COSC579: Image Morphing

Jeremy Bolton, PhD
Assistant Teaching Professor

GEORGETOWN UNIVERSITY

Outline

I. Multi-Image Vision: Example Morphing
   I. Finding Correspondences
   II. Global vs Local Warping
      I. Mesh
      II. Delaunay triangulation
   III. Intermediate Warp Representations
      I. Linear Interpolation
   IV. Morphing

Morphing Examples

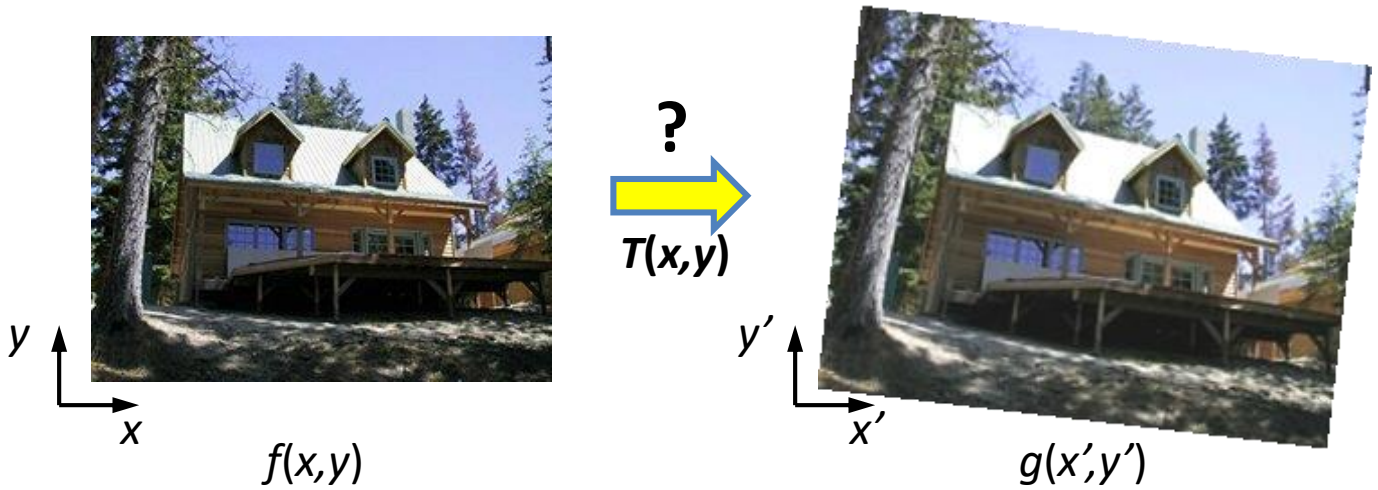
- 500 years of female portraits
  - https://youtu.be/nUDIoN-_Hxs
- Michael Jackson
  - https://youtu.be/4jd6dNrJRh4

Morphing Problem

- Can we use image warping techniques to “morph” one image into another?

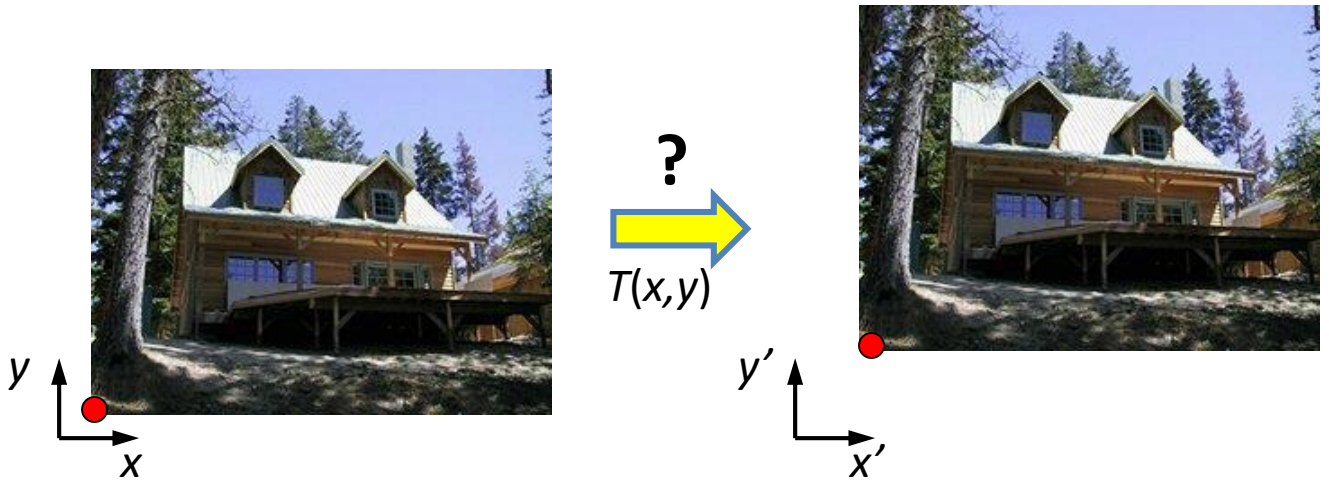
- Yes, but some problems will be encountered
  - Morph is a seamless transition but a warp is a (instant) transformation: How can we seamlessly transition from one image to another?
  - Point correspondences: Finding point correspondences in two images of the same scene is hard enough! When the images are of different scenes … What can be used as correspondences and how can we find them?

Recall: Recovering Transformations



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

Translation: # correspondences?

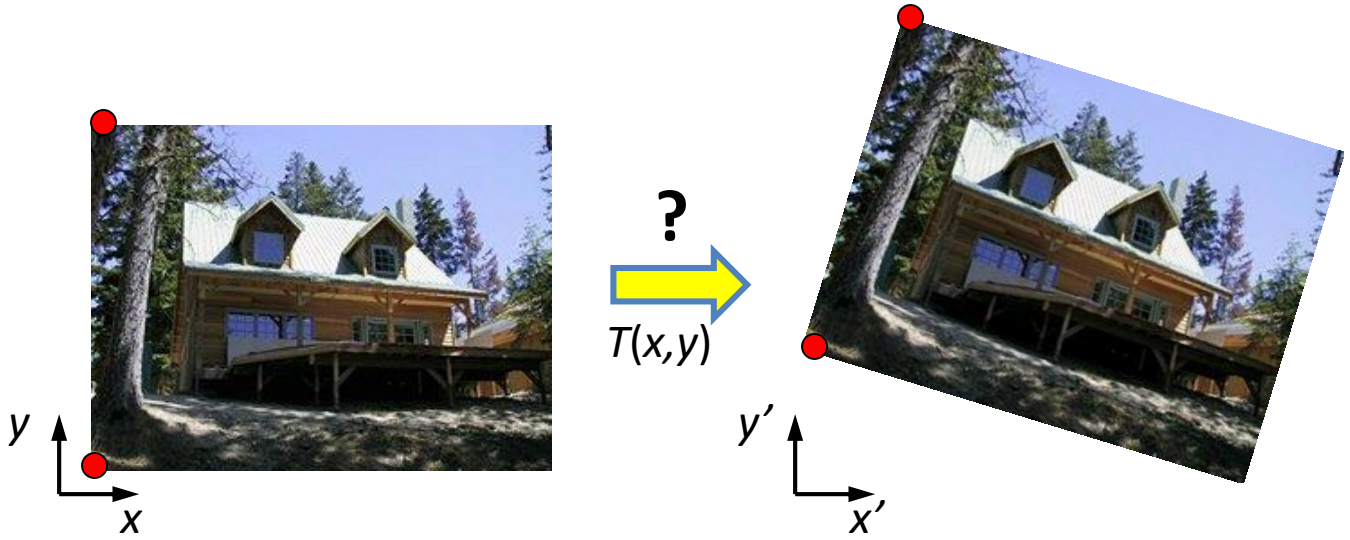


- How many correspondences needed for translation?
- How many Degrees of Freedom?
- What is the transformation matrix?

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

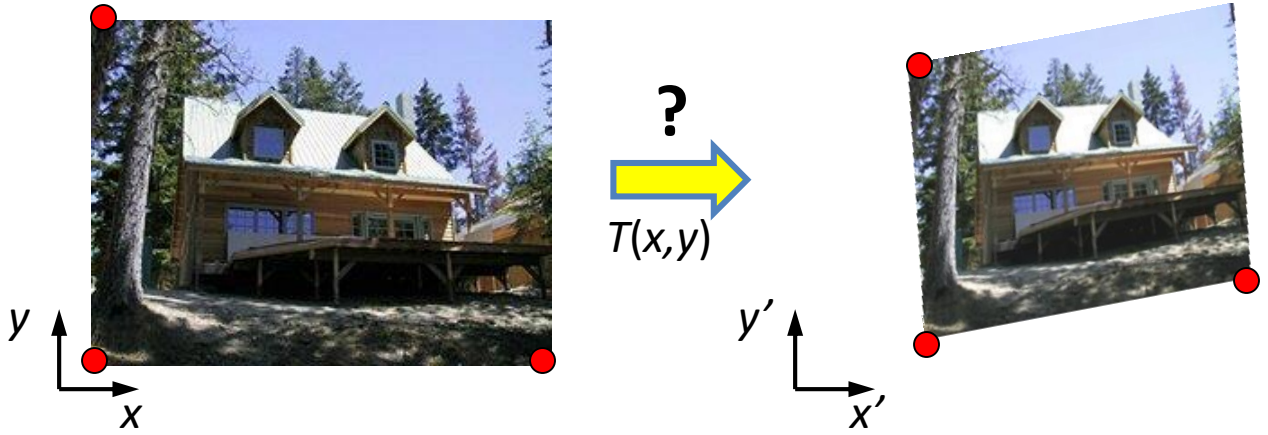
GEORGETOWN UNIVERSITY

Euclidian: # correspondences?



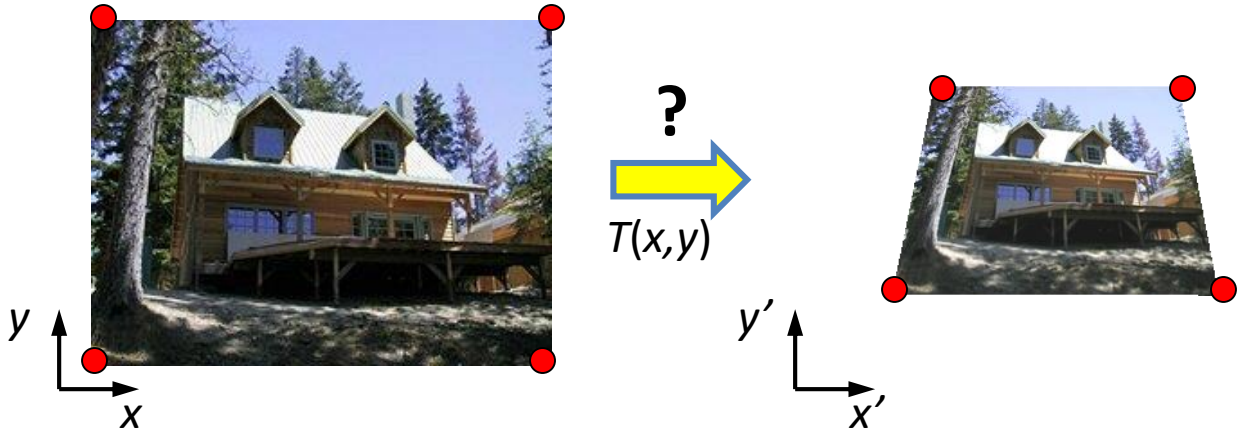
- How many correspondences needed for translation+rotation?
- How many DOF?

Affine: # correspondences?



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

Projective: # correspondences?



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

GEORGETOWN UNIVERSITY

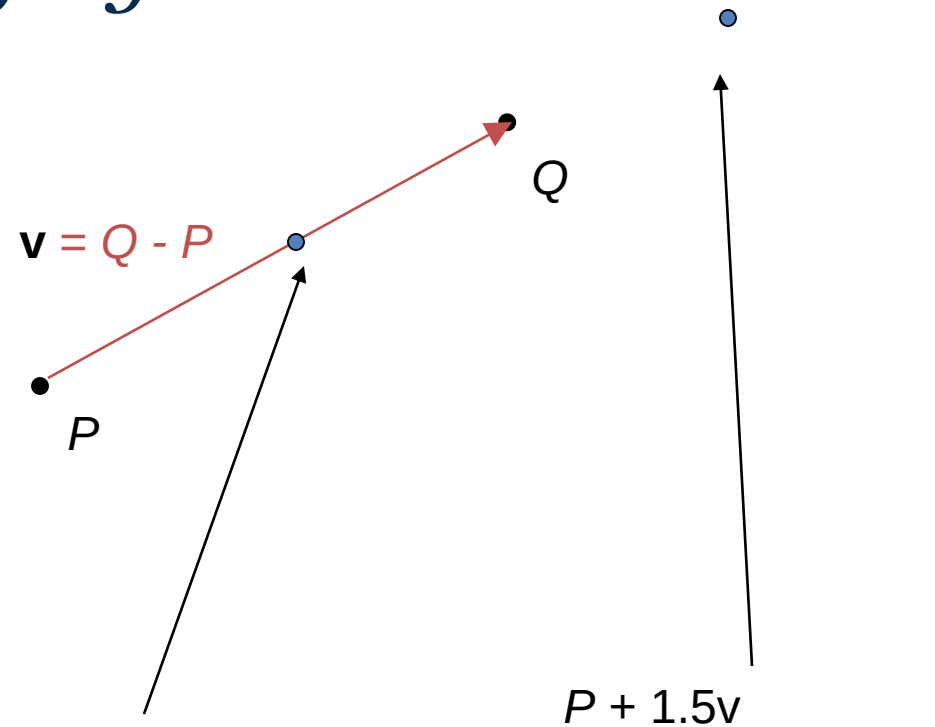
Facilitating Seamless Transition

- Simple idea.
  - Slowly cross-fade or cross-dissolve via interpolation, one image into another using some time index t, which is a sampling of the values between 0 and 1.
    - Monotonic
    - Uniform

$$Image_{halfway} = (1-t)*Image_1 + t*image_2$$

Averaging Points

What's the average of P and Q?



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

- P and Q can be anything:
  - points on a plane (2D) or in space (3D)
  - Colors in RGB or HSV (3D)
  - Whole images (m-by-n D)… etc.

GEORGETOWN UNIVERSITY

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 is the images are not aligned?

Idea #2: Align, then cross-disolve

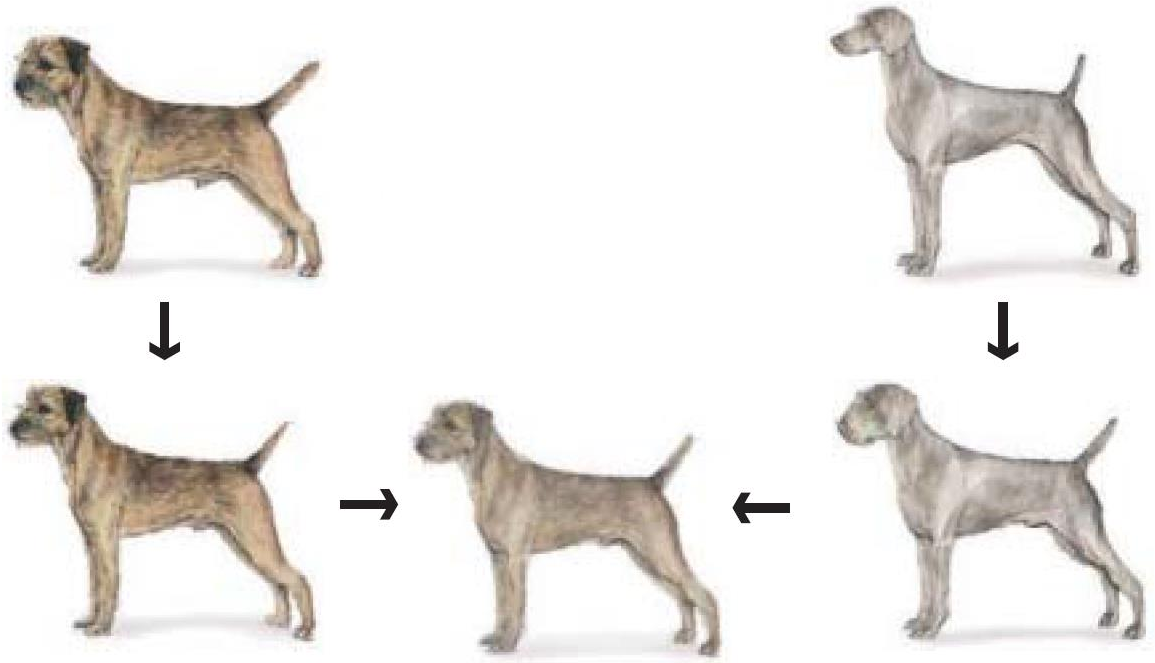
- Align first, then cross-dissolve
  - Alignment using global warp – picture still valid

GEORGETOWN UNIVERSITY

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

Idea #3: Local warp, then cross-dissolve



- Morphing procedure:

- *for every t,*

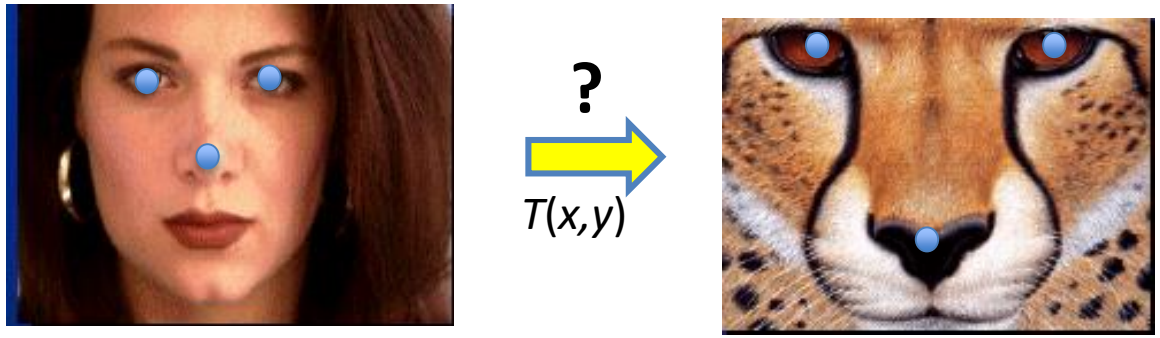
1. Find the average shape (the “mean dog”)
   - local warping
2. Find the average color
   - Cross-dissolve the warped images

GEORGETOWN UNIVERSITY

Local (non-parametric) Image Warping

- Need to specify a more detailed warp function
  - Global warps were functions of a few 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)

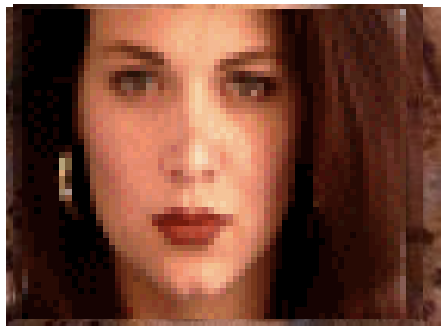
Morphing



- How can we define a warp between images of two different scenes? What's our goal?

- Identify correspondences so as to achieve desired effect
  - Is there a single homography that will map each pixel correctly in this instance?
  - Will a global warping scheme work?

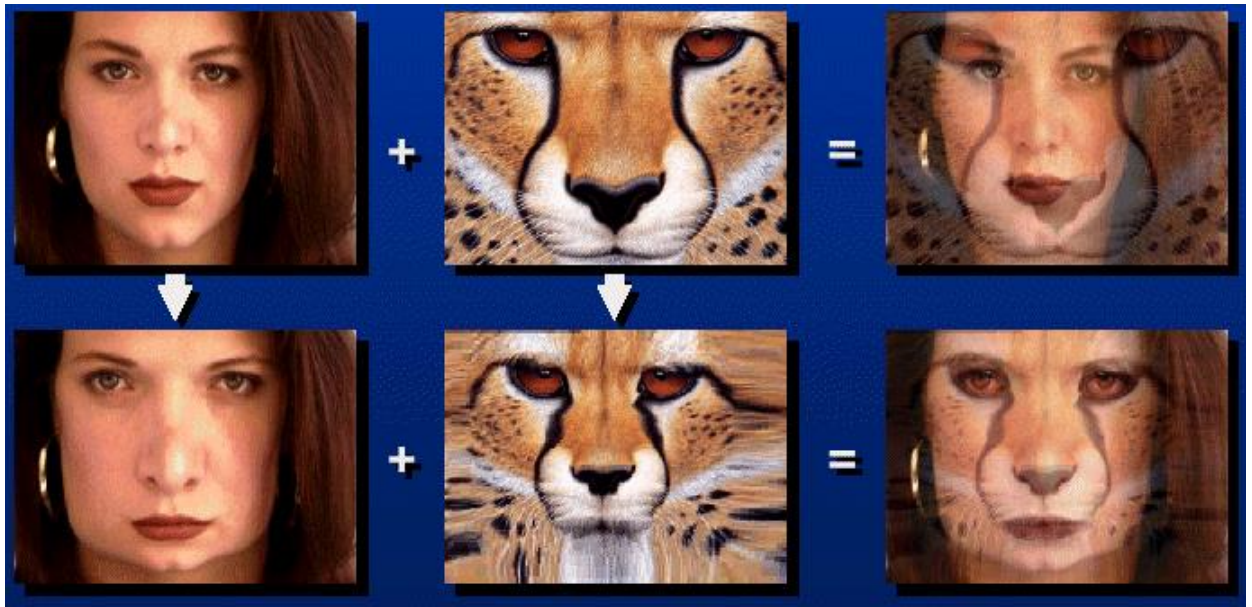
Morphing = Object Averaging


- 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

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



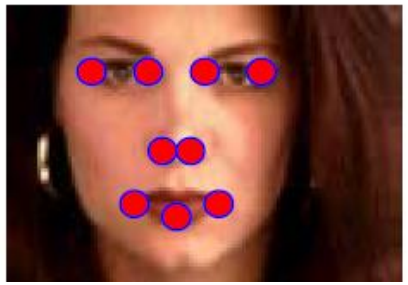
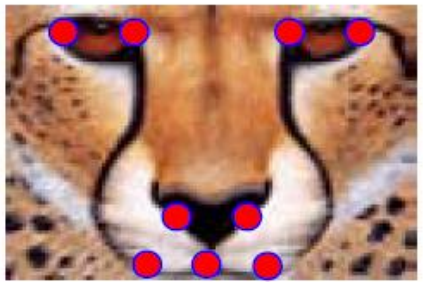
GEORGETOWN UNIVERSITY

Creating a Warp Sequence

- **Input:** two images $\mathbf{I}_0$ and $\mathbf{I}_N$


- **Output:** image seq. $\mathbf{I}_i$, with *i=1..N-1*

- User specifies sparse correspondences on the images
  - Pairs of vectors $\{(\mathbf{p}^0_j, \mathbf{p}^N_j)\}$

GEORGETOWN UNIVERSITY

Creating a Warp Sequence

- For each intermediate frame $\mathbf{I}_t$
  - Interpolate feature locations $\mathbf{p}^t_i = (1- \alpha(t))\, \mathbf{p}^0_i + \alpha(t)\, \mathbf{p}^1_i$
  - Perform **two** warps: one for $\mathbf{I}_0$, one for $\mathbf{I}_1$
    - Deduce a dense warp field from a few pairs of features
    - Warp the pixels
  - Linearly interpolate the two warped images

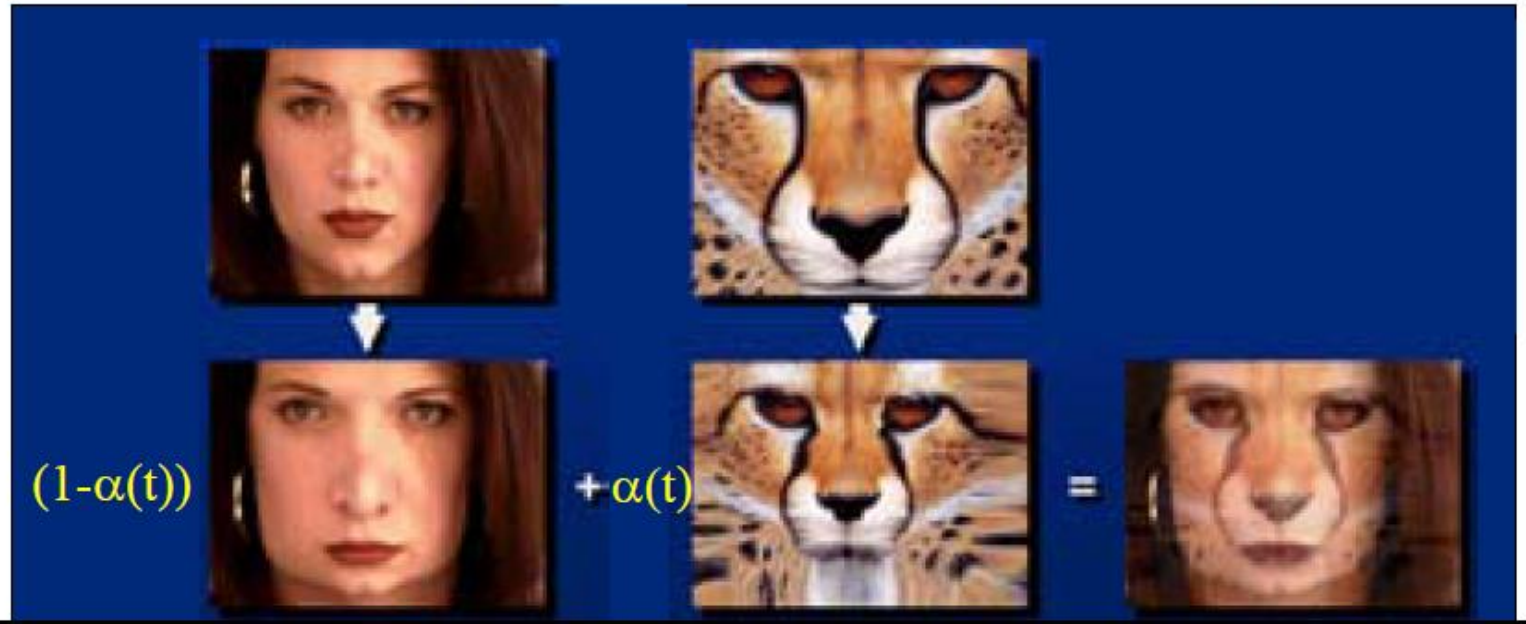
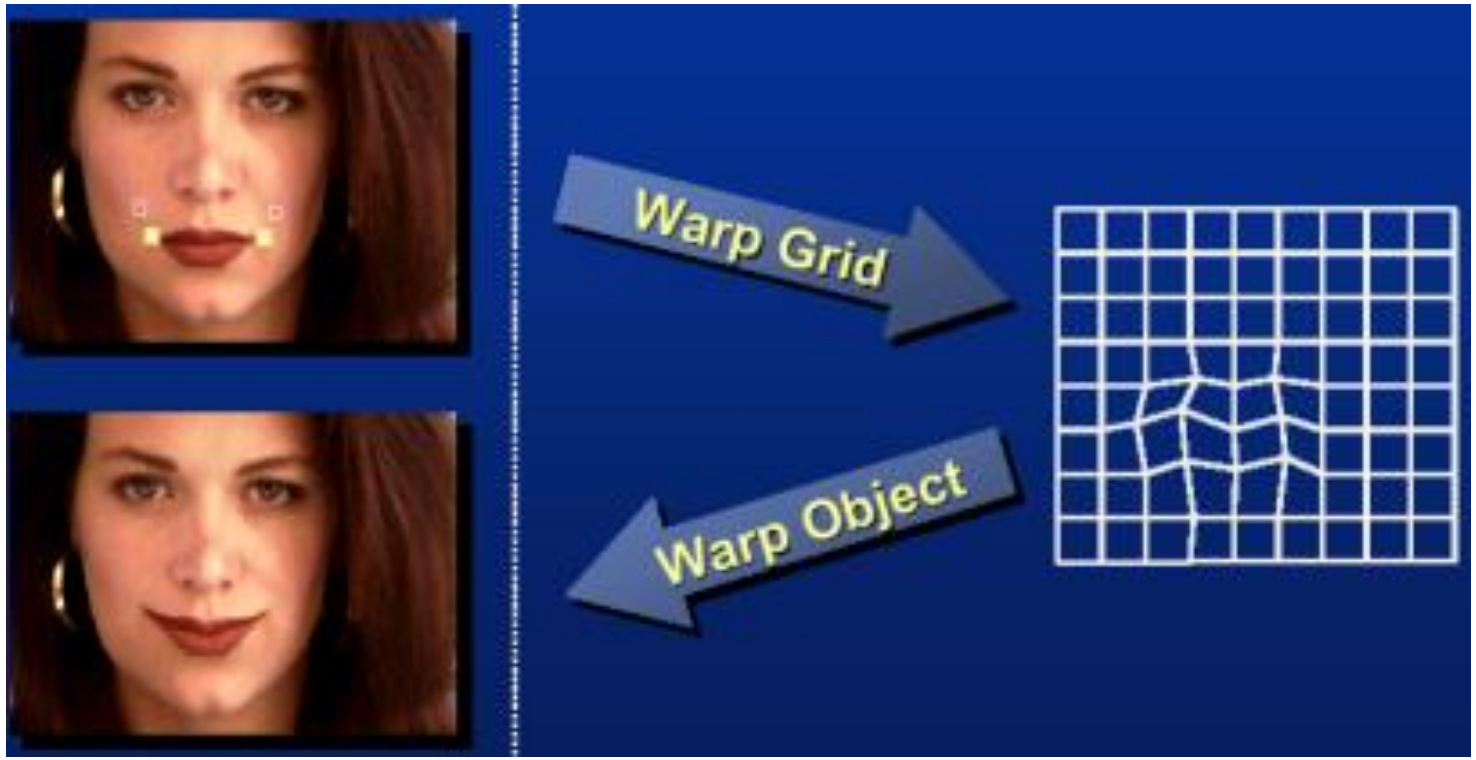


Image Warping – non-parametric

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

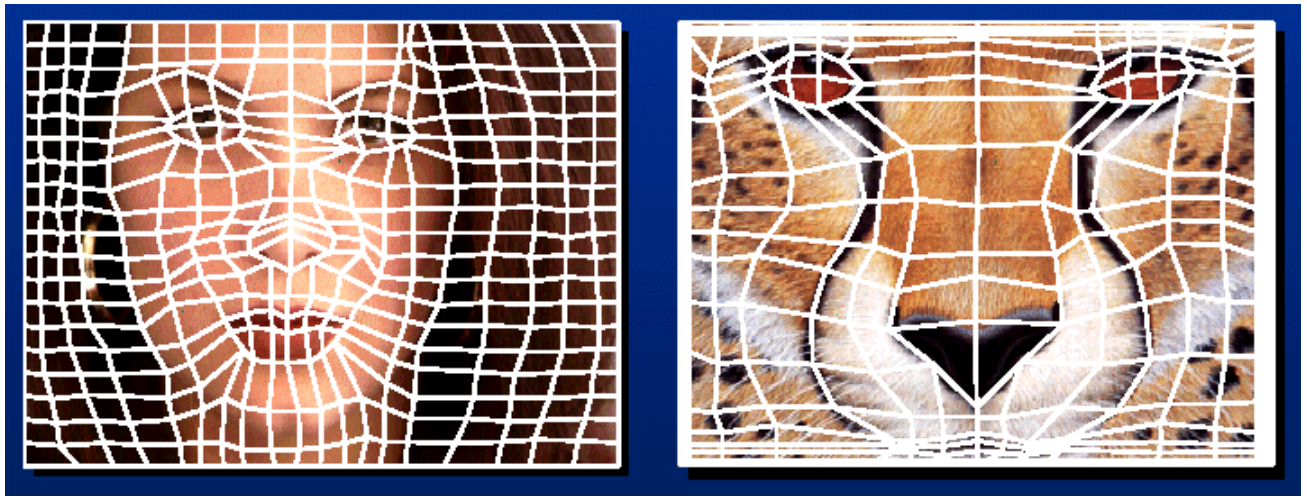


Warp specification - dense

- 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 an entire grid.

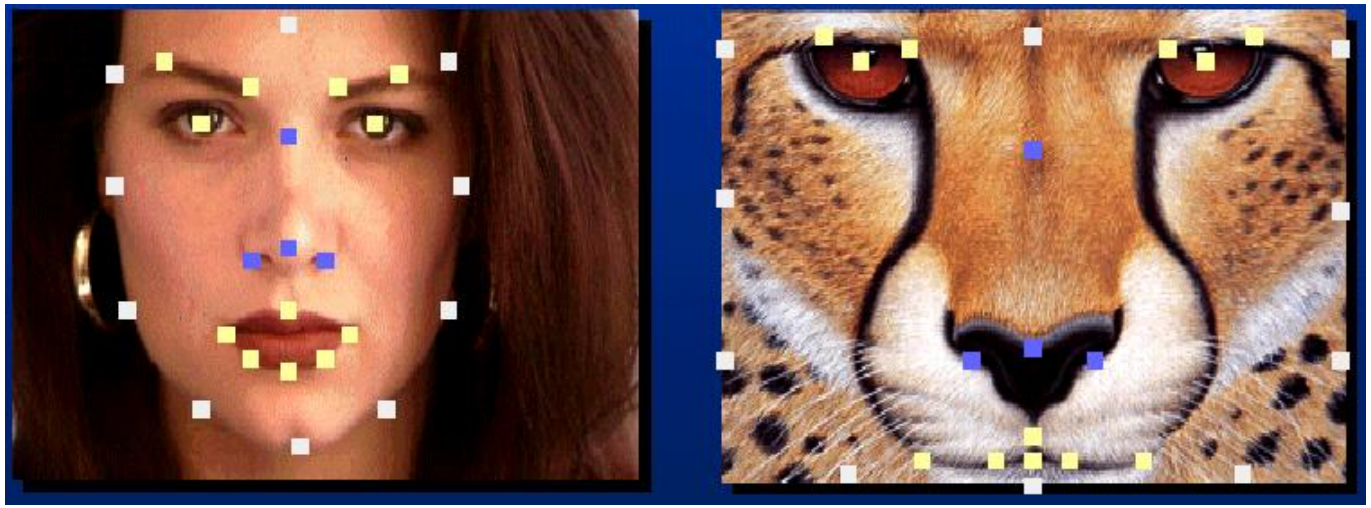
GEORGETOWN UNIVERSITY

Warp specification - sparse

- How can we specify the warp?

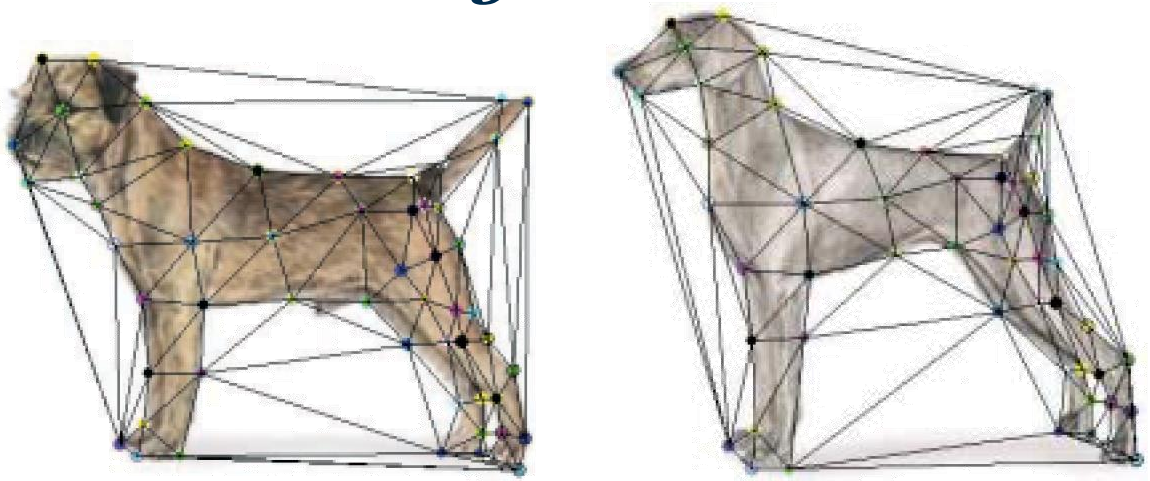
  Specify corresponding *points*

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



How do we go from feature points to pixels?

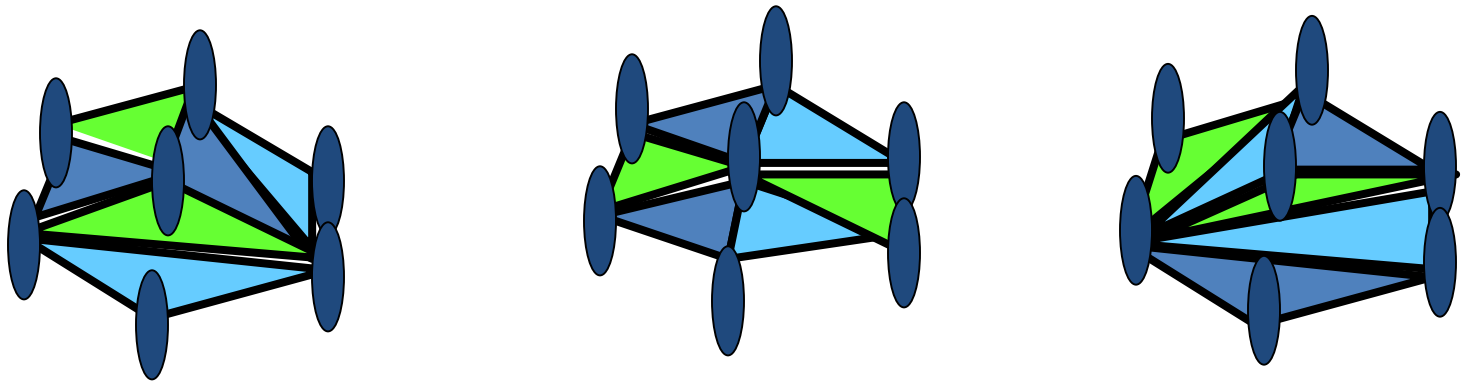
Triangular Mesh



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!

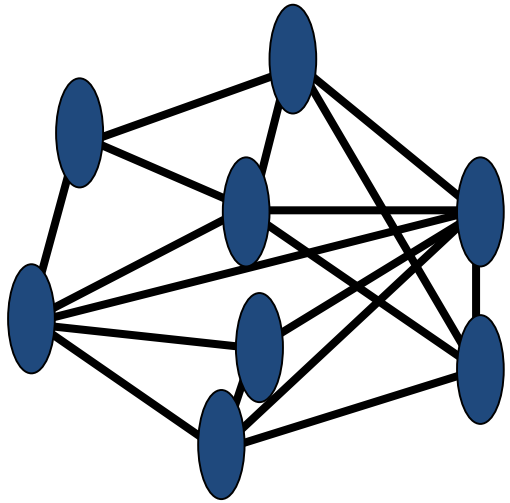
Triangulations

- 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.



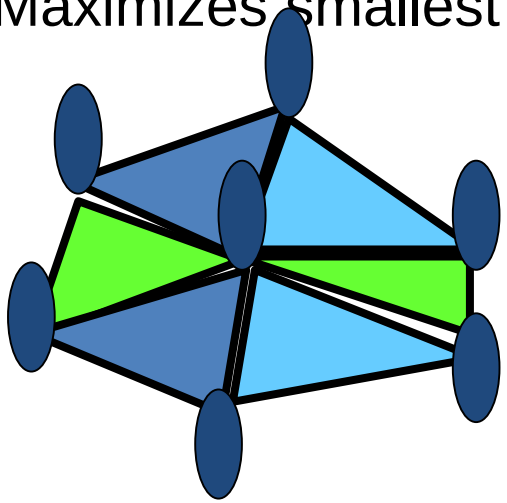
An $O(n^3)$ Triangulation Algorithm

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

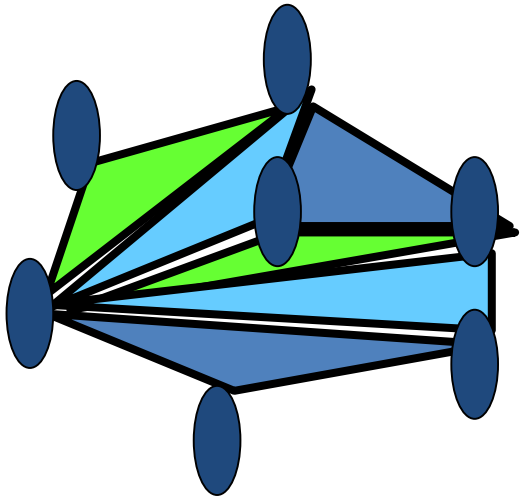


"Quality" Triangulations

- 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



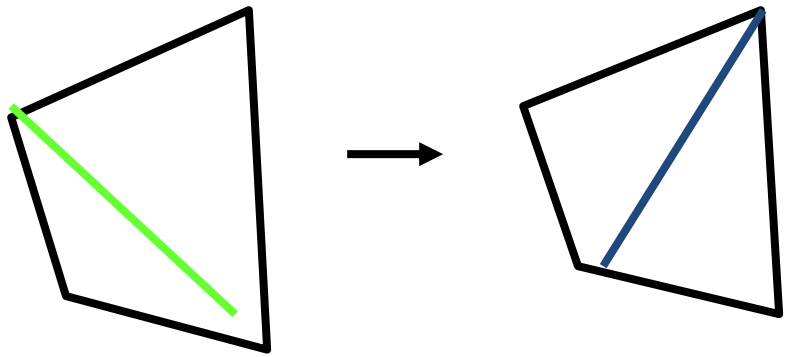
good



bad

Improving a Triangulation

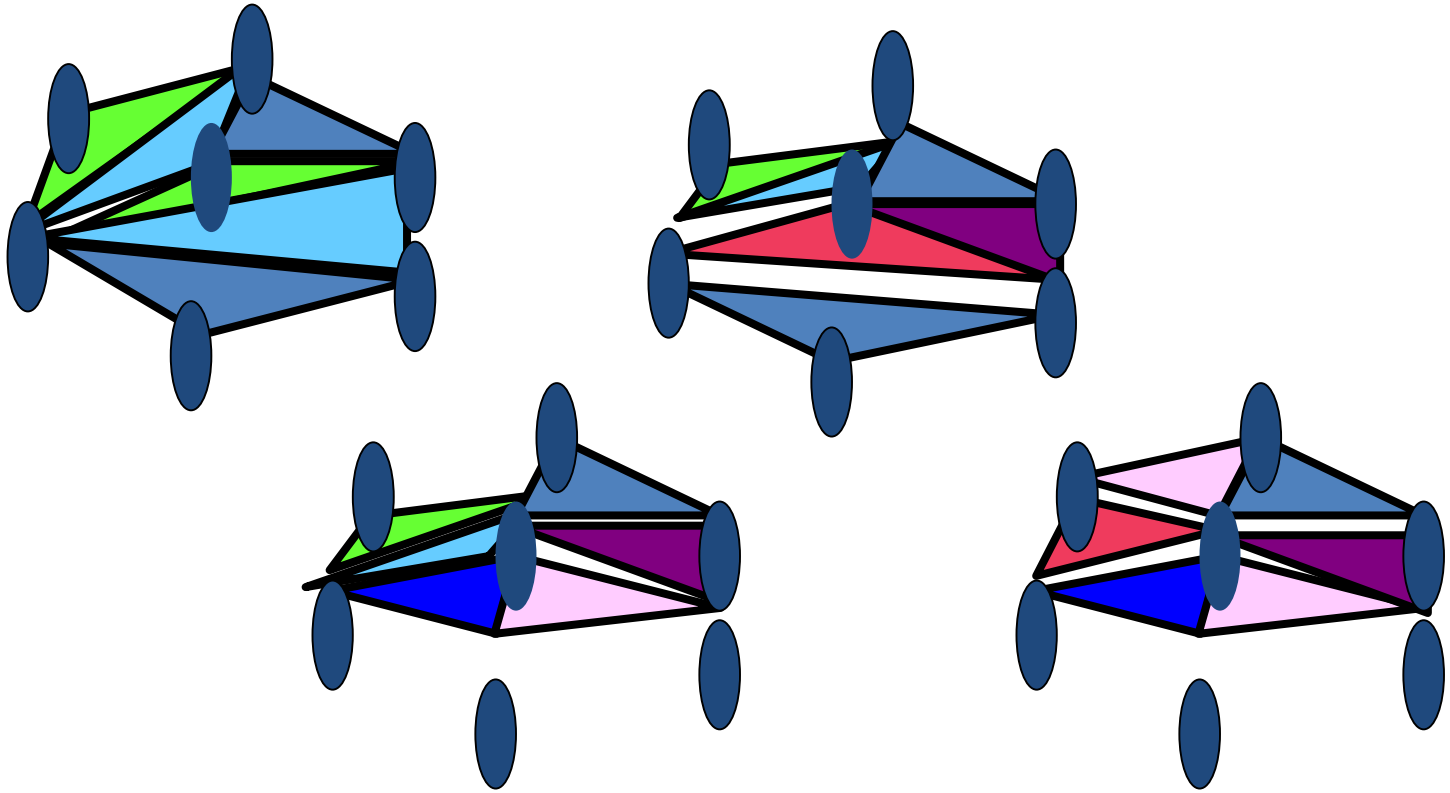
- In any convex quadrangle, an *edge flip* is possible. If this flip *improves* the triangulation locally, it also improves the global triangulation.



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

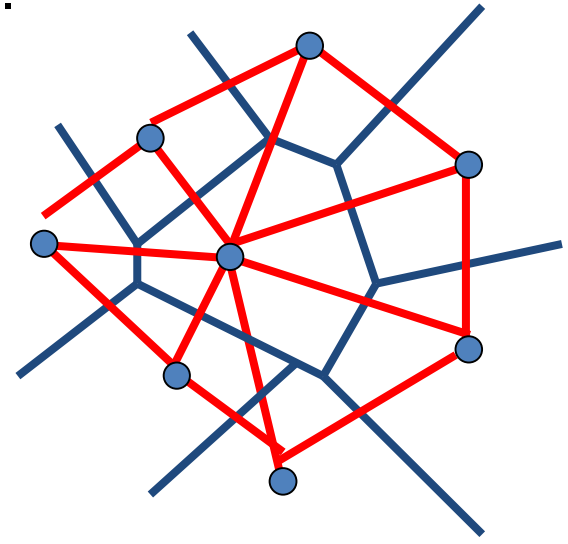
Naïve Delaunay Algorithm

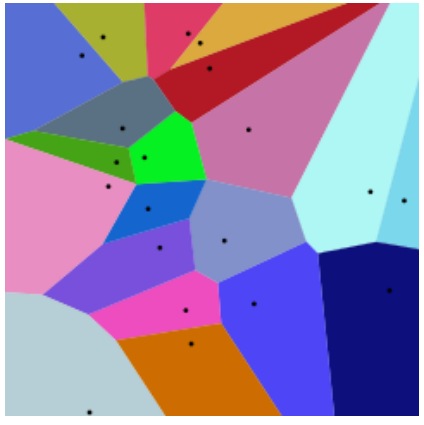
- Start with an arbitrary triangulation. Flip any illegal edge until no more exist.
- Could take a long time to terminate.



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.
- This is what Matlab's `delaunay` function uses.



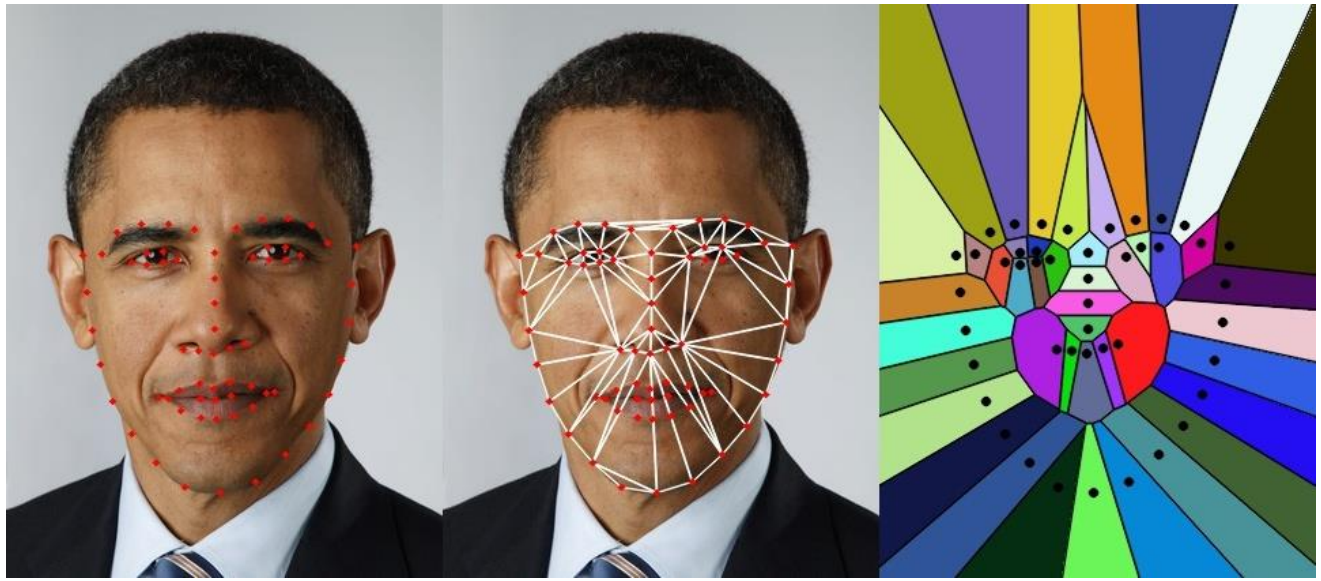


Given a set of points in a plane, a Voronoi diagram partitions the space such that the boundary lines are equidistant from neighboring points. Left is an example of a Voronoi diagram calculated from the points shown as black dots. You will notice that every boundary line passes through the center of two points. If you connect the points in neighboring Voronoi regions, you get a Delaunay triangulation!

GEORGETOWN UNIVERSITY

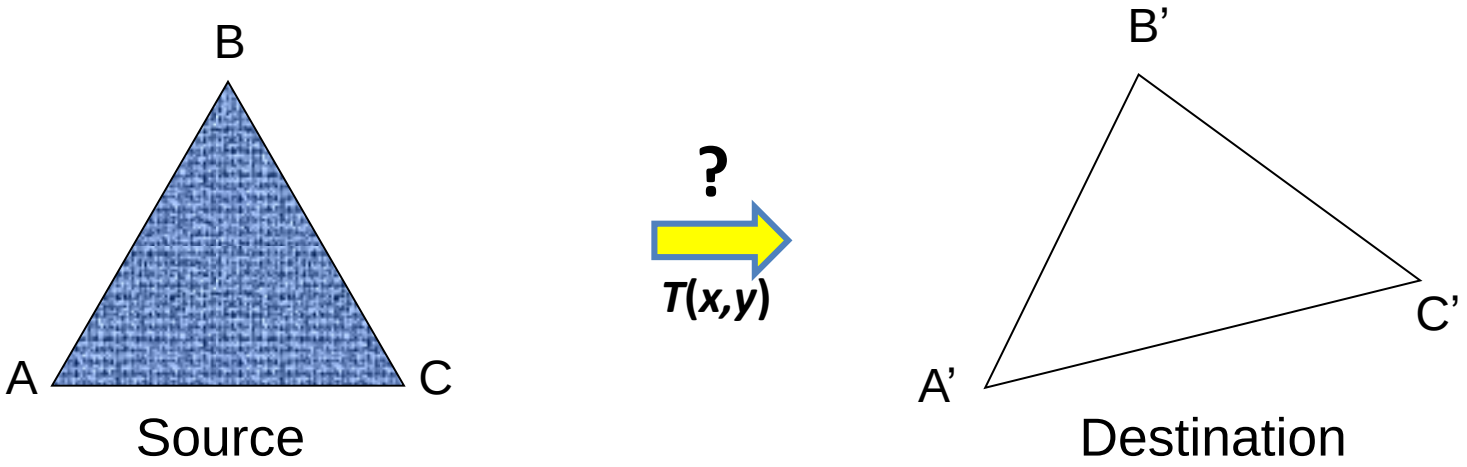
Voronoi (and Delaunay) Partitions in Python

- OpenCV:
  - https://www.learnopencv.com/delaunay-triangulation-and-voronoi-diagram-using-opencv-c-python/



GEORGETOWN UNIVERSITY

Example: warping triangles



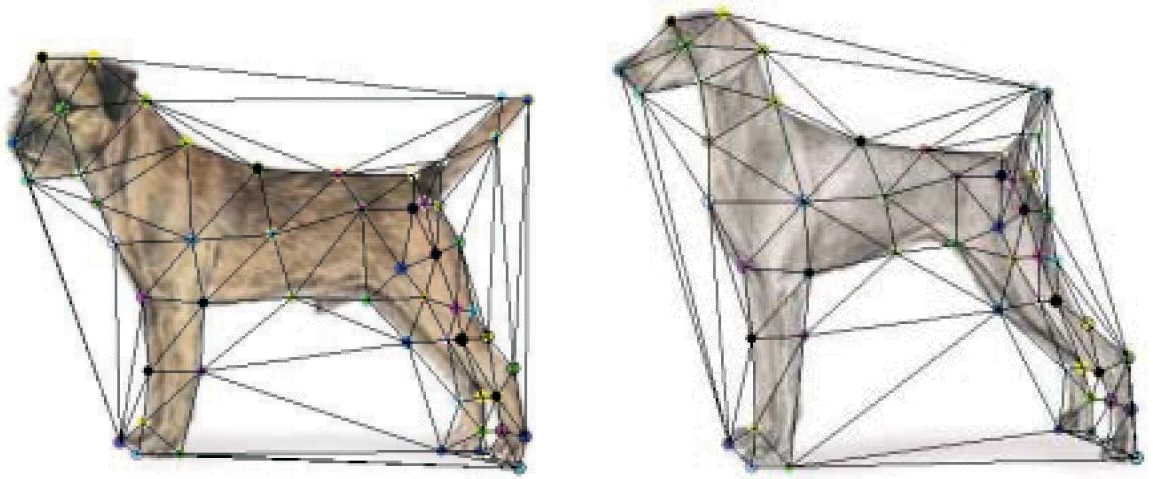
- Given two triangles: ABC and A'B'C' in 2D
- Need to find transform T to transfer all pixels from one to the other.
- Affine Assumption?
- How can we compute the transformation matrix?
  - Least Squares

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

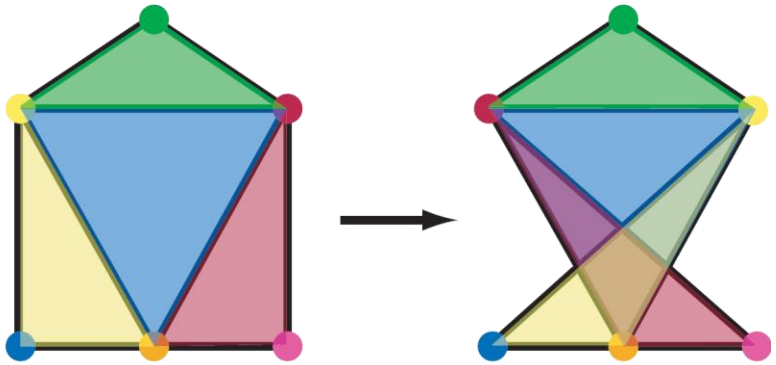
GEORGETOWN UNIVERSITY

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



Other Issues



- Beware of folding
  - You are probably trying to do something “weird” (3D folding)
- Remember to interpolate an intermediate coordinate system (for each triangle)
  - Map both images to the intermediate representation.
  - Implement as an inverseWarp to avoid “holes”
    - (inverse wrt the intermediate coordinate system)
- Extrapolation can sometimes produce interesting effects