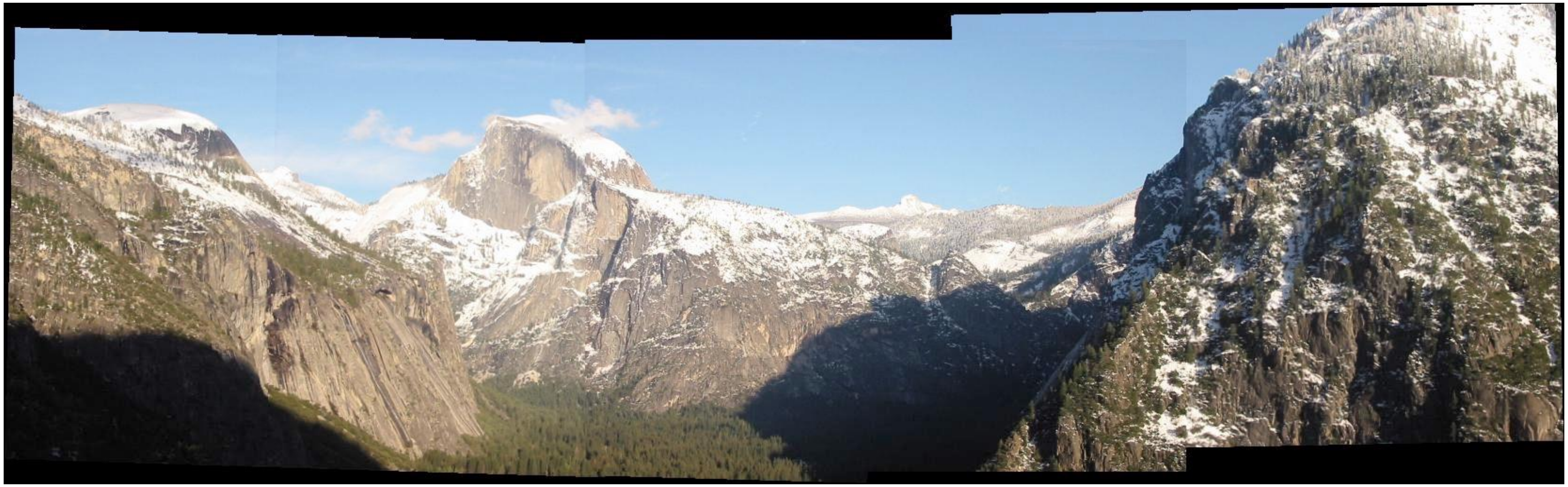
Image Stitching

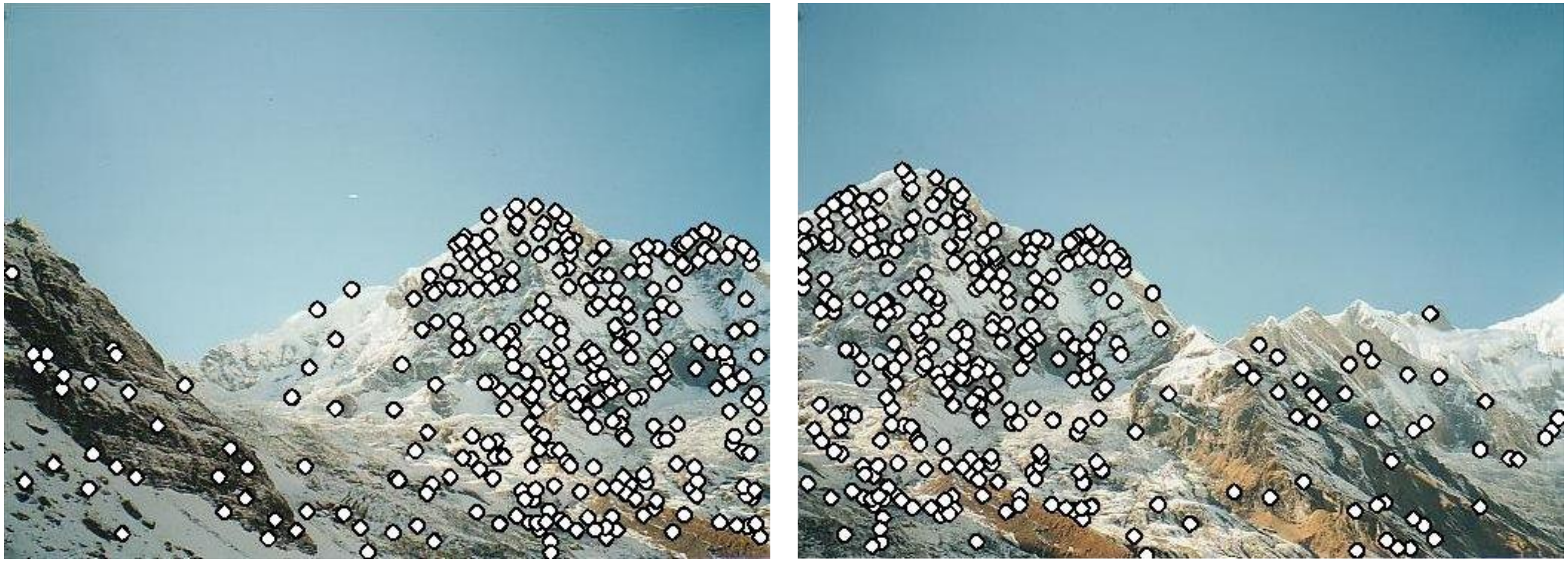


Gang Pan
gpan@zju.edu.cn

Image Stitching

Slide credit: Darya Frolova, Denis Simakov

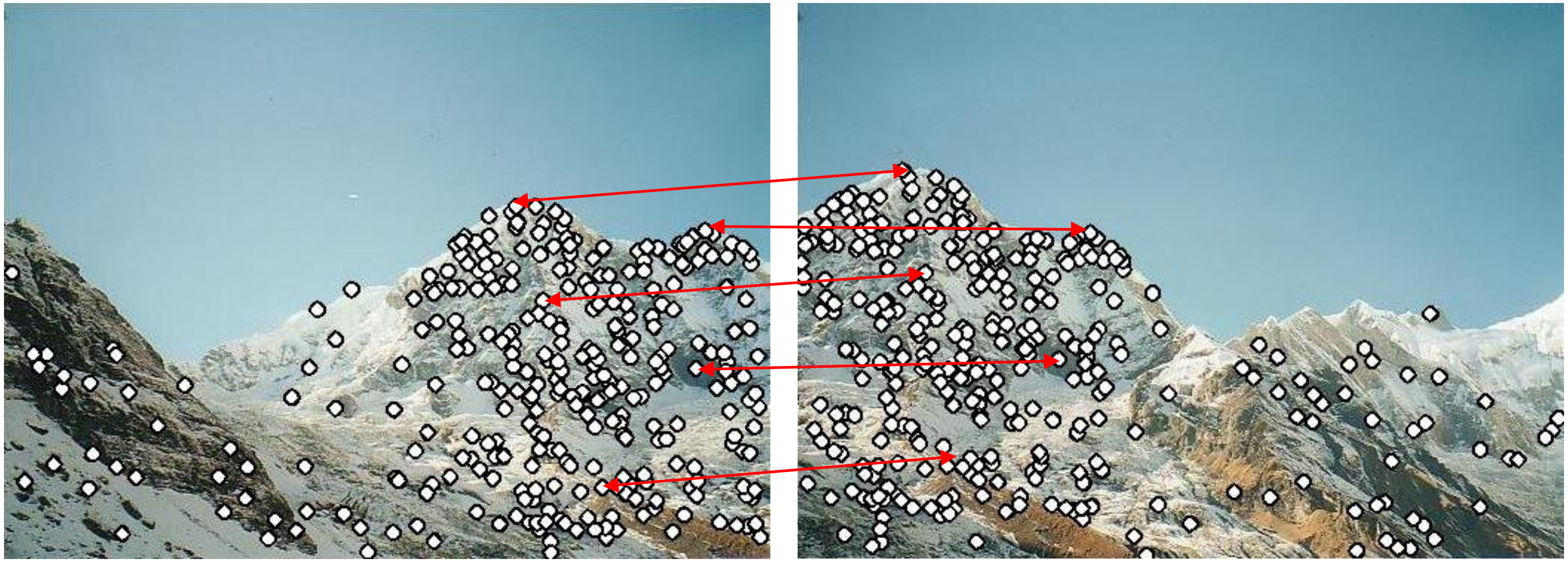
Application: Image Stitching



- Procedure:
  - Detect feature points in both images

Slide credit: Darya Frolova, Denis Simakov

Image Stitching



- Procedure:
  - Detect feature points in both images
  - Find corresponding pairs

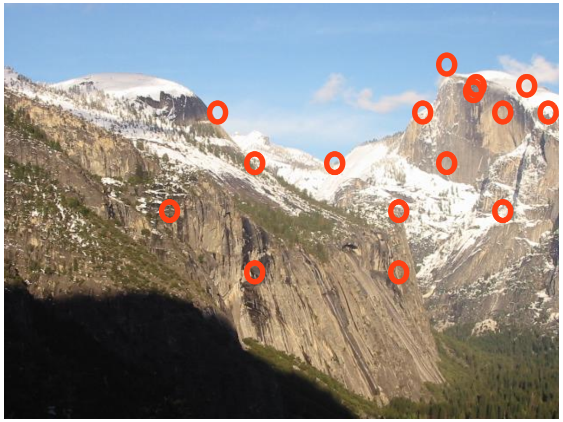
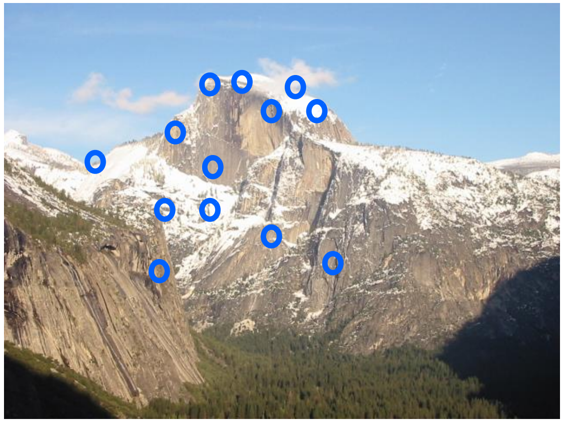
Slide credit: Darya Frolova, Denis Simakov

Application: Image Stitching

- Procedure:
  - Detect feature points in both images
  - Find corresponding pairs
  - Use these pairs to align the images

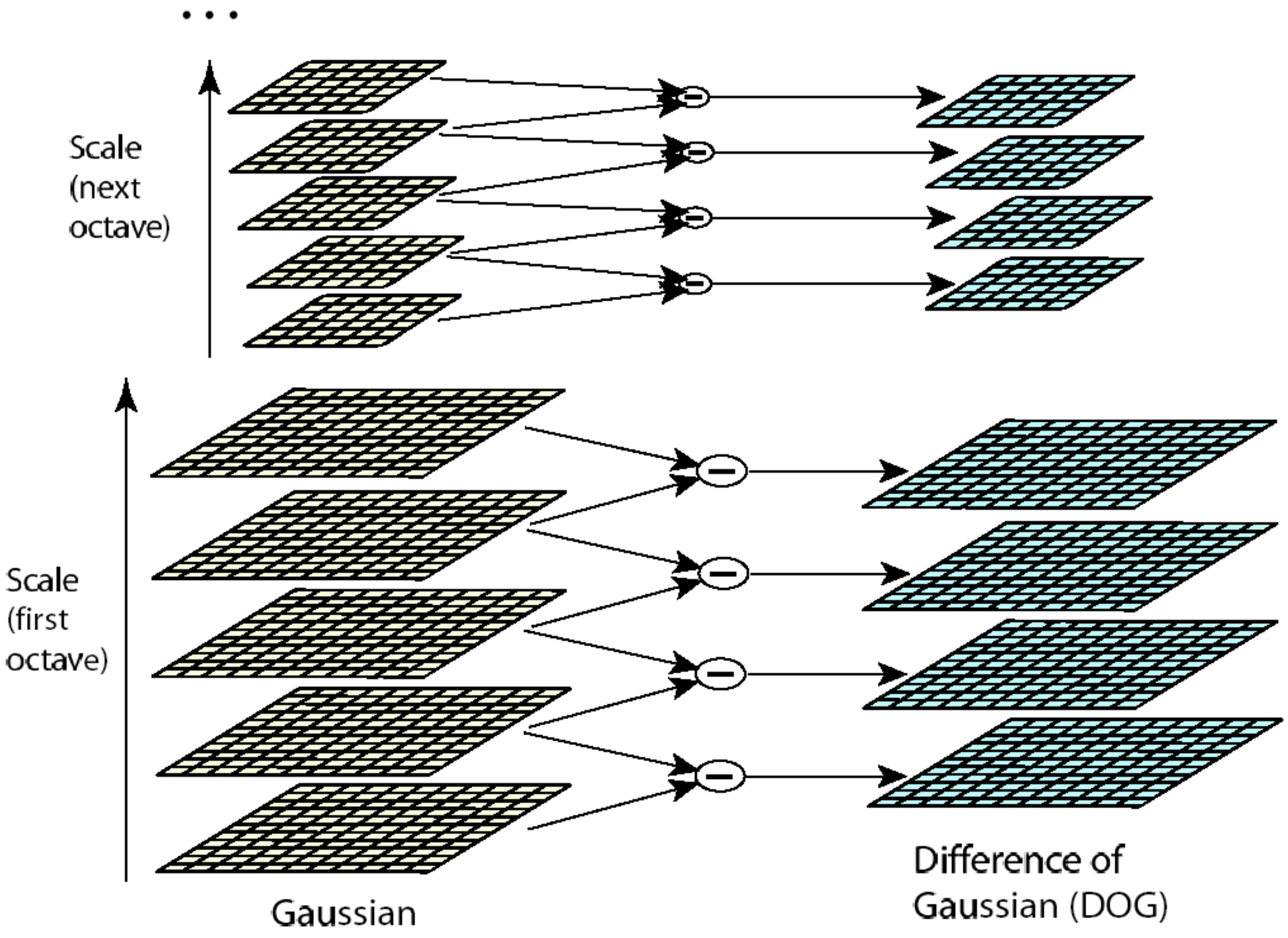
Slide credit: Darya Frolova, Denis Simakov

Main Flow

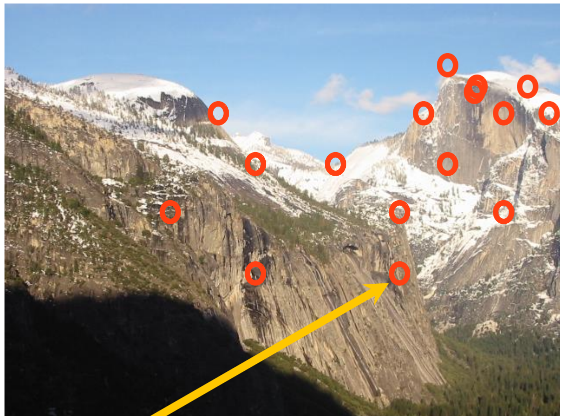
 

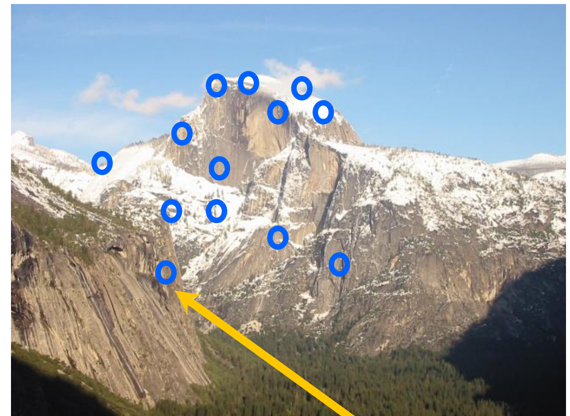
- Detect key points

Detect Key Points



Main Flow



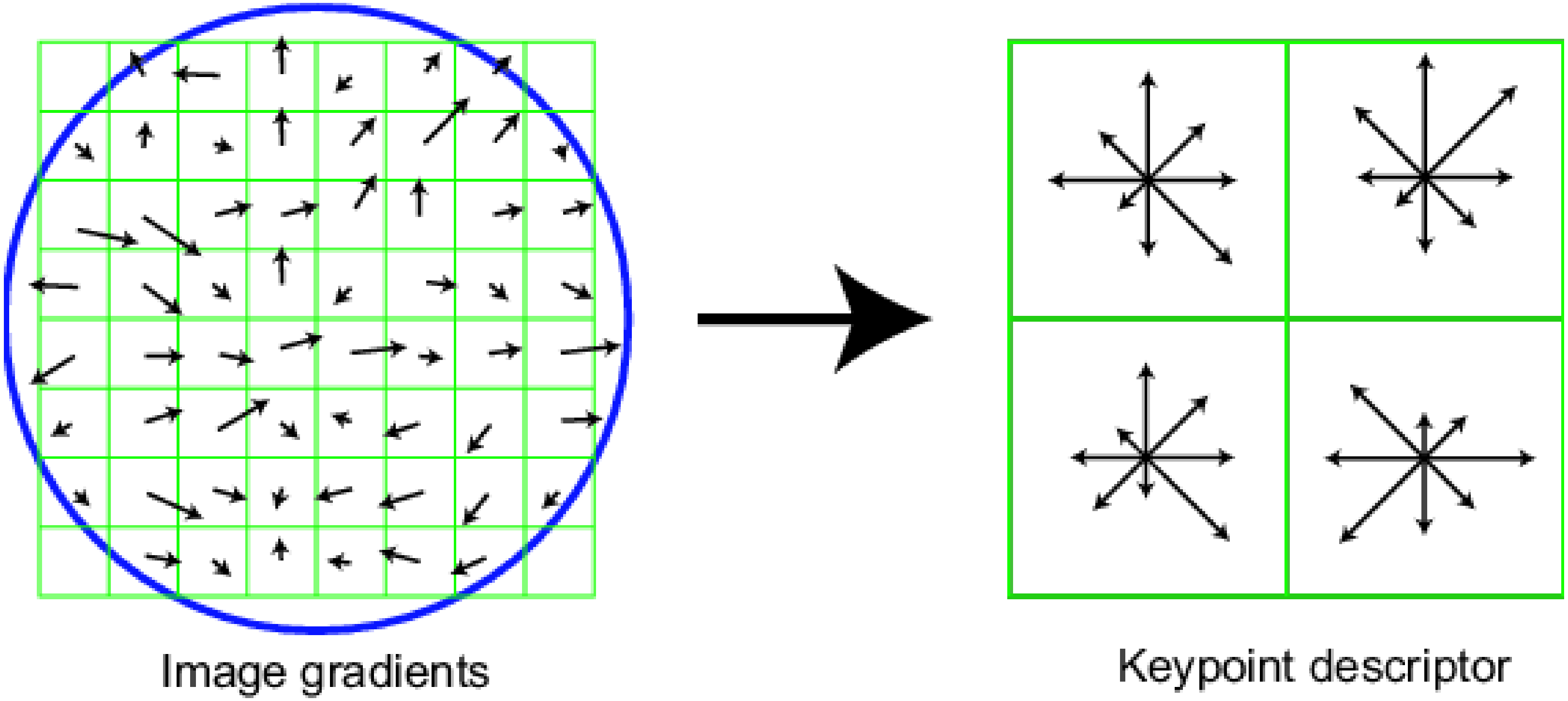


$$(u_1, u_2, \ldots, u_{128})$$

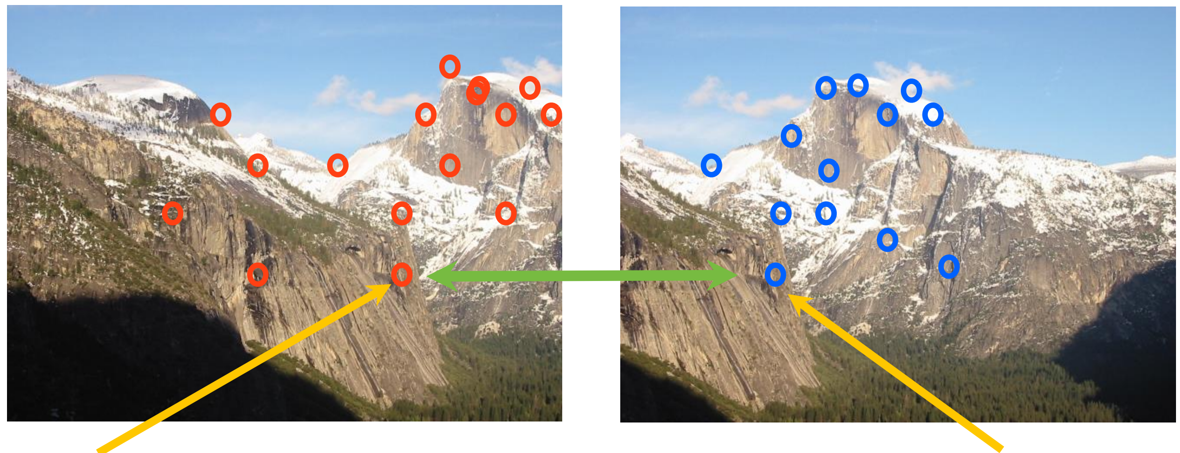
$$(v_1, v_2, \ldots, v_{128})$$

- Detect key points
- Build the SIFT descriptors

Build the SIFT Descriptors



Main Flow

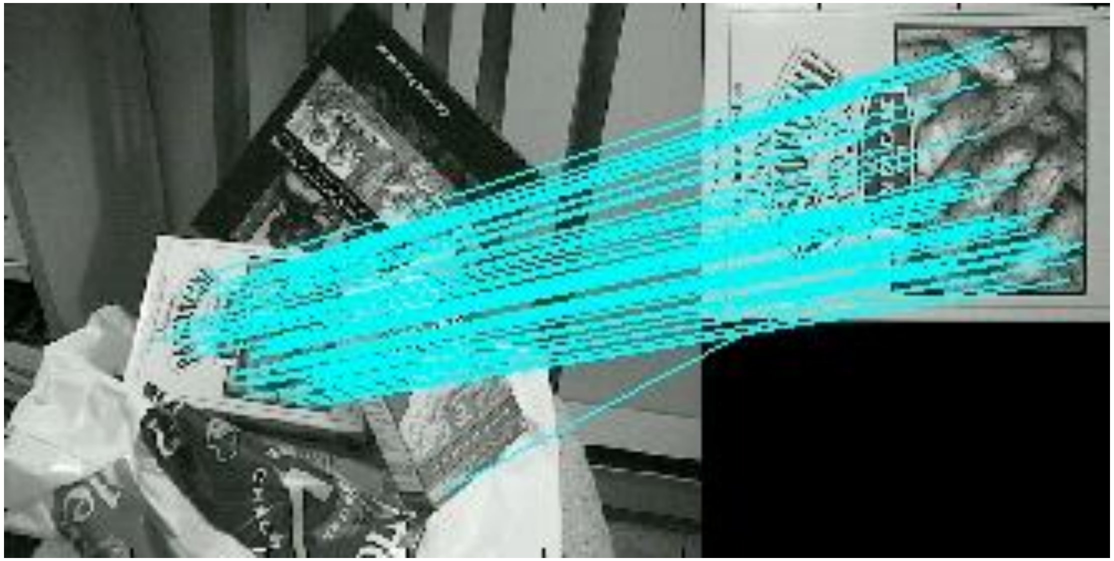


$(u_1, u_2, \ldots, u_{128})$ $(v_1, v_2, \ldots, v_{128})$

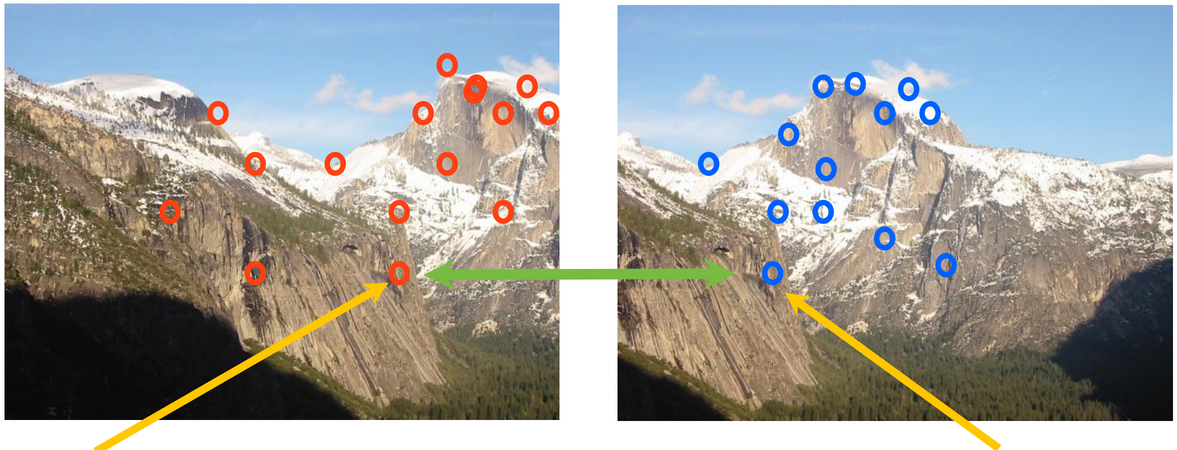
- Detect key points
- Build the SIFT descriptors
- Match SIFT descriptors

Match SIFT Descriptors

- Euclidean distance between descriptors



Main Flow



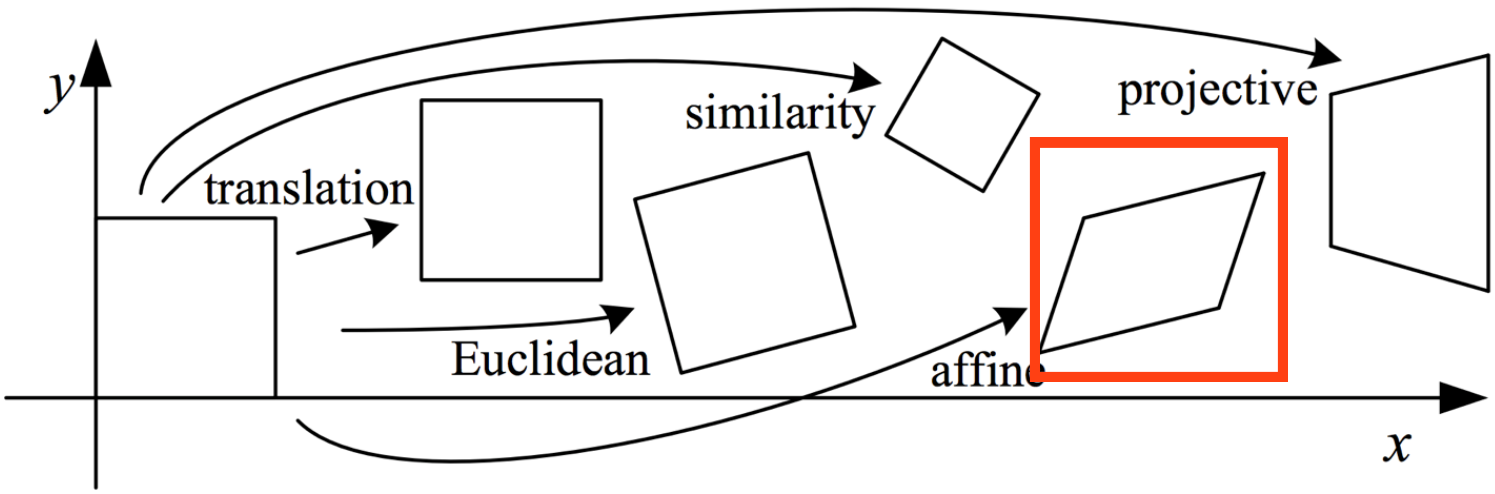
$(u_1, u_2, \ldots, u_{128})$ $(v_1, v_2, \ldots, v_{128})$

- Detect key points
- Build the SIFT descriptors
- Match SIFT descriptors
- **Fitting the transformation**

$$T = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

Fitting the transformation

- 2D transformations



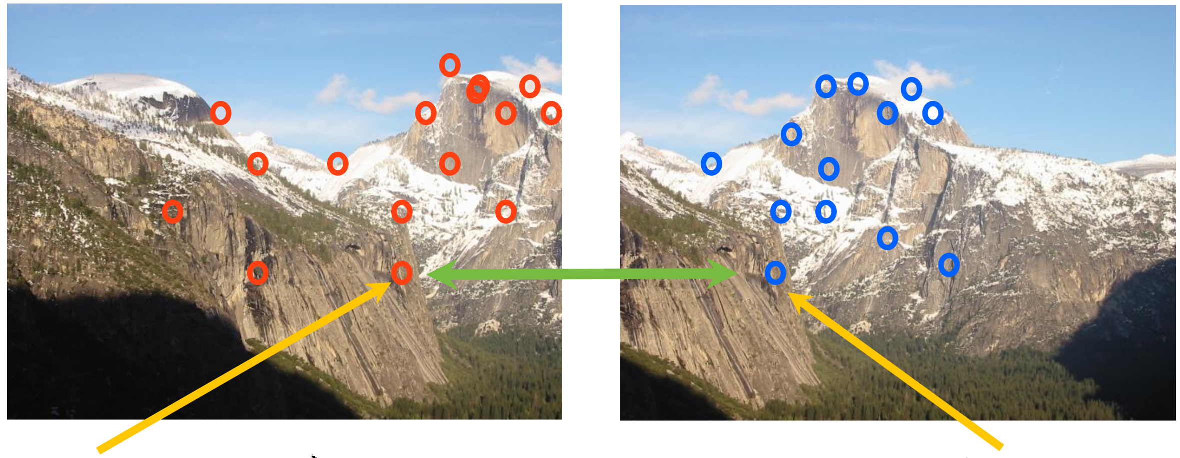
Skeleton Code

- Fit the transformation matrix

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

- Six variables
  - each point give two equations
  - at least three points
- Least squares

Main Flow

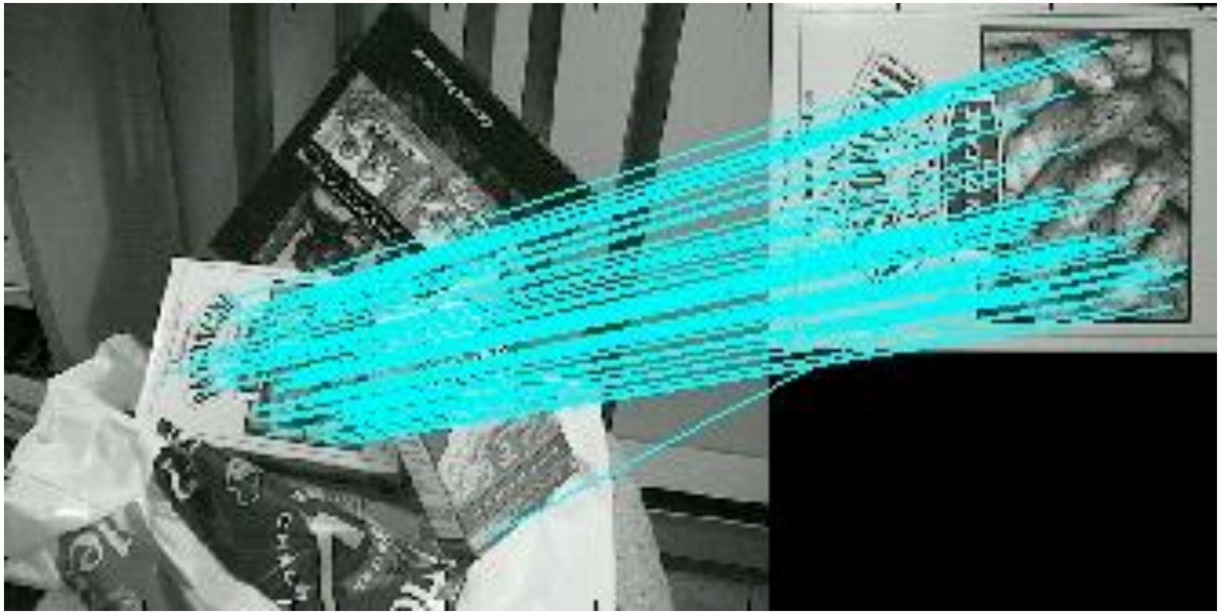


$(u_1, u_2, \ldots, u_{128})$ $(v_1, v_2, \ldots, v_{128})$

- Detect key points
- Build the SIFT descriptors
- Match SIFT descriptors
- Fitting the transformation
- RANSAC

RANSAC

- A further refinement of matches



RANSAC [Fischler & Bolles 1981]

- **RANdom SAmple Consensus**
- Approach: we want to avoid the impact of **outliers**, so let's look for "**inliers**", and use only those.
- Intuition: if an outlier is chosen to compute the current fit, then the resulting line won't have much **support from rest of the points**.

Slide credit: Kristen Grauman

RANSAC [Fischler & Bolles 1981]

RANSAC loop:

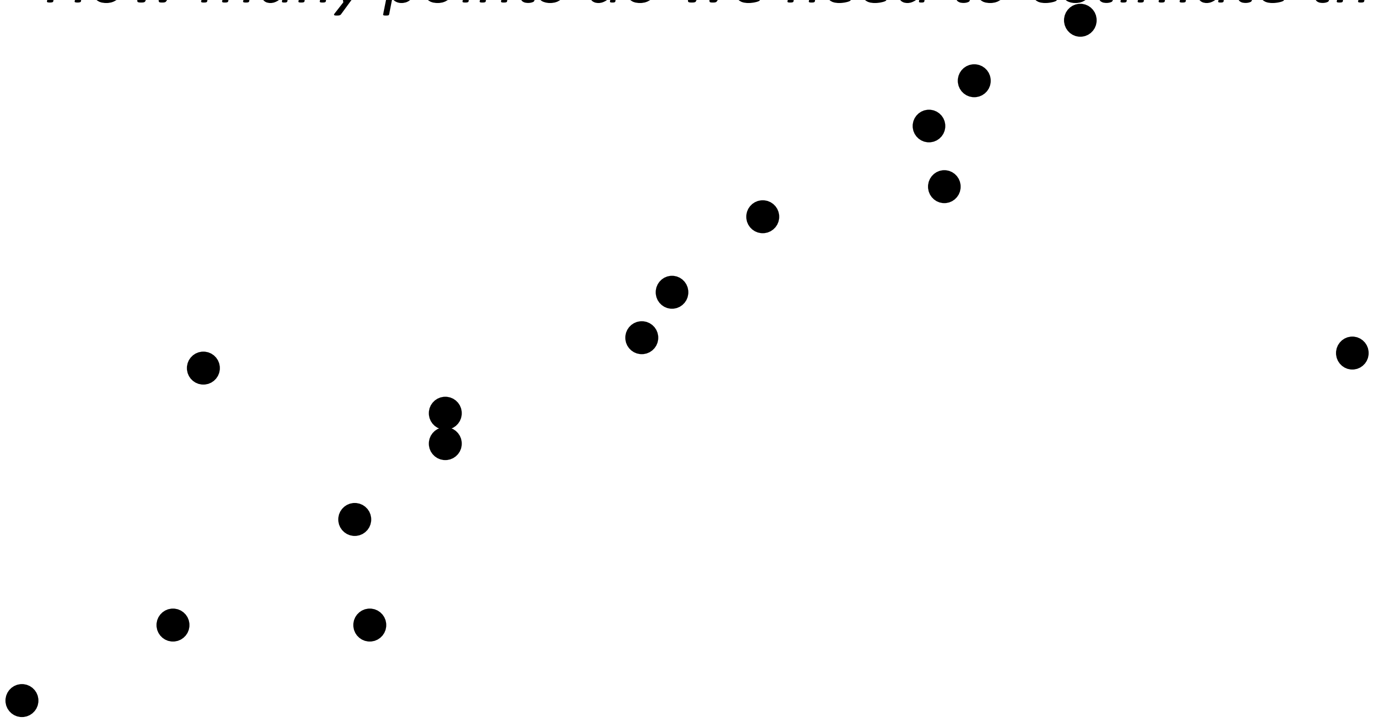
1. Randomly select a *seed group* of points on which to base transformation estimate (e.g., a group of matches)
2. Compute transformation from seed group
3. Find ***inliers*** to this transformation
4. If the number of inliers is sufficiently large, **re-compute** least-squares estimate of transformation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

Keep the transformation with the largest number of inliers

Slide credit: Kristen Grauman

RANSAC Line Fitting Example

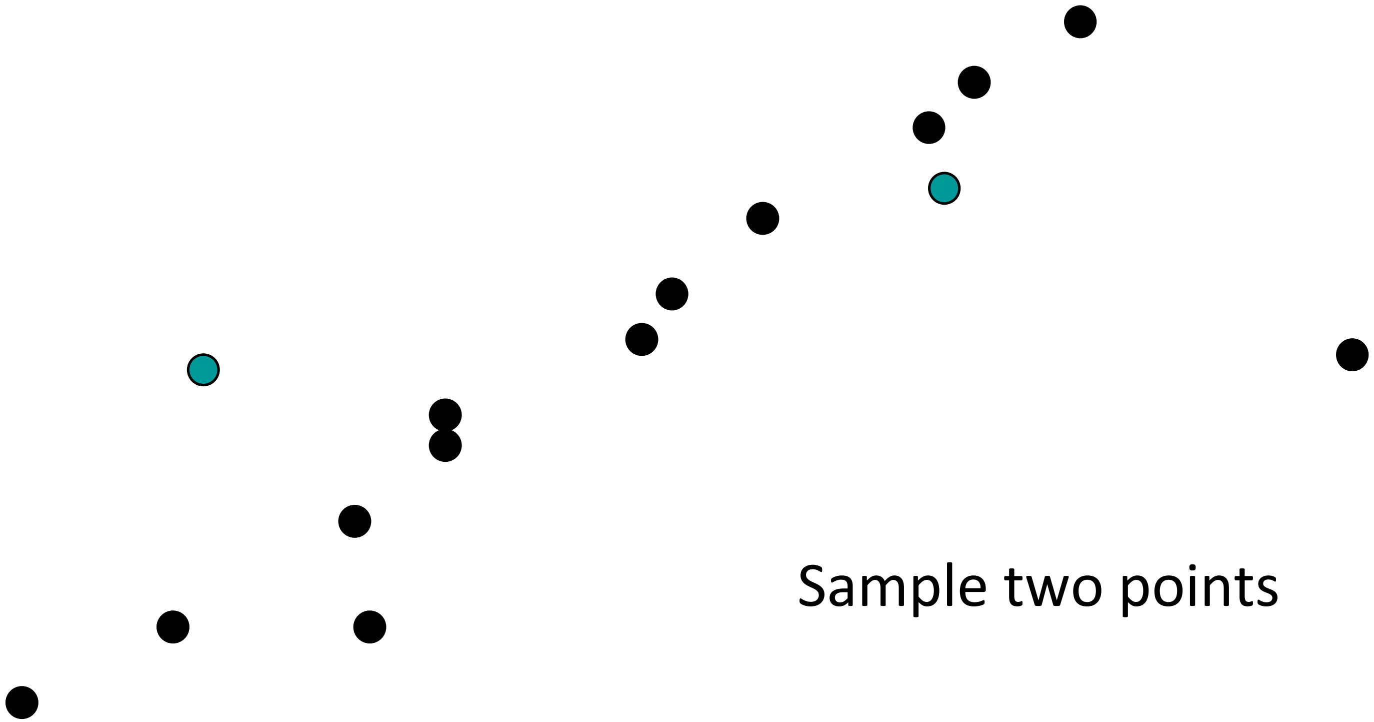
- Task: Estimate the best line
  - *How many points do we need to estimate the line?*



Slide credit: Jinxiang Chai

RANSAC Line Fitting Example

- Task: Estimate the best line

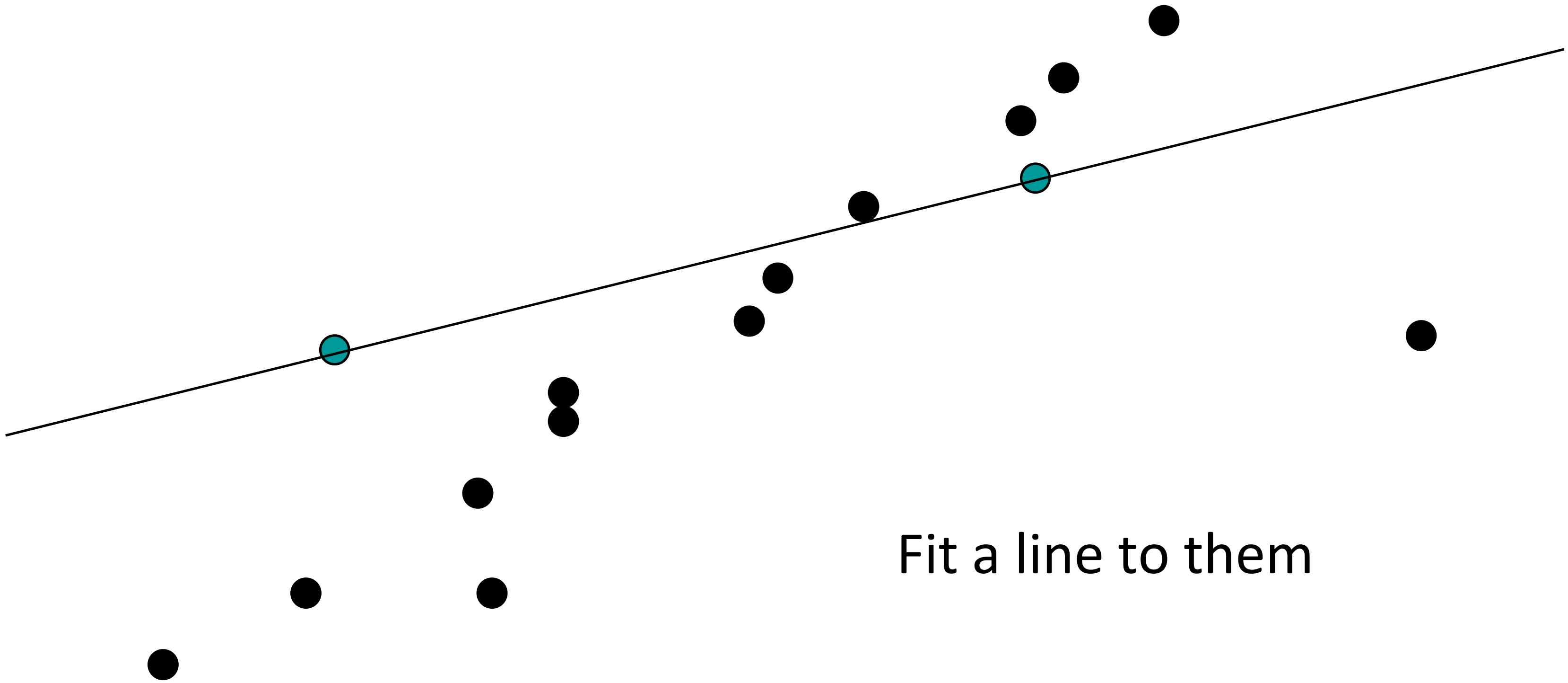


Sample two points

Slide credit: Jinxiang Chai

RANSAC Line Fitting Example

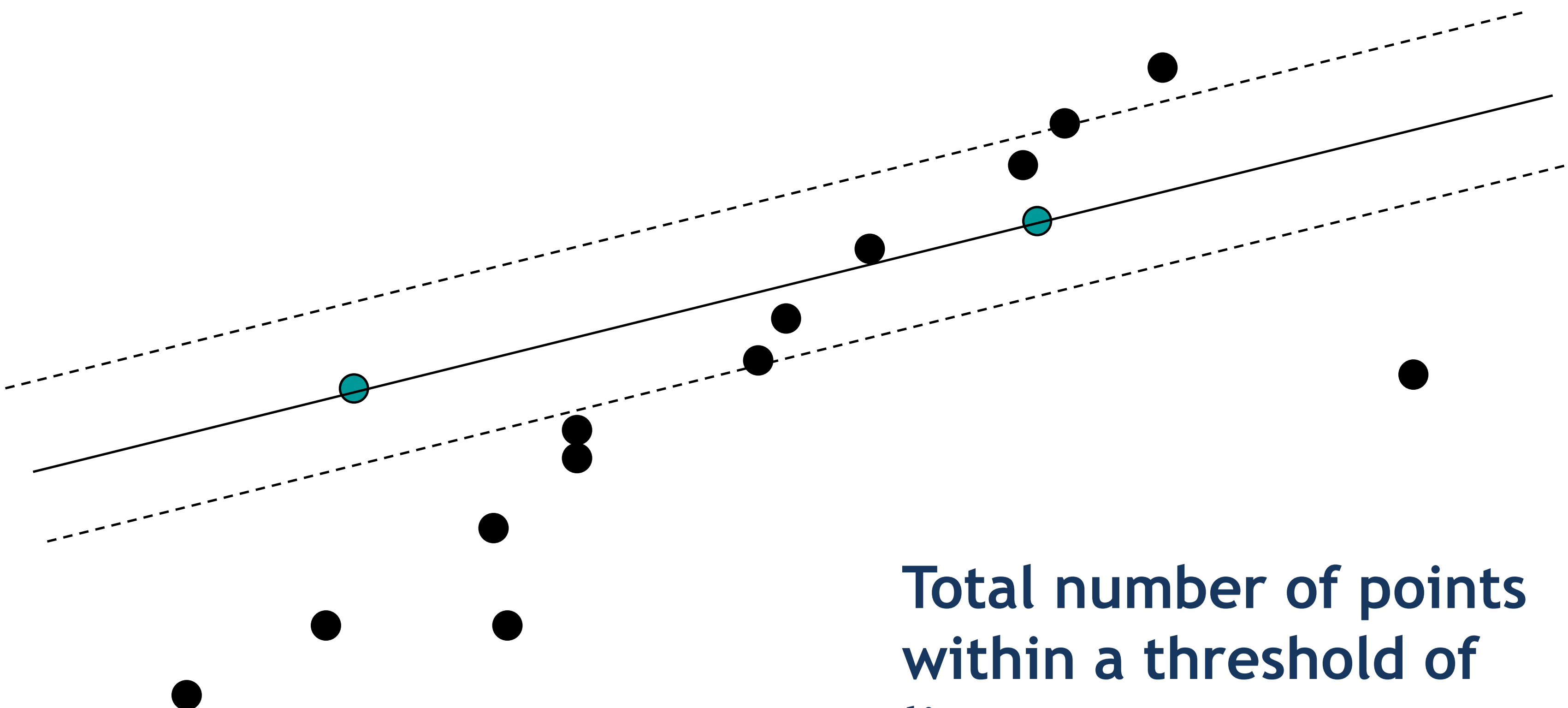
- Task: Estimate the best line



Slide credit: Jinxiang Chai

RANSAC Line Fitting Example

- Task: Estimate the best line

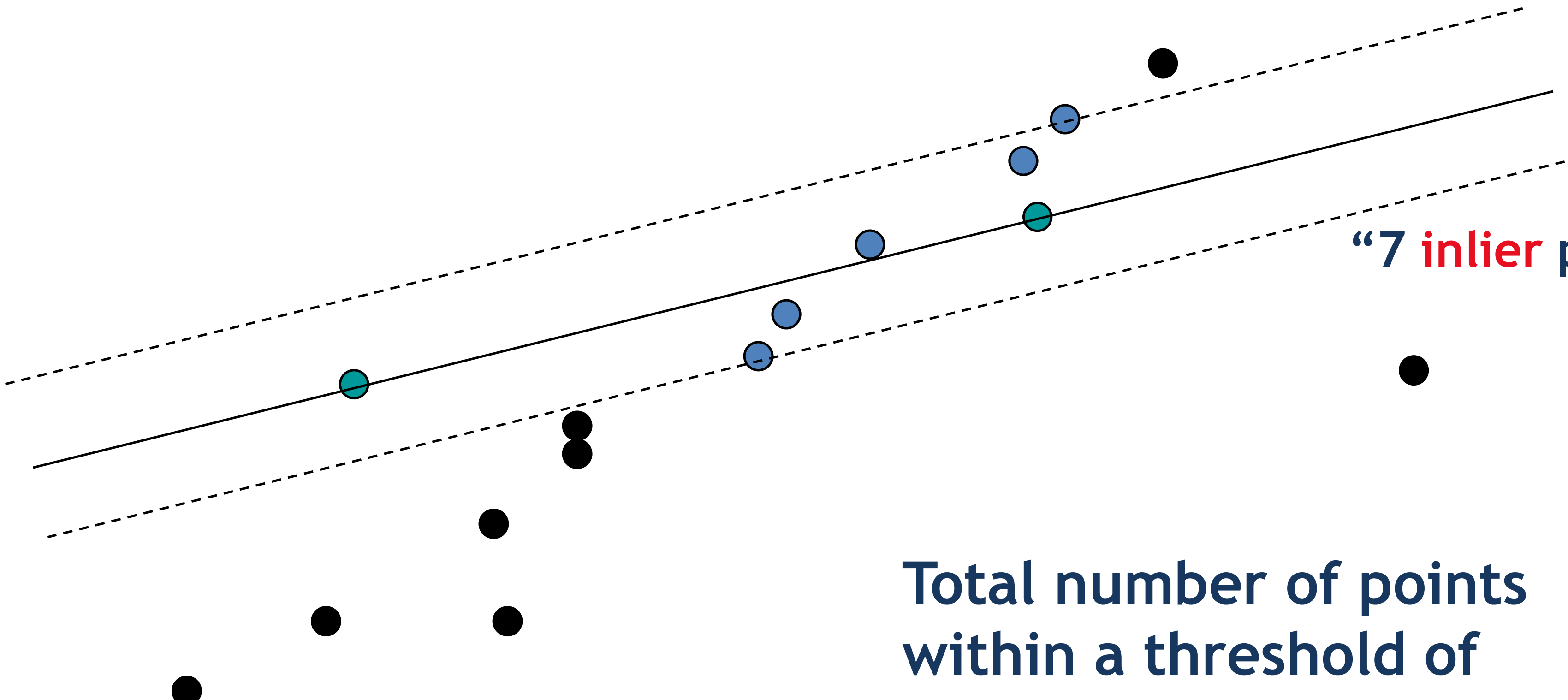


Total number of points within a threshold of line.

Slide credit: Jinxiang Chai

RANSAC Line Fitting Example

- Task: Estimate the best line

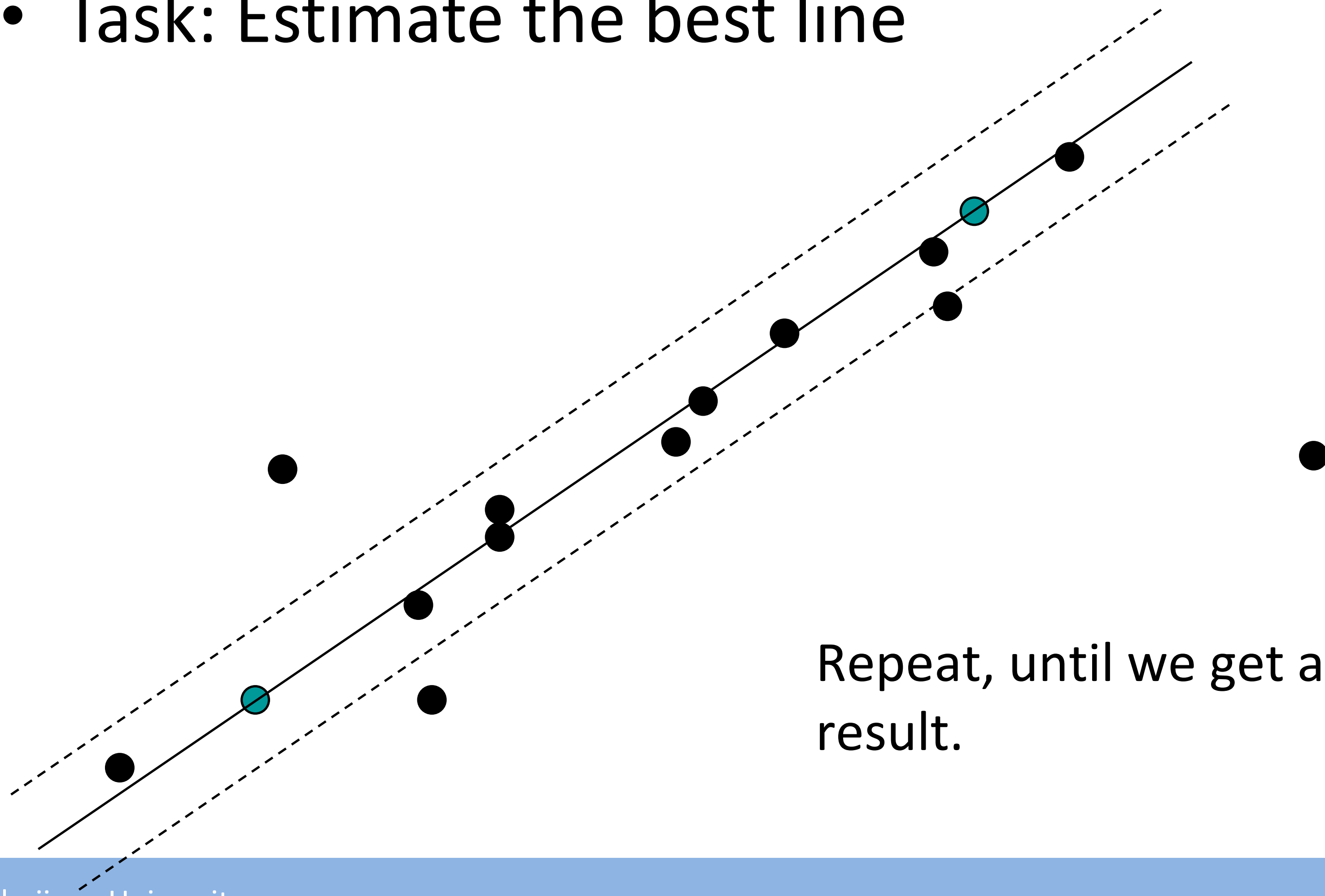


Total number of points within a threshold of line.

Slide credit: Jinxiang Chai

RANSAC Line Fitting Example

- Task: Estimate the best line

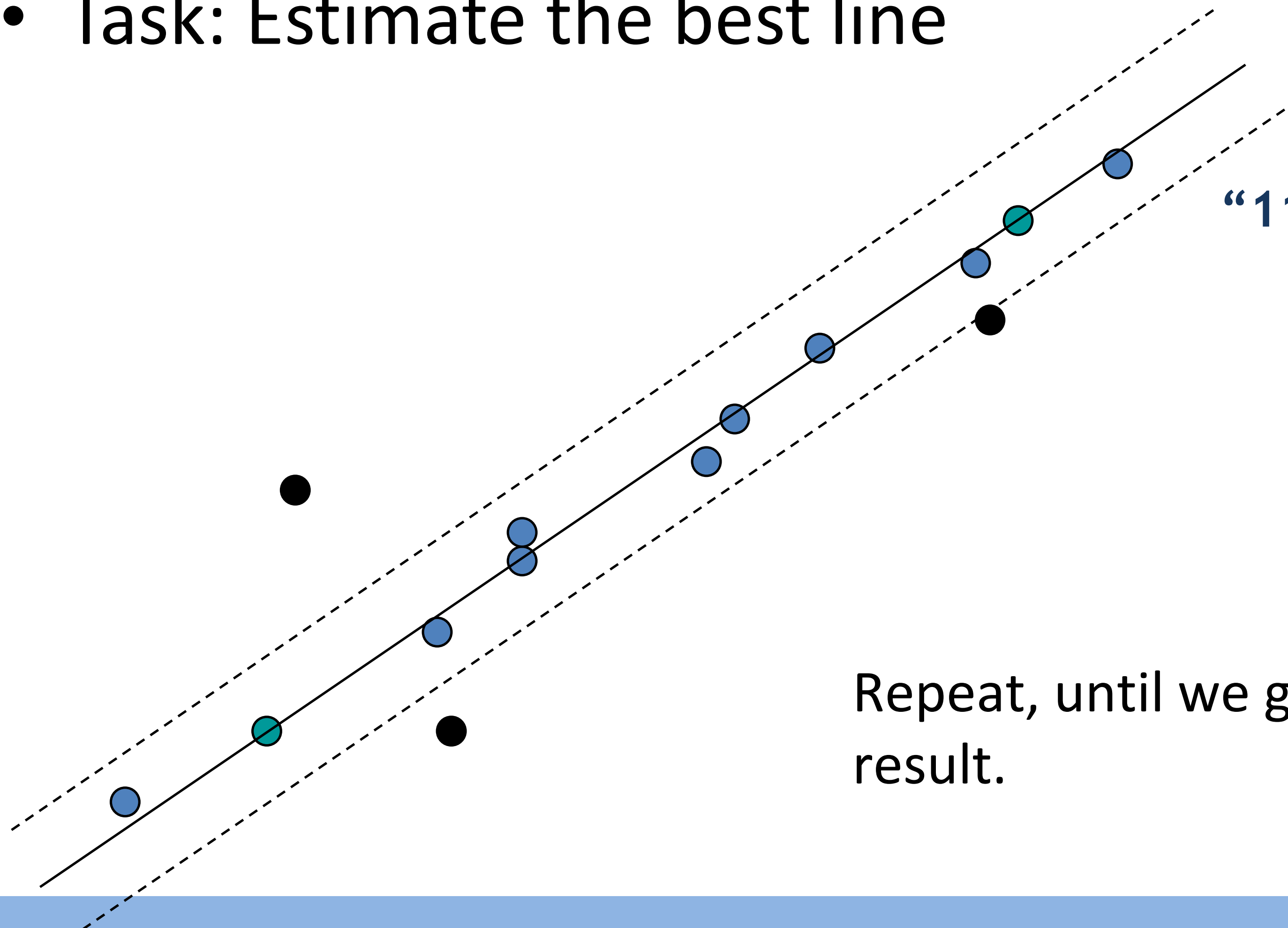


Repeat, until we get a good result.

Slide credit: Jinxiang Chai

RANSAC Line Fitting Example

- Task: Estimate the best line



Repeat, until we get a good result.

Slide credit: Jinxiang Chai

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: How many samples?

- How many samples are needed?
  - Suppose w is fraction of inliers (points from line).
  - n points needed to define hypothesis (2 for lines)
  - k samples chosen.
- Prob. that a single sample of n points is correct: w^n
- Prob. that all k samples fail is: $(1-w^n)^k$

$\Rightarrow$ Choose k high enough to keep this below desired failure rate.

Slide credit: David Lowe

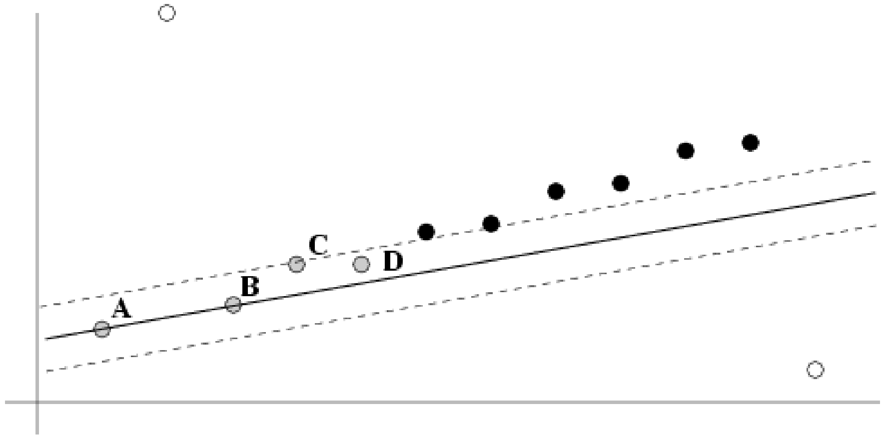
RANSAC: Computed k (p=0.99)

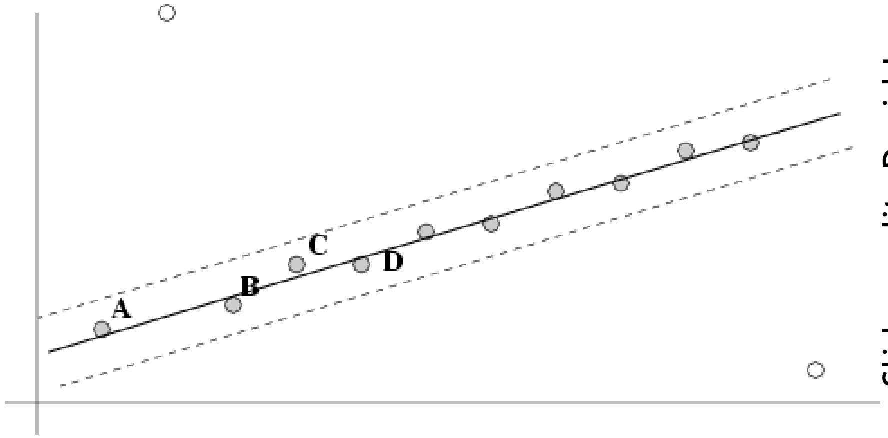
Sample size n	Proportion of outliers 5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Slide credit: David Lowe

After RANSAC

- RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers.
- Improve this initial estimate with estimation over all inliers (e.g. with standard least-squares minimization).
- But this may change inliers, so alternate fitting with re-classification as inlier/outlier.





Slide credit: David Lowe

RANSAC: Pros and Cons

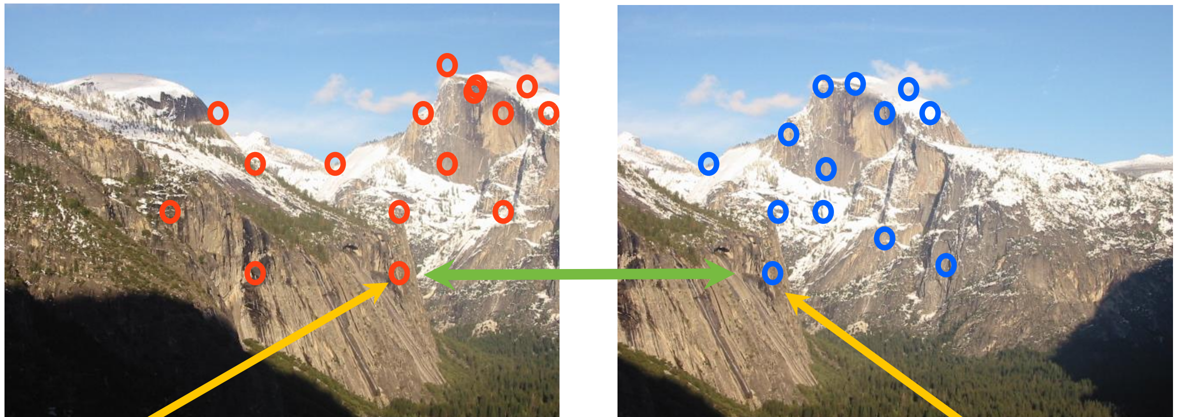
- **Pros:**
  - **General method** suited for a wide range of model fitting problems
  - **Easy** to implement and easy to calculate its failure rate
- **Cons:**
  - Only handles a moderate percentage of outliers without cost blowing up
  - Many real problems have high rate of outliers (but sometimes selective choice of random subsets can help)
- **A voting strategy**, The Hough transform, can handle high percentage of outliers

Skeleton Code

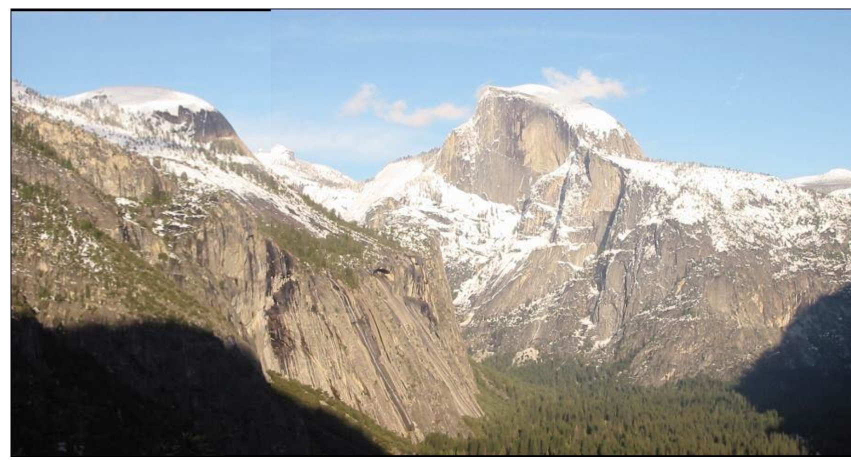
- RANSAC
  - ComputeError

$$\left\| \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} - H \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \right\|_2$$

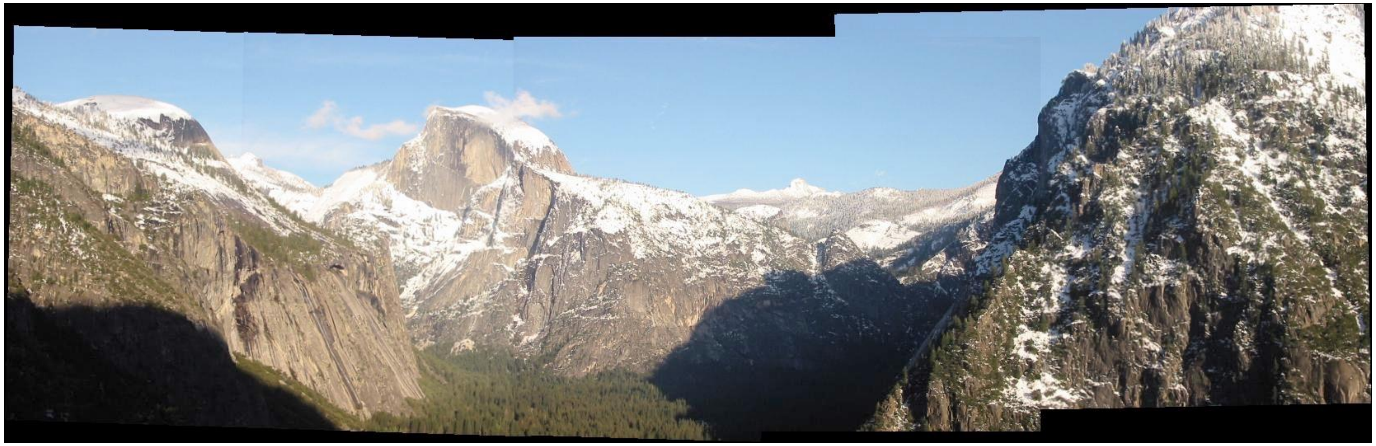
Main Flow



- Detect key points
- Build the SIFT descriptors
- Match SIFT descriptors
- Fitting the transformation
- RANSAC



Results



Results

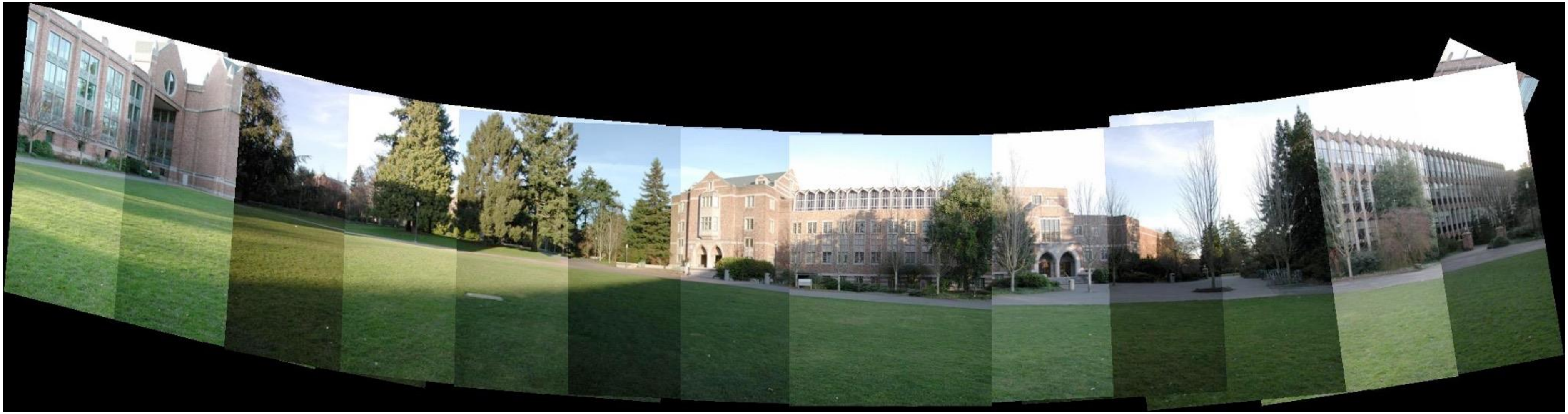
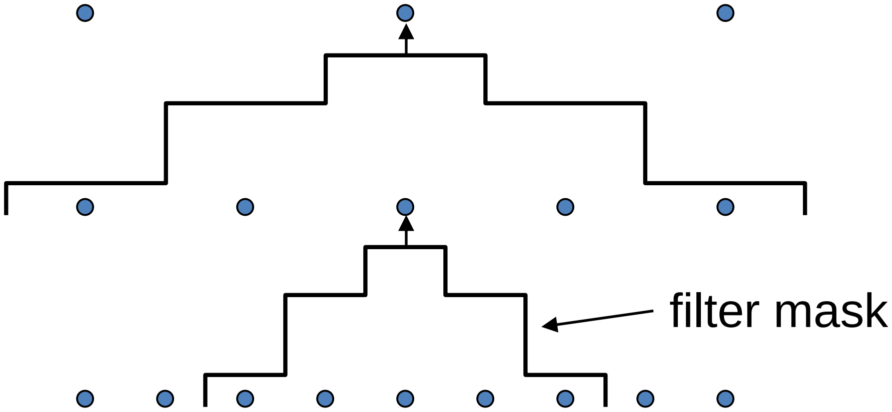
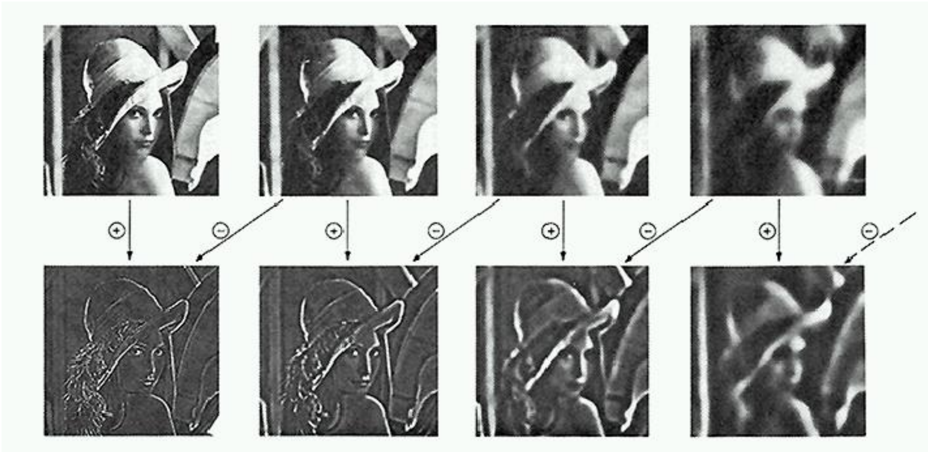


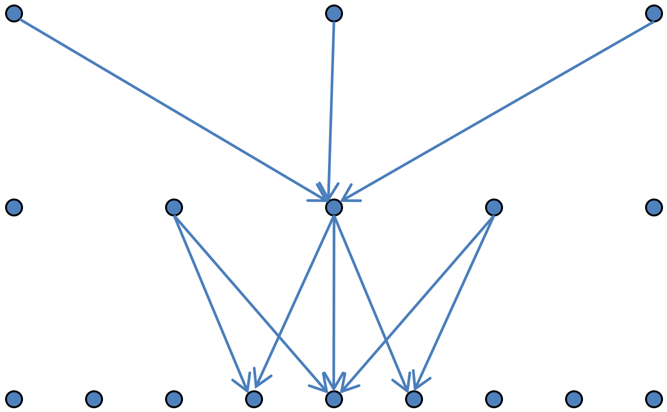
Image Blending

Pyramid Creation



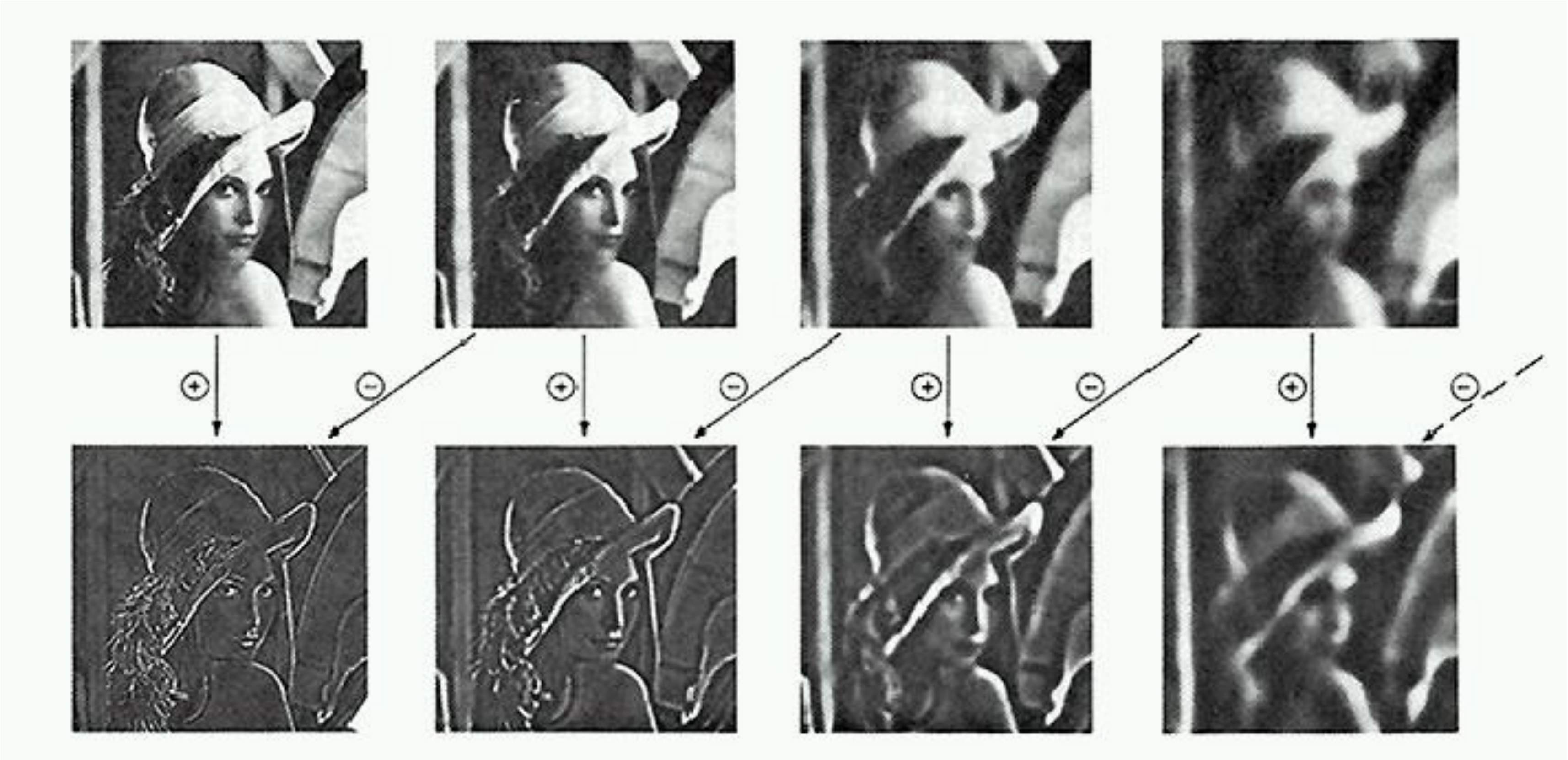


- "Gaussian" Pyramid
- "Laplacian" Pyramid
  - Created from Gaussian pyramid by subtraction
    $L_l = G_l - \text{expand}(G_{l+1})$



Octaves in the Spatial Domain

Lowpass Images

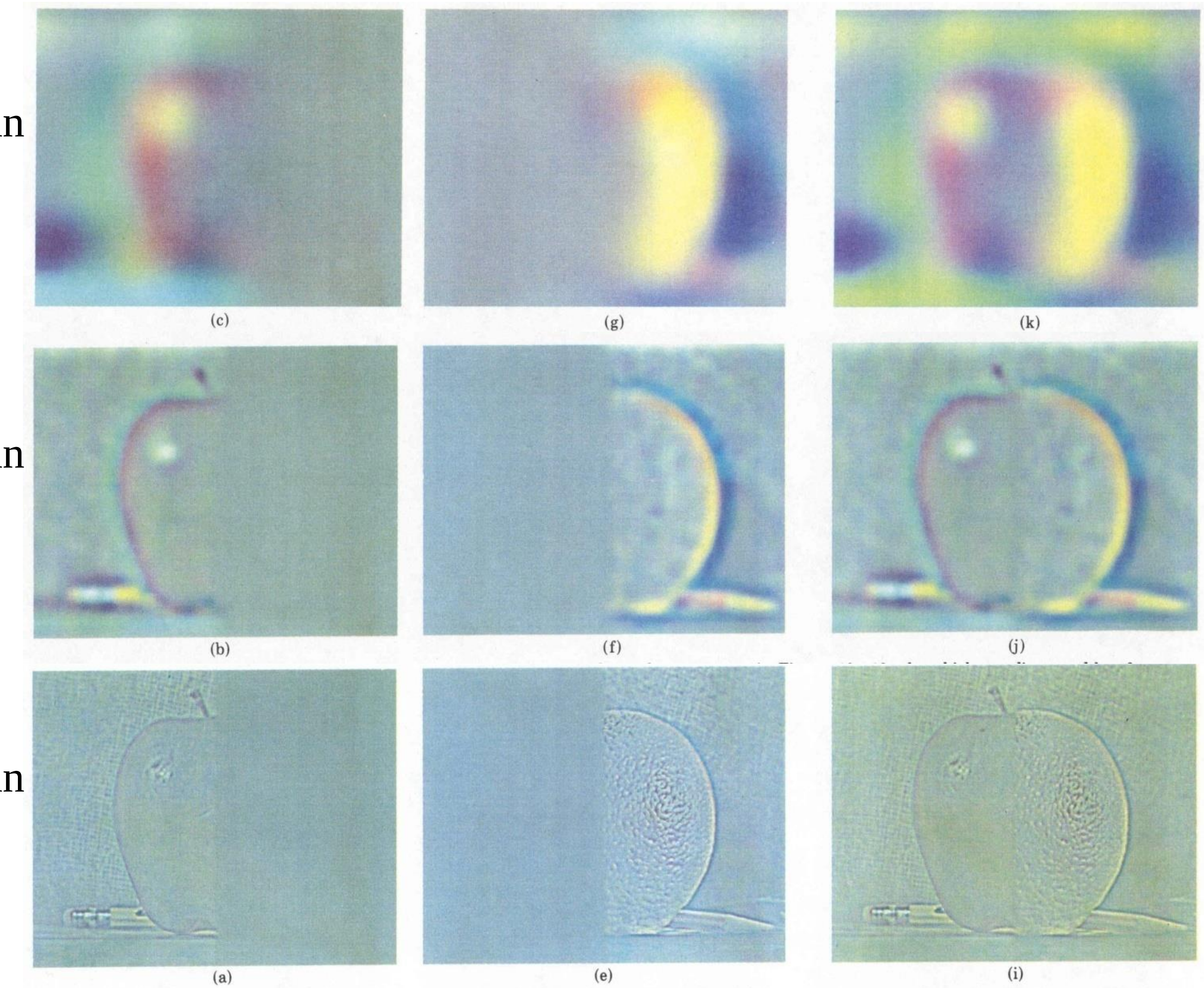


- Bandpass Images

Laplacian level 4

Laplacian level 2

Laplacian level 0

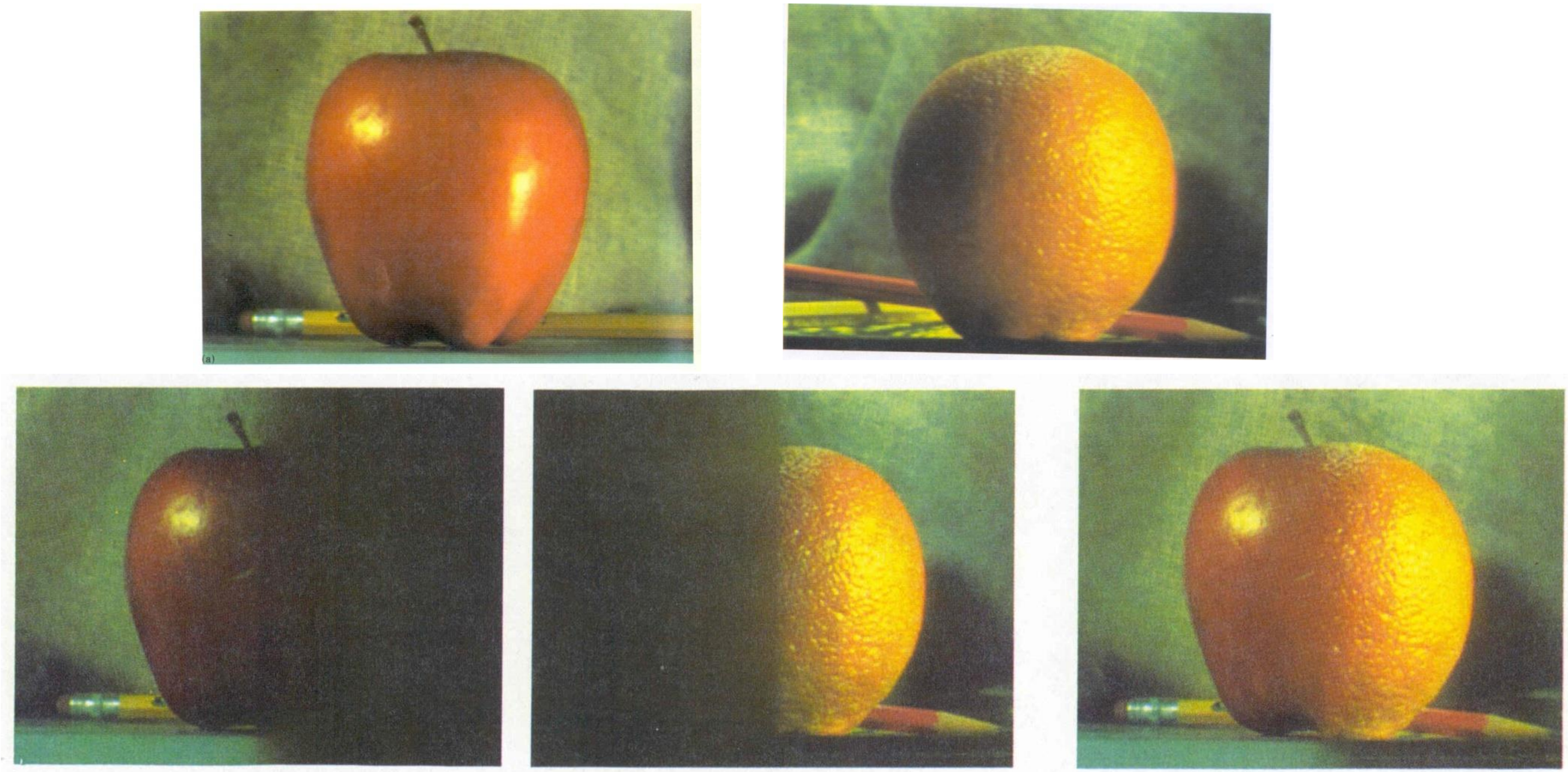


left pyramid

right pyramid

blended pyramid

Pyramid Blending



(d) (h) (l)

Main Flow

- Detect key points
- Build the SIFT descriptors
- Match SIFT descriptors
- Fitting the transformation
- RANSAC
- Image Blending