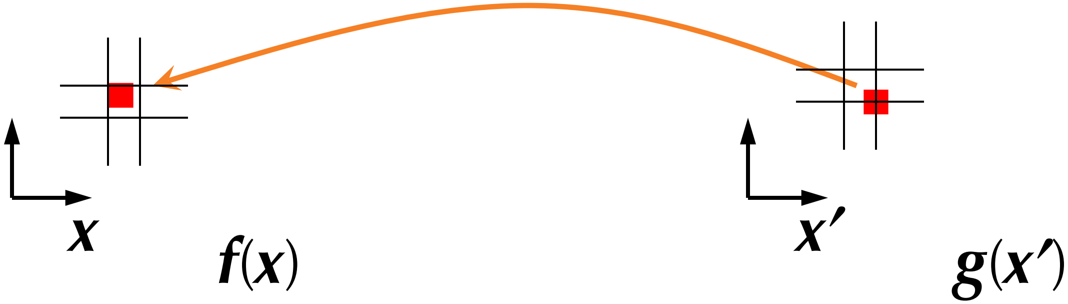
- Get each pixel $\mathbf{g}(\mathbf{x}')$ from its corresponding location $\mathbf{x}' = \mathbf{h}(\mathbf{x})$ in $\mathbf{f}(\mathbf{x})$
  - What if pixel comes from "between" two pixels?



Richard Szeliski

Inverse Warping

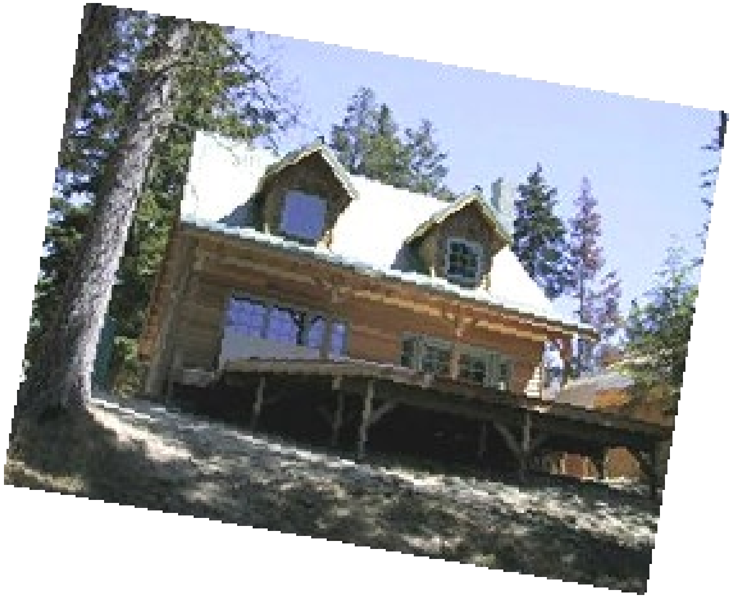
- Get each pixel $\mathbf{g}(\mathbf{x}')$ from its corresponding location $\mathbf{x}' = \mathbf{h}(\mathbf{x})$ in $\mathbf{f}(\mathbf{x})$
  - What if pixel comes from "between" two pixels?
  - Answer: *resample* color value from *interpolated* (*prefiltered*) source image



Richard Szeliski

Interpolation

- Possible interpolation filters:
  - nearest neighbor
  - bilinear
  - bicubic (interpolating)
  - sinc / FIR
- See COS 426 for details on how to avoid “jaggies”



Feature-Based Alignment

- Find keypoints; compute SIFT descriptors
- Generate candidate keypoint matches
- Use RANSAC to select a subset of matches
- Fit to find best image transformation
- Warp images according to transformation
- Blend images in overlapping regions

Blending

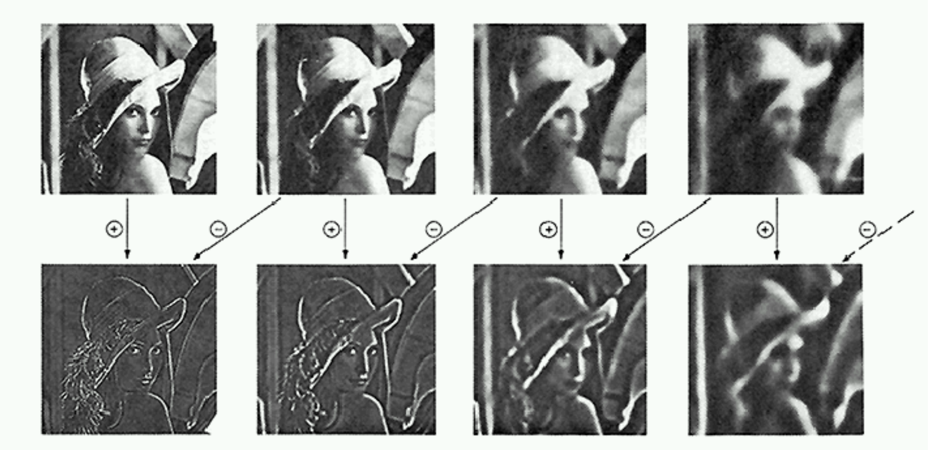
- Blend over too small a region: seams
- Blend over too large a region: ghosting

Multiresolution Blending

- Different blending regions for different levels in a pyramid [Burt & Adelson]
  - Blend low frequencies over large regions (minimize seams due to brightness variations)
  - Blend high frequencies over small regions (minimize ghosting)

Pyramid Creation

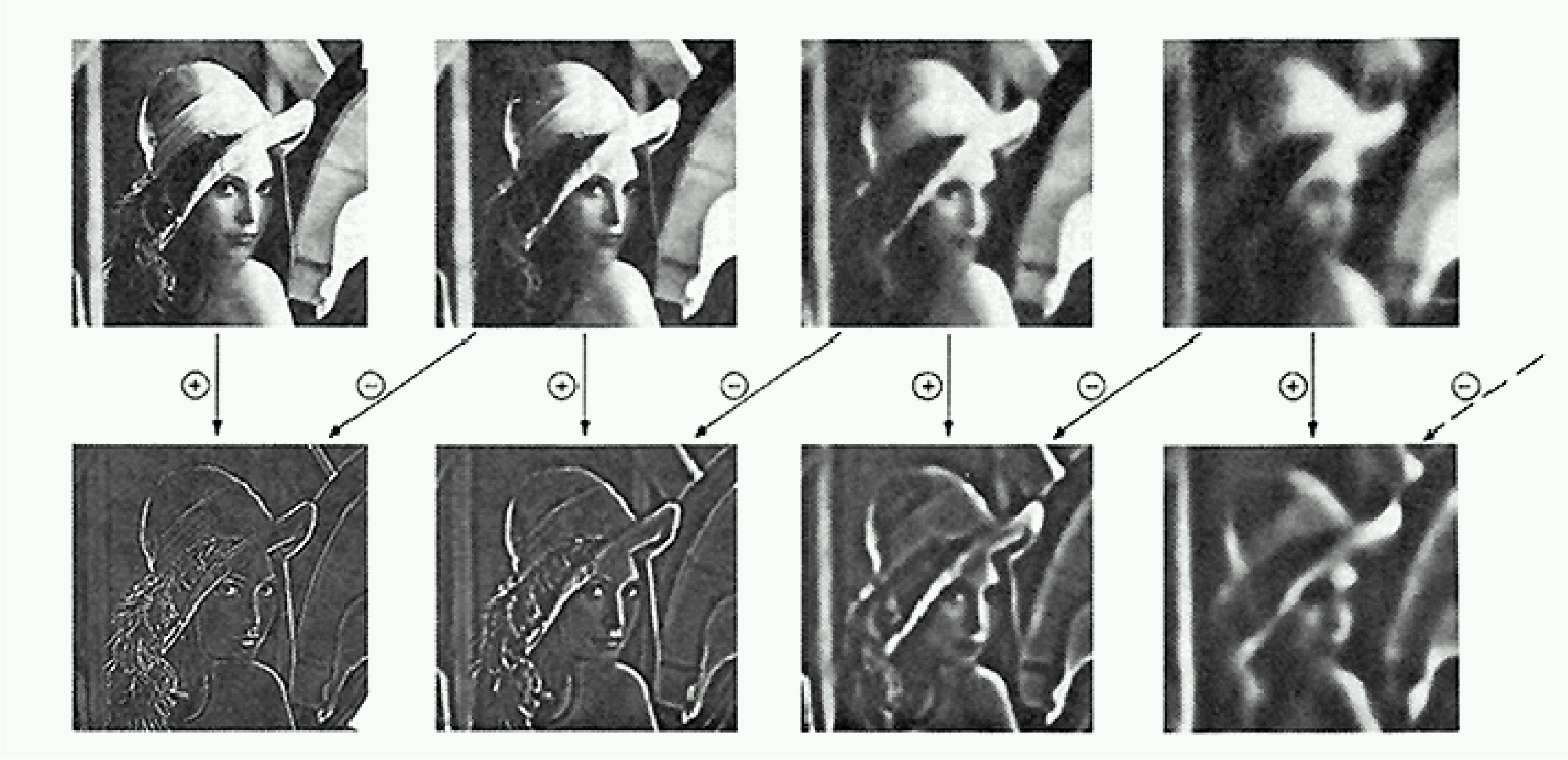
- "Gaussian" Pyramid
- "Laplacian" Pyramid
    - Created from Gaussian pyramid by subtraction $L_i = G_i - \text{expand}(G_{i+1})$



Richard Szeliski

Octaves in the Spatial Domain

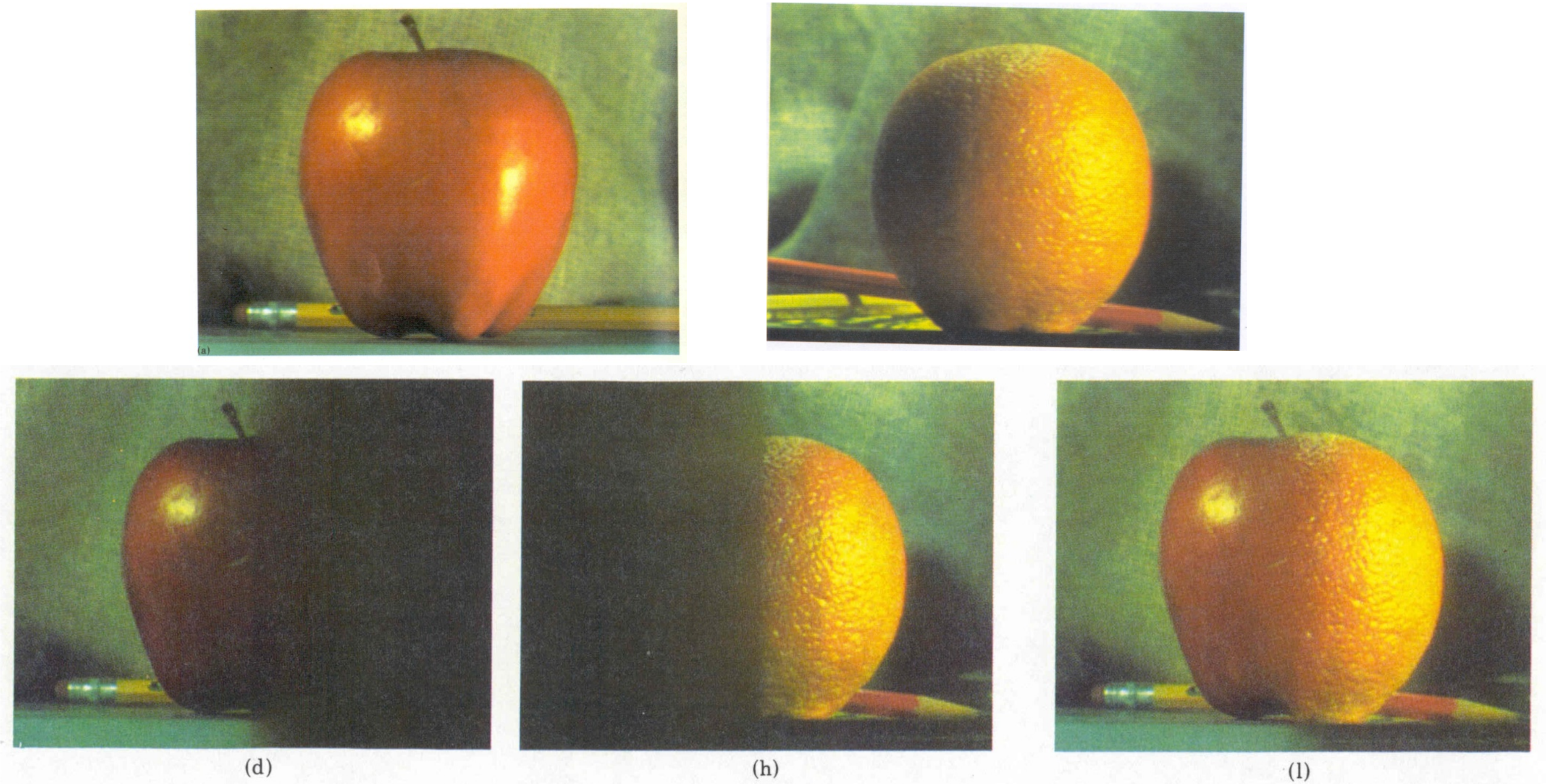
Lowpass Images



Bandpass Images

Richard Szeliski

Pyramid Blending



(d)

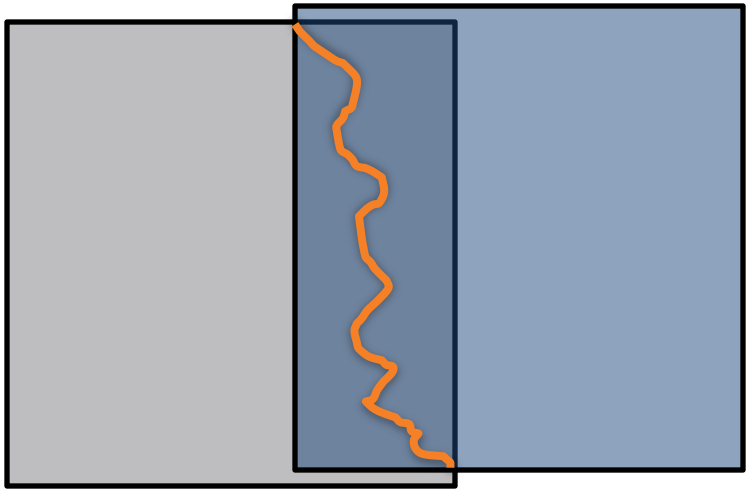
(h)

(l)

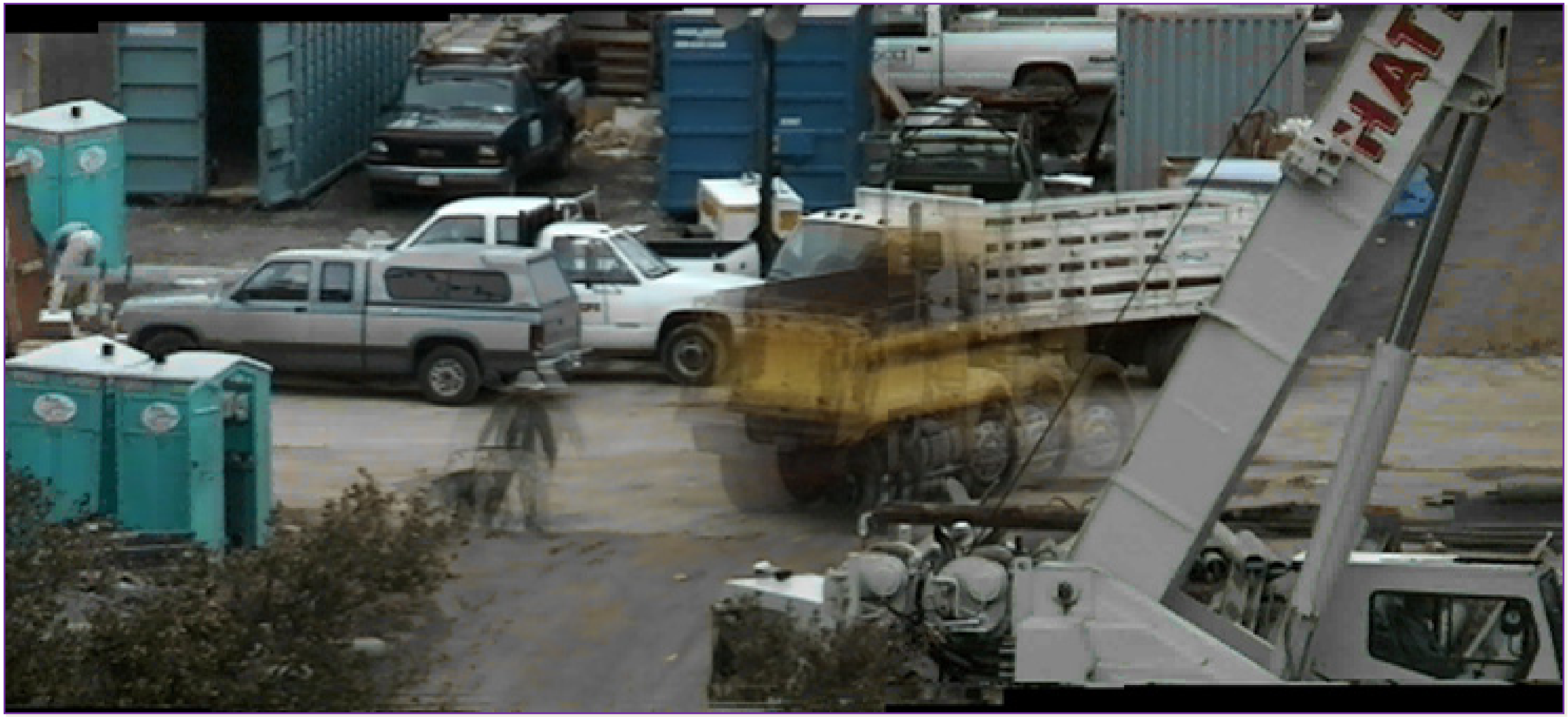
Richard Szeliski

Minimum-Cost Cuts

- Instead of blending high frequencies along a straight line, blend along line of minimum differences in image intensities



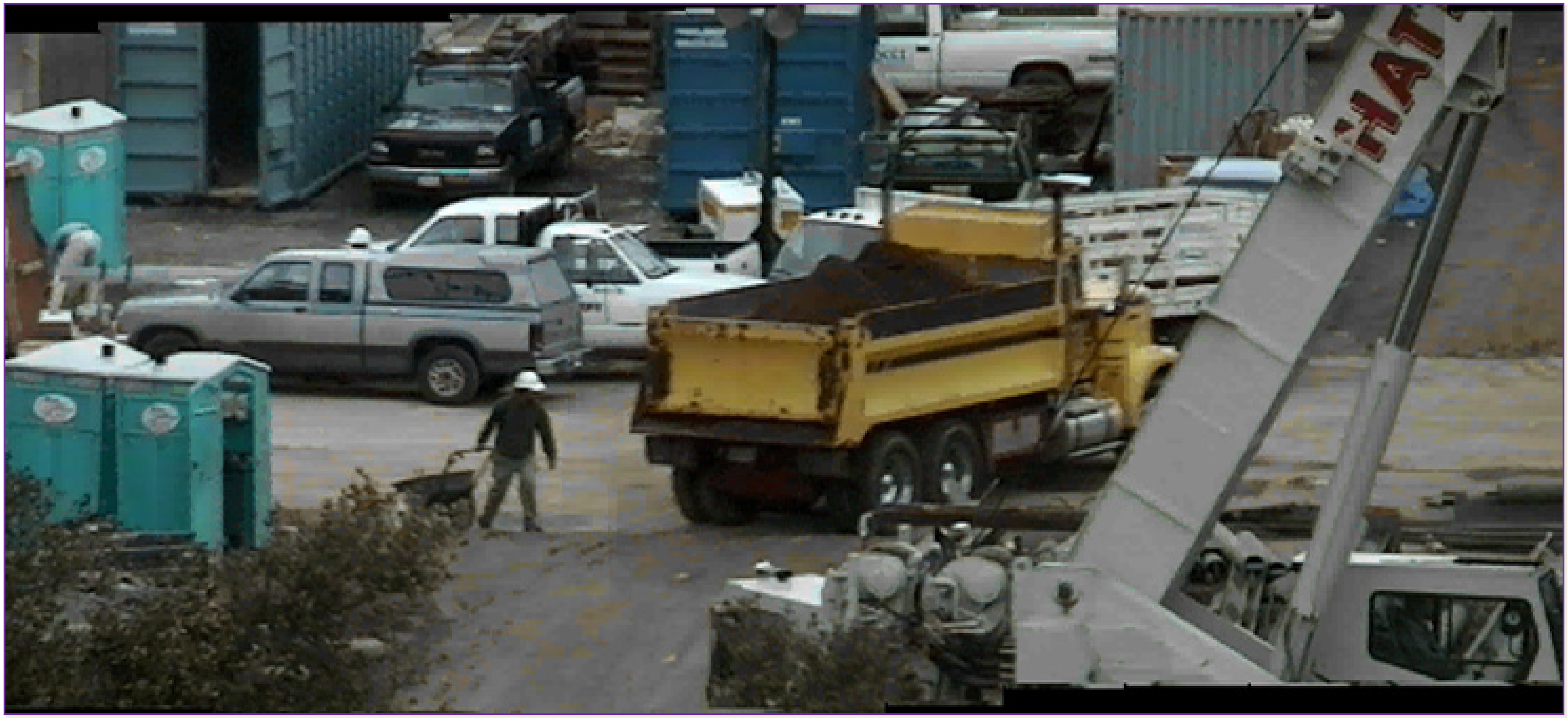
Minimum-Cost Cuts



Moving object, simple blending → blur

[Davis 98]

Minimum-Cost Cuts

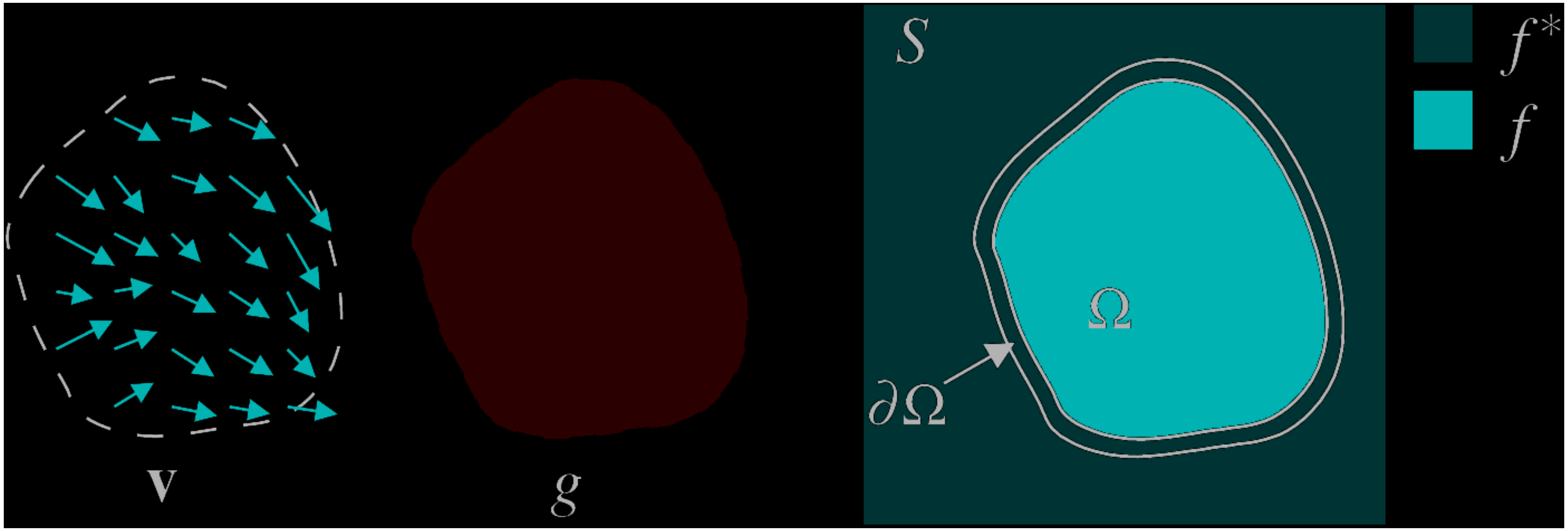


Minimum-cost cut → no blur

[Davis 98]

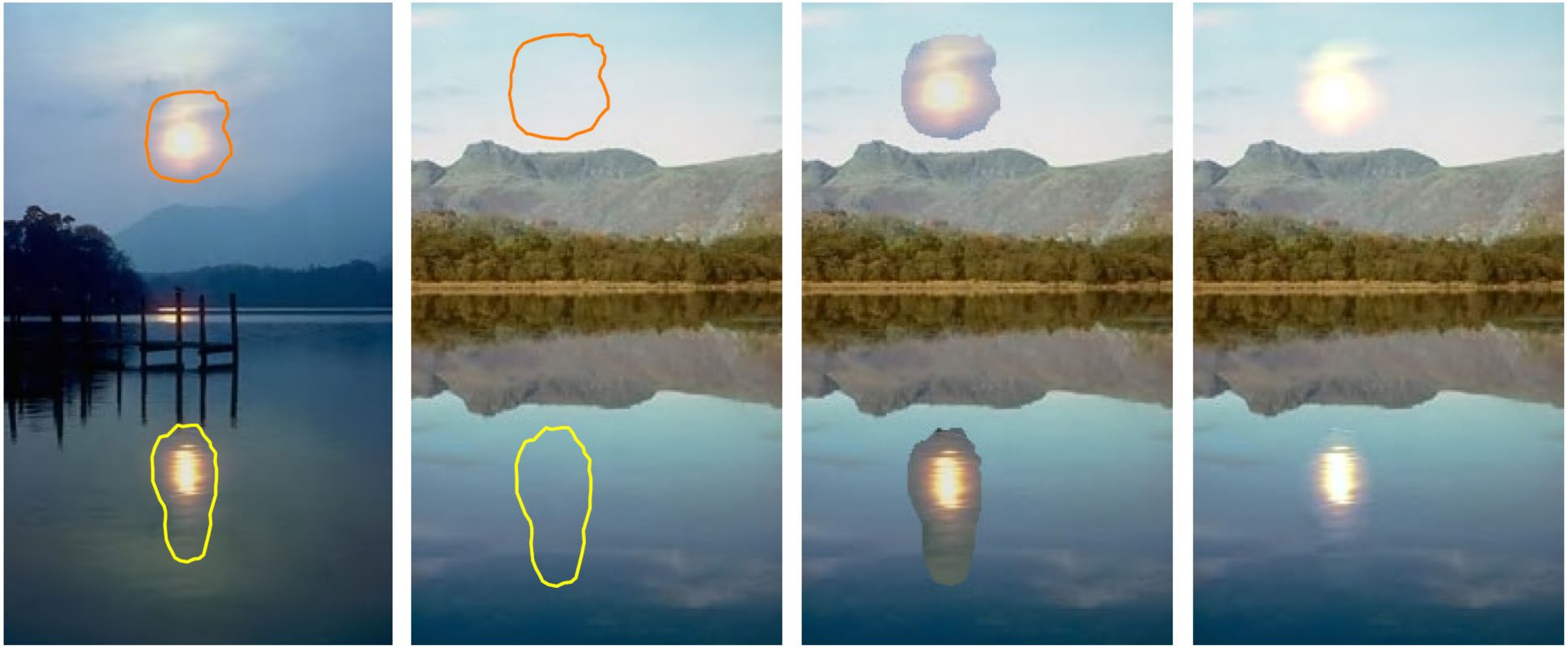
Poisson Image Blending

- Follow gradients of source subject to boundary conditions imposed by dest



$$\begin{cases} \nabla^2 f = \nabla \cdot \mathbf{v} \\ f|_{\partial\Omega} = f^*|_{\partial\Omega} \end{cases}$$

Poisson Image Blending



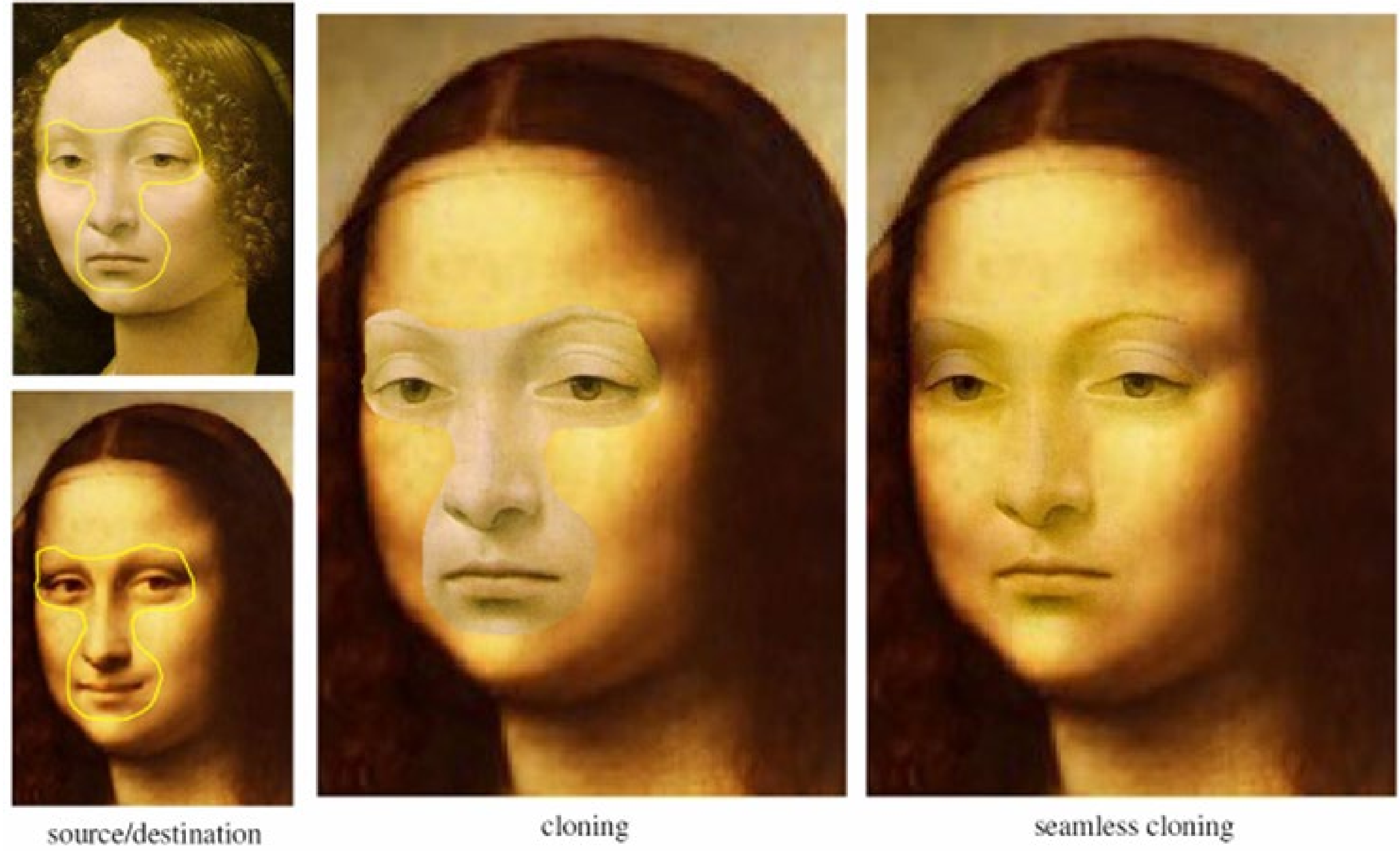
sources

destinations

cloning

seamless cloning

Poisson Image Blending



source/destination

cloning

seamless cloning

Recap: Feature-Based Alignment

- Find keypoints; compute SIFT descriptors
- Generate candidate keypoint matches
- Use RANSAC to select a subset of matches
- Fit to find best image transformation
- Warp images according to transformation
- Blend images in overlapping regions

Real-World Panoramic Stitching

- How to handle more than 2 frames?
  - Align each frame to the previous: simple, but can lead to drift in alignment
  - Optimize for all transformations at once: “bundle adjustment”

Real-World Panoramic Stitching

- How to handle extremely wide total field of view?
  - Project onto cylinder – allows 360° viewing