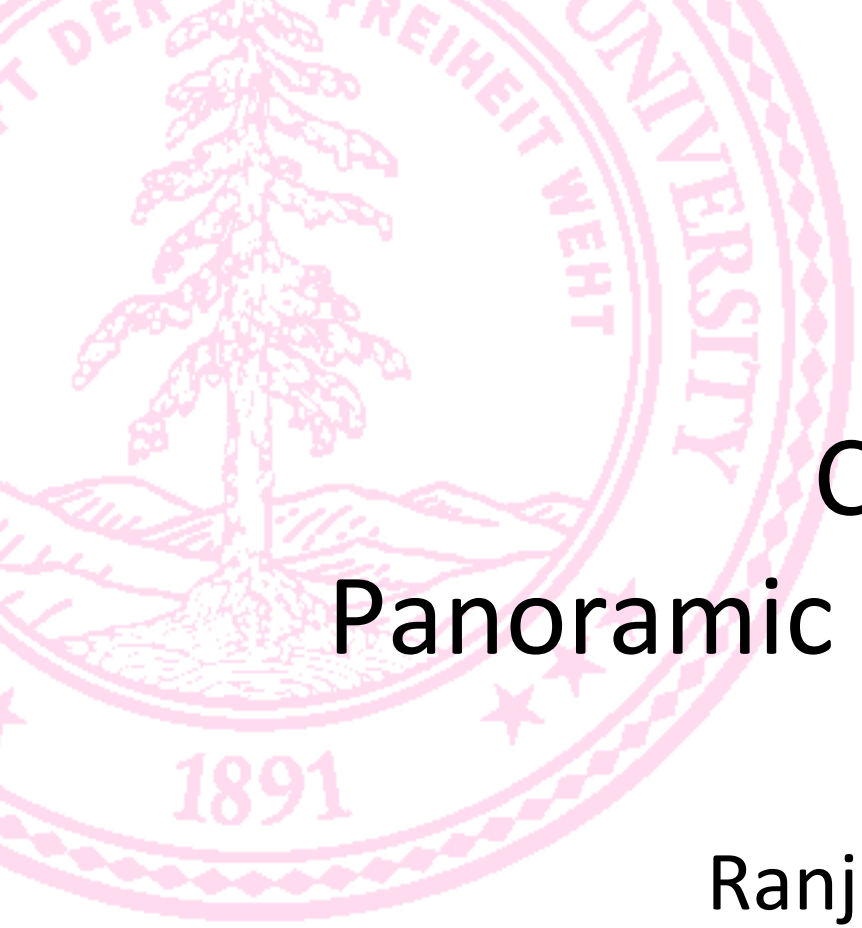


CS131
Panoramic Image Stitching

Ranjay Krishna

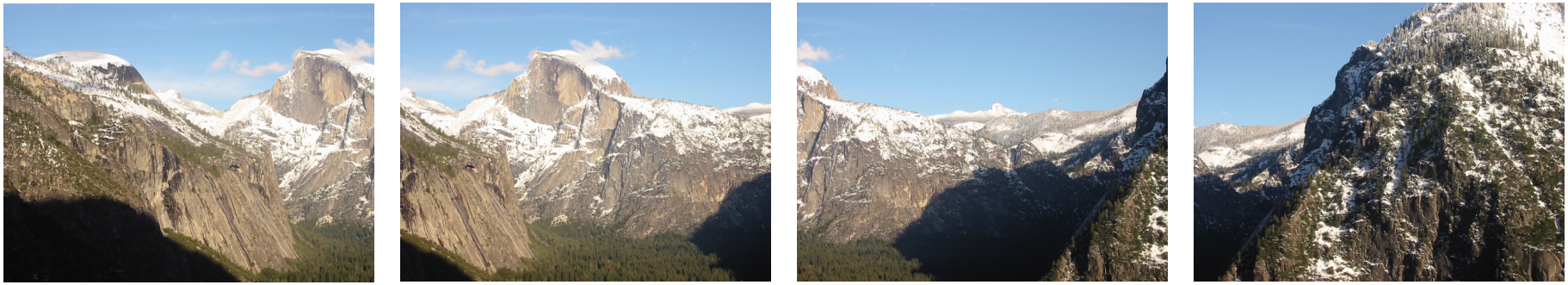
16-Oct-15

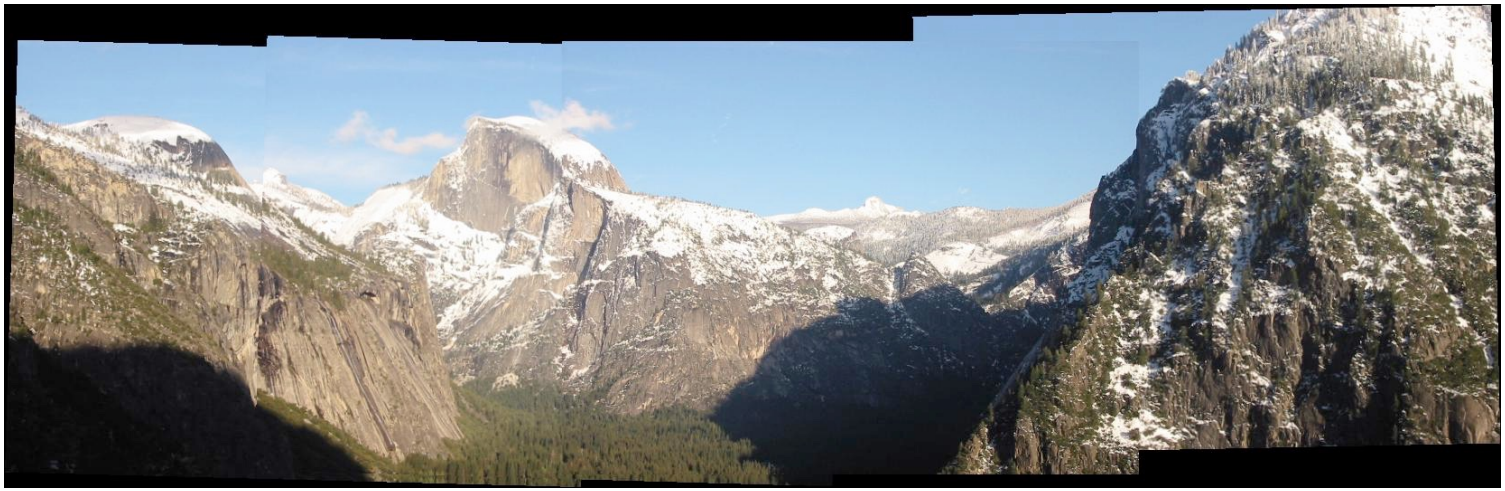
Agenda

- Objective
- Main flow
- Skeleton code
- Results

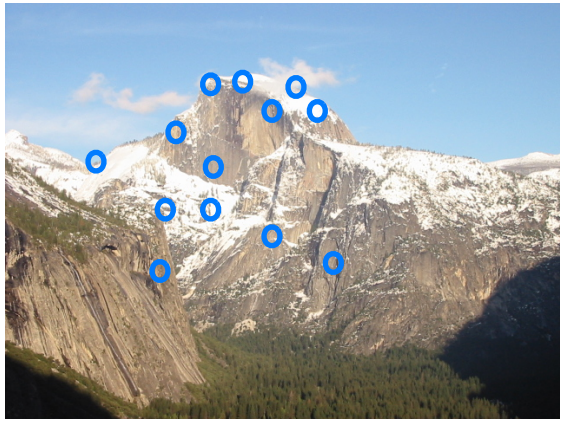
Objective

Multiple images into one panorama!



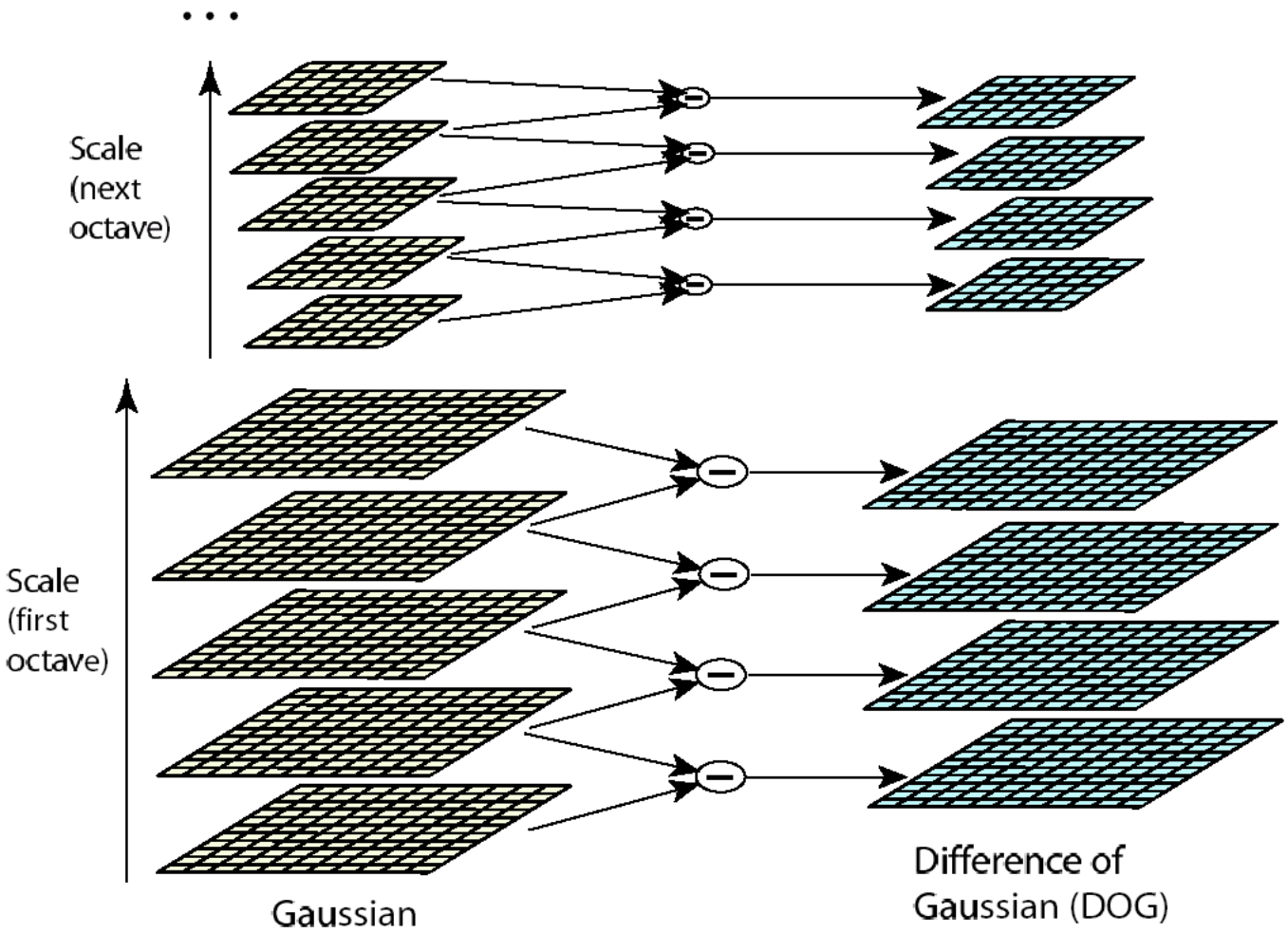


Main Flow



- Detect key points

Detect Key Points



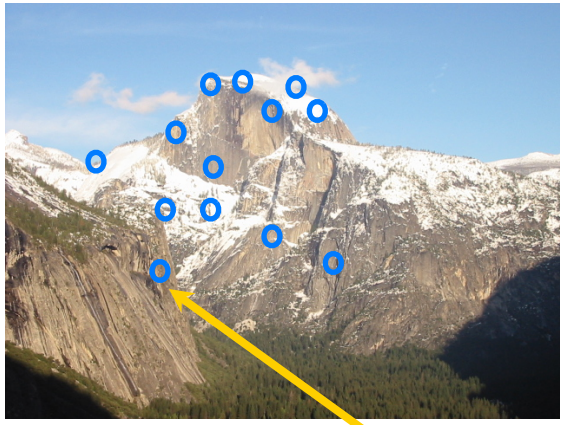
Skeleton Code

- Detect key points (Done for you!)
  - Under KeypointDetect

[feature, DoG pyr, Gaussian pyr] = detect_features(input image)

Main Flow

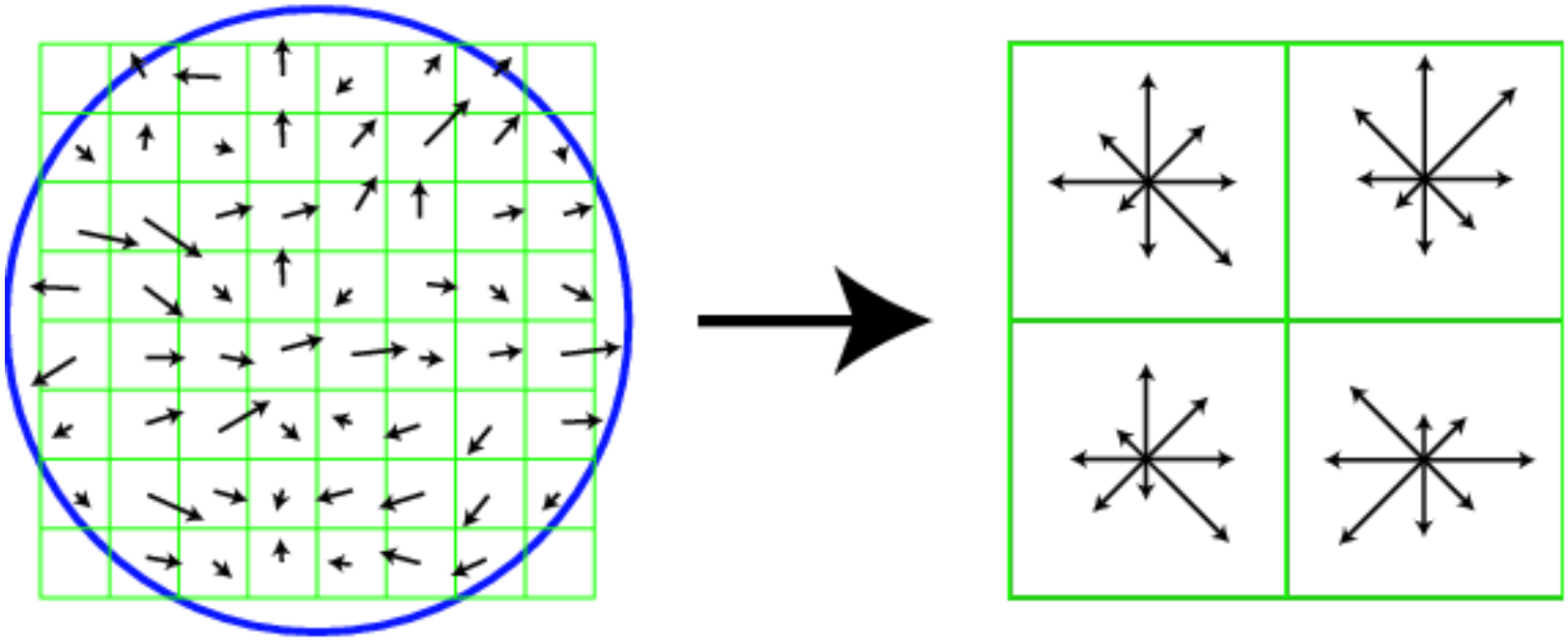
$(u_1, u_2, \ldots, u_{128})$



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

- Detect key points
- Build the SIFT descriptors

Build the SIFT Descriptors



This is just an illustration!

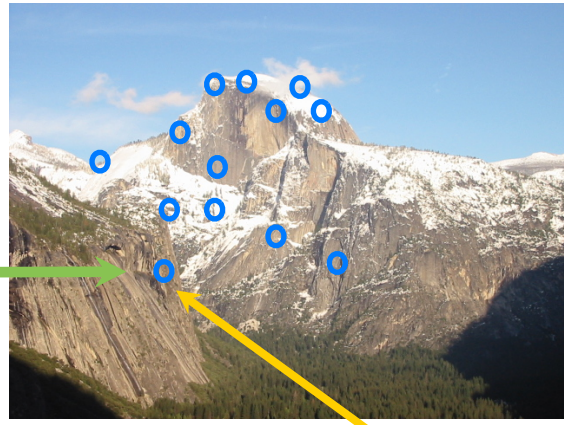
Skeleton Code

- Build the SIFT descriptors
  - Read this paper http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf first!
- Input
  - Gaussian pyramid
  - key point location
  - key point scale index
- Output
  - A set of 128-dim vectors

Skeleton Code

- Build the SIFT descriptors (30 lines of code)
  - Compute gradient magnitude and orientation
  - For each key point
    - Find a patch (tricky round-off)
    - Compute orientation of the patch
    - Build the histogram (edge case)

Main Flow



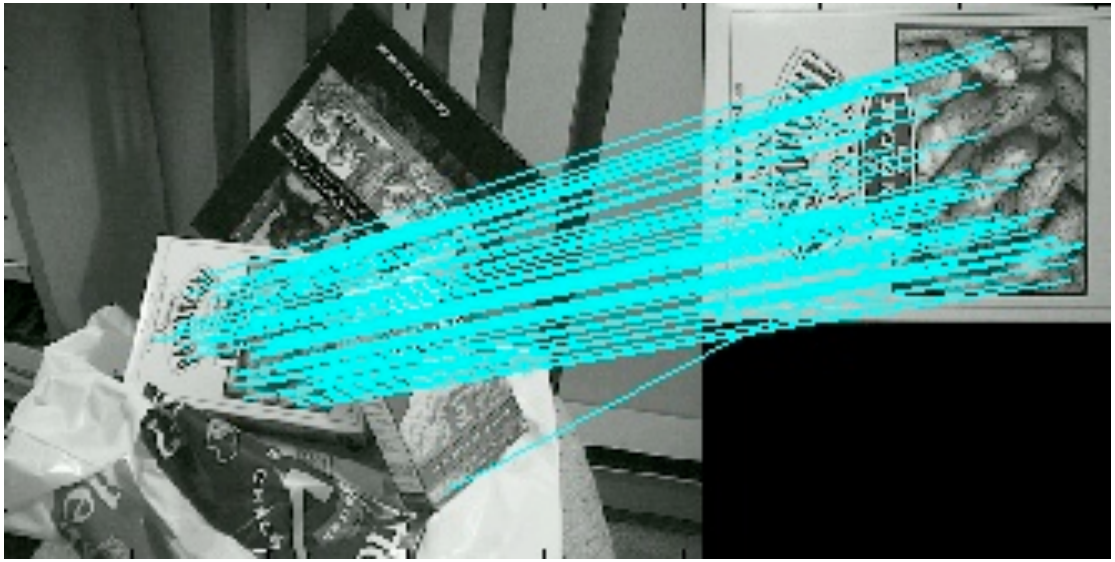
$(u_1, u_2, \ldots, u_{128})$

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

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

Match SIFT Descriptors

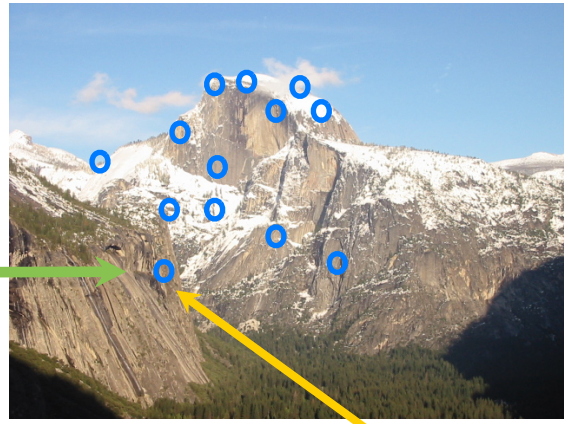
- Euclidean distance between descriptors



Skeleton Code

- Match SIFT descriptors (6 lines of code)
  - Input: D1, D2, thresh (default 0.7)
  - Output: match [D1's index, D2's index]
  - Try to use *one* for loop
  - Useful command
    - repmat
    - sort

Main Flow



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

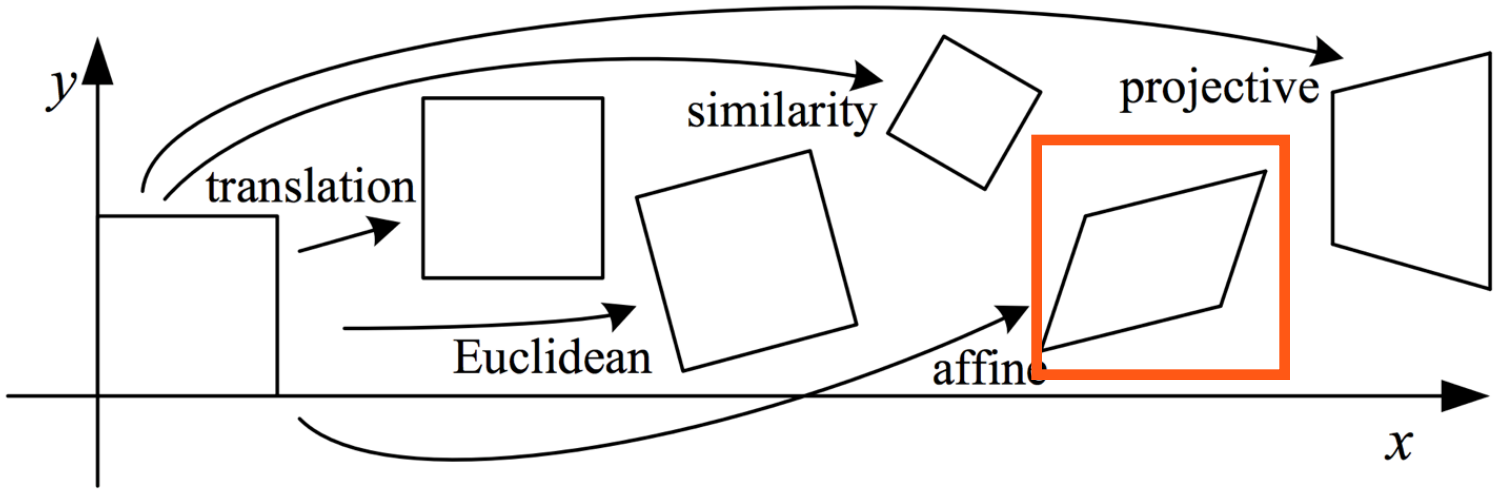
$(v_1, v_2, \ldots, v_{128})$

- 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



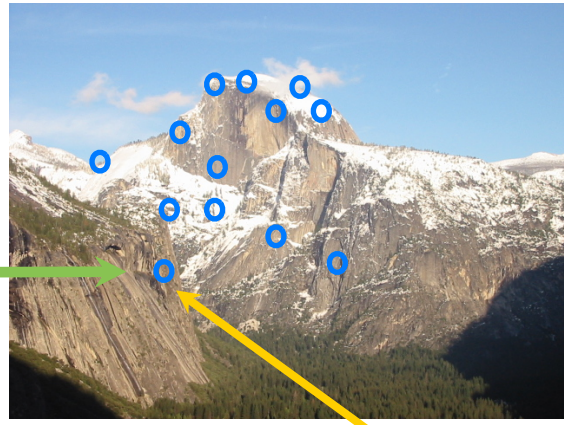
Skeleton Code

- 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



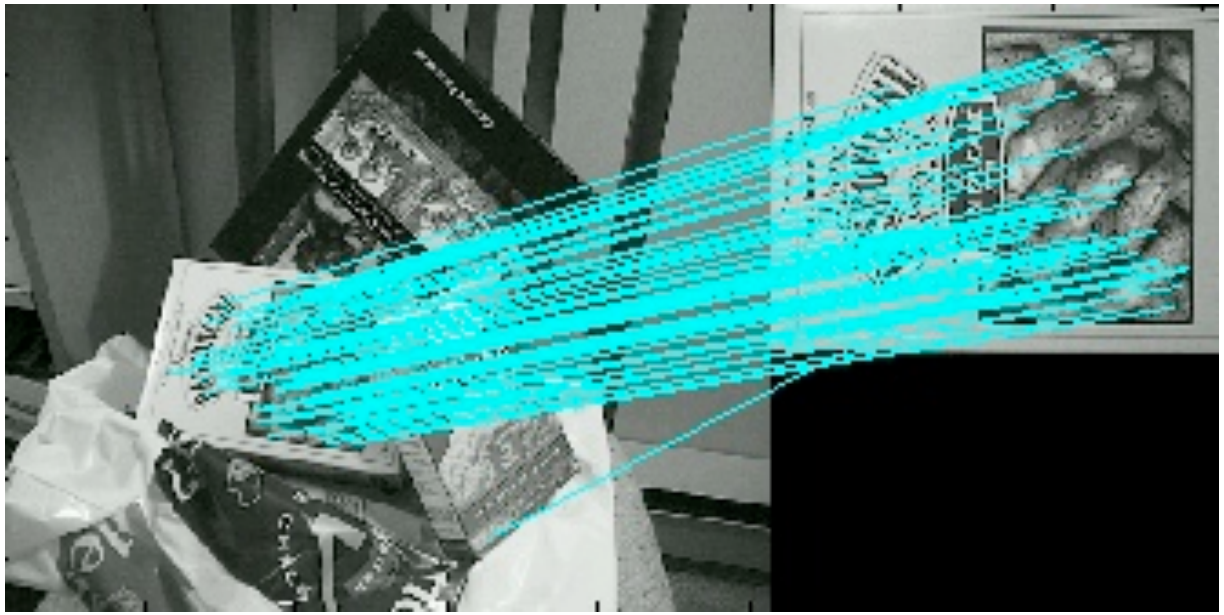
$(u_1, u_2, \ldots, u_{128})$

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

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

RANSAC

- A further refinement of matches

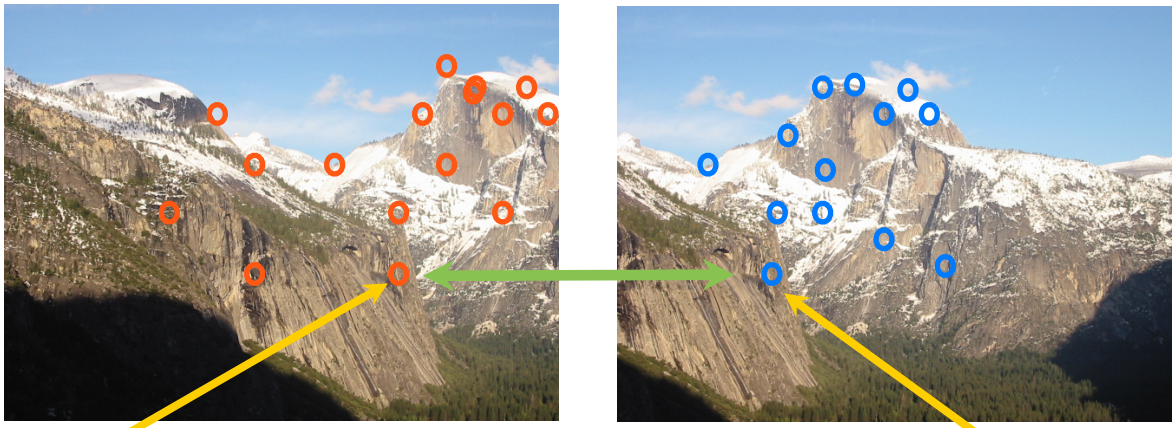


Skeleton Code

- 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



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

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

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

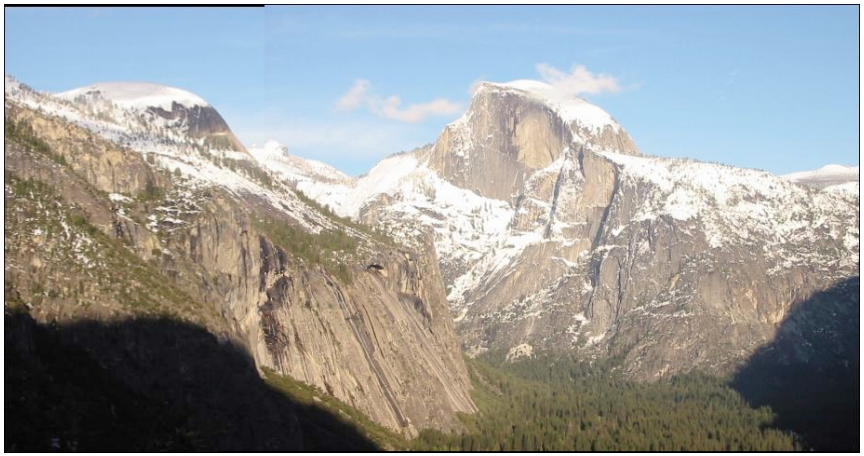


Image Stitching

- Almost done for you

Skeleton Code

- Multiple Stitch (2 lines of code)
  - A simplified case of real-world scenario
  - Transformation is associative and invertible
  - Useful command
    - pinv

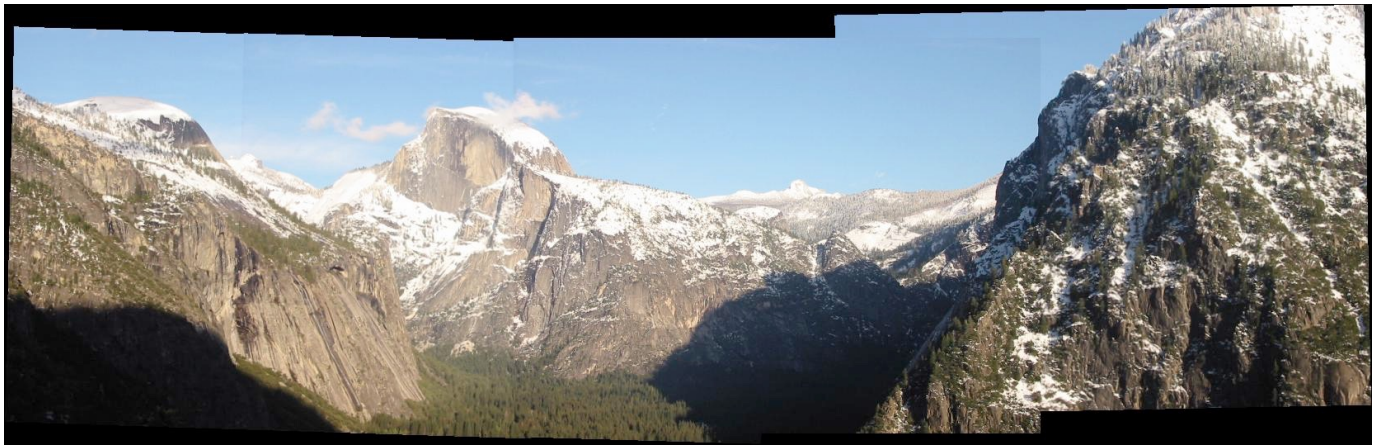
Skeleton Code

- Tester.m
  - Scripts that help you to get started
- Evaluate.m
  - Scripts that tests your solution
    - Load fixed input from checkpoint
    - Run your implementation
    - Compare results with reference solution

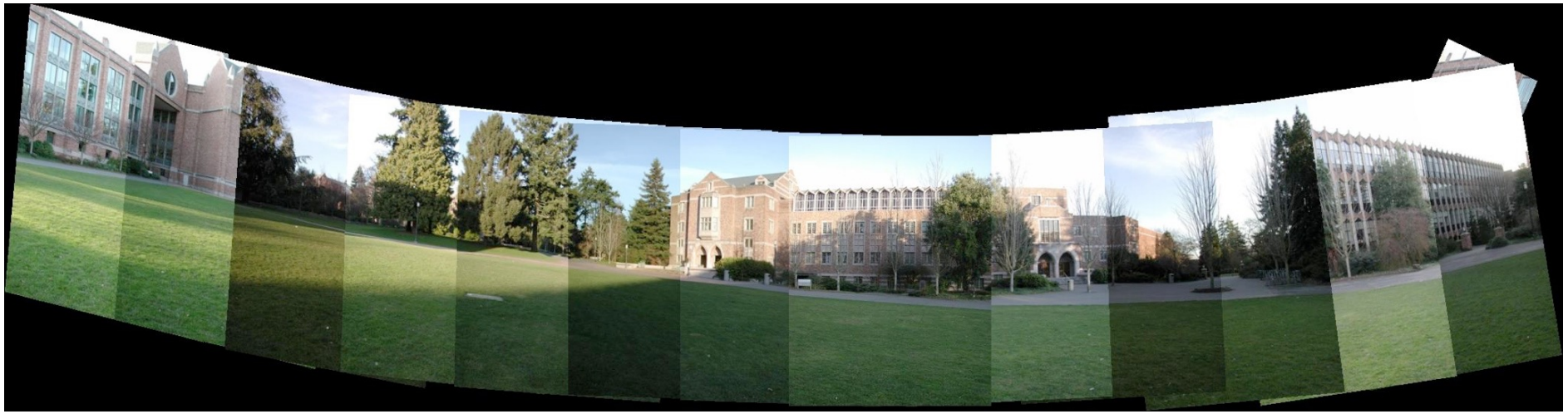
Requirement

- Due Date: 5pm Oct 30, 2015
- Electronic submission only
  - cs131submissions@gmail.com for code
  - Gradescope for report
- Code + Report
  - SIFT invariance and why it helps
  - DoG v.s. Dense SIFT
  - Why RANSAC
  - Your own stitches
  - Error discussion

Results



Results



Questions?