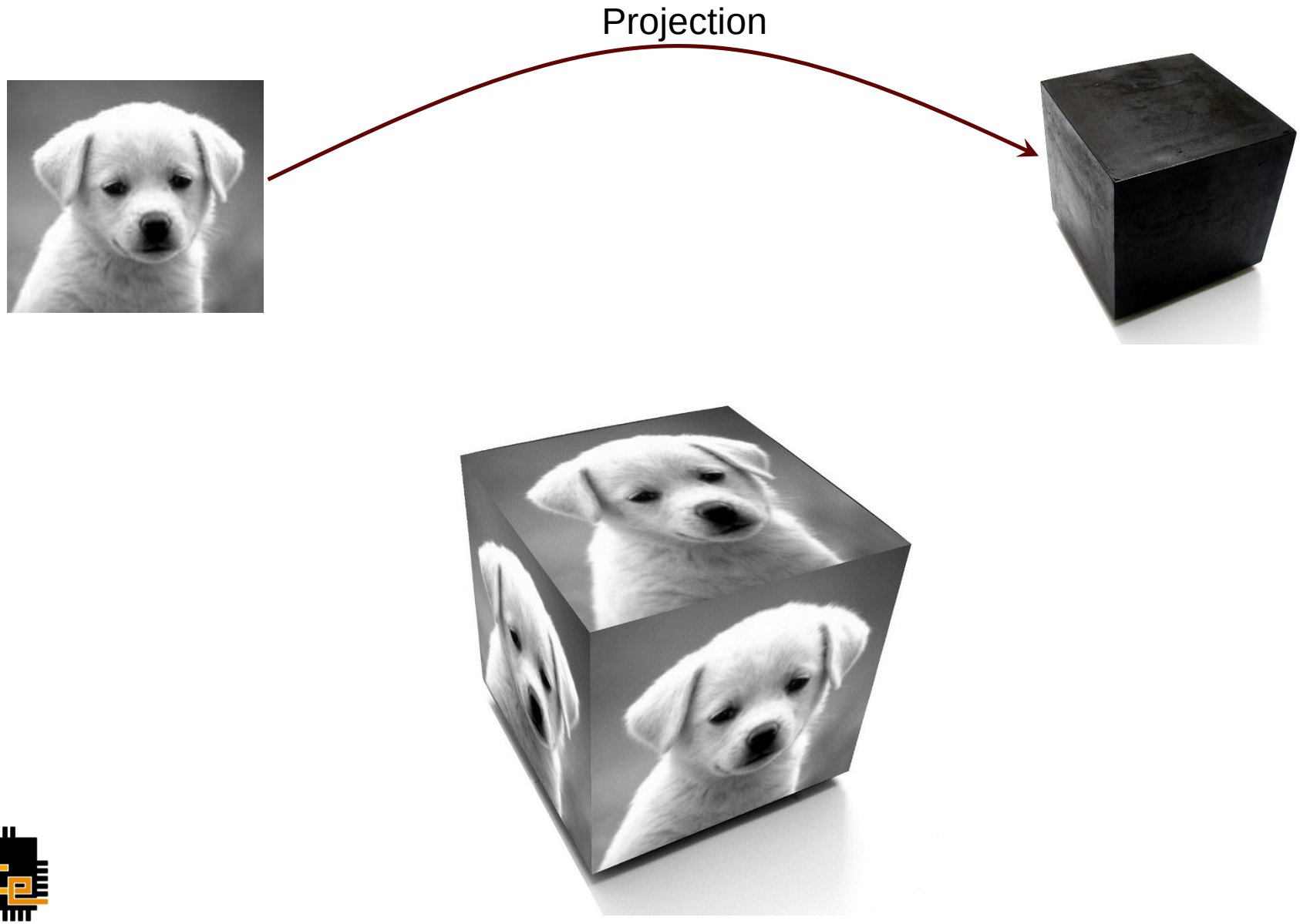
ECE 364
Software Engineering Tools Laboratory

Project Description

Homography

Motivation



Setup

You need to update your account with 3 modules:

- `numpy`
- `scipy`
- `Pillow`

Demo!

Project Stages

Three “conceptual” parts:

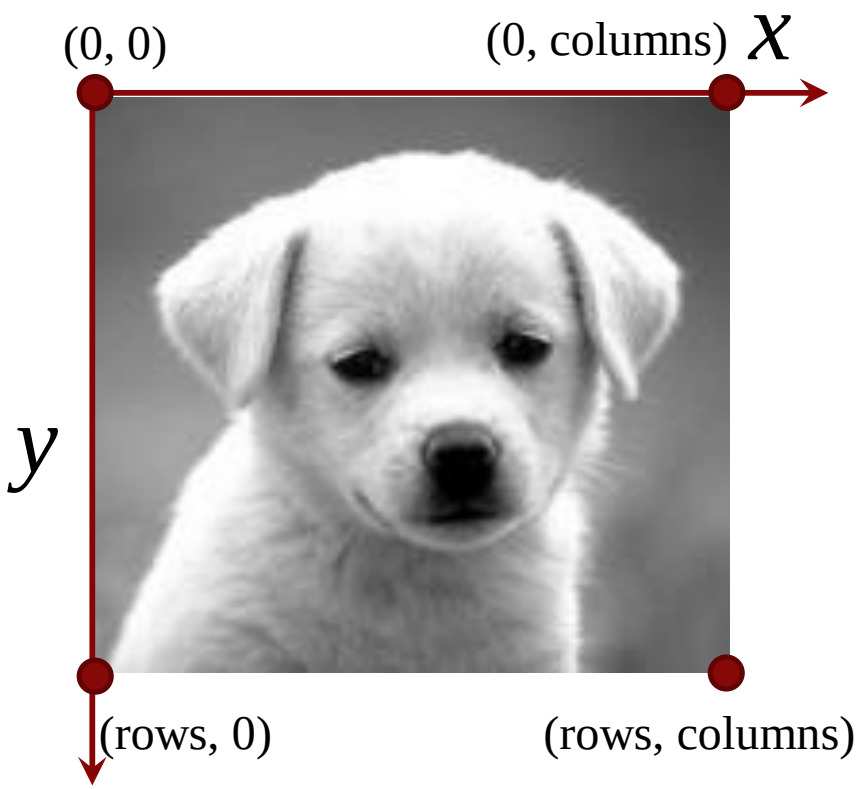
1. Apply a given homography.
2. Compute and apply a homography.
3. Apply transformation effects.

Start with grayscale images. Extend to color images.

Extra credit section: Composite Effects

You will use: classes, inheritance, Enum

Accessing Image Data



Note that:
x = column, y = row

Depending on the library function we can access data:

- *image(row, column)*
- *f(x, y)*
  Data outside limits is undefined

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

- *f(1.653, 45.95)*

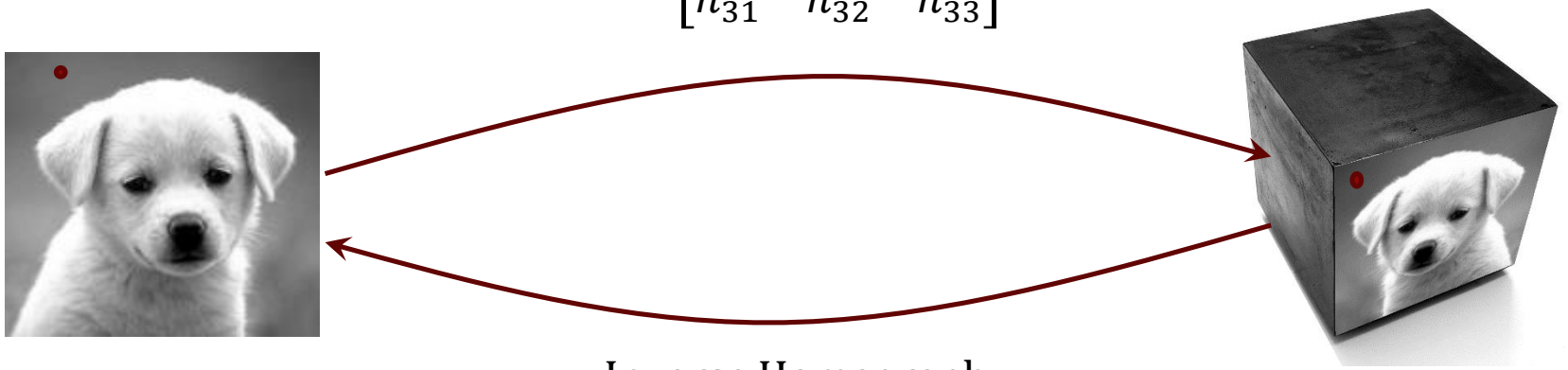
This is called 2-D interpolation:

- Create your own function
- `scipy.interpolate.iterp2d`
- `scipy.interpolate.RectBivariateSpline`

Applying Homography

Homography Matrix

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$



Inverse Homography

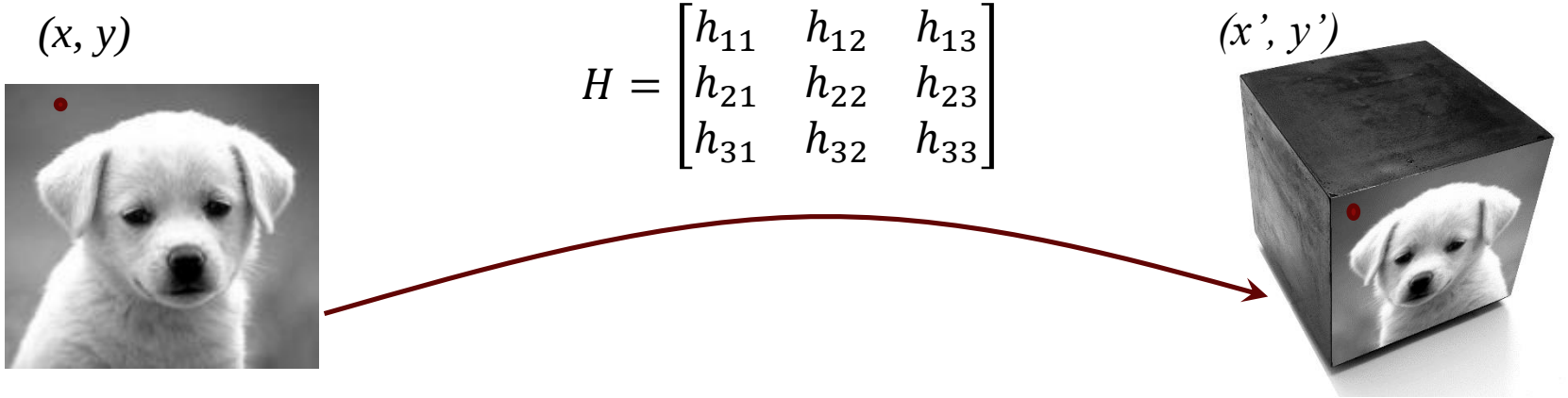
H^{-1}

How do we perform a homography operation?

Applying Homography

Homography Matrix

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

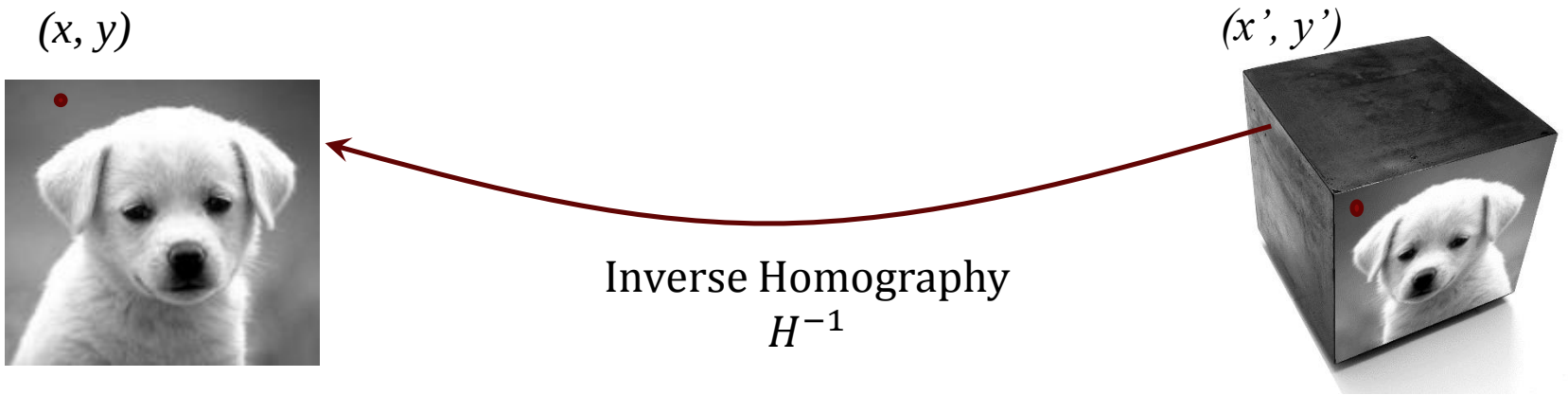


- Use homogeneous Coordinates:

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} / c = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

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

Applying Homography



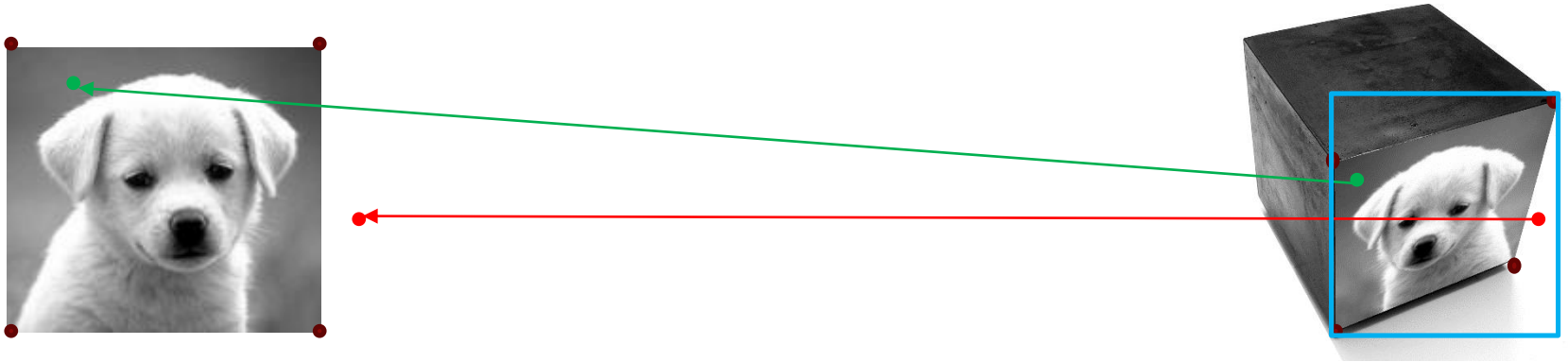
- Instead of: where do I assign *(x, y)* to?
  We do: where do get *(x', y')* from?

$$H^{-1}\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} / c = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Fractions are OK this way!

But we have another issue: Iteration Range!!

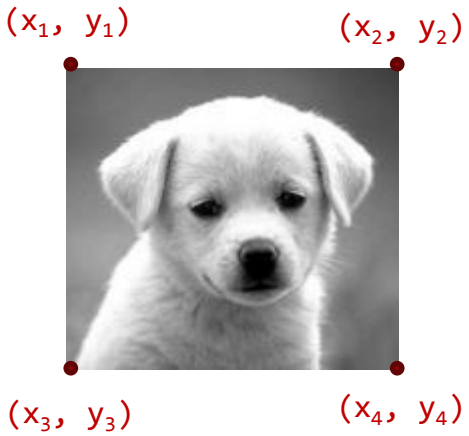
Applying Homography

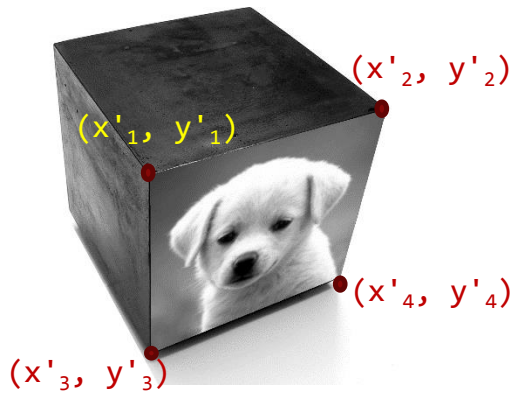


- Homography has been generated from point correspondences.
- Start by getting the bounding box, and iterating over all elements.
- Points inside target range will back project into the source image.
- Points outside target range will back project outside the source image!

But we have another issue: Iteration Range!!

Computing Homography





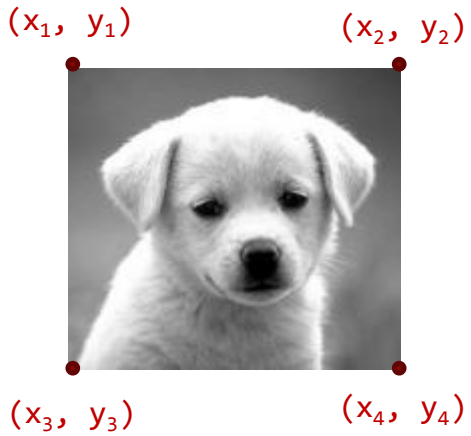
$$\begin{bmatrix} x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n \end{bmatrix} \begin{bmatrix} x'_n \\ y'_n \end{bmatrix}$$

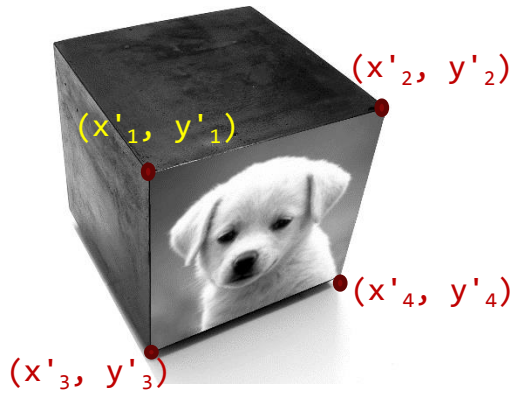
- From each correspondence pair, we will create two matrices.
- Stack all equations to get: $[8 \times 8].[8 \times 1] = [8 \times 1]$
- This gives the linear system $Ah = b$, where $h = [h_{11} \dots h_{32}]^T$

$$(h_{33} = 1)$$

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

Apply Transformation Effects

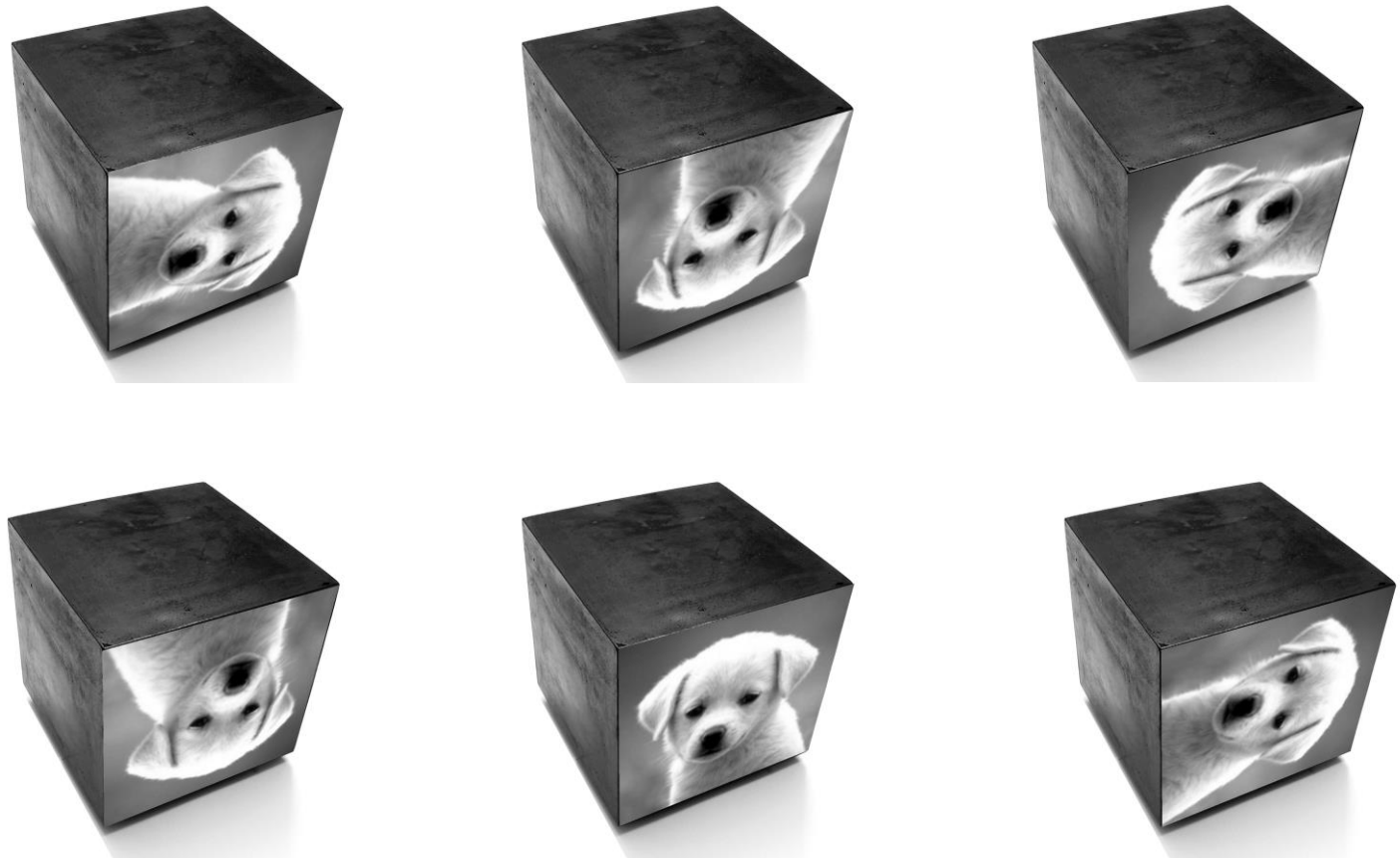




Manipulating the pair assignment can give us the following options:

No Effect
90° Rotation
180° Rotation
270° Rotation
Horizontal Flip
Vertical Flip
Transpose

Apply Transformation Effects



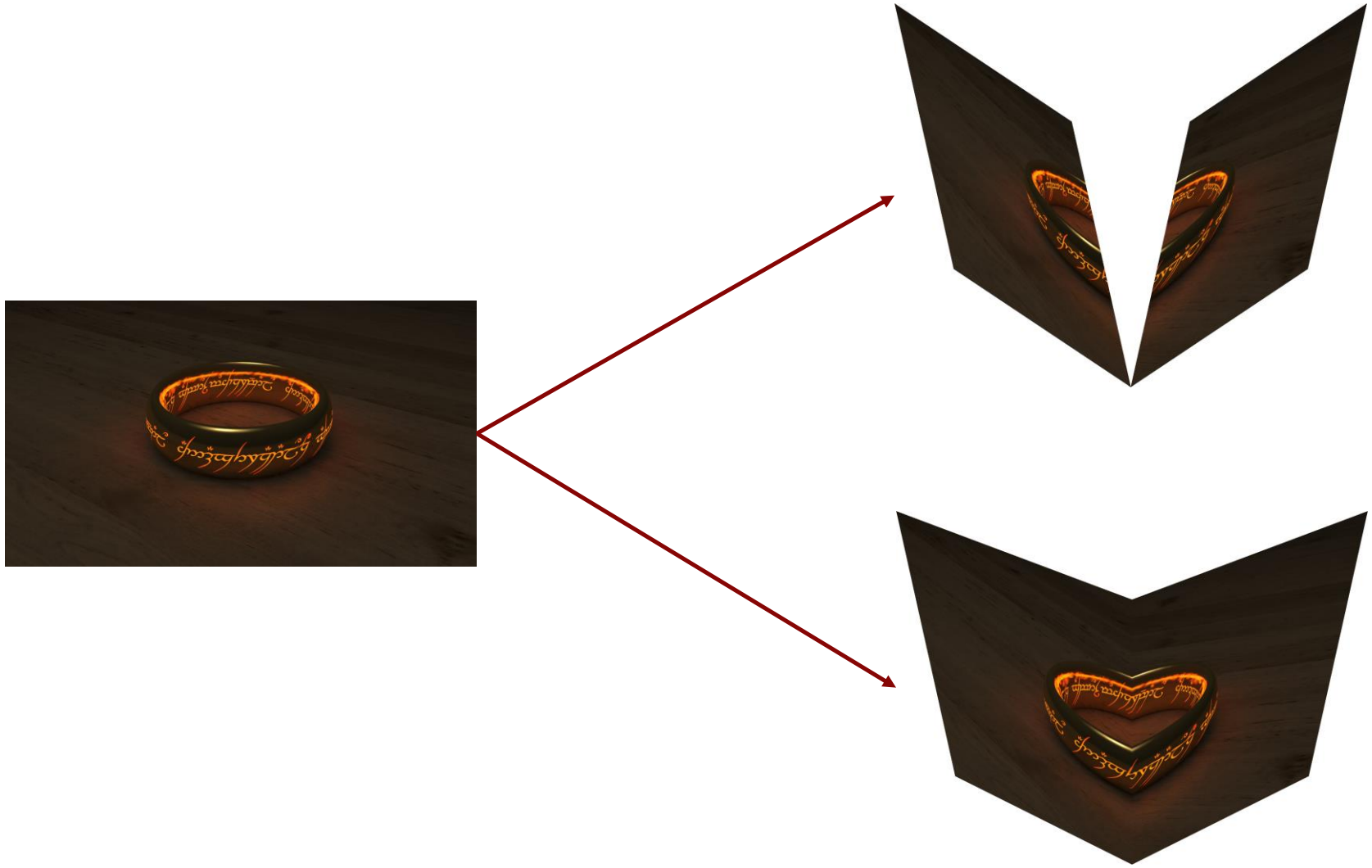
Apply Transformation Effects

We will store the effect in an `Enum` (new feature in 3.4)

```
nothing
rotate90
rotate180
rotate270
flipHorizontally
flipVertically
transponse
```

Homography on Color

Extra Credit



Extra Credit