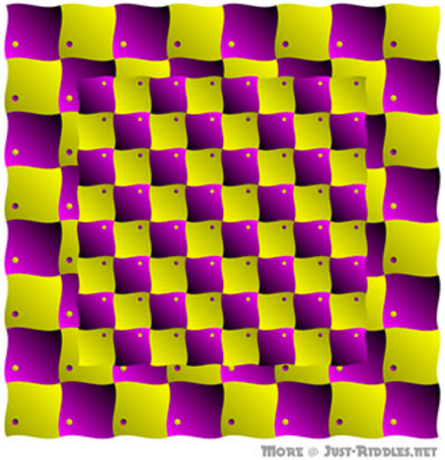
Affine and Perspective Warping

(Geometric Transforms)

Image Processing



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

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

Lecture Objectives

- Previously
  - Image Warping (Geometric Transforms)
- Today
  - Projective Warps
    - Affine Warping (review)
    - Perspective Warping

Outline

- **Projective Warps**
  - **Affine Review**
  - Perspective Warping
  - Concluding Remarks

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:

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

Homogeneous Coordinate System

- 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

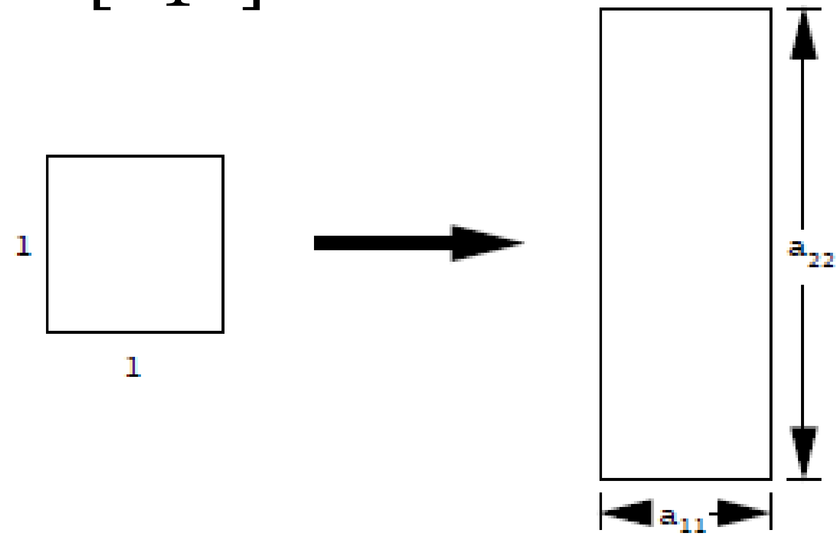
- Homogeneous Coordinates
  - allow affine transformations
    to be easily represented
    by matrix multiplications

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

Scale: an affine map transform

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

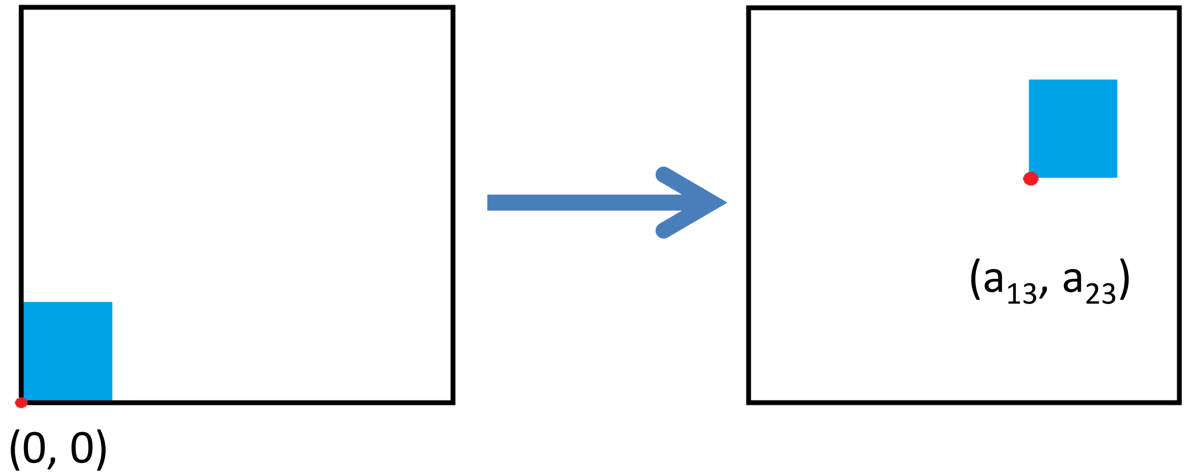
$$\begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11}u \\ a_{22}v \\ 1 \end{bmatrix}$$



Translate: an affine map transform

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

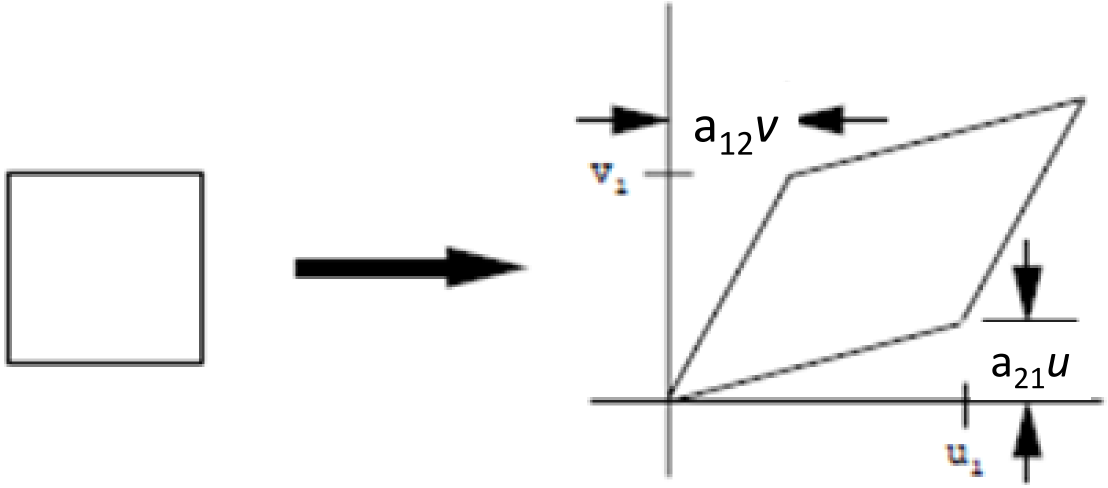
$$\begin{bmatrix}1 & 0 & a_{13}\\0 & 1 & a_{23}\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}u + a_{13}\\v + a_{23}\\1\end{bmatrix}$$



Shear: an affine map transform

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

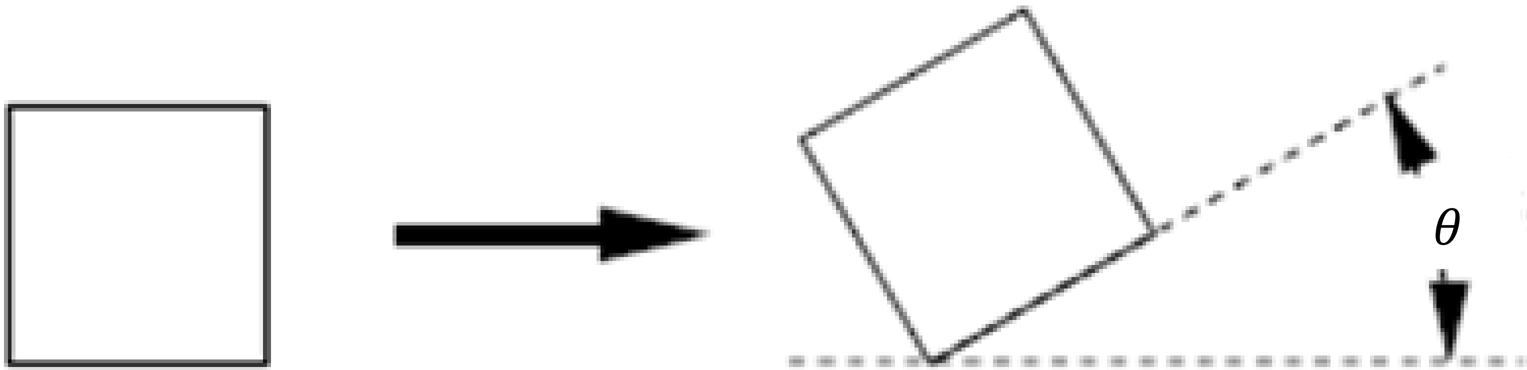
$$\begin{bmatrix} 1 & a_{12} & 0 \\ a_{21} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u + a_{12}v \\ a_{21}u + v \\ 1 \end{bmatrix}$$



Rotate: an affine map transform

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

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



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:

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

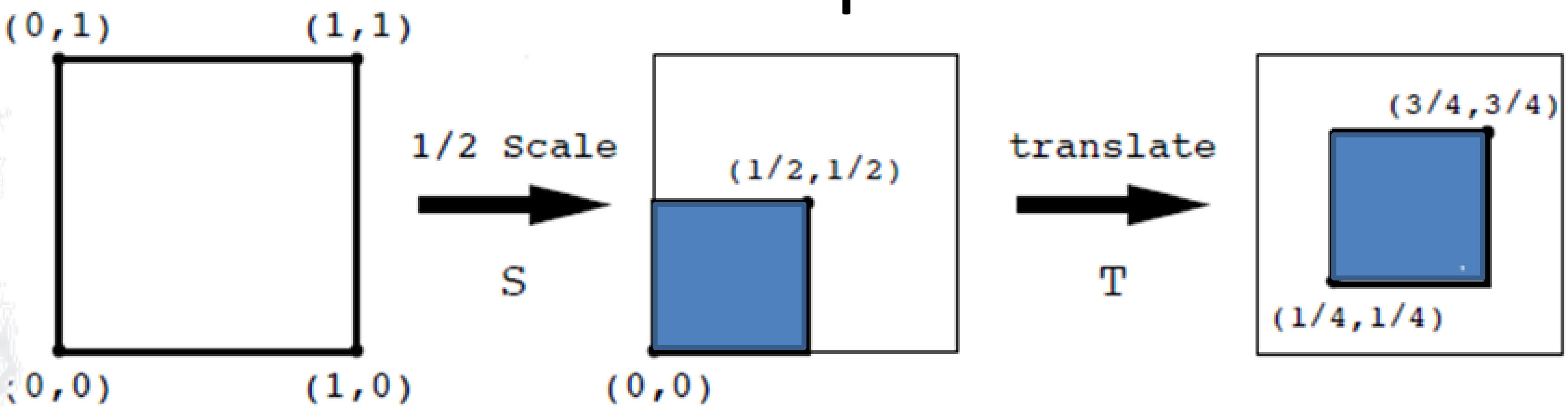
Composing Affine Warps

$$M = TSR,$$

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

- 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 ?

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

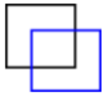
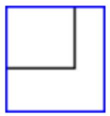
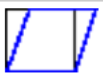
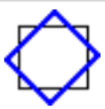
$$M = TS = \begin{bmatrix} 1/2 & 0 & 1/4 \\ 0 & 1/2 & 1/4 \\ 0 & 0 & 1 \end{bmatrix}$$

- 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 !

Affine Summary: ROW vector form

Affine Transform	Example	Transformation Matrix	
Translation		$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	t_x specifies the displacement along the *x* axis t_y specifies the displacement along the *y* axis.
Scale		$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	s_x specifies the scale factor along the *x* axis s_y specifies the scale factor along the *y* axis.
Shear		$\begin{bmatrix} 1 & sh_y & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	sh_x specifies the shear factor along the *x* axis sh_y specifies the shear factor along the *y* axis.
Rotation		$\begin{bmatrix} \cos(q) & \sin(q) & 0 \\ -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$	q specifies the angle of rotation.

Each matrix here is the transpose of what was just presented

Table from: http://www.mathworks.com/help/images/performing-general-2-d-spatial-transformations.html

IMPORTANT:

Know what your abstraction is.

We have been using column vector: $\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$

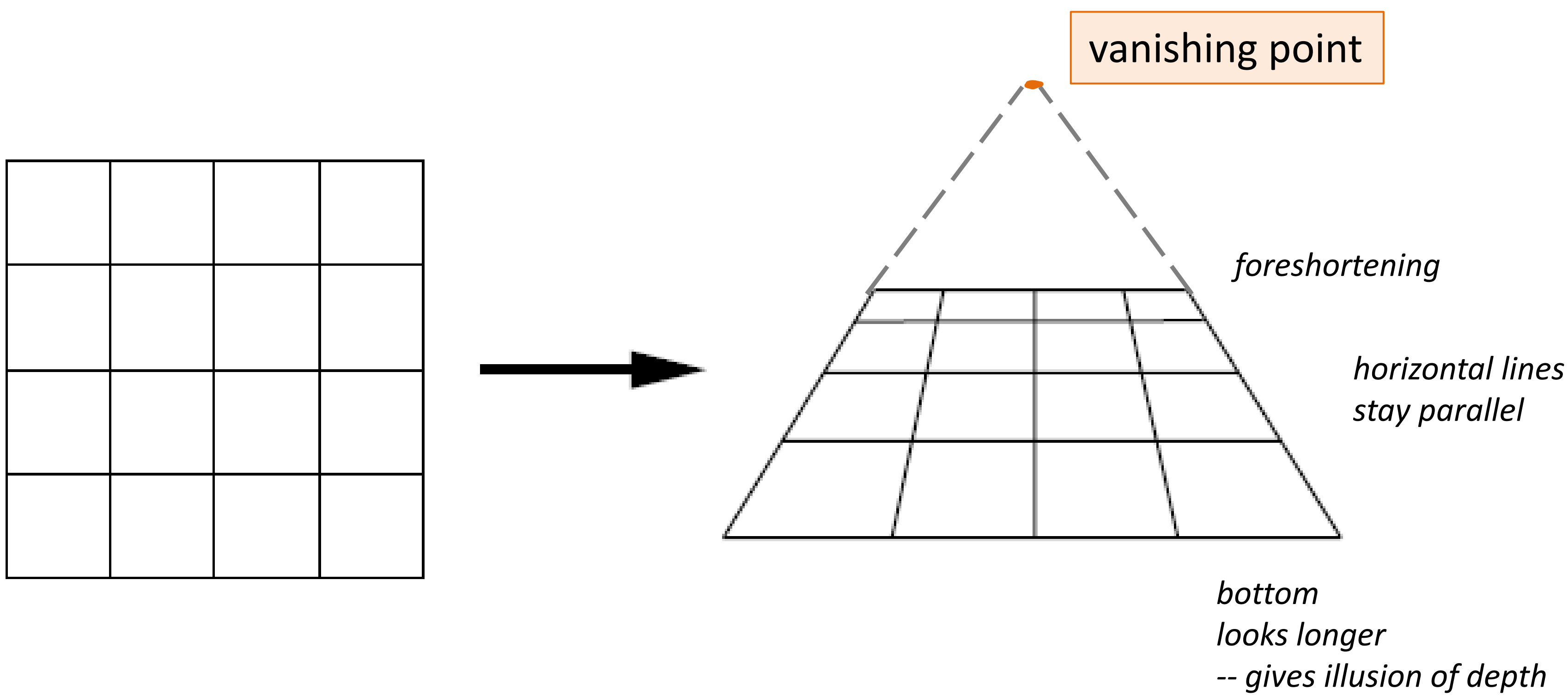
Others, as above, may expect row vector: $[u \quad v \quad 1]$

Outline

- Projective Warps
  - Affine Review
  - **Perspective Warping**
    - NOT affine
    - Subset of Projective Mapping
  - Concluding remarks

Perspective Warps: Non-Affine Transform

- Perspective warps are NOT affine
  - Not all parallel lines stay parallel
  - But lines do stay lines
  - And provides a 3D feeling



Aside: Perspective Warping will be seen to be a 'subset' of Projective Transforms
--- just as scale, translate, rotate, shear, are 'subsets' of general affine

Perspective Transform: Step 1

- Matrix Multiply
  - Third coordinate, w, of result is no longer 1

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ \boxed{a_{31}u + a_{32}v + 1} \end{bmatrix} = \text{w}$$

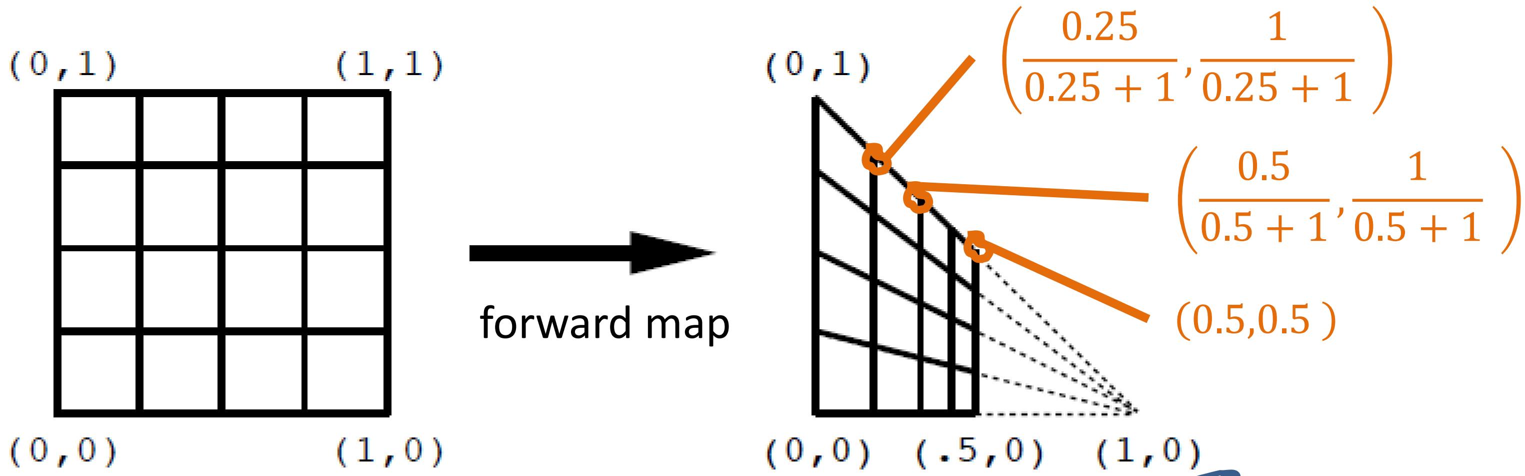
Perspective Transform: Part 2

- Restore points to homogeneous coordinates
  - with w = 1
  - divide each vector by its own *w* coordinate

$$\frac{1}{w}\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} u/w \\ v/w \\ 1 \end{bmatrix}$$

NOTE: *w* is different for each point

Example



$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$$P \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ u+1 \end{bmatrix} \Longrightarrow \begin{bmatrix} \frac{u}{u+1} \\ \frac{v}{u+1} \\ 1 \end{bmatrix}$$

vanishing point at infinity

$$\lim_{u \to \infty} \frac{u}{u+1} = 1$$

$$\lim_{u \to \infty} \frac{v}{u+1} = 0$$

Perspective Warp $\subseteq$ Projective Map

- *Perspective Warping*
  is a ‘type’ of Projective Transform

  – just as
    scale, translate, rotate, shear,
    are ‘types’ of general affine transforms

Projective Map: General Equation

$$x = \frac{a_{11}u + a_{12}v + a_{13}}{a_{31}u + a_{32}v + a_{33}}$$

$$y = \frac{a_{21}u + a_{22}v + a_{23}}{a_{31}u + a_{32}v + a_{33}}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{33} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{a_{11}u + a_{12}v + a_{13}}{a_{31}u + a_{32}v + a_{33}} \\ \frac{a_{21}u + a_{22}v + a_{23}}{a_{31}u + a_{32}v + a_{33}} \\ 1 \end{bmatrix}$$

Projective Back to Perspective

$$x = \frac{a_{11}u + a_{12}v + a_{13}}{a_{31}u + a_{32}v + a_{33}}$$

$$y = \frac{a_{21}u + a_{22}v + a_{23}}{a_{31}u + a_{32}v + a_{33}}$$

$$\begin{bmatrix} \boldsymbol{a_{11}} & \boldsymbol{a_{12}} & \boldsymbol{a_{13}} \\ \boldsymbol{a_{21}} & \boldsymbol{a_{22}} & \boldsymbol{a_{33}} \\ \boldsymbol{a_{31}} & \boldsymbol{a_{32}} & \boldsymbol{a_{33}} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} \dfrac{\boldsymbol{a_{11}u + a_{12}v + a_{13}}}{\boldsymbol{a_{31}u + a_{32}v + a_{33}}} \\ \dfrac{\boldsymbol{a_{21}u + a_{22}v + a_{23}}}{\boldsymbol{a_{31}u + a_{32}v + a_{33}}} \\ \mathbf{1} \end{bmatrix}$$

For the perspective case (just described) most of the coefficients becomes 0 or 1

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \dfrac{u}{a_{31}u + a_{32}v + 1} \\ \dfrac{v}{a_{31}u + a_{32}v + 1} \\ 1 \end{bmatrix}$$

Projective Back to Perspective

And we separated the process into TWO steps

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ a_{31}u + a_{32}v + 1 \end{bmatrix} = \begin{bmatrix} \dfrac{u}{a_{31}u + a_{32}v + 1} \\ \dfrac{v}{a_{31}u + a_{32}v + 1} \\ 1 \end{bmatrix}$$

step 1

step 2

Summary: Affine and Perspective Warps

- Warps are cool
- Affine warps are awesome
  - Can combine warps into one matrix
  - BUT order matters
- Perspective warps rock
  - Are NOT affine
  - Use same matrix idea as affine
  - Are a type of projective map

Outline

- Projective Warps
  - Affine Review
  - Perspective Warping
  - **Concluding Remarks**

Projective Warps

- Projective Warps are
  Affine, Perspective or Composite of the two
  - Affine is Perspective with w = 1

$$\frac{1}{w}\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} u/w \\ v/w \\ 1 \end{bmatrix}$$

$$\frac{1}{a_{31}u + a_{32}v + 1}\begin{bmatrix} u \\ v \\ a_{31}u + a_{32}v + 1 \end{bmatrix} = \begin{bmatrix} \dfrac{u}{a_{31}u + a_{32}v + 1} \\ \dfrac{v}{a_{31}u + a_{32}v + 1} \\ 1 \end{bmatrix}$$

Affine cases:
w = 1 → a31 and a32 = 0

Inverse Map

The general formula for an inverse of a matrix mapping, *M* is

$$M^{-1} = \frac{\mathcal{A}(M)}{|M|}$$

where |M| is the determinant of the matrix M
and A(M) is the adjoint of M

Inverse map of Projective Warp

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

$$|M| = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{32}a_{21} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{32}a_{23}$$

$$\mathcal{A}(M) = \begin{bmatrix} a_{22}a_{33} - a_{23}a_{32} & a_{13}a_{32} - a_{12}a_{33} & a_{12}a_{23} - a_{13}a_{22} \\ a_{23}a_{31} - a_{21}a_{33} & a_{11}a_{33} - a_{13}a_{31} & a_{13}a_{21} - a_{11}a_{23} \\ a_{21}a_{32} - a_{22}a_{31} & a_{12}a_{31} - a_{11}a_{32} & a_{11}a_{22} - a_{12}a_{21} \end{bmatrix}$$

$$M^{-1} = \frac{\mathcal{A}(M)}{|M|}$$

Summary of Warps and Process

- As an exercise it may be useful to closely follow the next few slides
  - Walk through some examples by hand and by code

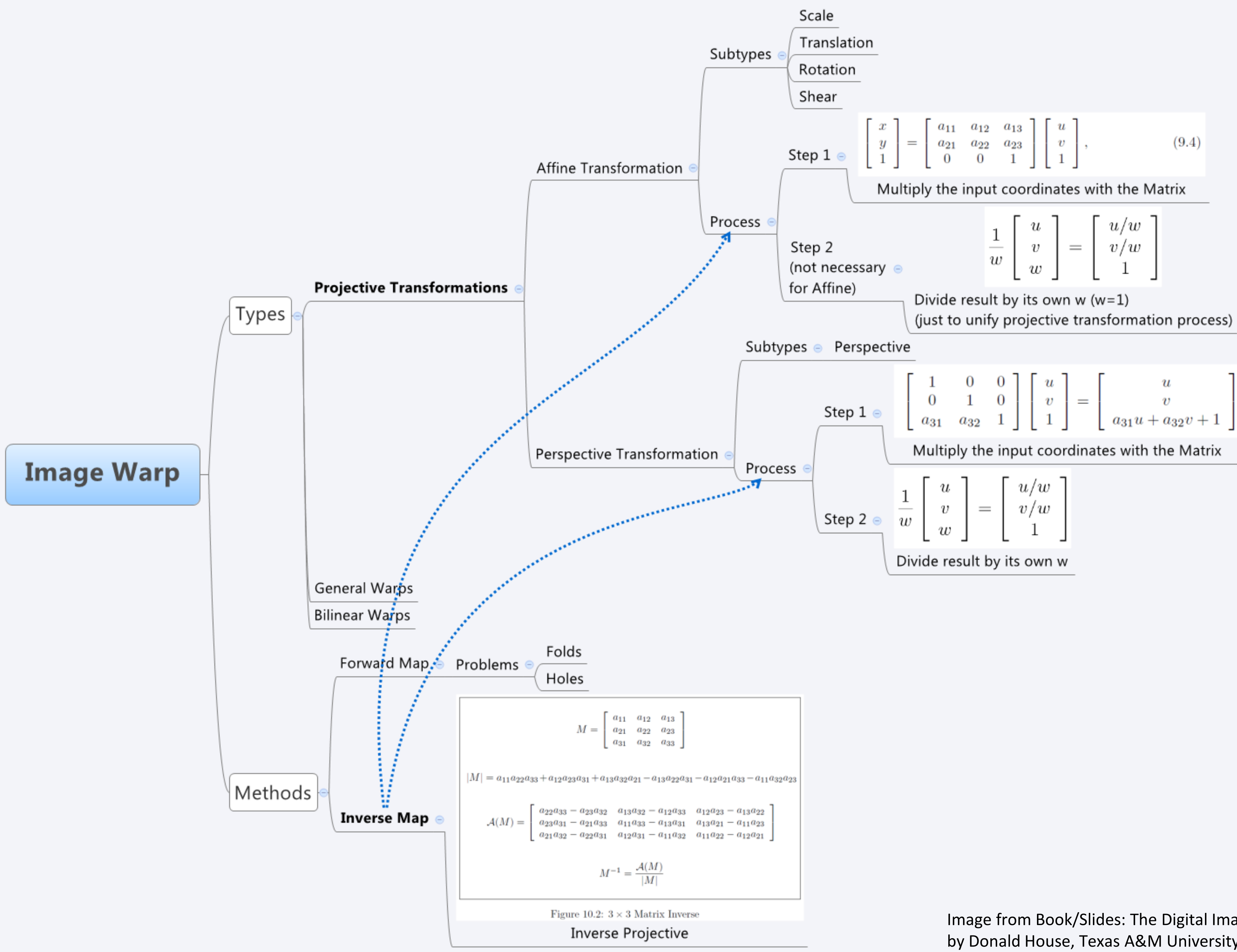


Image from Book/Slides: The Digital Image by Donald House, Texas A&M University

Step 1: Build Transformation Matrix *M*

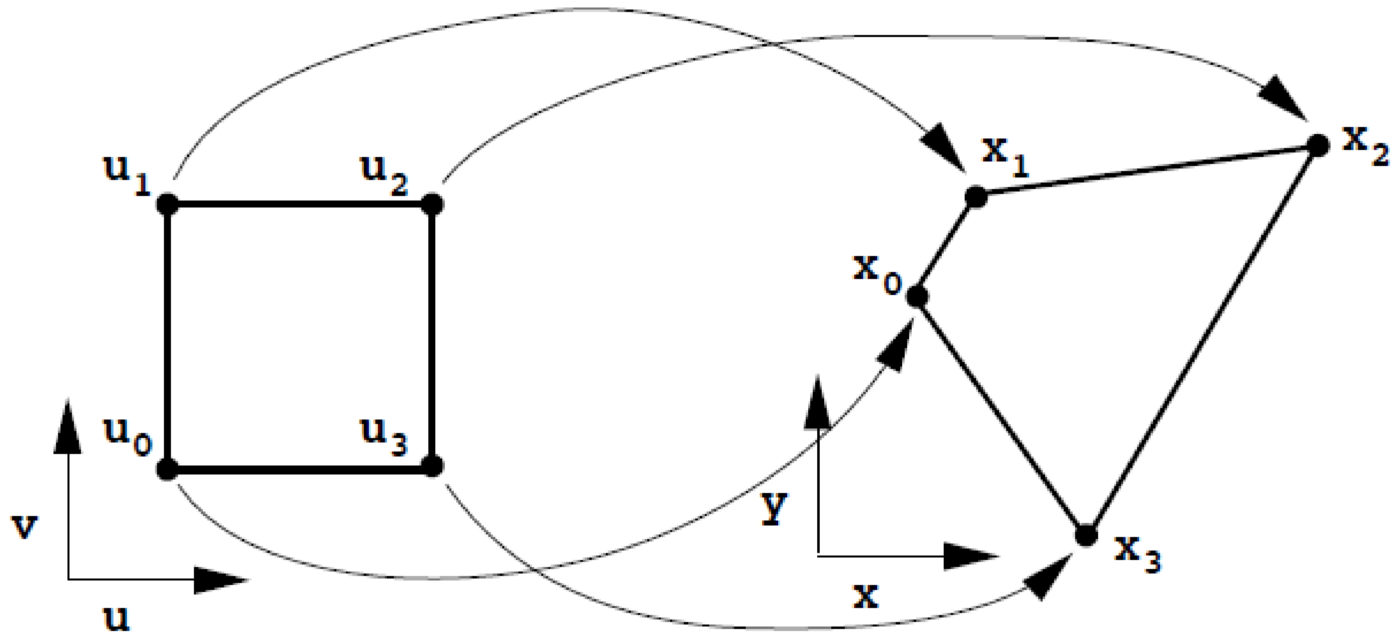
- A composite transformation matrix is easily constructed from a series of more simple transformations
  - Initialize *M* to the identity matrix

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  - Then for each simpler transform, *T*, pre-multiply the matrix *M* by *T*
    - Replacing M by the product *TM*

$$M \leftarrow TM$$

Step 2: (Think) Forward Map the corners



$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = M \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{33} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}$$

$$x_i = \frac{a_{11}u_i + a_{12}v_i + a_{13}}{a_{31}u_i + a_{32}v_i + a_{33}}$$

$$y_i = \frac{a_{21}u_i + a_{22}v_i + a_{23}}{a_{31}u_i + a_{32}v_i + a_{33}}$$

ASIDE:
This is a good debug check to make sure matrix is correct
See if the corners via forward map go where you expect

Step 3: Find the Inverse Transform

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

$$|M| = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{32}a_{21} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{32}a_{23}$$

$$\mathcal{A}(M) = \begin{bmatrix} a_{22}a_{33} - a_{23}a_{32} & a_{13}a_{32} - a_{12}a_{33} & a_{12}a_{23} - a_{13}a_{22} \\ a_{23}a_{31} - a_{21}a_{33} & a_{11}a_{33} - a_{13}a_{31} & a_{13}a_{21} - a_{11}a_{23} \\ a_{21}a_{32} - a_{22}a_{31} & a_{12}a_{31} - a_{11}a_{32} & a_{11}a_{22} - a_{12}a_{21} \end{bmatrix}$$

$$M^{-1} = \frac{\mathcal{A}(M)}{|M|}$$

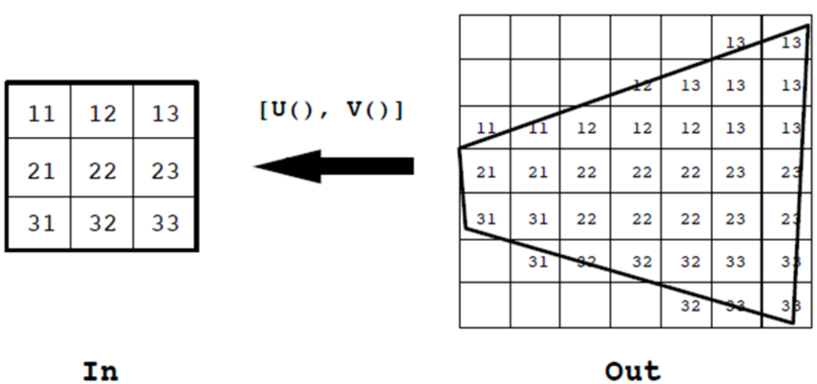
$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = M \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \rightarrow \boxed{M^{-1} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}}$$

Step 4: Apply the Inverse Transform

- Loop through the output image pixel by pixel
  - Identify the input image pixel each is mapped to
    - and assign each the corresponding color

```
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))];
```

$$\begin{bmatrix} U(x_i, y_i) \\ V(x_i, y_i) \\ 1 \end{bmatrix} = M^{-1} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$



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