Image Stitching

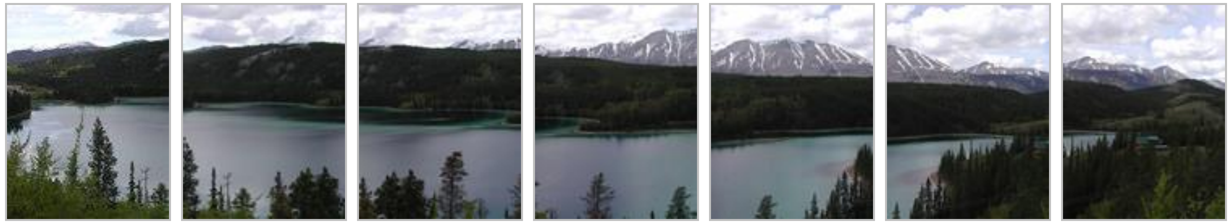
Jiayuan Gu

gujy1@shanghaitech.edu.cn

Many slides modified from Prof. Derek Hoiem CS543

Image Stitching

- Combine two or more overlapping images to make one larger image

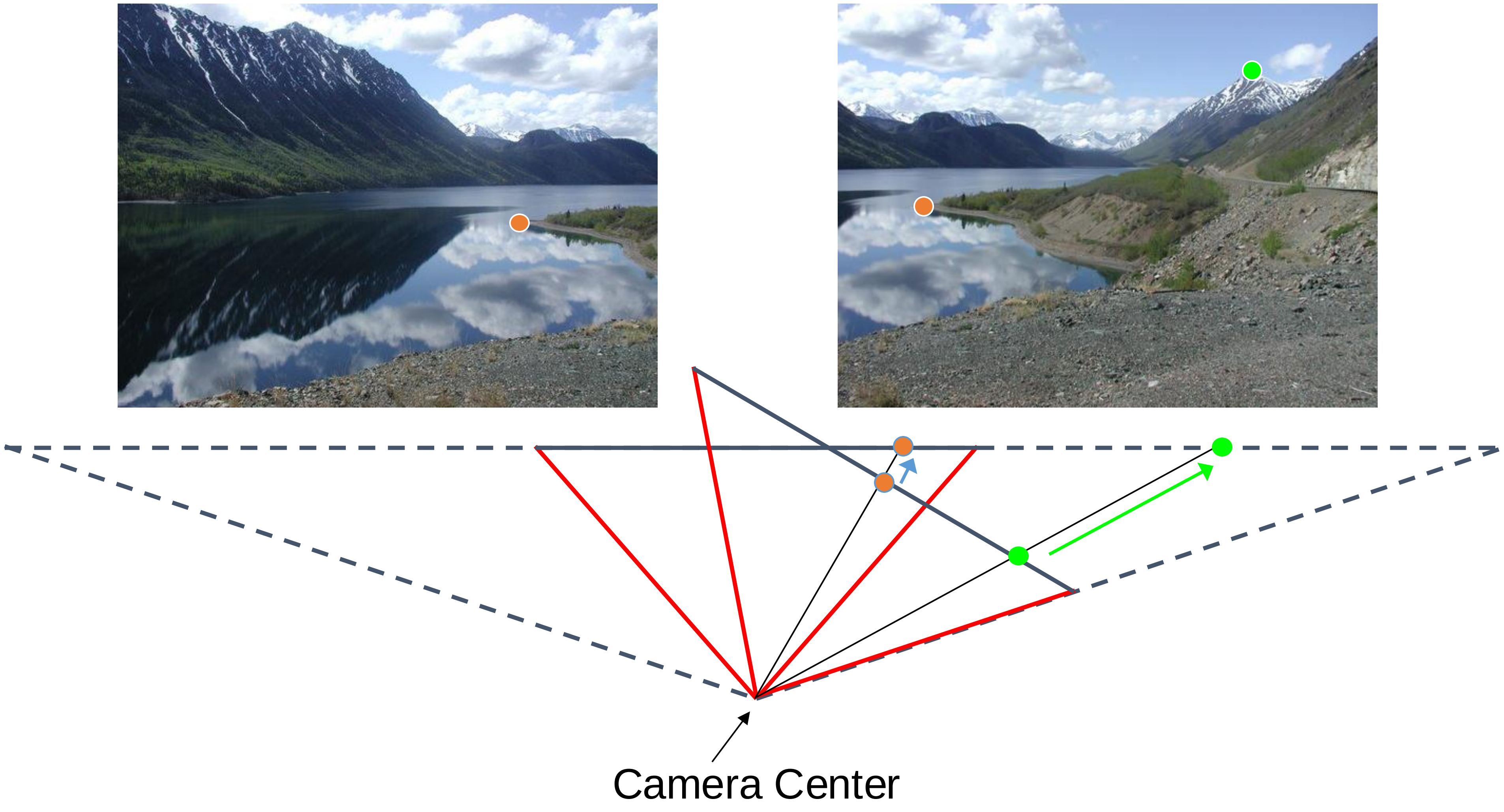


Slide credit: [Vaibhav Vaish]

Outline

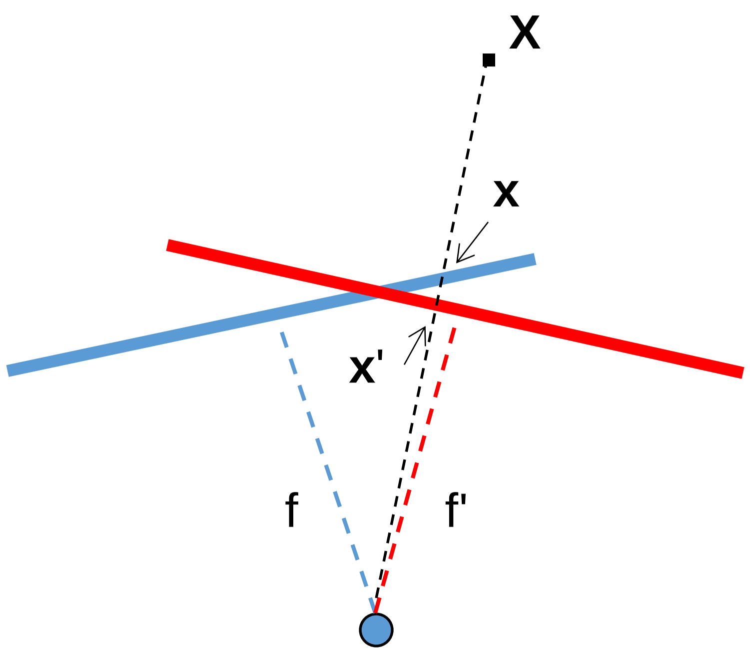
- Homography: math behind image stitching
- Solving homographies
- Pipeline
  - Keypoint detection
  - RANSAC
  - Blending

Illustration



Problem Statement of Image Stitching

- camera 1: $x = K[R\ t]X$
- camera 2: $x' = K'[R't']X$
- no translation: $t = t' = 0$
- Image stitching is to compute x given x'
  - $x' = K'R'R^{-1}K^{-1}x \stackrel{\text{def}}{=} Hx$
- Typically, only R and f change, but in general H has 8 DoF



Homography

- 2D projective geometry is the study of properties of the projective plane $\mathbb{P}^2$ that are invariant under a group of transformations known as ***projectivities***

- A *projectivity* is an invertible mapping from points in $\mathbb{P}^2$ (that is homogeneous 3-vectors) to points in $\mathbb{P}^2$ that maps lines to lines

- A *projectivity* is also called a *collineation*, a *projective transformation* or a *homography*

Projective Transformation (Homography)

Definition 2.11. Projective transformation. A planar projective transformation is a linear transformation on homogeneous 3-vectors represented by a non-singular 3×3 matrix:

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \tag{2.5}$$

or more briefly, $\mathbf{x}' = \mathtt{H}\mathbf{x}$.

Briefly, the planar homography relates the transformation between two planes (up to a scale factor)

Example of Homography

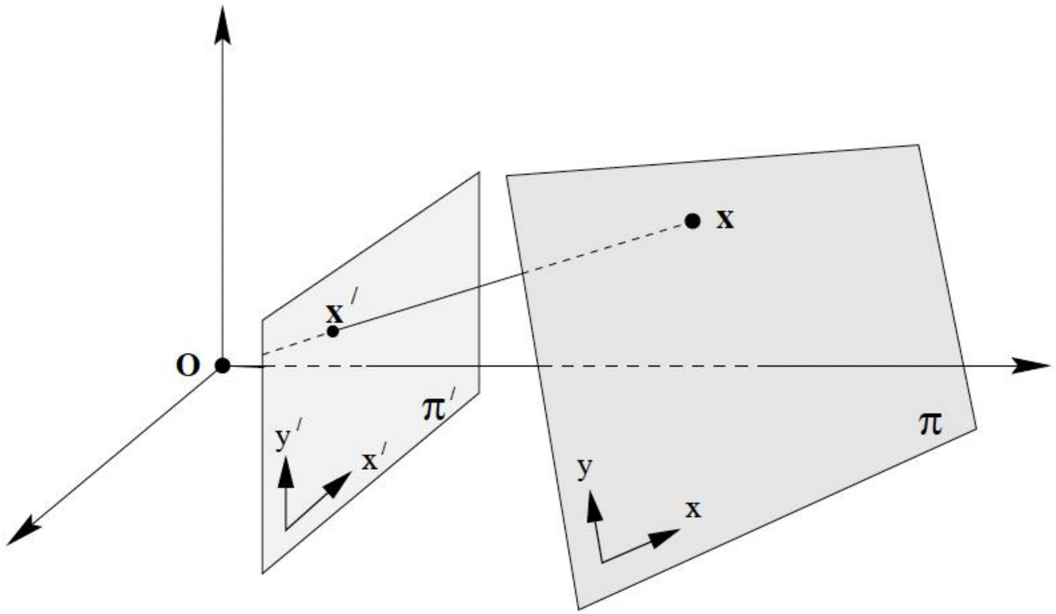
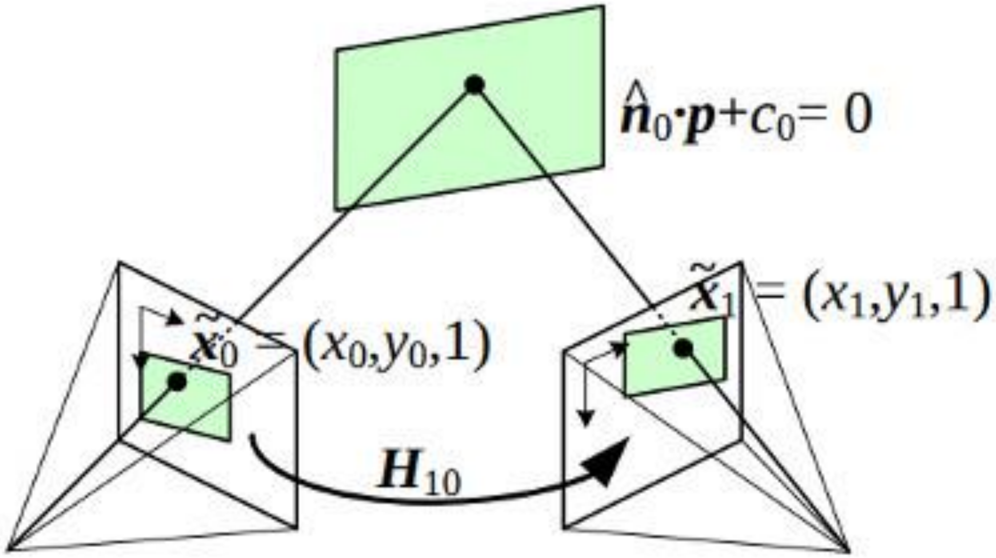
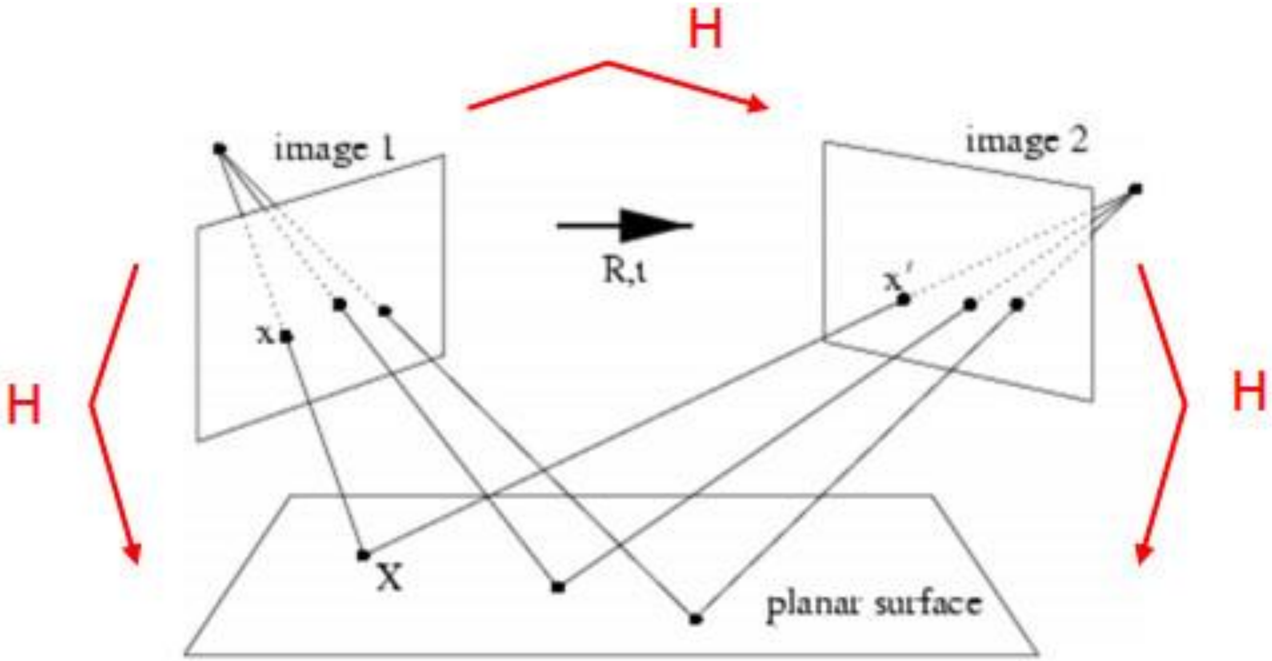


Fig. 2.3. **Central projection maps points on one plane to points on another plane.** *The projection also maps lines to lines as may be seen by considering a plane through the projection centre which intersects with the two planes* π *and* π'*. Since lines are mapped to lines, central projection is a projectivity and may be represented by a linear mapping of homogeneous coordinates* $\mathbf{x}' = \mathtt{H}\mathbf{x}$.

e.g., a planar surface and the image plane

Example of Homography

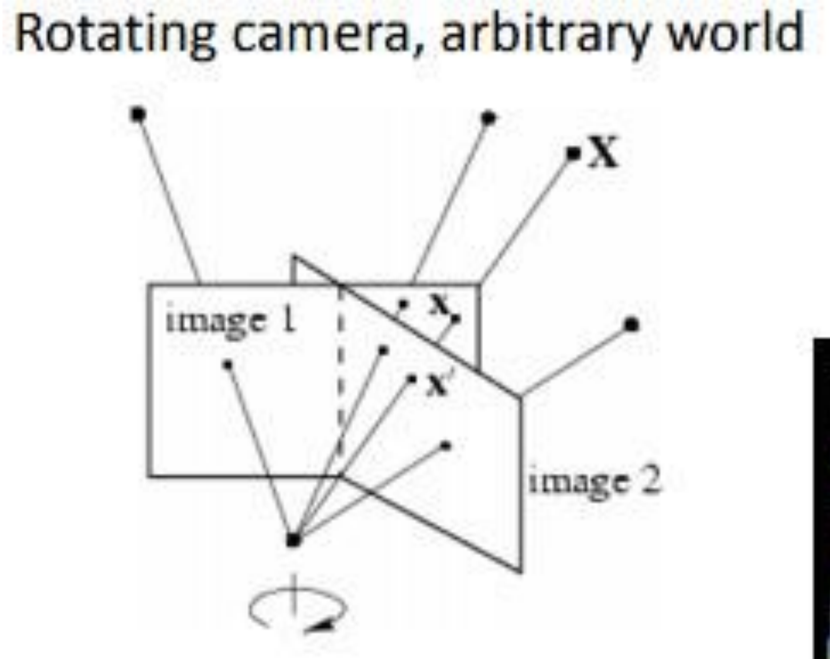


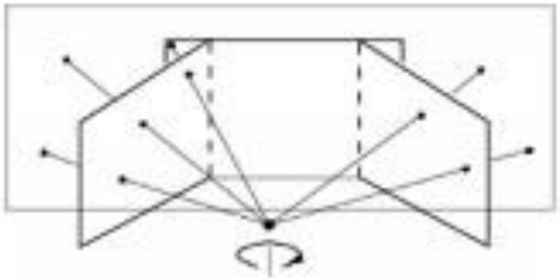


e.g., a planar surface viewed by two camera positions

Figure from OpenCV tutorial

Example of Homography



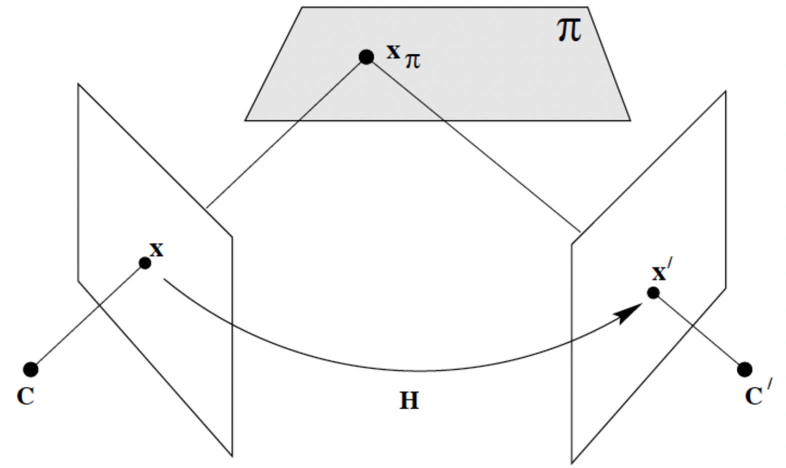


e.g., a rotating camera around its axis of projection,
equivalent to consider that the points are on a plane at infinity

Figure from OpenCV tutorial

Homography vs. Epipolar Geometry

- Epipolar constraint: $x'^T F x = 0$
- Homography: $\mathrm{x}' = Hx$
- Questions:
  - How is homography related to 3D scene points?
    - 3D scene points need to be on a planar surface
  - Why can we analyze non-planar surface in the setting of rotating cameras?
    - Only the direction affects the pixel coordinates

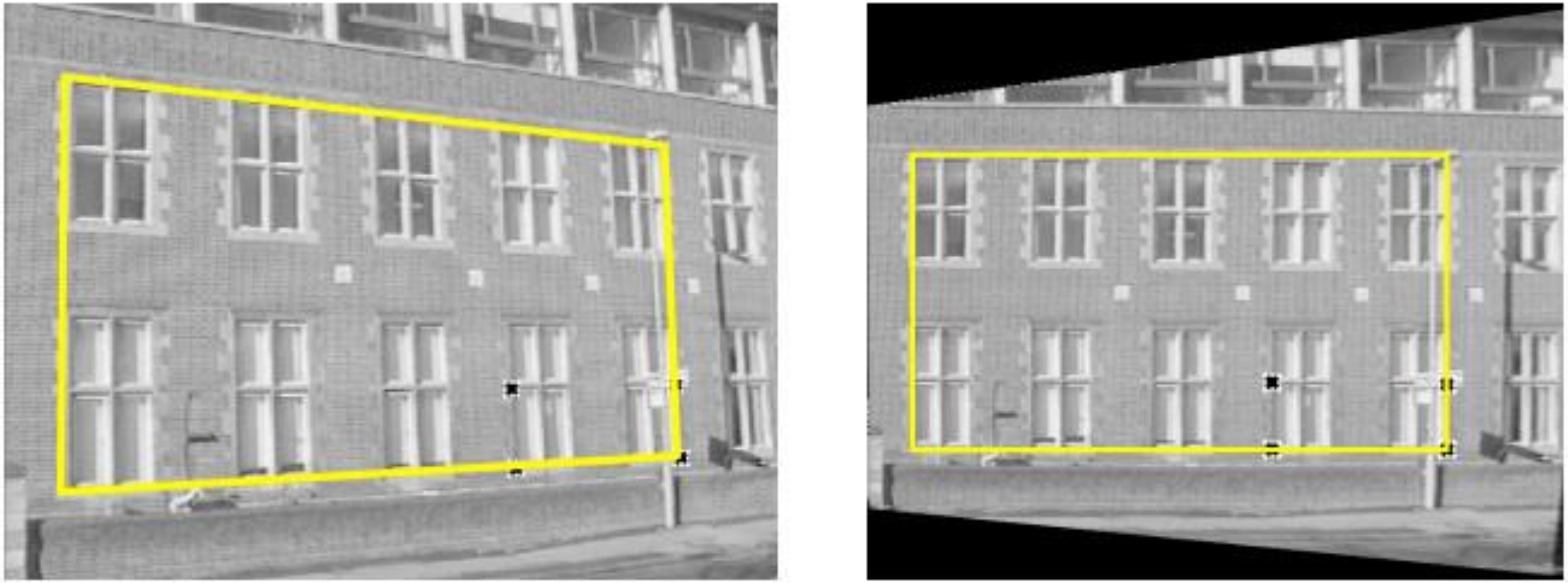


Application of Homography

Camera pose estimation from coplanar points for augmented reality with marker for instance

Figure from OpenCV tutorial

Application of Homography



from Hartley & Zisserman

Removing perspective distortion

Application of Homography

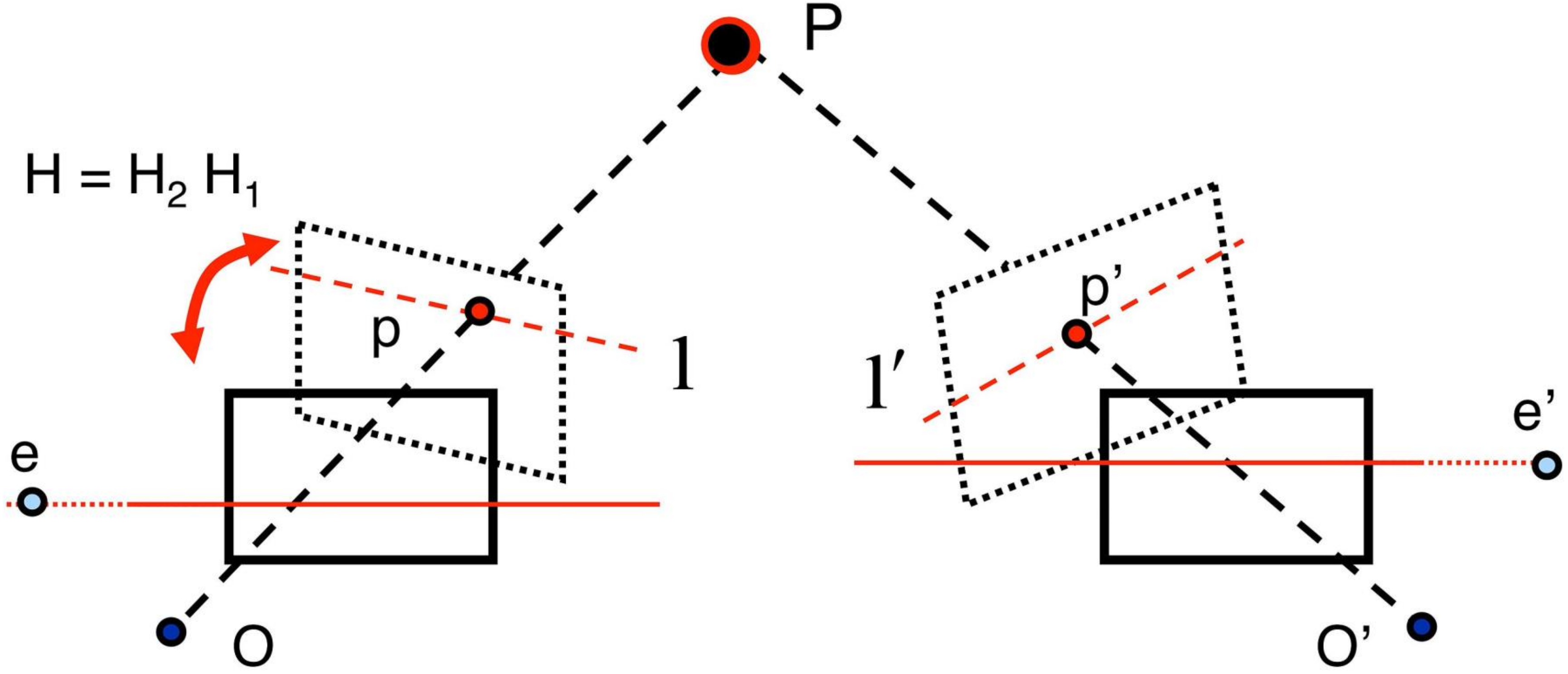
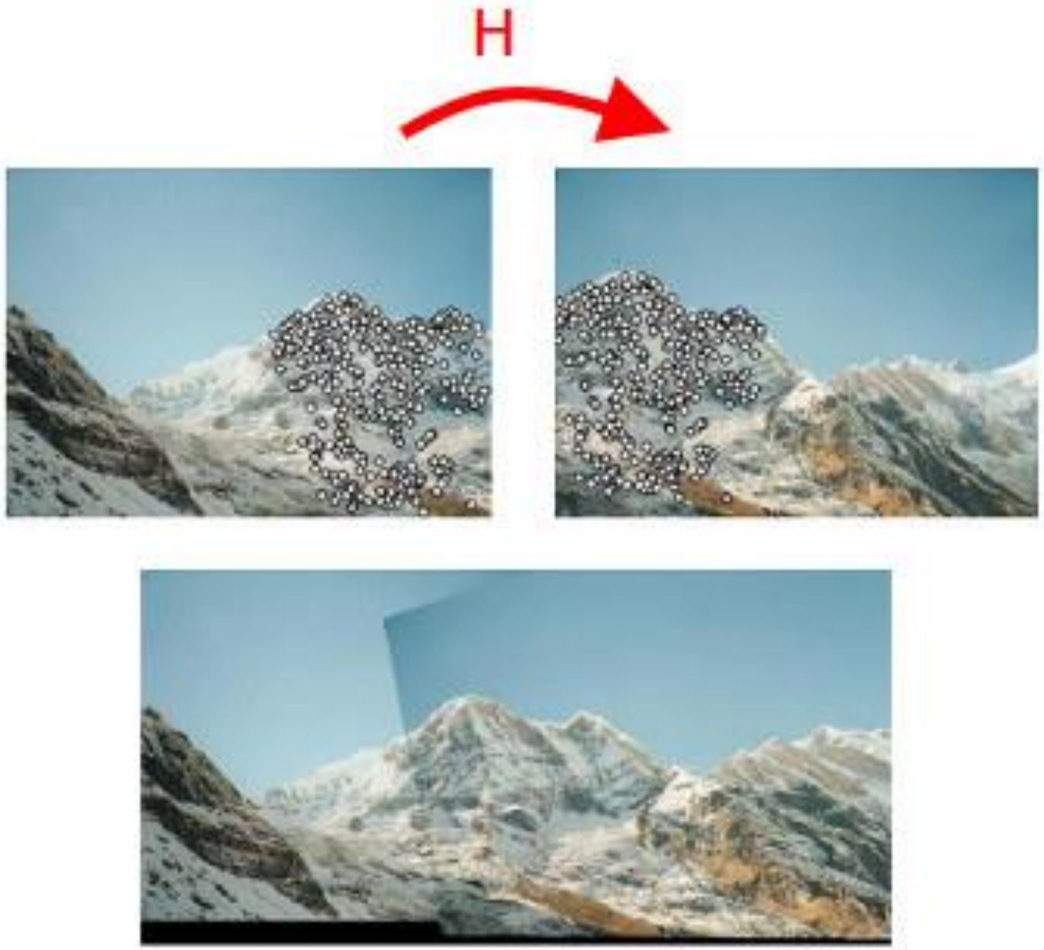


Image rectification: converged camera to parallel camera

Application of Homography



Panorama stitching

Figure from OpenCV tutorial

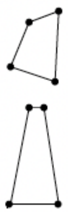
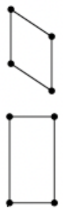
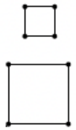
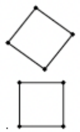
More about Projectivity

Theorem 2.10. *A mapping* $h : \mathbb{P}^2 \rightarrow \mathbb{P}^2$ *is a projectivity if and only if there exists a non-singular* 3×3 *matrix* $\mathtt{H}$ *such that for any point in* $\mathbb{P}^2$ *represented by a vector* $\mathbf{x}$ *it is true that* $h(\mathbf{x}) = \mathtt{H}\mathbf{x}$.

Read by yourself

Multiple View Geometry, Chapter 2.3

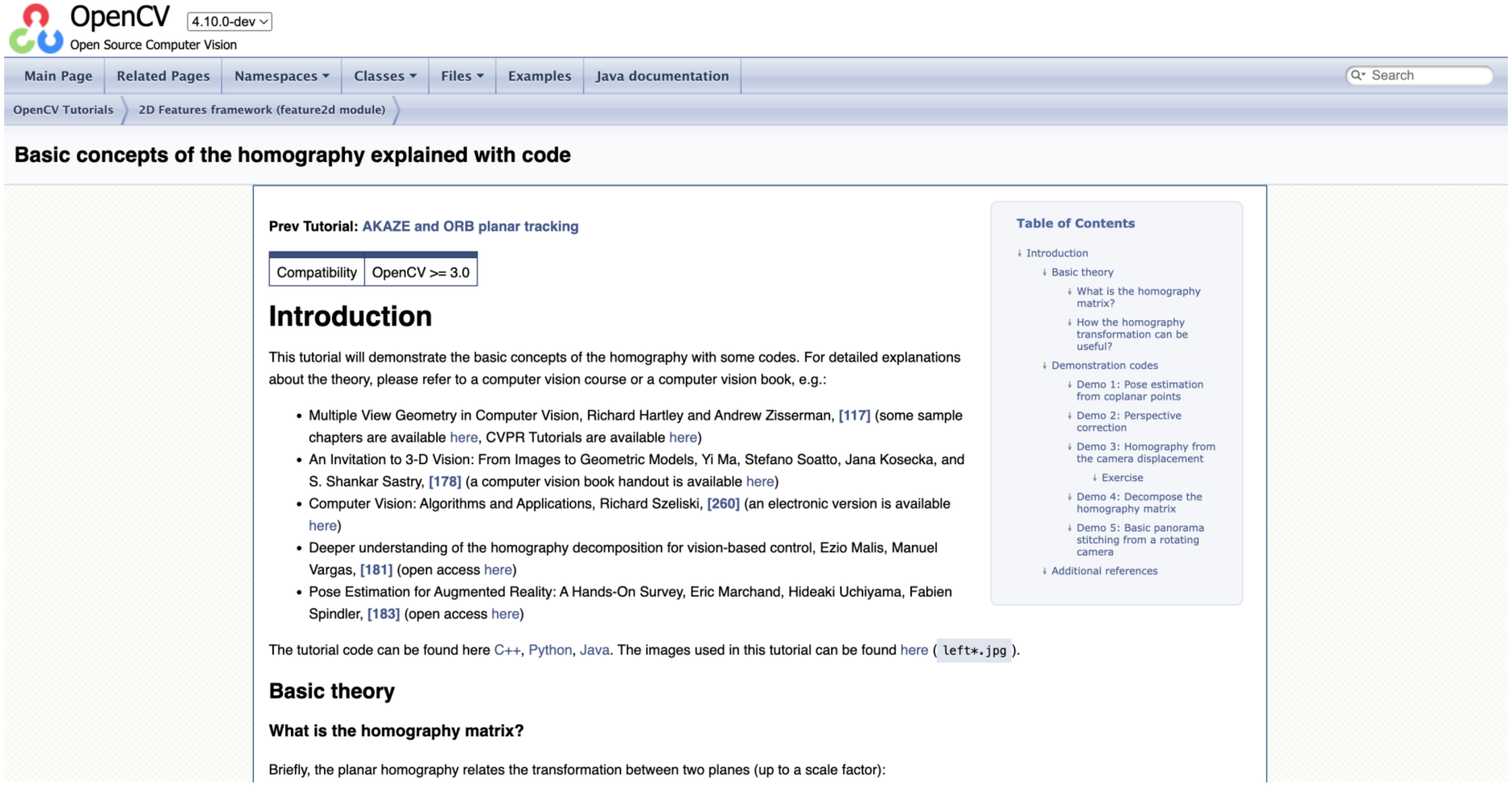
Different Types of 2D Transformation

Group	Matrix	Distortion	Invariant properties
Projective 8 dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		Concurrency, collinearity, **order of contact**: intersection (1 pt contact); tangency (2 pt contact); inflections (3 pt contact with line); tangent discontinuities and cusps. cross ratio (ratio of ratio of lengths).
Affine 6 dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Parallelism, ratio of areas, ratio of lengths on collinear or parallel lines (e.g. midpoints), linear combinations of vectors (e.g. centroids). The line at infinity, $\mathbf{l}_\infty$.
Similarity 4 dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Ratio of lengths, angle. The circular points, **I**, **J** (see section 2.7.3).
Euclidean 3 dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Length, area

Read by yourself

Multiple View Geometry, Chapter 2.4

Basic Concepts of Homography



OpenCV 4.10.0-dev
Open Source Computer Vision

Main Page | Related Pages | Namespaces | Classes | Files | Examples | Java documentation | Search

OpenCV Tutorials > 2D Features framework (feature2d module)

Basic concepts of the homography explained with code

Prev Tutorial: AKAZE and ORB planar tracking

Compatibility	OpenCV >= 3.0

Introduction

This tutorial will demonstrate the basic concepts of the homography with some codes. For detailed explanations about the theory, please refer to a computer vision course or a computer vision book, e.g.:

- Multiple View Geometry in Computer Vision, Richard Hartley and Andrew Zisserman, [117] (some sample chapters are available here, CVPR Tutorials are available here)
- An Invitation to 3-D Vision: From Images to Geometric Models, Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry, [178] (a computer vision book handout is available here)
- Computer Vision: Algorithms and Applications, Richard Szeliski, [260] (an electronic version is available here)
- Deeper understanding of the homography decomposition for vision-based control, Ezio Malis, Manuel Vargas, [181] (open access here)
- Pose Estimation for Augmented Reality: A Hands-On Survey, Eric Marchand, Hideaki Uchiyama, Fabien Spindler, [183] (open access here)

The tutorial code can be found here C++, Python, Java. The images used in this tutorial can be found here (`left*.jpg`).

Basic theory

What is the homography matrix?

Briefly, the planar homography relates the transformation between two planes (up to a scale factor):

Table of Contents

- Introduction
  - Basic theory
    - What is the homography matrix?
    - How the homography transformation can be useful?
  - Demonstration codes
    - Demo 1: Pose estimation from coplanar points
    - Demo 2: Perspective correction
    - Demo 3: Homography from the camera displacement
      - Exercise
    - Demo 4: Decompose the homography matrix
    - Demo 5: Basic panorama stitching from a rotating camera
  - Additional references

Read by yourself

https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html

Solving Homography

- Given several matched points in two images, how shall we compute homography H

- Problem statement
  - We observe x, x'
  - We know $\mathrm{x}' = Hx$
  - We solve H

- Is this problem similar to other problems we have learned?

Solving Homography with DLT

- Direct Linear Transformation (DLT)

$$\mathbf{x}' = \mathbf{H}\mathbf{x} \qquad \mathbf{x}' = \begin{bmatrix} w'u' \\ w'v' \\ w' \end{bmatrix} \qquad \mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

$$\begin{bmatrix} -u & -v & -1 & 0 & 0 & 0 & uu' & vu' & u' \\ 0 & 0 & 0 & -u & -v & -1 & uv' & vv' & v' \end{bmatrix} \mathbf{h} = \mathbf{0} \qquad \mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix}$$

4 points are needed

Solving Homography with DLT

$$\begin{bmatrix} -u_1 & -v_1 & -1 & 0 & 0 & 0 & u_1u_1' & v_1u_1' & u_1' \\ 0 & 0 & 0 & -u_1 & -v_1 & -1 & u_1v_1' & v_1v_1' & v_1' \\ & & & & \vdots & & & & \\ 0 & 0 & 0 & -u_n & -v_n & -1 & u_nv_n' & v_nv_n' & v_n' \end{bmatrix} \mathbf{h} = \mathbf{0} \Rightarrow \mathbf{Ah} = \mathbf{0}$$

h is the eigenvector corresponding to the smallest eigenvalue of $A^{\mathrm{T}}\mathrm{A}$ (equivalent to using SVD)

Recall: Least Square Solution

$$\min_{x} x^T A^T A x \text{, s.t.} \left|\left|\mathrm{x}\right|\right| = 1$$

Least square objective

$$L(x, \lambda) = x^T A^T A x - \lambda(x^T x - 1)$$

Lagrange multiplier

$$\frac{\partial L}{\partial x} = 2A^T A x - 2\lambda x = 0$$

$$A^T A x = \lambda x$$

x is the eigenvector with the smallest eigenvalue of $A^T A$

Recall: Lagrange Multiplier

The basic idea is to convert a constrained problem into a form such that the derivative test of an unconstrained problem can still be applied.

$$\min_{x} f(x), s.t.\ g(x) = 0$$

constrained

$$\Downarrow$$

$$\mathcal{L}(x, \lambda) \triangleq f(x) + \lambda g(x)$$

unconstrained

$$\frac{\partial \mathcal{L}(x, \lambda)}{\partial x} = 0, \frac{\partial \mathcal{L}(x, \lambda)}{\partial \lambda} = 0$$

Refer to wiki

Normalized DLT

- Normalize coordinates for each image
  - $\tilde{x} = Tx, \tilde{x}' = T'x'$

- Compute $\widetilde{H}$ using DLT in normalized coordinates

- Denormalize: $H = T'^{-1}\widetilde{H}T$

Overview of Image Stitching

1. Detect keypoints in each image(e.g., SIFT)

2. Match keypoints of two images

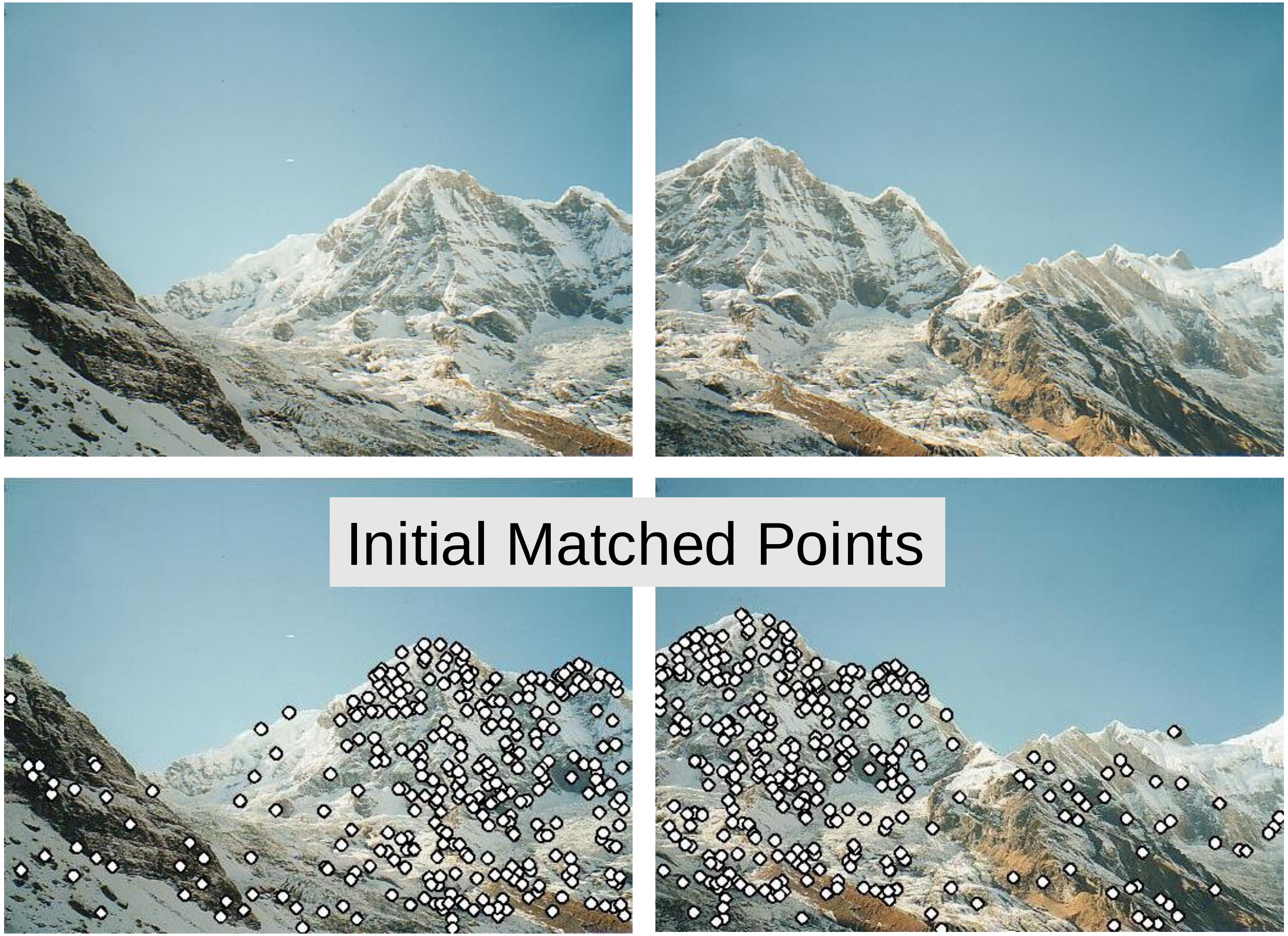
3. Estimate homography with 4 matched keypoints using RANSAC

4. Combine images

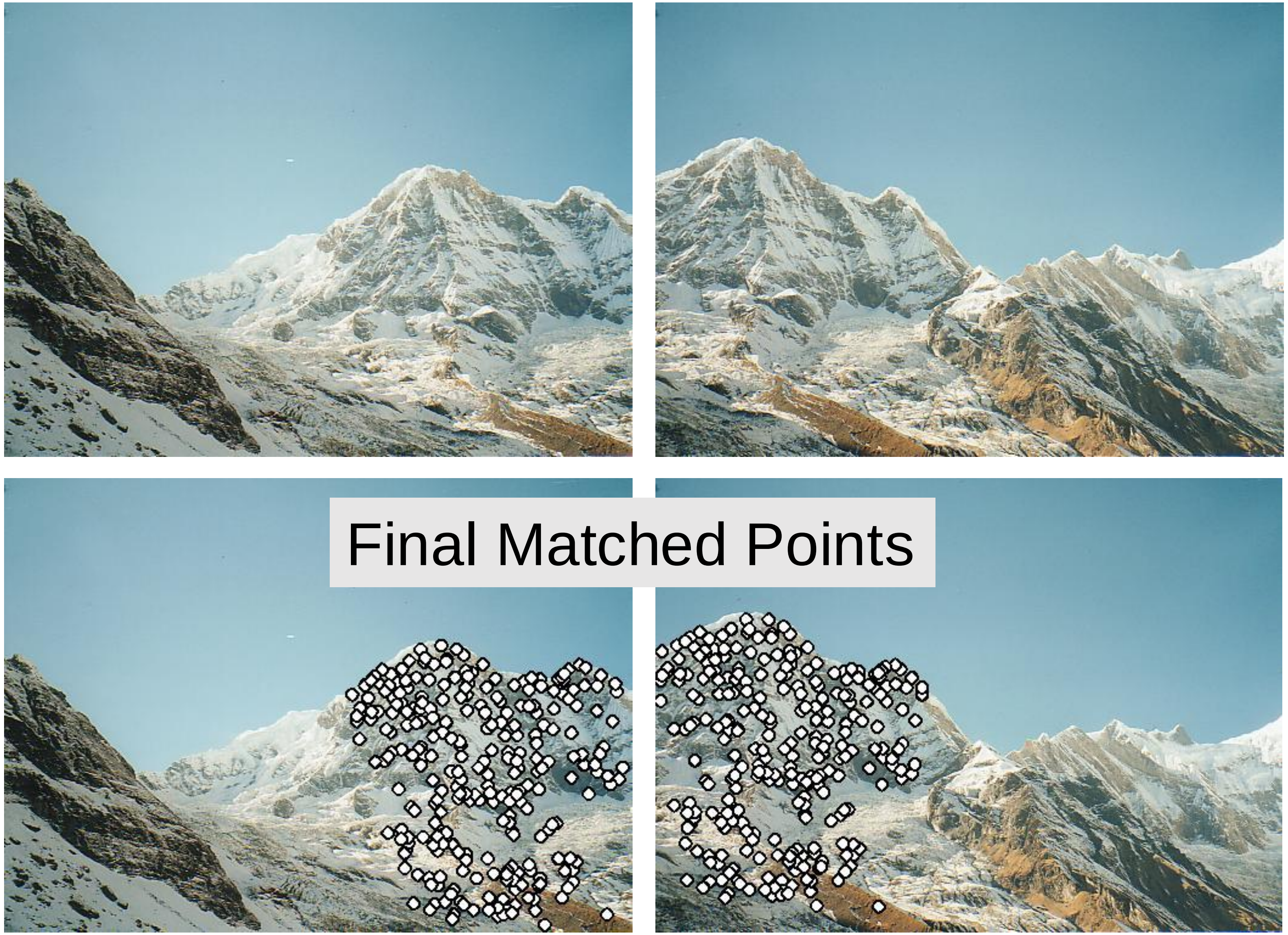
RANSAC for Homography Estimation

1. Choose number of samples *N*
2. Choose 4 random potential matches
3. Compute ***H*** using normalized DLT
4. Project points from ***x*** to ***x'*** for each potentially matching pair
5. Count points with projected distance < t (e.g., t=3 pixels)
6. Repeat steps 2-5 *N* times

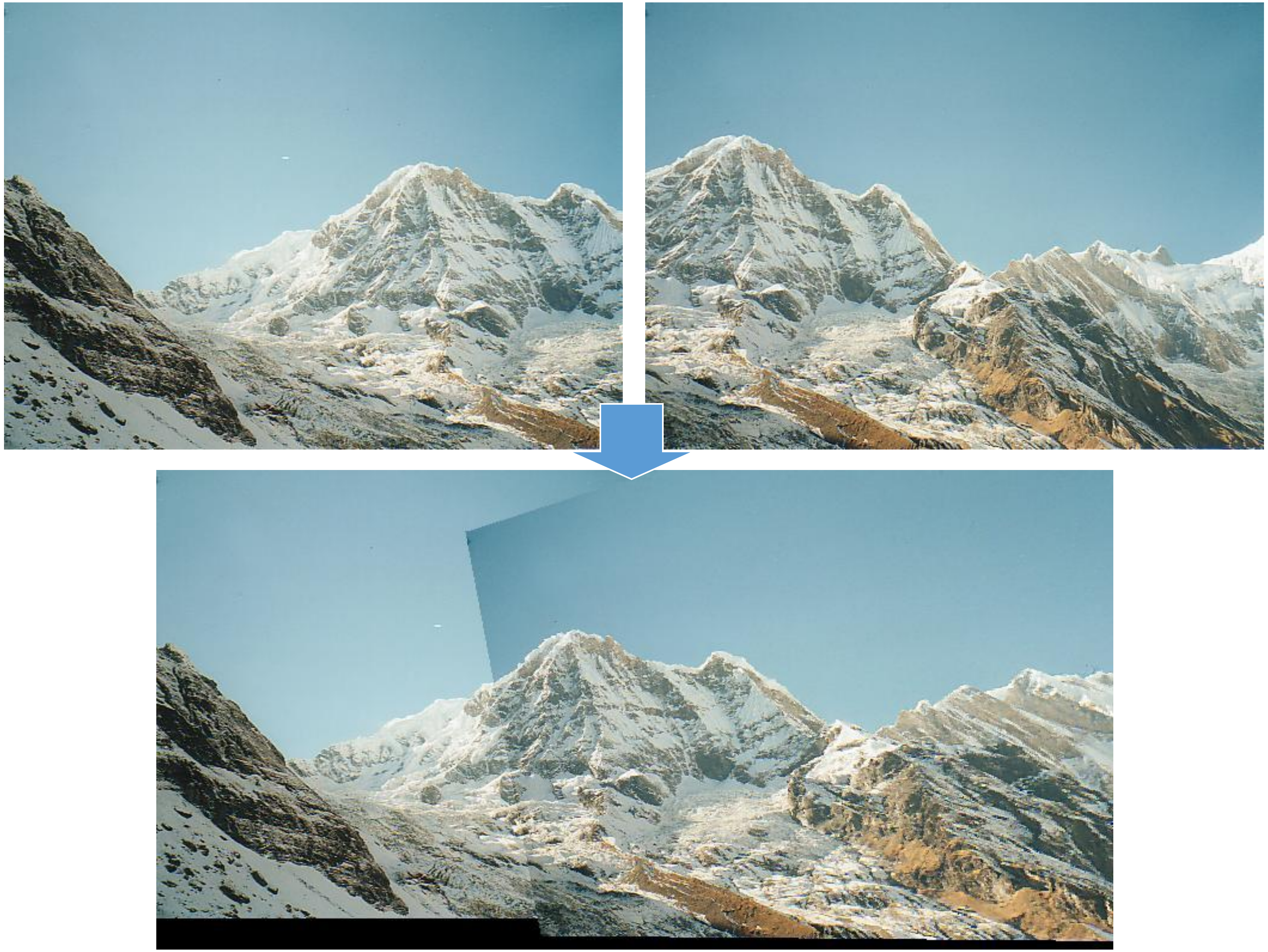
RANSAC for Homography



RANSAC for Homography

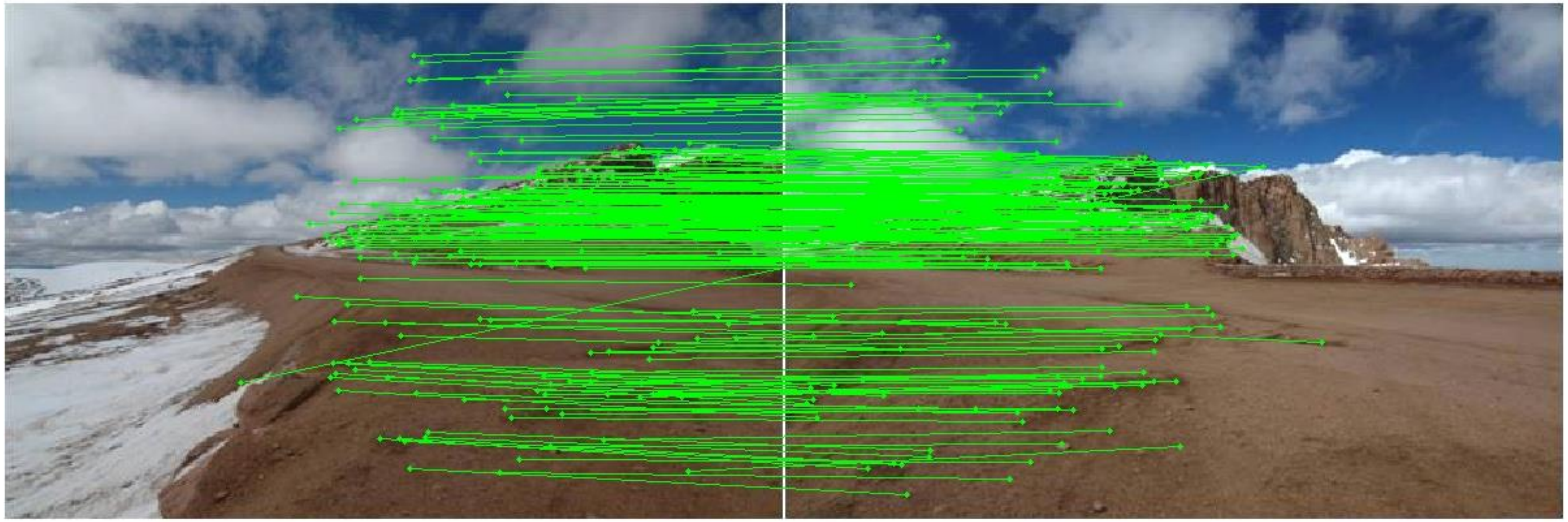


RANSAC for Homography

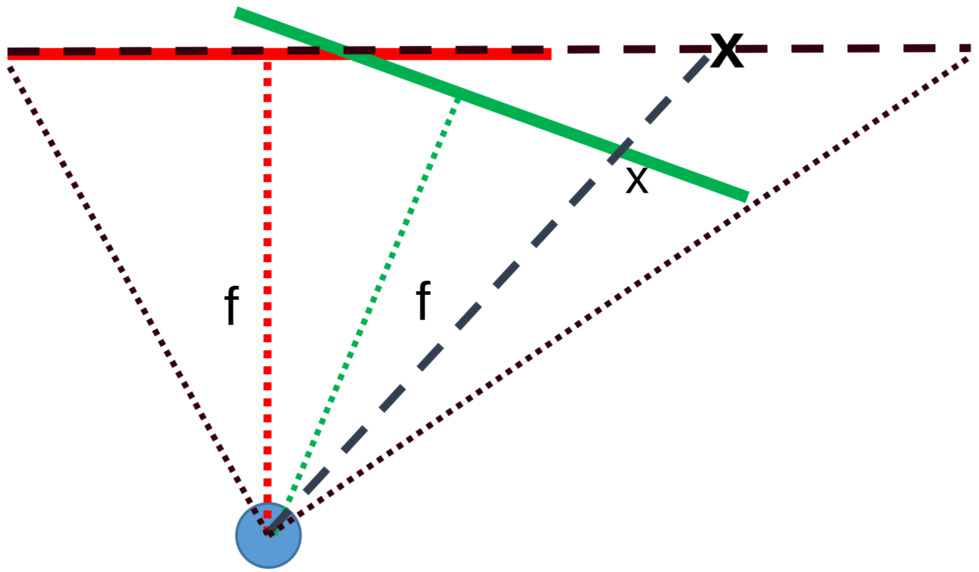


Choosing a Projection Surface

Different choices: planar, cylindrical, spherical, cubic, etc.

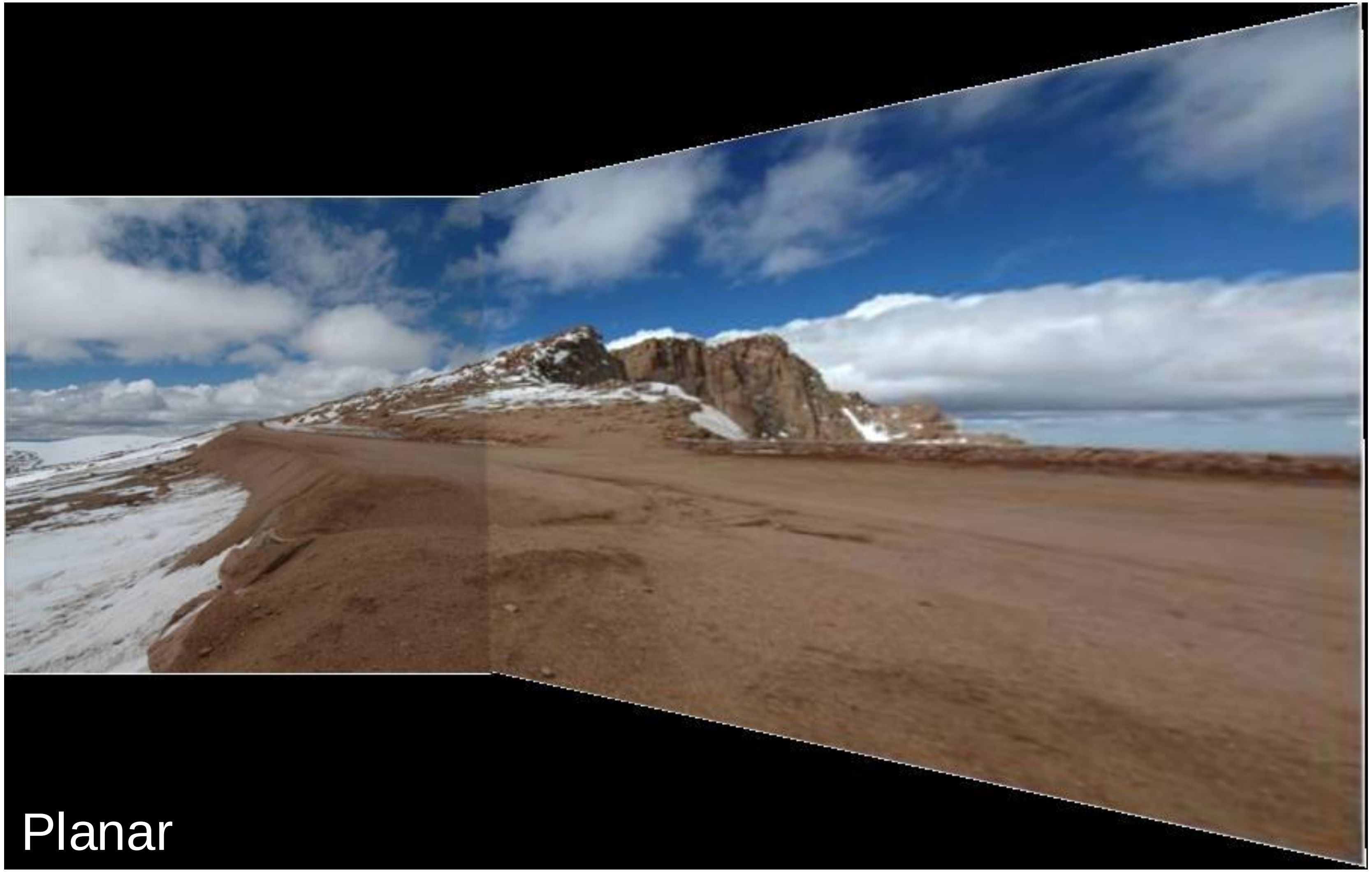


Planar Mapping



1) For red image: pixels are already on the planar surface
2) For green image: map to first image plane

Planar Projection

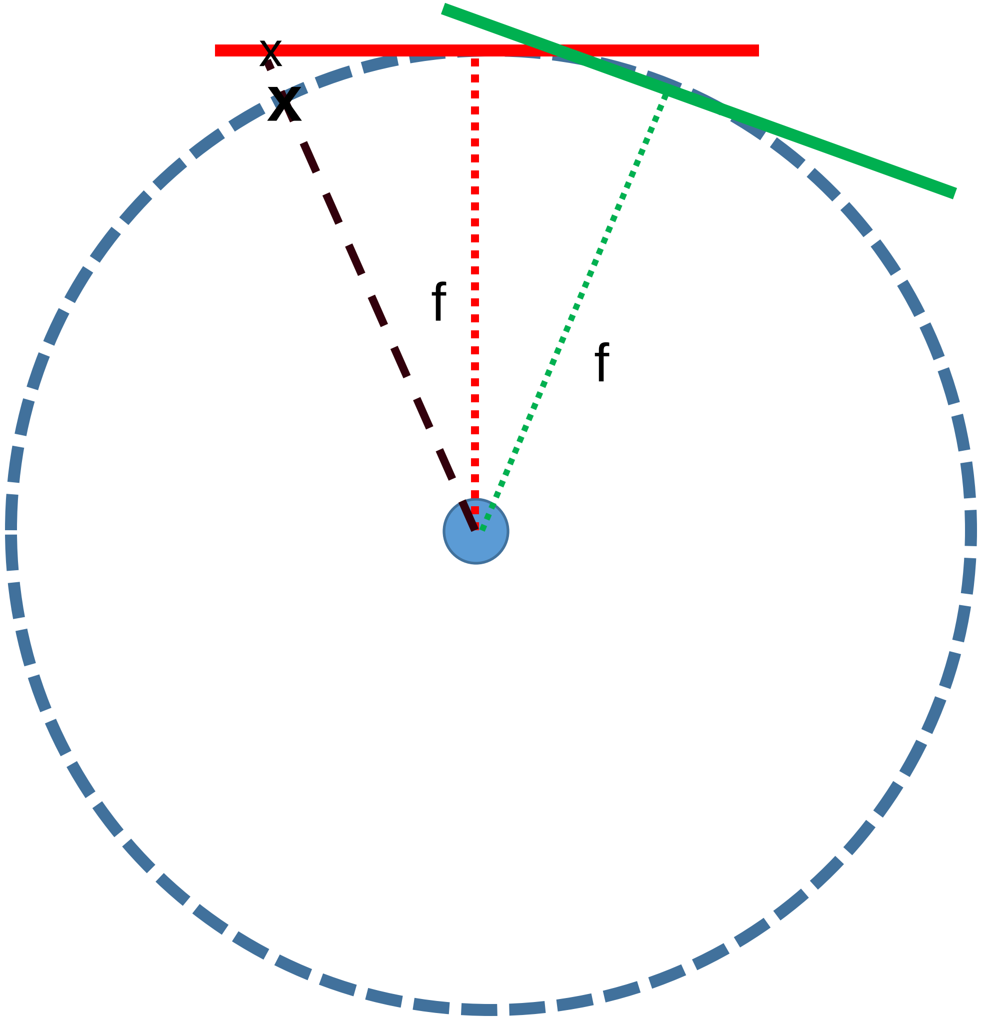


Photos by Russ Hewett

Planar Projection

Planar

Cylindrical Mapping



1) For red image: compute h, theta on cylindrical surface from (u, v)
2) For green image: map to first image plane, then map to cylindrical surface

Cylindrical Projection

Cylindrical

Cylindrical Projection

Cylindrical


Planar

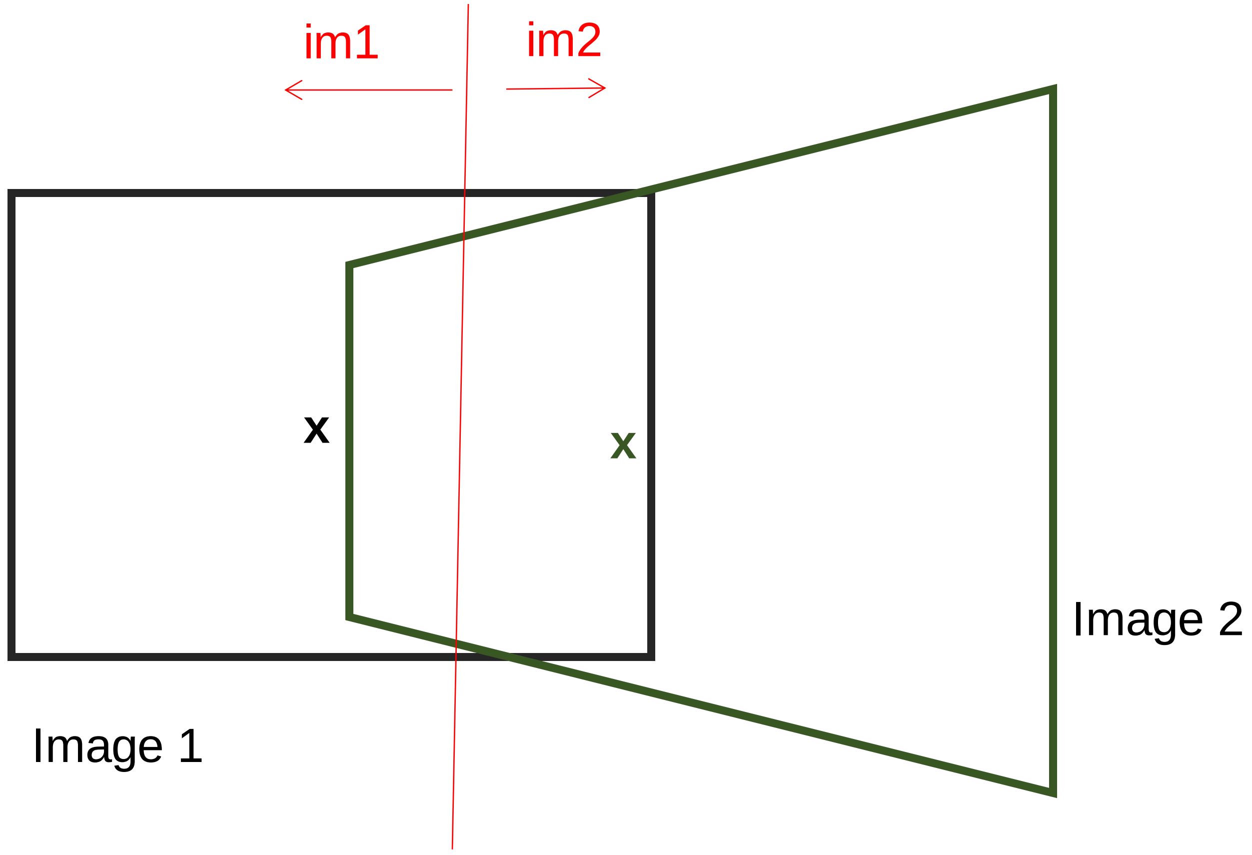

Cylindrical

Details to Make it Look Good

- Choosing seams
- Blending

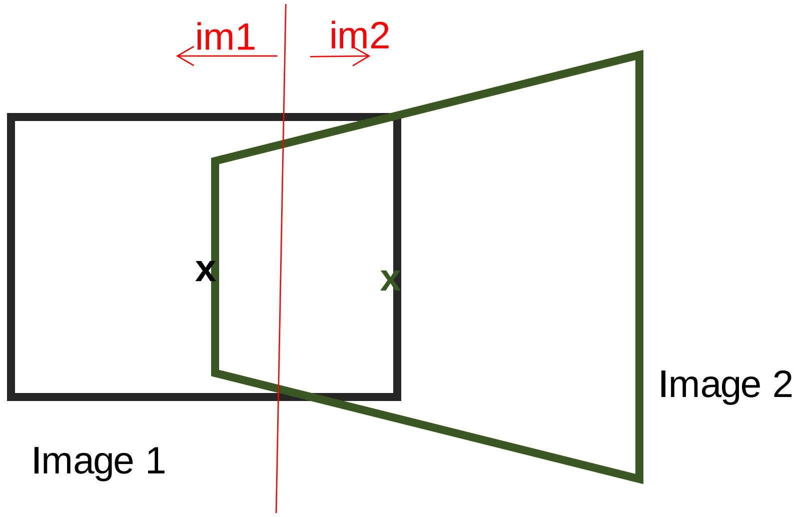
Choosing seams

- Easy method
  - Assign each pixel to image with nearest center



Choosing seams

- Easy method
  - Assign each pixel to image with nearest center
  - Create a mask:
    - `mask(y, x) = 1` iff. pixel should come from im1
  - Smooth boundaries (called “feathering”):
    - `mask_sm = imfilter(mask, gausfil);`
  - Composite
    - `imblend = im1_c.*mask + im2_c.*(1-mask);`



Choosing seams

- Better method: dynamic program to find seam along well-matched regions

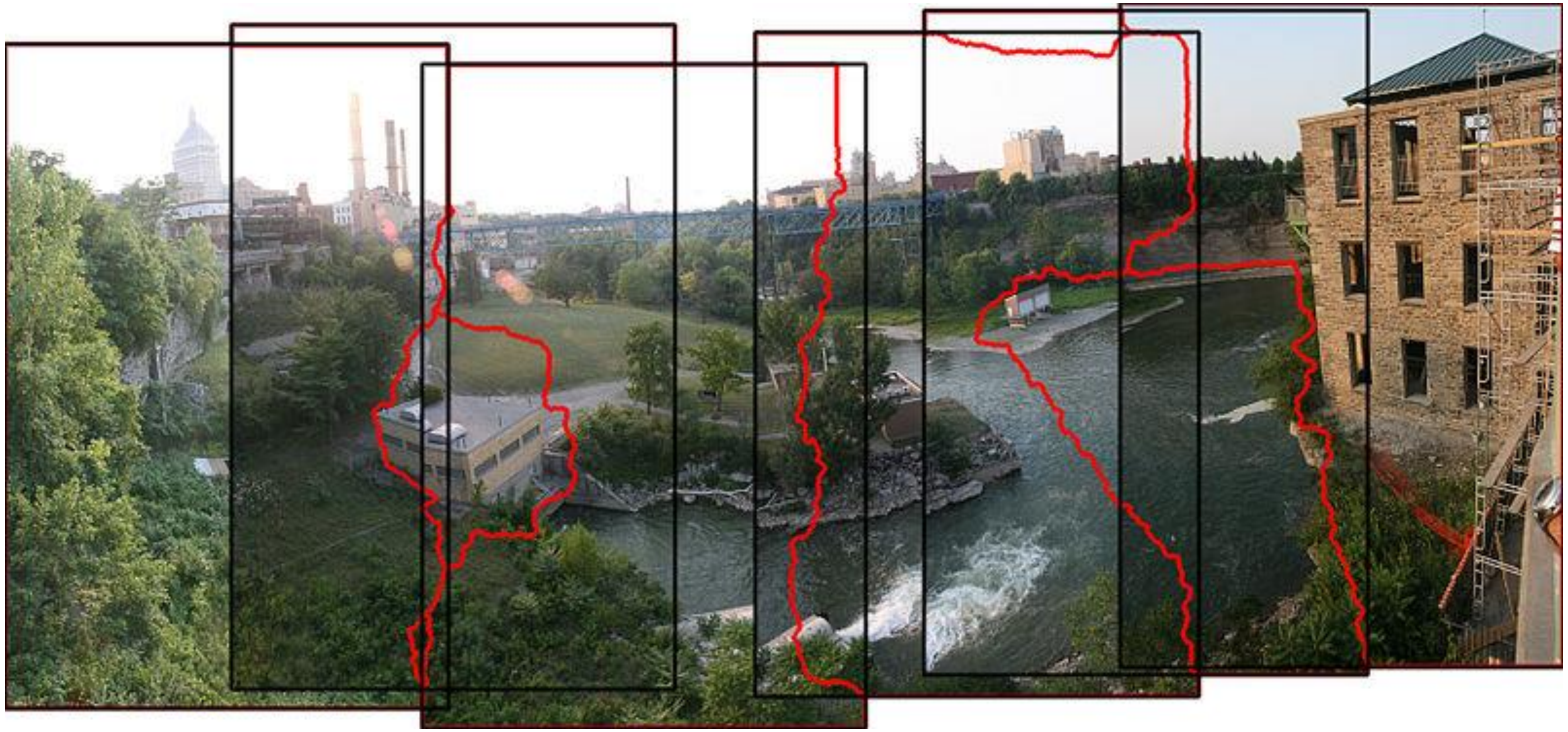
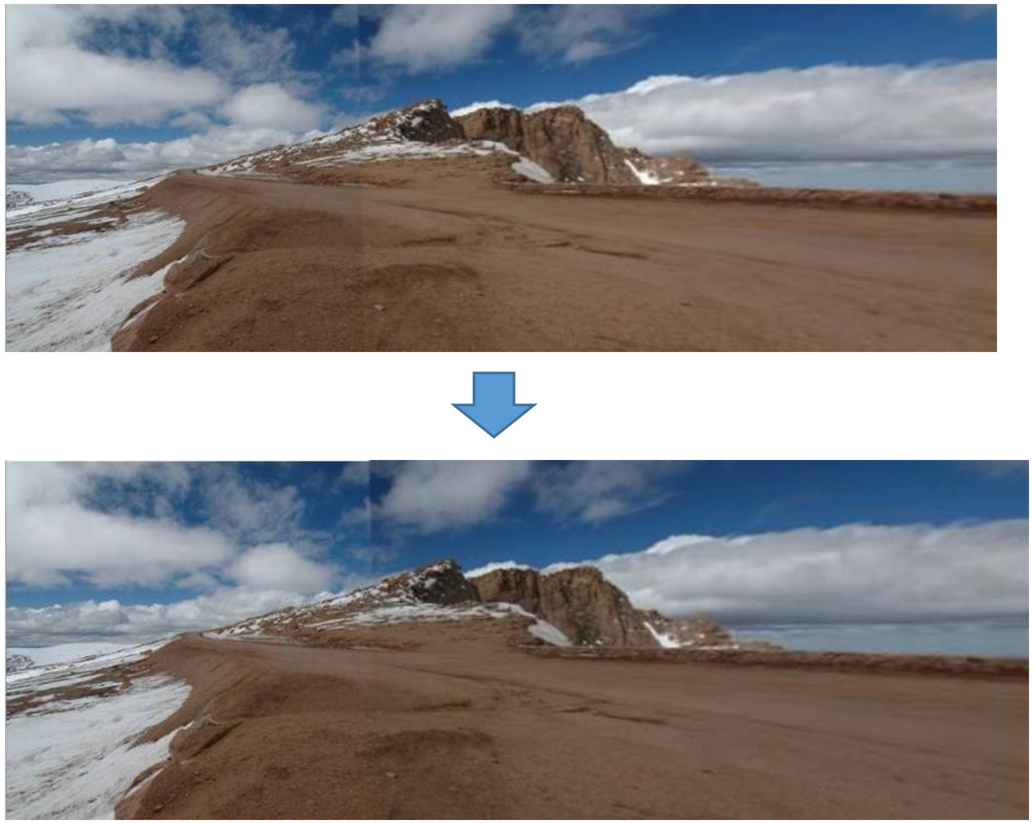
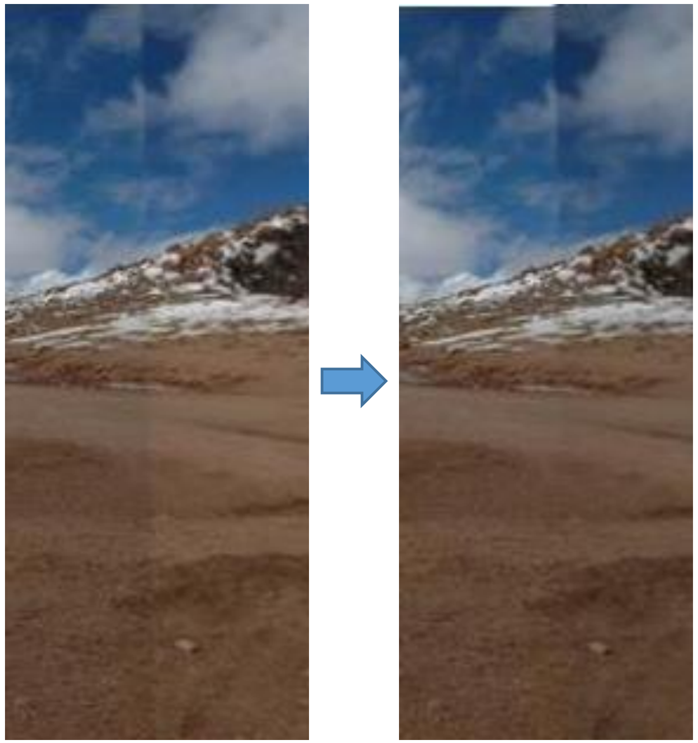


Illustration: http://en.wikipedia.org/wiki/File:Rochester_NY.jpg

Gain compensation

- Simple gain adjustment
  - Compute average RGB intensity of each image in overlapping region
  - Normalize intensities by ratio of averages



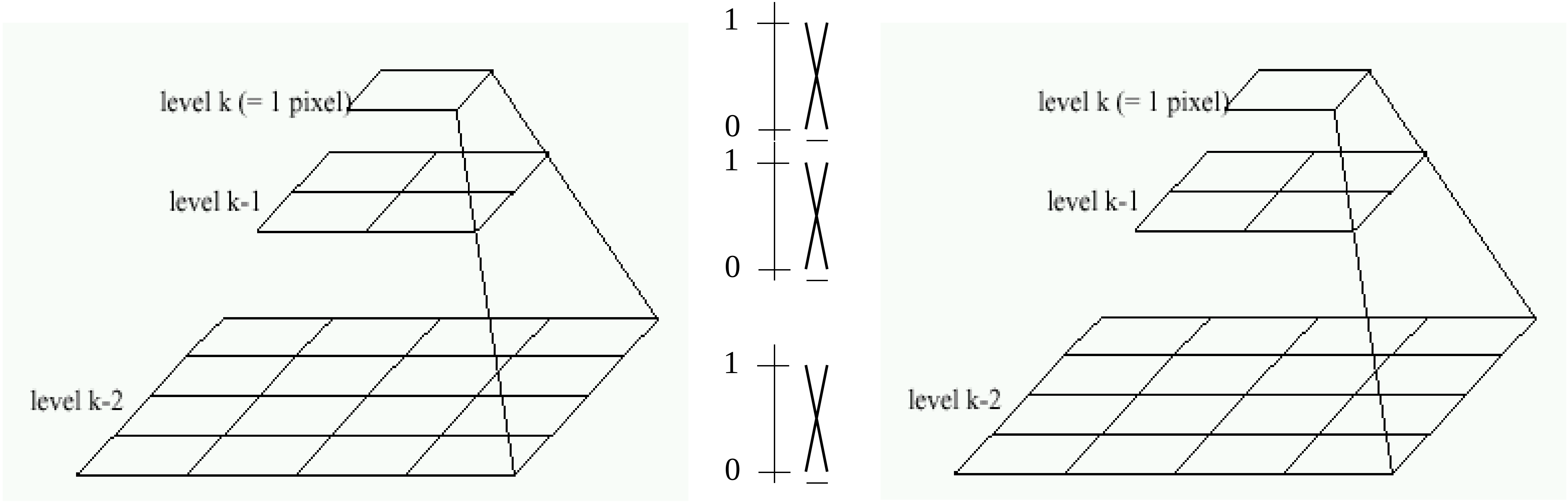


Multi-band Blending

- A multiresolution spline with application to image mosaics (Burt & Adelson 1983)
  - Blend frequency bands over range $\propto \lambda$

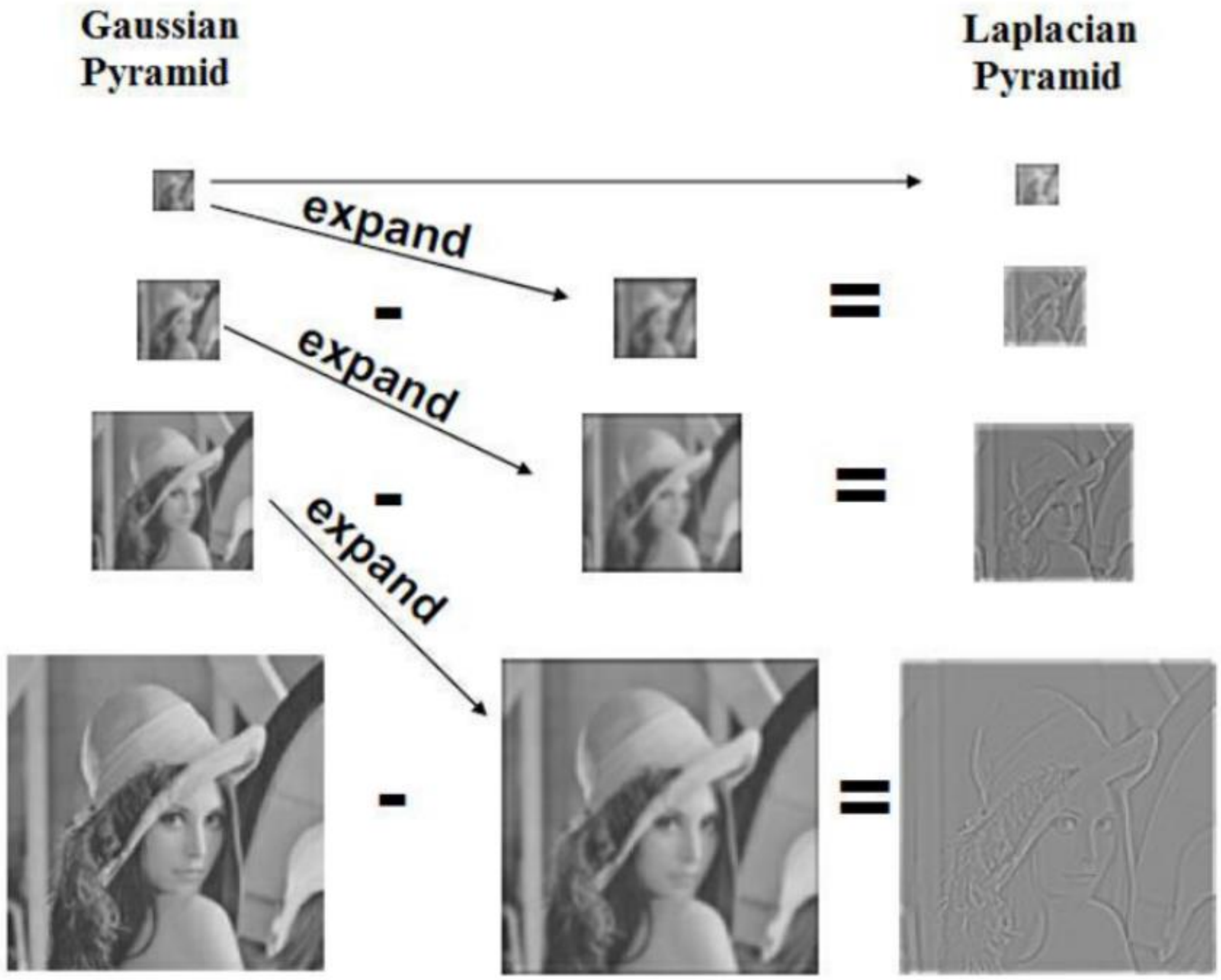
Multiband Blending with Laplacian Pyramid

- At low frequencies, blend slowly
- At high frequencies, blend quickly



Left pyramid blend Right pyramid

Laplacian Pyramid

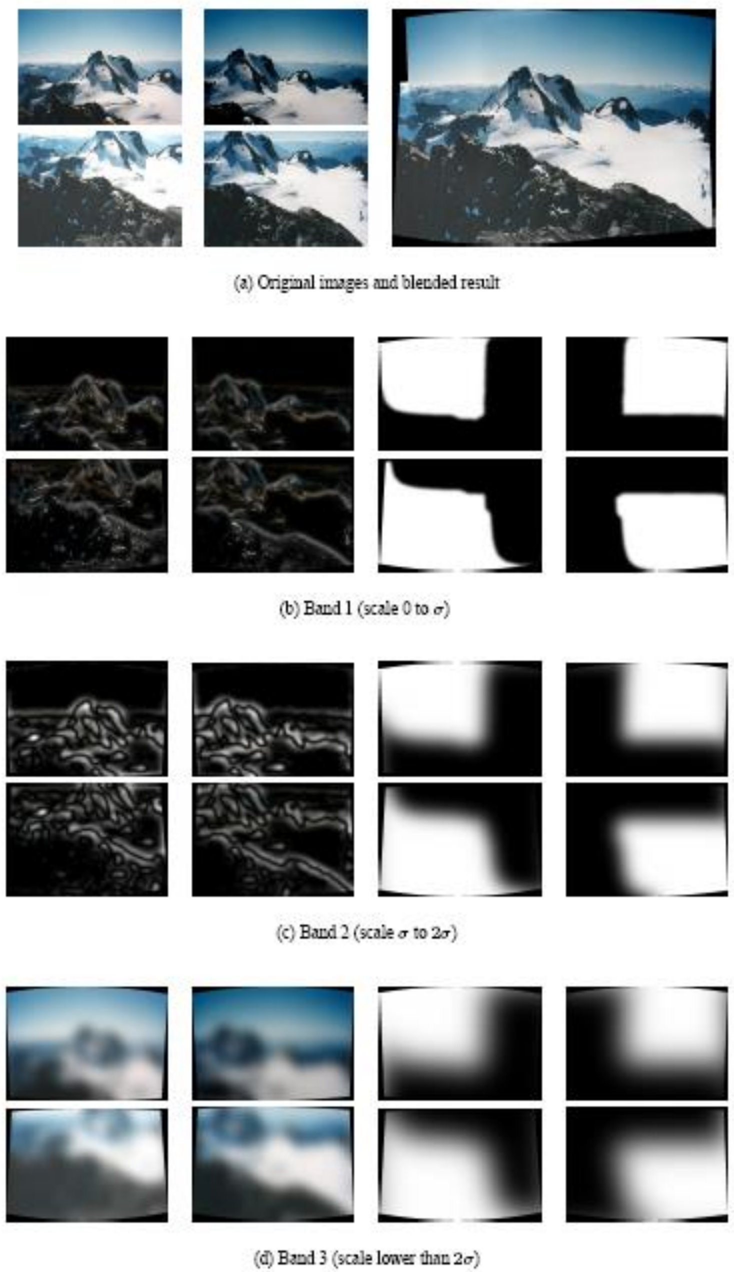


Multiband Blending

1. Compute Laplacian pyramid of images and mask

2. Create blended image at each level of pyramid

3. Reconstruct complete image



Laplacian pyramids

Blending comparison (IJCV 2007)

(a) Linear blending

(b) Multi-band blending

Blending Comparison

(b) Without gain compensation

(c) With gain compensation

(d) With gain compensation and multi-band blending

How does iPhone panoramic stitching work?

- Capture images at 30 fps
- Stitch the central 1/8 of a selection of images
  - Select which images to stitch using the accelerometer and frame-to-frame matching
  - Faster and avoids radial distortion that often occurs towards corners of images
- Alignment
  - Initially, perform cross-correlation of small patches aided by accelerometer to find good regions for matching
  - Register by matching points (KLT tracking or RANSAC with FAST (similar to SIFT) points) or correlational matching
- Blending
  - Linear (or similar) blending, using a face detector to avoid blurring face regions and choose good face shots (not blinking, etc)

http://www.patentlyapple.com/patently-apple/2012/11/apples-cool-iphone-5-panorama-app-revealed-in-5-patents.html

Further Reading

- Rick Szeliski’s alignment/stitching tutorial
- Recognising Panoramas