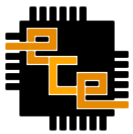
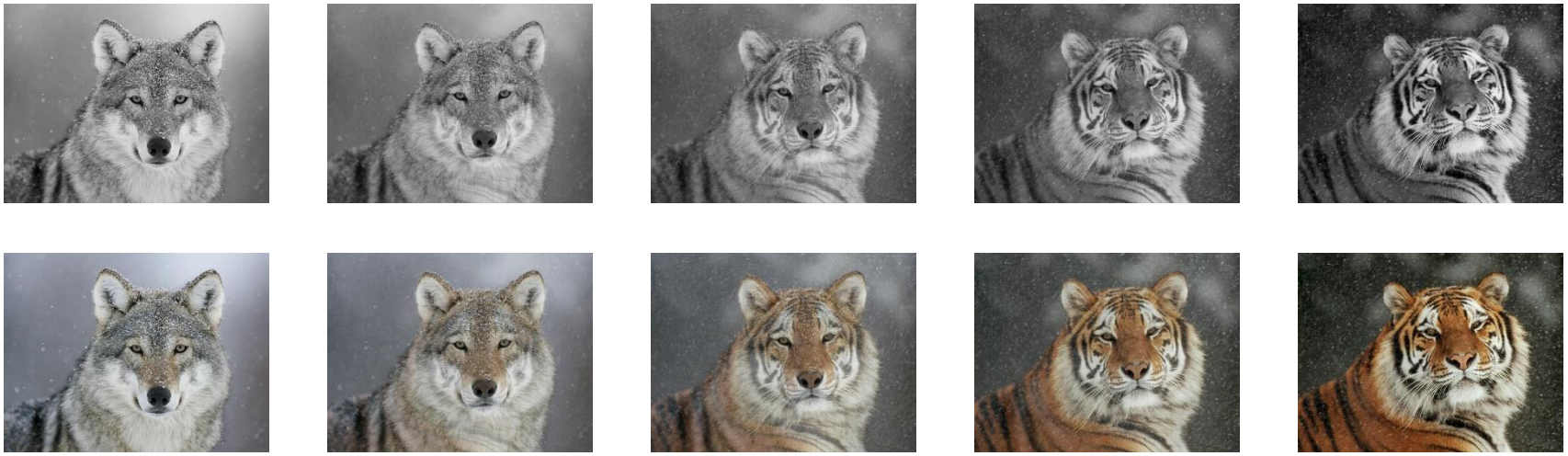
ECE 364
Software Engineering Tools Laboratory

Project Description

Image Morphing



Motivation



See Videos!

Check the following links for a full description of the project:

https://www.learnopencv.com/face-morph-using-opencv-cpp-python/
http://andrew.gibiansky.com/blog/image-processing/image-morphing/

Today's Lecture

- Project Description and Tasks

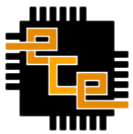
  What is This Project About?

- Theoretical & Programming Foundations

  The Background Information that you will need.

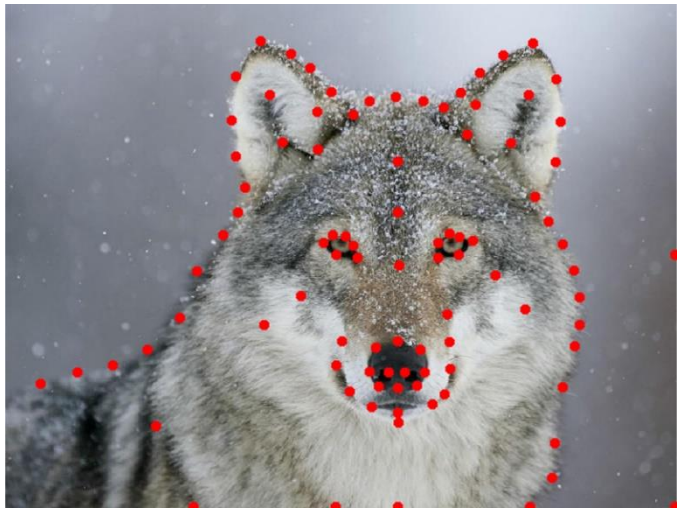
- Working Steps and Milestones

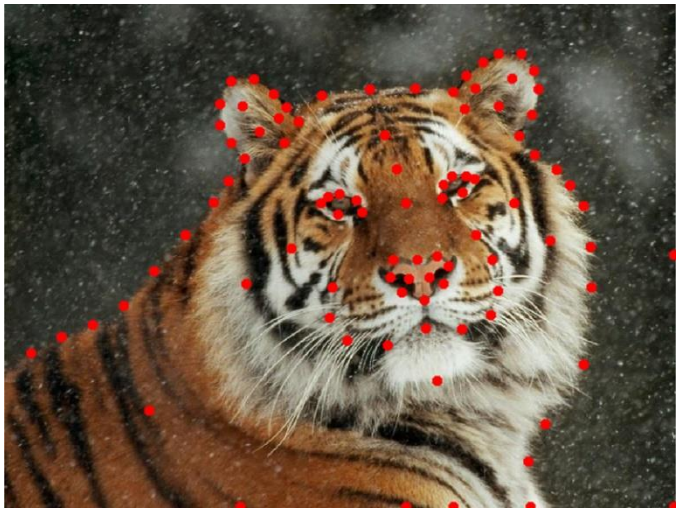
  What are you expected to submit?



Project Description

Input: 1) Two images, Left & Right.
2) Point Correspondences (Point Pairs).





Pairs: 1) Same Number of Points in Both Images.
2) Each Point in Left has a Specific Point in Right.

Project Description

Output:

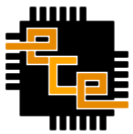
- A single image (blended & morphed.)

- Note:
  - Base Requirement is for Grayscale Images and Video.
  - Color Examples are for Illustration.

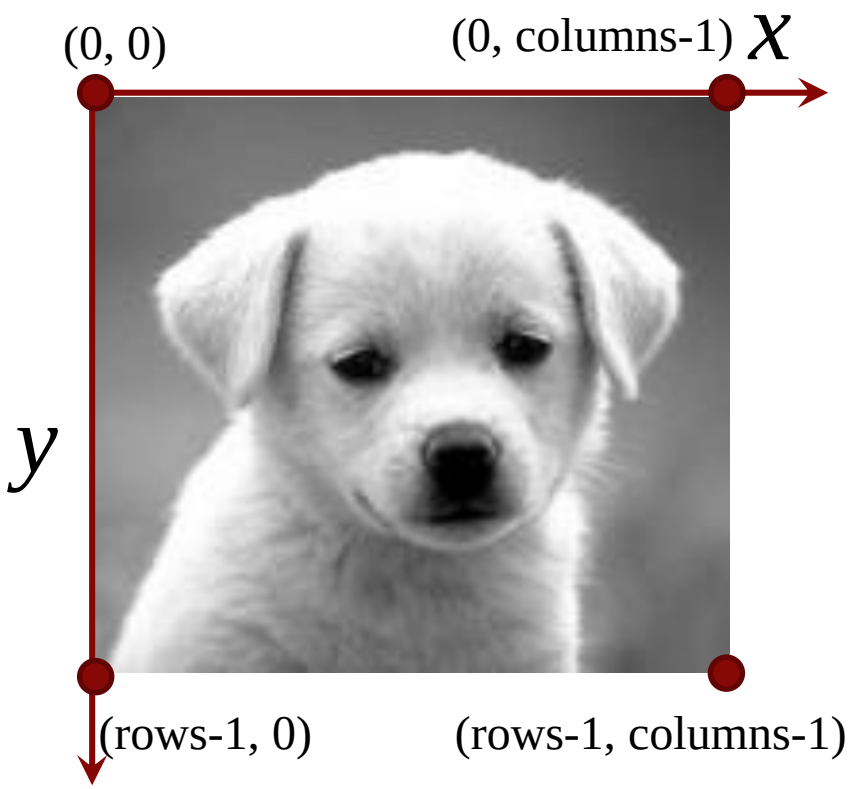
Foundations: Libraries

- We will be using several libraries that are already present in your Anaconda installation
  - `numpy` ⟸ Main Library
  - `scipy`
  - `imageio`
- These libraries are the standard ones for Python Scientific Computing.
- You need to spend time with these libraries.

Aside from the Python Standard Library and these ones, do not use any other library before you request permission.



Foundations: Image Data



Note that:
x = column, y = row

Depending on the library function we can access data:

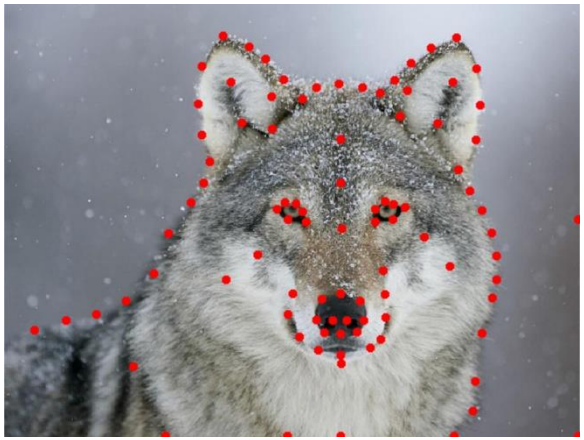
- *image(row, column)*
- *f(x, y)*

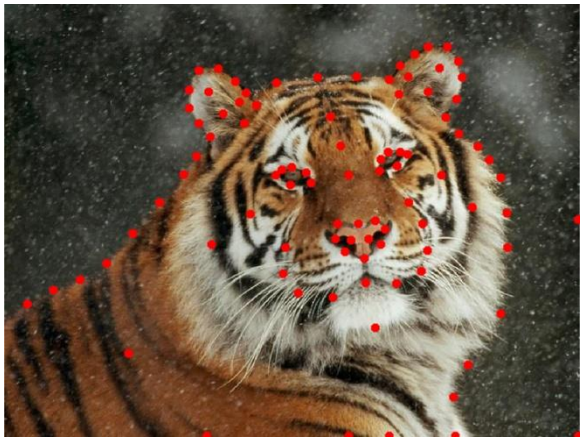
Using *(x, y)* we can access:

- *f(1.653, 45.95)*

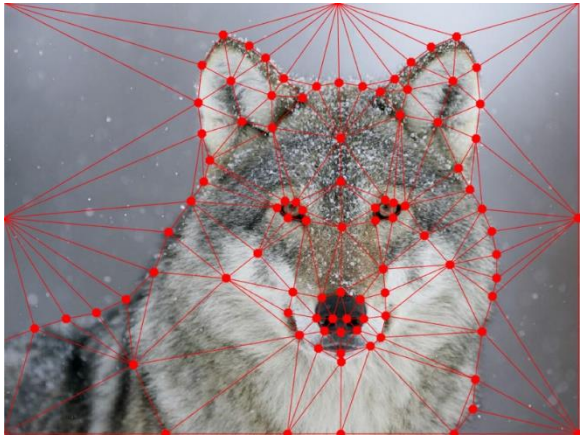
This is called 2-D interpolation

Foundations: Delaunay Triangulation

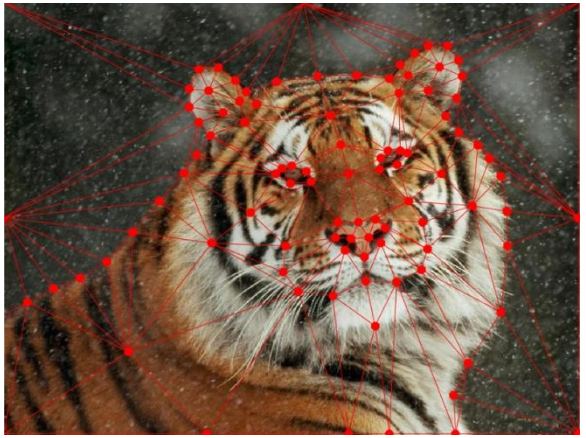




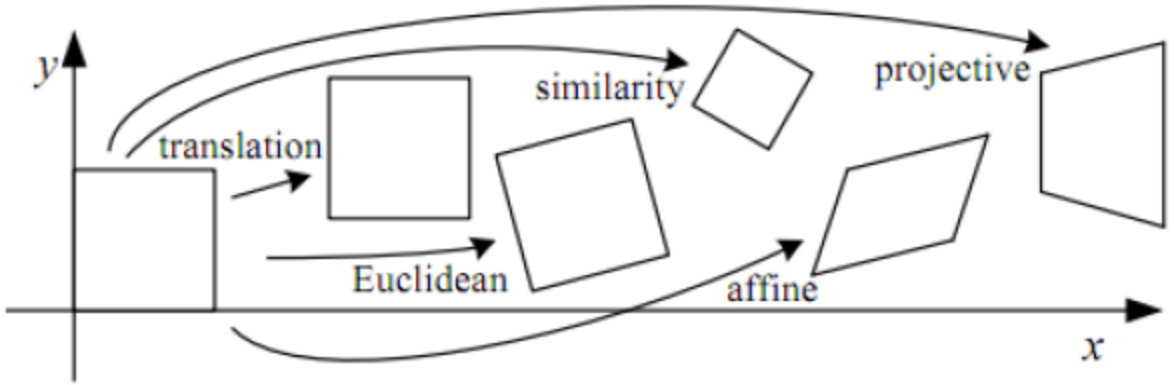
Apply Delaunay

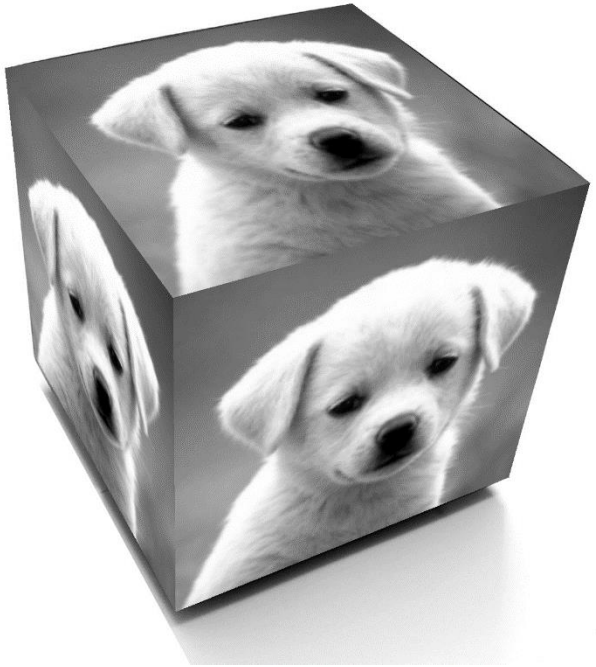


Triangles from Correspondences



Foundations: Projective Transformation

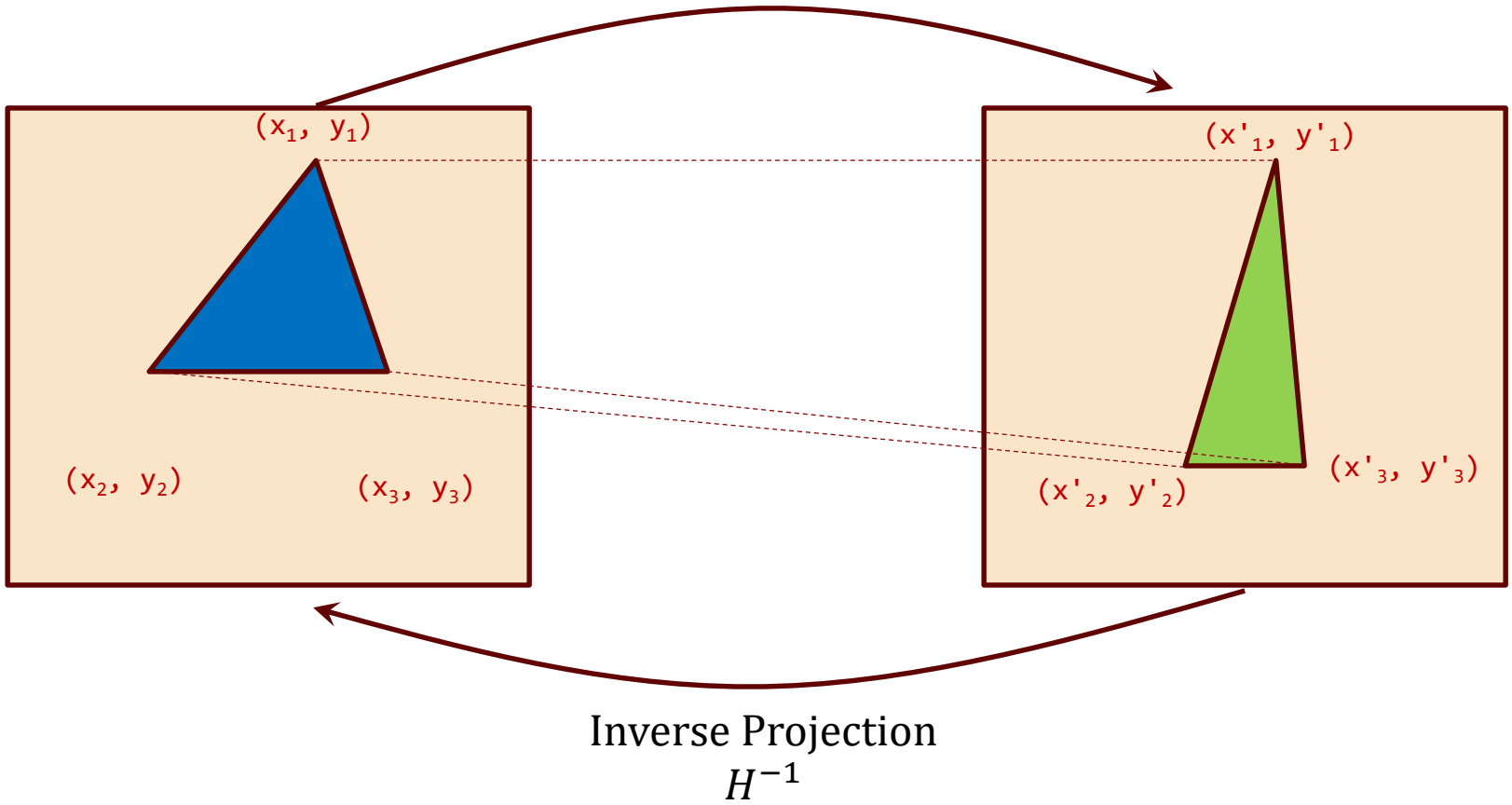




Foundations: Affine Transformation

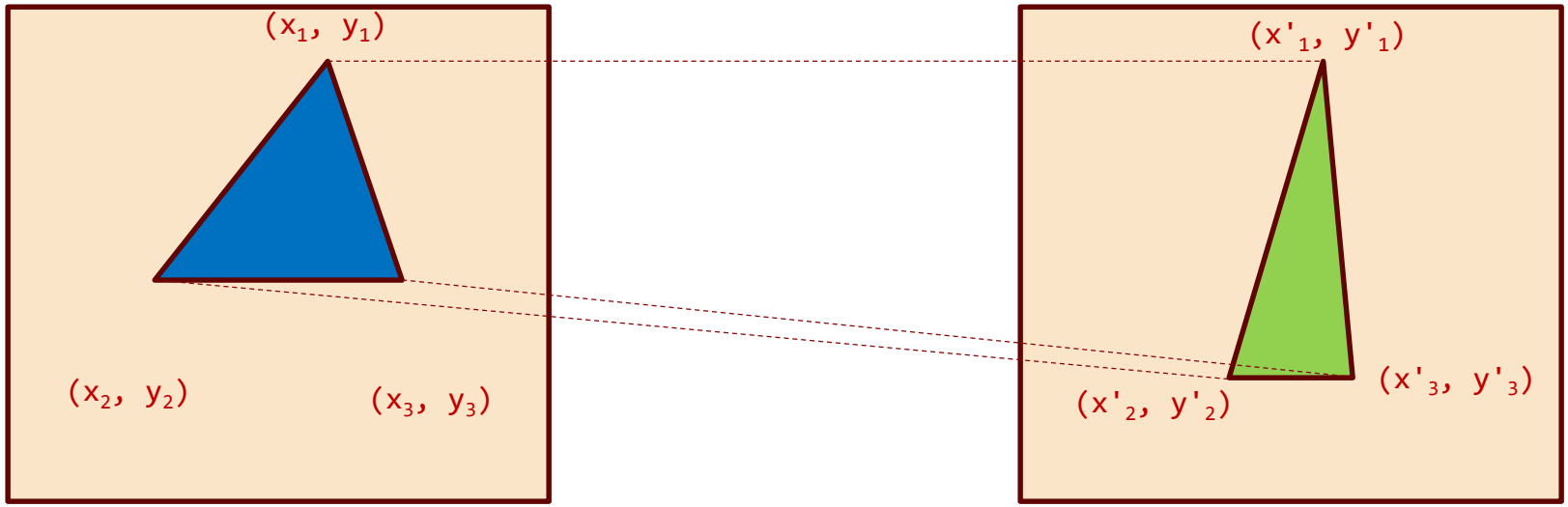
Affine Projection Matrix

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{bmatrix}$$



Foundations: Obtaining Matrix

- Find a matrix for every triangle.

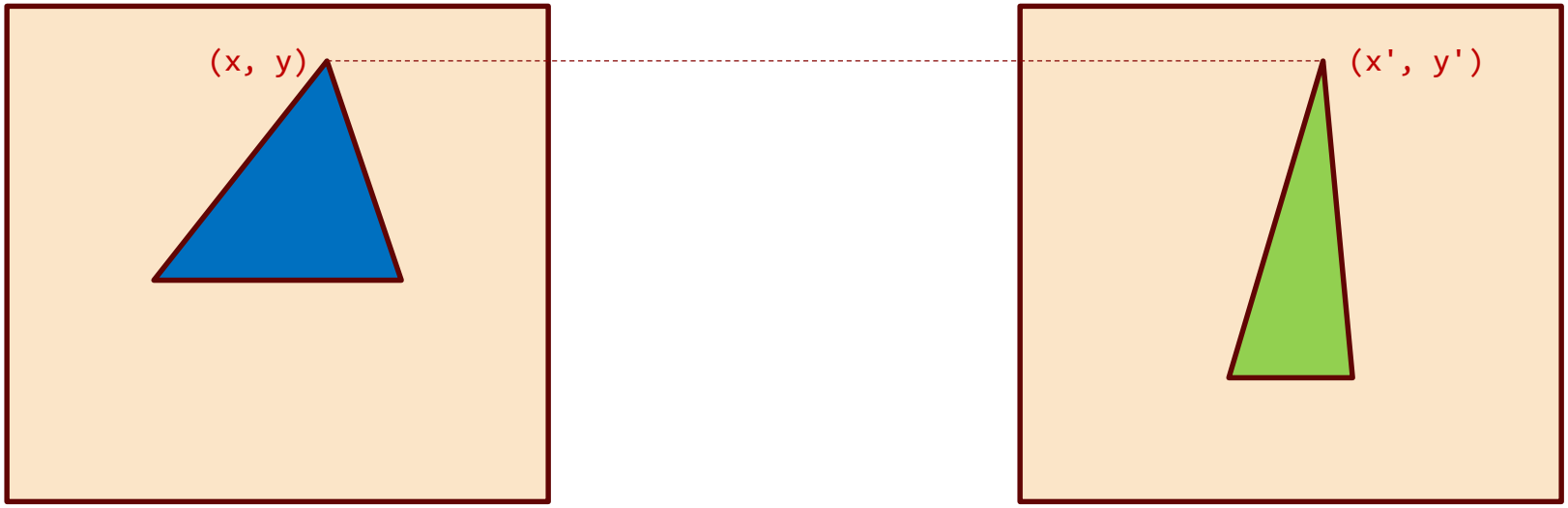


$$\begin{bmatrix} x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix} \quad \begin{bmatrix} x'_n \\ y'_n \end{bmatrix}$$

- From each correspondence pair, we will create two matrices.
- Stack all equations to get: $[6 \times 6].[6 \times 1] = [6 \times 1]$
- This gives the linear system $Ah = b$, where $h = [h_{11} \ldots h_{23}]^T$

Use: `h = numpy.linalg.solve(A, b)`

Foundations: Applying Projection



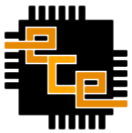
- Use homogeneous Coordinates:

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

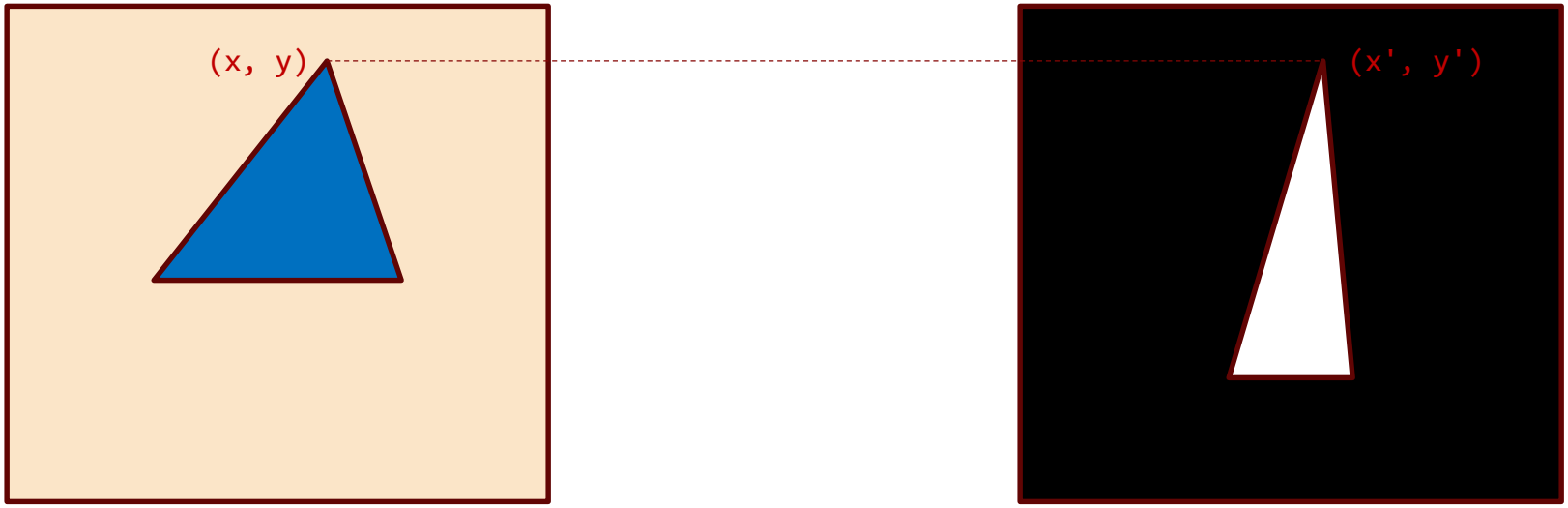
- Example:

$$\begin{bmatrix} 0.5 & 1.0 & 1.0 \\ 1.0 & 0.5 & 1.0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 6.0 \\ 6.5 \\ 1 \end{bmatrix}$$

- We will have an assignment problem!!!
  (We can read from fractional indices, but not assign!)



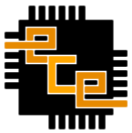
Foundations: Applying Projection



- Instead of: where do I assign *(x, y)* to?
  We do: where do get *(x', y')* from?

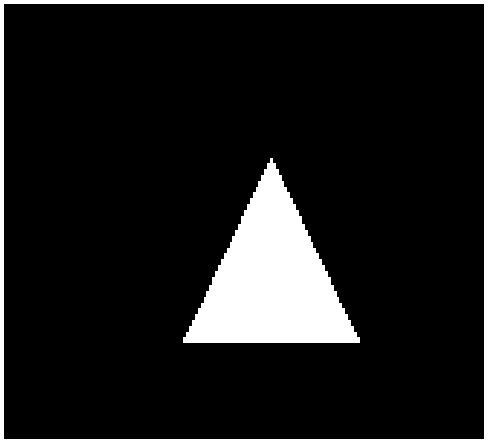
$$H^{-1}\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Fractions are OK this way! Use Bilinear Interpolation.

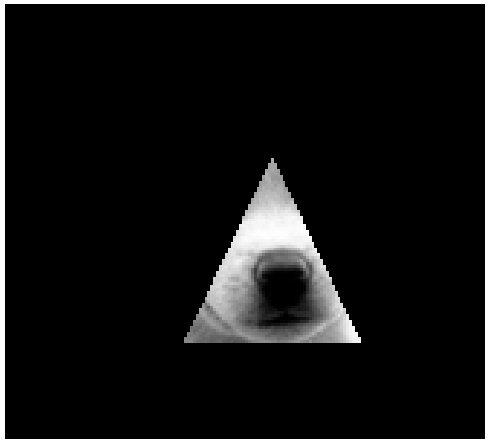


Foundations: Masking & ROI

Image



ROI Mask



Points in ROI

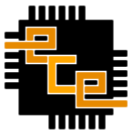
Foundations: Applying Projection

- Summary of Affine Transformation:

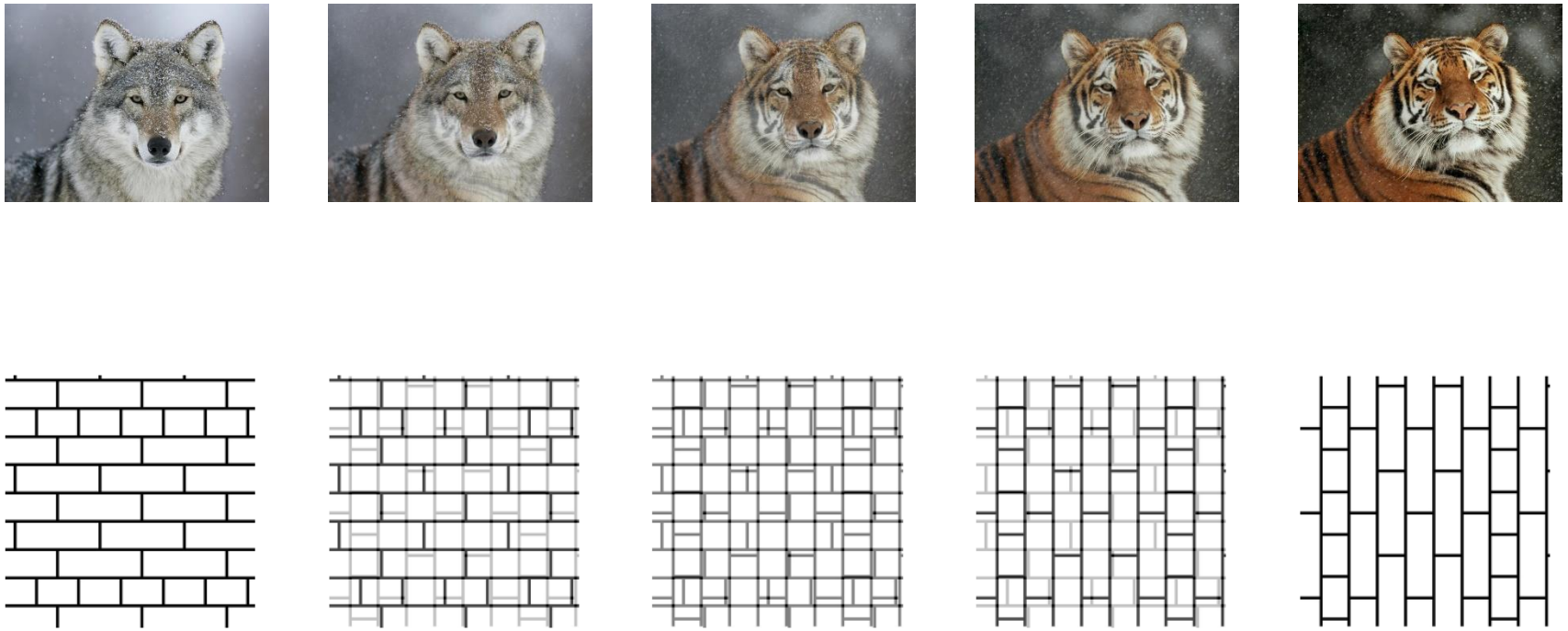
  1- Find forward and backward projection matrices.

  2- Find “Empty” Target points, using Point-in-Polygon test, or ImageDraw().polygon.

  3- Perform inverse projection to obtain target point values from source points.



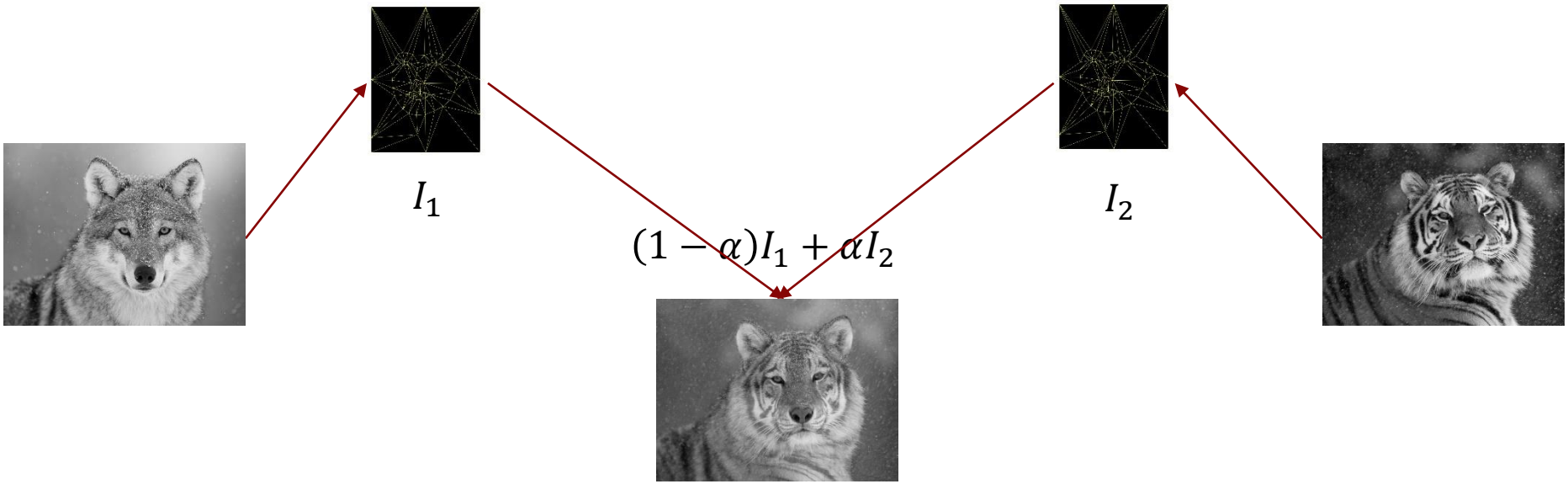
Foundations: Alpha Blending



(a) Image 1. (b) $\alpha = 0.25$. (c) $\alpha = 0.50$. (d) $\alpha = 0.75$. (e) Image 2.

Foundations: Alpha Blending

For a given α , generate I_1 and I_2, then Blend.



Requirements & Tasks

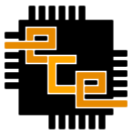
Initial Resources:

- This Lecture.
- Project Document.
- Test Data and Test Script.

Tasks:

- Understand the Foundations and Requirements.
  - Please do NOT post questions about how function X in library Y works.
- Unlike Labs: **The Design Published is NOT sufficient**.
  - Classes Given are the Public API ONLY.
  - Create a Skeleton Design from Input to Output.
  - Work you Way Through to Fill the Gaps.

You ONLY need to submit a code file. (More for Extra Credits.)



Extra Credit Tasks

You can choose one or more of these tasks.

- Morphing Color Image
- Generating a Morph Video
- Creating Personal Morph Video
- Your code should follow some Performance Metrics.

NOTE: Plan ahead whether you want to do these or not.

ee364g07	1.04		ee364b17	4.66		ee364e06	6.69		ee364b09	16.25		ee364b25	126.86		ee364e04	229.14
ee364b03	1.91		ee364g02	4.66		ee364c18	6.93		ee364g29	35.17		ee364b21	128.53		ee364e01	250.64
ee364c21	3.11		ee364c07	4.69		ee364e15	6.96		ee364d26	38.24		ee364a16	128.80		ee364g18	251.13
ee364g10	3.49		ee364c03	4.72		ee364d10	6.96		ee364b12	38.45		ee364c24	130.55		ee364b27	296.77
ee364d09	3.60		ee364g13	5.00		ee364b24	7.43		ee364d25	58.50		ee364f02	136.27		ee364b22	297.62
ee364g14	3.79		ee364e27	5.04		ee364g15	9.79		ee364d28	74.70		ee364g08	137.02		ee364g05	305.32
ee364e11	3.80		ee364e02	5.06		ee364f07	10.28		ee364e14	86.10		ee364a08	137.03		ee364b08	350.83
ee364e10	4.12		ee364e05	5.16		ee364d14	10.35		ee364g01	88.13		ee364a06	137.48		ee364c25	354.78
ee364g21	4.14		ee364g09	5.35		ee364f05	10.36		ee364c20	89.21		ee364f25	147.91		ee364f01	442.53
ee364g03	4.32		ee364g26	5.43		ee364b04	11.65		ee364e08	89.64		ee364d17	148.73		ee364a11	507.37
ee364d06	4.37		ee364c26	5.51		ee364e30	11.86		ee364a03	92.16		ee364e25	151.49		ee364d08	580.96
ee364f19	4.43		ee364d19	5.52		ee364g17	12.42		ee364f22	93.29		ee364d22	152.00		ee364c15	1260.12
ee364c06	4.62		ee364g25	5.58		ee364g06	12.45		ee364b05	93.54		ee364d02	158.34			
ee364b16	4.63		ee364e22	5.69		ee364g20	13.00		ee364c16	103.83		ee364e16	162.07			
ee364c09	4.64		ee364d20	5.91		ee364b19	13.24		ee364f18	124.62		ee364e24	206.20			

Performance Results from Spring ‘18

Next Phase: GUI