SAIR
Spatial AI & Robotics Lab

CSE 473/573-A

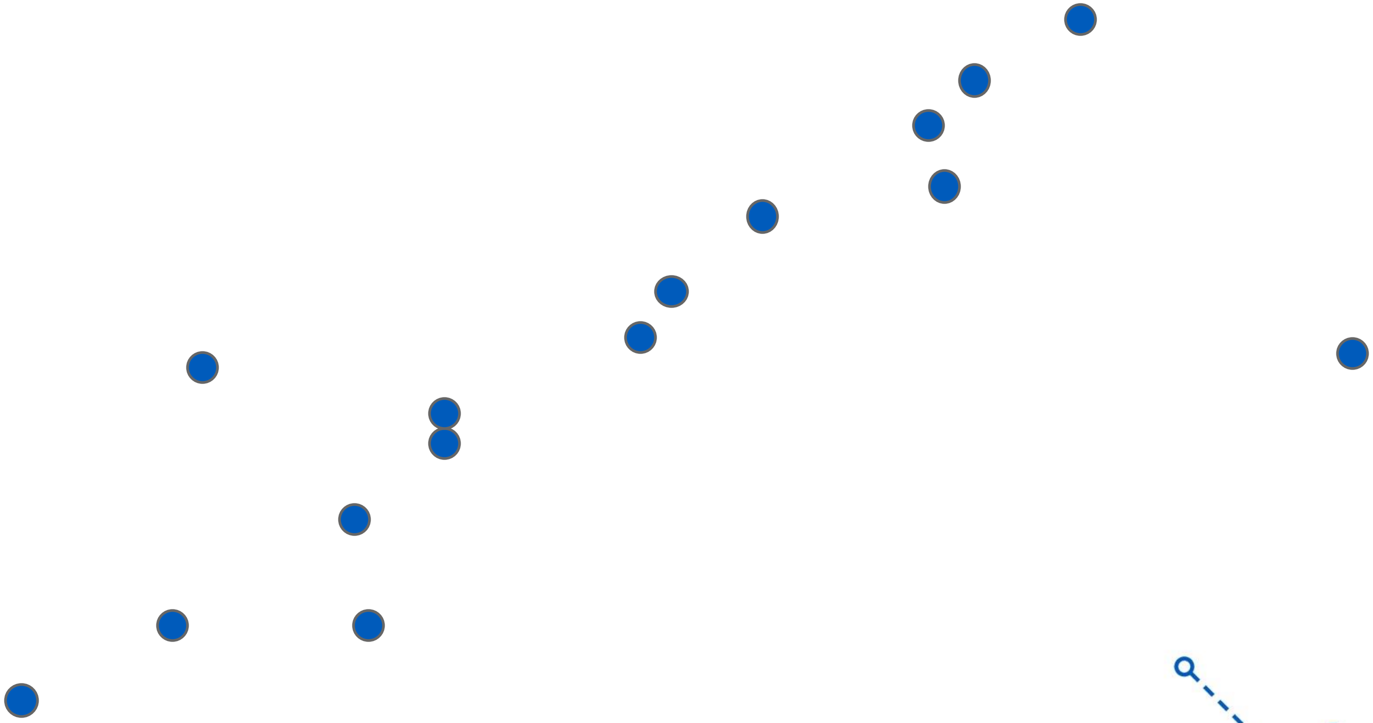
L12: RANSAC & STITCHING

Chen Wang
Spatial AI & Robotics Lab
Department of Computer Science and Engineering

University at Buffalo The State University of New York

Problem: Line Fitting

- Not all data may be representative
- There may be “outliers”
- If you use them, your result may not be accurate



Jinxiang Chai, CMU

Solutions?

- Least Squares Fit?
  - Closed for solution…
  - Sensitive to outliers

- Hypothesize and Test
  - Try out as many lines as we want
  - Keep the best lines
  - But which are the best?

RANSAC

- **RAN**dom **S**ample **C**onsensus
  - An iterative method for estimating a mathematical model from a data set that contains outliers.

- Motivation: we want to avoid the impact of outliers, so let’s look for “inliers”, and use those only.

- Idea: if an outlier is chosen to compute the current model, then the model won’t have much support from rest of the points.

RANSAC

RANSAC loop:

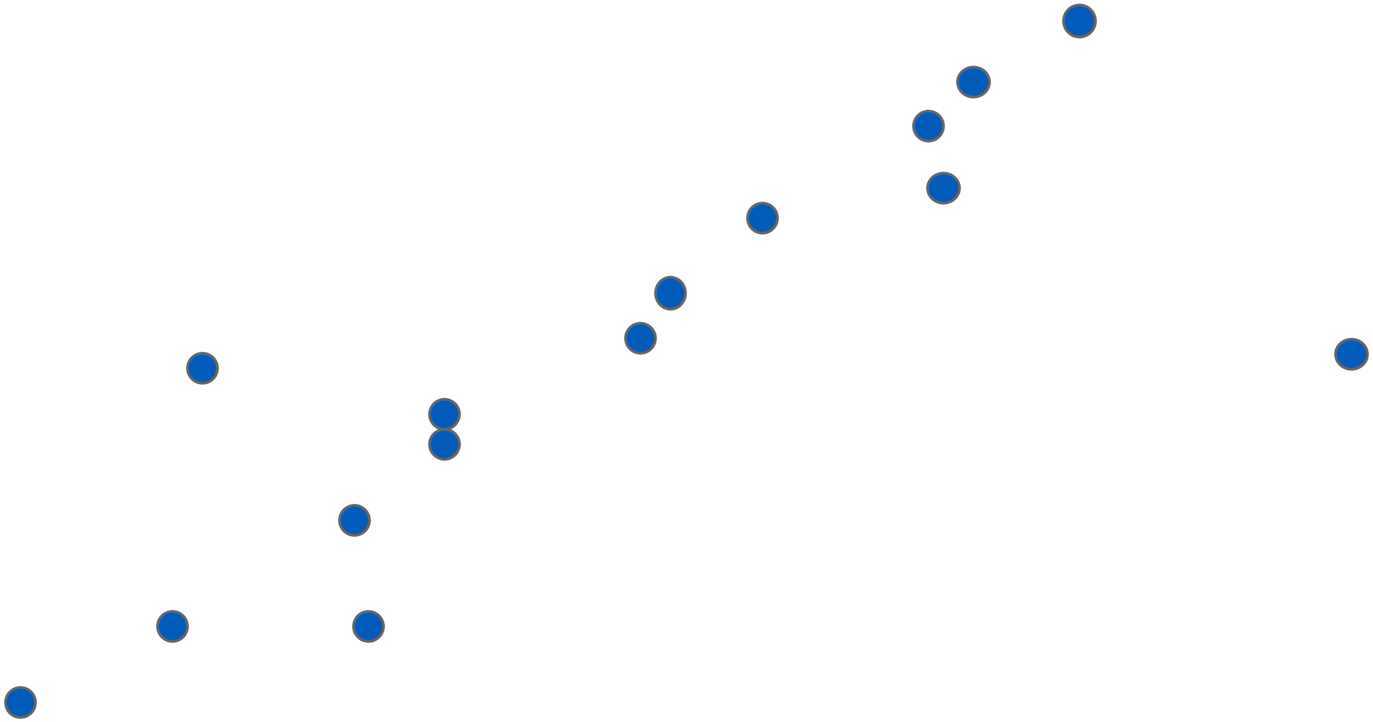
1. Randomly select a **seed group** of points on which to base transformation estimate (e.g., a group of matches)
2. Compute transformation (**model**) from seed group
3. Find **inliers** to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate on all inliers.

- Keep the model with the **largest number of inliers.**

RANSAC Line Fitting Example

Task:

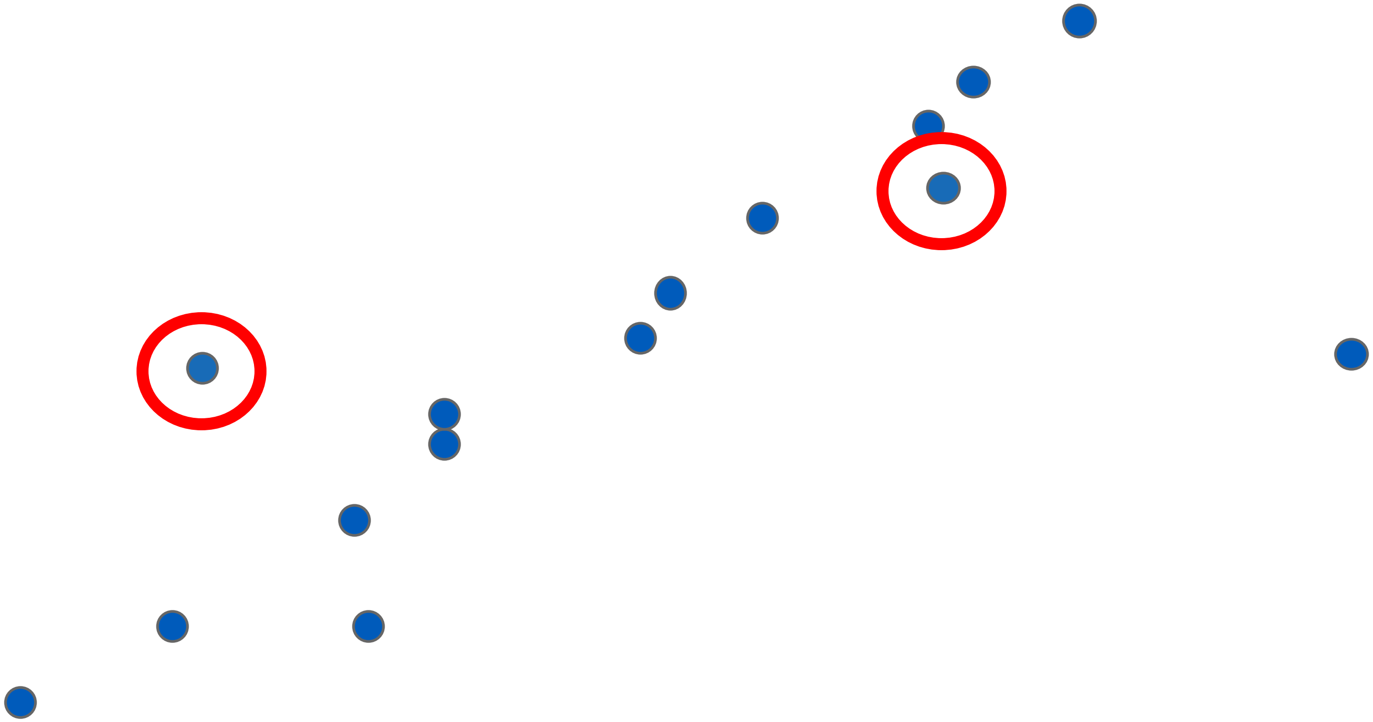
Estimate best line



Slide credit: Jinxiang Chai, CMU

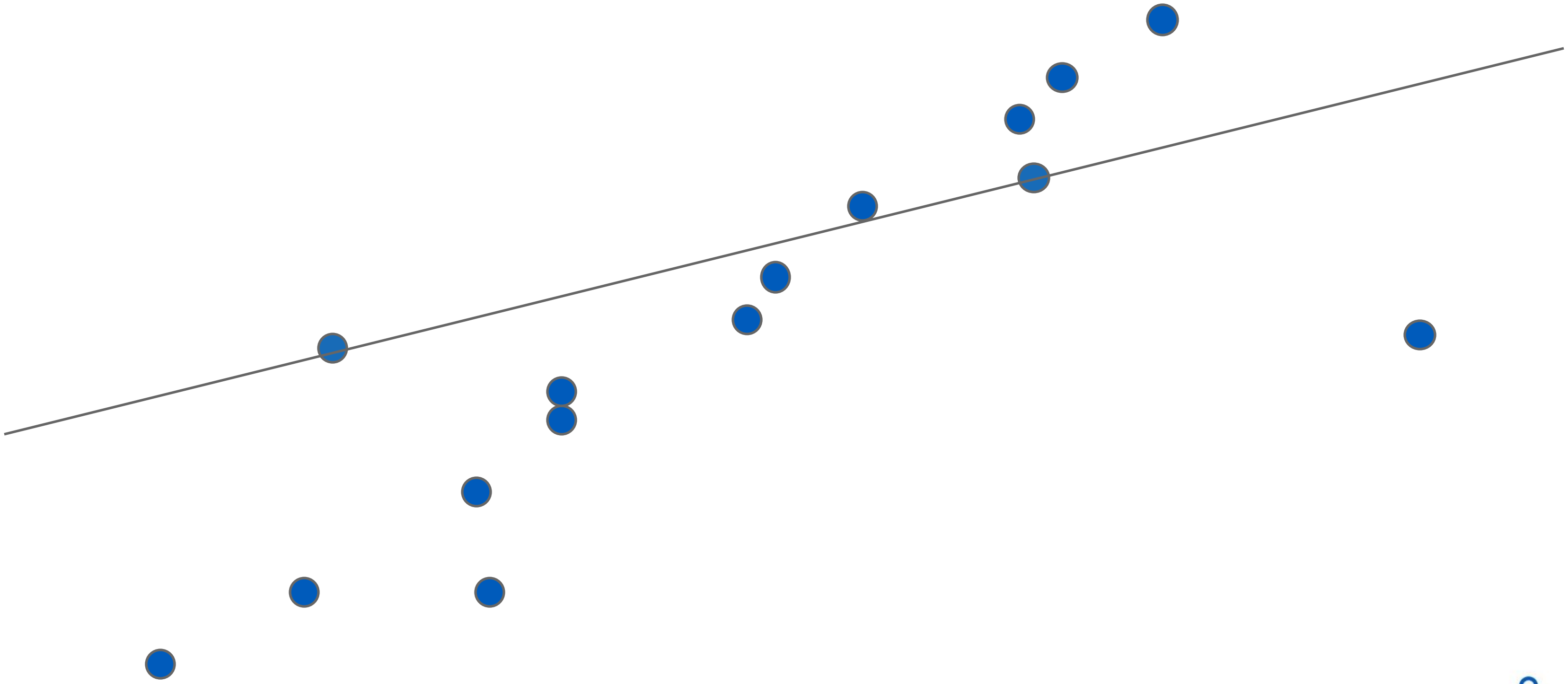
RANSAC Line Fitting Example

Sample two points



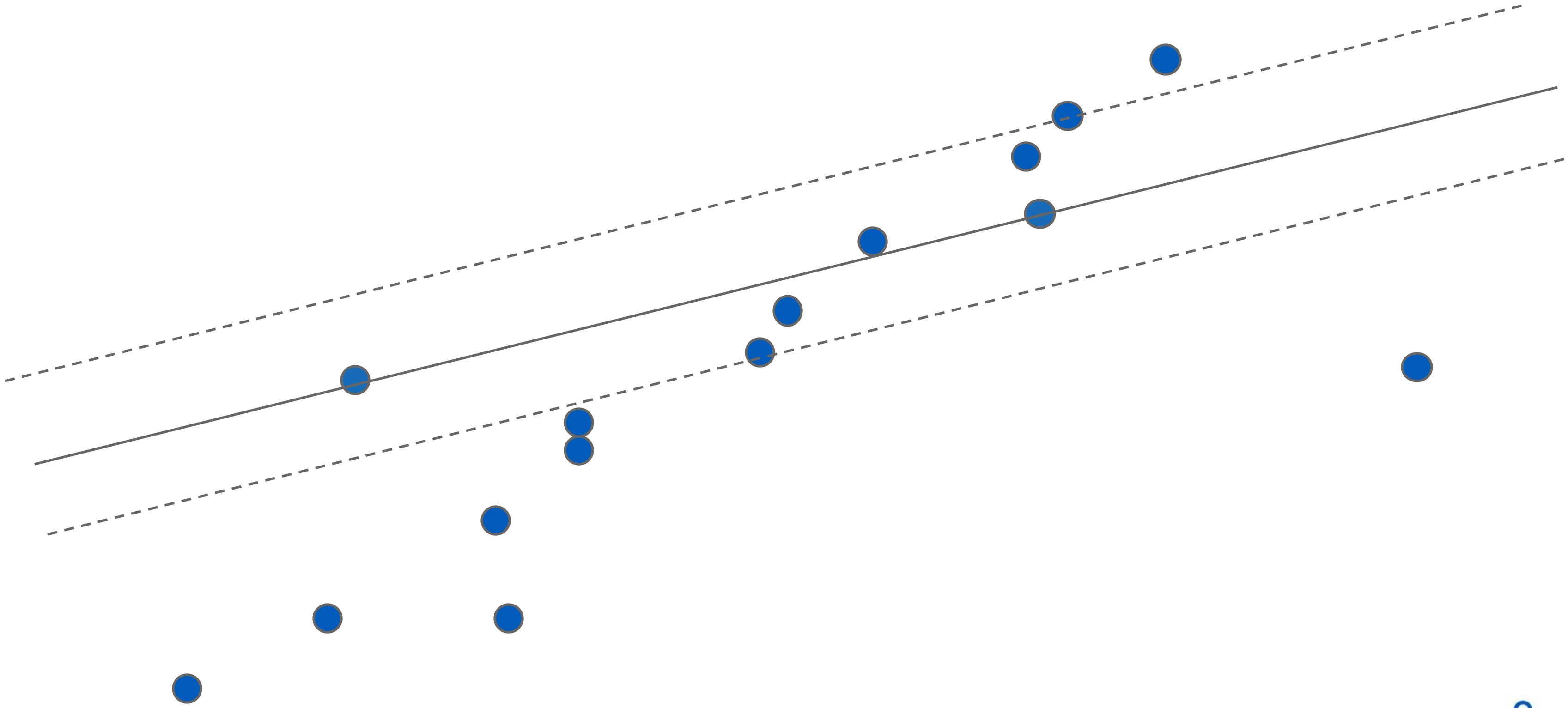
RANSAC Line Fitting Example

Fit Line



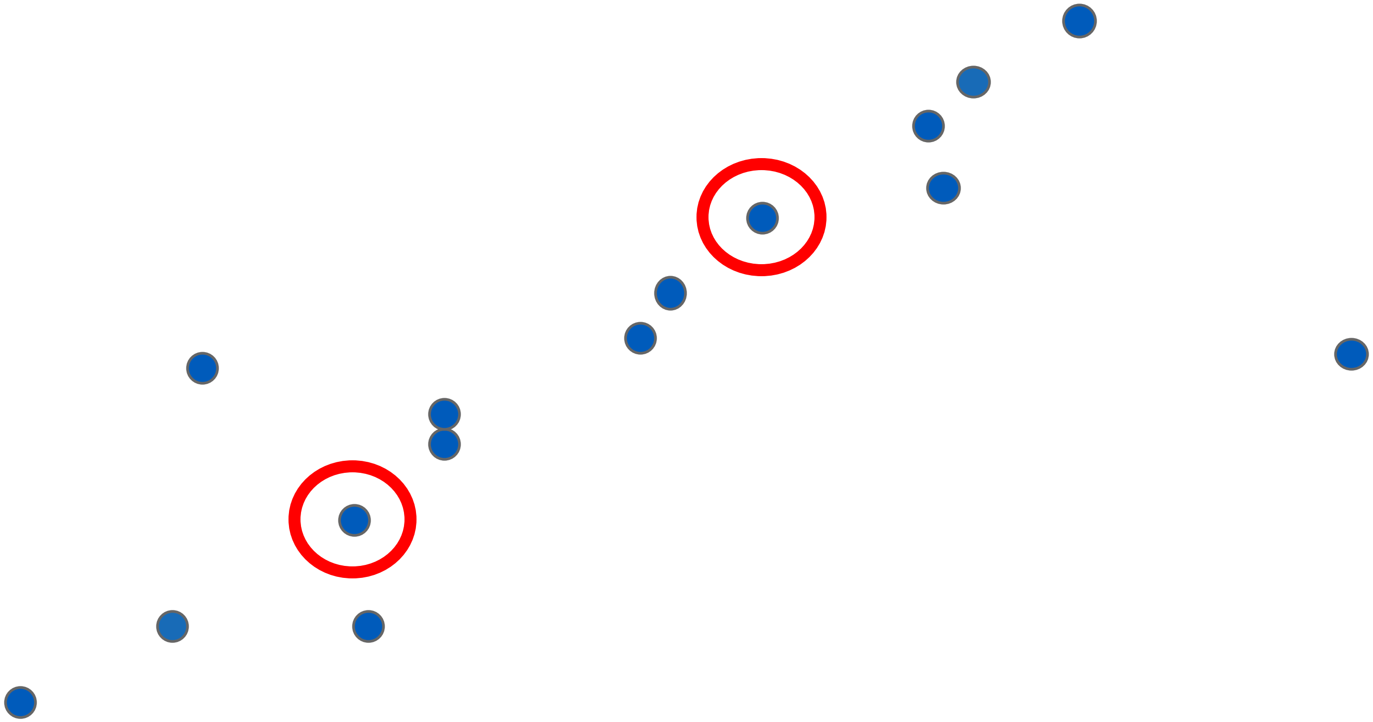
RANSAC Line Fitting Example

Total number of points within a threshold of line.



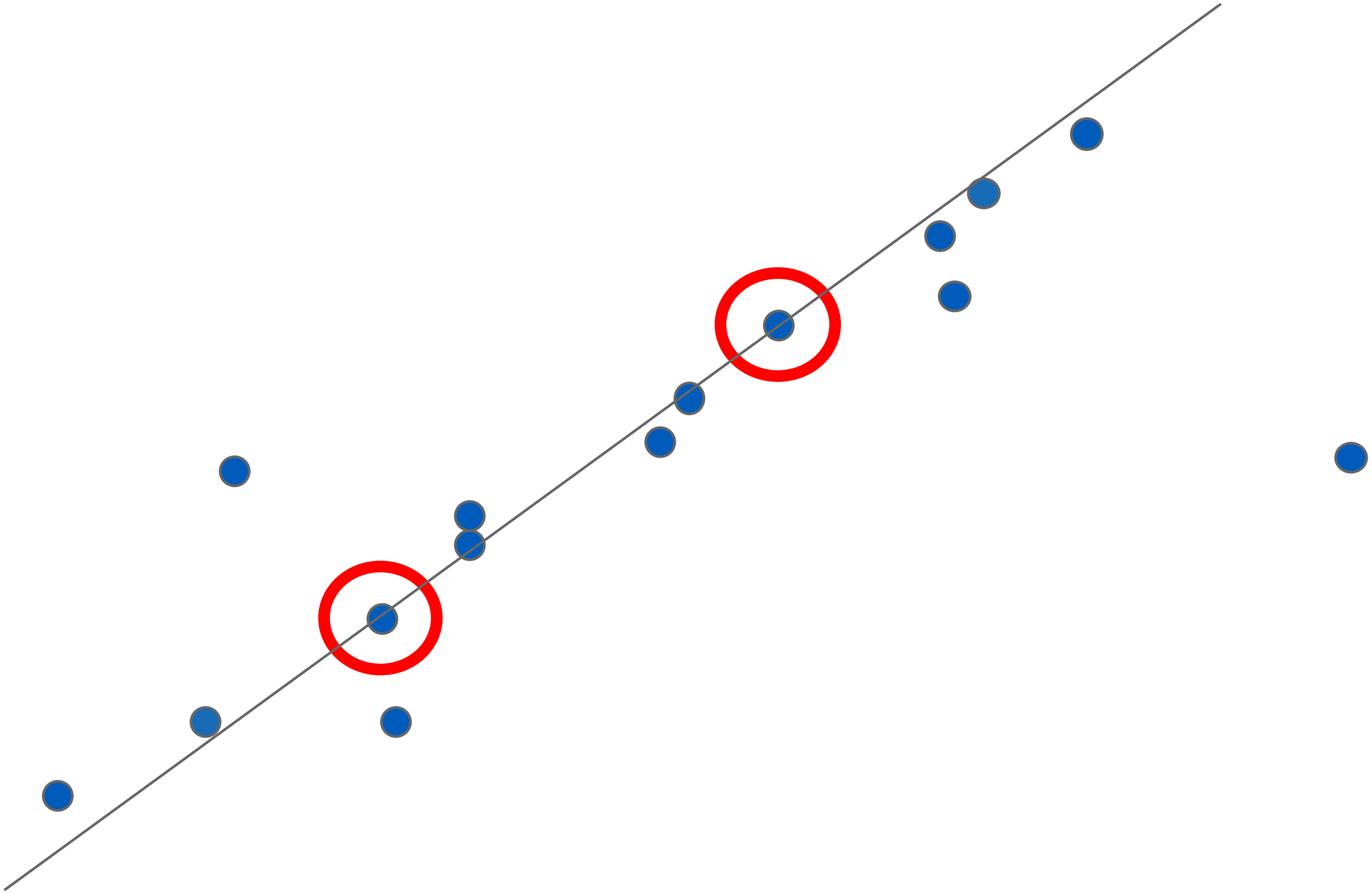
RANSAC Line Fitting Example

Repeat, until get a good result



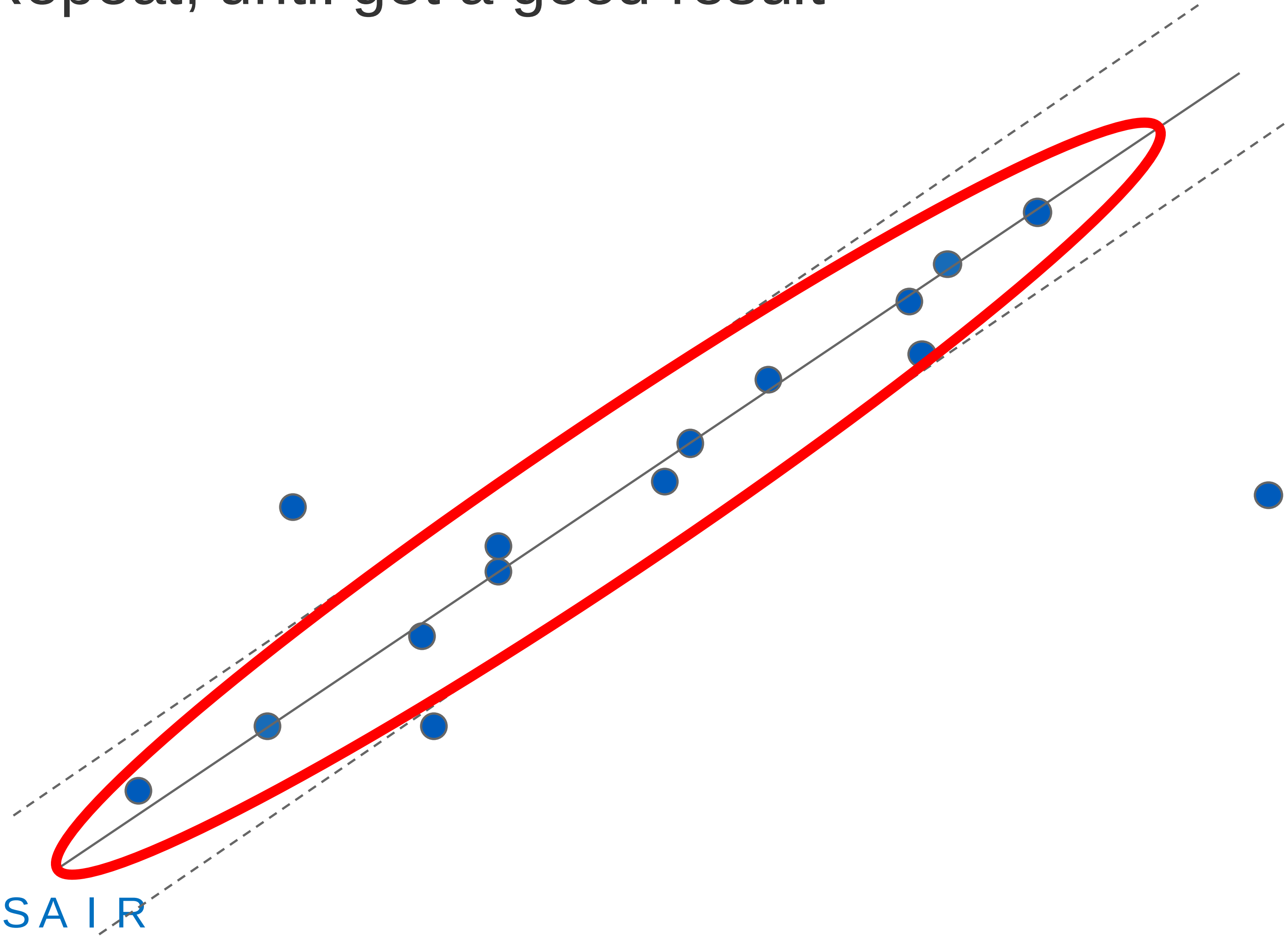
RANSAC Line Fitting Example

Repeat, until get a good result



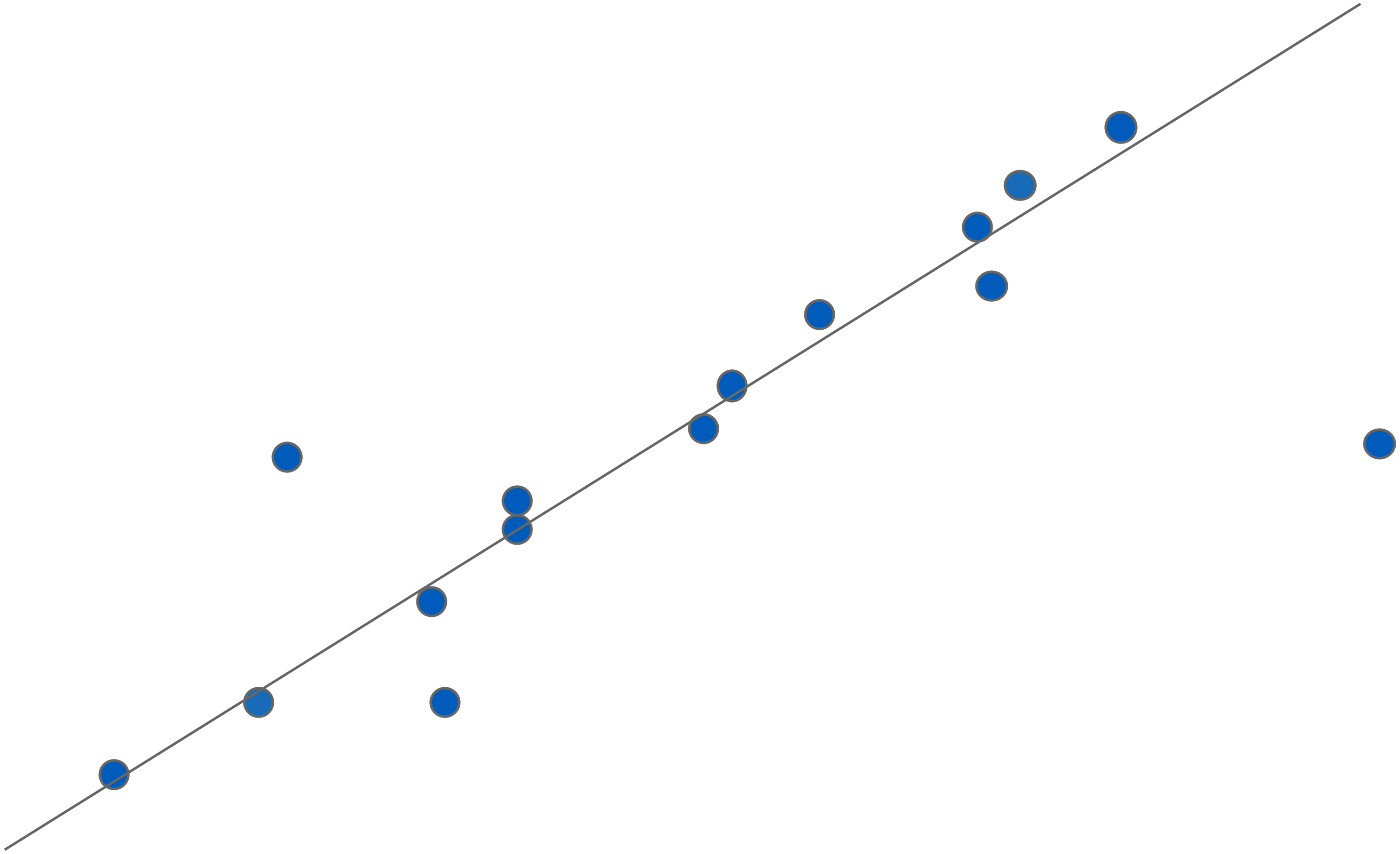
RANSAC Line Fitting Example

Repeat, until get a good result



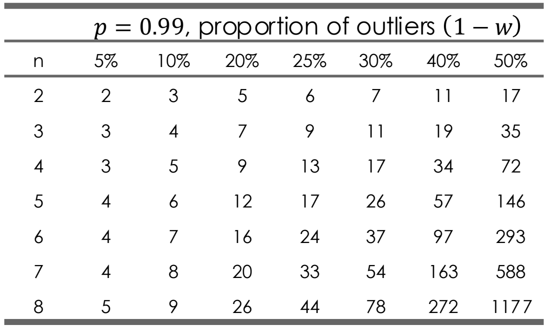
RANSAC Line Fitting Example

Repeat, until get a good result



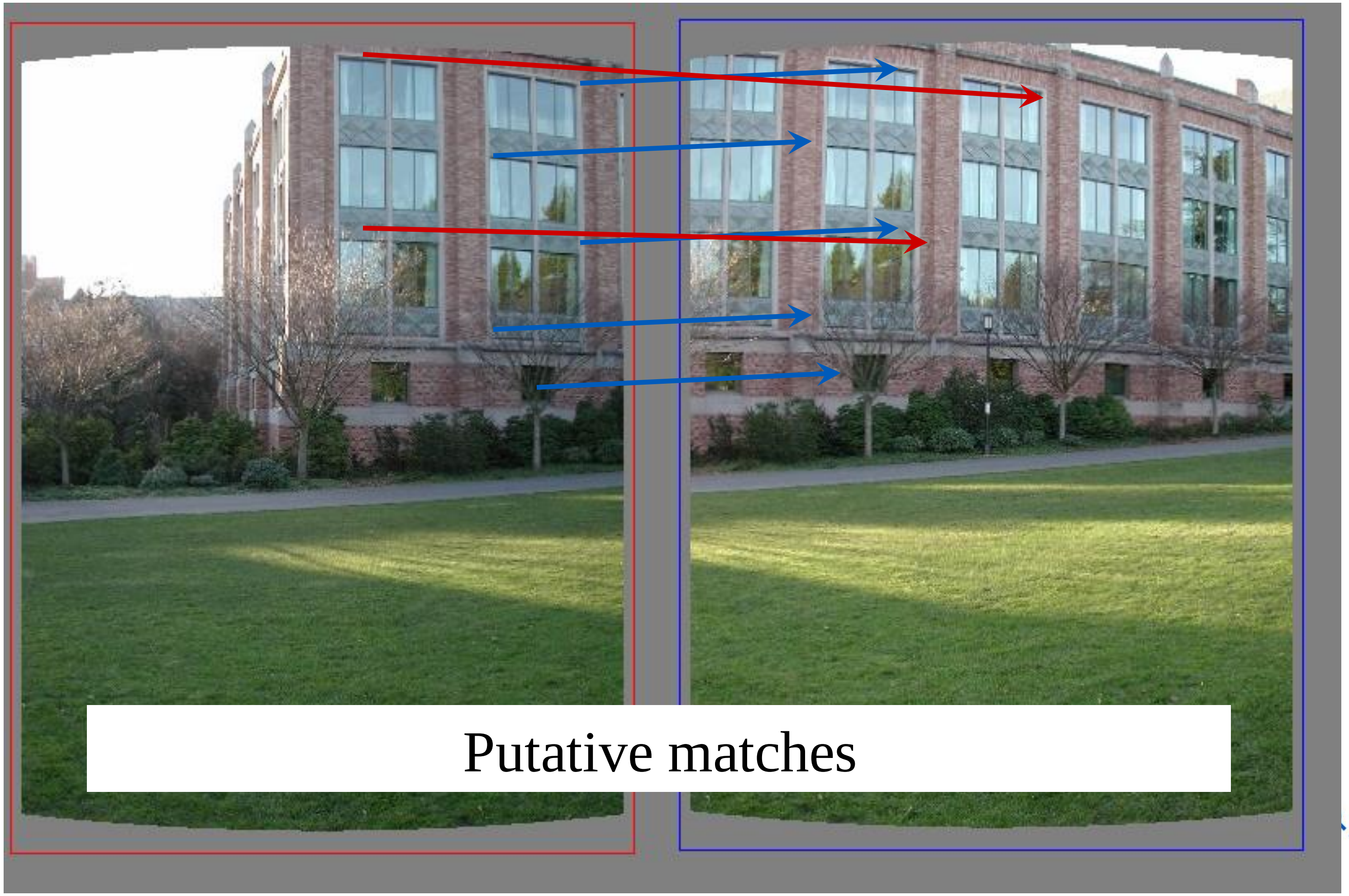
How to choose parameters?

- Number of sampled points n: minimum points to fit a model.
- Inlier threshold δ.
  - Choose δ so that a good point with noise is likely within threshold.
- To determine the number of iterations K.
  - Desired probability of success (p): at least one useful result.
  - Let w be the probability of choosing an inlier when selecting a point.
    - w = number of inliers in data / number of points in data
  - n points selected independently for estimating a model.
  - w^n: the probability that all n points are inliers.
  - $1-w^n$: probability of at least one of the n points is an outlier.
  - $(1-w^n)^K$: after k iterations, never select a set of n inlier points.
  - $1-p=(1-w^n)^K$
  - $K=\frac{\log(1-p)}{\log(1-w^n)}$



	$p = 0.99$, proportion of outliers $(1-w)$						
n	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

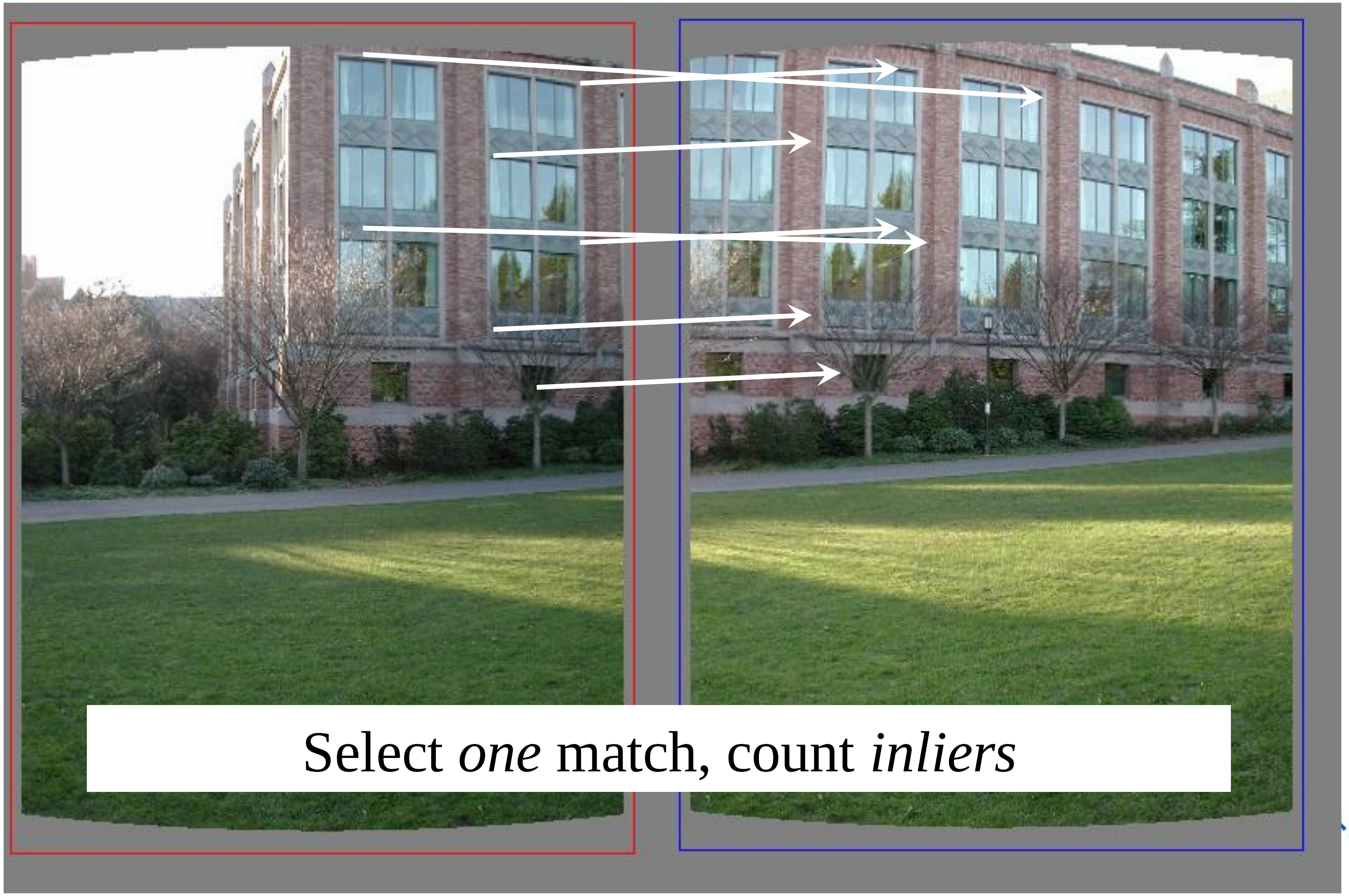
RANSAC example: Translation



Putative matches

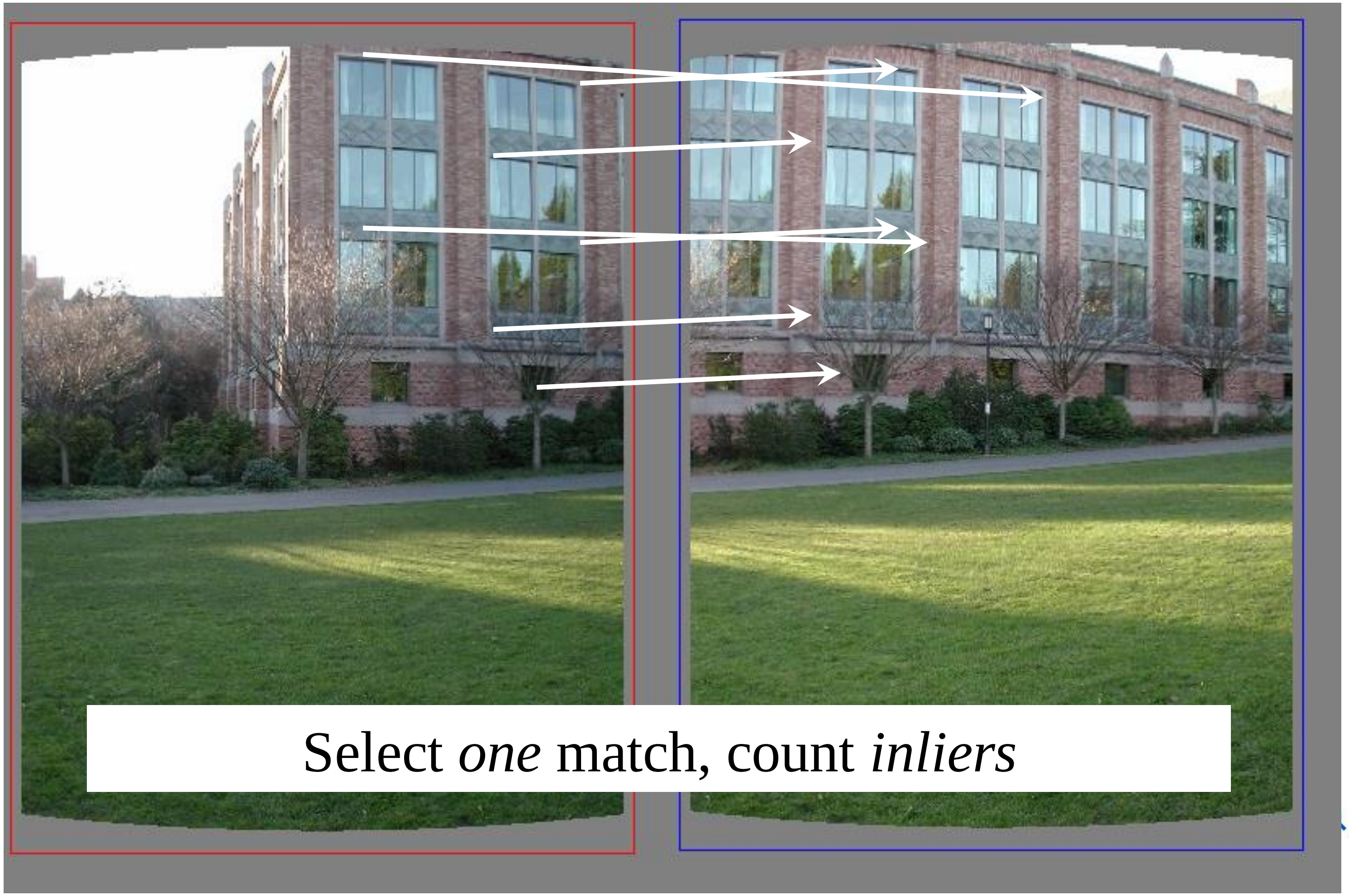
Source: Rick Szeliski

RANSAC example: Translation



Select *one* match, count *inliers*

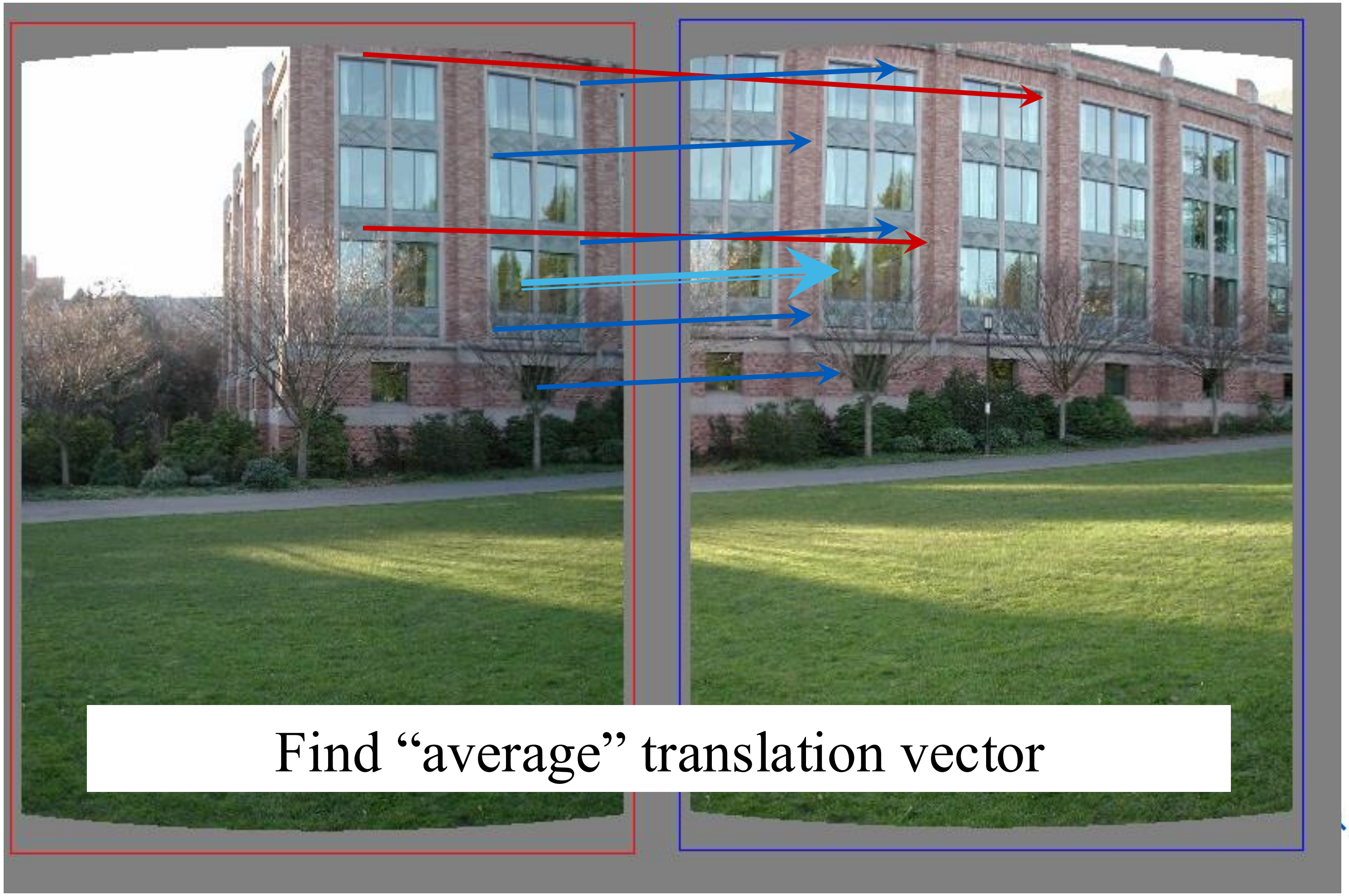
RANSAC example: Translation



Select *one* match, count *inliers*

SAIR
Spatial AI & Robotics Lab

RANSAC example: Translation



Find “average” translation vector

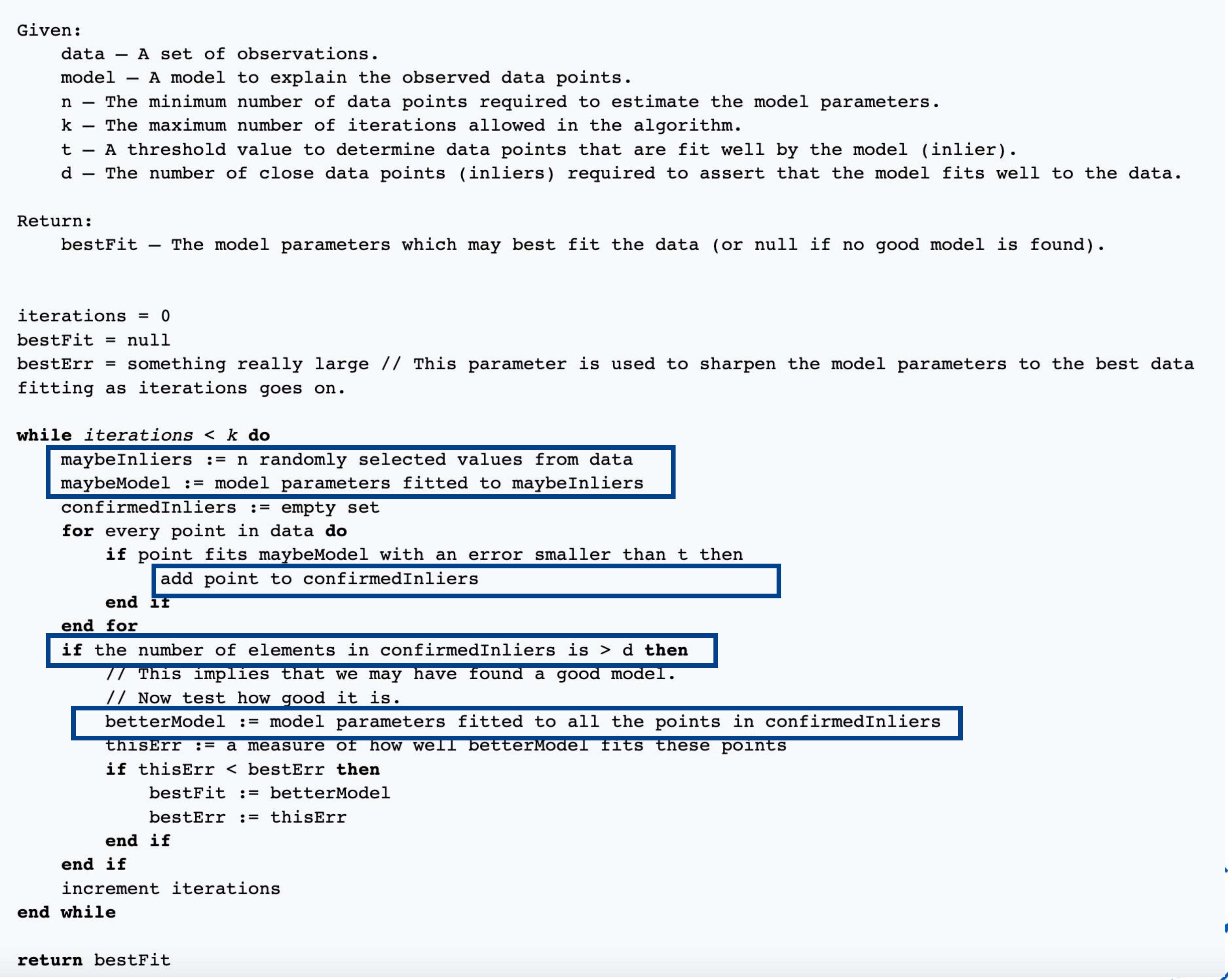
Summary

- Choose:
  - Inlier threshold
    - Related to the amount of noise we expect
  - Number of rounds
    - Related to the percentage of outliners we expect
    - Related to the probability of success we are hoping for.

RANSAC ALGORITHM

- Input:
  - Data: a set of observed data points
  - Model: a model to fit the data
  - Threshold: a threshold to determine inliers
- Output: BestModel: the model with the most inliers
- Repeat for a fixed number of iterations:
  - 1. Select a random subset of data
  - 2. Fit the model to the data points in the subset
  - 3. Determine the inliers by comparing the fitted model to data
  - 4. If the number of inliers exceeds the threshold
    - re-estimate the model using all the inliers
  - 5. Store the model if it has the most inliers seen so far
- Return BestModel

RANSAC Algorithm



```
Given:
    data – A set of observations.
    model – A model to explain the observed data points.
    n – The minimum number of data points required to estimate the model parameters.
    k – The maximum number of iterations allowed in the algorithm.
    t – A threshold value to determine data points that are fit well by the model (inlier).
    d – The number of close data points (inliers) required to assert that the model fits well to the data.

Return:
    bestFit – The model parameters which may best fit the data (or null if no good model is found).


iterations = 0
bestFit = null
bestErr = something really large // This parameter is used to sharpen the model parameters to the best data
fitting as iterations goes on.

while iterations < k do
    maybeInliers := n randomly selected values from data
    maybeModel := model parameters fitted to maybeInliers
    confirmedInliers := empty set
    for every point in data do
        if point fits maybeModel with an error smaller than t then
             add point to confirmedInliers
        end if
    end for
    if the number of elements in confirmedInliers is > d then
        // This implies that we may have found a good model.
        // Now test how good it is.
        betterModel := model parameters fitted to all the points in confirmedInliers
        thisErr := a measure of how well betterModel fits these points
        if thisErr < bestErr then
            bestFit := betterModel
            bestErr := thisErr
        end if
    end if
    increment iterations
end while

return bestFit
```

RANSAC Properties

Good

- Robust to outliers
- Applicable for larger number of model parameters than Hough transform.
- Optimization parameters are easier to choose than Hough transform.
  - Lines with normal form works for Hough, but slope-intercept form not.

Bad

- Computational time grows quickly with outliers and parameters
  - While Hough transform grows quickly with number of parameters.
- Not good for getting multiple fits
  - Hough transform can fit multiple lines simultaneously.

More applications

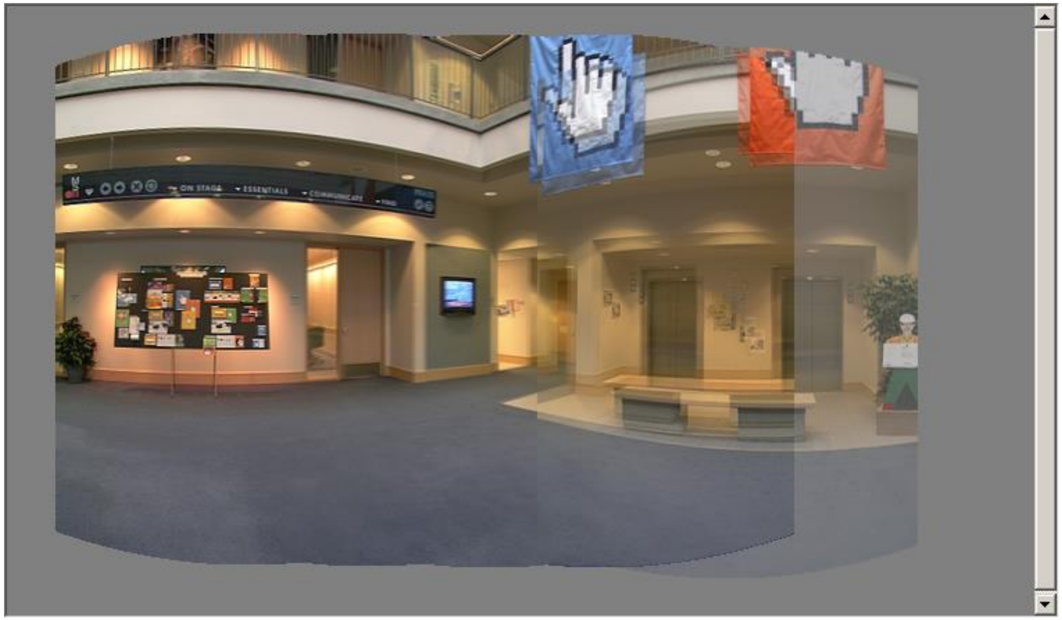
- Computing a homography (e.g., image stitching)
- Estimating fundamental matrix (relating two views)

Content

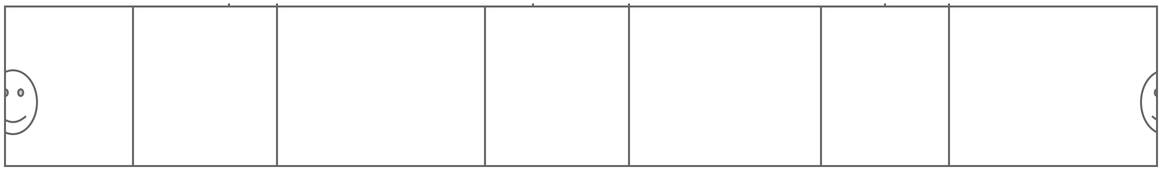
- Stitching (Blending)
  - Panorama, Blending, Deghosting
  - Feathering, Pyramid Blending
  - Laplacian Blending

Image Stitching

1. Align the images over each other
   - camera pan ↔ translation on cylinder
2. Blend the images together



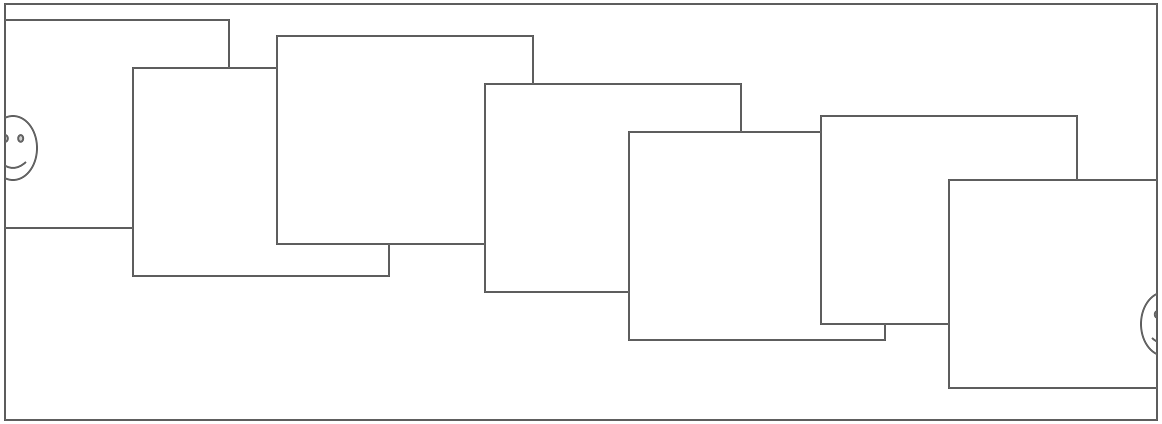
Assembling the panorama



- Stitch pairs together, blend, then crop

Szeliski

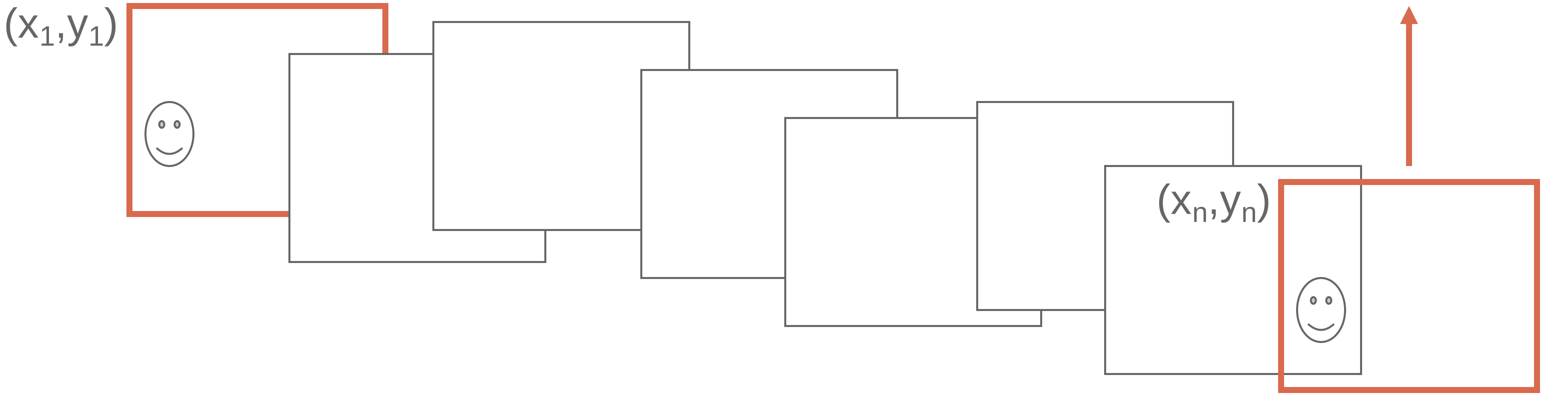
Problem: Drift



- Error accumulation
  - small (vertical) errors accumulate over time
  - apply correction so that sum = 0 (for 360° pan.)

Szeliski

Problem: Drift



copy of first image

- Solution
  - add another copy of first image at the end
  - this gives a constraint: $y_n = y_1$
  - there are a bunch of ways to solve this problem
    - add displacement of $(y_1 – y_n)/(n -1)$ to each image after the first
    - compute a global warp: $y' = y + ax$
    - run a big optimization problem, incorporating this constraint
      - best solution, but more complicated.

Full-view (360° spherical) panoramas

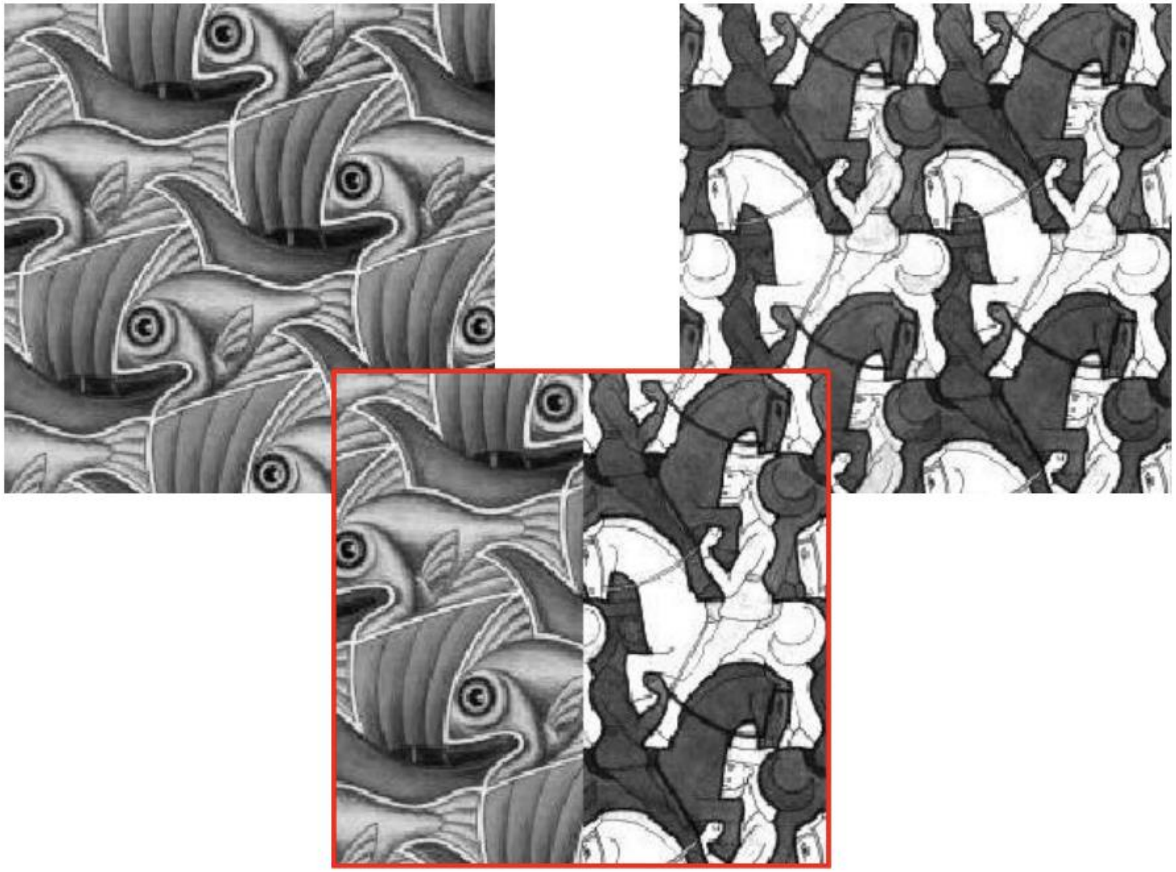
Full-view Panorama

Texture Mapped Model

Szeliski

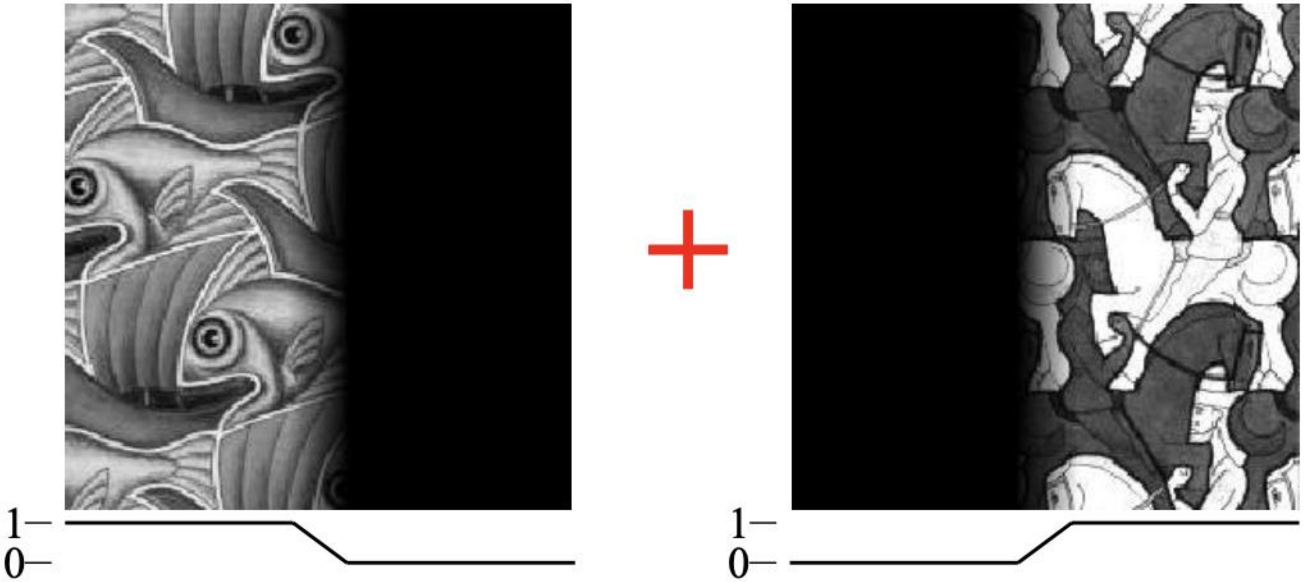
Image Blending

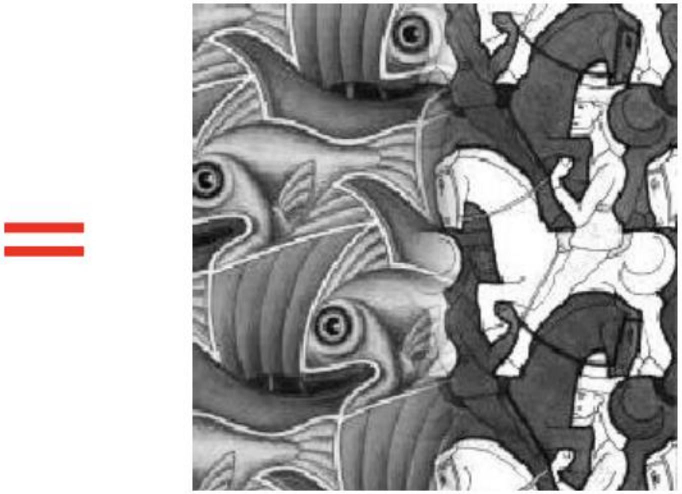
- How do we put images together so there is no obvious divisions?



Szelisk

Alpha Blending / Feathering





$$I_{blend} = \alpha I_{left} + (1-\alpha) I_{right}$$

Setting alpha: simple averaging

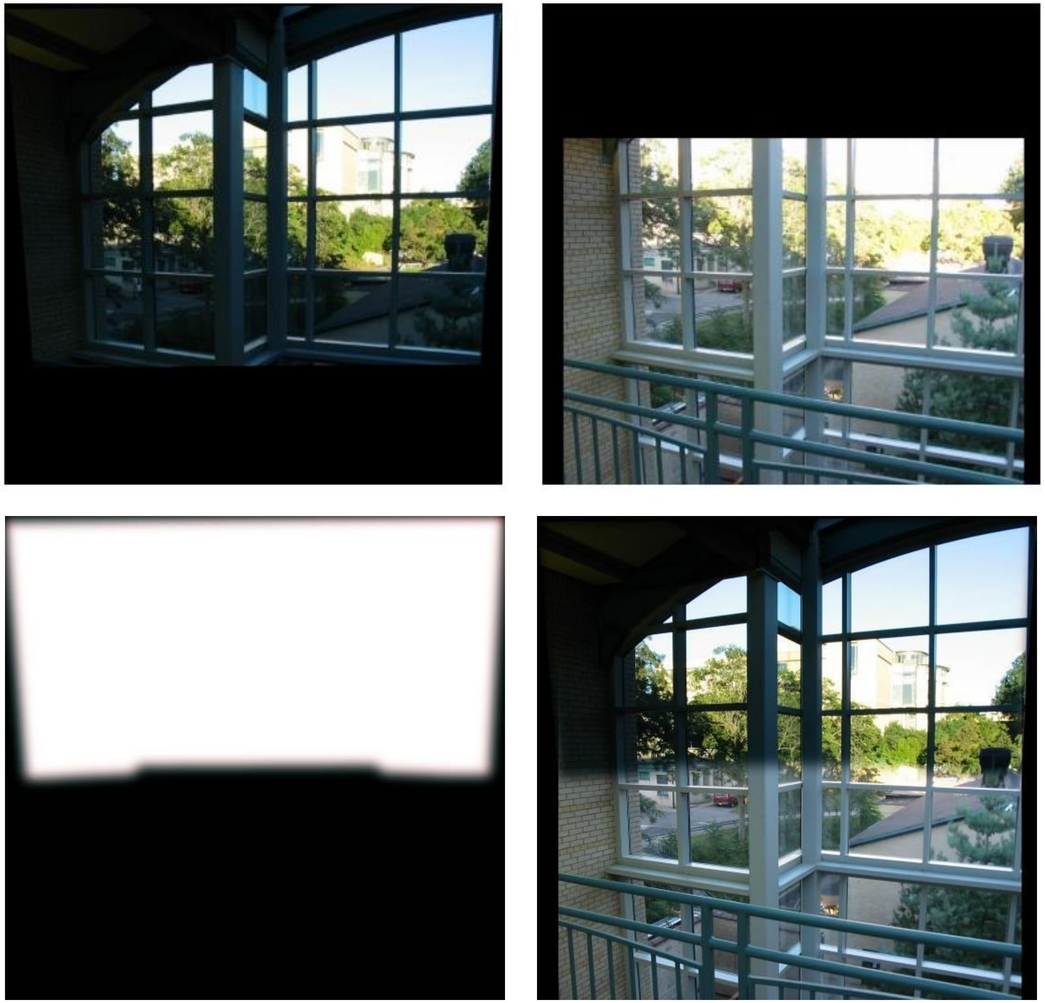
- Alpha = 0.5 in the overlap region

Setting alpha: center seam

- Alpha = logical(dtrans1>dtrans2)

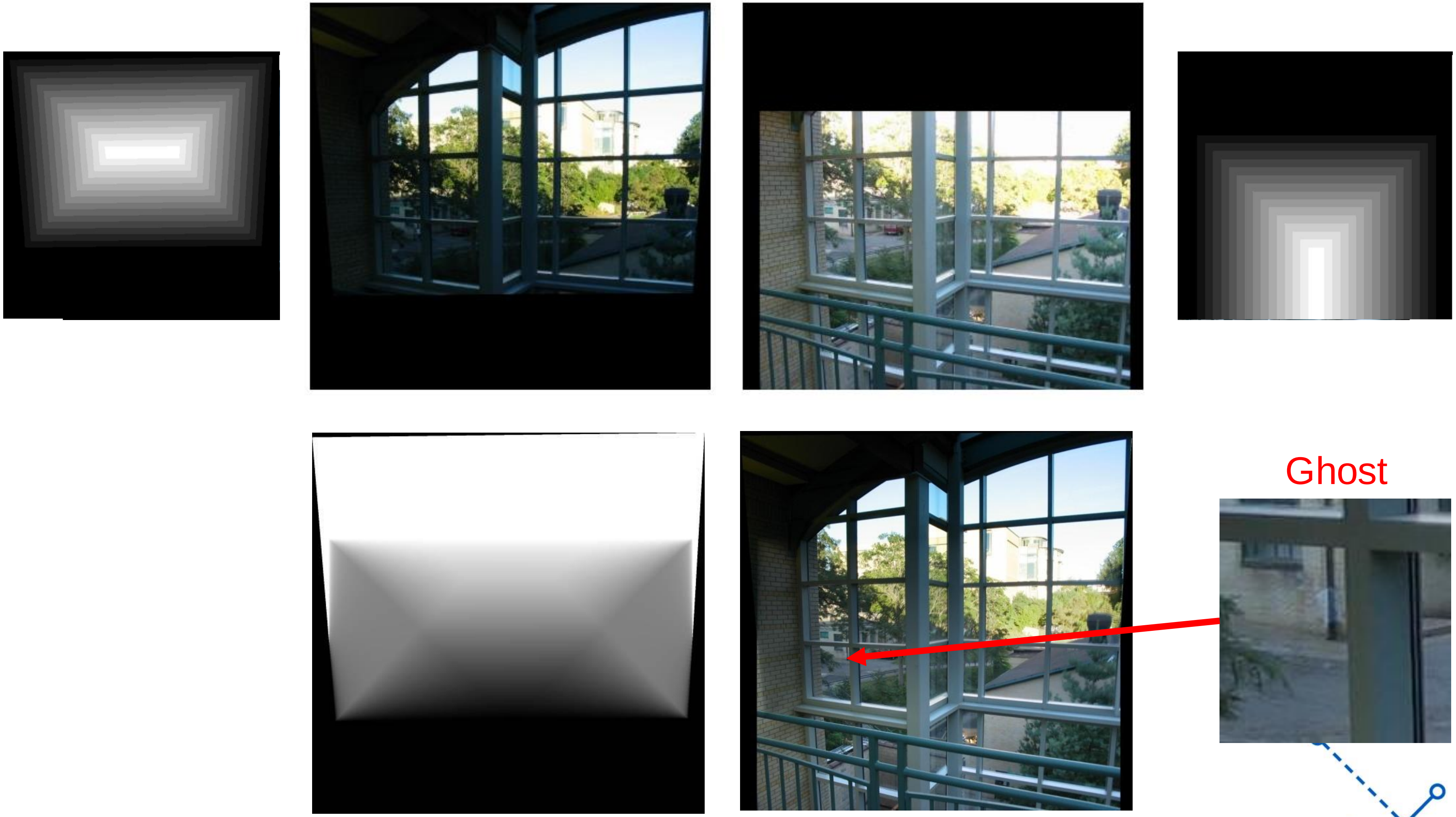
Setting alpha: blurred seam

- Alpha = blurred

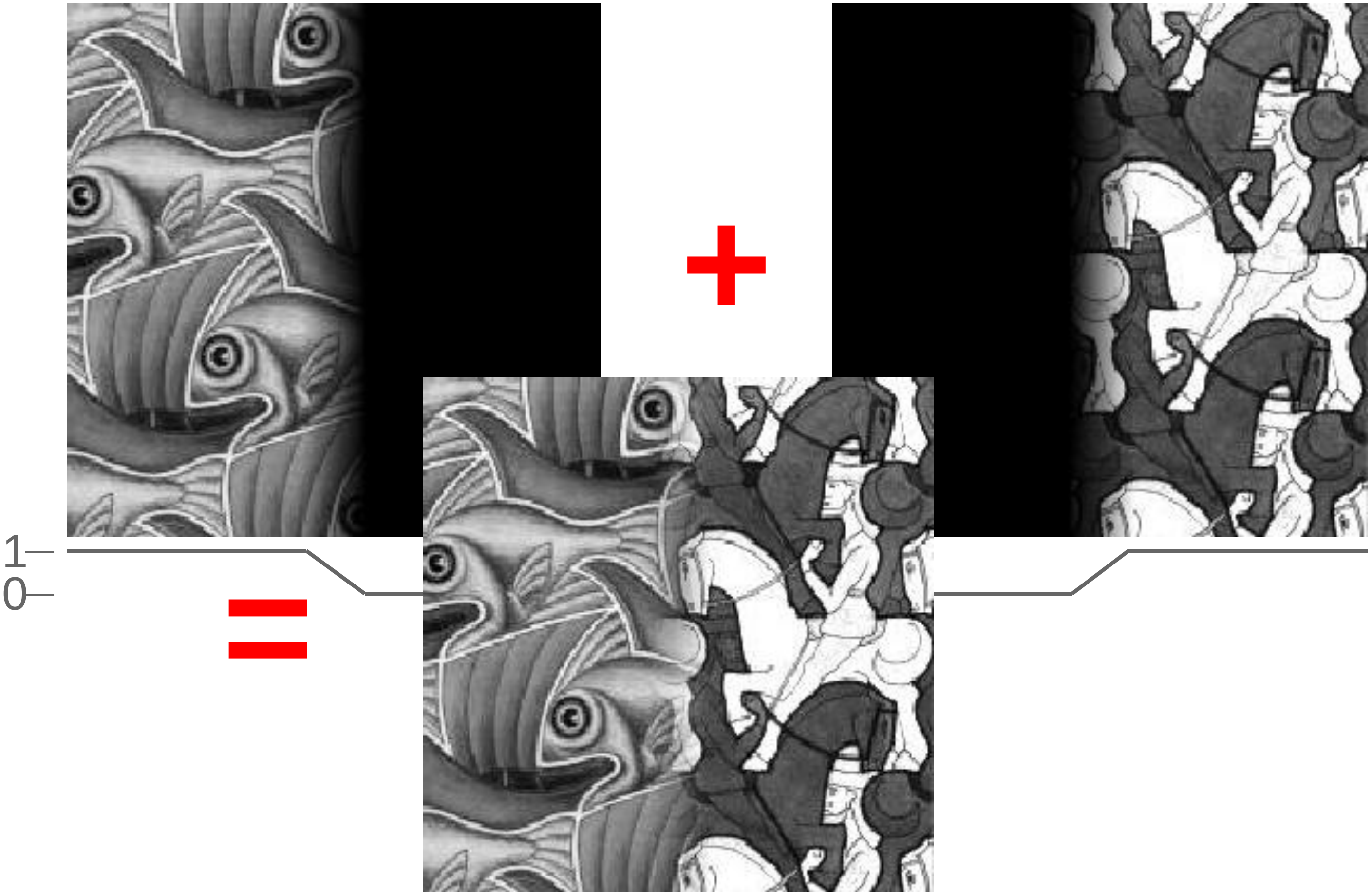


Setting alpha: center weighting

- Alpha = dtrans1 / (dtrans1+dtrans2)

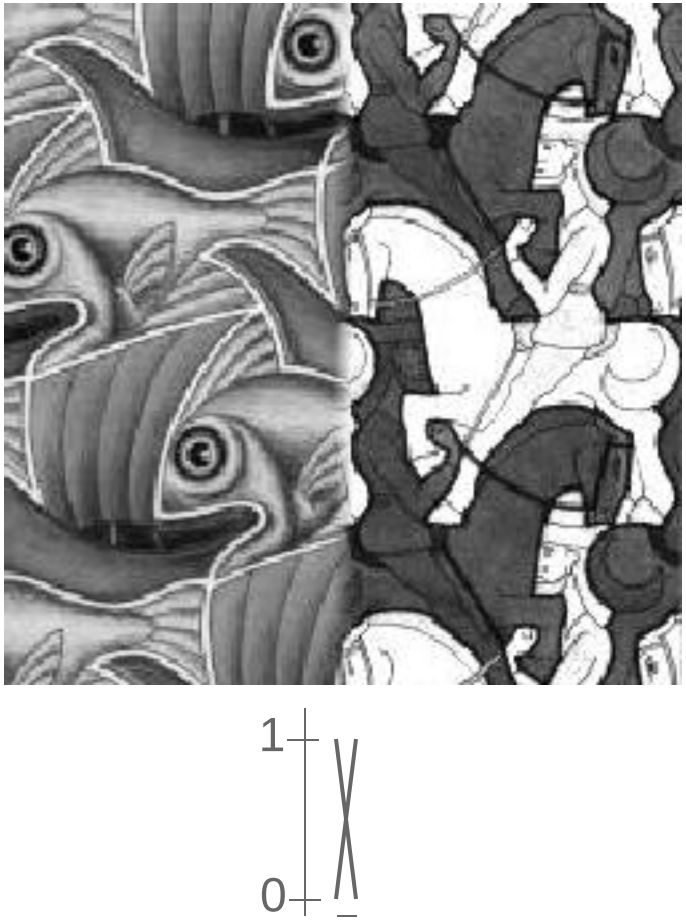


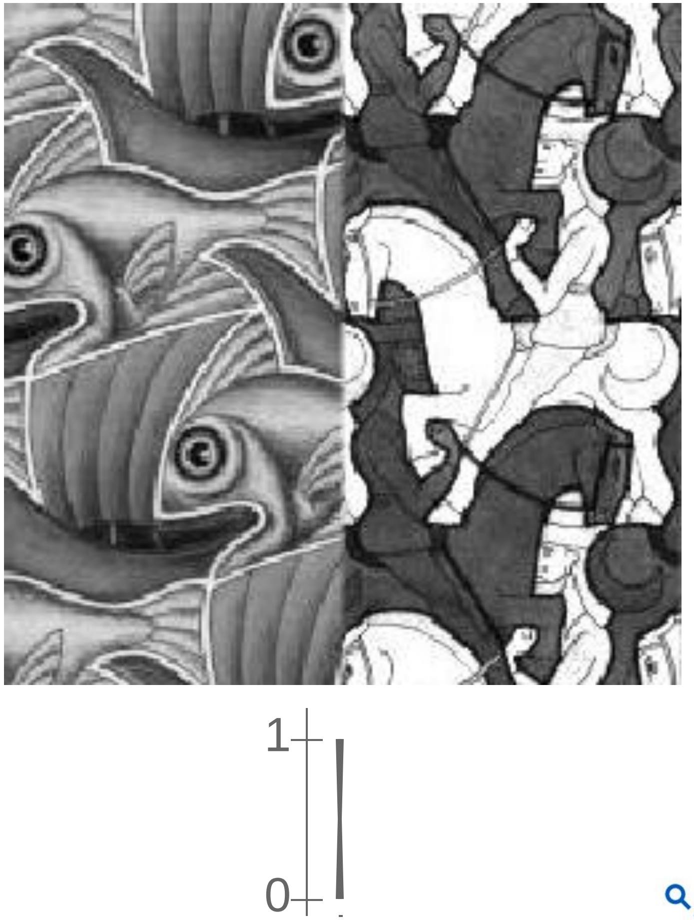
Feathering



Szelisk

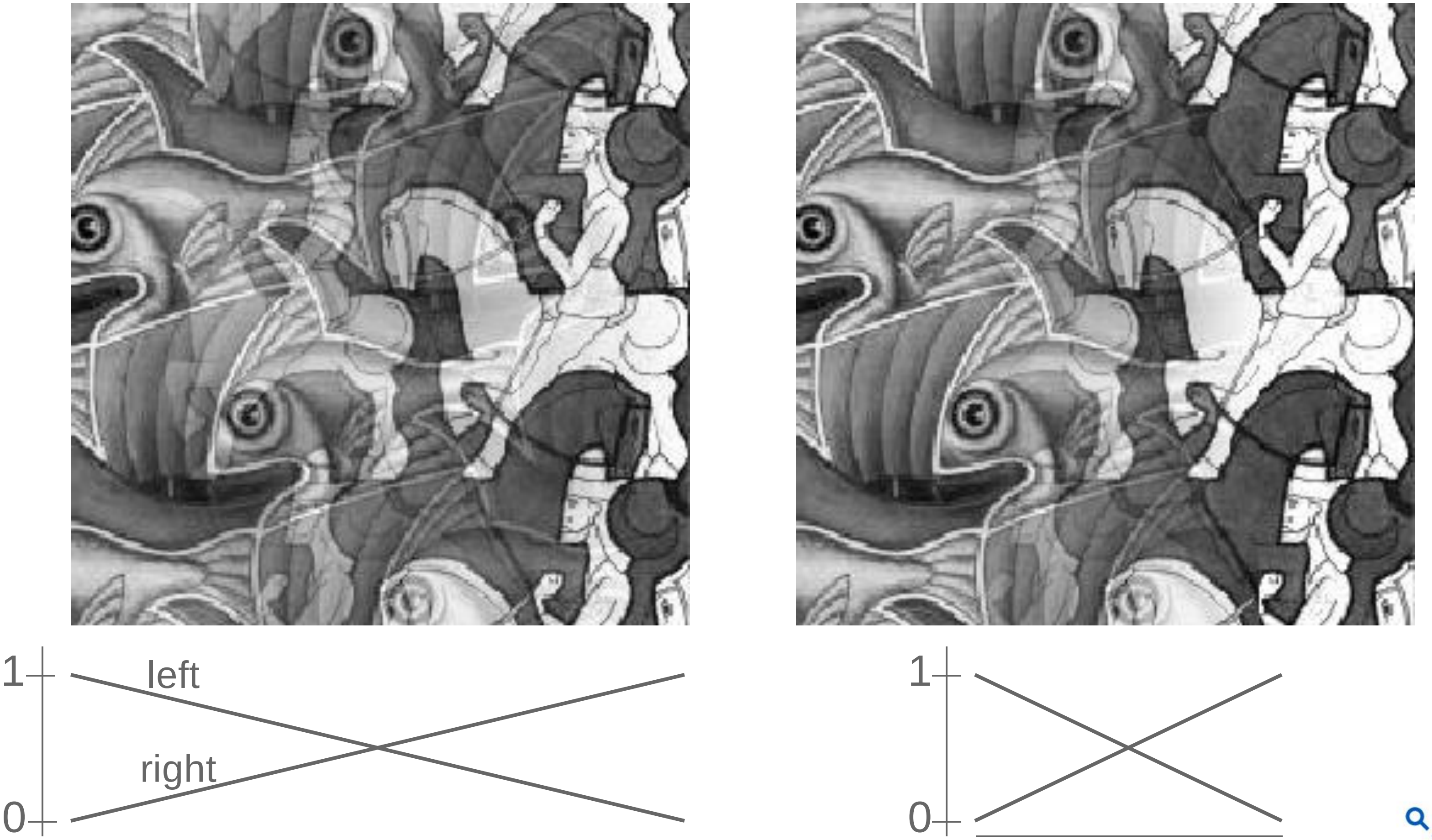
Effect of window size





Szelisk

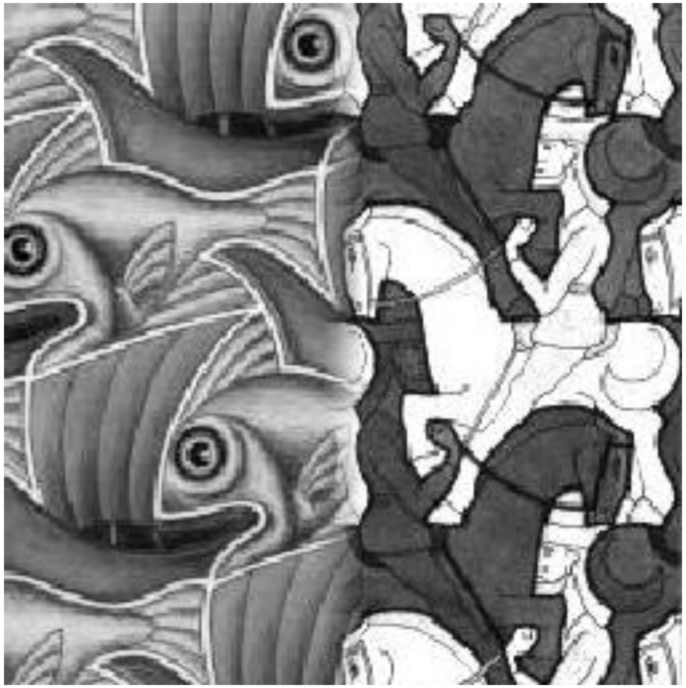
Effect of window size

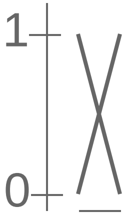


Szeliski

Good Window Size

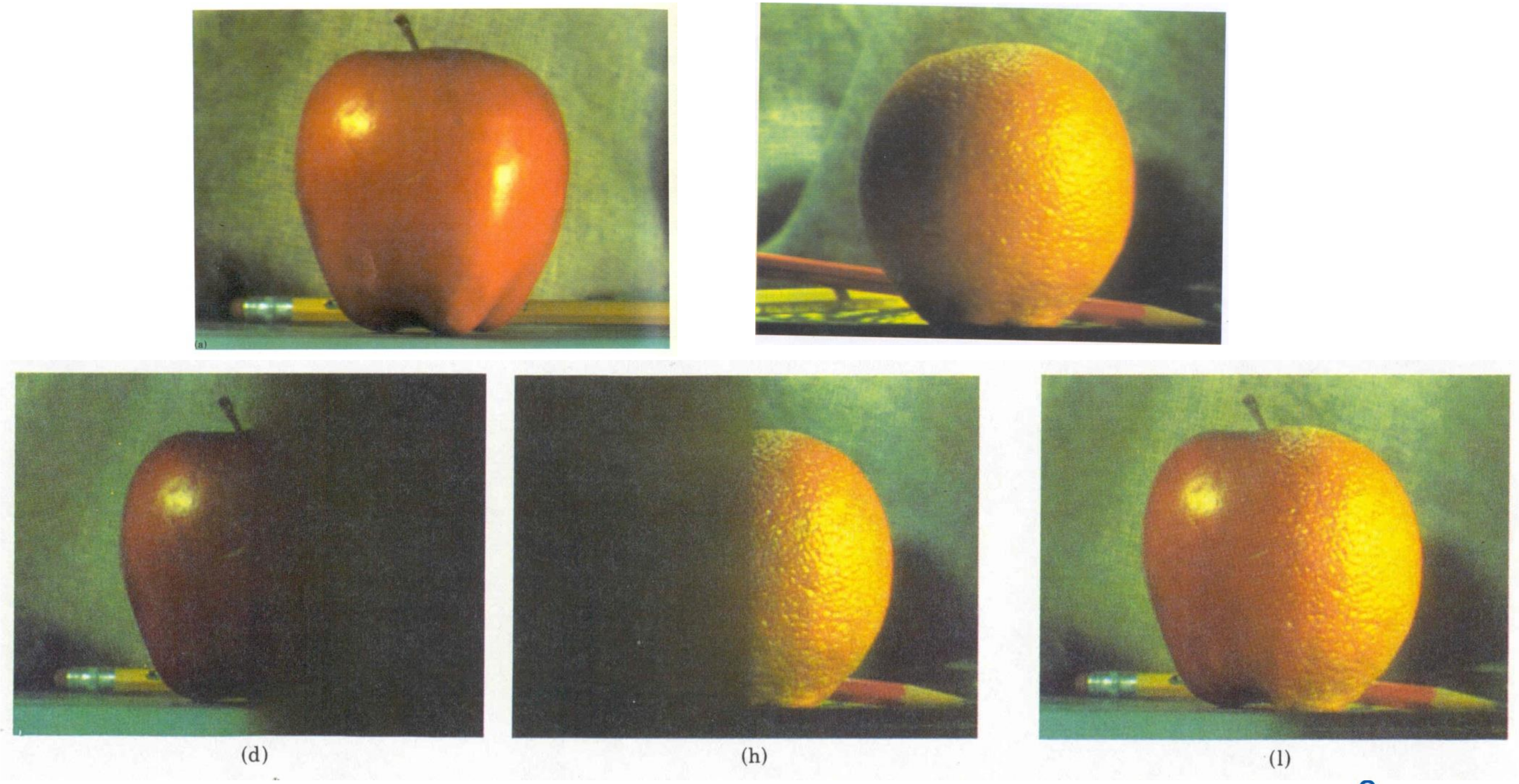
- “Optimal” Window: smooth but not ghosted.
- To avoid seams
  - window = size of largest prominent feature
- To avoid ghosting
  - window <= 2*size of smallest prominent feature





SAIR
Spatial AI & Robotics Lab

Pyramid Blending



Burt, P. J. and Adelson, E. H., [A multiresolution spline with applications to image mosaics](), ACM Transactions on Graphics, 42(4), October 1983, 217-236.

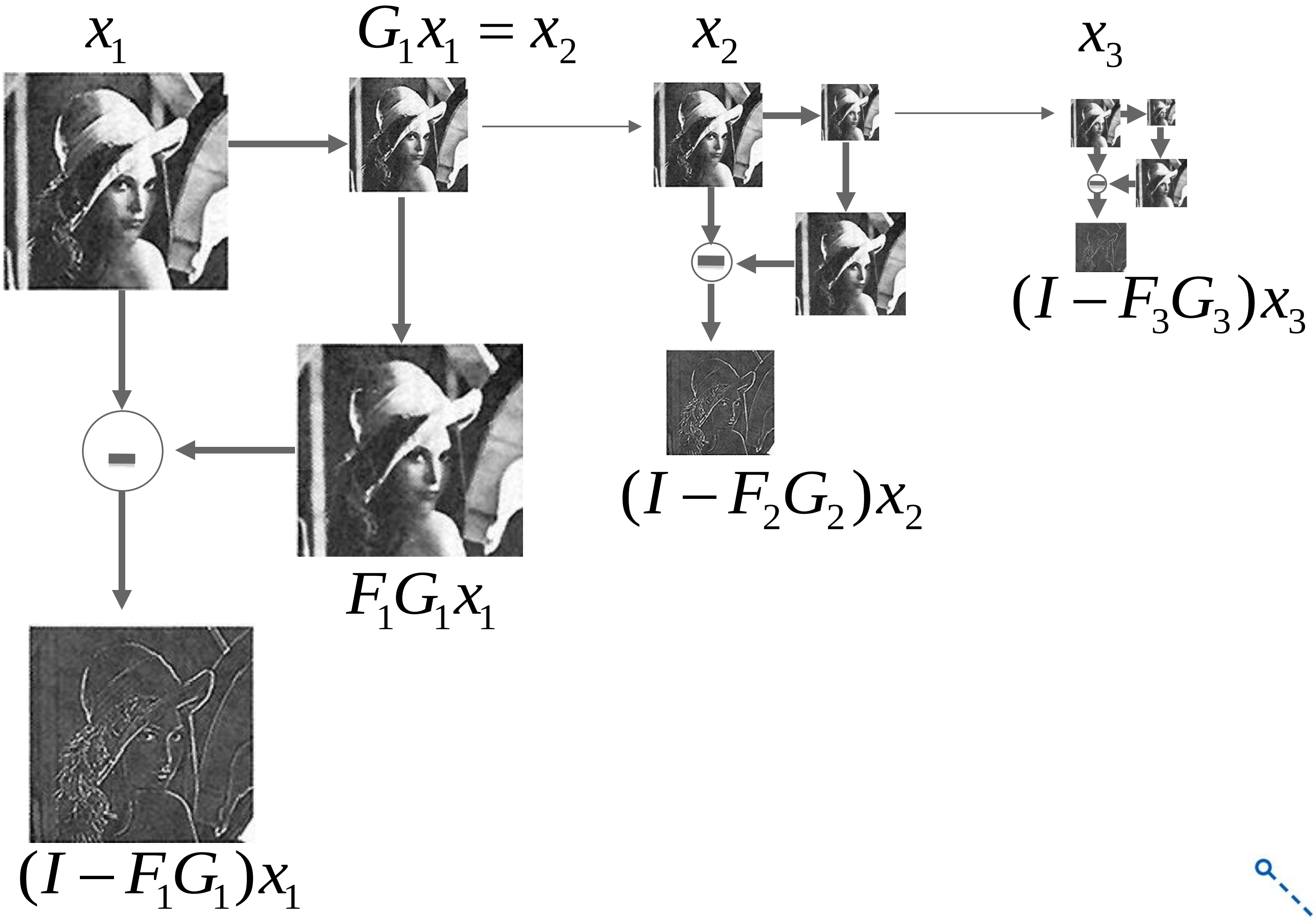
Szeliski

Laplacian Pyramid (Recap)

- **Difference between up-sampled Gaussian pyramid and Gaussian pyramid**.

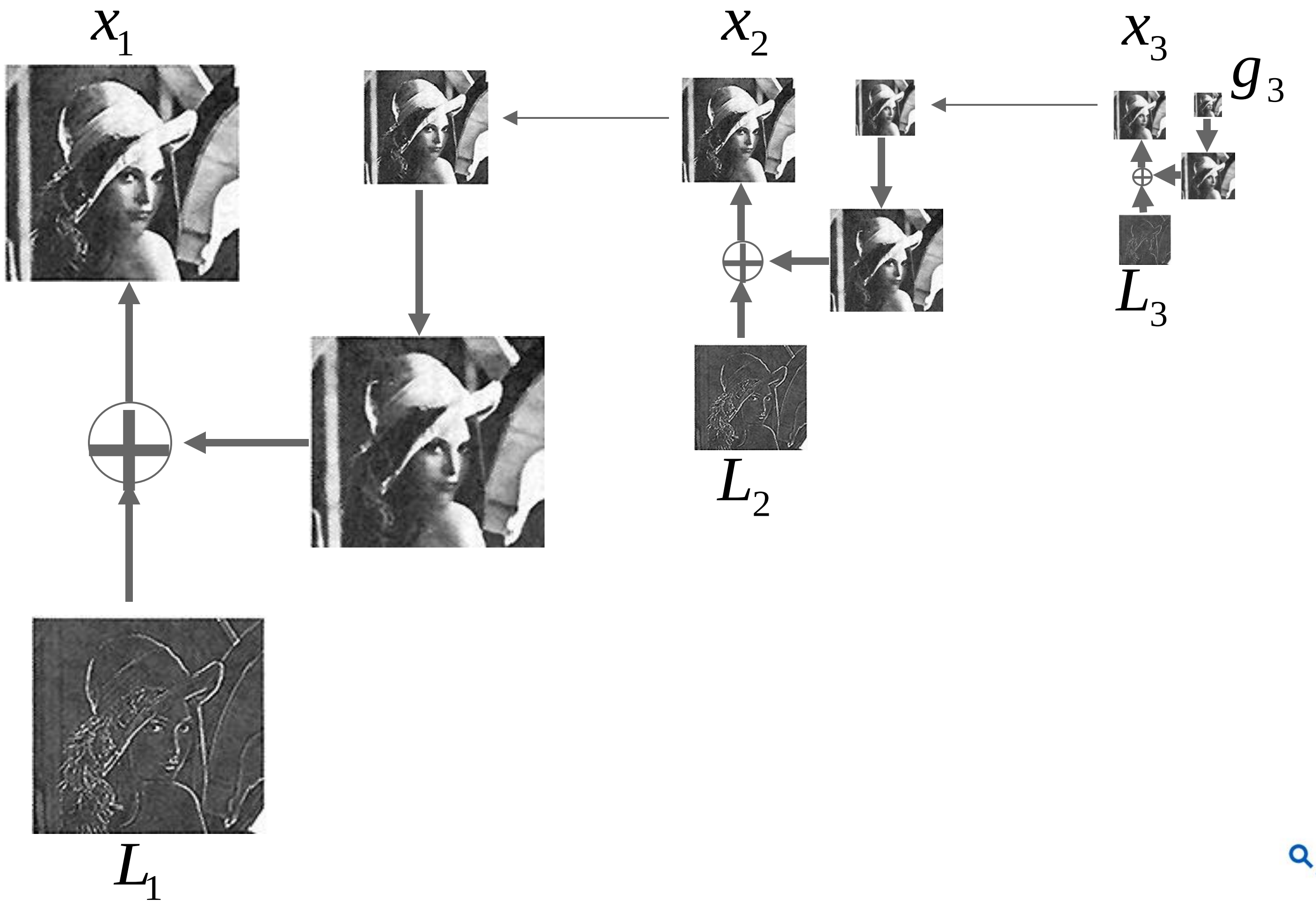
- **Band pass filter** - each level represents spatial frequencies (largely) unrepresented at other level.

Laplacian pyramid (Recap)



Laplacian Pyramid (Recap)

- Reconstruction: recover x_1 from L_1, L_2, L_3 and g_3



Laplacian Pyramid (Recap)

- Information captured at each level of a Gaussian (top) and Laplacian (bottom) pyramid
  - showing full resolution.

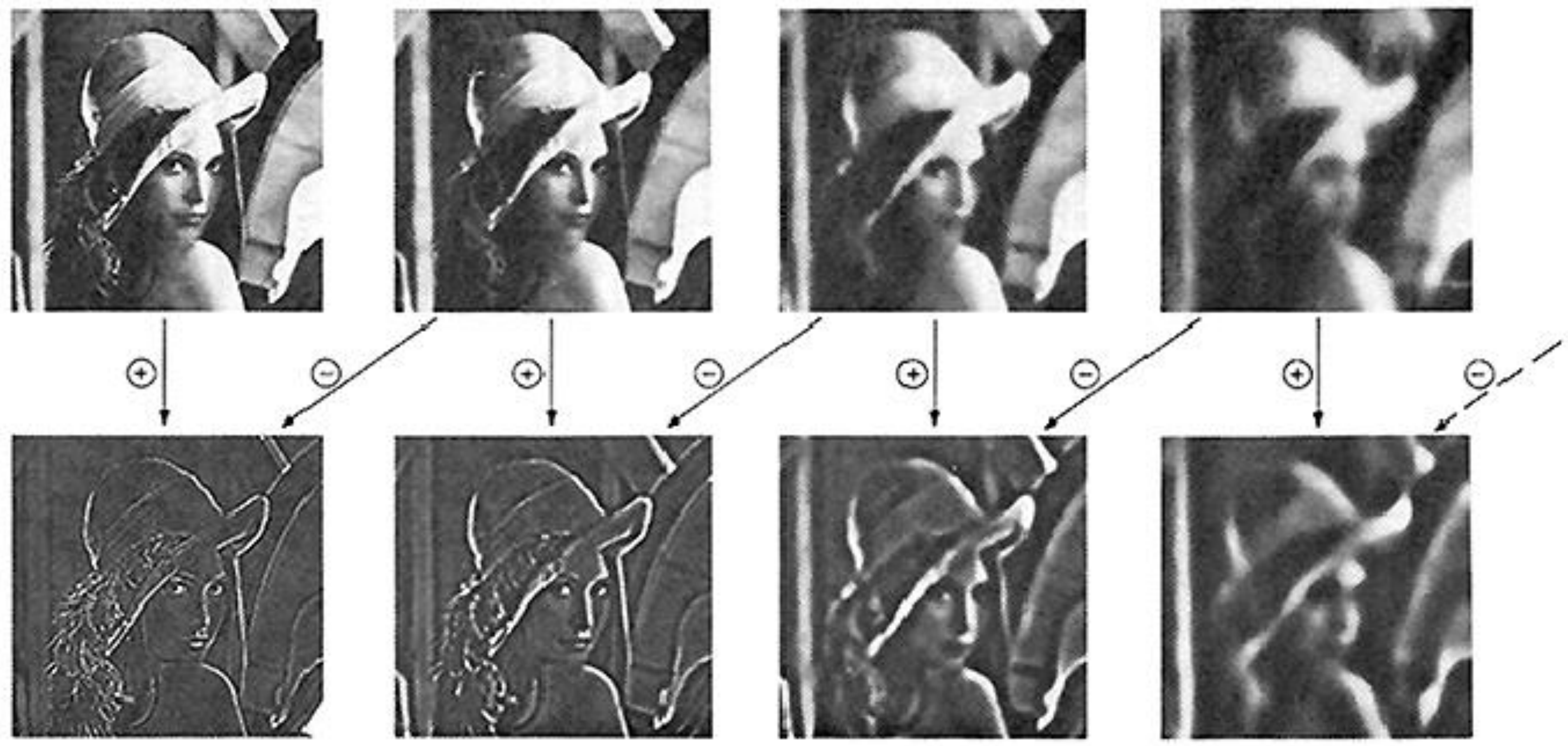
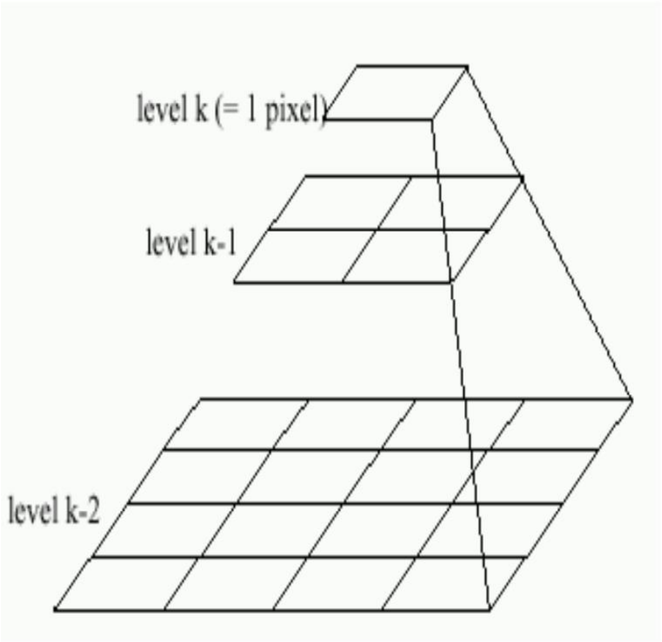
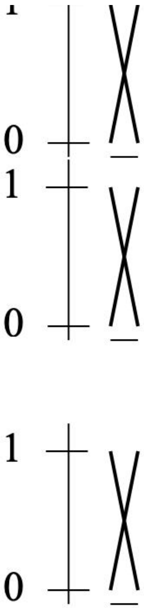


Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.

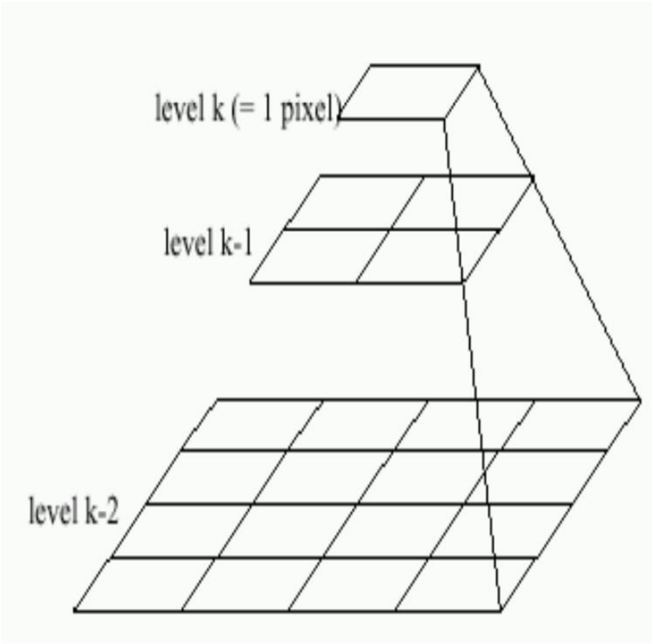
Pyramid Blending



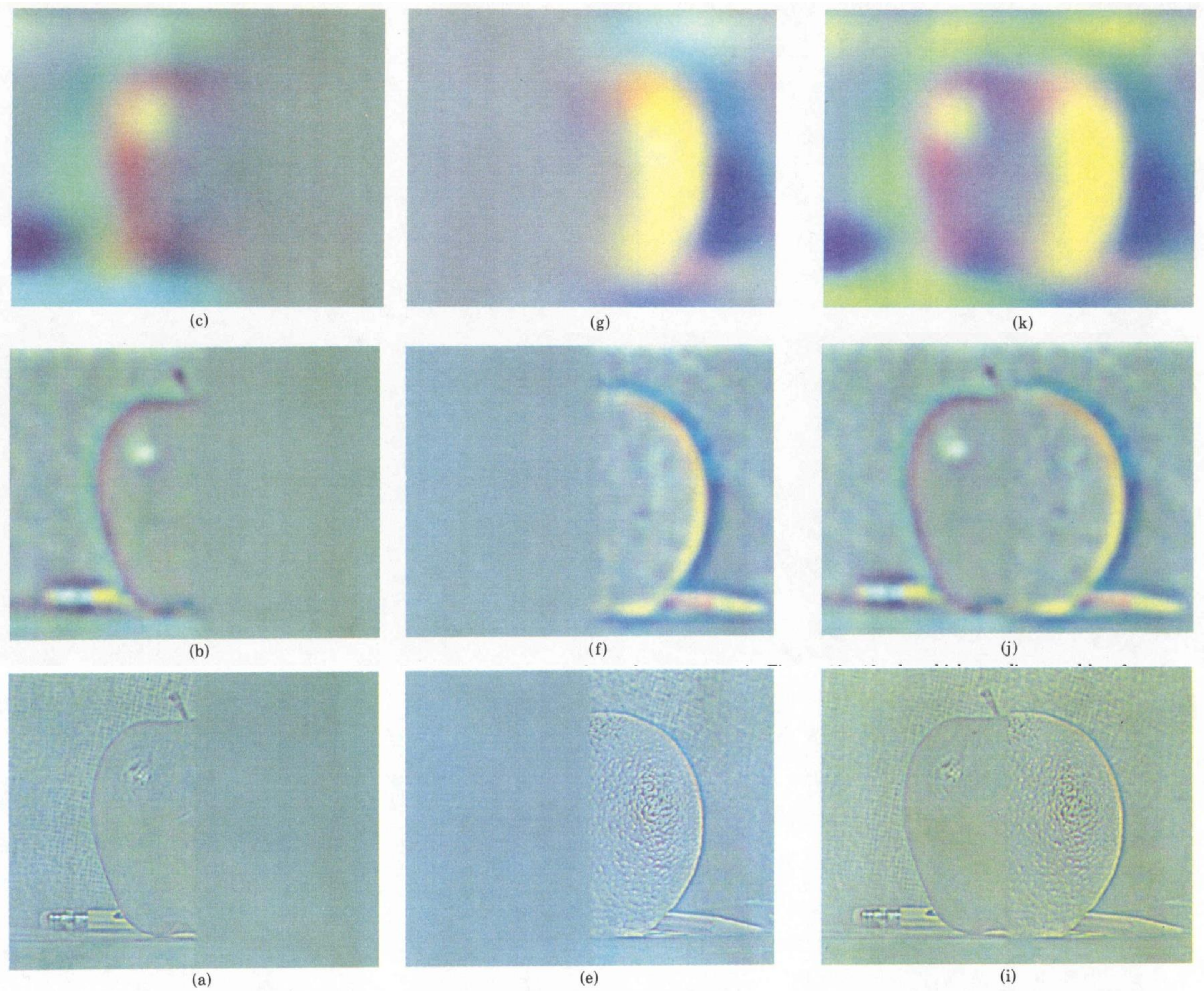
Left pyramid



blend



Right pyramid



(c) (g) (k)

(b) (f) (j)

(a) (e) (i)

Laplacian
level
4

Laplacian
level
2

Laplacian
level
0

left pyramid

right pyramid

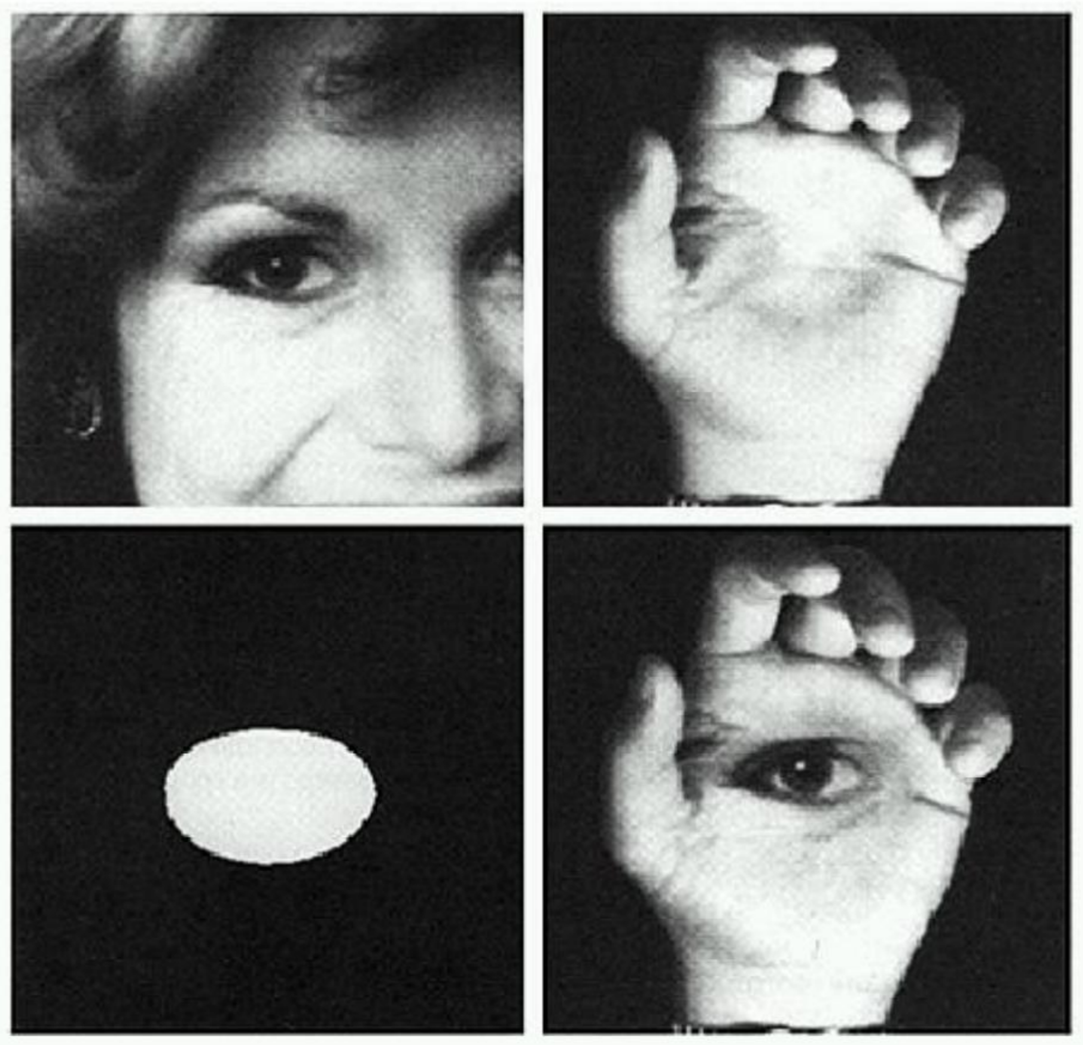
blended pyramid

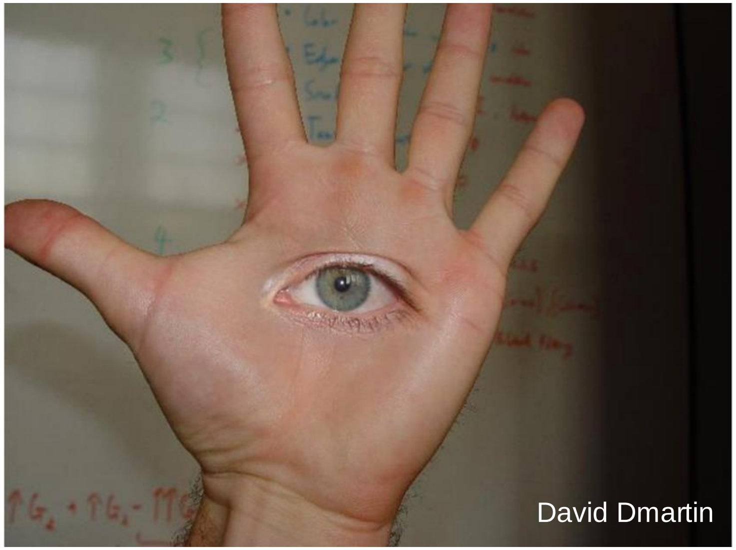
Szeliski

Laplacian Pyramid: Blending

- 1. Build Laplacian pyramids LA and LB from images A and B
- 2. Build a Gaussian pyramid GR from selected region R
- 3. Combined pyramid LS from LA / LB using GR as weights:
  - LS(i, j) = GR(i, j) * LA(i, j) + (1 - GR(i, j)) * LB(i, j)
- 4. Collapse the LS pyramid to get the final blended image

Blending Regions





David Dmartin

Chris Cameron

Season Blending

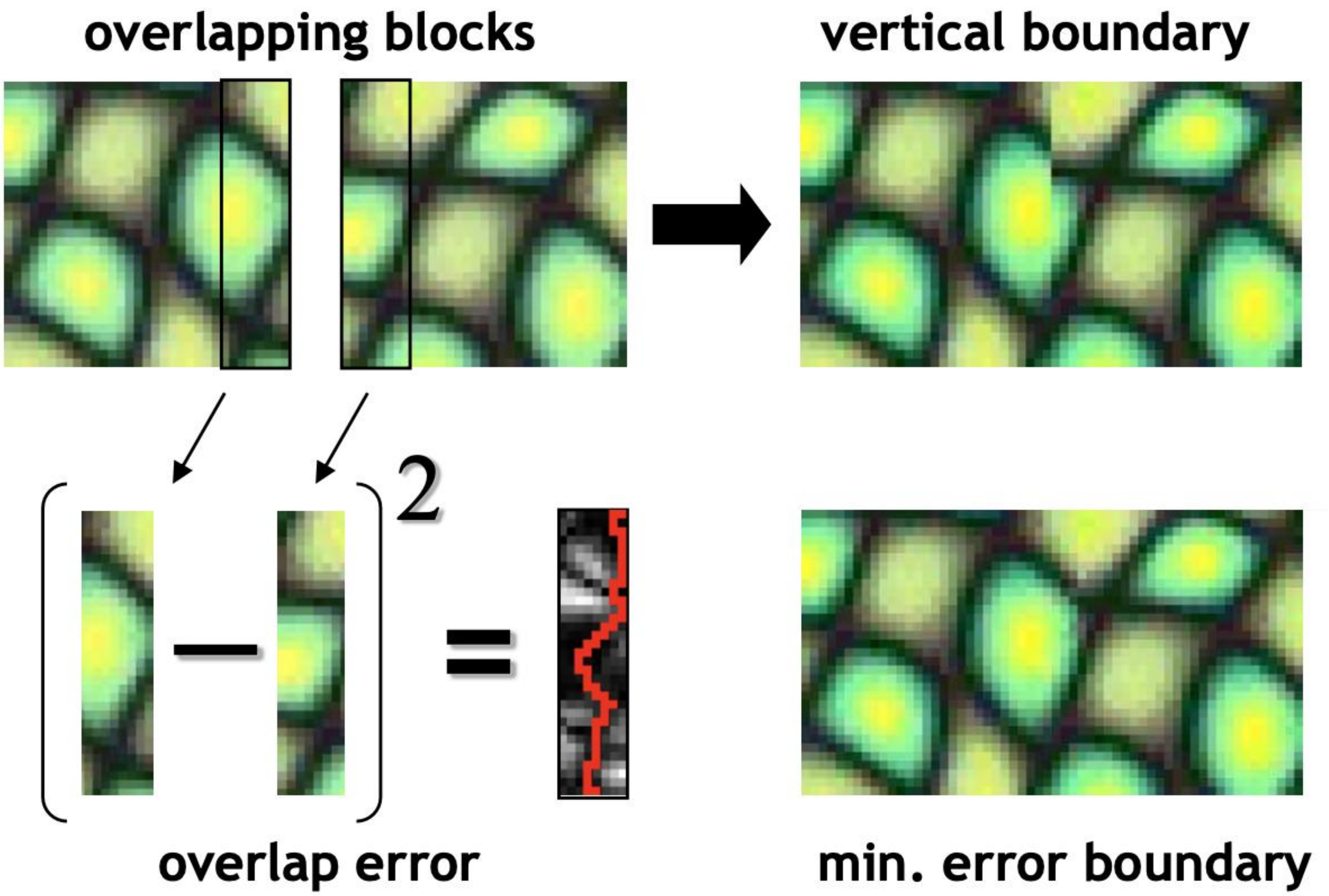
Don’t blend, CUT!

- So far we only tried to blend between two images.
- What about finding an optimal seam?

Don’t blend, CUT!

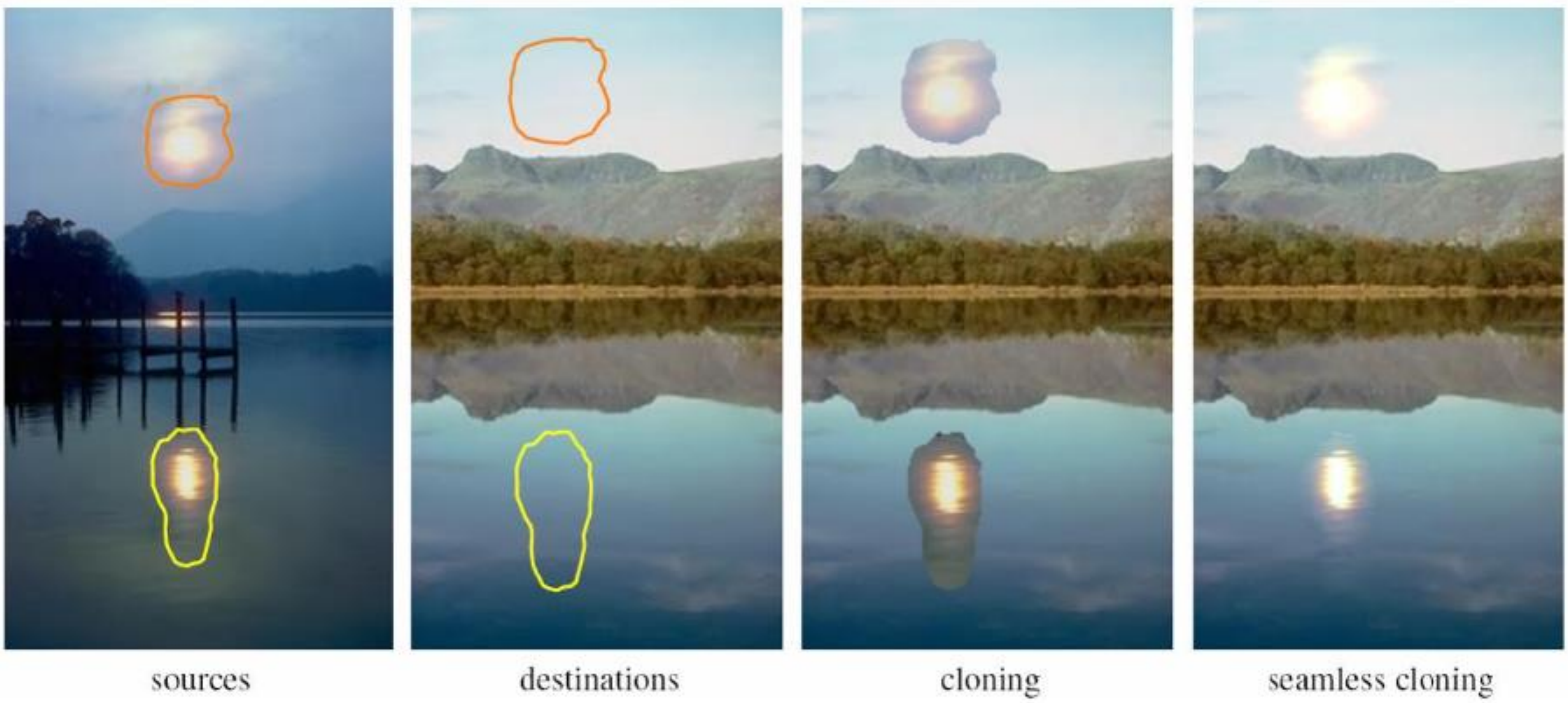
- Segment the mosaic
  - Single source image per segment
- Avoid artifacts along boundaries
  - Dijkstra’s algorithm

Minimal error boundary



Gradient Domain Blending

- Blend the gradients of the two images, then integrate.

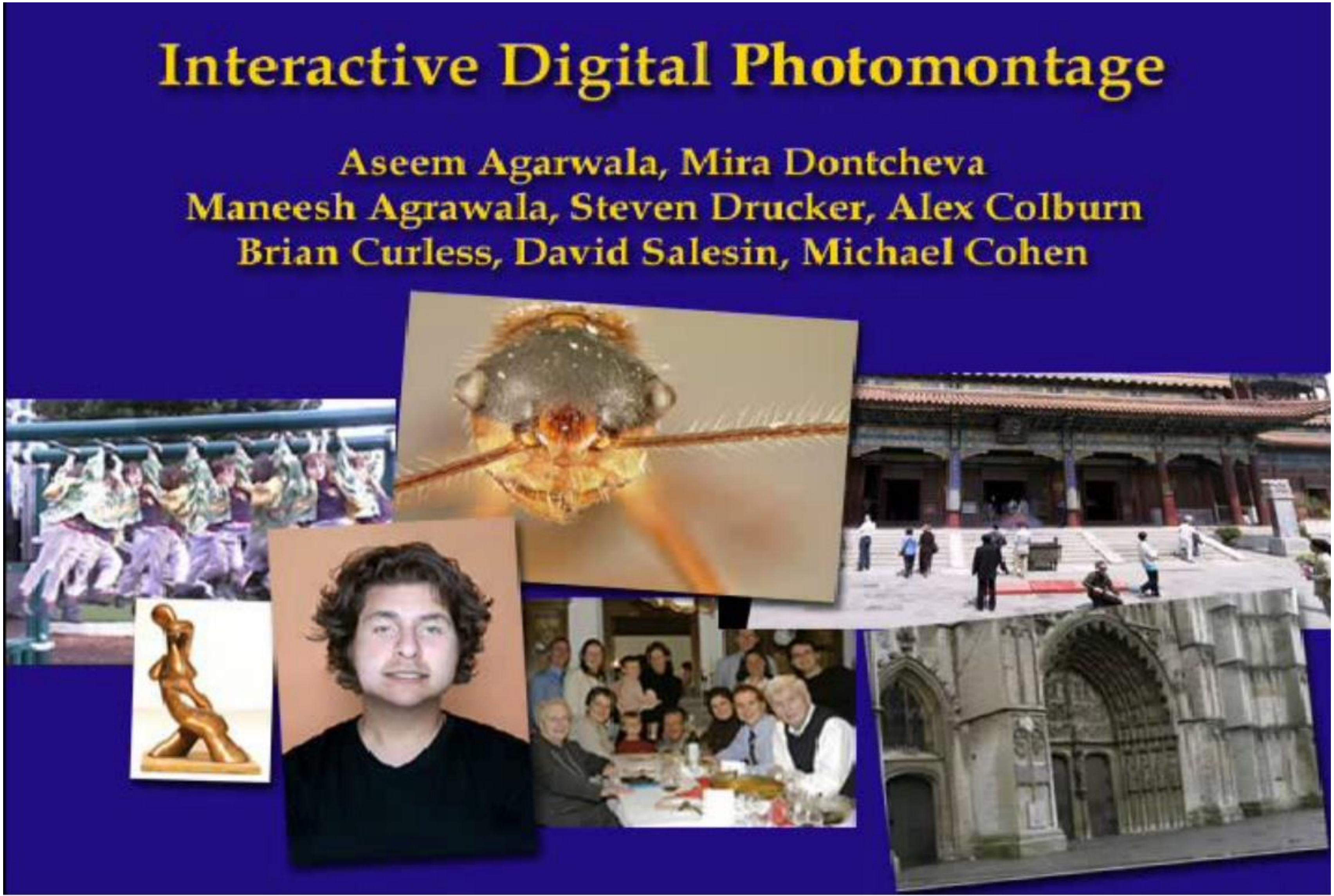


Perez et al, SIGGRAPH 2003

Gradient Domain Blending

- Blend the gradients of the two images, then integrate.

Watch a video



Interactive Digital Photomontage

Aseem Agarwala, Mira Dontcheva
Maneesh Agrawala, Steven Drucker, Alex Colburn
Brian Curless, David Salesin, Michael Cohen

https://youtu.be/Rp7uDRdQRFc

Final thought: What is a “panorama”?

- Tracking a subject
- Repeated (best) shots
- Multiple exposures
- “Infer” what photographer wants?
- Also referred to as “Computational Photography”

Szeliski

Stitching

- Feature Detection
- Feature Matching
- Homography
- RANSAC
- Global alignment
- Warping
- Blending


(a) (b)