https://tinyurl.com/mrx54fxt

Computer Vision 1

HC4a

Global Transforms & Image Stitching

Dr. Martin Oswald, Dr. Dimitris Tzionas, Dr. Arun Mukundan,
(m.r.oswald, d.tzionas, a.mukundan)@uva.nl

University of Amsterdam

computer vision lab

Last Week

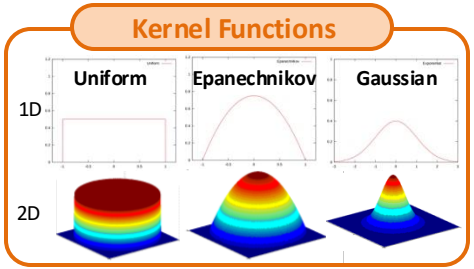
- Motion
- Optical Flow (OF)
  - Aperture Problem
  - Lukas-Kanade OF
  - Coarse-to-fine OF Estimation
  - Applications
- Tracking
  - Template Tracking
  - Tracking by Detection
  - Mean-shift Tracking

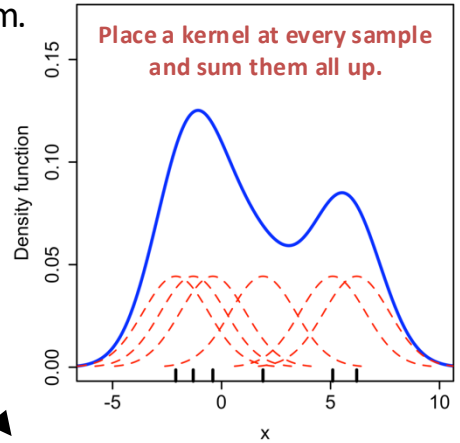
Mean Shift Tracking

Detour: Kernel Density Estimation (KDE)

Idea: Estimate a probability distribution of which a given point set could be sampled from. Compose the distribution as a sum of n basis/kernel functions K.

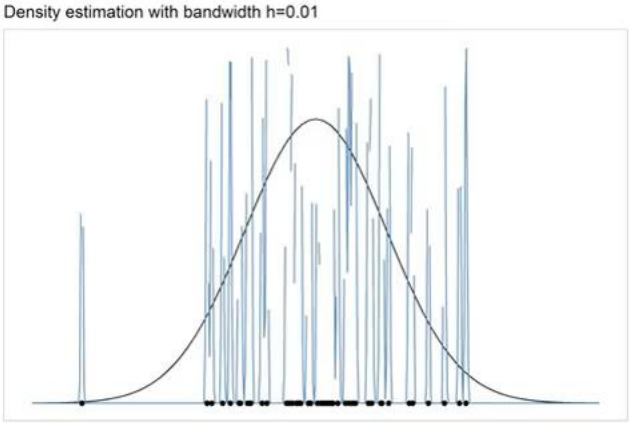
$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

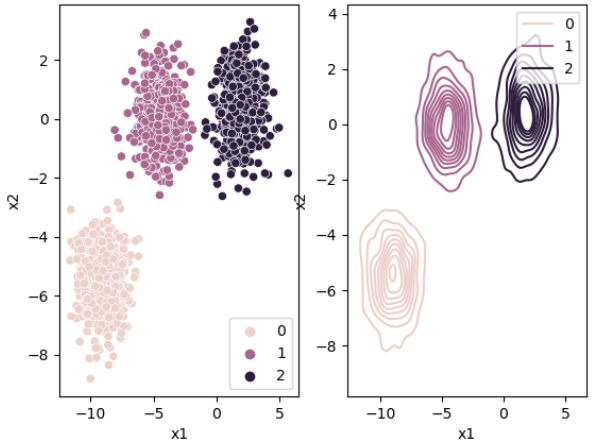




Bandwidth h:

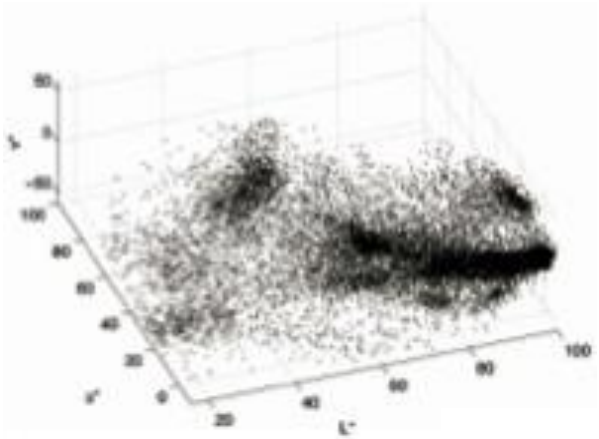
Defines level of detail and number of modes.

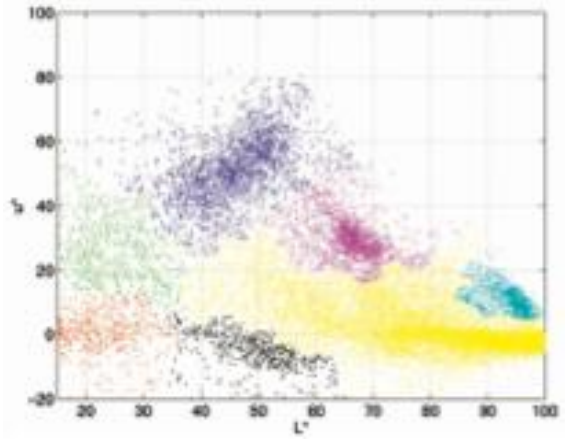


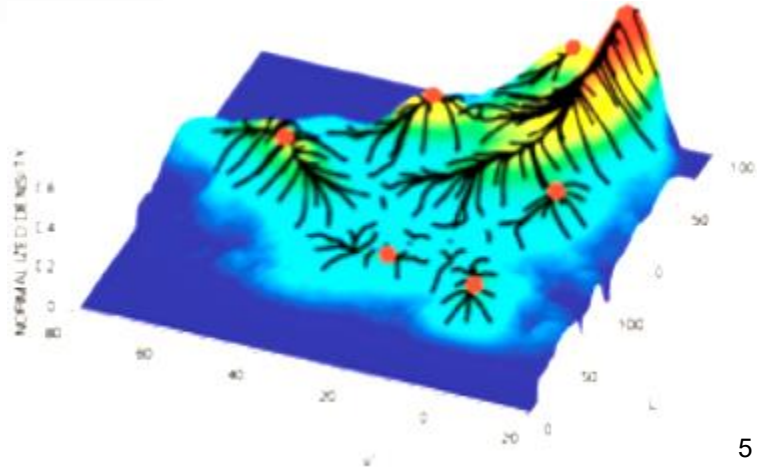


Mean Shift Clustering

Input image and its histogram





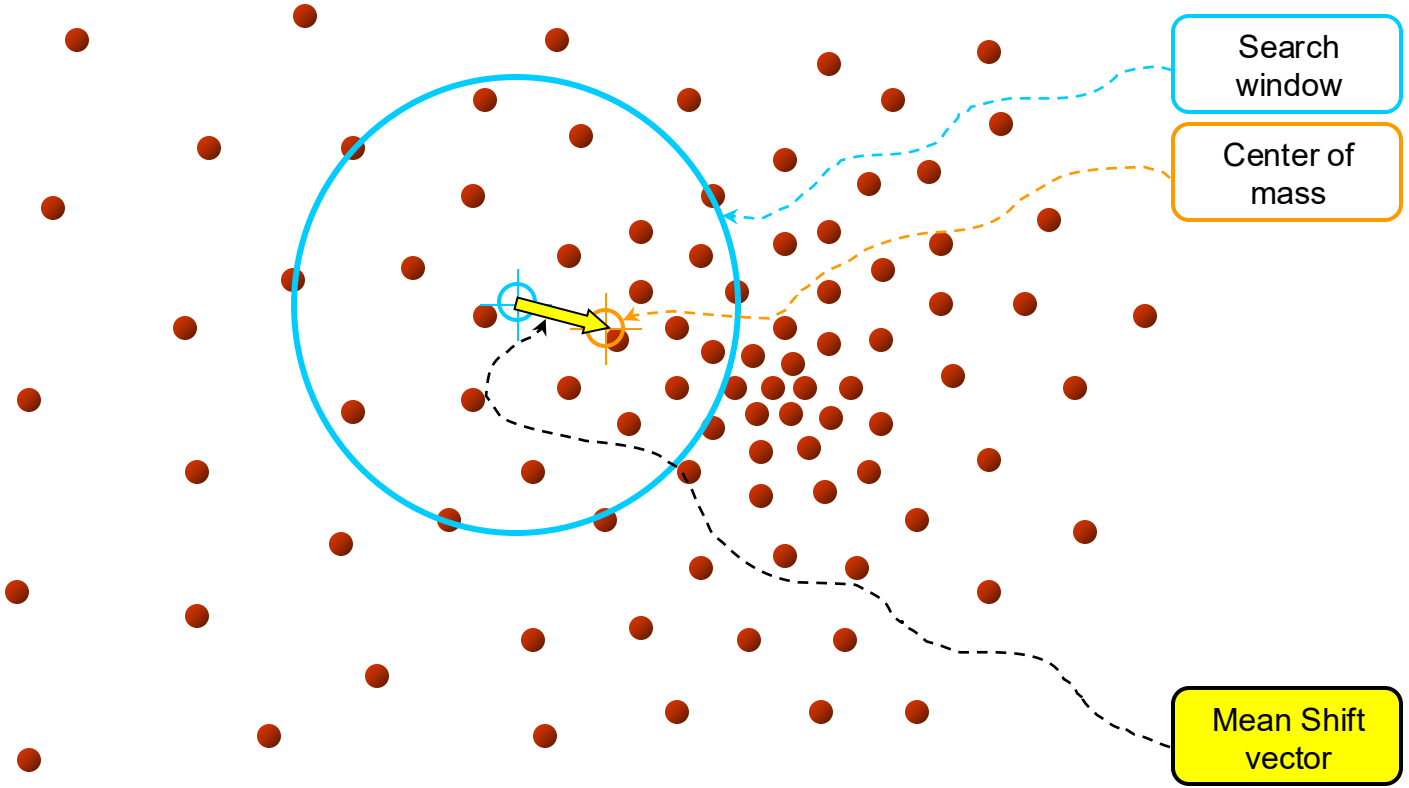


Idea: Density-based clustering

Goal: Given a bandwidth parameter and a kernel function, find all modes of a given probability distribution.

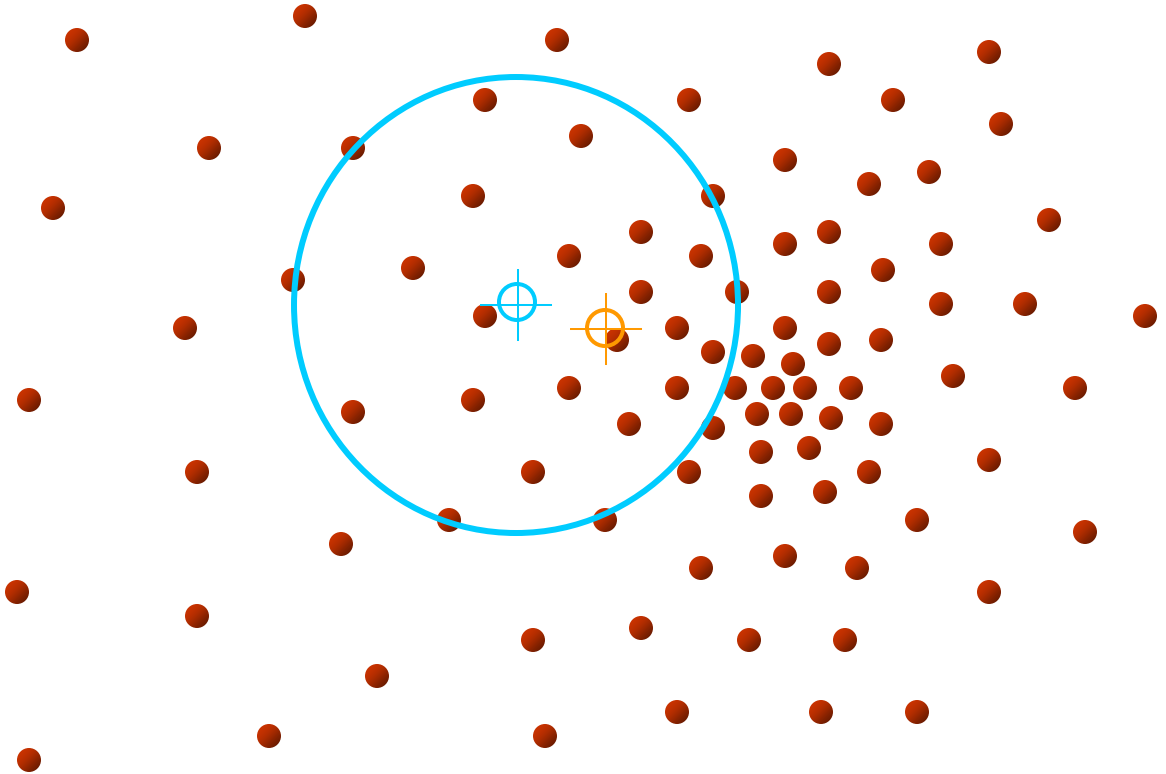
[Comaniciu, Meer. Mean shift: A robust approach toward feature space analysis, TPAMI, 2002]

Mean Shift Algorithm



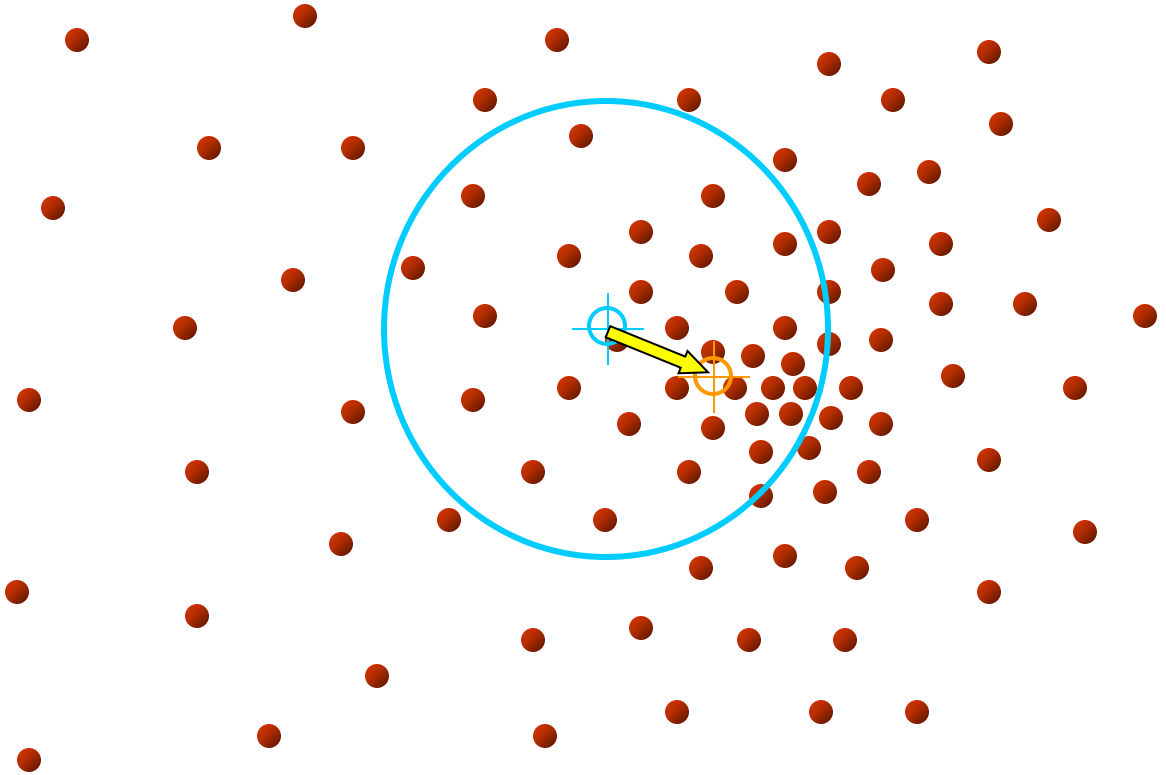
[Slide credit: Y. Ukrainitz & B. Sarel]

Mean Shift Algorithm



[Slide credit: Y. Ukrainitz & B. Sarel]

Mean Shift Algorithm



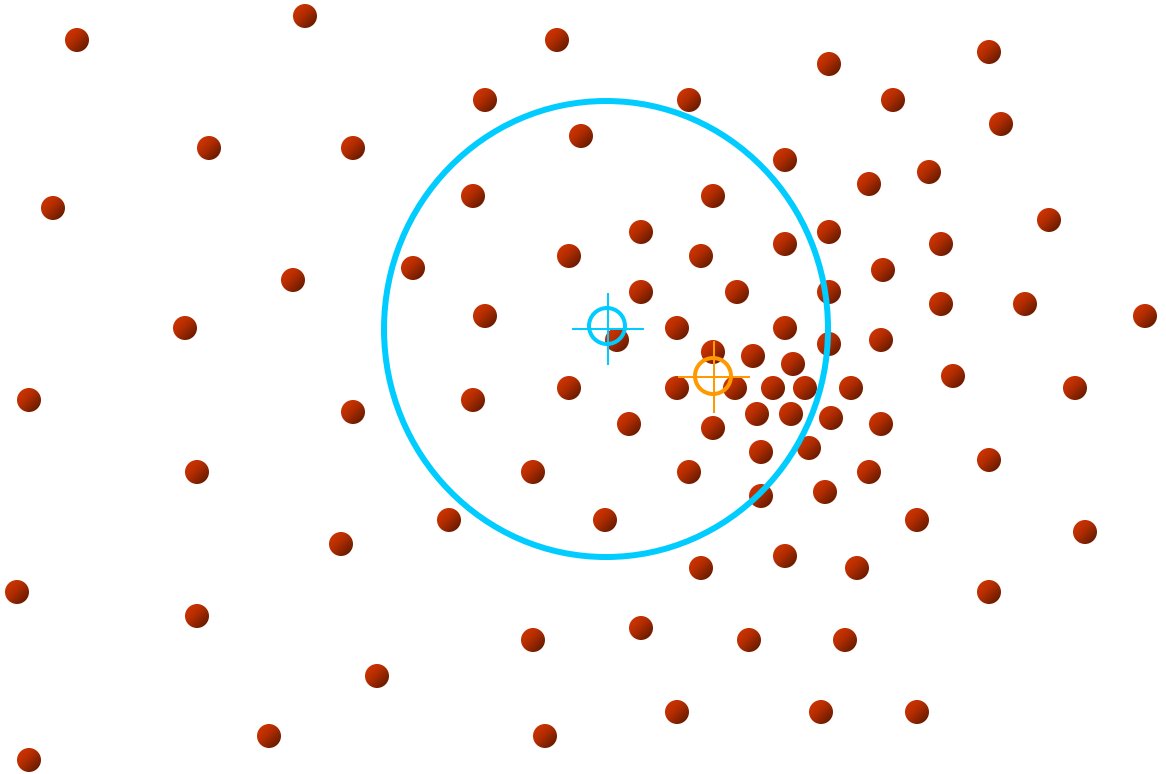
Search window

Center of mass

Mean Shift vector

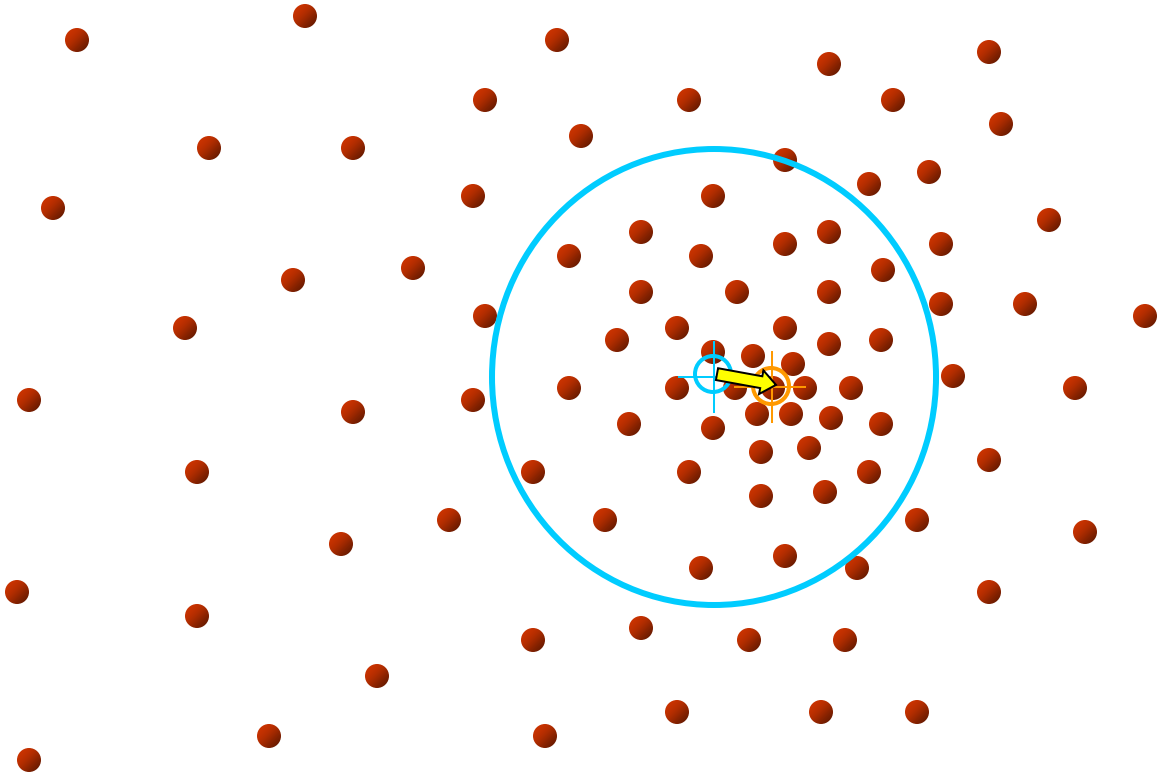
[Slide credit: Y. Ukrainitz & B. Sarel]

Mean Shift Algorithm



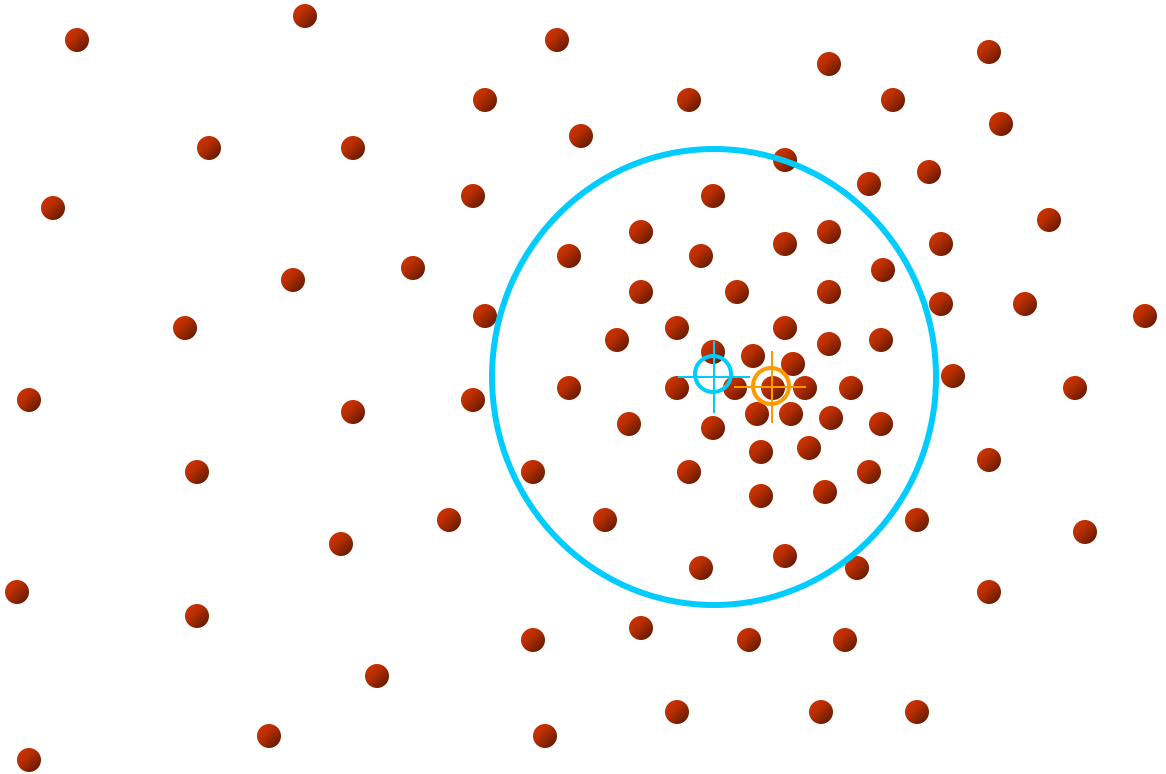
[Slide credit: Y. Ukrainitz & B. Sarel]

Mean Shift Algorithm



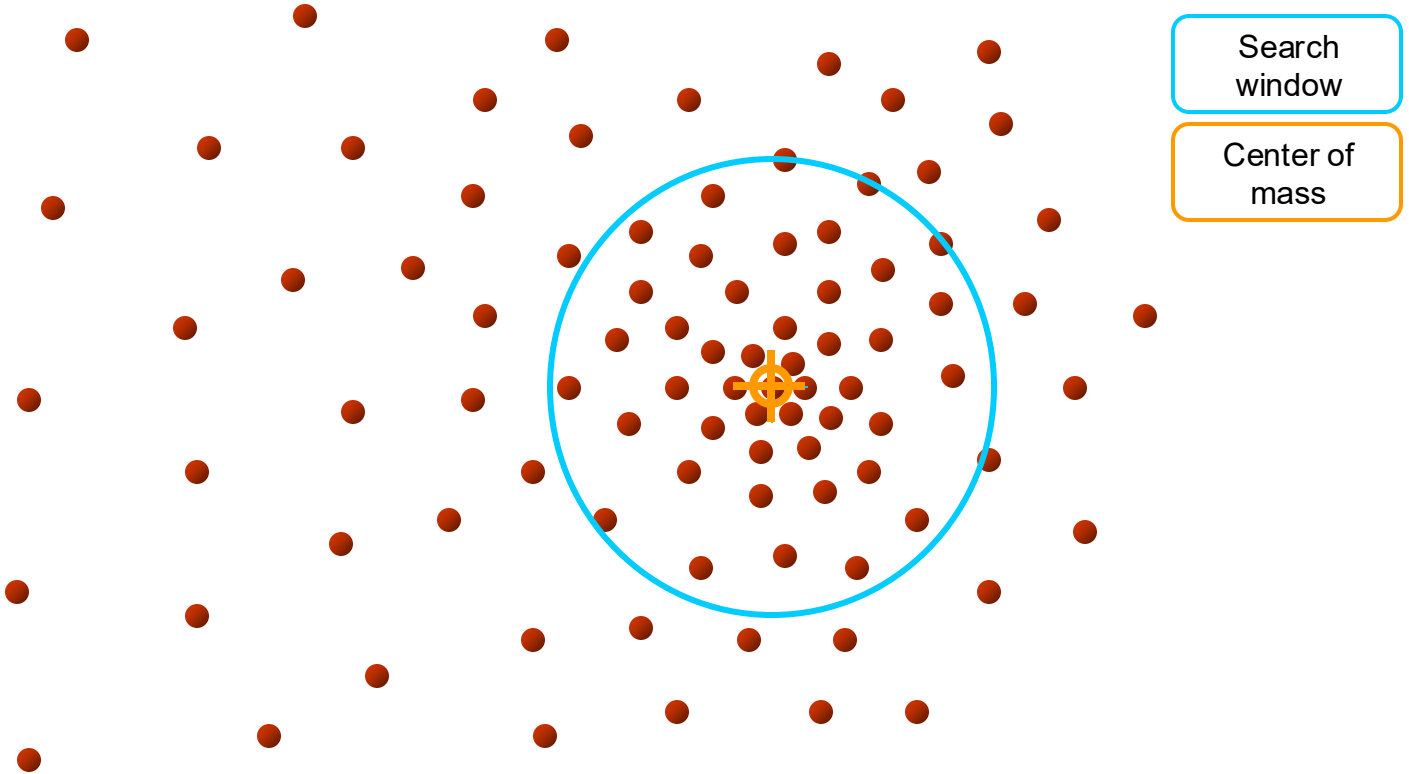
[Slide credit: Y. Ukrainitz & B. Sarel]

Mean Shift Algorithm



[Slide credit: Y. Ukrainitz & B. Sarel]

Mean Shift Algorithm

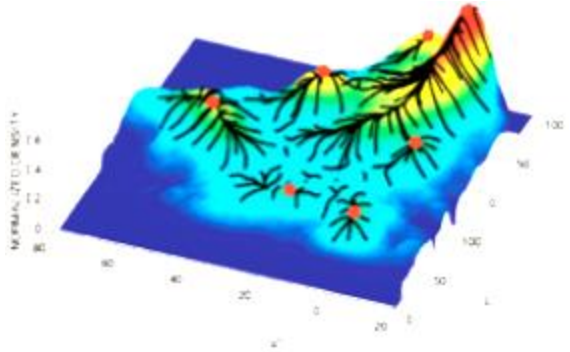


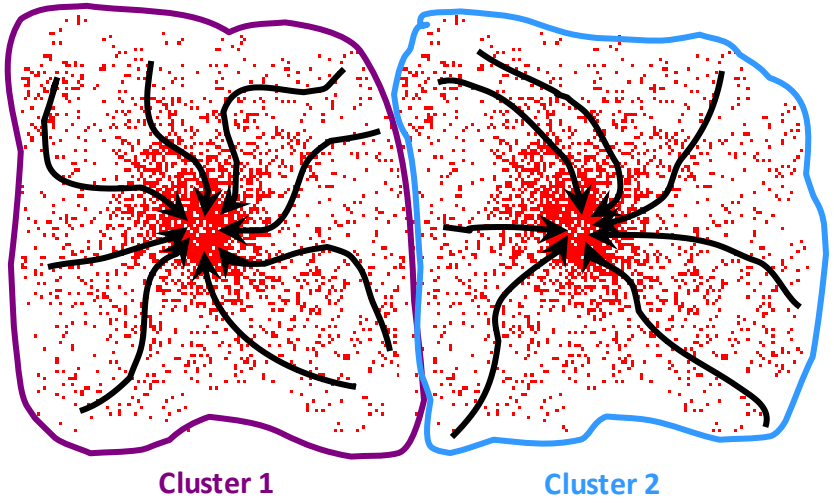
[Slide credit: Y. Ukrainitz & B. Sarel]

Mean Shift Algorithm

The mean shift algorithm seeks the **modes** of the given point set

1. Choose kernel and bandwidth
2. For each point:
   a) Center a window with a kernel on that point
   b) Compute the mean of the data in the search window
   c) Center the search window at the new mean location
   d) Repeat (b,c) until convergence
3. Assign points that lead to nearby modes to the same cluster



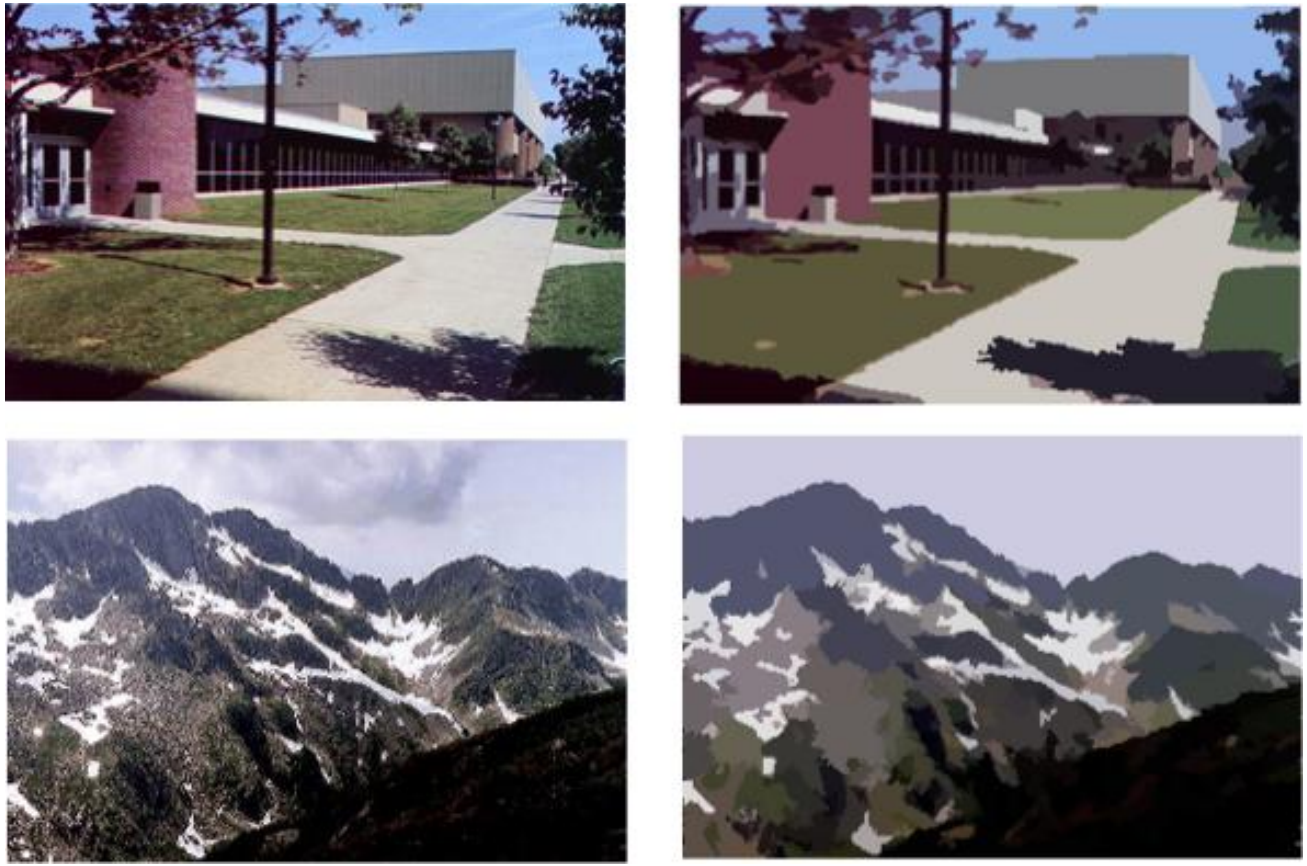


[Figure from: Y. Ukrainitz & B. Sarel]

- **Attraction basin:** Region for which all trajectories lead to the same mode
- **Cluster:** All points in the attraction basin of a mode
- **Note:** Bandwith and kernel type controls the size of attraction basins and the number of modes; and thus eventually the number of found clusters.

Mean Shift Segmentation

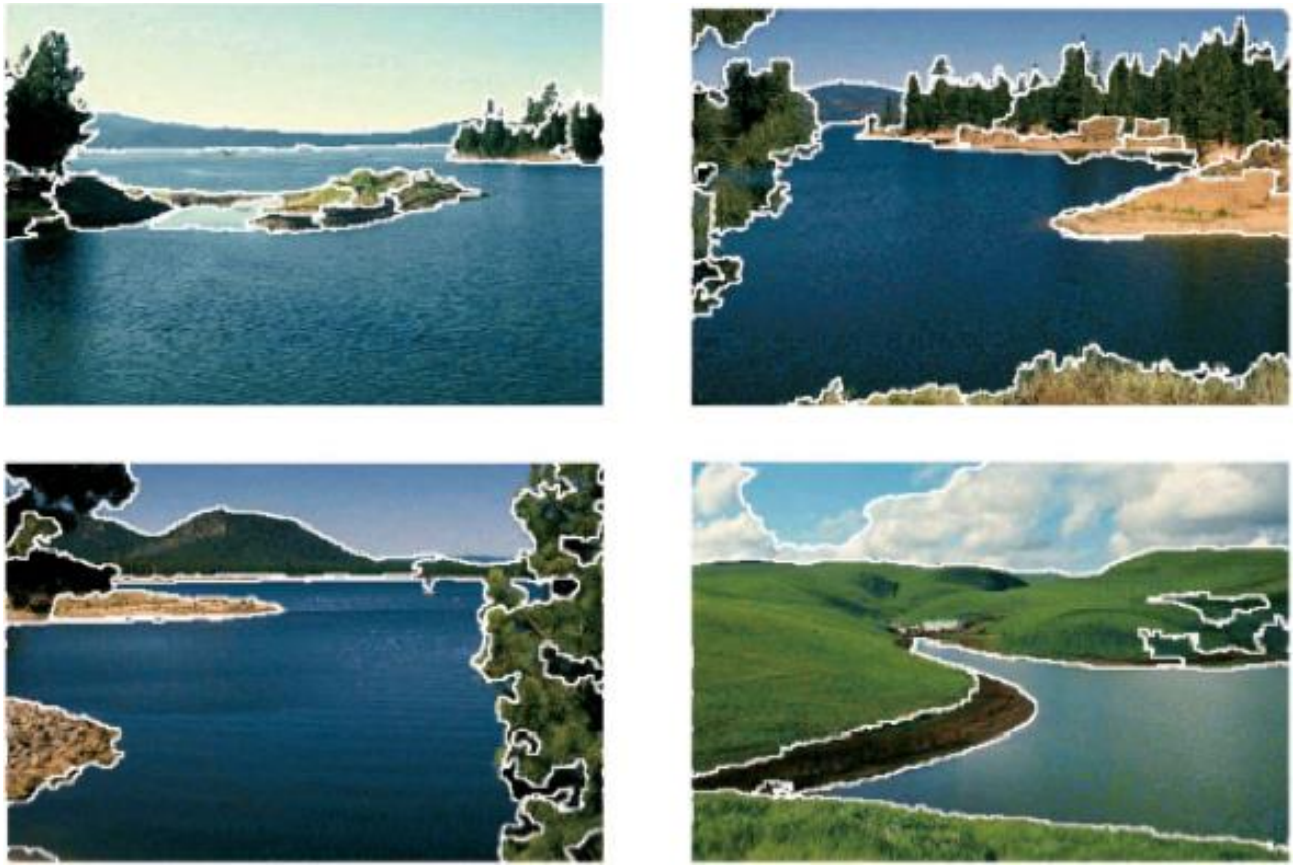
Results of clustering in LUV color space.



[Comaniciu, Meer. Mean shift: A robust approach toward feature space analysis, TPAMI, 2002]

Mean Shift Segmentation

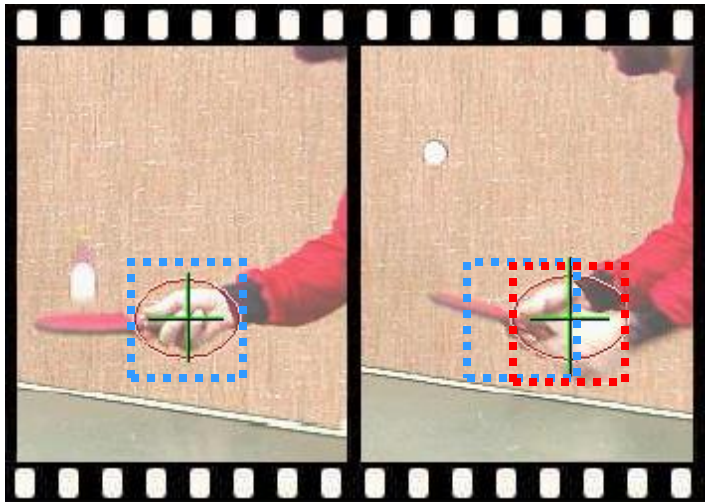
Results of clustering in LUV color space.



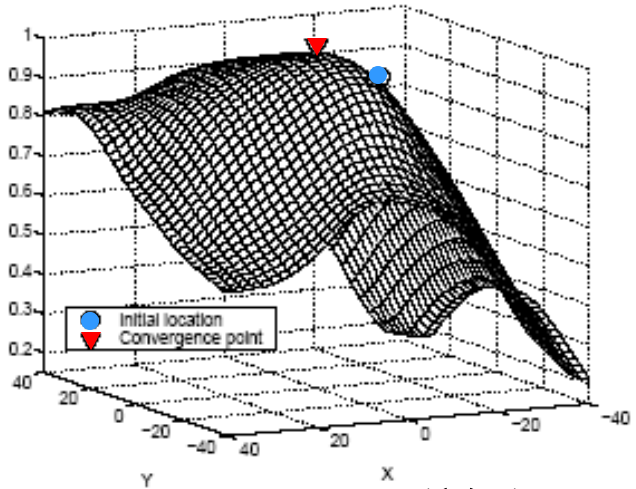
[Comaniciu, Meer. Mean shift: A robust approach toward feature space analysis, TPAMI, 2002]

Mean Shift Tracking

Define a **target region**: $\vec{q} = (q_1, \dots, q_m)$ with color distribution from m pixels.
In the next frame search the vicinity
with a **candidate region**: $\vec{p}(y) = (p_1(y), \dots, p_m(y))$ at position y.
Width a **similarity function** $f[\vec{p}(y), \vec{q}]$ like cosine similarity mean shift can find the next best position of the target candidate region.



target region: $\vec{q}$ candidate region: $\vec{p}(y)$



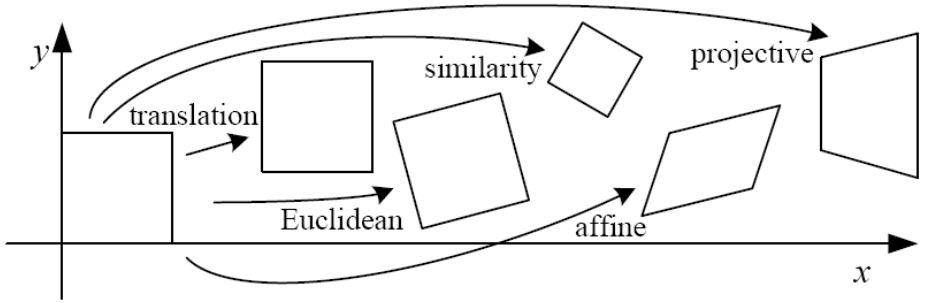
similarity function $f[\vec{p}(y), \vec{q}]$

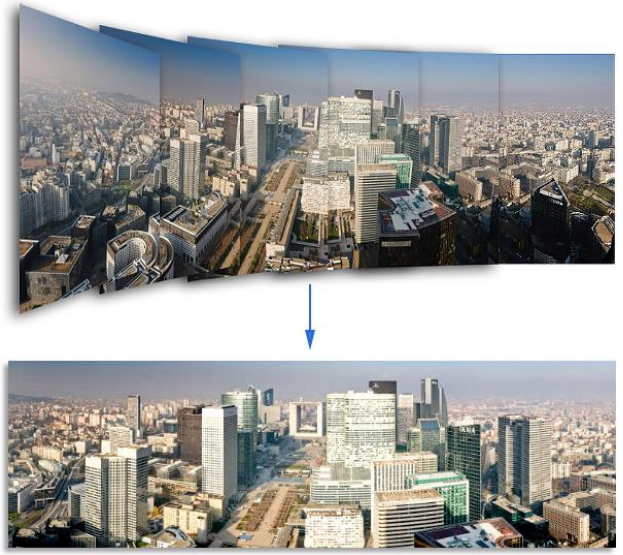
In every time step:
Use mean shift update to find the best location y by locally maximizing the region similarity function starting from the previous frame position.

Mean Shift Tracking Results

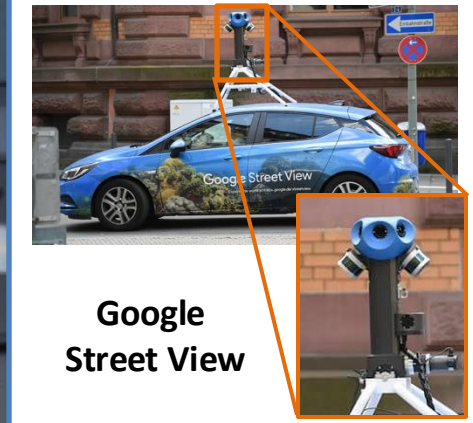
Today's Agenda

- **Global operations**
  - Global Transformations
  - Image Warping
  - Image Morphing
- **Image Stitching**
  - Transformation computation
    - Reprojection model
    - Key point matching
    - RANSAC
    - Solving transformation
  - Planar mapping
  - Blending
    - Feathering
    - Multi-band blending
    - Further improvements





Today's Motivation: How to do Panoramas?



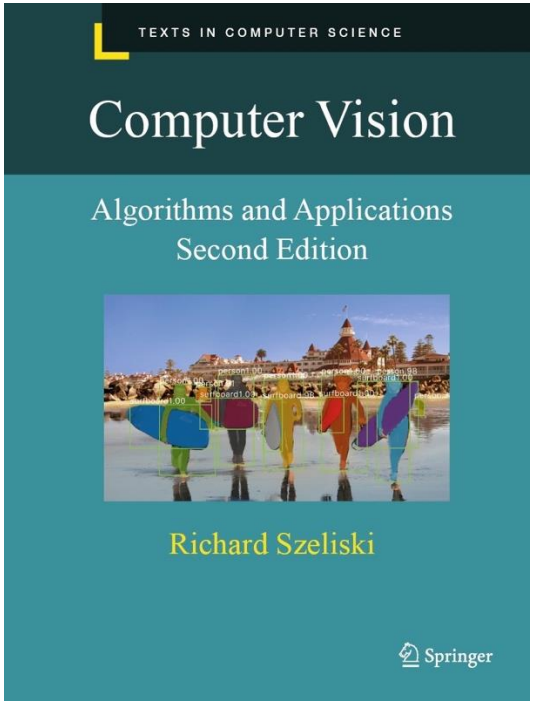
Google Street View

= Huge collection of stitched spherical images

[360° Panorama stitching with Google camera app: https://youtu.be/4sXw-v2fabU]

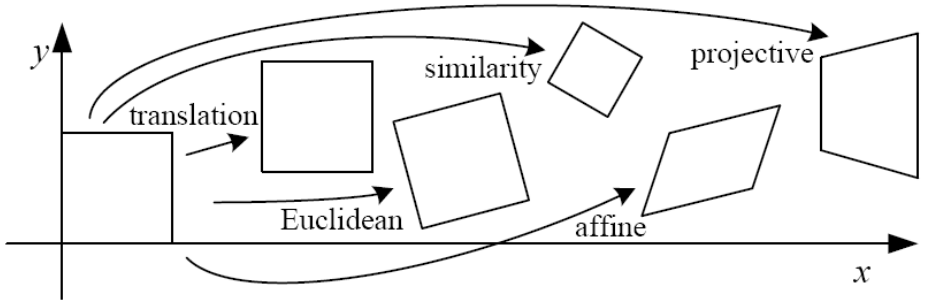
Textbook

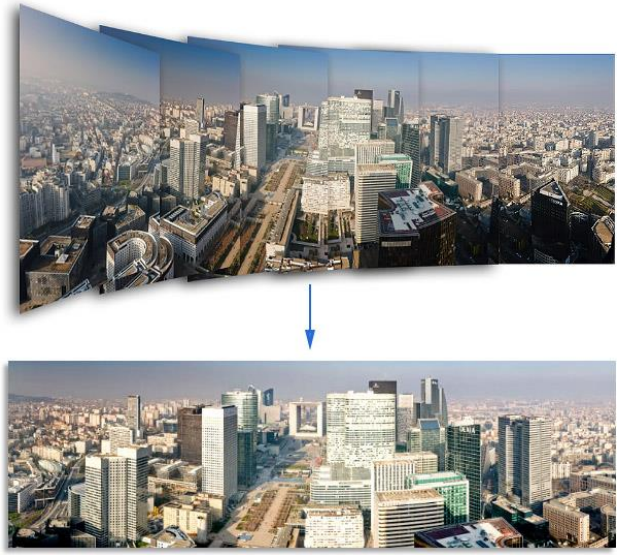
- Chapter 8



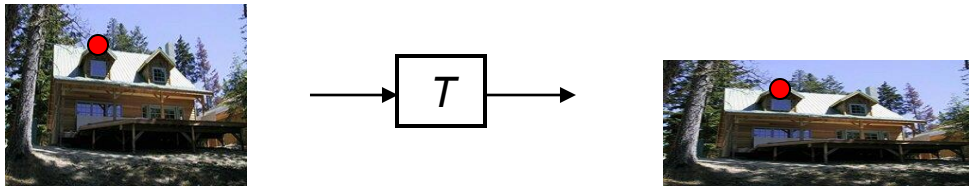
Today's Agenda

- **Global operations**
  - **Global Transformations**
  - Image Warping
  - Image Morphing
- Image Stitching
  - Transformation computation
    - Reprojection model
    - Key point matching
    - RANSAC
    - Solving transformation
  - Planar mapping
  - Blending
    - Feathering
    - Multi-band blending
    - Further improvements





Parametric (Global) Transformations



p = (x,y)

p' = (x',y')

Transformation T is a coordinate-changing operation:

$$p' = T(p)$$

What does it mean that *T* is global?

- *T* is the same for any point p in the entire image domain

T can be described by just a few numbers (parameters)

For linear transformations, we can represent T as a matrix

$$p' = \mathbf{T}p$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

Common Linear Transformations

Original

Transformed

Translation

Rotation

Scaling

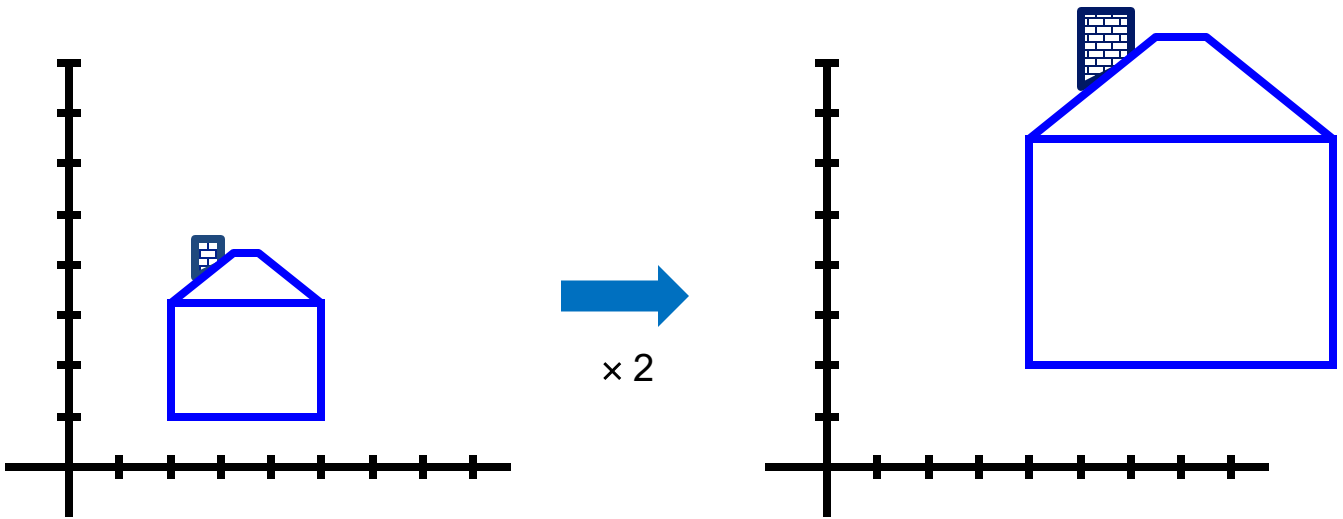
Affine

Perspective

[Slide credit (next few slides): A. Efros and/or S. Seitz]

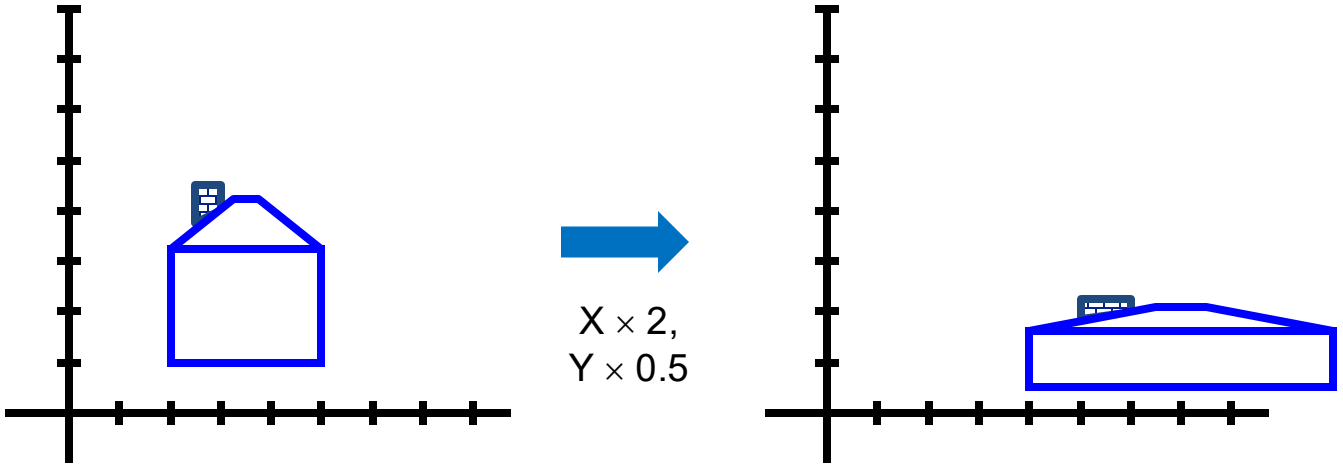
Scaling

- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:



Scaling

- *Non-uniform scaling*: different scalars per component:



Scaling

- Scaling operation:

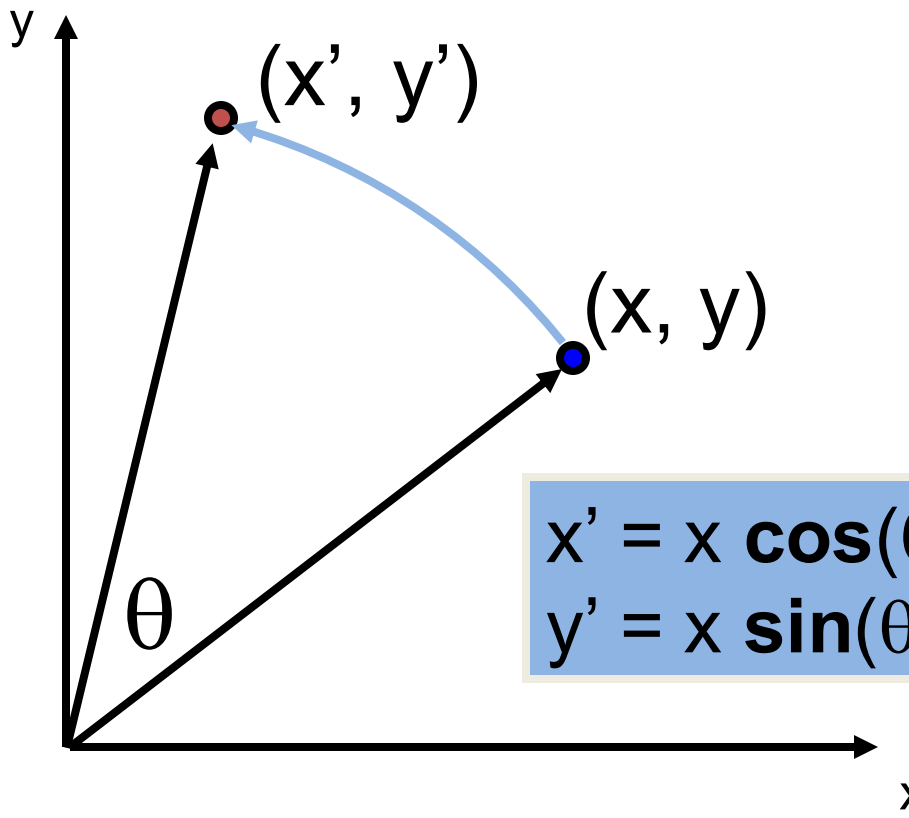
$$x' = ax$$

$$y' = by$$

- Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D Rotation



$$x' = x\cos(\theta) - y\sin(\theta)$$
$$y' = x\sin(\theta) + y\cos(\theta)$$

2D Rotation

This is easy to capture in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Even though sin(θ) and cos(θ) are nonlinear functions of θ,

- ***x' is a linear combination of x and y***
- ***y' is a linear combination of x and y***

What is the inverse transformation?

- Rotation by –θ
- For rotation matrices $\mathbf{R}^{-1} = \mathbf{R}^{T}$ (because of orthogonality)

2D Translation

- Q: How can we represent translation as a 3x3 matrix?

$$x' = x + t_x$$

$$y' = y + t_y$$

2D Translation

- *Homogeneous coordinates*
  → represent coordinates in 2 dimensions with a 3-vector

$$\begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{\text{homogeneous coords}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2D Translation

- Using the rightmost column:

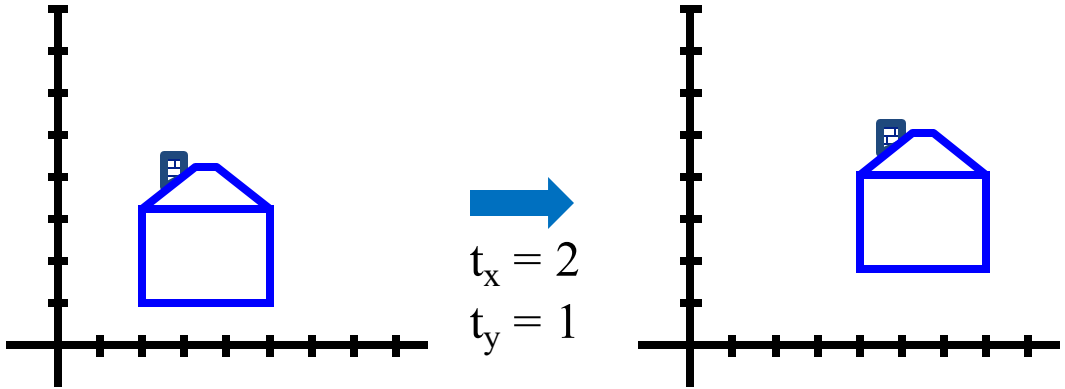
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2D Translation

- Example of translation

Homogeneous Coordinates

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$



Basic 2D Transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh}_x & 0 \\ \boldsymbol{sh}_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Shear

Affine Transformations - Intro

[Steve Seitz, Affine Transformations, 2022]

Affine Transformations

- Affine transformations are combinations of …
  - Linear transformations, and
  - Translations
- Properties of affine transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition
  - Models change of basis

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Basic 2D Transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

Affine is any combination of translation, scale, rotation, and shear

Transformation Composition

- Transformations can be combined by matrix multiplication

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$\mathbf{p}'$ = $T(t_x,t_y)$ $R(\Theta)$ $S(s_x,s_y)$ $\mathbf{p}$

Projective Transformations (a.k.a. Homographies)

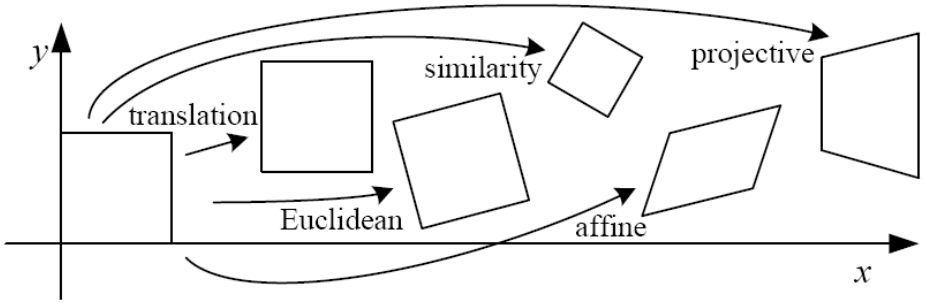
- Projective transformations …
  - Affine transformations, and
  - Projective warps
- Properties of projective transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Ratios are not preserved
  - Closed under composition
  - Models change of basis

$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}$$

$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}$$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Basic 2D Transformations



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

These transformations are a nested set of groups.
Closed under composition and inverse is a member.

Reminder: Groups

A **group** (G, ·) is a set *G* with a binary operation on *G*, denoted "·", that combines any two elements *a* and *b* to form an element of G, denoted *a·b*, such that the following requirements (group axioms) are satisfied:

- **Closedness:** For all *a, b* in *G*, one has *(a·b)* in *G*.
- **Associativity:** For all *a, b, c* in *G*, one has $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
- **Identity element:** There exists an element *e* in *G* such that, for every *a* in *G*, one has
  $e \cdot a = a$ and $a \cdot e = a$.
  Such an element is unique and called the *identity element* of the group.
- **Inverse element:** For each *a* in *G*, there exists an element *b* in *G* such that
  $a \cdot b = e$ and $b \cdot a = e$, where *e* is the identity element.
  For each *a*, the element *b* is unique; it is called the inverse of *a* and denoted a^{-1}.

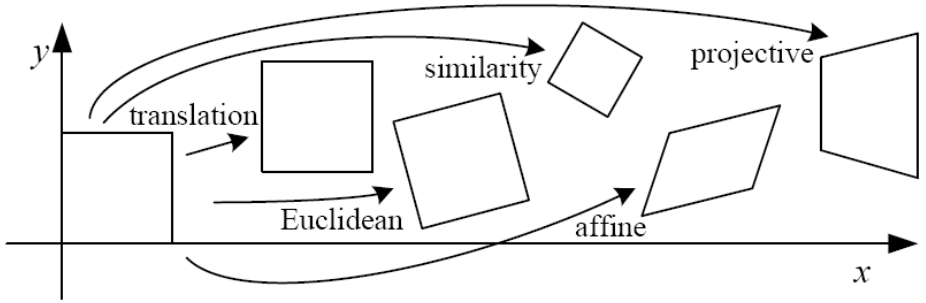
Overview: Matrix Groups

Set of square n x n matrices	General Linear Group	Orthogonal Group	Special Orthogonal Group
$\mathcal{M}(n)$	$\supset GL(n) = \{A \in \mathcal{M}(n) \mid \det(A) \neq 0\}$	$\supset O(n) = \{R \in GL(n) \mid R^T R = I\}$	$\supset SO(n) = \{R \in O(n) \mid det(R) = 1\}$
	Invertible square matrices	Orthogonal matrices	Rotation matrices

General Linear Group	Affine Group	Euclidean Group	Special Euclidean Group
$GL(n+1)$	$\supset \mathcal{A}(n) = \left\{ \begin{pmatrix} A & b \\ 0 & 1 \end{pmatrix} \middle\| A \in GL(n), b \in \mathbb{R}^n \right\}$	$\supset E(n) = \left\{ \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \middle\| R \in O(n), t \in \mathbb{R}^n \right\}$	$\supset SE(n) = \left\{ \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \middle\| R \in SO(n), t \in \mathbb{R}^n \right\}$
	Affine matrices		Rigid-body motions

Today's Agenda

- **Global operations**
  - Global Transformations
  - **Image Warping**
  - Image Morphing
- Image Stitching
  - Transformation computation
    - Reprojection model
    - Key point matching
    - RANSAC
    - Solving transformation
  - Planar mapping
  - Blending
    - Feathering
    - Multi-band blending
    - Further improvements



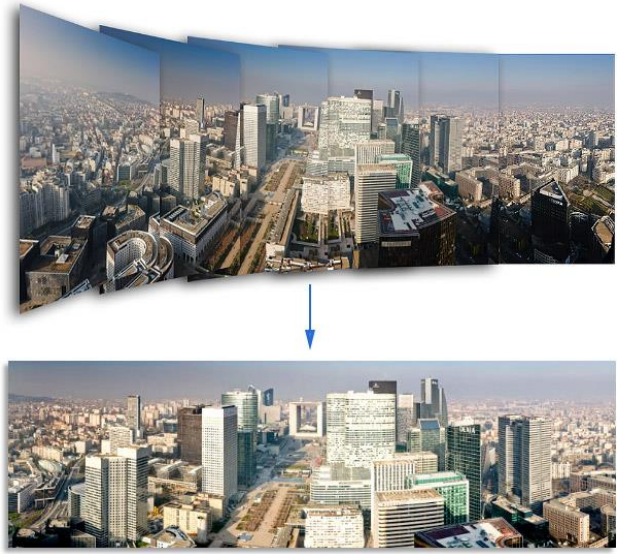
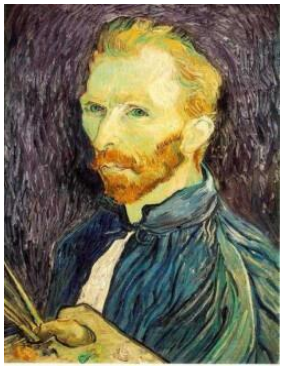
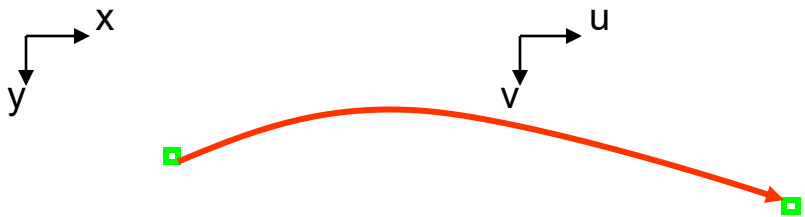


Image Warping

[Image source: http://www.jeffrey-martin.com]

Image Warping



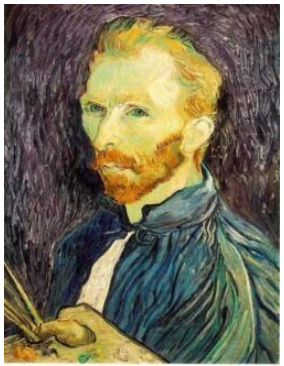


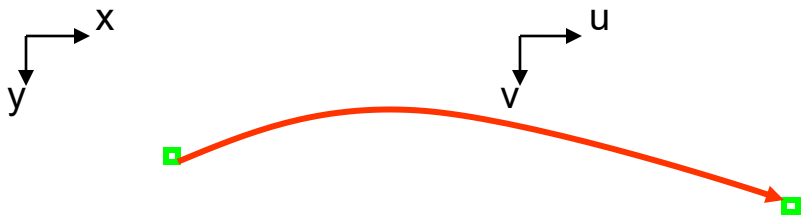
S(x,y)

T(u,v)

Given a coordinate transform function [f,g] (or the inverse i.e., [F,G]) and source image S(x,y), how do we compute a transformed target image T(u,v)?

Forward Warping





S(x,y)

T(u,v)

Forward warping algorithm:

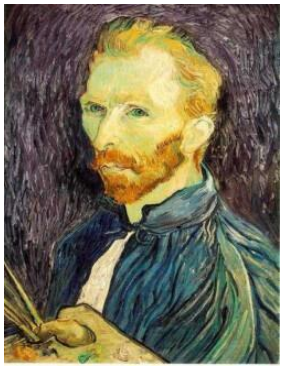
for y = y_{min} to y_{max}

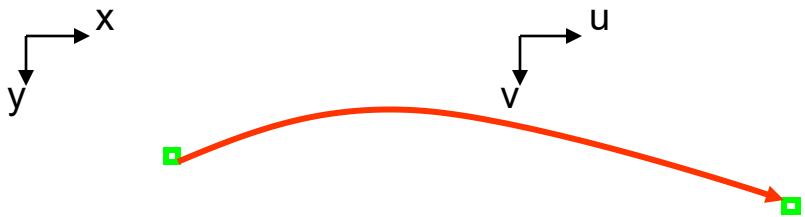
for x = x_{min} to x_{max}

u = f(x,y); v = g(x,y)

copy pixel at source S(x,y) to T(u,v)

Forward Warping





S(x,y)

T(u,v)

Forward warping algorithm:

for y = y_{min} to y_{max}

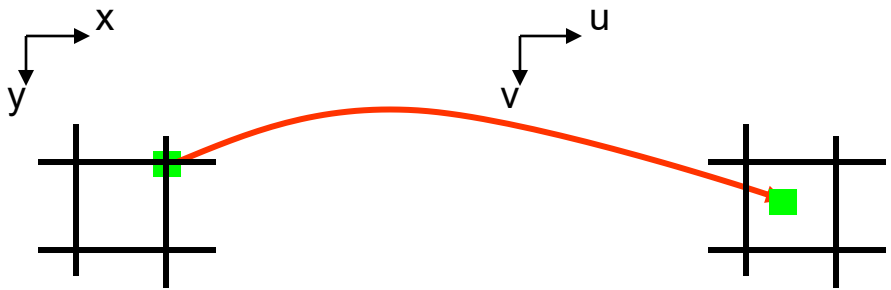
for x = x_{min} to x_{max}

u = f(x,y); v = g(x,y)

copy pixel at source S(x,y) to T(u,v)

- Any problems for forward warping?

Forward Warping

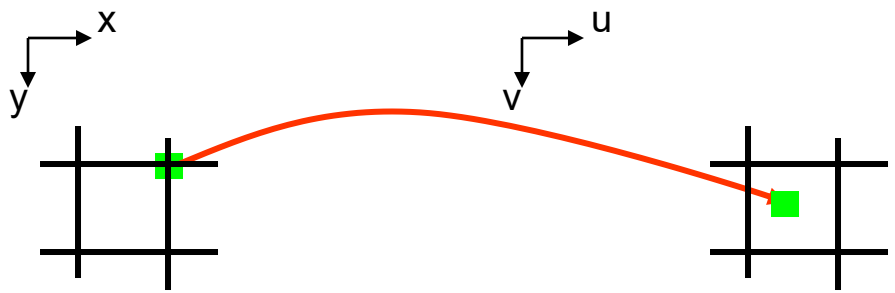


S(x,y)

T(u,v)

Q: What if the transformed pixel located between pixels?

Forward Warping



S(x,y)

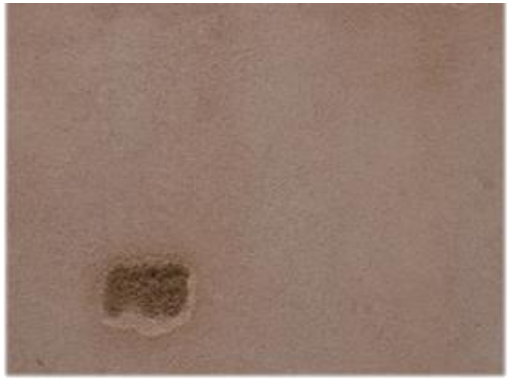
T(u,v)

Q: What if the transformed pixel located between pixels?

A: Distribute color among neighboring pixels

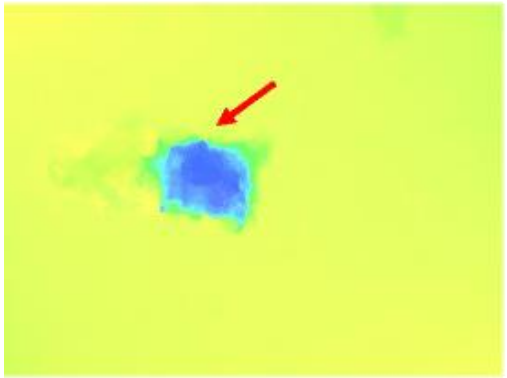
- known as “splatting”

Forward Warping - Problems

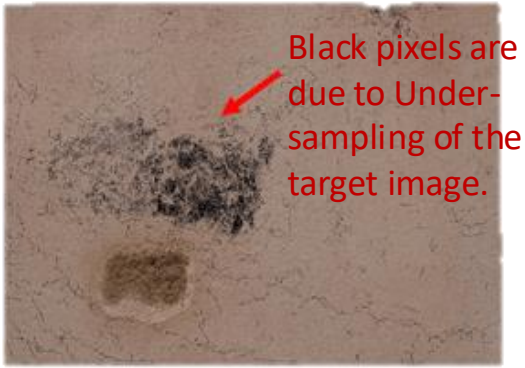


Target image

Reference image



Optical flow



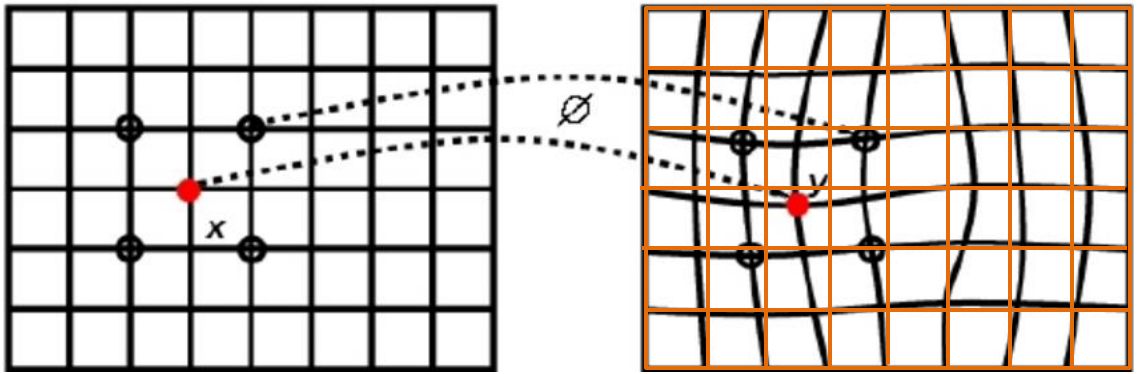
Forward warping

Backward warping

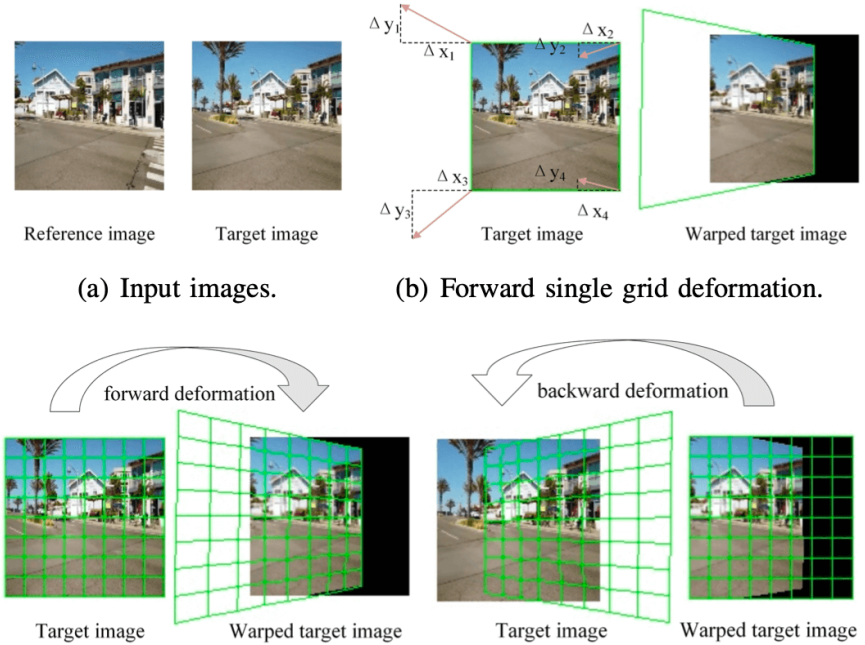
[Yu et al., Split-Attention Multiframe Alignment Network for Image Restoration, IEEE Access, 2020]

Forward Warping - Problems

- Target image is oversampled for converging flows
- Target image is undersampled for diverging flows



Forward vs. Backward Warping

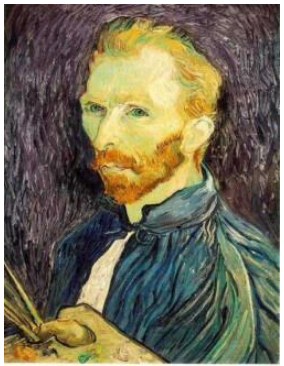


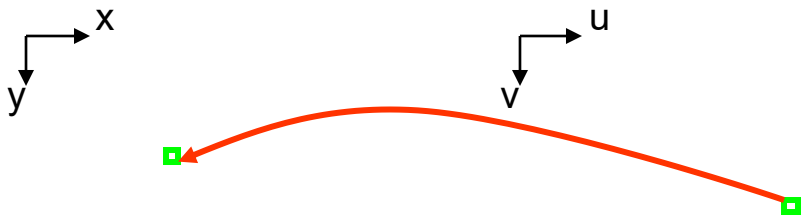
(a) Input images. (b) Forward single grid deformation.

(c) Forward multi-grid deformation. (d) Backward multi-grid deformation.

Fig. 6. Comparison between the forward deformation and backward deformation.

Inverse/Backward Warping





S(x,y)

T(u,v)

Inverse warping algorithm:

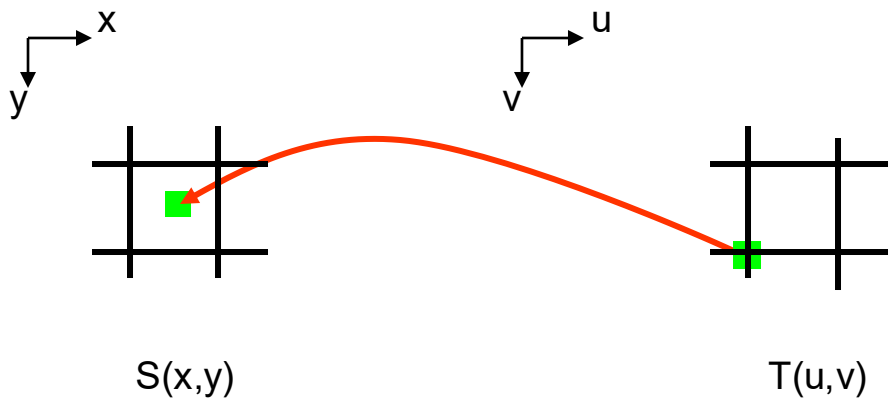
for $v = v_{min}$ to v_{max}

for $u = u_{min}$ to u_{max}

$x = F(u,v)$; $y = G(u,v)$

copy pixel at source S(x,y) to T(u,v)

Image Warping

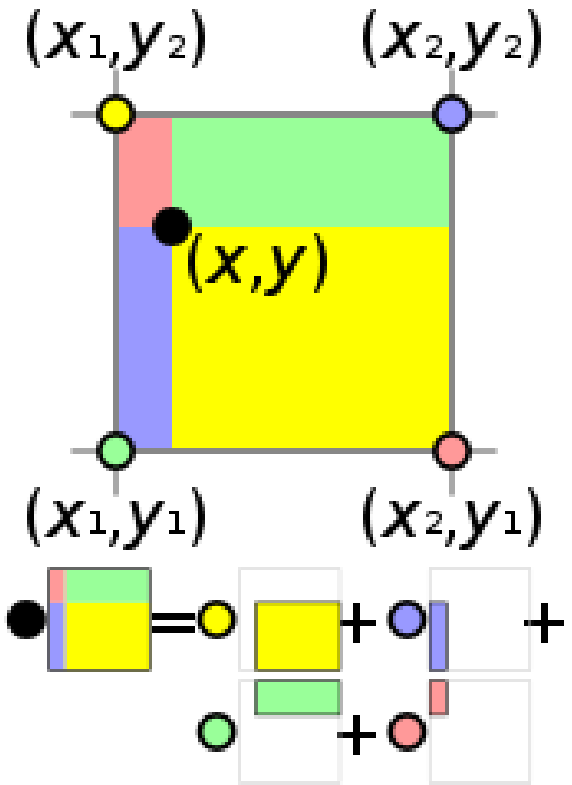


Q: What if pixel comes from “between” two pixels?

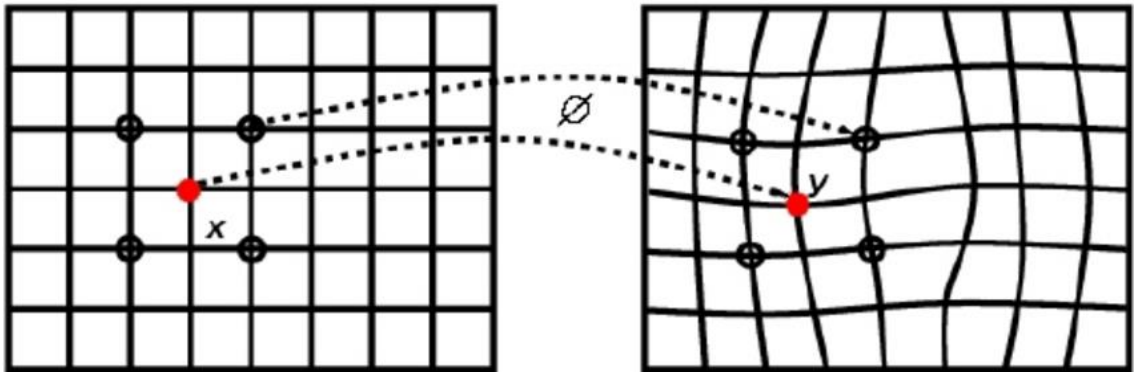
A: Interpolate color values from neighboring pixels

Bilinear Interpolation

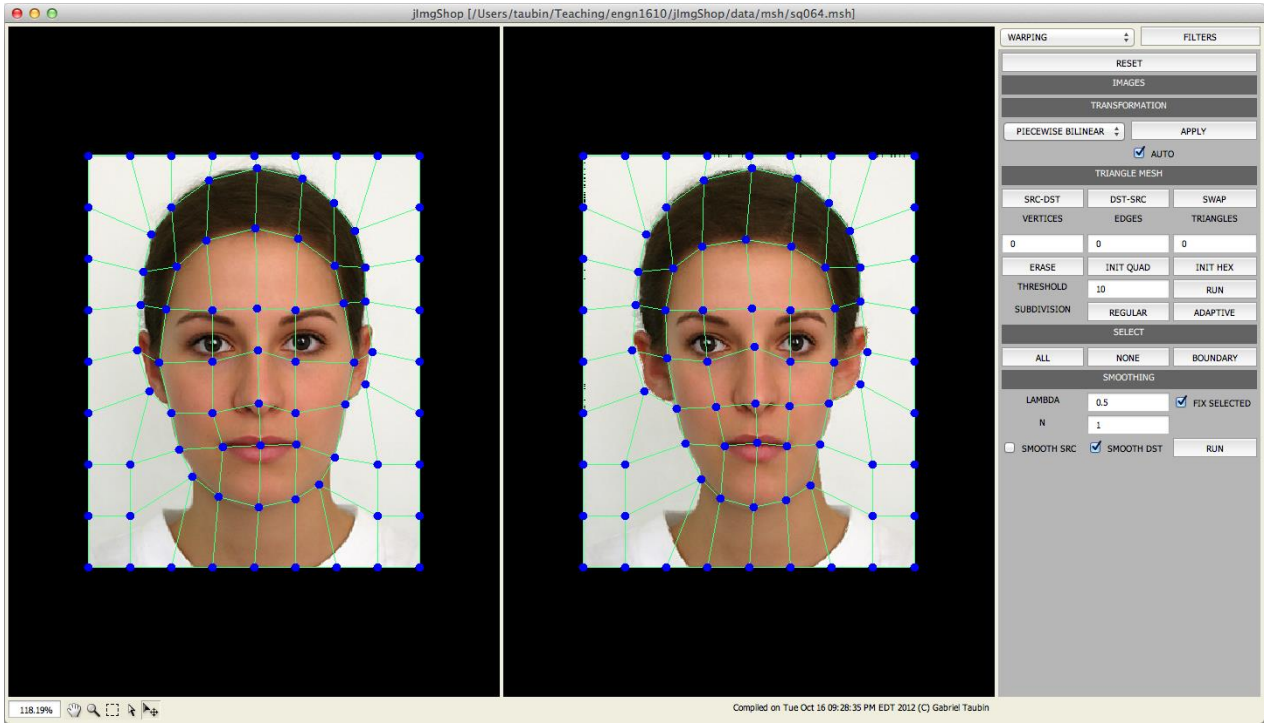
- **Goal:** Given an image with colors defined on integer grid locations, how to compute the color at a sub-pixel coordinate (x,y)?
- **Approach:** The color of a point between discrete grid locations is defined as the weighted linear interpolation of the colors of the four pixel neighbors.
  The weights are according to the area of the rectangle between point (x,y) and the corresponding neighbor.



Piece-wise Linear Warping



Piece-wise Linear Warping



Piece-wise Linear Warping

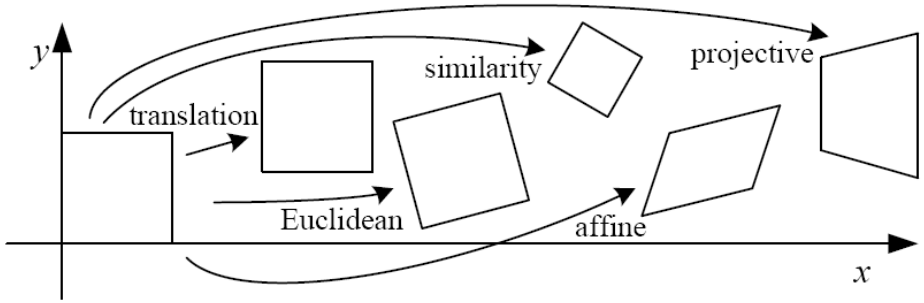
n point image deformation

Iterative ARAP
(has problems with stretching and shrinking)

[Source: Youtube]

Today's Agenda

- **Global operations**
  - Global Transformations
  - Image Warping
  - **Image Morphing**
- Image Stitching
  - Transformation computation
    - Reprojection model
    - Key point matching
    - RANSAC
    - Solving transformation
  - Planar mapping
  - Blending
    - Feathering
    - Multi-band blending
    - Further improvements



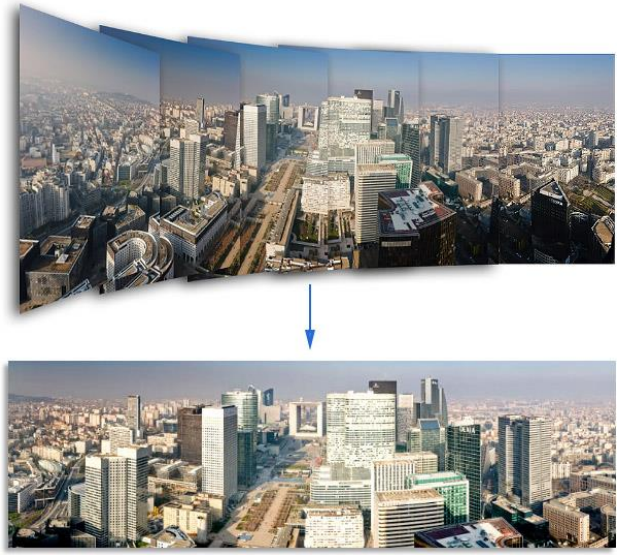


Image Morphing

- 1991 – First photorealistic face morping
- Michael Jackson – Black or White (official music video)
- Lots of manual work: corresponding points for eyes, mouth, etc. where manually selected
- Back then: 4-5h of computing time for a few frames of a single morph transition

[https://computeranimationhistory-cgi.jimdofree.com/michael-jackson-black-or-white-1991]

Image Morphing

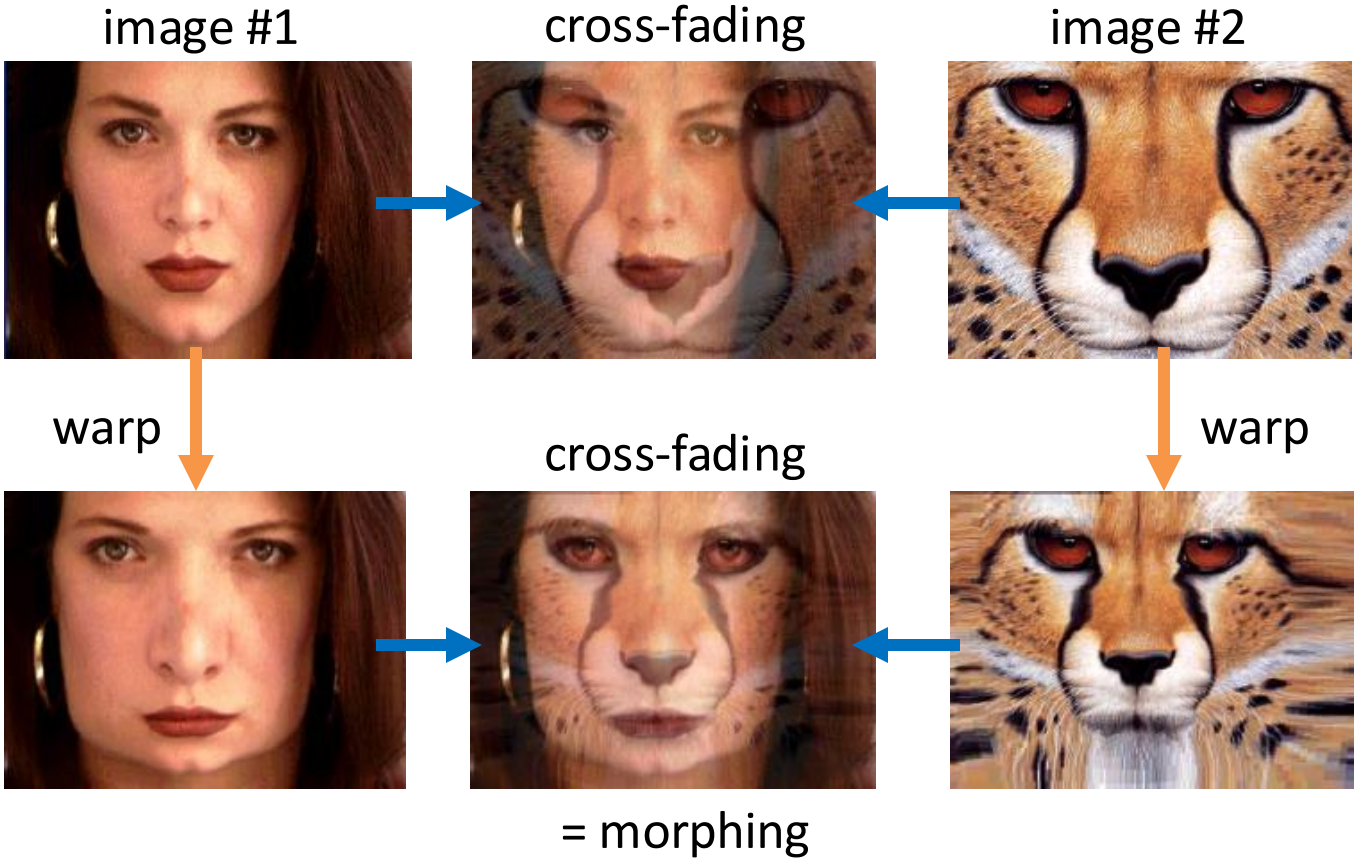


Image Morphing

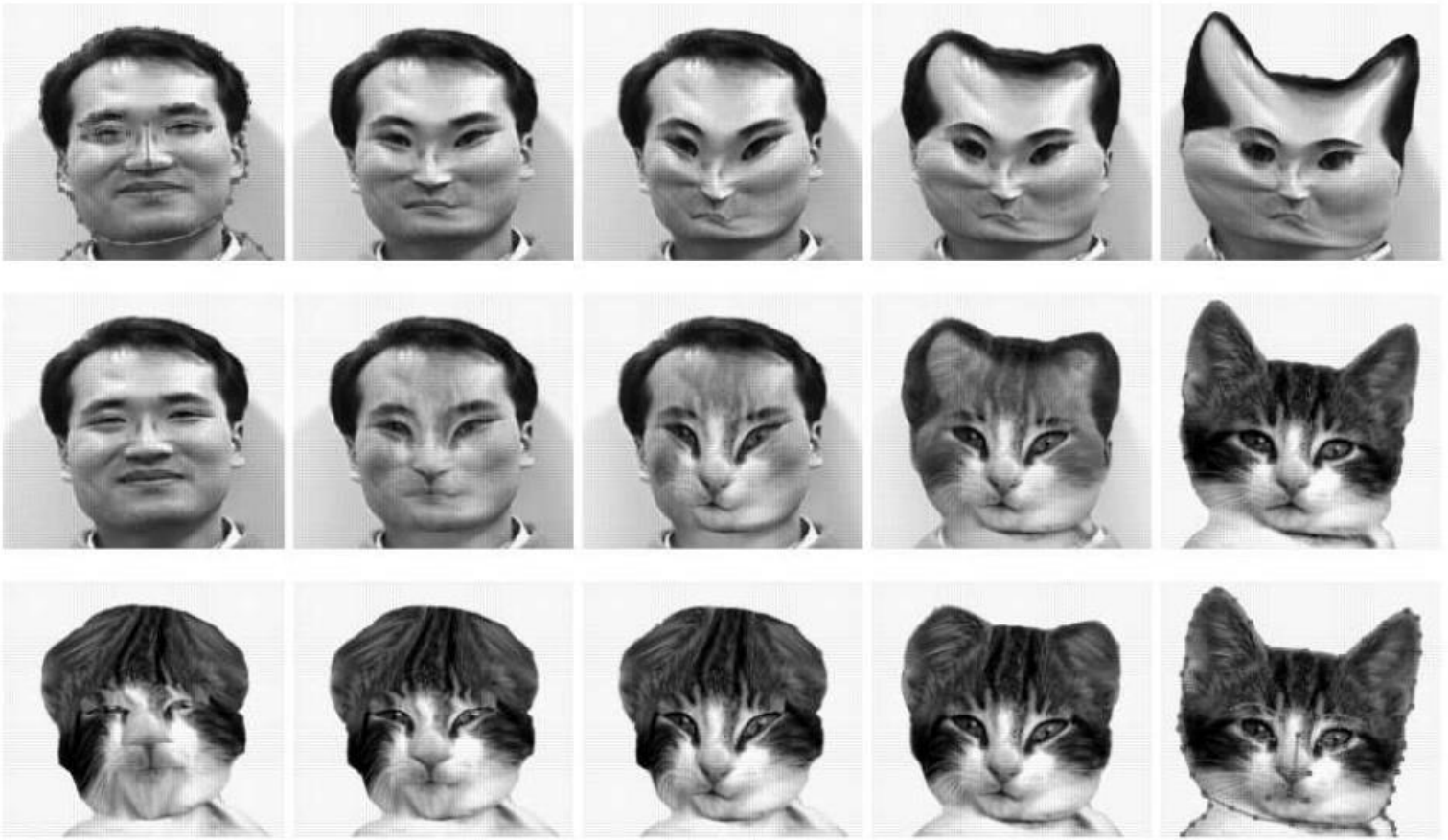
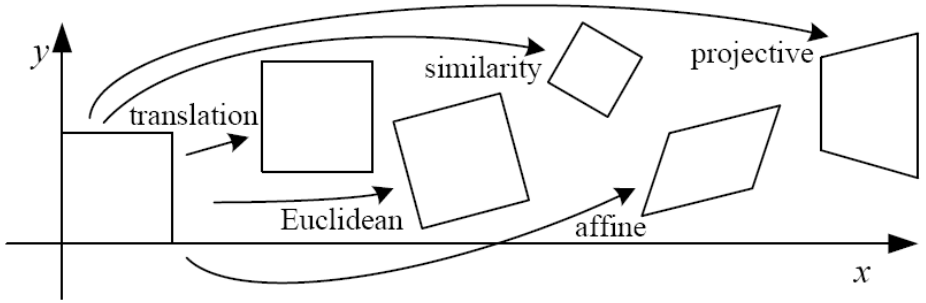


Image Morphing

Today's Agenda

- Global operations
  - Global Transformations
  - Image Warping
  - Image Morphing
- **Image Stitching**
  - Transformation computation
    - Reprojection model
    - Key point matching
    - RANSAC
    - Solving transformation
  - Planar mapping
  - Blending
    - Feathering
    - Multi-band blending
    - Further improvements



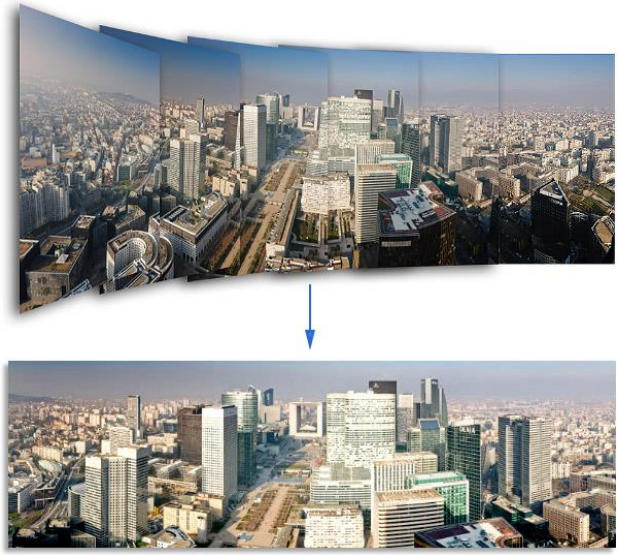


Image Stitching

Goal: Combine two or more overlapping images into one large image.

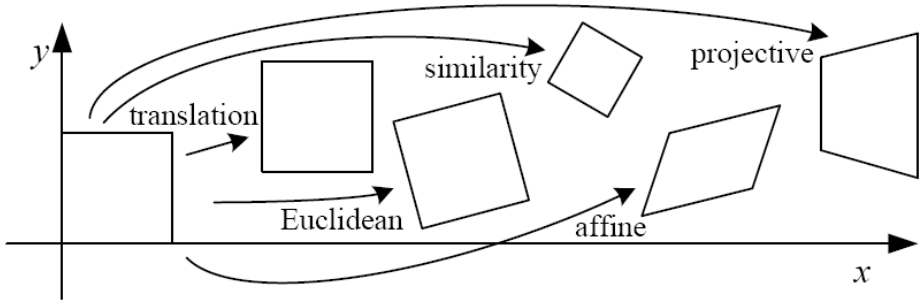
Typical Steps:

- Transformation computation
  - Reprojection model
  - Key point matching
  - Solving transformation
  - RANSAC
- Planar mapping
- Blending

[Image source: https://panoramic-photo-guide.com/best-photo-stitching-software-to-beginners.html]

Today's Agenda

- Global operations
  - Global Transformations
  - Image Warping
  - Image Morphing
- **Image Stitching**
  - **Transformation computation**
    - **Reprojection model**
    - Key point matching
    - RANSAC
    - Solving transformation
  - Planar mapping
  - Blending
    - Feathering
    - Multi-band blending
    - Further improvements



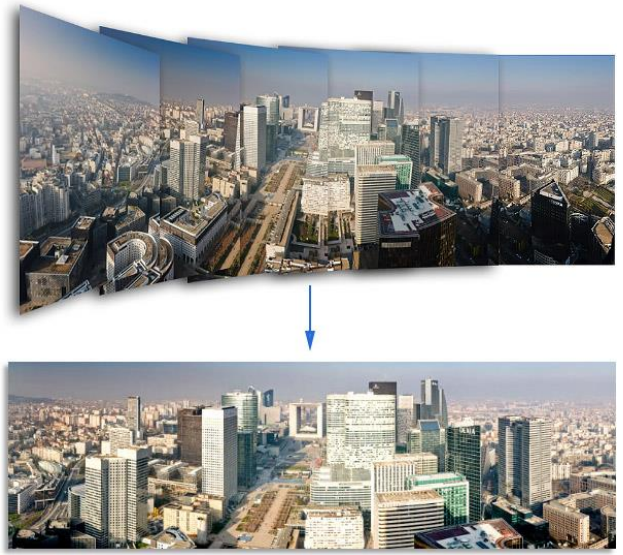
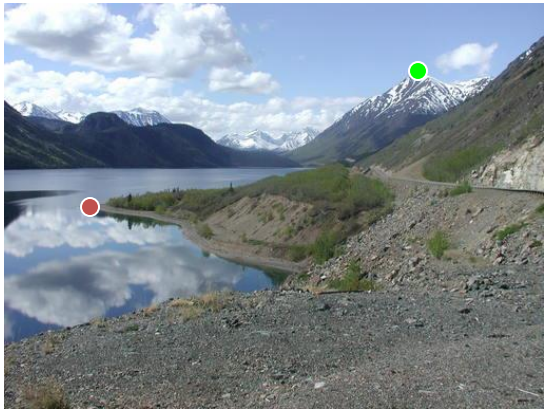


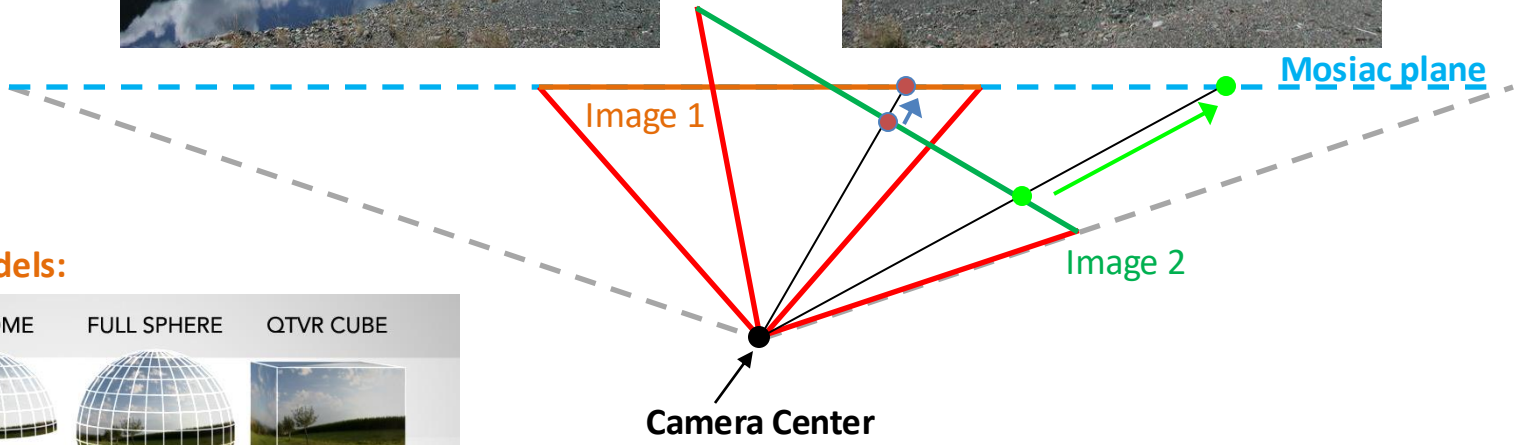
Image Reprojection Model

Image 1

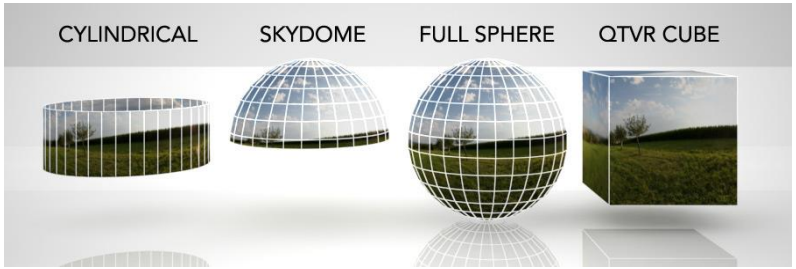
Image 2



Planar projection:

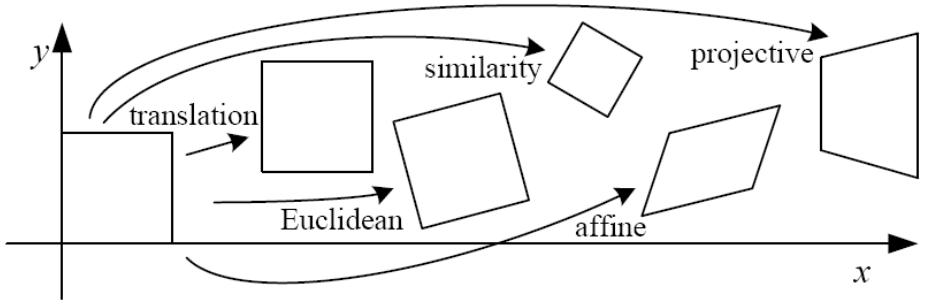


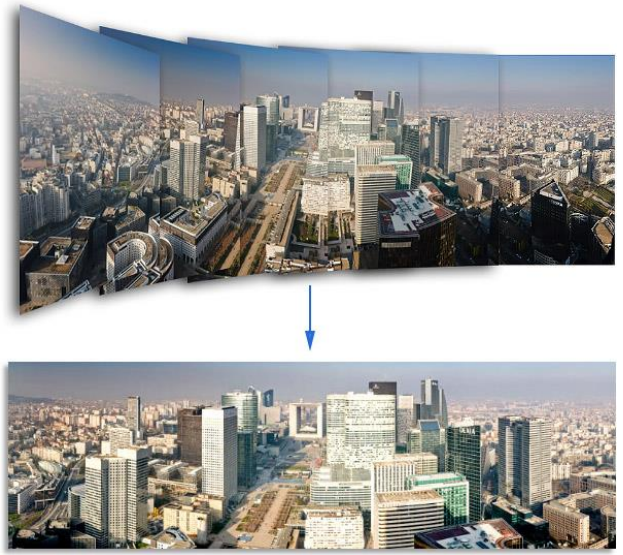
Other projection models:



Today's Agenda

- Global operations
  - Global Transformations
  - Image Warping
  - Image Morphing
- **Image Stitching**
  - **Transformation computation**
    - Reprojection model
    - **Key point matching**
    - RANSAC
    - Solving transformation
  - Planar mapping
  - Blending
    - Feathering
    - Multi-band blending
    - Further improvements





Keypoint Matching

Input Images

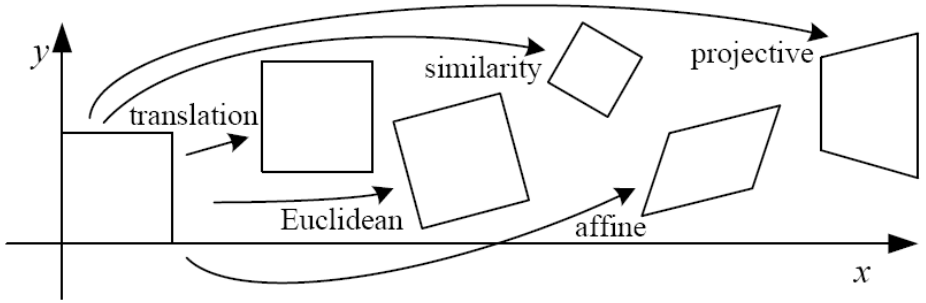
SIFT Features

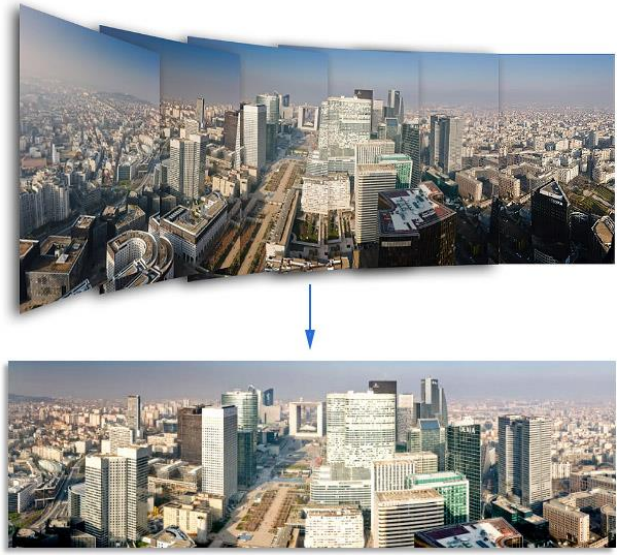
Nearest Neighbor Matches

[Image source: Cansin Yildiz]

Today's Agenda

- Global operations
  - Global Transformations
  - Image Warping
  - Image Morphing
- **Image Stitching**
  - **Transformation computation**
    - Reprojection model
    - Key point matching
    - **RANSAC**
    - Solving transformation
  - Planar mapping
  - Blending
    - Feathering
    - Multi-band blending
    - Further improvements





RANSAC

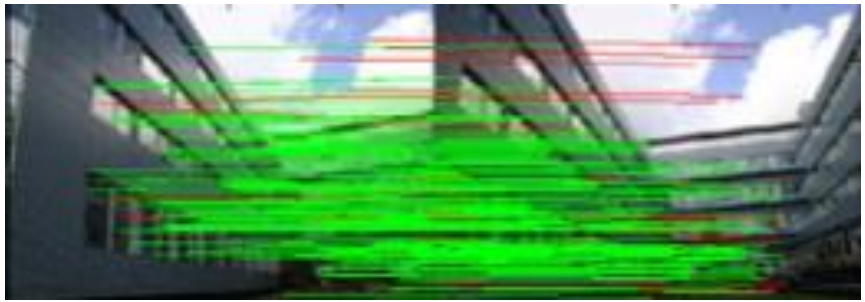
- Describe a Homography:

  Points from images a and b are related by $p_b = H_{ab}\, p_a$

Nearest Neighbor Matches

Iterate:

1. Randomly select four feature pairs
2. Compute exact homography H
3. Compute inliers: $\mathbf{dist}(\boldsymbol{p_b}, \boldsymbol{H_{ab}\, p_a}) < \boldsymbol{\epsilon}$
4. Keep largest set of of inliers
5. Compute least-squares homography estimate on all inliers

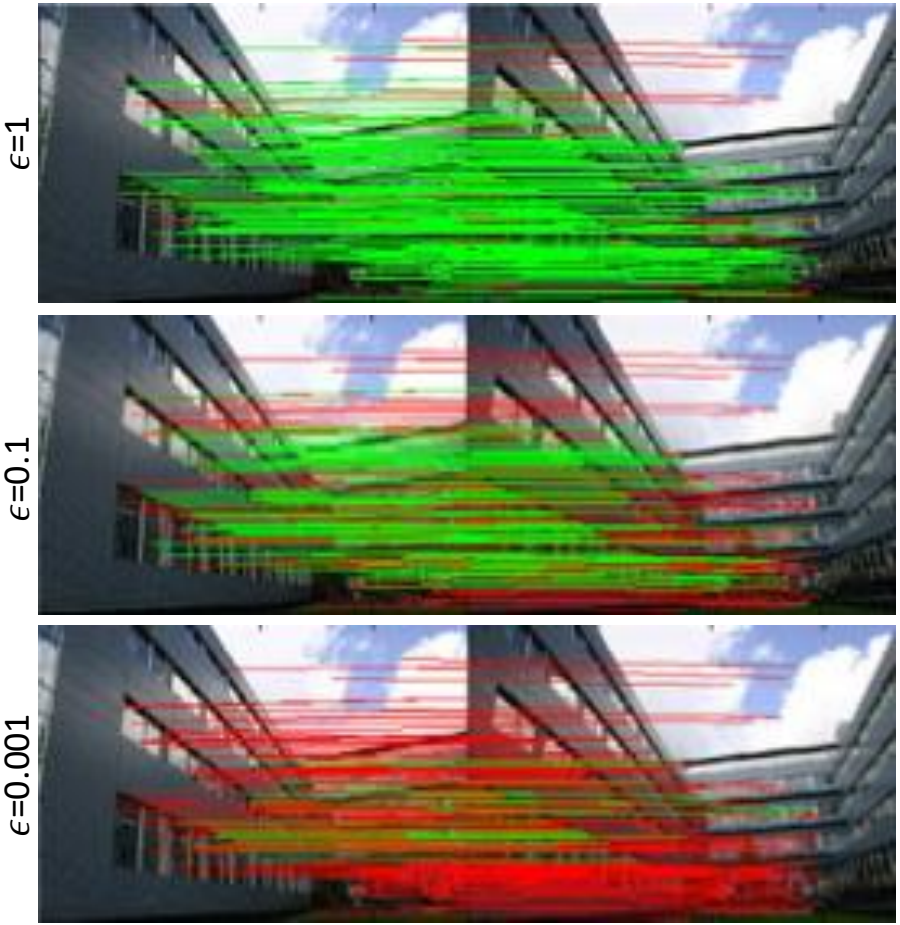


Inliers and Outliers

[Image source: Cansin Yildiz]

RANSAC

- Various inlier thresholds ϵ:



Iterate:

1. Randomly select four feature pairs
2. Compute exact homography H
3. Compute inliers: $\mathbf{dist}(\boldsymbol{p_b}, \boldsymbol{H_{ab}}\, \boldsymbol{p_a}) < \boldsymbol{\epsilon}$
4. Keep largest set of of inliers
5. Compute least-squares homography estimate on all inliers

[Image source: Cansin Yildiz]

Final Stitching Result

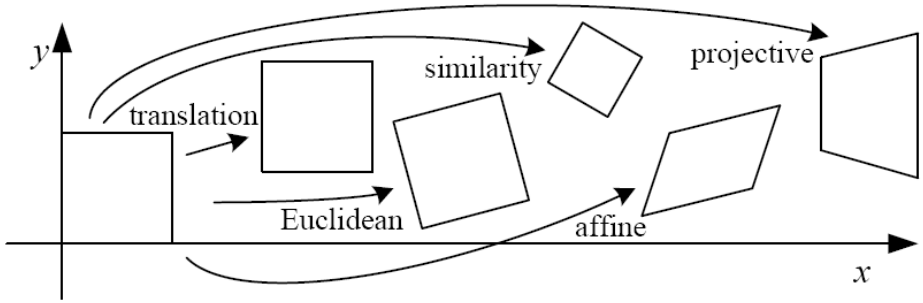
Input Images

Stitched Images

[Image source: Cansin Yildiz]

Today's Agenda

- Global operations
  - Global transformations
  - Image Warping
  - Image Morphing
- **Image Stitching**
  - **Transformation computation**
    - Reprojection model
    - Key point matching
    - RANSAC
    - **Solving transformation**
  - Planar mapping
  - Blending
    - Feathering
    - Multi-band blending
    - Further improvements



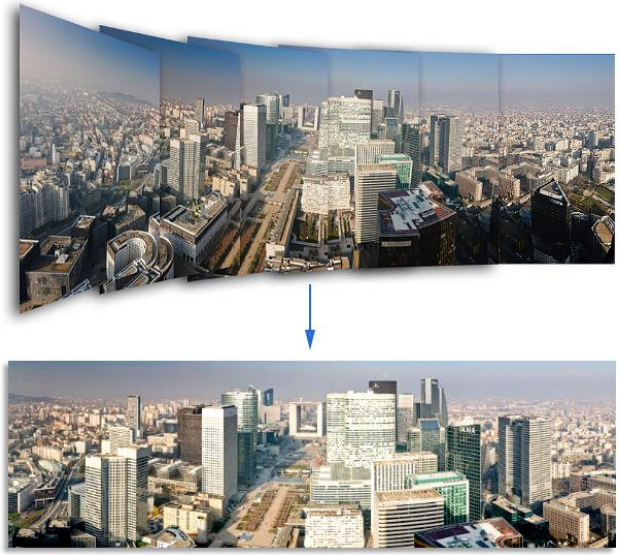
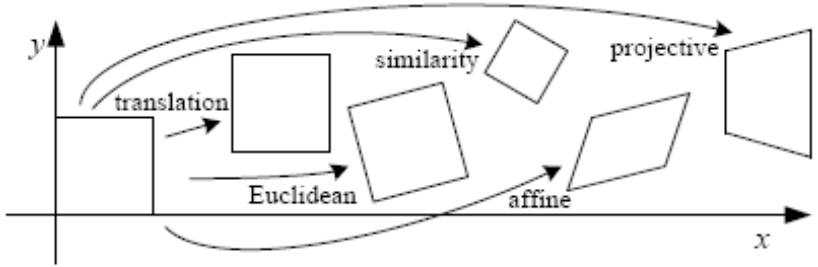
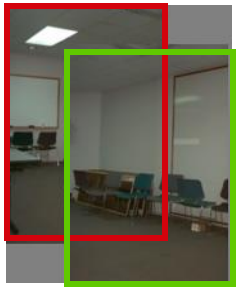
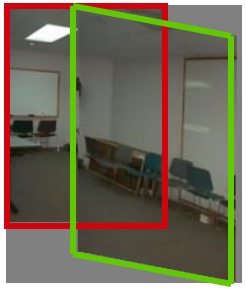
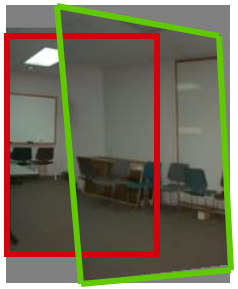
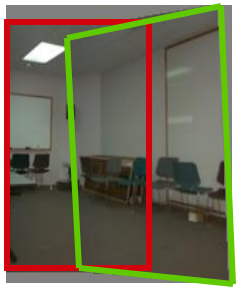
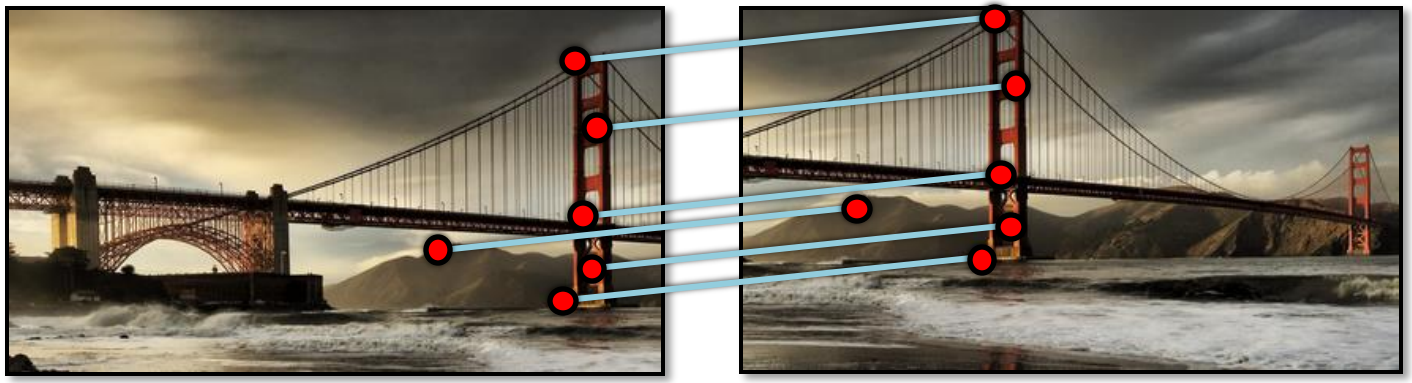


Image Transformation Models



2D Translation	2D Affine	3D Perspective	3D rotation
			
2 unknowns	6 unknowns	8 unknowns	3 unknowns

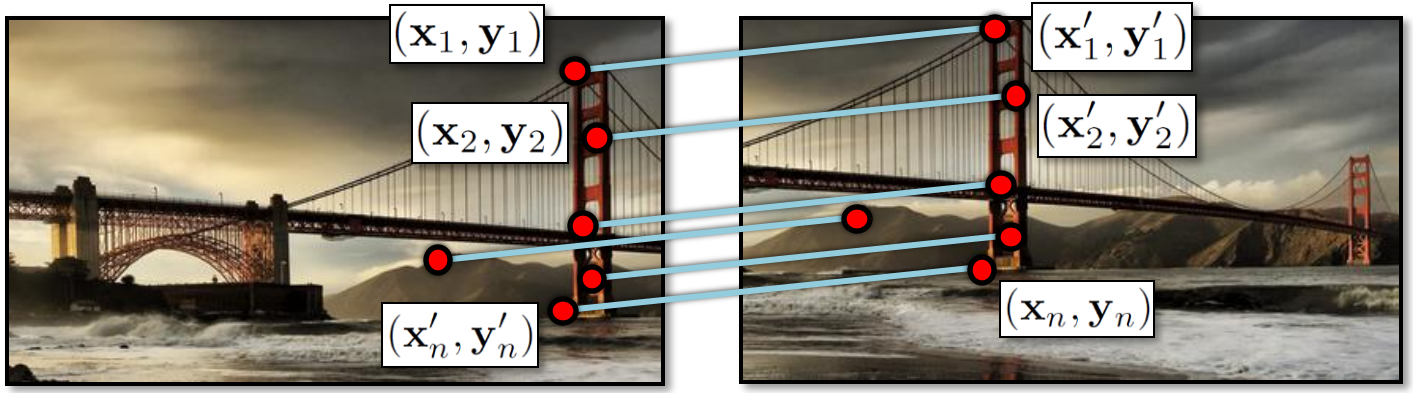
Simple Case: (I) Translation



$(\mathbf{x}_t, \mathbf{y}_t)$

How do we solve for $(\mathbf{x}_t, \mathbf{y}_t)$?

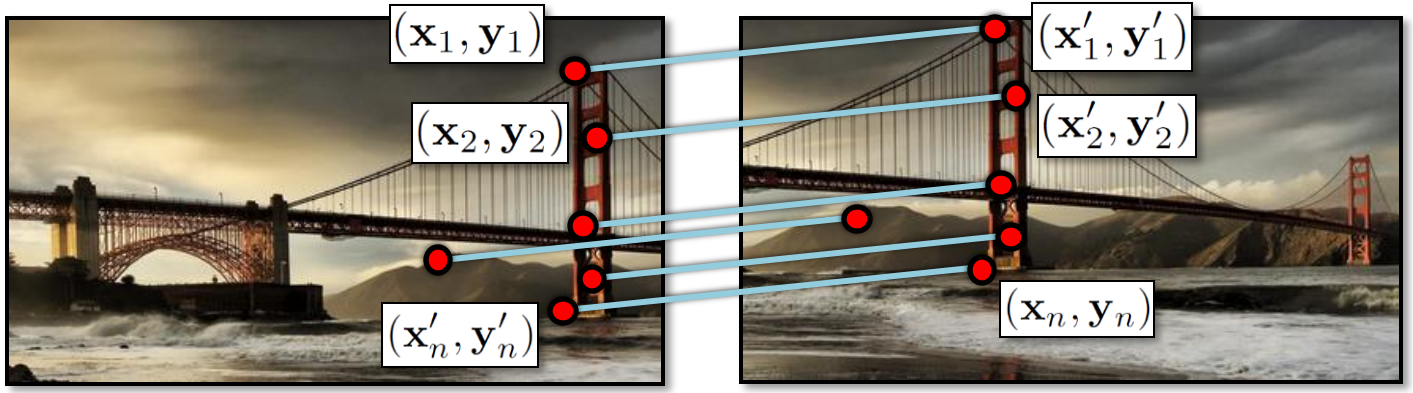
Simple Case: (I) Translation



Displacement of match *i* = $\left(\mathbf{x}'_i - \mathbf{x}_i, \mathbf{y}'_i - \mathbf{y}_i\right)$

Mean displacement $\left(\mathbf{x}_t, \mathbf{y}_t\right) = \left(\frac{1}{n}\sum_{i=1}^{n} \mathbf{x}'_i - \mathbf{x}_i, \frac{1}{n}\sum_{i=1}^{n} \mathbf{y}'_i - \mathbf{y}_i\right)$

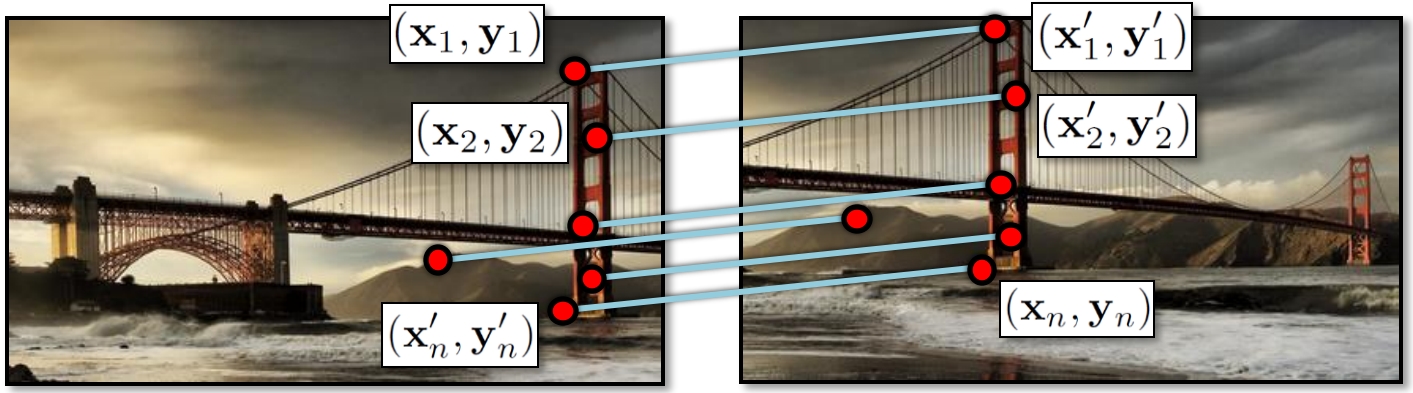
Simple Case: (I) Translation



$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$
$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- System of linear equations
  - What are the knowns? Unknowns?
  - How many unknowns? How many equations (per match)?

Simple Case: (I) Translation



$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$
$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- Problem: more equations than unknowns
  - "Overdetermined" system of equations
  - We will find the *least squares* solution

Least squares formulation

- For each point $(\mathbf{x}_i, \mathbf{y}_i)$

$$\begin{aligned} \mathbf{x}_i + \mathbf{x_t} &= \mathbf{x}'_i \\ \mathbf{y}_i + \mathbf{y_t} &= \mathbf{y}'_i \end{aligned}$$

- we define the *residuals* as

$$\begin{aligned} r_{\mathbf{x}_i}(\mathbf{x}_t) &= (\mathbf{x}_i + \mathbf{x}_t) - \mathbf{x}'_i \\ r_{\mathbf{y}_i}(\mathbf{y}_t) &= (\mathbf{y}_i + \mathbf{y}_t) - \mathbf{y}'_i \end{aligned}$$

Least squares formulation

- Goal: minimize sum of squared residuals

$$C(\mathbf{x}_t, \mathbf{y}_t) = \sum_{i=1}^{n} \left(r_{\mathbf{x}_i}(\mathbf{x}_t)^2 + r_{\mathbf{y}_i}(\mathbf{y}_t)^2 \right)$$

- “Least squares” solution
- For translations, is equal to mean displacement

Least squares

- Find **t** that minimizes $\mathbf{At} = \mathbf{b}$

$$||\mathbf{At} - \mathbf{b}||^2$$

- To solve, form the *normal equations*

$$\mathbf{A}^{\mathrm{T}}\mathbf{At} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$$

$$\mathbf{t} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

Solving for (I) Translations

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \vdots & \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ x'_2 - x_2 \\ y'_2 - y_2 \\ \vdots \\ x'_n - x_n \\ y'_n - y_n \end{bmatrix}$$

$$\underset{2n \text{ x } 2}{\mathbf{A}} \quad \underset{2 \text{ x } 1}{\mathbf{t}} = \underset{2n \text{ x } 1}{\mathbf{b}}$$

Solving for (II) Affine Transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- How many unknowns?
- How many equations per match?
- How many matches do we need?

Solving for (II) Affine Transformations

- Residuals:

$$
\begin{aligned}
r_{x_i}(a,b,c,d,e,f) &= (ax_i + by_i + c) - x'_i \\
r_{y_i}(a,b,c,d,e,f) &= (dx_i + ey_i + f) - y'_i
\end{aligned}
$$

- Cost function:

$$
C(a,b,c,d,e,f) = \sum_{i=1}^{n} \left(r_{x_i}(a,b,c,d,e,f)^2 + r_{y_i}(a,b,c,d,e,f)^2 \right)
$$

Solving for (II) Affine Transformations

- Matrix form

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ & & & \vdots & & \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}$$

$$\underset{2n \text{ x } 6}{\mathbf{A}} \qquad \underset{6 \text{ x } 1}{\mathbf{t}} = \underset{2n \text{ x } 1}{\mathbf{b}}$$

Solving for (III) Homographies

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x_i' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y_i' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$\begin{aligned} x_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

Solving for (III) Homographies

$$
\begin{aligned}
x_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\
y_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12}
\end{aligned}
$$

$$
\begin{bmatrix}
x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\
0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i'
\end{bmatrix}
\begin{bmatrix}
h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22}
\end{bmatrix}
=
\begin{bmatrix}
0 \\ 0
\end{bmatrix}
$$

Direct Linear Transforms

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\underset{2n \times 9}{\mathbf{A}} \qquad \underset{9 \times 1}{\mathbf{h}} \qquad \underset{2n \times 1}{\mathbf{0}}$$

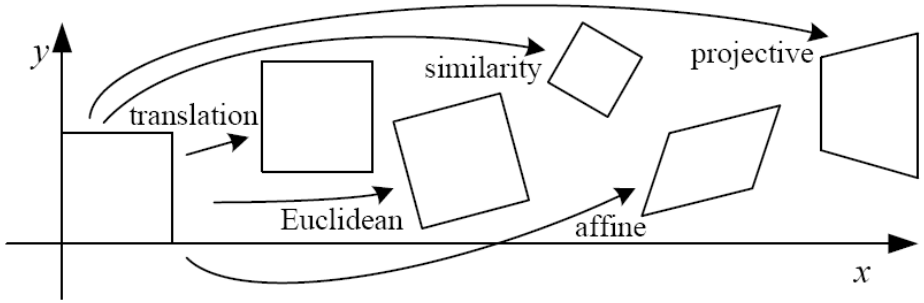
Defines a least squares problem: $\text{minimize } \|\mathbf{A}\mathbf{h} - 0\|^2$

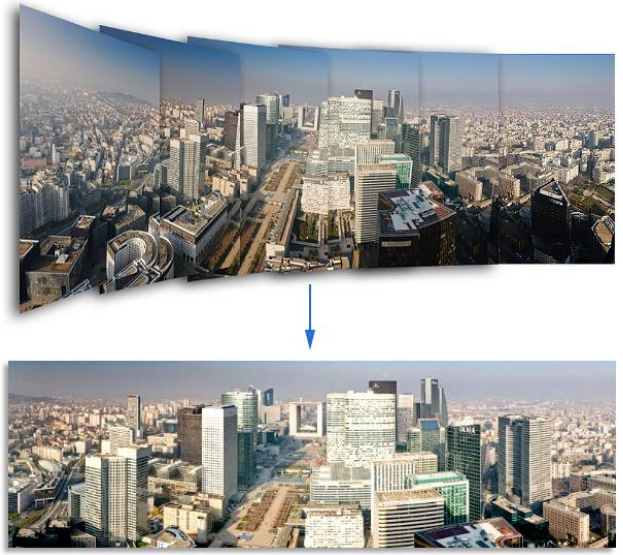
- Since $\mathbf{h}$ is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
- Works with 4 or more points

→ Solution for homography representing the projective camera transformation between two image planes

Today's Agenda

- Global operations
  - Global Transformations
  - Image Warping
  - Image Morphing
- **Image Stitching**
  - Transformation computation
    - Reprojection model
    - Key point matching
    - RANSAC
    - Solving transformation
  - **Planar mapping**
  - Blending
    - Feathering
    - Multi-band blending
    - Further improvements

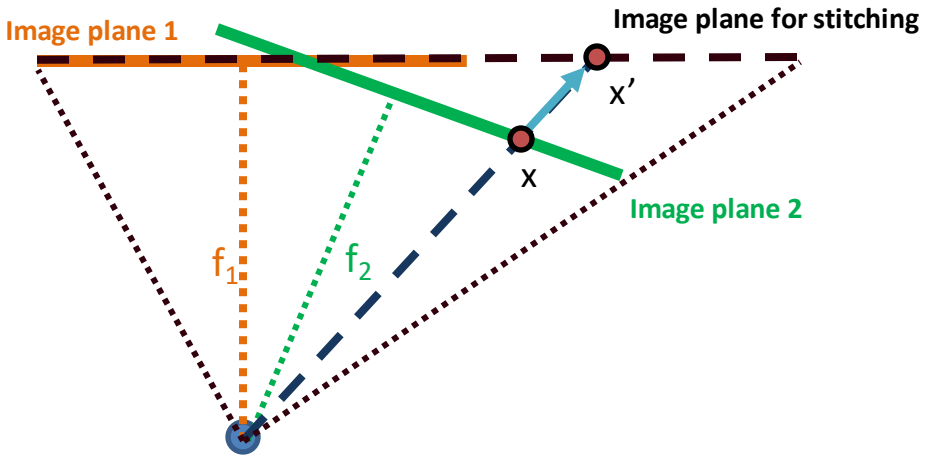




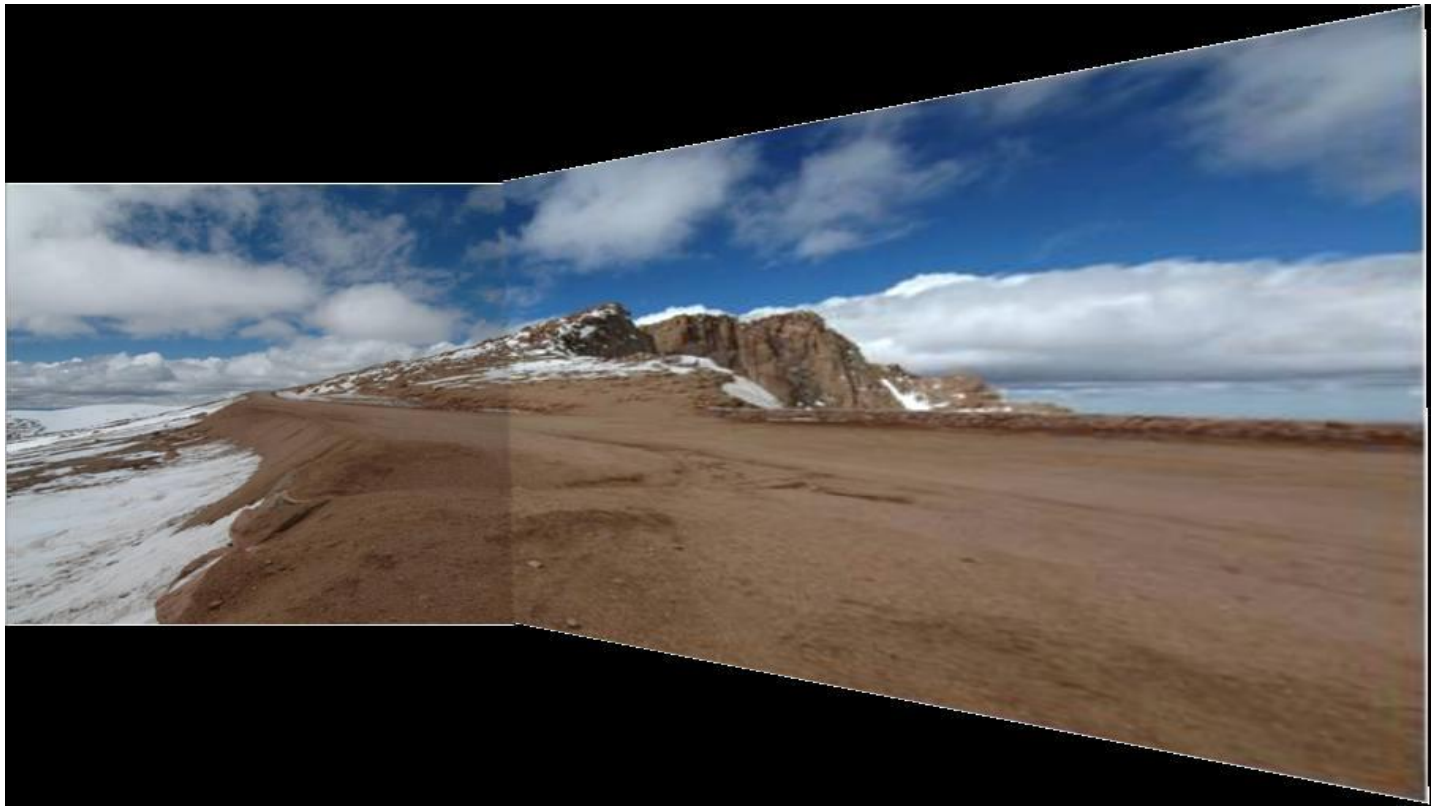
Planar Projection (Homography)

- **Goal:** Reproject several images into a new stitched image.
  Here, the image domain is **planar**, but could also be cylindrical, spherical, etc.

- **Approach:** projective mapping (homography)
  describes how coordinates
  from one image plane
  transform to coordinates in
  an other image plane.



Planar Projection (Homography)



[Photo by Russ Hewett]

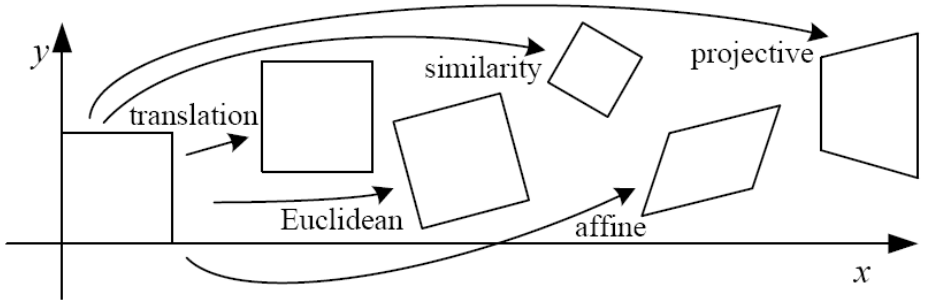
Planar Projection (Homography)

Planar

[Photo by Russ Hewett]

Today's Agenda

- Global operations
  - Global Transformations
  - Image Warping
  - Image Morphing
- Image Stitching
  - Transformation computation
    - Reprojection model
    - Key point matching
    - RANSAC
    - Solving transformation
  - Planar mapping
  - **Blending**
    - Feathering
    - Multi-band blending
    - Further improvements



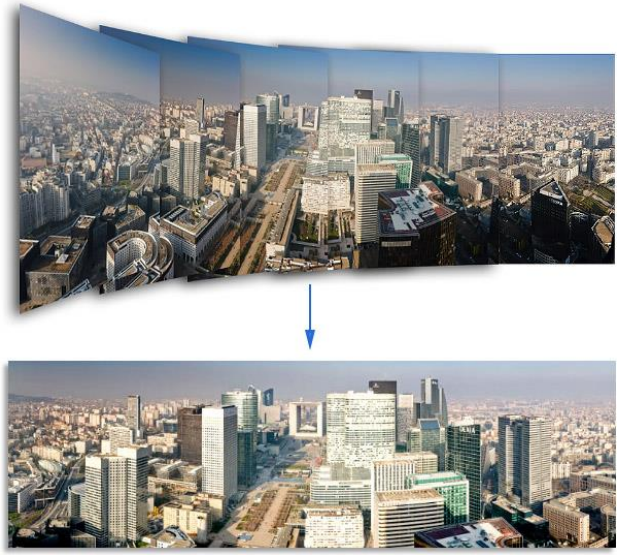
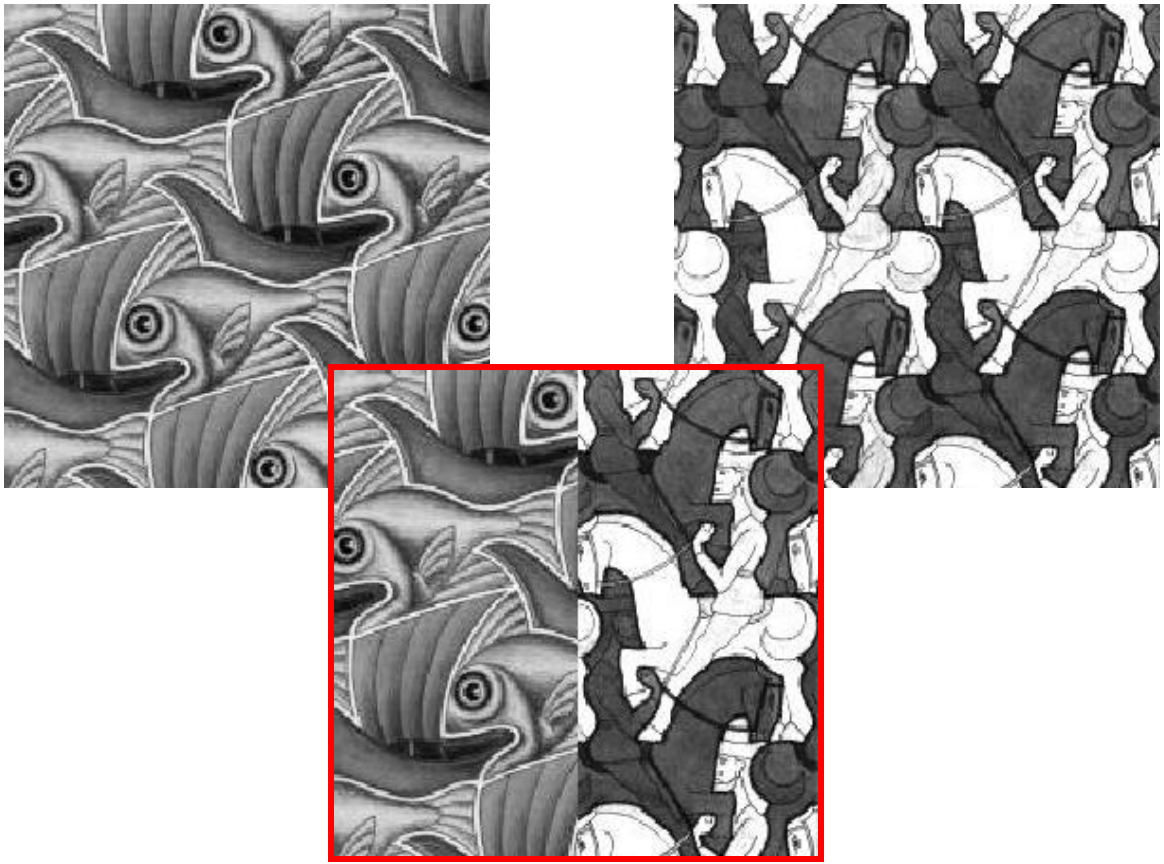
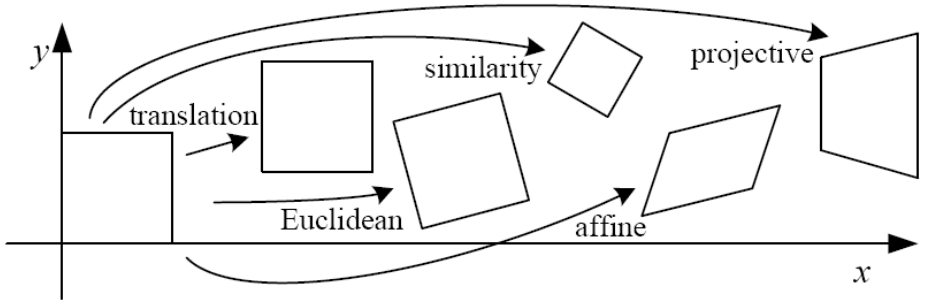


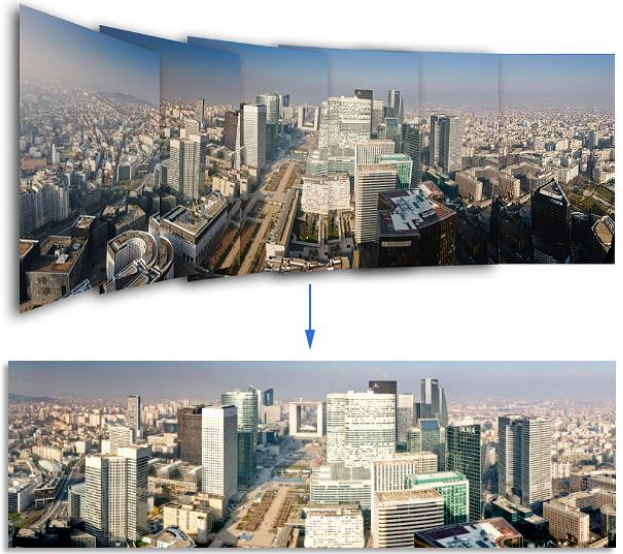
Image Blending



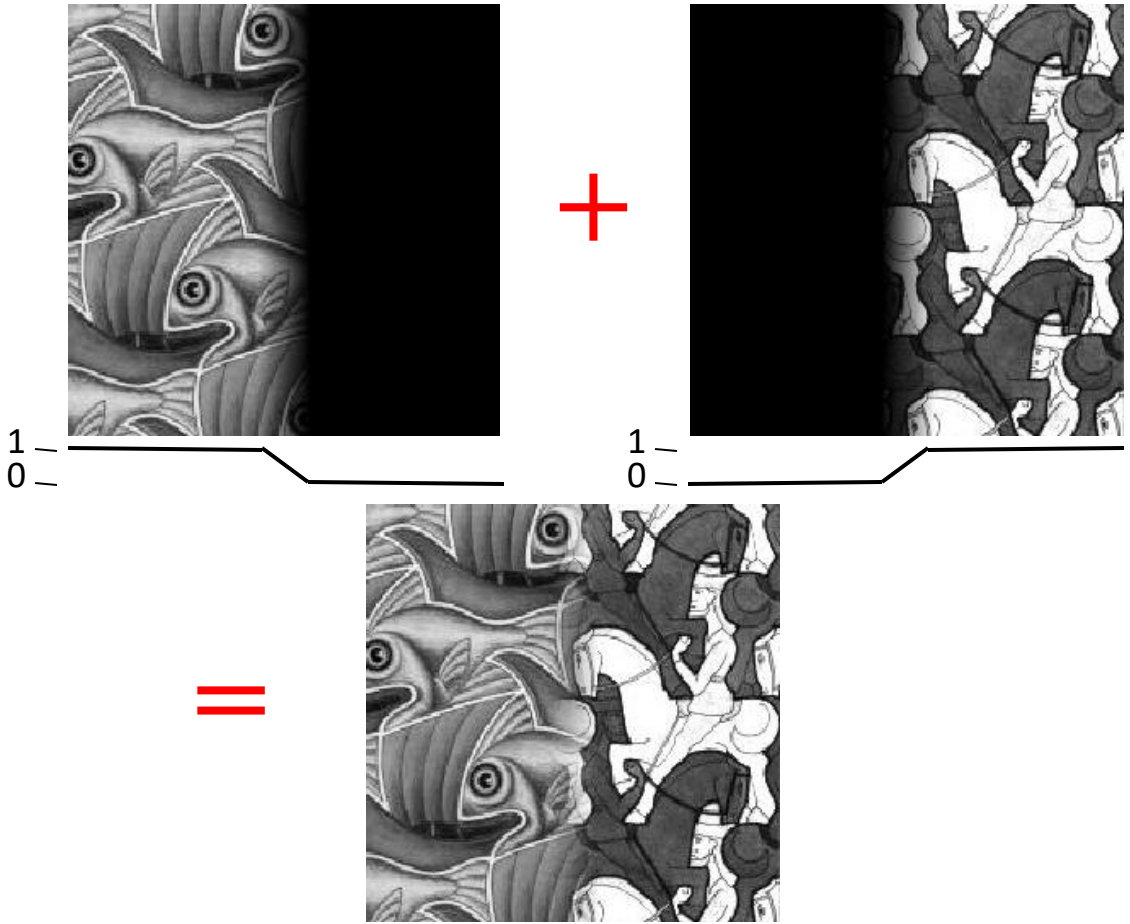
Today's Agenda

- Global operations
  - Global Transformations
  - Image Warping
  - Image Morphing
- Image Stitching
  - Transformation computation
    - Reprojection model
    - Key point matching
    - RANSAC
    - Solving transformation
  - Planar mapping
  - **Blending**
    - **Feathering**
    - Multi-band blending
    - Further improvements

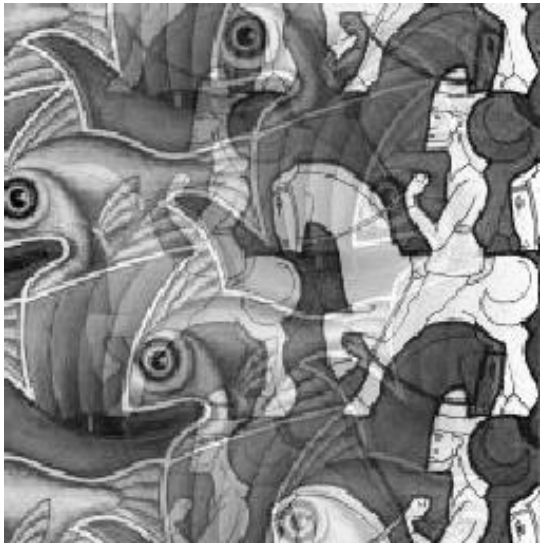


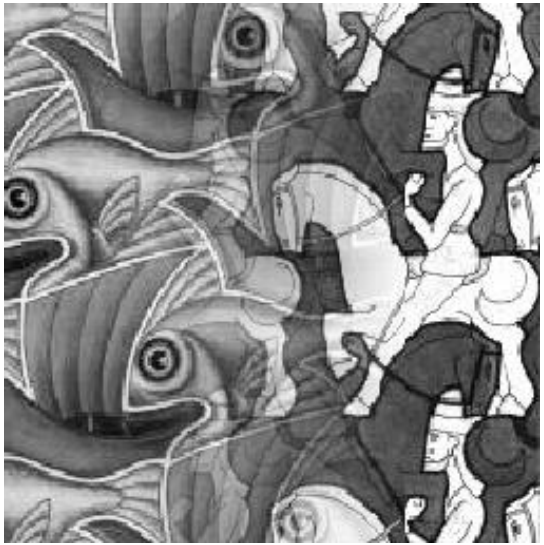


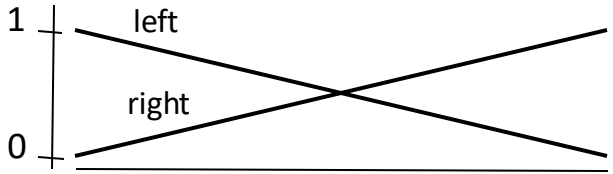
Feathering

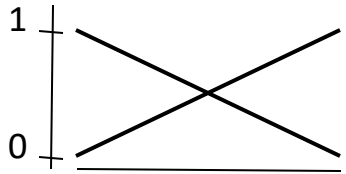


Effect of Window (ramp-width) Size



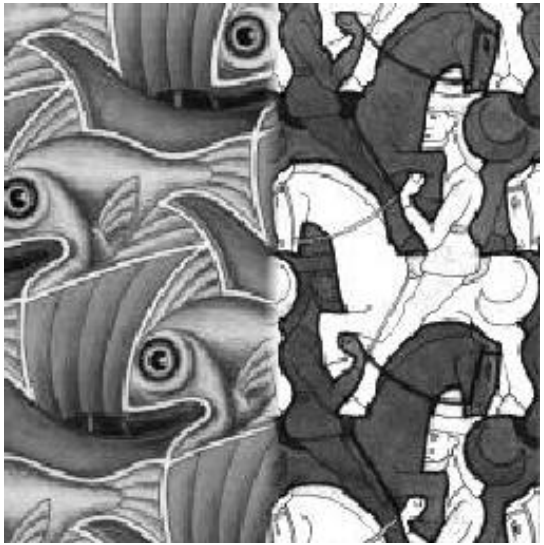


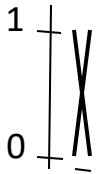


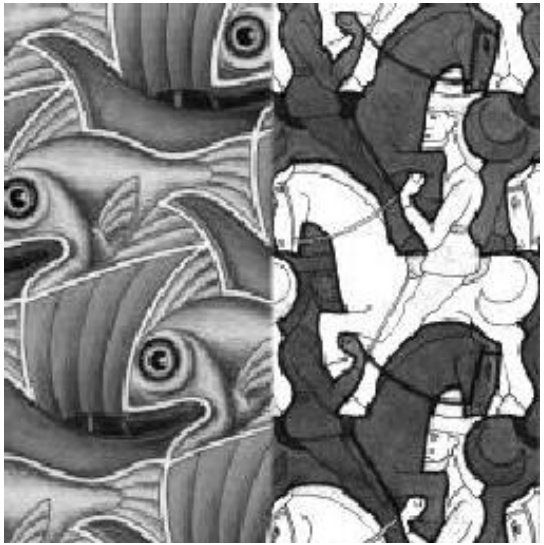


“cross-fading”

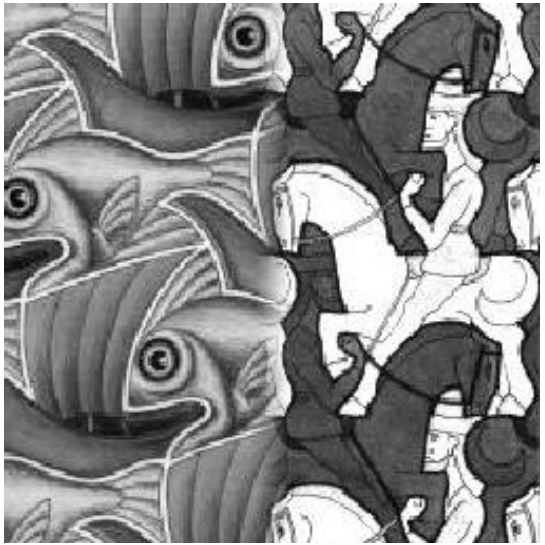
Effect of Window (ramp-width) Size

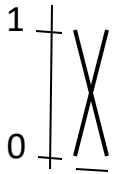






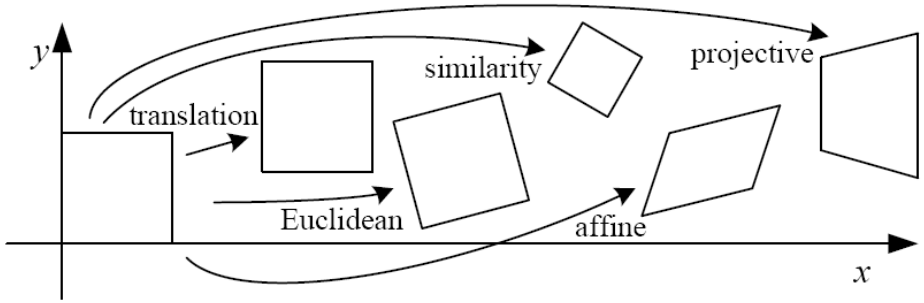
Good Window Size

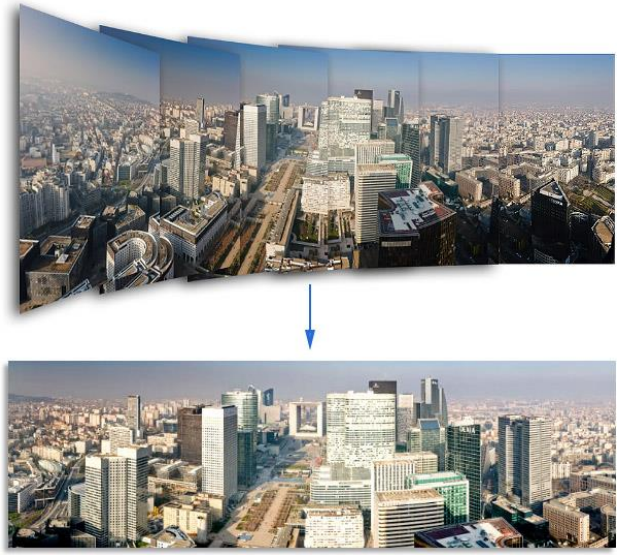




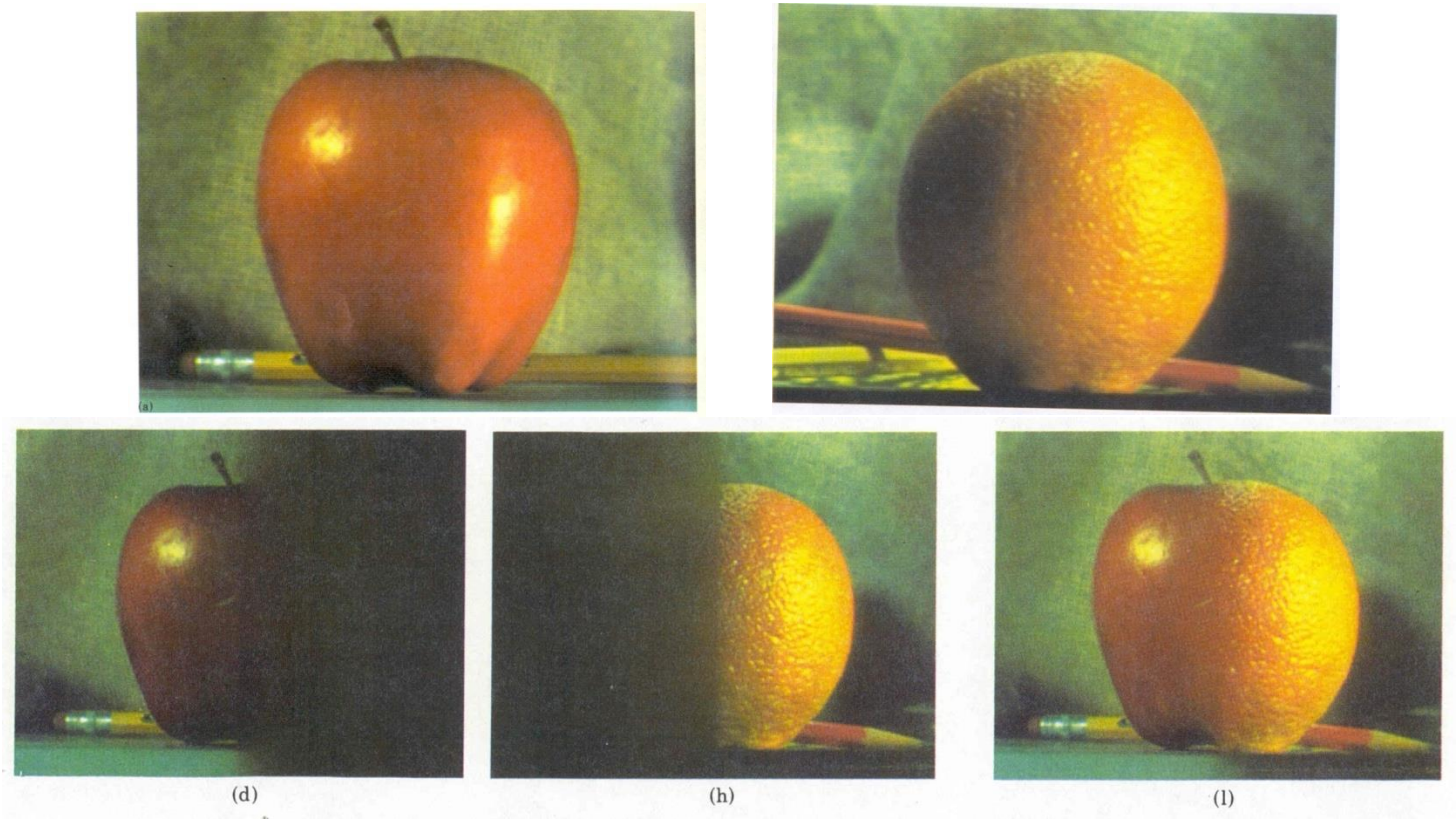
Today's Agenda

- Global operations
  - Global Transformations
  - Image Warping
  - Image Morphing
- Image Stitching
  - Transformation computation
    - Reprojection model
    - Key point matching
    - RANSAC
    - Solving transformation
  - Planar mapping
  - **Blending**
    - Feathering
    - **Multi-band blending**
    - Further improvements





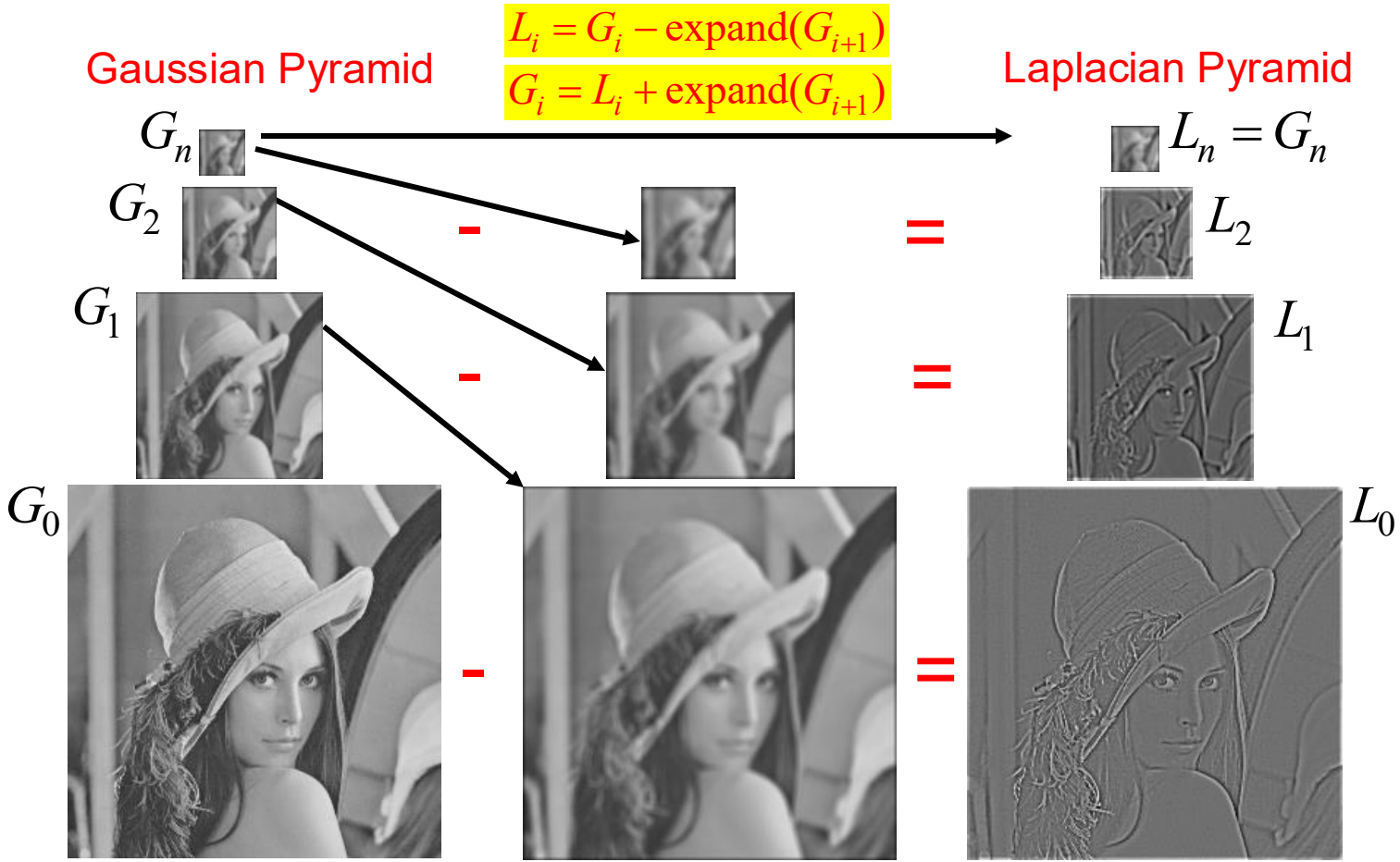
Multi-band Blending



Create a Laplacian pyramid, blend each level

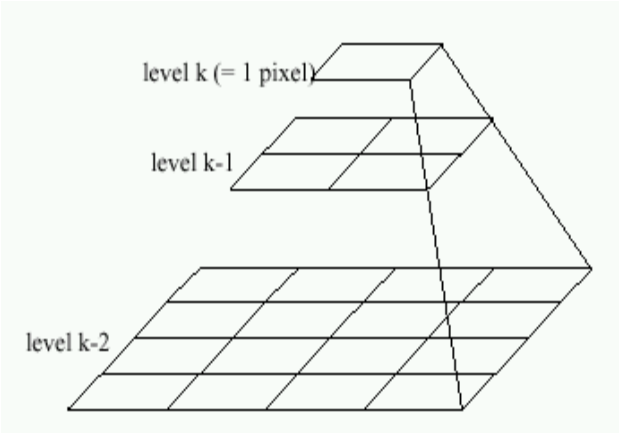
Burt, P. J. and Adelson, E. H., A multiresolution spline with applications to image mosaics, ACM Transactions on Graphics, 42(4), October 1983, 217-236.

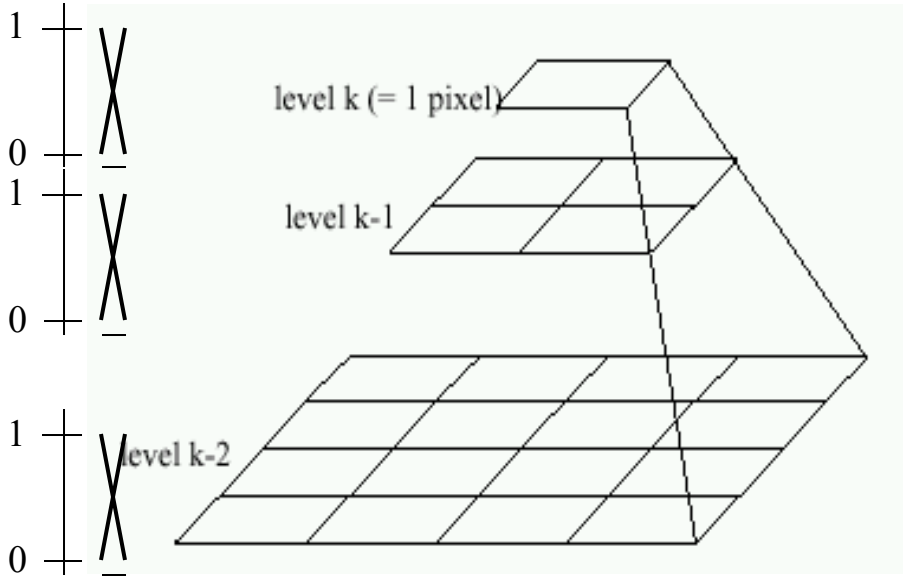
Laplacian Pyramid



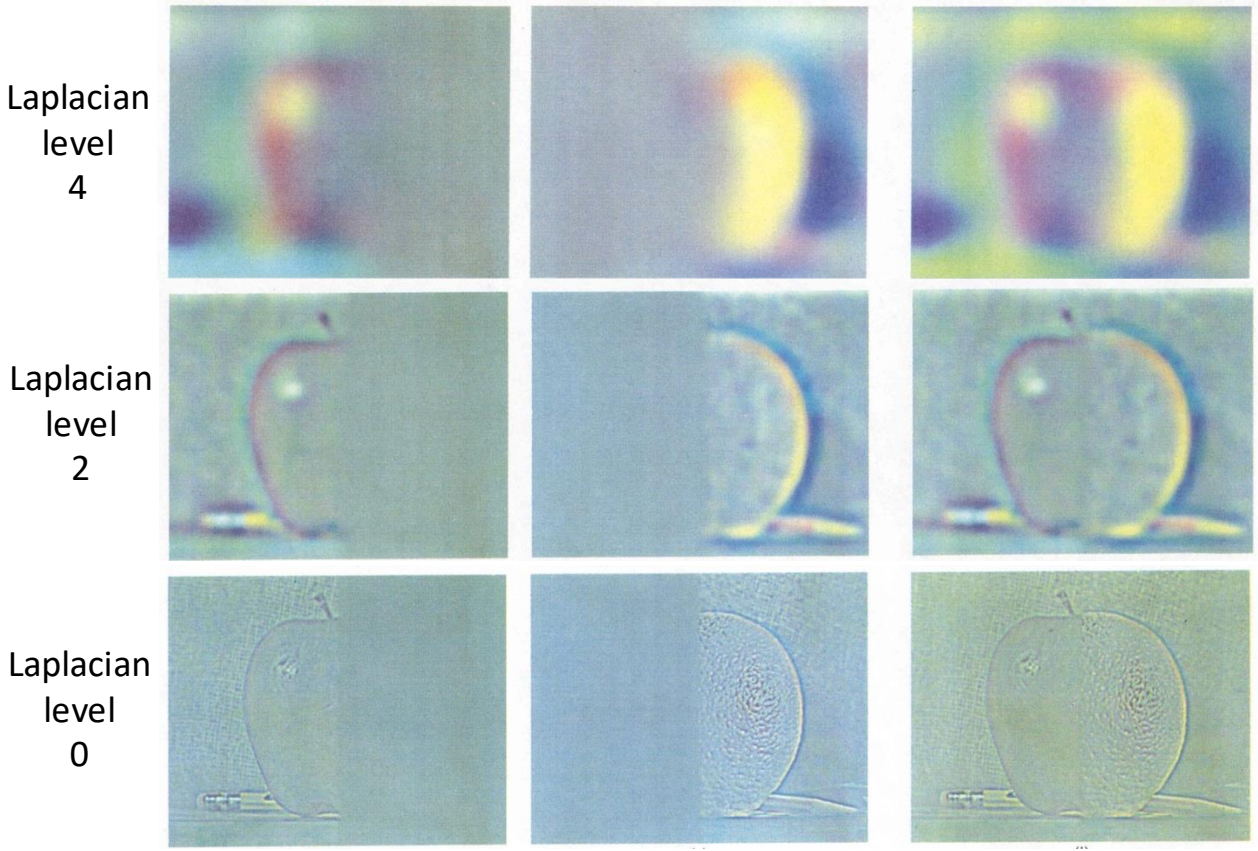
Multi-band Blending with Laplacian Pyramid

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





Multi-band Blending with Laplacian Pyramid



Multi-band Blending

- Compute Laplacian pyramid of images and mask
- Create blended image at each level of pyramid
  - At low frequencies, blend slowly
  - At high frequencies, blend quickly
- Reconstruct complete image

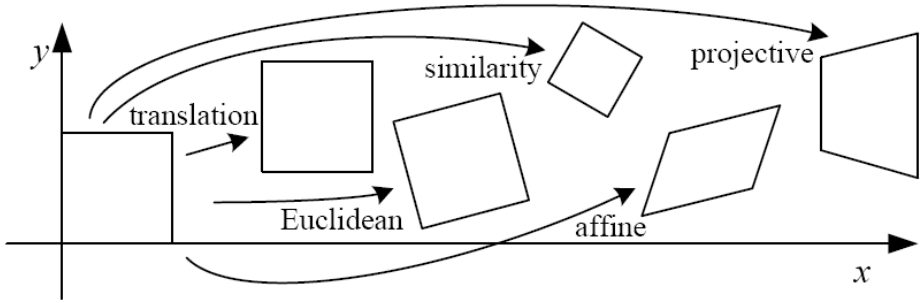
Multi-band Blending

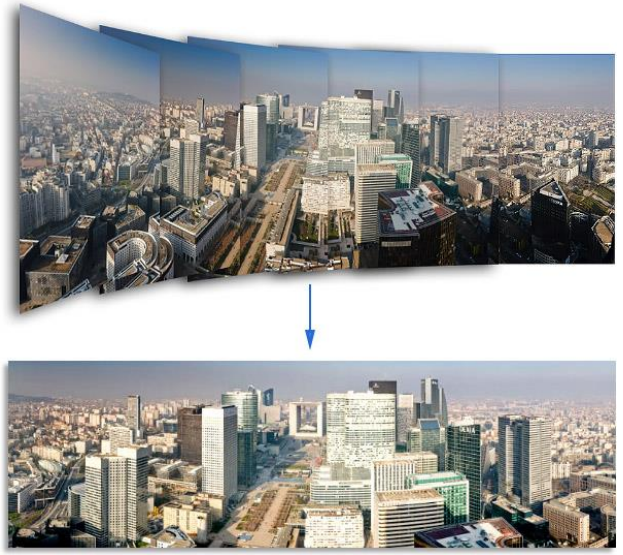
(a) Linear blending

(b) Multi-band blending

Today's Agenda

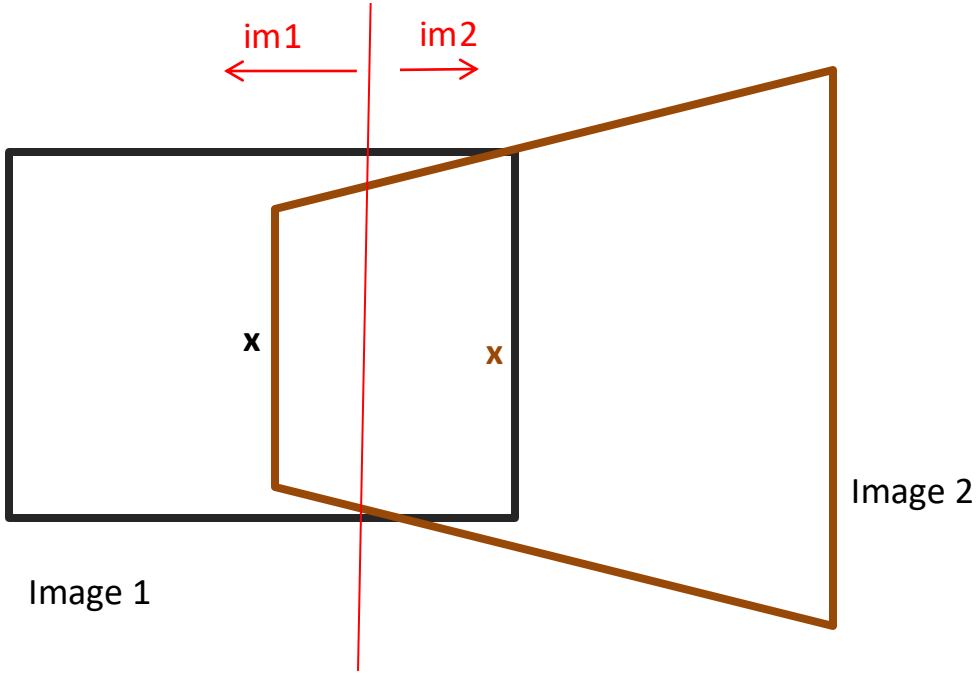
- Global operations
  - Global Transformations
  - Image Warping
  - Image Morphing
- Image Stitching
  - Transformation computation
    - Reprojection model
    - Key point matching
    - RANSAC
    - Solving transformation
  - Planar mapping
  - **Blending**
    - Feathering
    - Multi-band blending
    - **Further improvements**





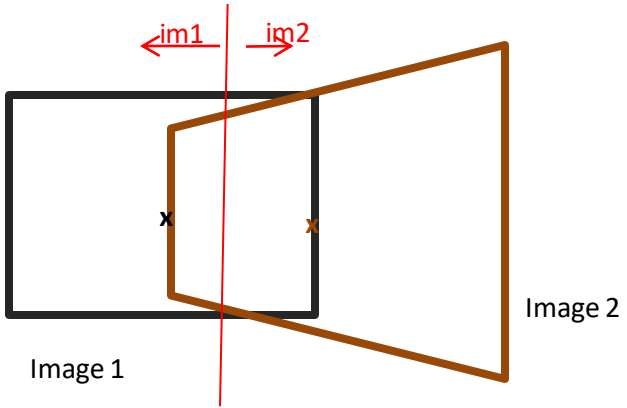
Choosing Seams

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



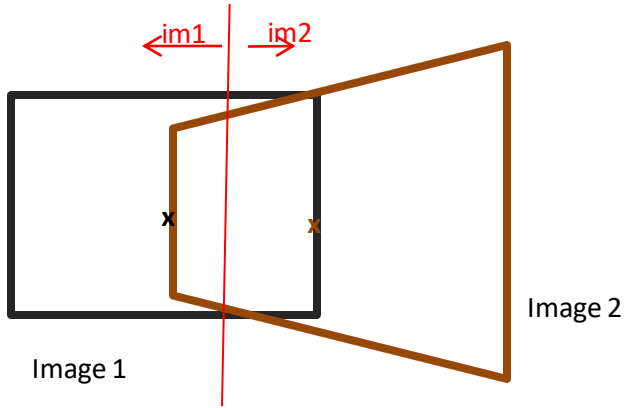
Choosing Seams

- Easy method
  - Assign each pixel to image with nearest center
  - Create a mask:
  - Smooth boundaries (“feathering”):
  - Composite



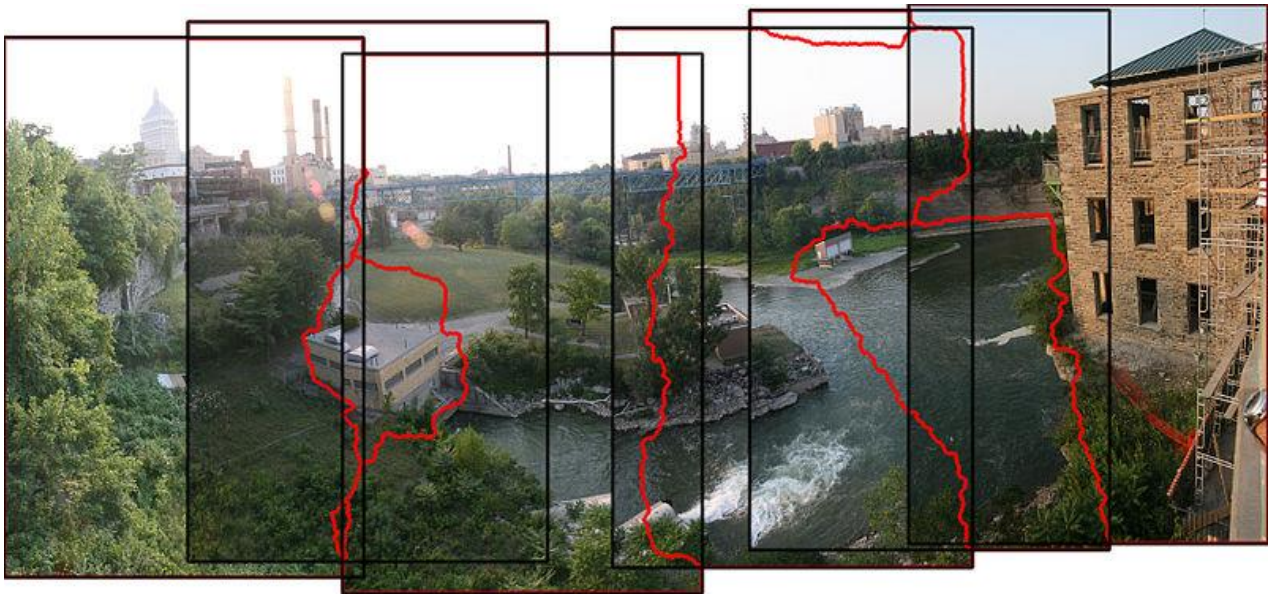
Choosing Seams

- Easy method
  - Assign each pixel to image with nearest center
  - Create a mask:
  - Smooth boundaries (“feathering”):
  - Composite



Choosing Seams

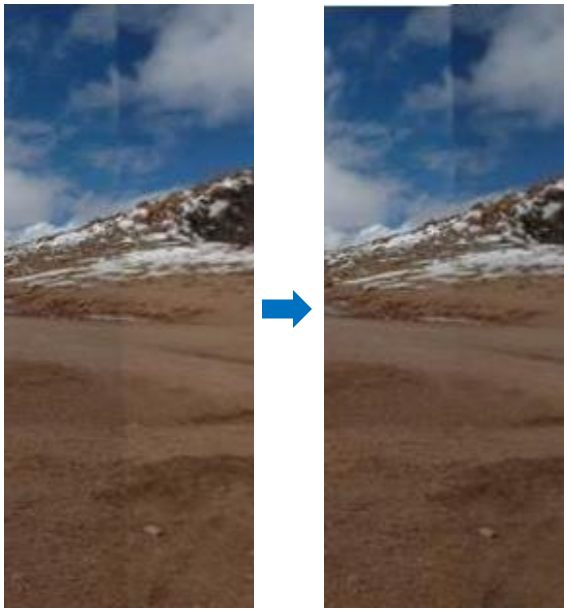
- Better method: dynamic program to find seam along well-matched regions



[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



Blending Comparison

(b) Without gain compensation

(c) With gain compensation

(d) With gain compensation and multi-band blending

De-Ghosting

- Remove inconsistencies between pixels from different overlapping images
- Simplest method: use **median** pixel color instead of **average**

Questions?

Disclaimer

Many of the slides used here are obtained from online resources (including many open lecture materials) without appropriate acknowledgement. They are used here for the sole purpose of classroom teaching. All the credit and all the copyrights belong to the original authors. You should not copy it, redistribute it, put it online, or use it for any other purposes than for this course.