Image Warping

(Geometric Transforms)

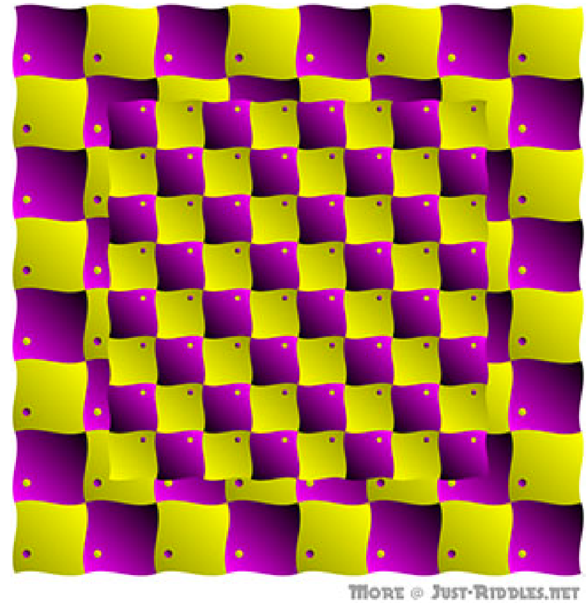
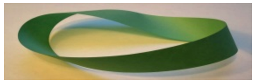


Image Processing



Brent M. Dingle, Ph.D. 2015
Game Design and Development Program
Mathematics, Statistics and Computer Science
University of Wisconsin - Stout

Material in this presentation is largely based on/derived from
presentation(s) and book: The Digital Image by Dr. Donald House at Texas A&M University

Lecture Objectives

- Previously
  - Image Interpolation

- Today
  - Image Warping (Geometric Transforms)

Outline

- **Warping (Geometric Transforms)**
  - General Image Warps
  - Forward and Inverse Mapping

Apples of All Shapes and Sizes

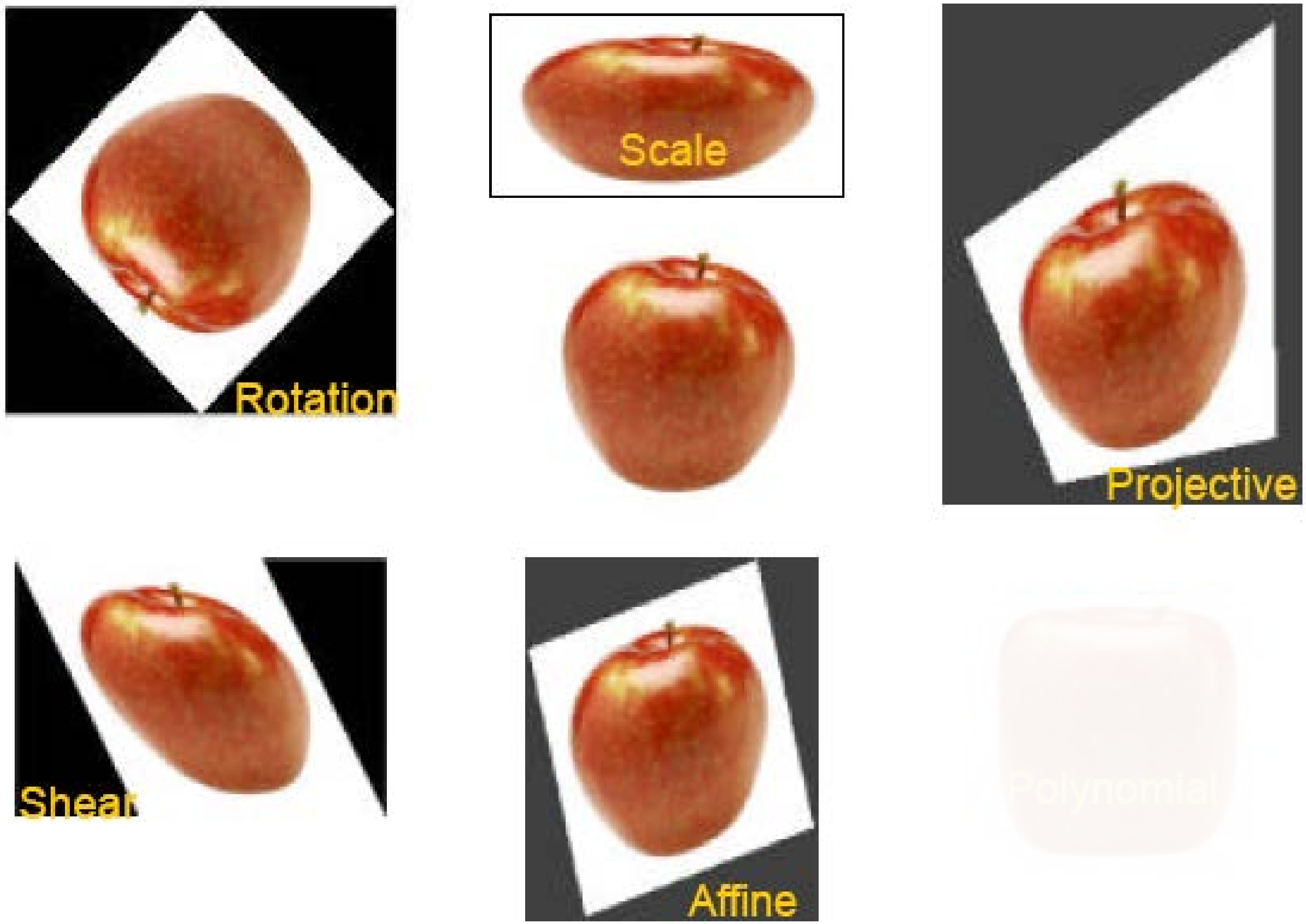
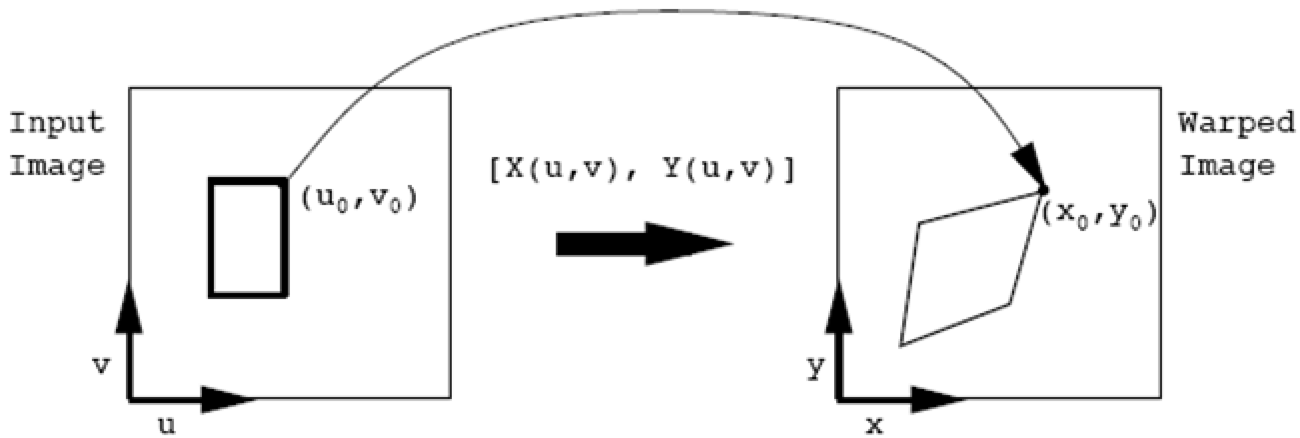


Image Warps

- Mapping and functions



X() and Y() map (u, v) to (x, y)

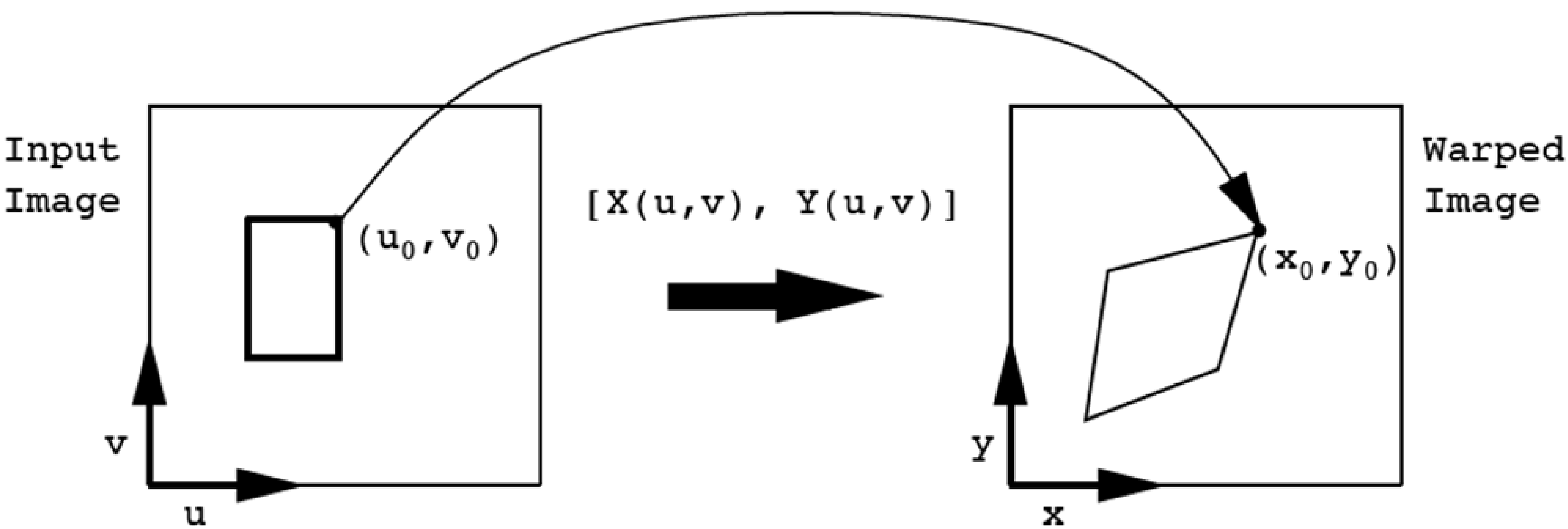
$$x = X(u, v),$$
$$y = Y(u, v),$$

in vector notation: $$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} X(u,v) \\ Y(u,v) \end{bmatrix}$$

Forward Mapping: [X(), Y()]

- If the image was continuous
  and the forward map: [X(), Y()]
  is relatively smooth (no discontinuities)

  - Then for each old image pixel (u, v)
    we can paint new pixel (x, y)
    using the same color



Forward Map

- Digital image are NOT continuous
  - They consist of a finite number of samples = pixels
  - Allowing the following algorithm to perform a mapping

```
for(v = 0; v < in_height; v++)
  for(u = 0; u < in_width; u++)
    Out[round(X(u,v))][round(Y(u,v))] = In[u][v];
```

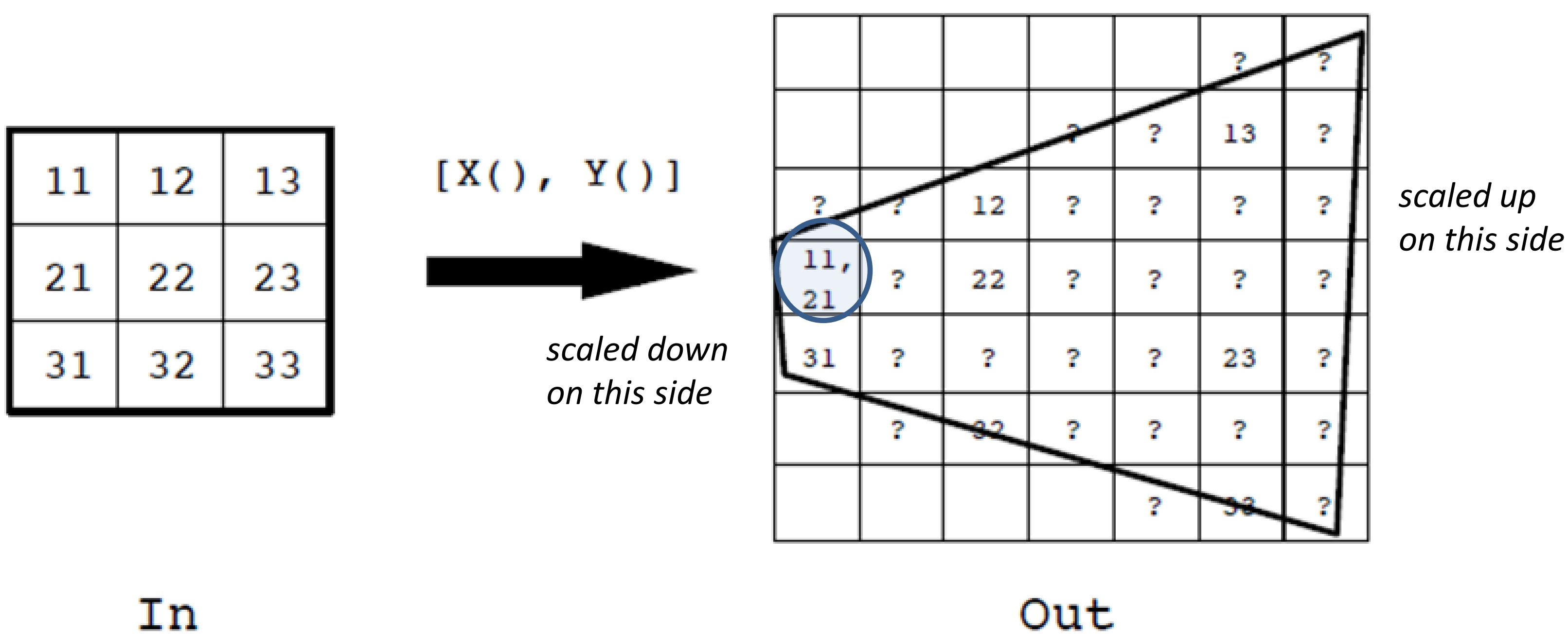
(x , y)

original image's pixel color at (u, v)

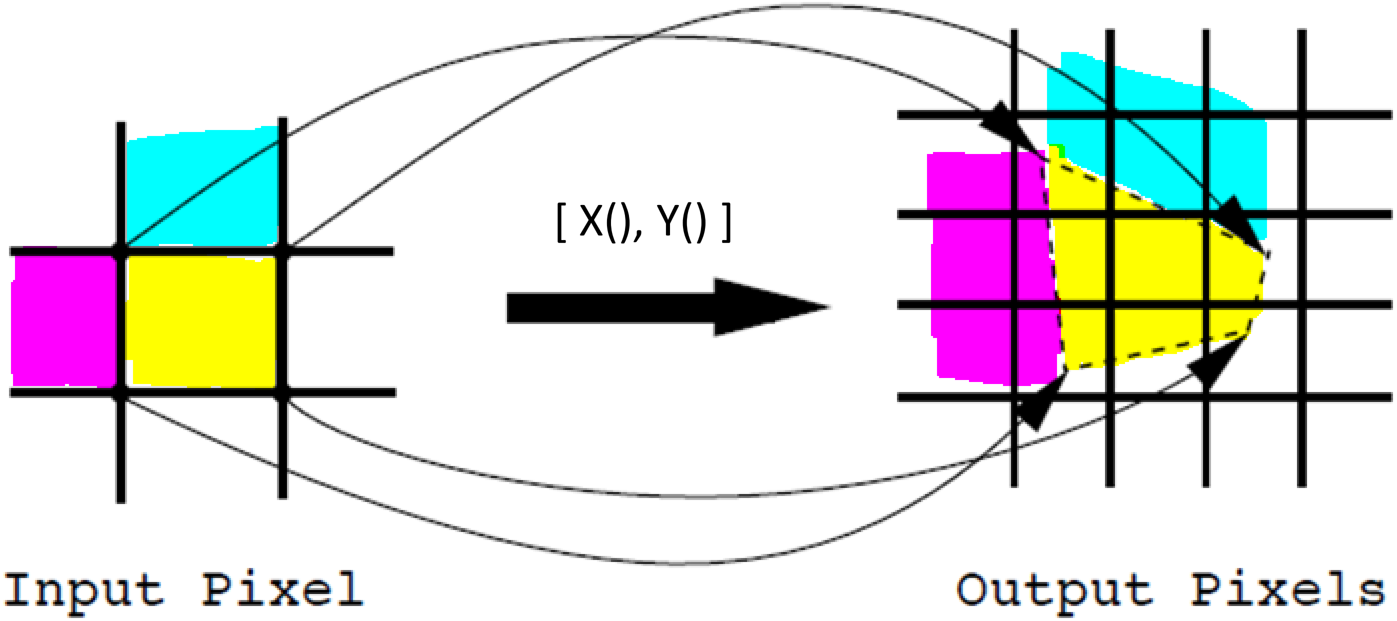
Problem with Forward Map

- With loss of “continuous/infinite” points
  - one-to-on mapping cannot happen
  - forward map leaves holes and folds

? = hole in new image
= fold in new image



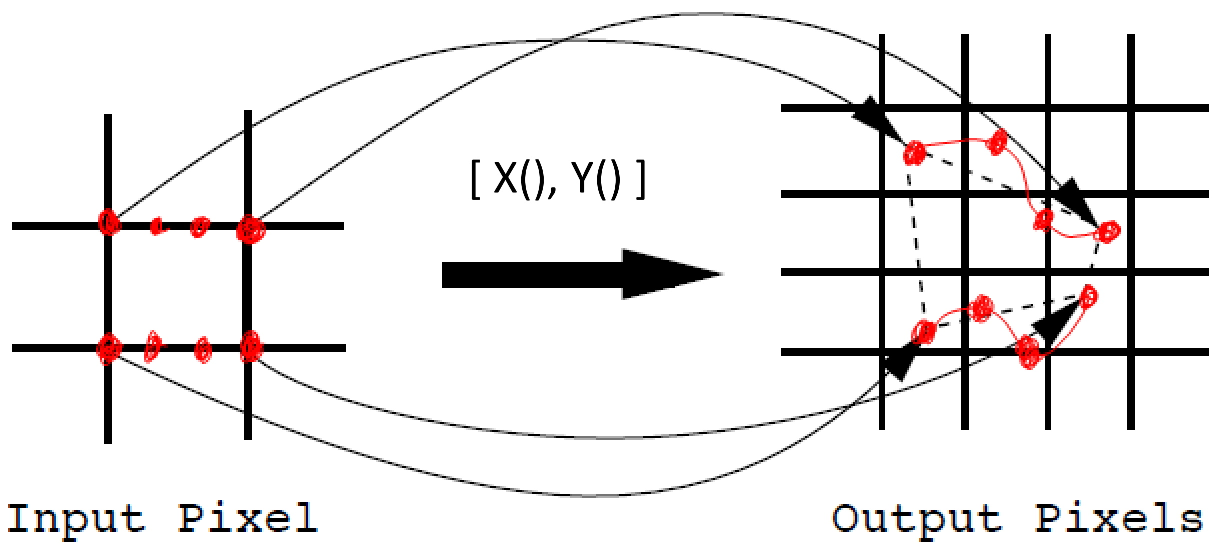
Searching for a Solution



Could try weighting average of some kind
to mix colors, to fill holes and fix folds…

Still Searching…

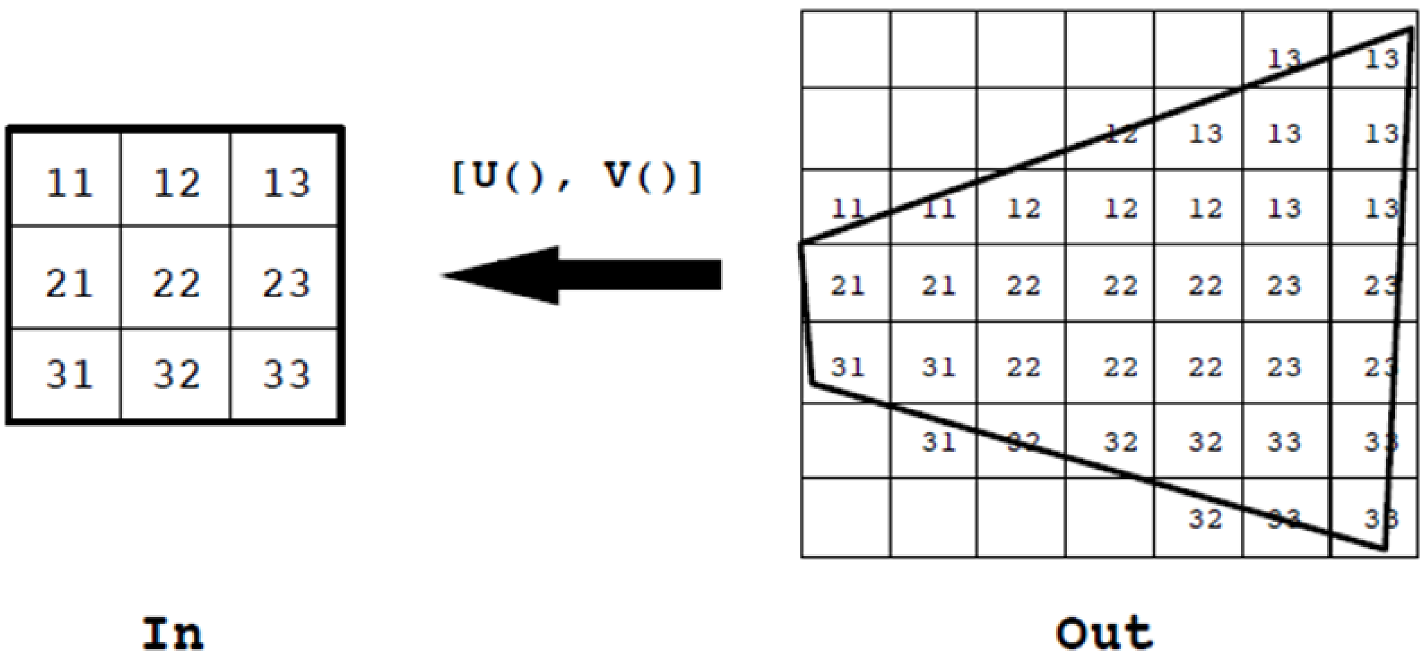
- If map is to curvy lines/edges that averaging becomes more difficult
  - not very "pixel-to-pixel"



Must be an easier way…

Inverse Map Solution

- Invert the problem
  - Look at each output pixel and determine what input pixel(s) map to it
    - instead of sending each input pixel to an output pixel



No holes or folds !!!

Inverse Map: [U(), V()]

- Recall the forward map:

$$x = X(u, v),$$
$$y = Y(u, v),$$

- Invert the mapping functions X() and Y()

$$u = U(x, y),$$
$$v = V(x, y),$$

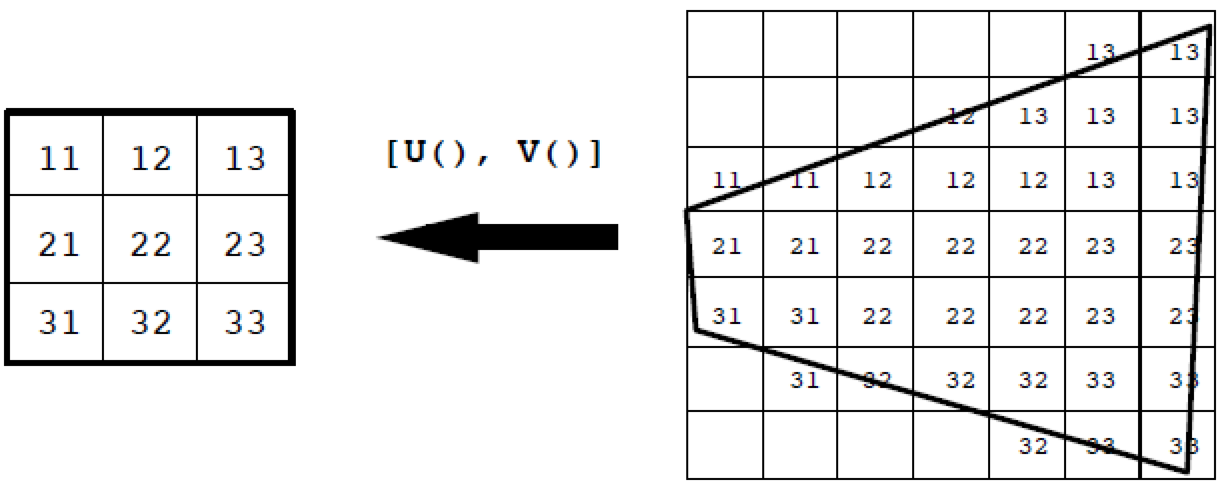
or in matrix form:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} U(x, y) \\ V(x, y) \end{bmatrix}$$

Inverse Mapping Algorithm

```
for(y = 0; y < out_height; y++)
  for(x = 0; x < out_width; x++)
    Out[x][y] = In[round(U(x,y))][round(V(x,y))];
```

(u , v)



In

Out

The inverse map provides a complete covering

Forward vs Inverse Maps

Forward Map:

```
for(v = 0; v < in_height; v++)
  for(u = 0; u < in_width; u++)
    Out[round(X(u,v))][round(Y(u,v))] = In[u][v];
```

(x , y)

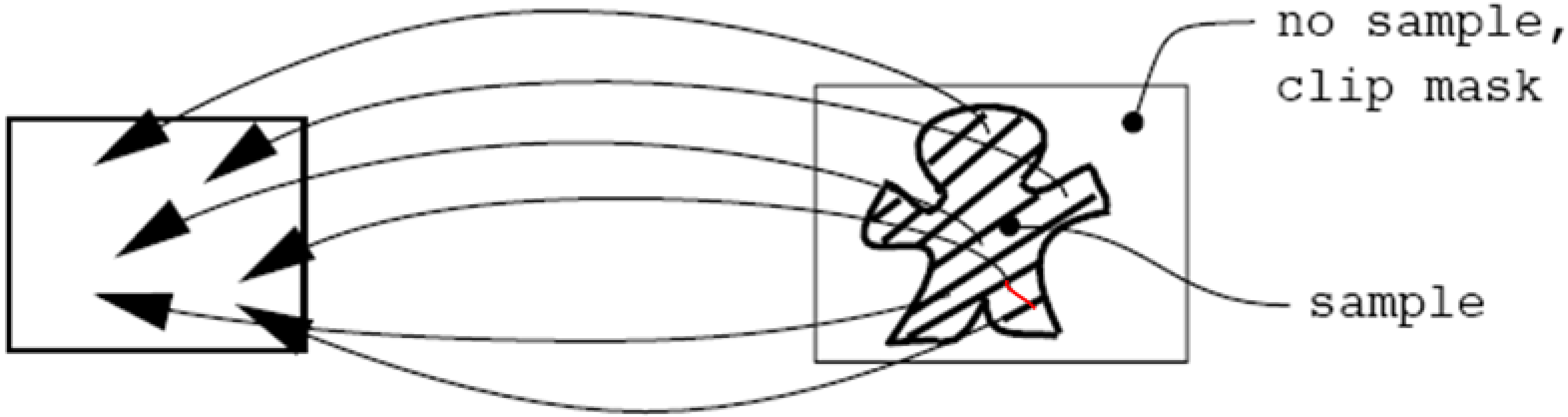
Inverse Map:

```
for(y = 0; y < out_height; y++)
  for(x = 0; x < out_width; x++)
    Out[x][y] = In[round(U(x,y))][round(V(x,y))];
```

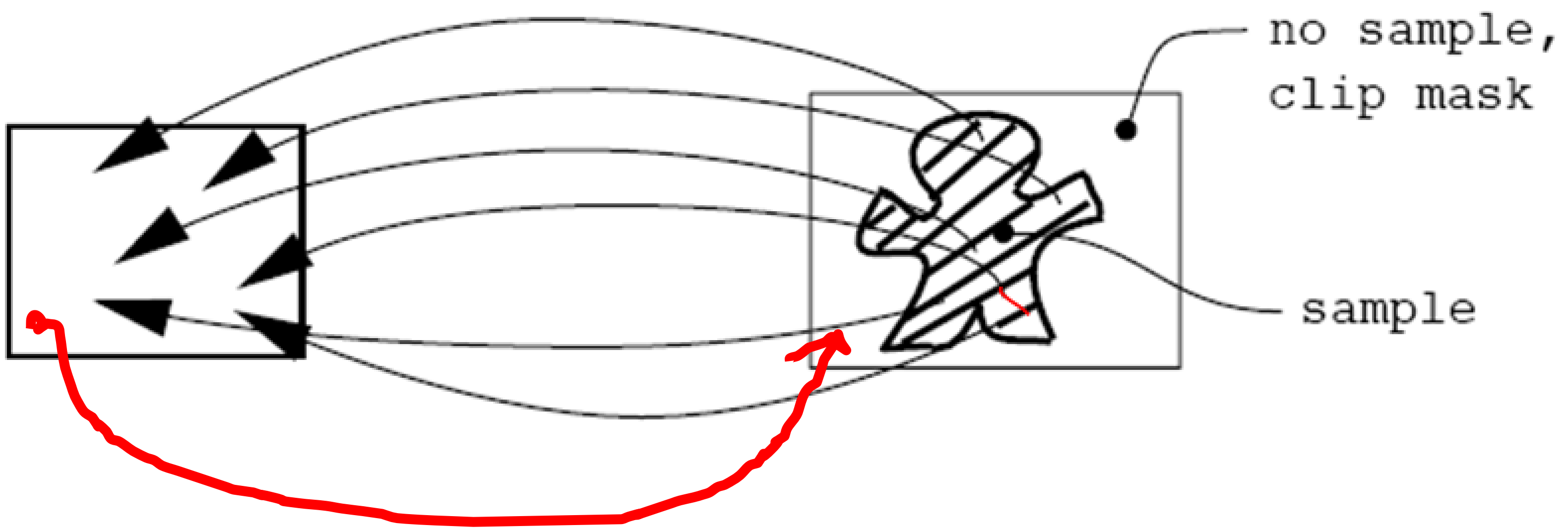
(u , v)

Inverse Supports Clipping Too!

Warping and clipping
all together, same time



Inverse Supports Clipping Too!



A forward map would map this pixel to the new image and require it to be cleared off later

Affine Map

- A geometric transformation that maps points and parallel lines to points and parallel lines

- General form of an affine map:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11}u + a_{12}v + a_{13} \\ a_{21}u + a_{22}v + a_{23} \end{bmatrix}$$

$$X(u,v) = a_{11}u + a_{12}v + a_{13}$$
$$Y(u,v) = a_{21}u + a_{22}v + a_{23}$$

a_{ij} are coefficient constants

Affine Maps: Matrix Form

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11}u + a_{12}v + a_{13} \\ a_{21}u + a_{22}v + a_{23} \end{bmatrix}$$

$$X(u,v) = a_{11}u + a_{12}v + a_{13},$$
$$Y(u,v) = a_{21}u + a_{22}v + a_{23}$$

in matrix form looks like:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

What about Infinity?

- Euclidean/Cartesian space cannot handle points at infinity
  → cannot describe projective space

Image from: http://www.songho.ca/math/homogeneous/homogeneous.html

The railroad tracks become narrower
as they meet the horizon
→ parallel lines intersect at infinity

Projective space allows for this effect

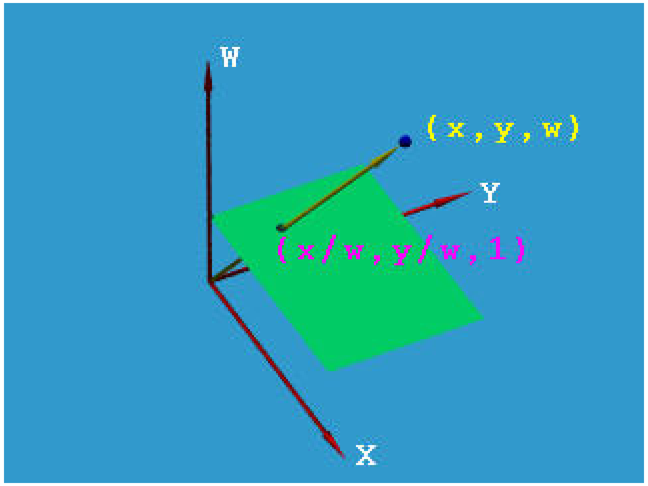
AND

It is easy to transform points
to and from Cartesian space
into and out of Projective space
…

Homogeneous Coordinate System

- Every (Cartesian) point
  has an identical third coordinate

$$\underset{\text{Homogeneous}}{(x, y, w)} \quad \Leftrightarrow \quad \underset{\text{Cartesian}}{\left(\frac{x}{w}, \frac{y}{w}\right)}$$



NOTE: Conversion from Cartesian coordinates to Homogeneous coordinates is unique
however,
Conversion from Homogeneous coordinates to Cartesian is NOT unique

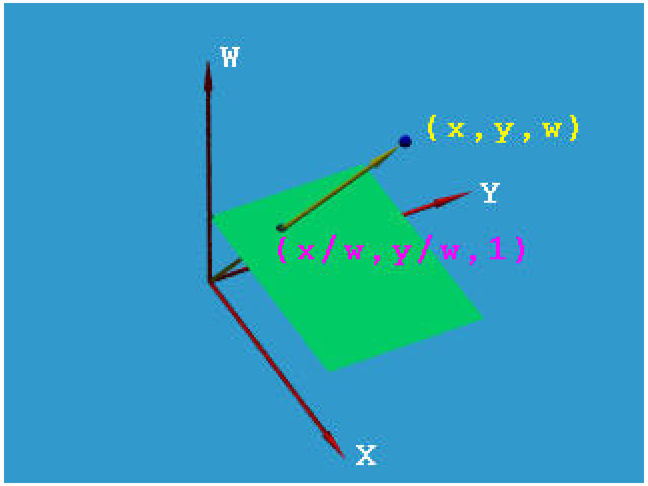
Image from: http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/geometry/homo-coor.html

Homogeneous Coordinate System

- Every (Cartesian) point
  has an identical third coordinate

(u, v, 1) ⇔ (x, y)

Homogeneous Cartesian



Affine
will focus on the plane defined by
w = 1

Image will be (u, v, 1)
with u and v the pixel coordinates

NOTE: Conversion from Cartesian coordinates to Homogeneous coordinates is unique
however,
Conversion from Homogeneous coordinates to Cartesian is NOT unique

Image from: http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/geometry/homo-coor.html

Matrix Translation

- Given homogeneous coordinates (u, v, 1)
  - Find Cartesian coordinates (x, y)

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Explicitly, x then would be calculated as:

$$x = a_{11}u + a_{12}v + a_{13} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

AGAIN:

Conversion from Homogeneous coordinates to Cartesian is NOT unique

coeffs a_{ij} will describe how we want to warp an image,

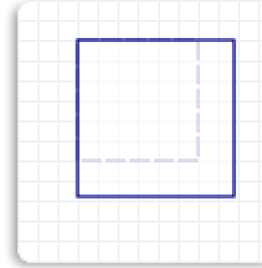
Example: a_{13} is a translation distance for x

Points to Remember

- Homogeneous Coordinates
  - allow affine transformations
    to be easily represented
    by matrix multiplications

- Affine Maps
  - always have an inverse
  - can be represented in matrix form
    (via homogeneous coords)

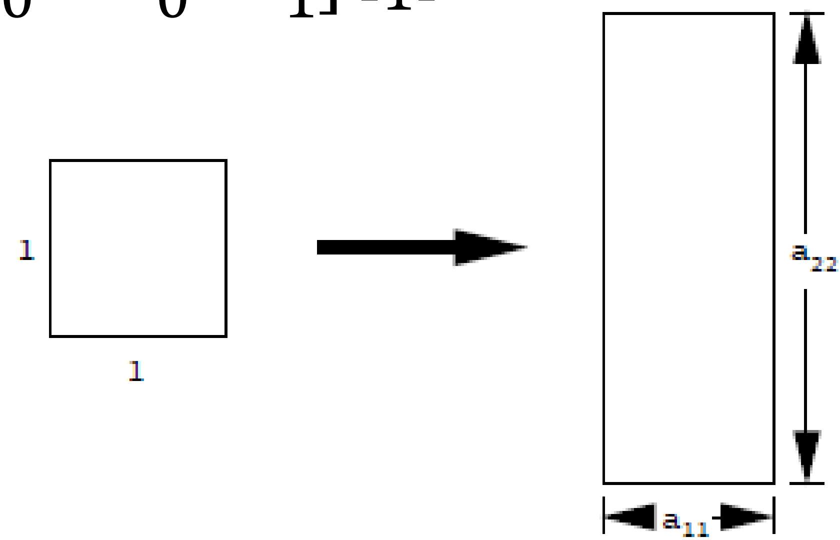
Scale: an affine map transform



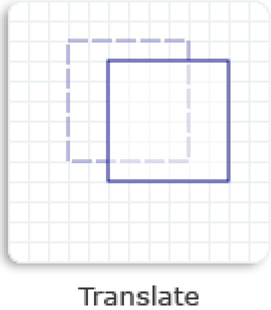
Scale

- scale(x, y) = ($a_{11}u$, $a_{22}v$)

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11}u \\ a_{22}v \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

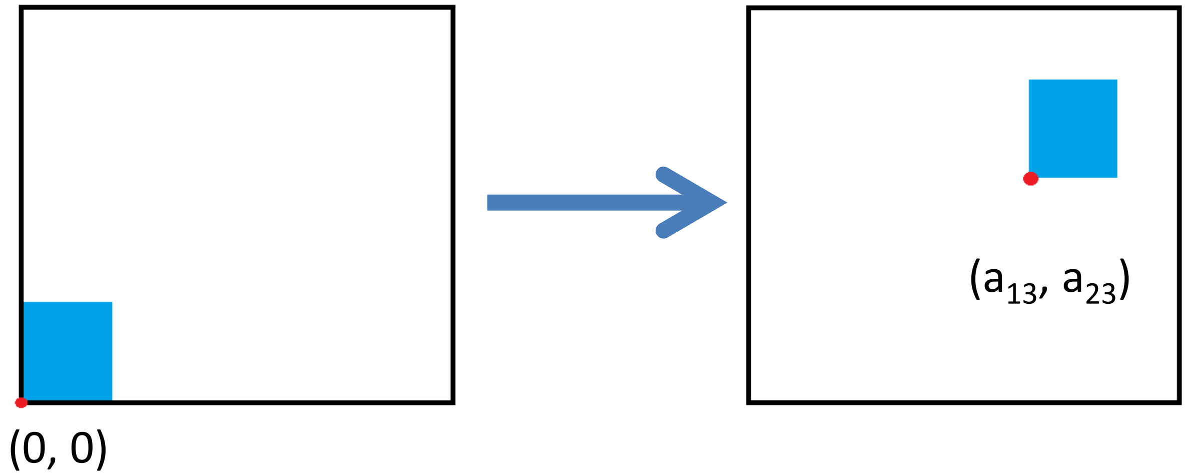


Translate: an affine map transform

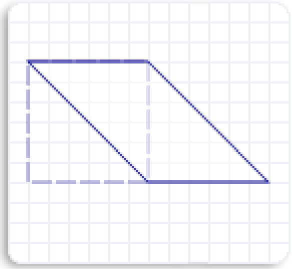


- translate $(x, y) = (u + a_{13},\ v + a_{23})$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u + a_{13} \\ v + a_{23} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a_{13} \\ 0 & 1 & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

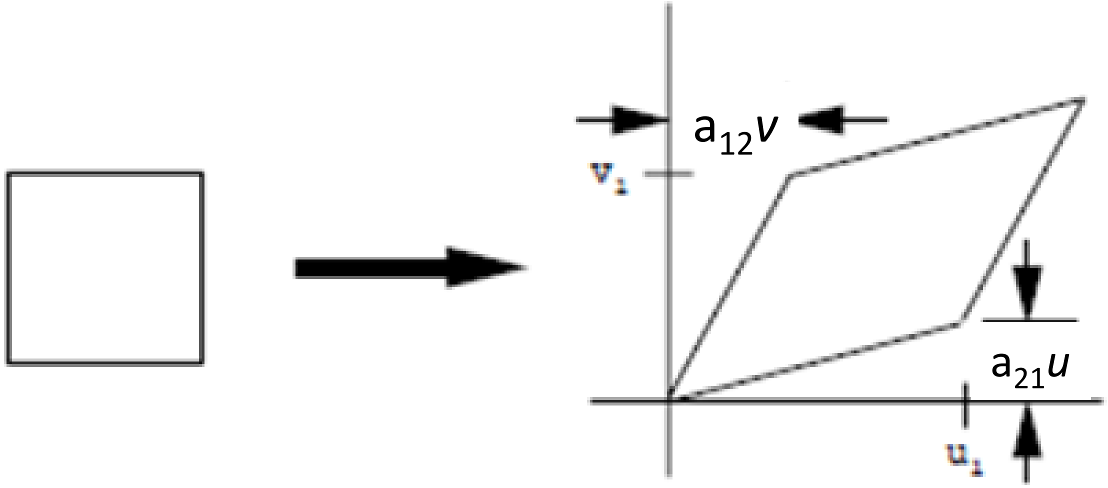


Shear: an affine map transform

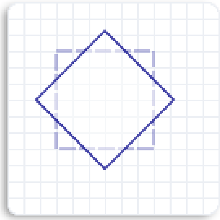


- shear (x, y) = ($u + a_{12}v$, $a_{21}u + v$)

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u + a_{12}v \\ a_{21}u + v \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & a_{12} & 0 \\ a_{21} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

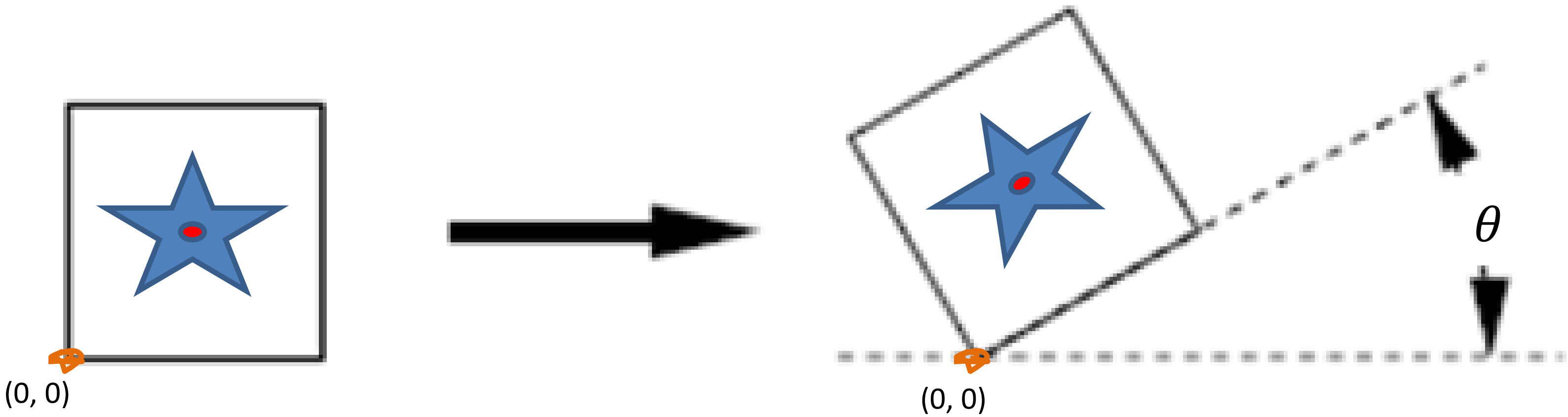


Rotate: an affine map transform



- rotate (x, y) = $(u\ \cos\theta - v\ \sin\theta, u\ \sin\theta + v\cos\theta)$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} u\ \cos\theta - v\ \sin\theta \\ u\ \sin\theta + v\cos\theta \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$



NOTE: rotation is around (0,0)… which might not achieve the expected result

Composing Affine Warps

- R is a rotation
- S is a scale
- T is a translation

$$R \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}$$

$$S \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} u'' \\ v'' \\ 1 \end{bmatrix}$$

$$T \begin{bmatrix} u'' \\ v'' \\ 1 \end{bmatrix} = \begin{bmatrix} u''' \\ v''' \\ 1 \end{bmatrix}$$

First do a rotation, followed by a scale, then a translation
Denote this as:

$$T(S(R \begin{bmatrix} u \\ v \\ 1 \end{bmatrix})) = \begin{bmatrix} u''' \\ v''' \\ 1 \end{bmatrix}$$

By associative property can also denote it as:

$$((TS)R) \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u''' \\ v''' \\ 1 \end{bmatrix}$$

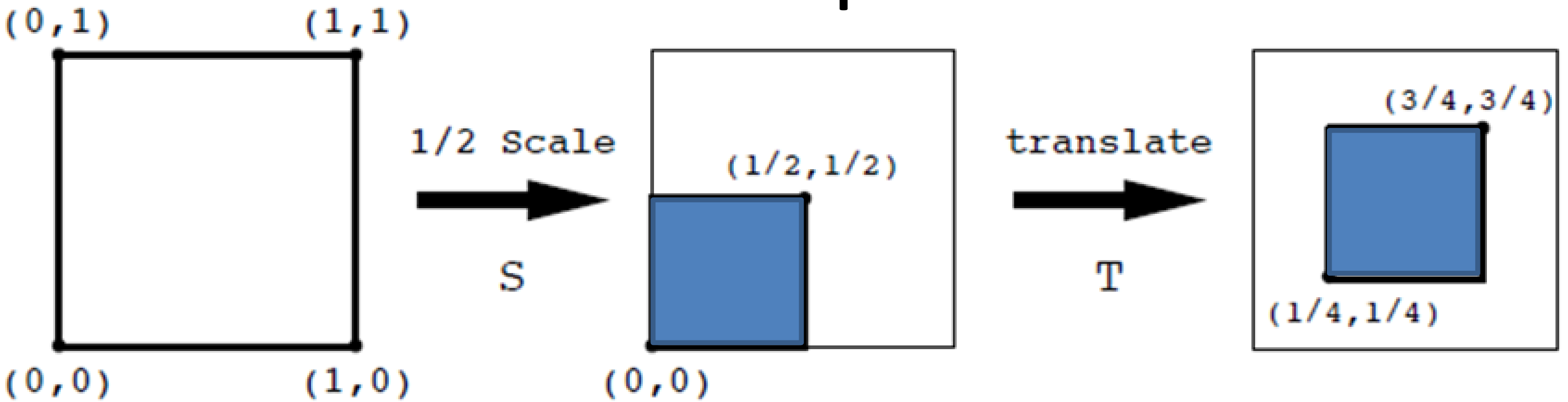
Composing Affine Warps

$$M = TSR,$$

$$M \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u''' \\ v''' \\ 1 \end{bmatrix}$$

- All translations, scales, and rotations can be done using one matrix
- Yields ONE SIMPLE representation
  - Important: order of operations when creating the matrix does matter, be careful
  - i.e. operations are NOT commutative

Example



$$S = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, T = \begin{bmatrix} 1 & 0 & 1/4 \\ 0 & 1 & 1/4 \\ 0 & 0 & 1 \end{bmatrix}$$

$$TS = \begin{bmatrix} 1 & 0 & 1/4 \\ 0 & 1 & 1/4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = TS = \begin{bmatrix} 1/2 & 0 & 1/4 \\ 0 & 1/2 & 1/4 \\ 0 & 0 & 1 \end{bmatrix}$$

T and S Commutative ?

$$TS = \begin{bmatrix} 1 & 0 & 1/4 \\ 0 & 1 & 1/4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = TS = \begin{bmatrix} 1/2 & 0 & 1/4 \\ 0 & 1/2 & 1/4 \\ 0 & 0 & 1 \end{bmatrix}$$

- What is ST ?

$$ST = \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1/4 \\ 0 & 1 & 1/4 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/2 & 0 & 1/8 \\ 0 & 1/2 & 1/8 \\ 0 & 0 & 1 \end{bmatrix}$$

NOT commutative !

Summary: Warps

- Warps are cool
- Homogeneous coordinates are cool
  - Can represent affine warps in one "matrix way"
- Affine warps are awesome
  - Can combine warps into one matrix
  - BUT order matters

Questions?

- Beyond D2L
  - Examples and information can be found online at:
    - *http://docdingle.com/teaching/cs.html*

    - *Continue to more stuff as needed*

Extra Reference Stuff Follows

Credits

- Much of the content derived/based on slides for use with the book:
  - *Digital Image Processing,* Gonzalez and Woods

- Some layout and presentation style derived/based on presentations by
  - Donald House, Texas A&M University, 1999
  - Bernd Girod, Stanford University, 2007
  - Shreekanth Mandayam, Rowan University, 2009
  - Igor Aizenberg, TAMUT, 2013
  - Xin Li, WVU, 2014
  - George Wolberg, City College of New York, 2015
  - Yao Wang and Zhu Liu, NYU-Poly, 2015
  - Sinisa Todorovic, Oregon State, 2015