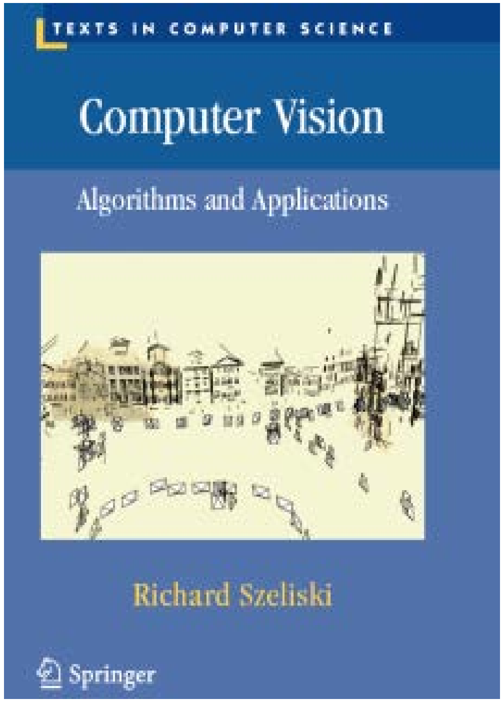
Morphing

14 May 2015

Warping, morphing, mosaic

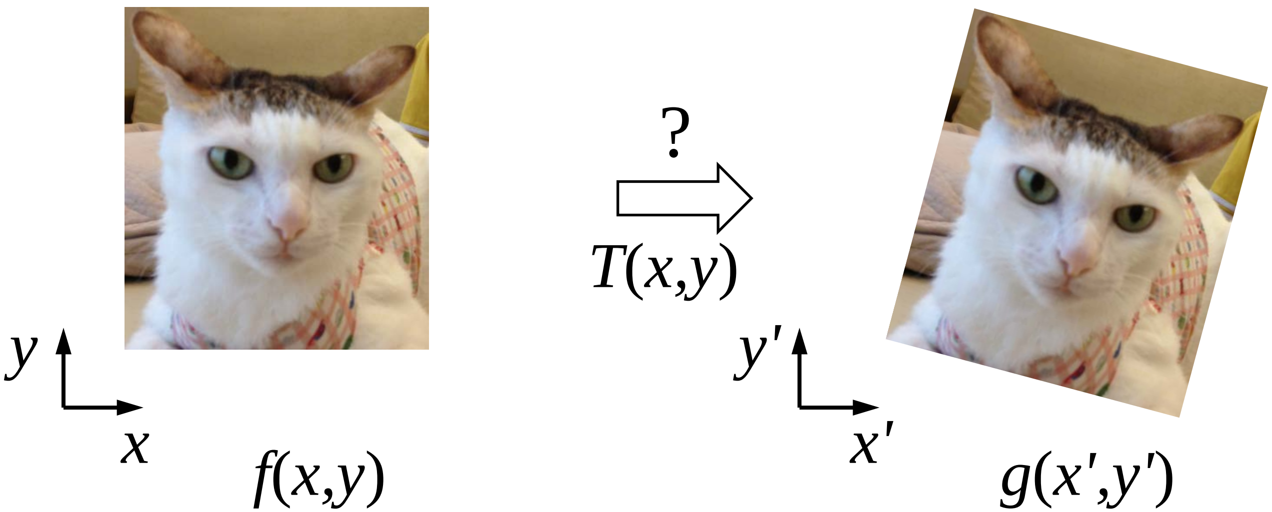
- *Slides from **Durand and Freeman (MIT), Efros (CMU, Berkeley), Szeliski (MSR), Seitz (UW), Lowe (UBC)***

http://szeliski.org/Book/



Morphing

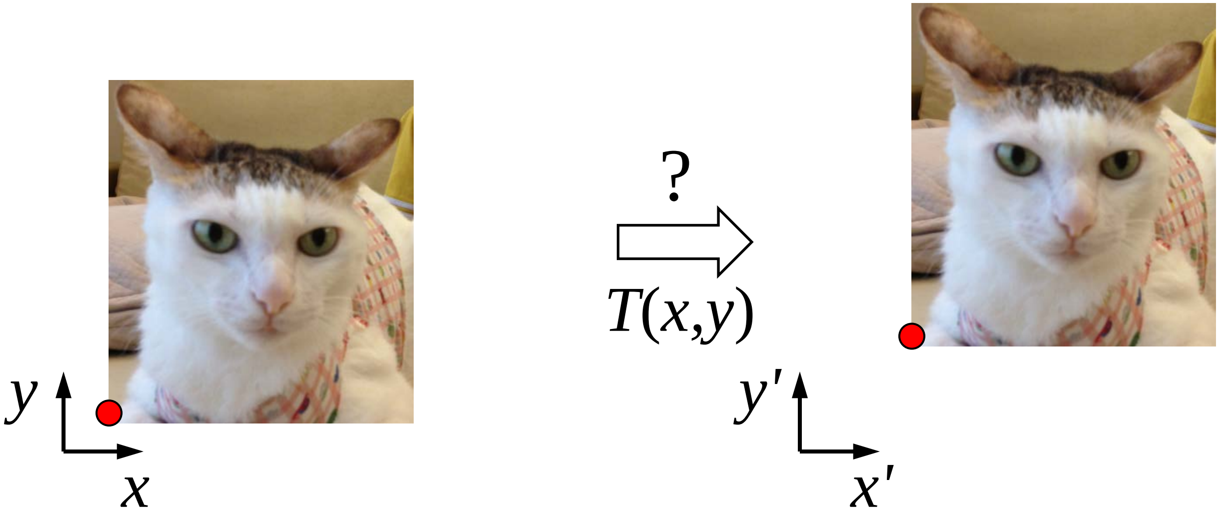
Recovering Transformations



- What if we know f and g and want to recover the transform T?
  - Let user provide correspondences
    - How many do we need?

[Efros]

Translation: # Correspondences?

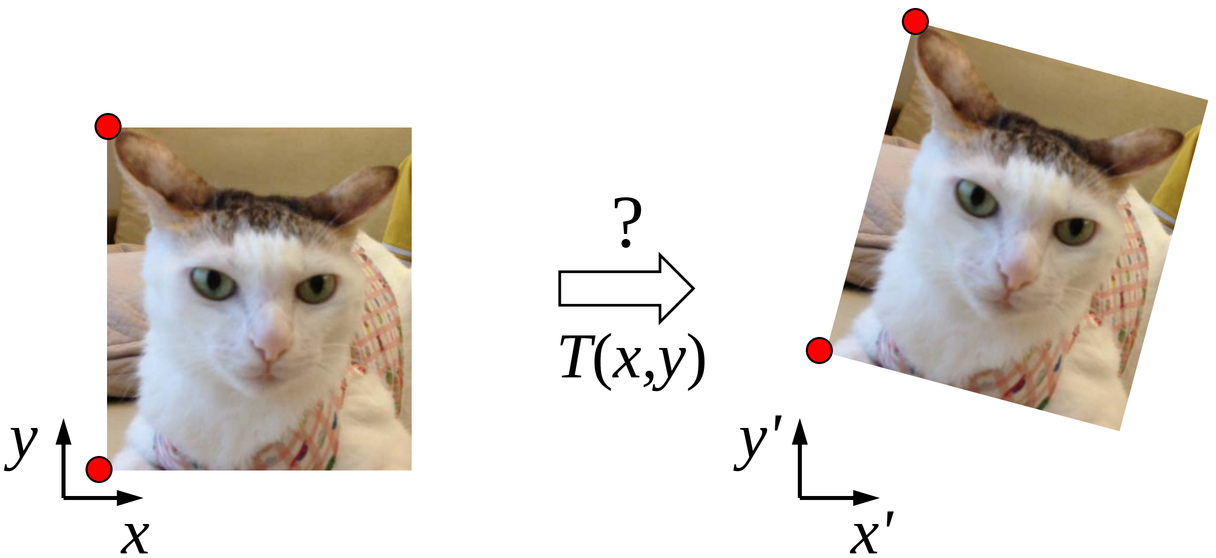


- How many correspondences needed for translation?
- How many Degrees of Freedom?
- What is the transformation matrix?

$$M = \begin{bmatrix} 1 & 0 & p'_x - p_x \\ 0 & 1 & p'_y - p_y \\ 0 & 0 & 1 \end{bmatrix}$$

[Efros]

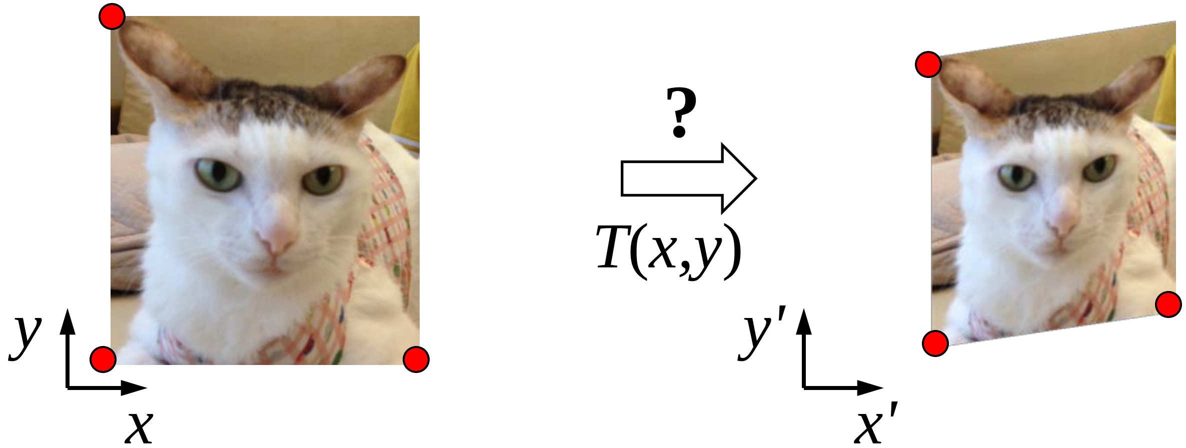
Euclidian: # Correspondences?



- How many correspondences needed for translation + rotation?
- How many DOF?

[Efros]

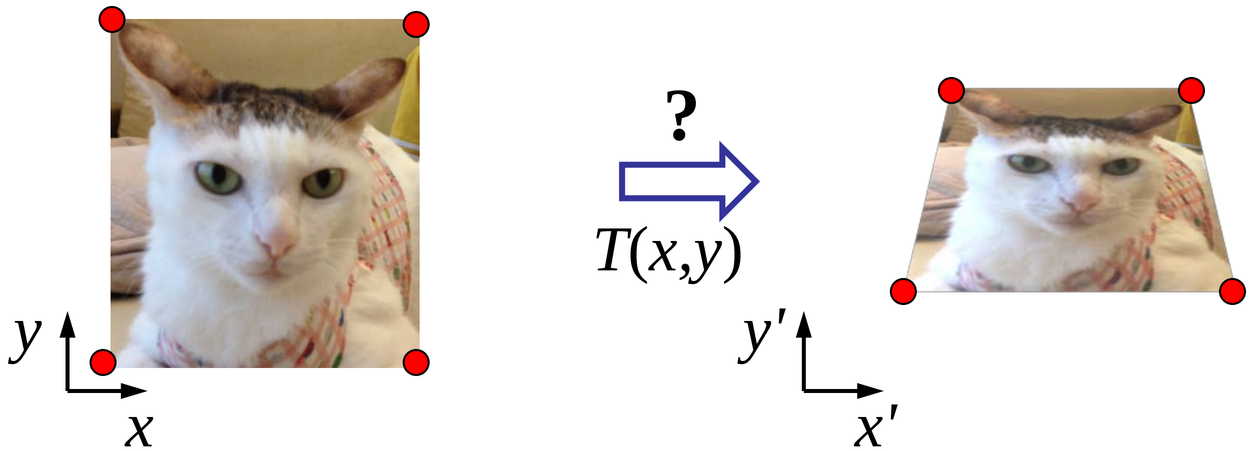
Affine: # Correspondences?



- How many correspondences needed for affine?
- How many DOF?

[Efros]

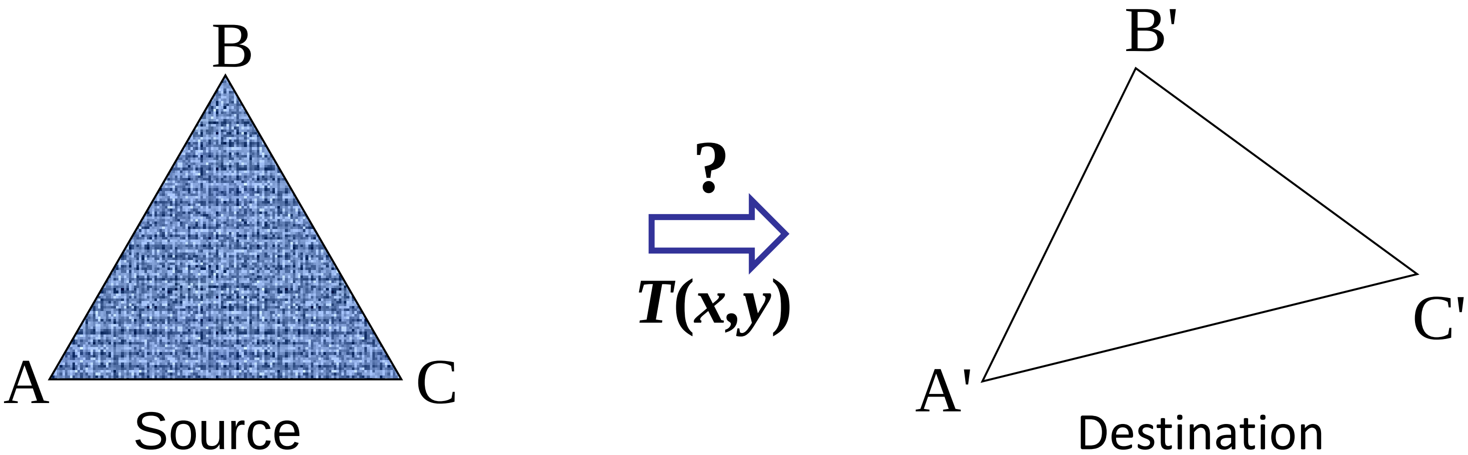
Projective: # Correspondences?



- How many correspondences needed for projective?
- How many DOF?

[Efros]

Example: Warping Triangles

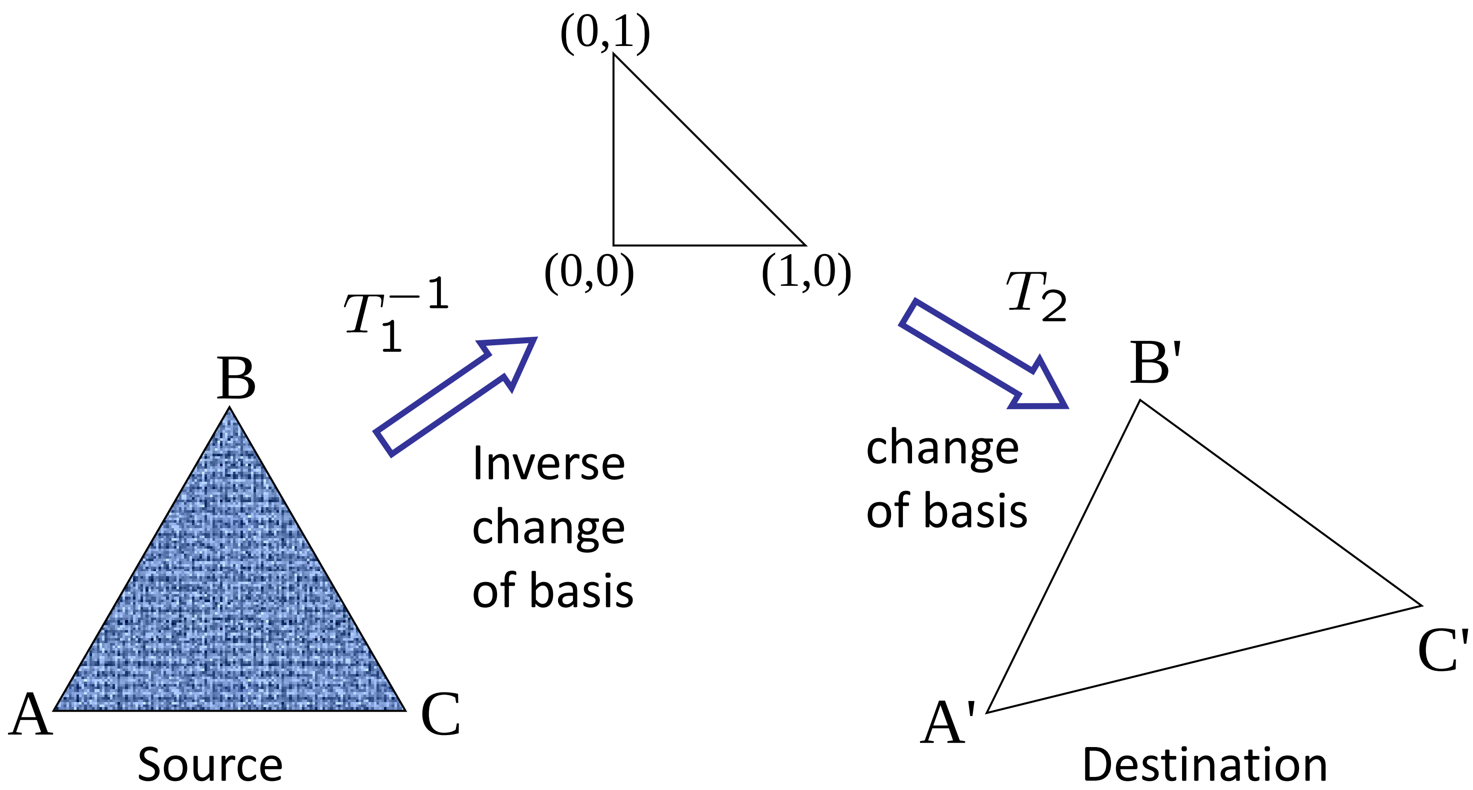


- Given two triangles: ABC and A'B'C' in 2D (12 numbers)
- Need to find transform T to transfer all pixels from one to the other.
- What kind of transformation is T?
- How can we compute the transformation matrix:

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

[Efros]

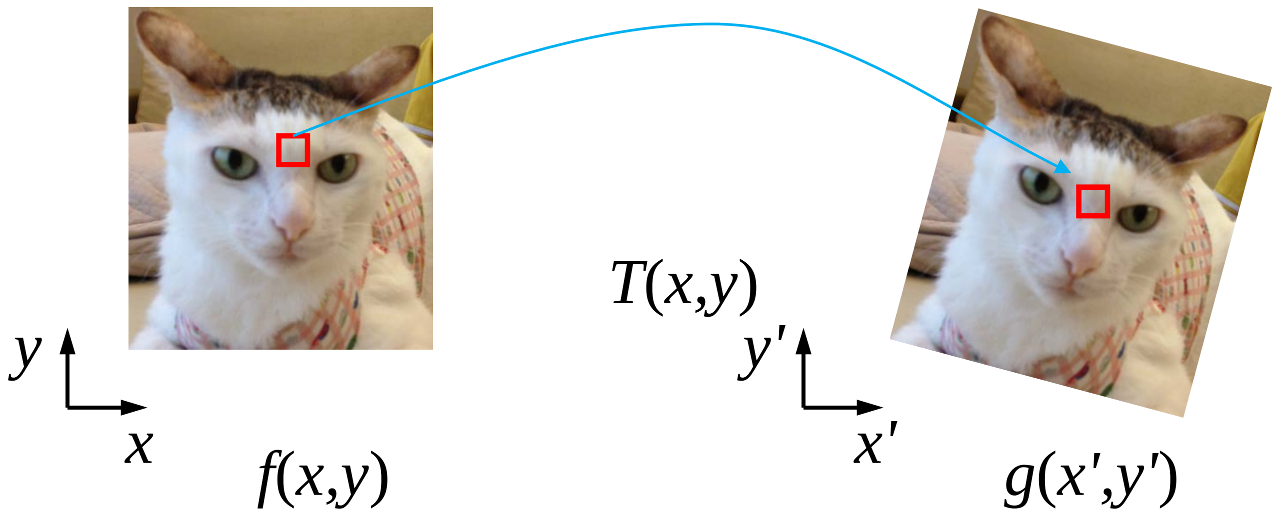
Warping Triangles



Don’t forget to move the origin too!

[Efros]

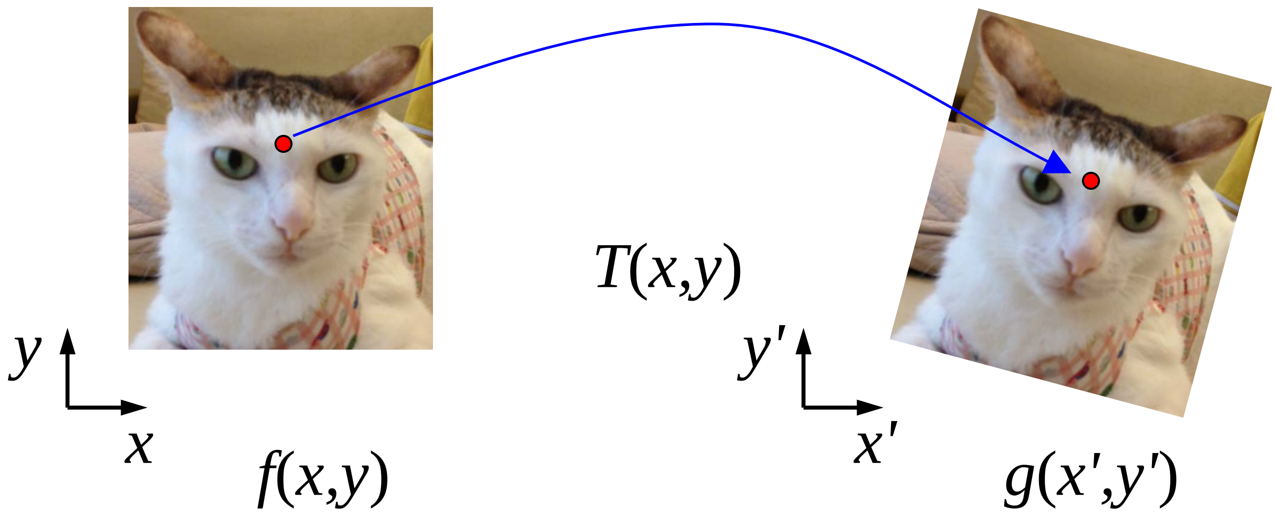
Image Warping



- Given a coordinate transform $(x',y') = T(x,y)$ and a source image $f(x,y)$, how do we compute a transformed image $g(x',y') = f(T(x,y))$?

[Efros]

Forward Warping

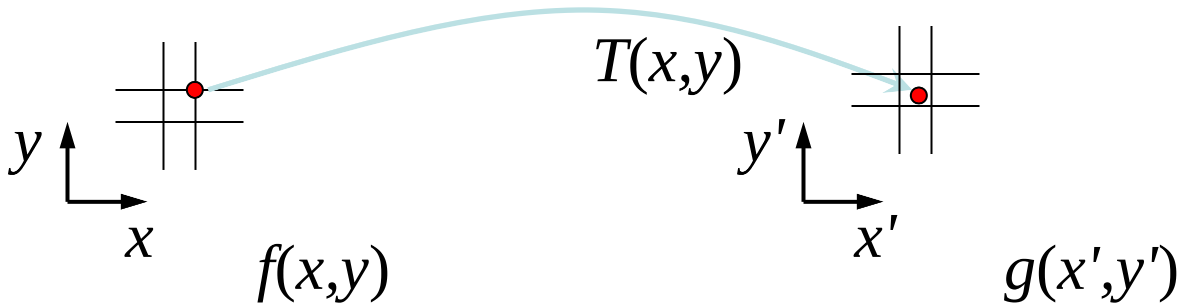


- Send each pixel $f(x,y)$ to its corresponding location $(x',y') = T(x,y)$ in the second image

Q: What if pixel lands "between" two pixels?

[Efros]

Forward Warping



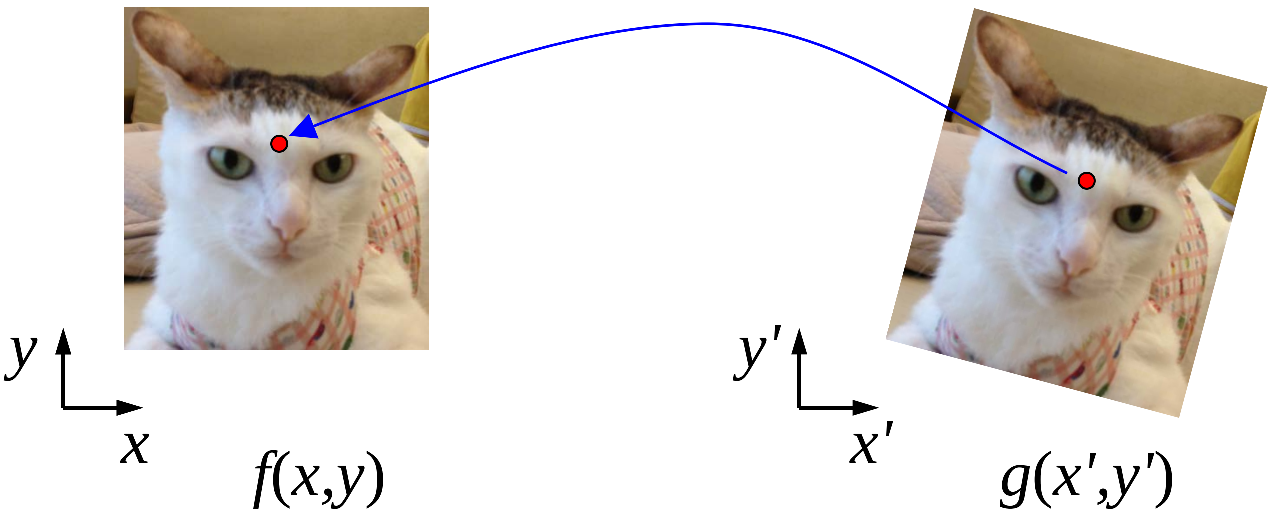
- Send each pixel $f(x,y)$ to its corresponding location
- $(x',y') = T(x,y)$ in the second image

Q: What if pixel lands "between" two pixels?

A: Distribute color among neighboring pixels (x',y')

[Efros]

Inverse Warping

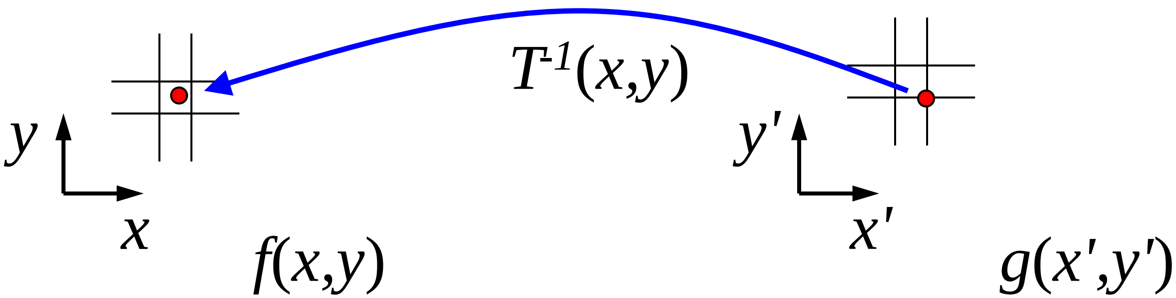


- Get each pixel $g(x',y')$ from its corresponding location $(x,y) = T^{-1}(x',y')$ in the first image

Q: What if pixel comes from "between" two pixels?

[Efros]

Inverse Warping



- Get each pixel $g(x',y')$ from its corresponding location
  $(x,y) = T^{-1}(x',y')$ in the first image

Q: What if pixel comes from "between" two pixels?

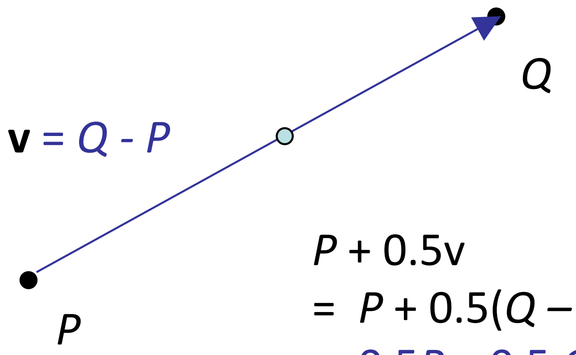
A: *Interpolate* color value from neighbors

- nearest neighbor, bilinear, Gaussian, bicubic

[Efros]

Linear Interpolation

What's the average of P and Q?



$P + 0.5\mathrm{v}$

$= P + 0.5(Q - P)$

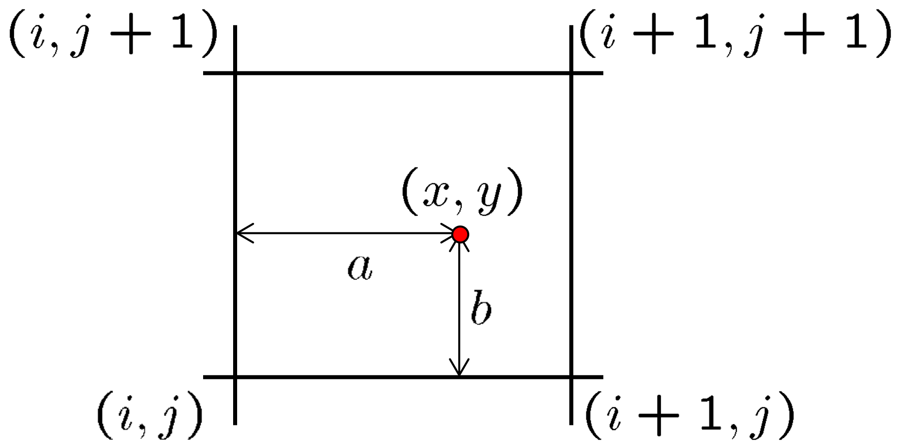
$= 0.5P + 0.5\,Q$

Linear Interpolation
(Affine Combination):
New point *aP + bQ,*
defined only when a+*b* = 1
So *aP*+*bQ* = *aP*+(1-*a*)*Q*

[Efros]

Bilinear Interpolation

- Sampling at $f(x,y)$:



$$
\begin{aligned}
f(x,y) = \;& (1-a)(1-b) && f[i,j] \\
& +a(1-b) && f[i+1,j] \\
& +ab && f[i+1,j+1] \\
& +(1-a)b && f[i,j+1]
\end{aligned}
$$

[Efros]

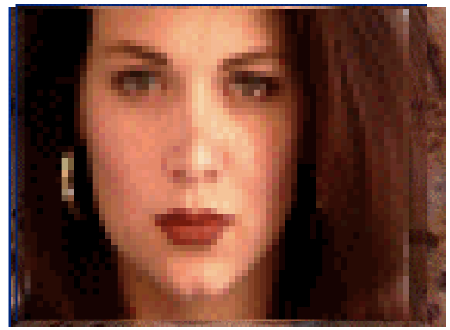
Forward vs. Inverse Warping

- Q: Which is better?

- A: Usually inverse—eliminates holes
  - However, it requires an invertible warp function—not always possible...

[Efros]

Morphing = Object Averaging


- The aim is to find "an average" between two objects
  - Not an average of two images of objects…
  - …but an image of the average object!
  - How can we make a smooth transition in time?
    - Do a "weighted average" over time *t*
- How do we know what the average object looks like?
  - We haven't a clue!
  - But we can often fake something reasonable
    - Usually required user/artist input

[Efros]

Idea #1: Cross-Dissolve

- Interpolate whole images:
- $\text{Image}_{\text{halfway}} = (1\text{-}t)^*\text{Image}_1 + t^*\text{image}_2$
- This is called **cross-dissolve** in film industry

- But what if the images are not aligned?

[Efros]

Idea #2: Align, then Cross-Dissolve

- Align first, then cross-dissolve
  - Alignment using global warp – picture still valid

[Efros]

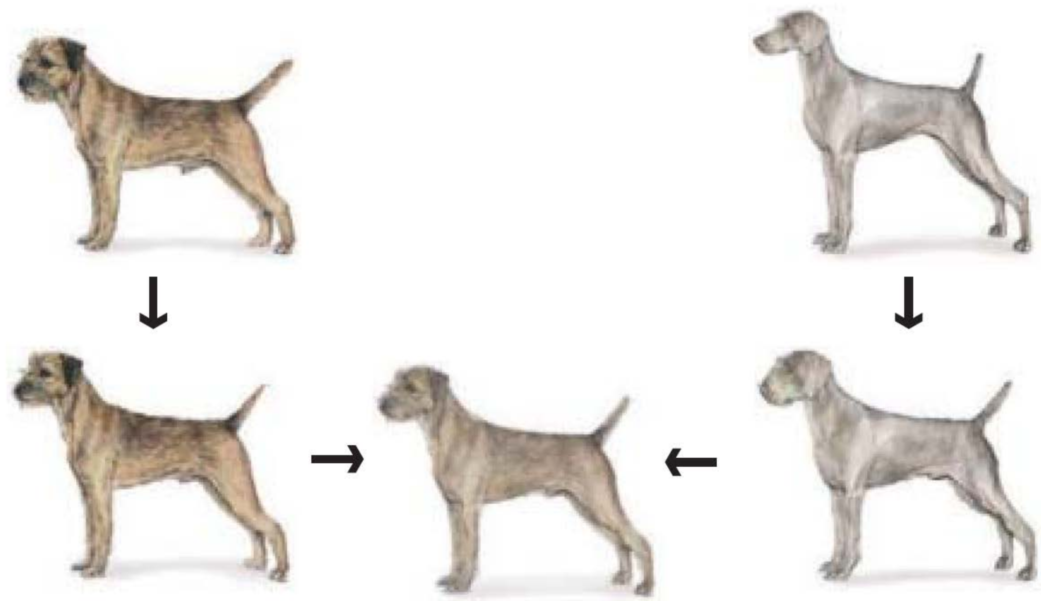
Dog Averaging

- What to do?
  - Cross-dissolve doesn't work
  - Global alignment doesn't work
    - Cannot be done with a global transformation (e.g. affine)

- Feature matching!
  - Nose to nose, tail to tail, etc.
  - This is a local (non-parametric) warp

[Efros]

Idea #3: Local Warp, then Cross-Dissolve



- Morphing procedure: *for every t,*

1. Find the average shape (the "mean dog")
   - Local warping
2. Find the average color
   - Cross-dissolve the warped images

[Efros]

Local (Non-Parametric) Image Warping

- Need to specify a more detailed warp function
  - Global warps were functions of a few (2,4,8) parameters
  - Non-parametric warps $u(x,y)$ and $v(x,y)$ can be defined independently for every single location x,y!
  - Once we know vector field u,v we can easily warp each pixel (use backward warping with interpolation)

[Efros]

Linear Interpolation

Mesh Warping

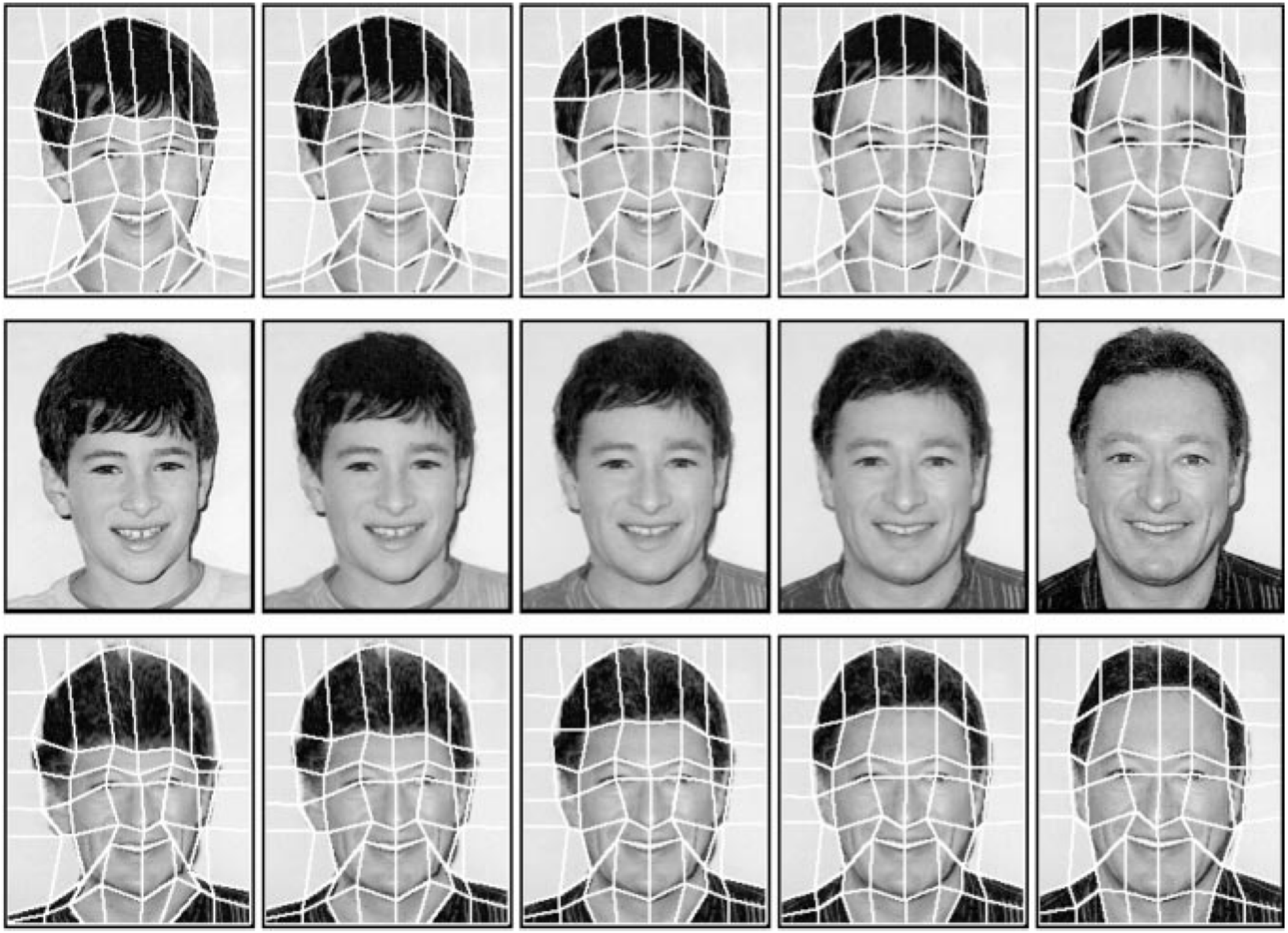
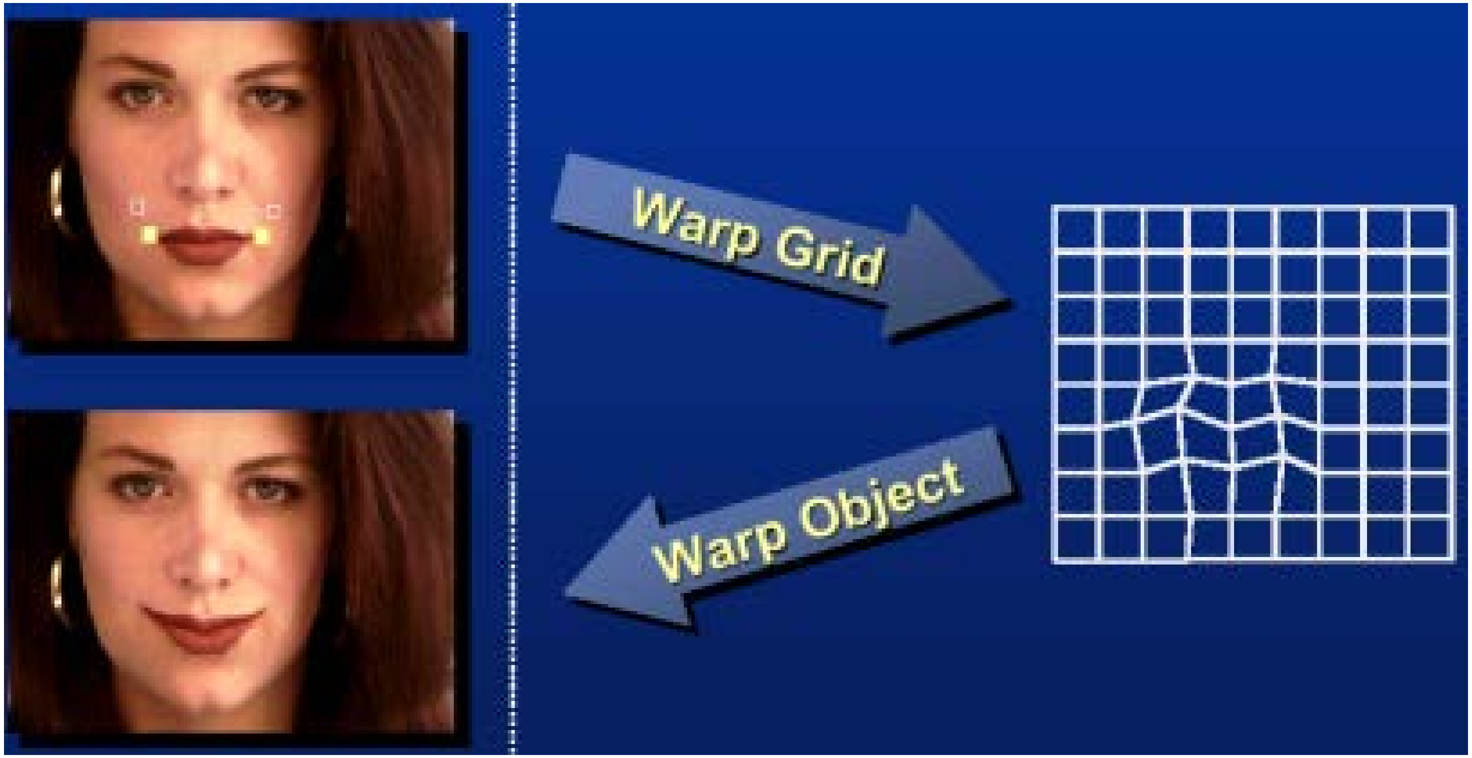


Image Warping – Non-Parametric

- Move control points to specify a spline warp
- Spline produces a smooth vector field



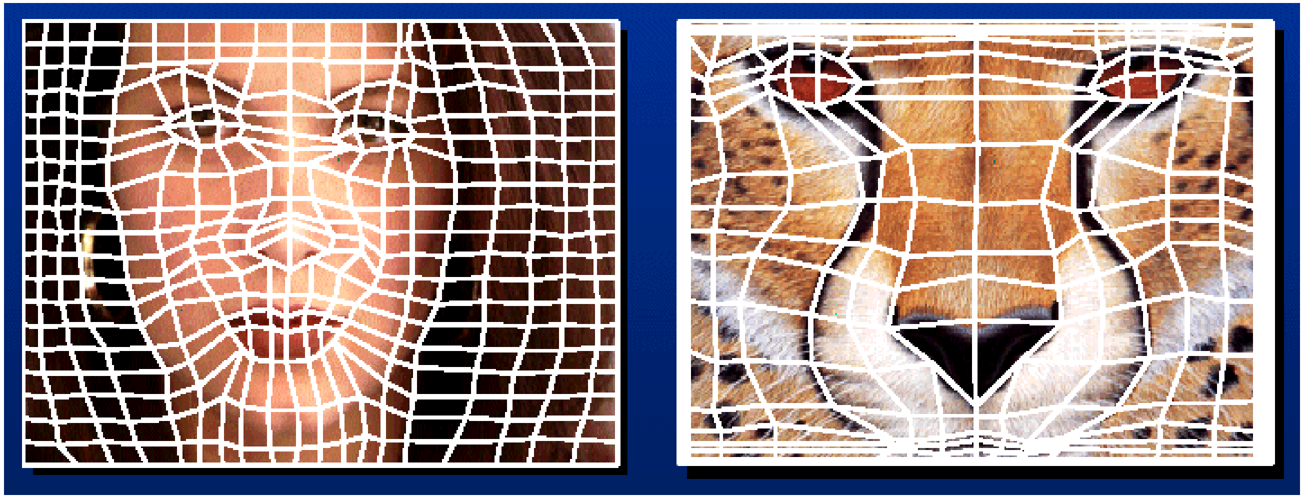
[Efros]

Warp Specification – Dense

- How can we specify the warp?

  Specify corresponding *spline control points*

  - *interpolate* to a complete warping function



But we want to specify only a few points, not a grid

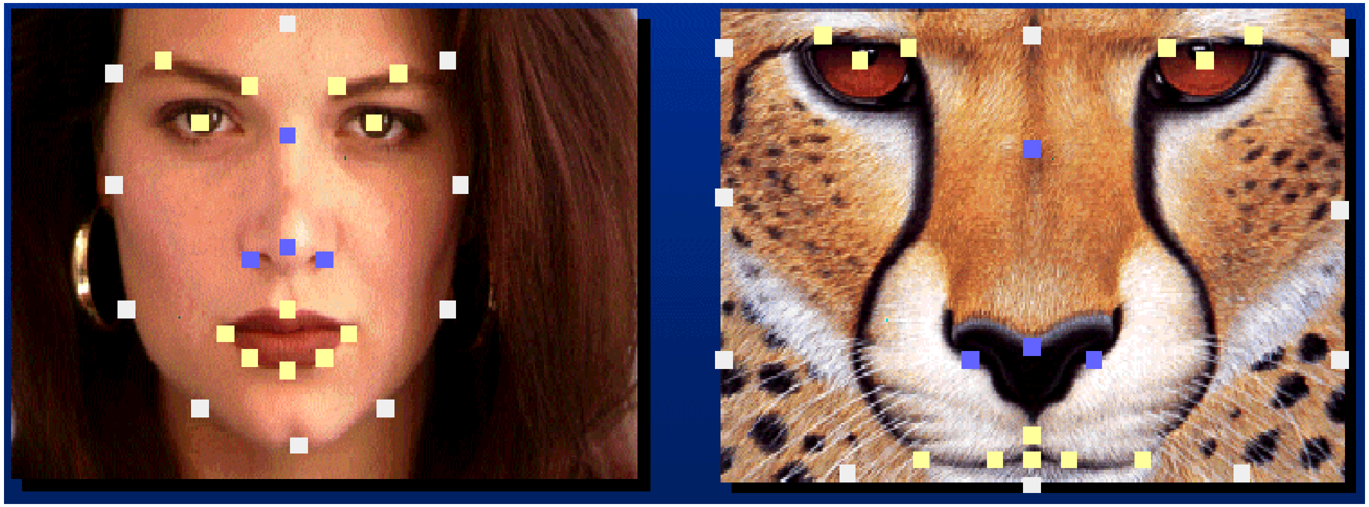
[Efros]

Warp Specification – Sparse

- How can we specify the warp?

  Specify corresponding *points*

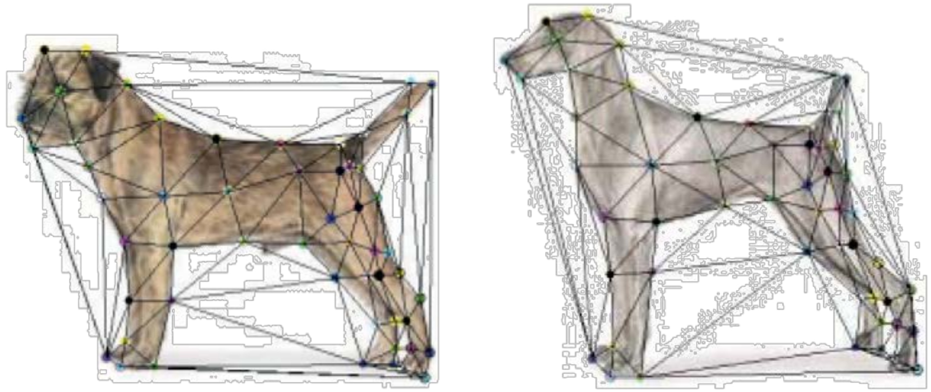
  - *interpolate* to a complete warping function
  - How do we do it?



How do we go from feature points to pixels?

[Efros]

Triangular Mesh

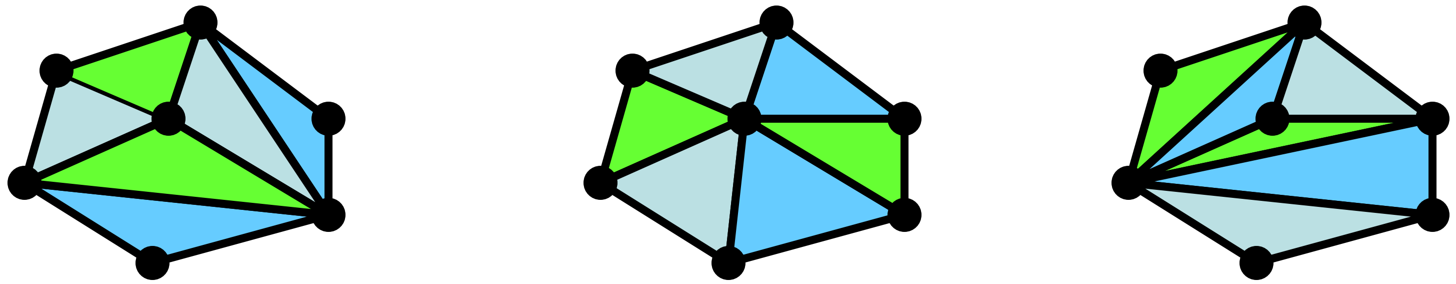


1. Input correspondences at key feature points
2. Define a triangular mesh over the points
   - Same mesh in both images!
   - Now we have triangle-to-triangle correspondences
3. Warp each triangle separately from source to destination
   - How do we warp a triangle?
   - 3 points = affine warp!

[Efros]

Triangulations

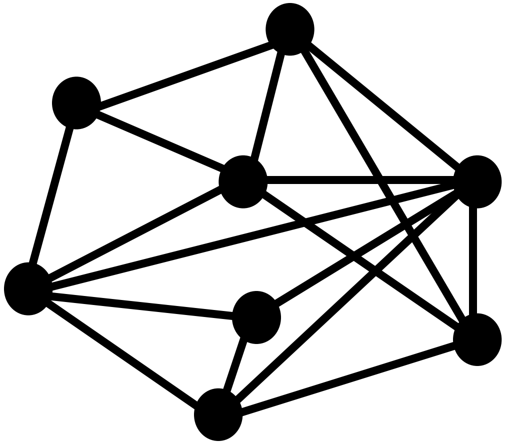
- A *triangulation* of set of points in the plane is a *partition* of the convex hull to triangles whose vertices are the points, and do not contain other points.
- There are an exponential number of triangulations of a point set.



[Efros]

An O(n^3) Triangulation Algorithm

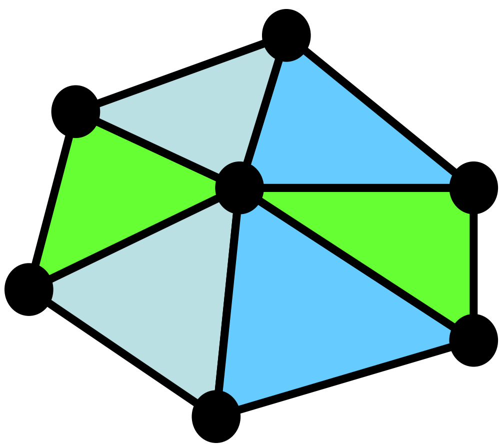
- Repeat until impossible:
  - Select two sites.
  - If the edge connecting them does not intersect previous edges, keep it.



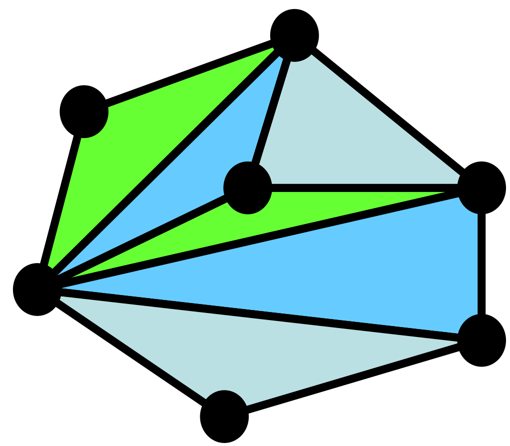
[Efros]

"Quality" Triangulations

- Let $\alpha(T) = (\alpha_1, \alpha_2 ,.., \alpha_{3t})$ be the vector of angles in the triangulation T in increasing order.
- A triangulation T_1 will be "better" than T_2 if $\alpha(T_1) > \alpha(T_2)$ lexicographically.
- The Delaunay triangulation is the "best"
  - *Maximizes smallest angles*



good

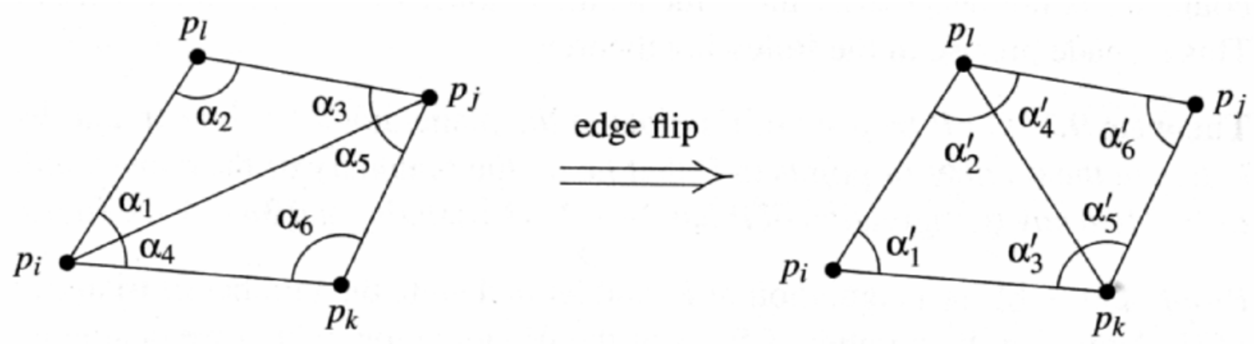


bad

[Efros]

Improving a Triangulation

- In any convex quadrangle, an *edge flip* is possible. If this flip *improves* the triangulation locally, it also improves the global triangulation.



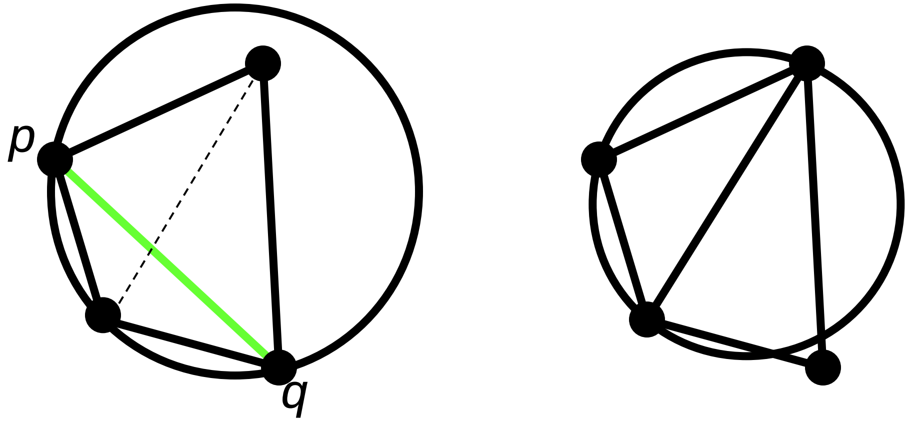
  - If an edge flip improves the triangulation, the first edge is called *illegal*.

$$\min_{1\leqslant i\leqslant 6} \alpha_i < \min_{1\leqslant i\leqslant 6} \alpha'_i.$$

[Efros]

Illegal Edges

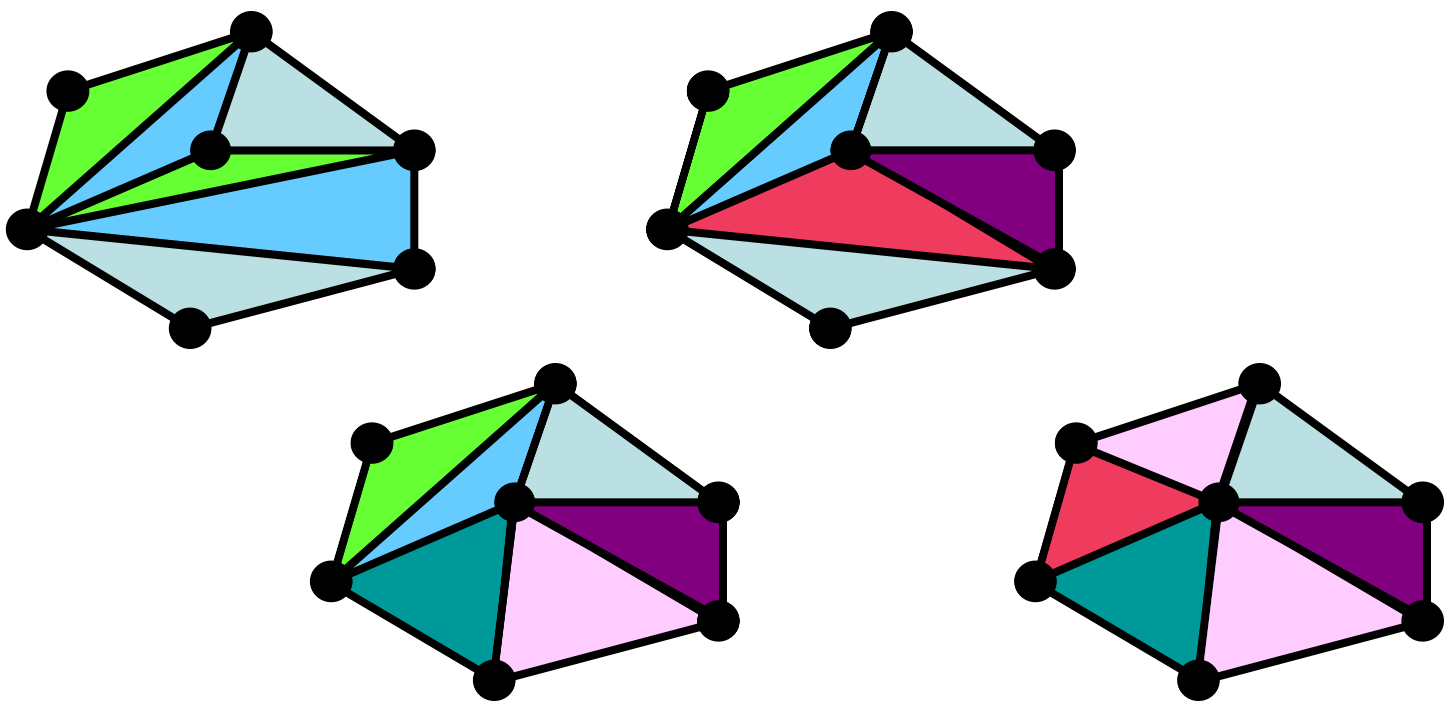
- **Lemma:** An edge *pq* is illegal iff one of its opposite vertices is inside the circle defined by the other three vertices.
- **Proof:** By Thales' theorem.



  - **Theorem:** A Delaunay triangulation does not contain illegal edges.
  - **Corollary:** A triangle is Delaunay iff the circle through its vertices is empty of other sites.
  - **Corollary:** The Delaunay triangulation is not unique if more than three sites are co-circular.

Naïve Delaunay Algorithm

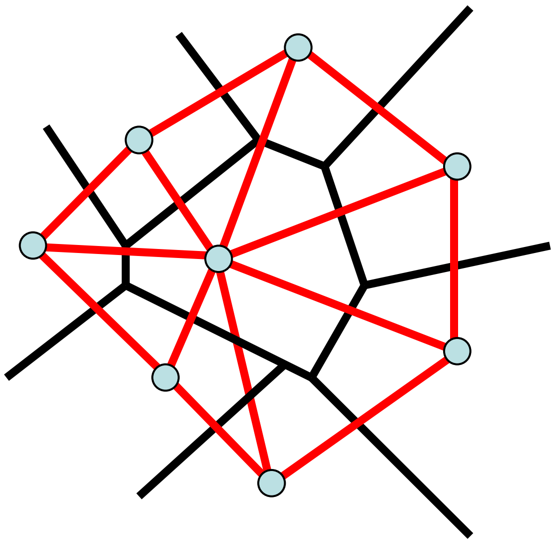
- Start with an arbitrary triangulation. Flip any illegal edge until no more exist.
- Could take a long time to terminate.



[Efros]

Delaunay Triangulation by Duality

- General position assumption: There are no four co-circular points.
- Draw the dual to the Voronoi diagram by connecting each two neighboring sites in the Voronoi diagram.



- **Corollary:** The DT may be constructed in $O(n\log n)$ time.

[Efros]

MATLAB

```
x = rand(1,10);
y = rand(1,10);
TRI = delaunay(x,y);
triplot(TRI,x,y);
axis([0 1 0 1]);
hold on
plot(x,y,'or');
hold off
```

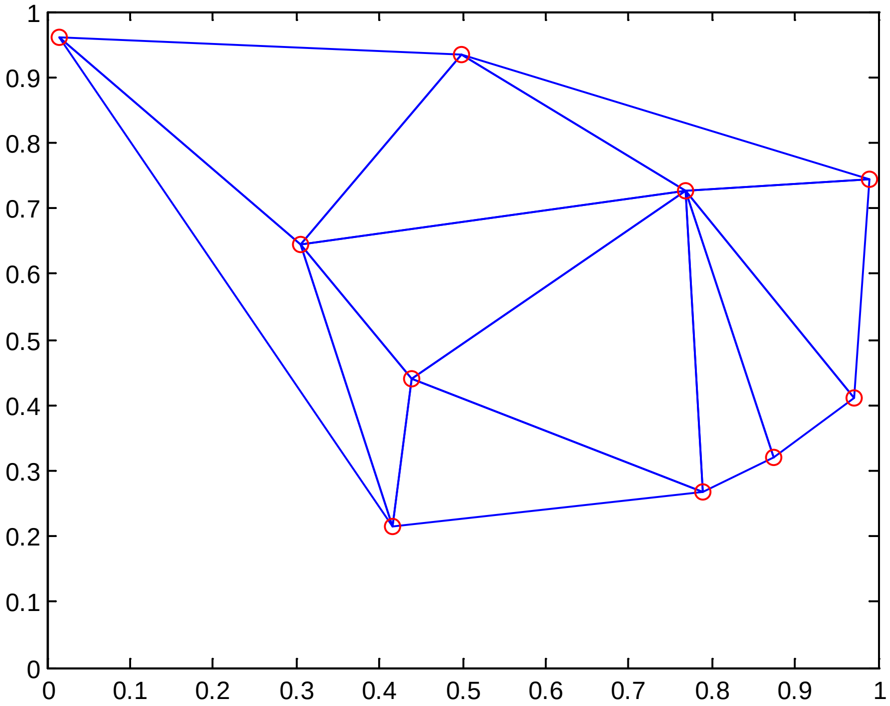
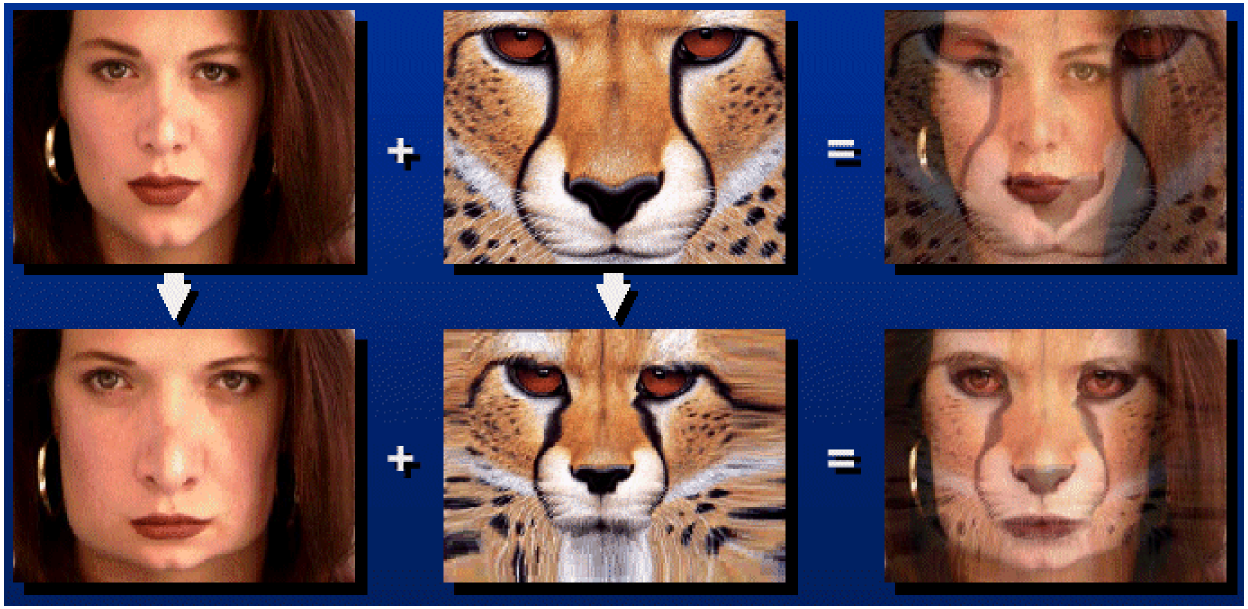


Image Morphing

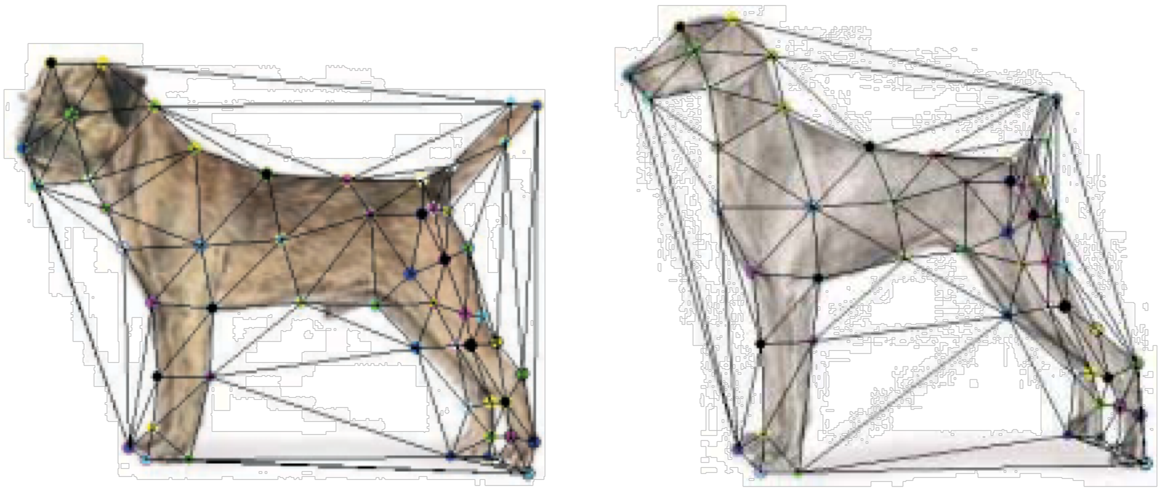
- We know how to warp one image into the other, but how do we create a morphing sequence?
  1. Create an intermediate shape (by interpolation)
  2. Warp both images towards it
  3. Cross-dissolve the colors in the newly warped images



[Efros]

Warp Interpolation

- How do we create an intermediate warp at time t?
  - Assume $t = [0,1]$
  - Simple linear interpolation of each feature pair
  - $(1-t)*p1+t*p0$ for corresponding features $p0$ and $p1$



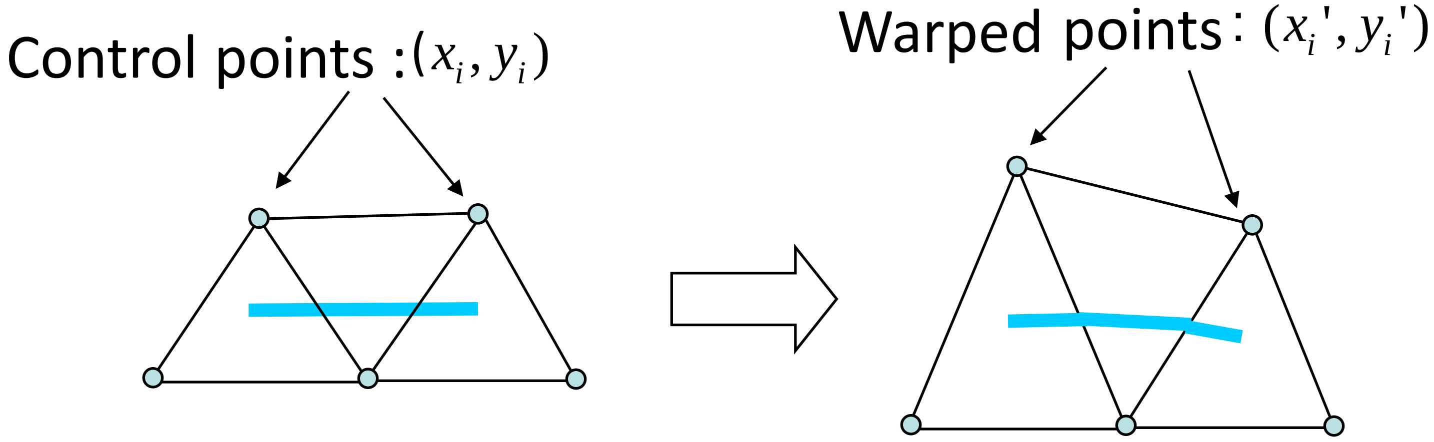
[Efros]

Warping Texture

- Problem:
  - Given corresponding points in two images, how do we warp one into the other?

- Two common solutions
  1. Piecewise linear using triangle mesh
  2. Thin-plate spline interpolation

Interpolation Using Triangles

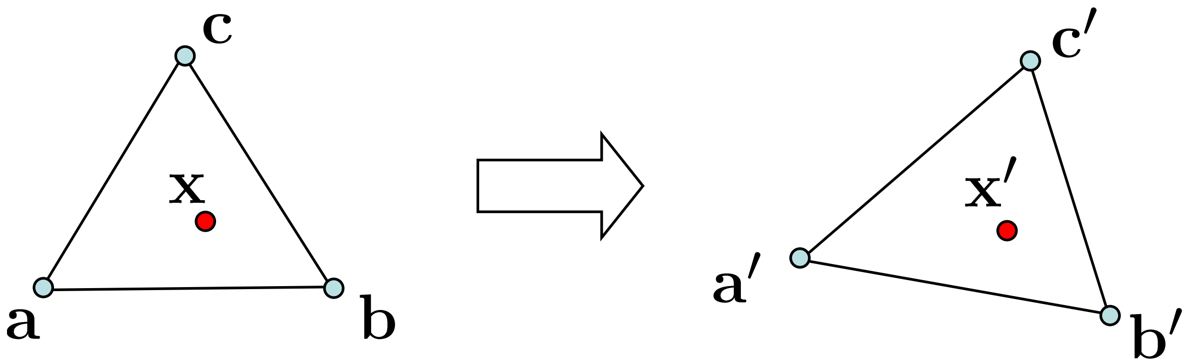


Region of interest enclosed by triangles

Moving nodes changes each triangle

Just need to map regions between two triangles

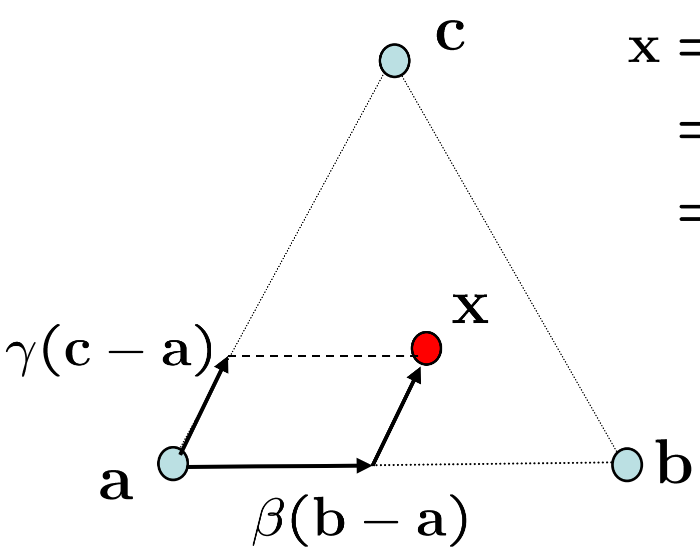
Barycentric Coordinates



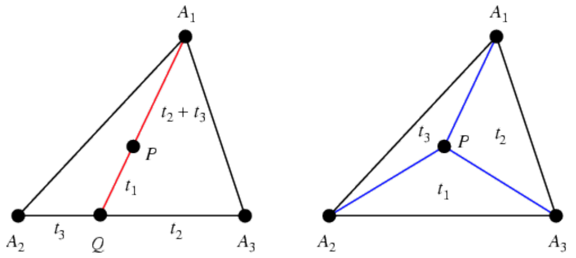
$$\mathbf{x} = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c} \qquad \mathbf{x}' = \alpha\mathbf{a}' + \beta\mathbf{b}' + \gamma\mathbf{c}'$$

$$\alpha + \beta + \gamma = 1$$

Barycentric Coordinates



$$\begin{aligned}\mathbf{x} &= \mathbf{a} + \beta(\mathbf{b}-\mathbf{a}) + \gamma(\mathbf{c}-\mathbf{a}) \\ &= (1-\beta-\gamma)\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c} \\ &= \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}\end{aligned}$$



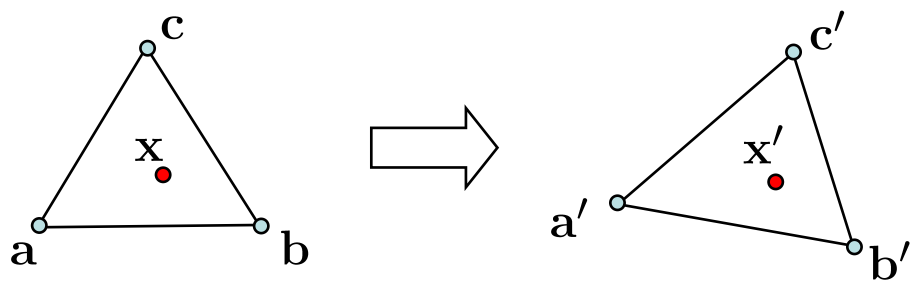
$$\mathbf{x} = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$$
$$\alpha + \beta + \gamma = 1$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}$$

Three linear equations in 3 unknowns

Interpolation Using Triangles

- To find out where each pixel in new image comes from in old image
  - Determine which triangle it is in
  - Compute its barycentric coordinates
  - Find equivalent point in equivalent triangle in original image
- Only well defined in region of 'convex hull' of control points



$$\mathbf{x} = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c} \qquad \mathbf{x}' = \alpha\mathbf{a}' + \beta\mathbf{b}' + \gamma\mathbf{c}'$$

Thin-Plate Spline Interpolation

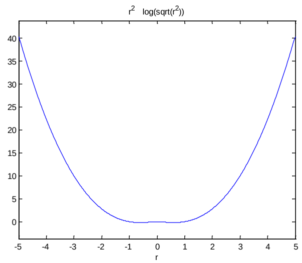
- Define a smooth mapping function $(x',y')=f(x,y)$ such that
  - It maps each point (x,y) onto (x',y') and does something smooth in between
  - Defined everywhere, even outside convex hull of control points

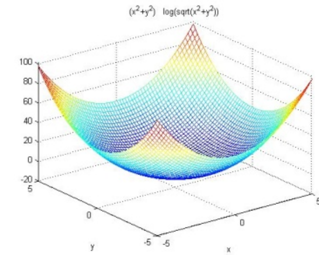
$$f(x_i, y_i) = (x'_i, y'_i) \text{ for all } i = 1, \ldots, n$$

Thin-Plate Spline Interpolation

- Function has form

$$f(x,y) = (f_x(x,y), f_y(x,y))$$





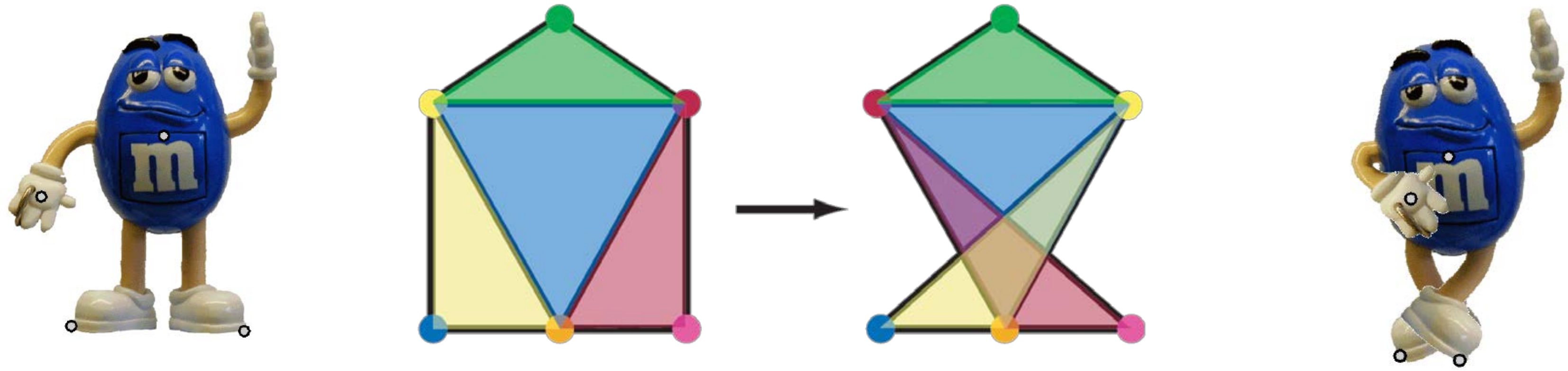
$$f_x(x,y) = a_x + b_x\, x + \textstyle\sum_{i=1}^{n} w_{x_i} r_i^2 \log r_i$$

$$f_y(x,y) = a_y + b_y\, y + \textstyle\sum_{i=1}^{n} w_{y_i} r_i^2 \log r_i$$

where $r_i^2 = (x - x_i)^2 + (y - y_i)^2$

the parameters ($x, b_x, w_{x_i}, a_y, b_y, w_{y_i}$) are found by solving the linear equations given by $f(x_i, y_i) = (x_i', y_i')$

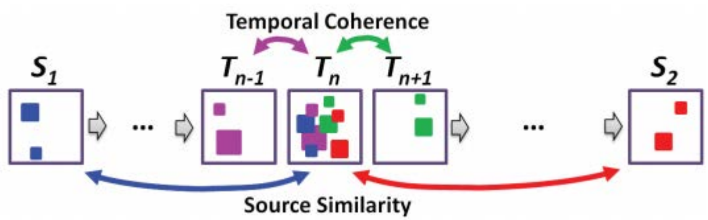
Other Issues



- Beware of folding
  - You are probably trying to do something 3D-ish
- Morphing can be generalized into 3D
  - If you have 3D data
- Extrapolation can sometimes produce interesting effects
  - Caricatures

Idea #4: Regenerative Morphing

- Regenerative Morphing
  - Shechtman, Rav-Acha, Irani, and Seitz
  - CVPR 2010
- Bidirectional similarity
- PatchMatch



Mosaic

Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = $50 \times 35^{\circ}$

Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°

Why Mosaic?

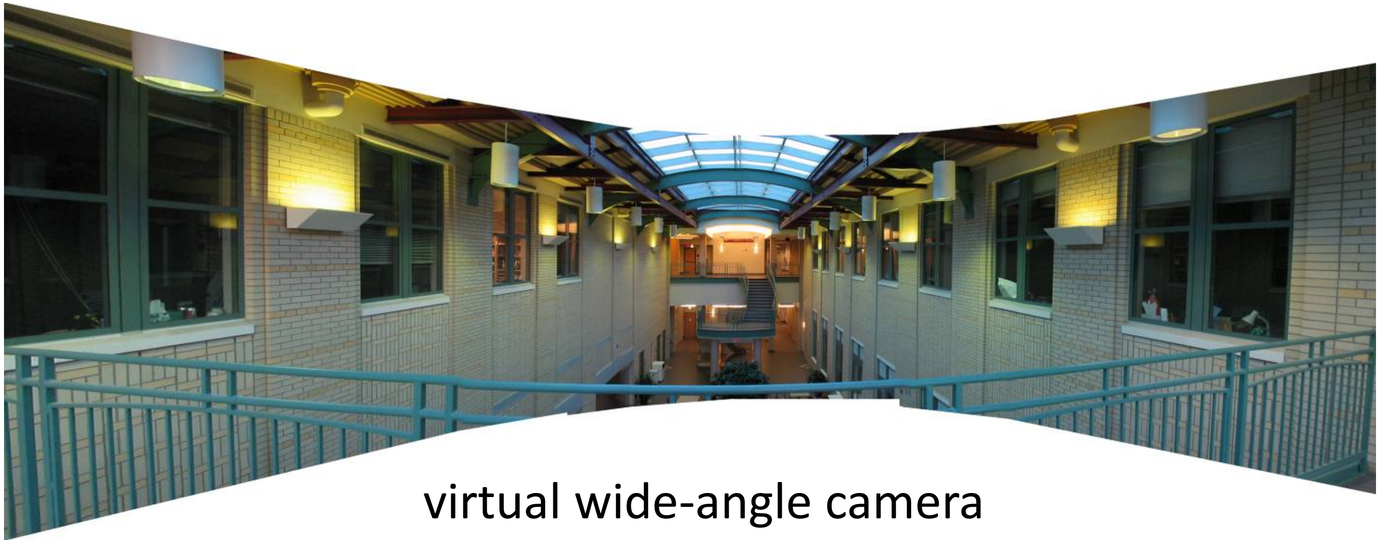
- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°
  - Panoramic Mosaic = 360 x 180°

Mosaics: Stitching Images Together



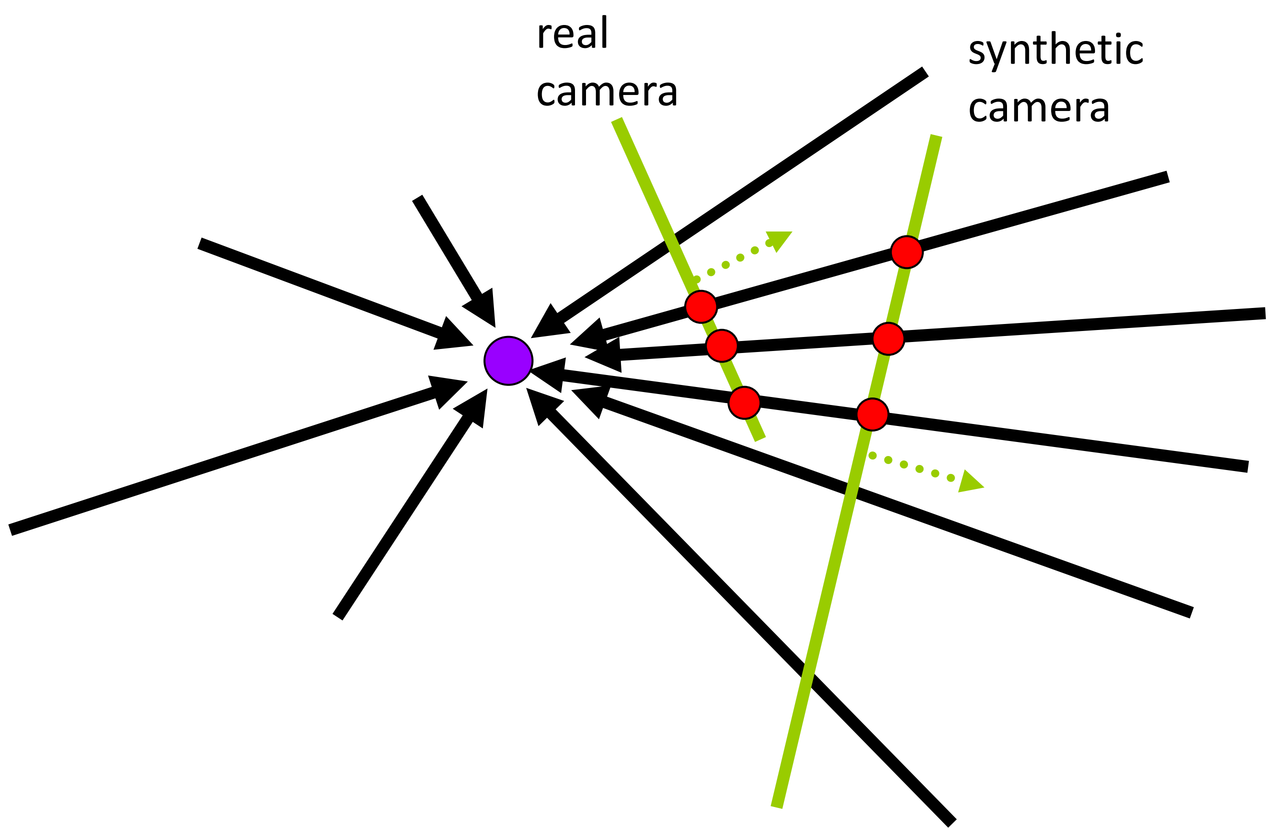




virtual wide-angle camera

A Pencil of Rays Contains All Views



Can generate any synthetic camera view
as long as it has **the same center of projection**!

How to Do It?

- Basic procedure
  - Take a sequence of images from the same position
    - **Rotate** the camera about its **optical center**
  - Compute transformation between second image and first
  - Transform the second image to overlap with the first
  - Blend the two together to create a mosaic
  - If there are more images, repeat

- …but **wait**, why should this work at all?
  - What about the 3D geometry of the scene?
  - Why aren’t we using it?

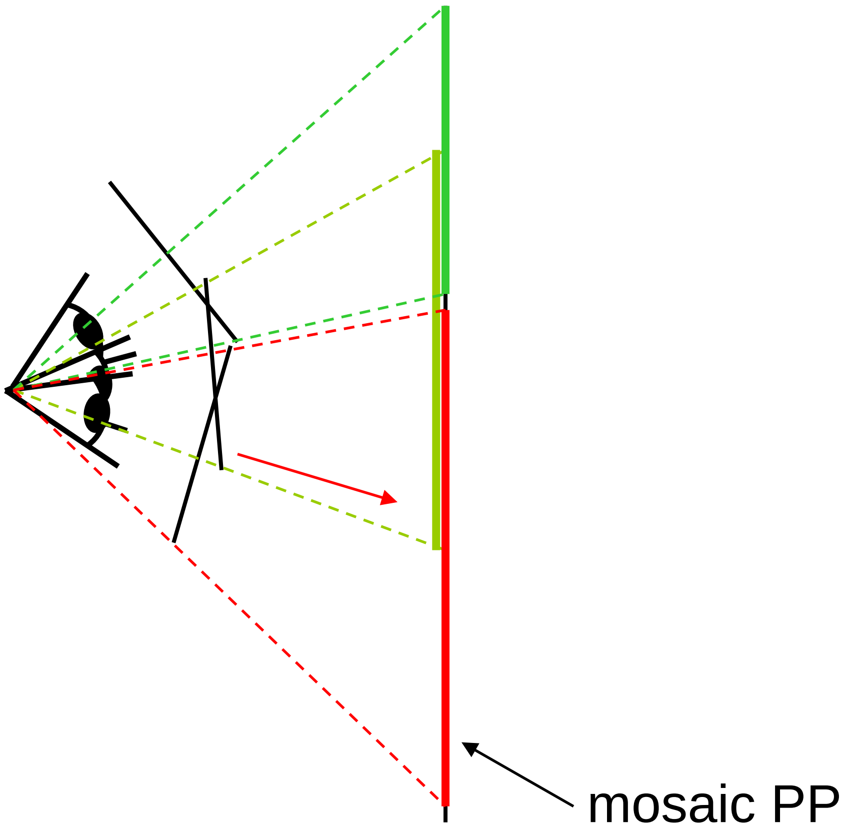
Aligning Images

left on top

right on top

Translations are not enough to align the images

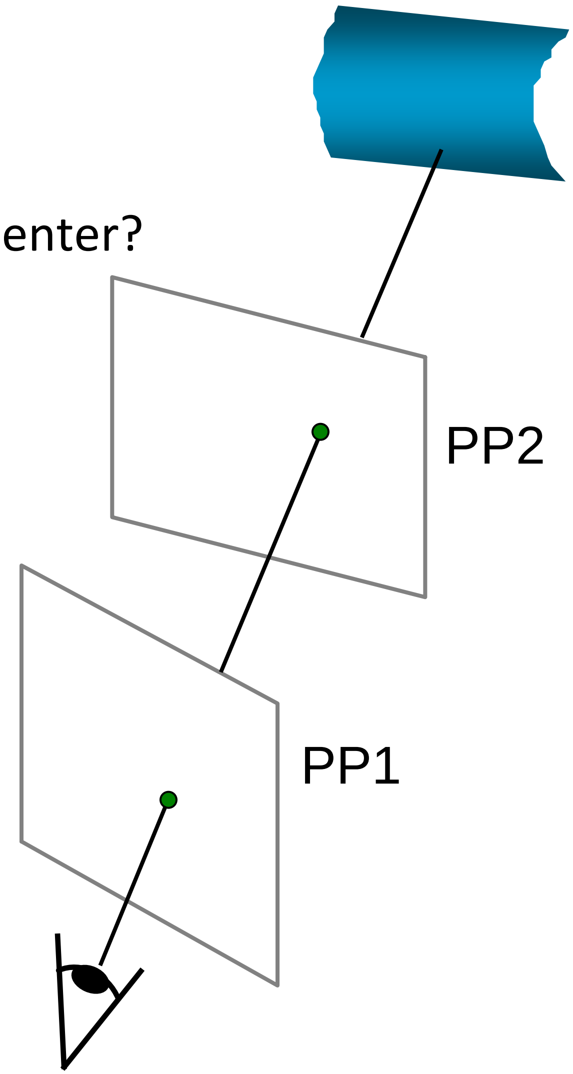
Image Re-projection



- The mosaic has a natural interpretation in 3D
  - The images are re-projected onto a common plane
  - The mosaic is formed on this plane
  - Mosaic is a **synthetic wide-angle camera**

Image Re-projection

- Basic question
  - How to relate two images from the same camera center?
    - how to map a pixel from PP1 to PP2
- Answer
  - Cast a ray through each pixel in PP1
  - Draw the pixel where that ray intersects PP2



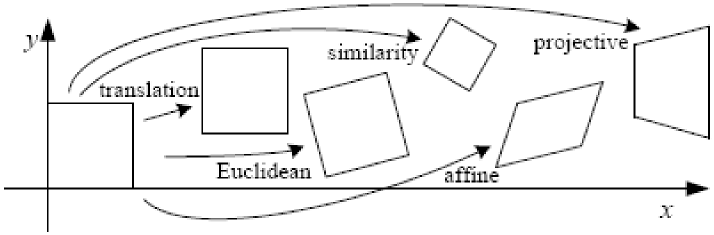
But don't we need to know the geometry of the two planes in respect to the eye?

Observation:
Rather than thinking of this as a 3D re-projection, think of it as a 2D **image warp** from one image to another

Back to Image Warping

Which t-form is the right one for warping PP1 into PP2?

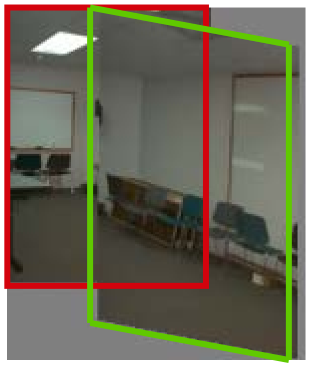
e.g. translation, Euclidean, affine, projective

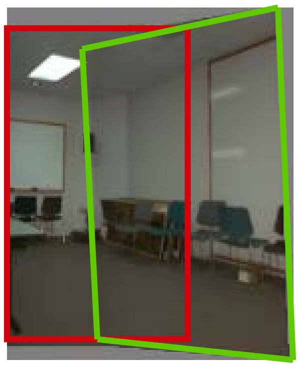


Translation

Affine

Projective





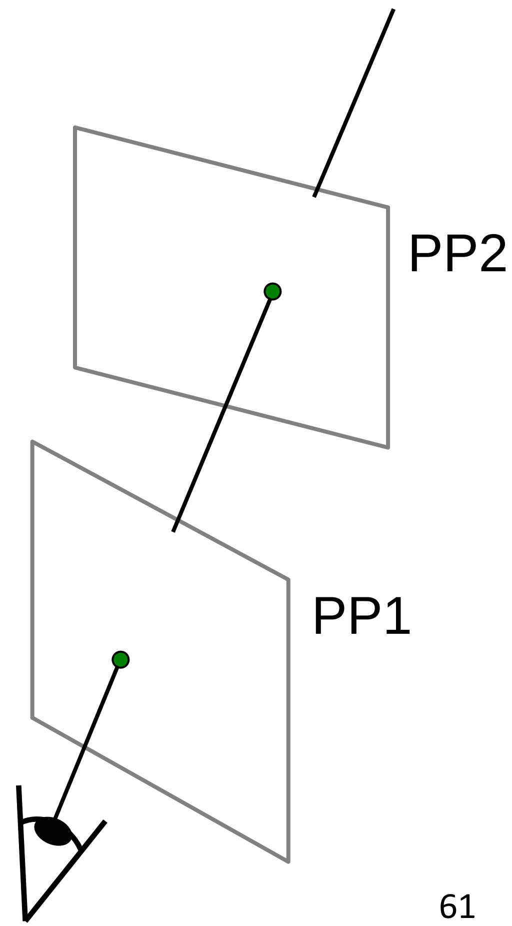
2 unknowns

6 unknowns

8 unknowns

Homography

- A: Projective – mapping between any two PPs with the same center of projection
  - rectangle should map to arbitrary quadrilateral
  - parallel lines aren't
  - but must preserve straight lines
  - same as: project, rotate, re-project



called Homography

$$\underbrace{\begin{bmatrix} w\,x' \\ w\,y' \\ w \end{bmatrix}}_{\mathbf{p}'} = \underbrace{\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\mathbf{p}}$$

To apply a homography $\mathbf{H}$

- Compute $\mathbf{p}' = \mathbf{H}\mathbf{p}$ (regular matrix multiply)
- Convert $\mathbf{p}'$ from homogeneous to image coordinates

Image Warping with Homographies

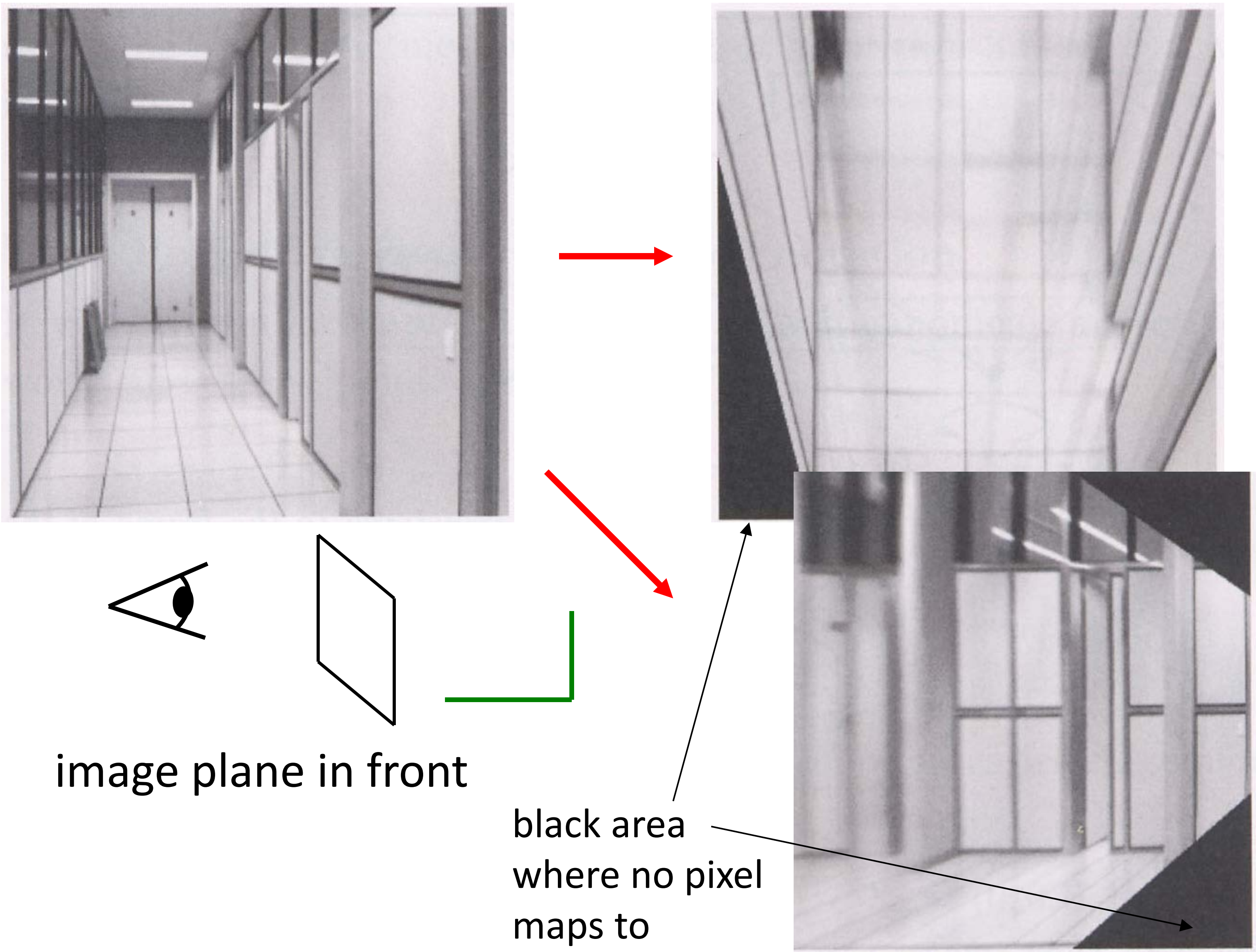
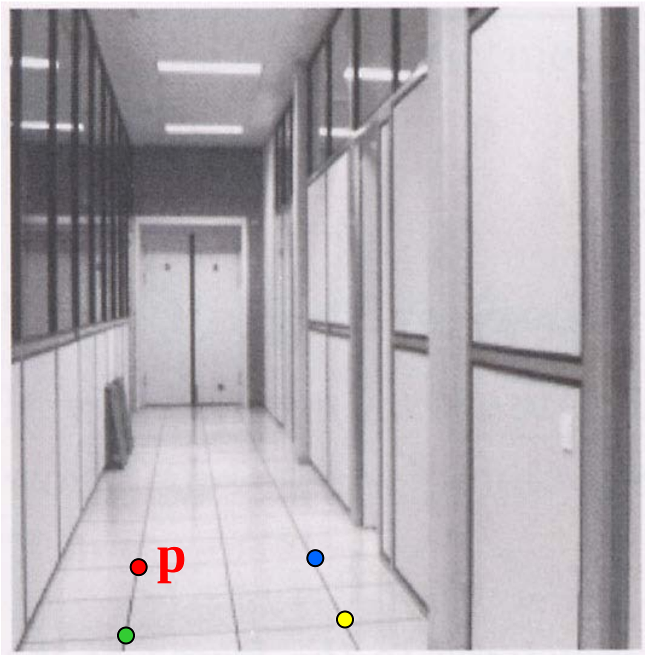
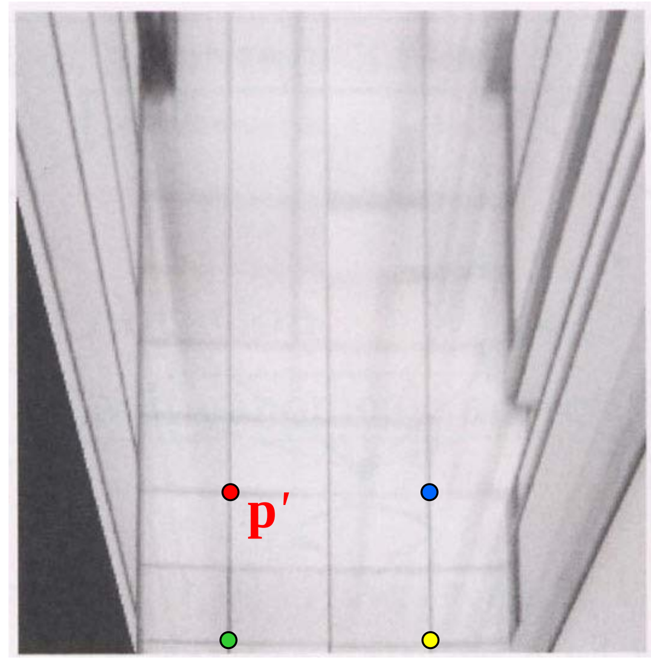


Image Rectification





To unwarp (rectify) an image

- Find the homography $\mathbf{H}$ given a set of $\mathbf{p}$ and $\mathbf{p}'$ pairs
- How many correspondences are needed?
- Tricky to write $\mathbf{H}$ analytically, but we can solve for it!
  - Find such $\mathbf{H}$ that "best" transforms points $\mathbf{p}$ into $\mathbf{p}'$
  - Use least-squares! (over-constrained)

Solving for Homographies

$$\mathbf{p}' = \mathbf{H}\mathbf{p}$$

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

- Can set scale factor i=1. So, there are 8 unkowns.
- Set up a system of linear equations:

$$\mathbf{A}\mathbf{h} = \mathbf{b}$$

- where vector of unknowns $\mathbf{h} = [a,b,c,d,e,f,g,h]^T$
- Need at least 8 eqs, but the more the better…
- Solve for h. If over-constrained, solve using least-squares:

$$h^* = \arg\min_{\mathbf{h}} \|\mathbf{A}\mathbf{h} - \mathbf{b}\|^2$$

- Can be done in MATLAB using **“\”** command

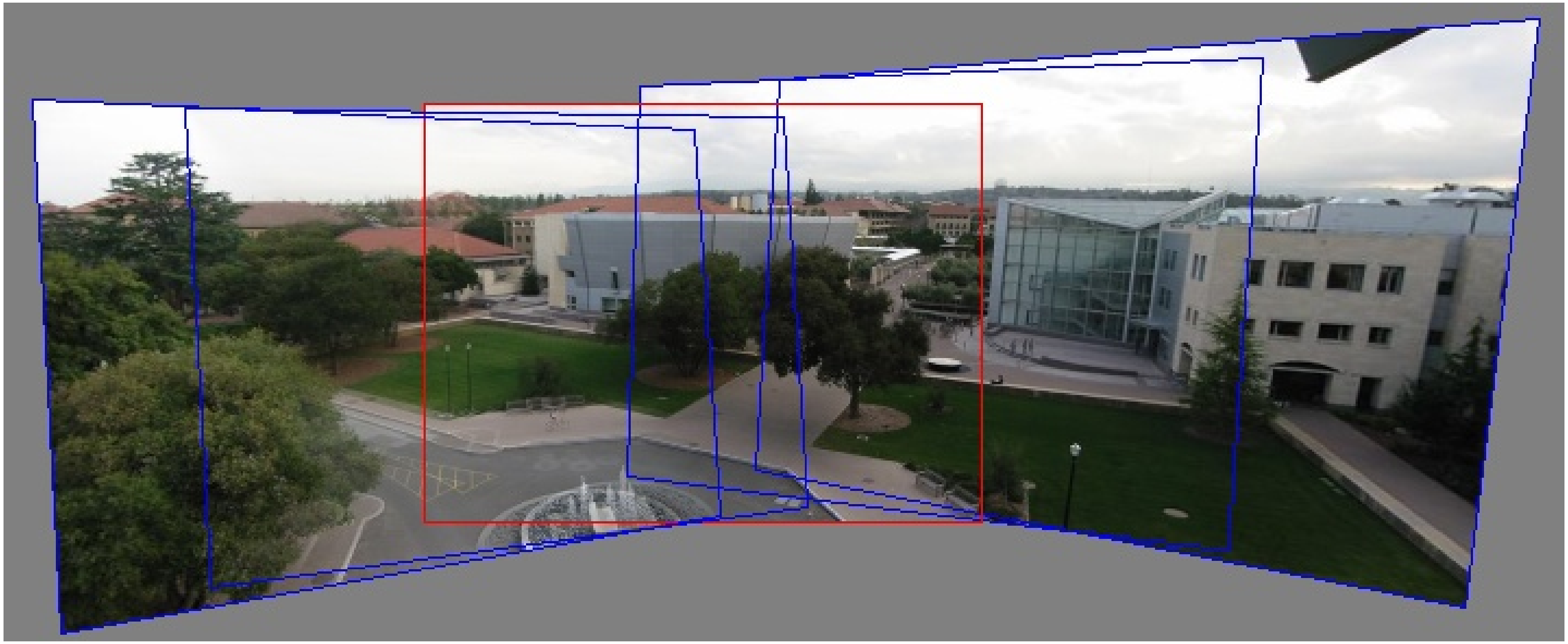
Fun with Homographies

original image

St.Petersburg
photo by A. Tikhonov

virtual camera rotations

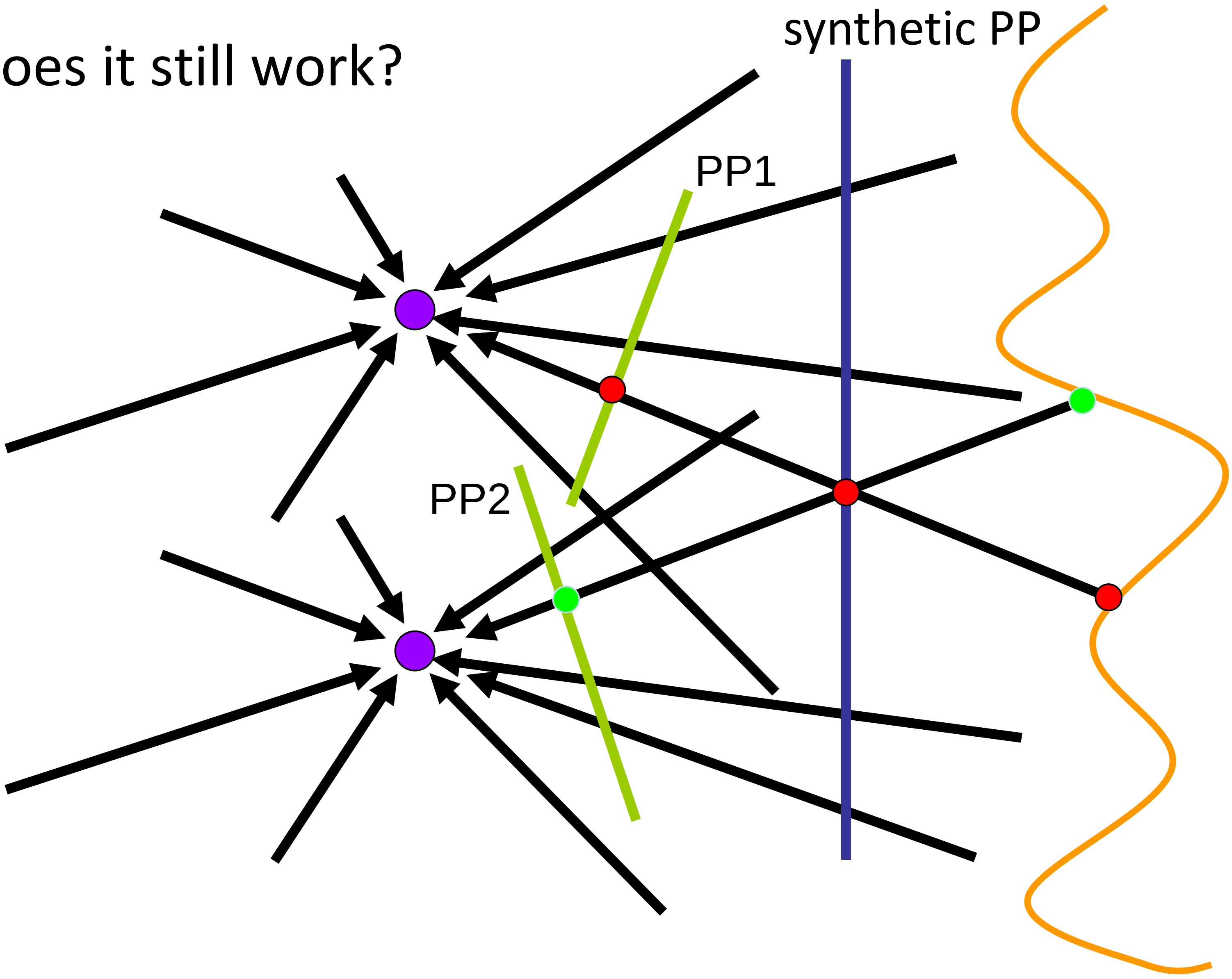
Panoramas



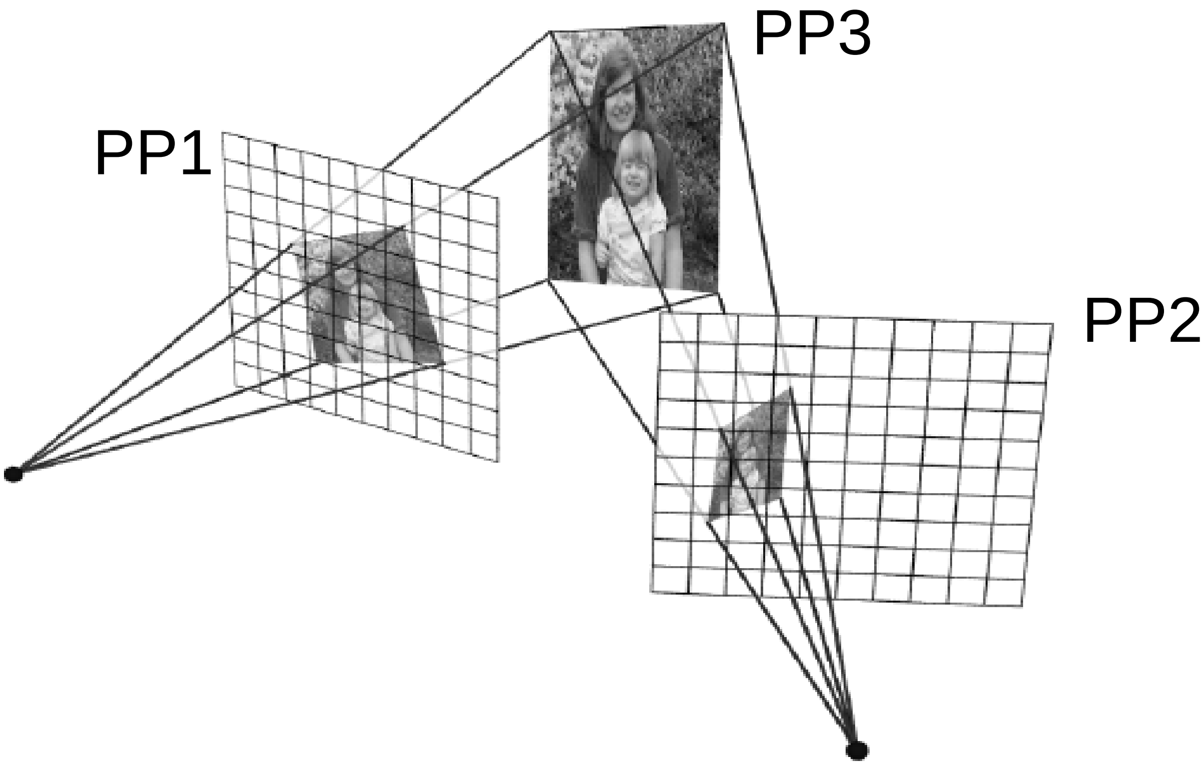
1. Pick one image (red)
2. Warp the other images towards it (usually, one by one)
3. Blend

Changing Camera Center

- Does it still work?

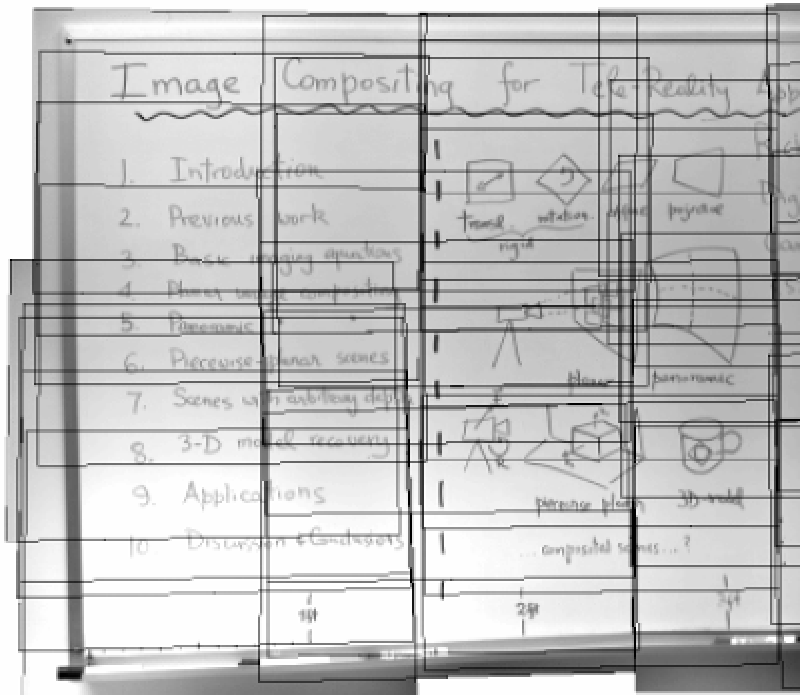


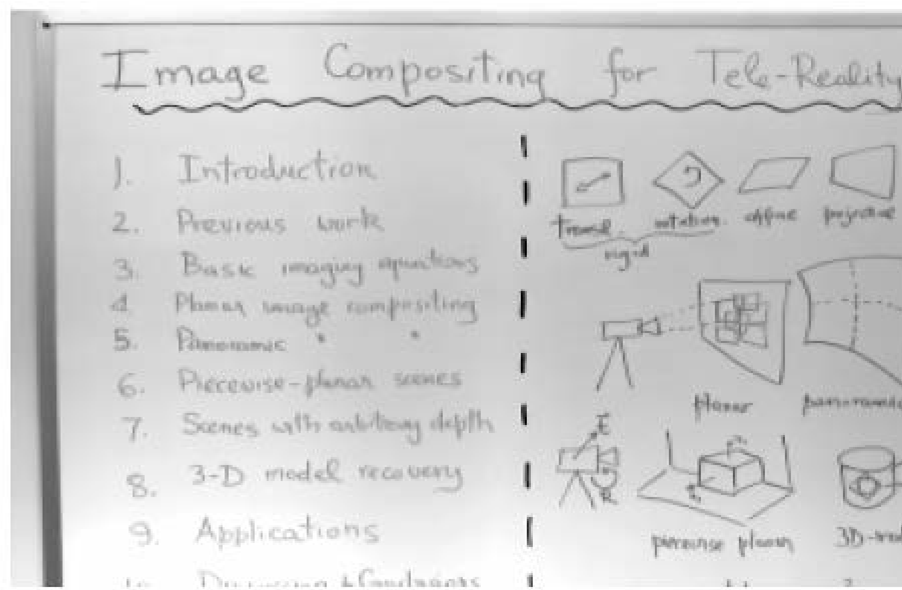
Planar Scene (or Far Away)



- PP3 is a projection plane of both centers of projection, so we are OK!
- This is how big aerial photographs are made

Planar Mosaic





Application of Homography in Graphics

Anamorphosis

海洋龍宮 Art by Kurt Wenner
Photo by 魚夫

http://en.wikipedia.org/wiki/Anamorphosis

http://www.illusionworks.com/

http://www.ted.com/talks/lang/eng/al_seckel_says_our_brains_are_mis_wired.html

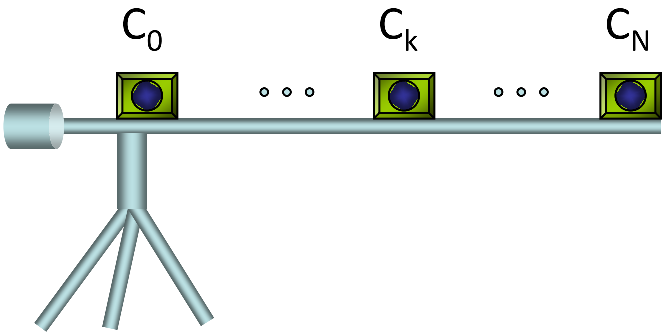
István Orosz: Mirror Anamorphosis with Column

[wikipedia]

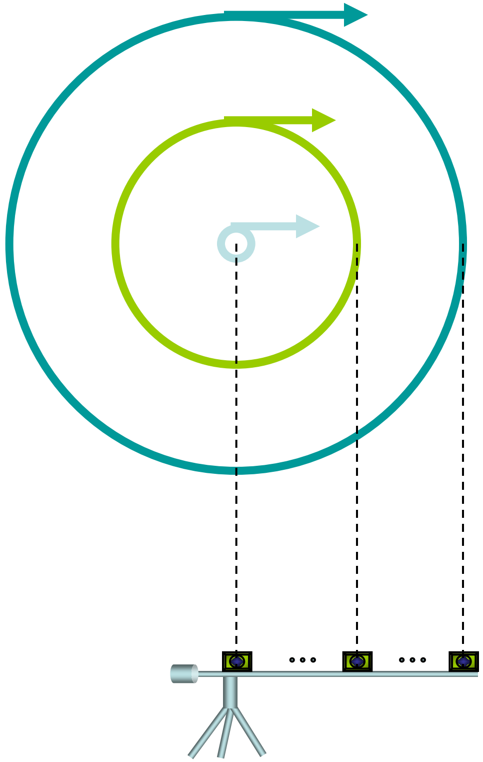
Another Example of Homographies

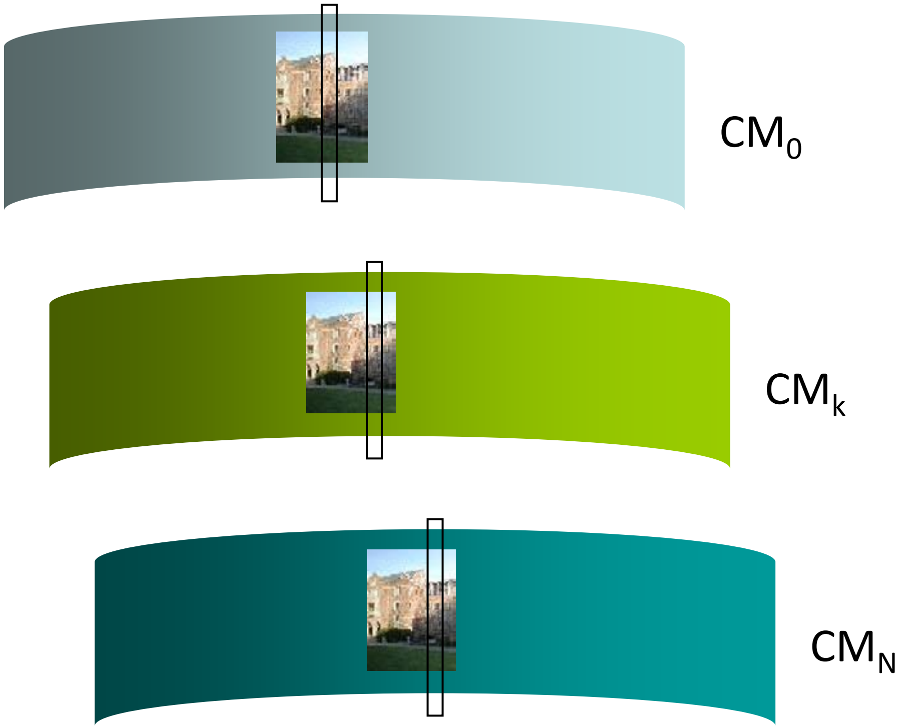
created by Brett Allen

Concentric Mosaic [Shum & He]

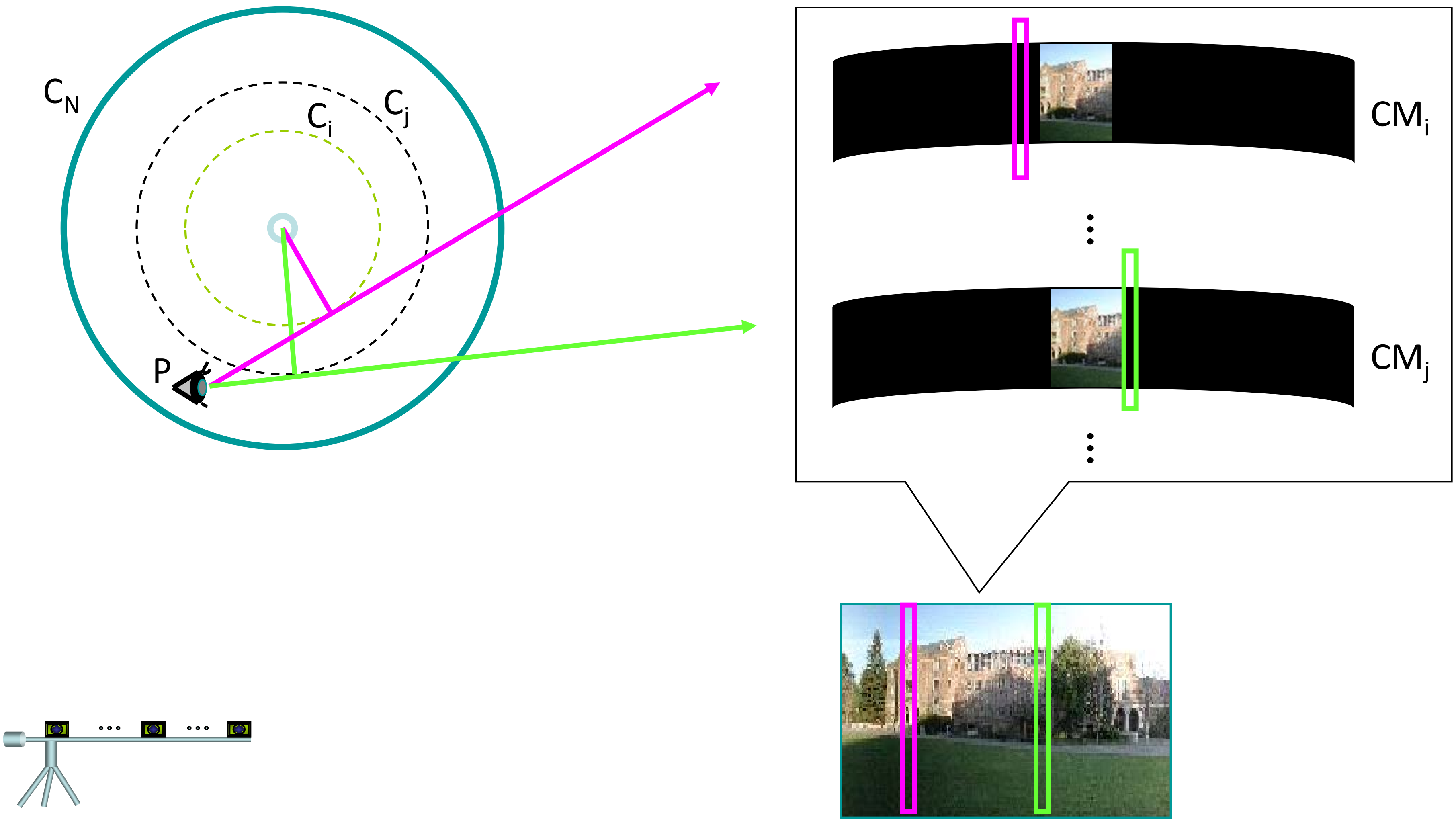


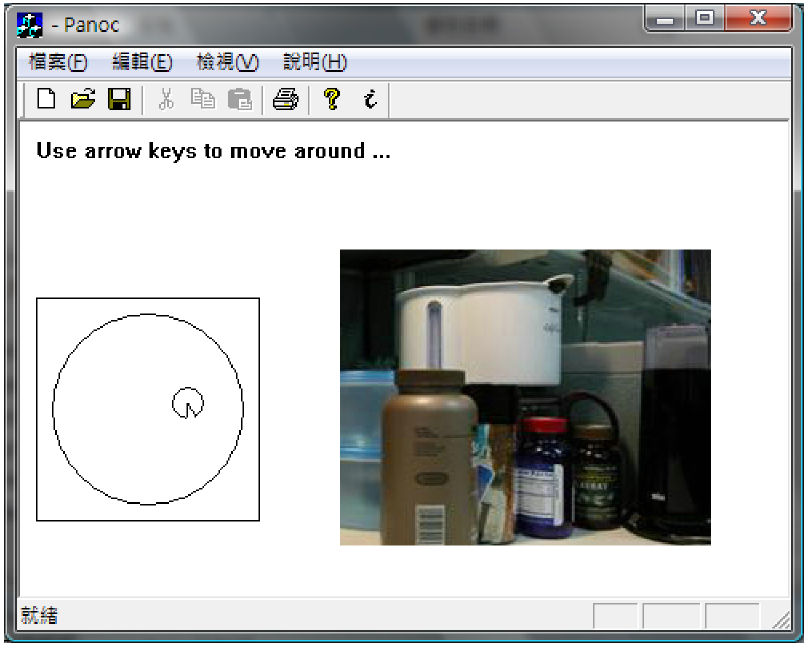
Construction of a Concentric Mosaic

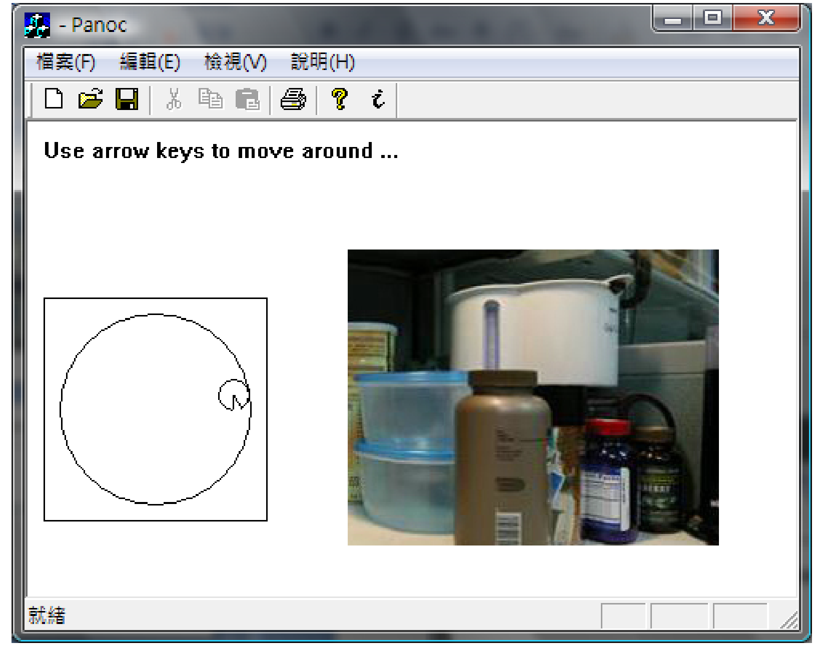


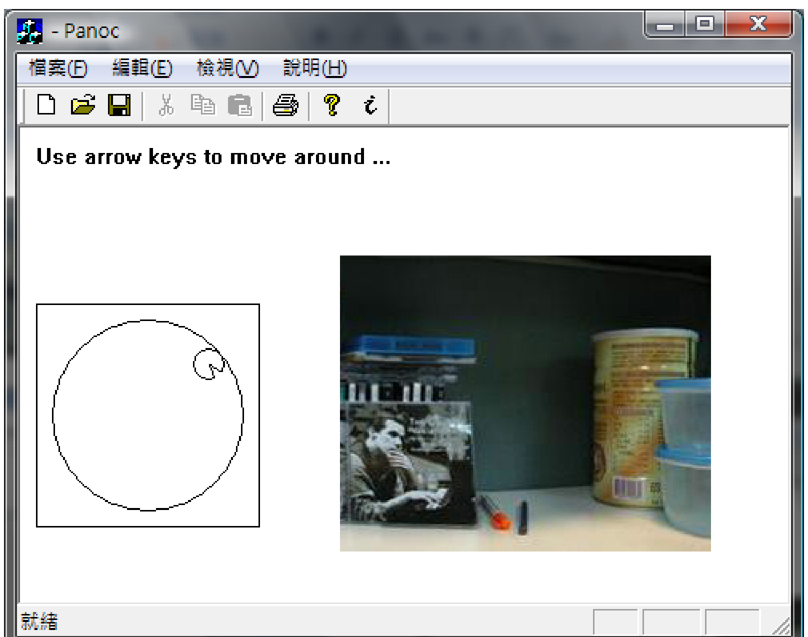


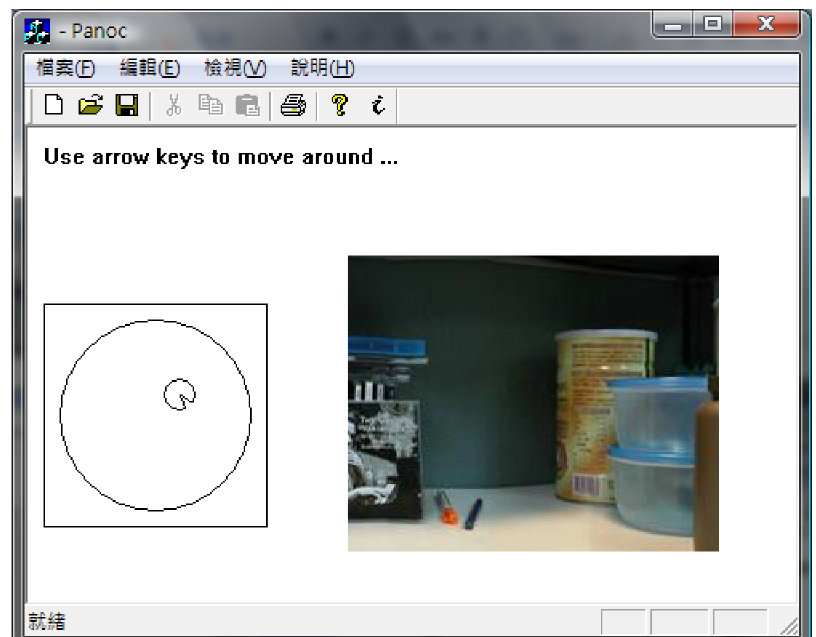
Rendering a Novel View




- Panoc
檔案(F) 編輯(E) 檢視(V) 說明(H)
Use arrow keys to move around ...
就緒


- Panoc
檔案(F) 編輯(E) 檢視(V) 說明(H)
Use arrow keys to move around ...
就緒


- Panoc
檔案(F) 編輯(E) 檢視(V) 說明(H)
Use arrow keys to move around ...
就緒


- Panoc
檔案(F) 編輯(E) 檢視(V) 說明(H)
Use arrow keys to move around ...
就緒

Sweep Panorama