CPSC 4040/6040 Computer Graphics Images

Joshua Levine
levinej@clemson.edu

Lecture 19
Projective Warping and Bilinear Warping

Nov. 3, 2015

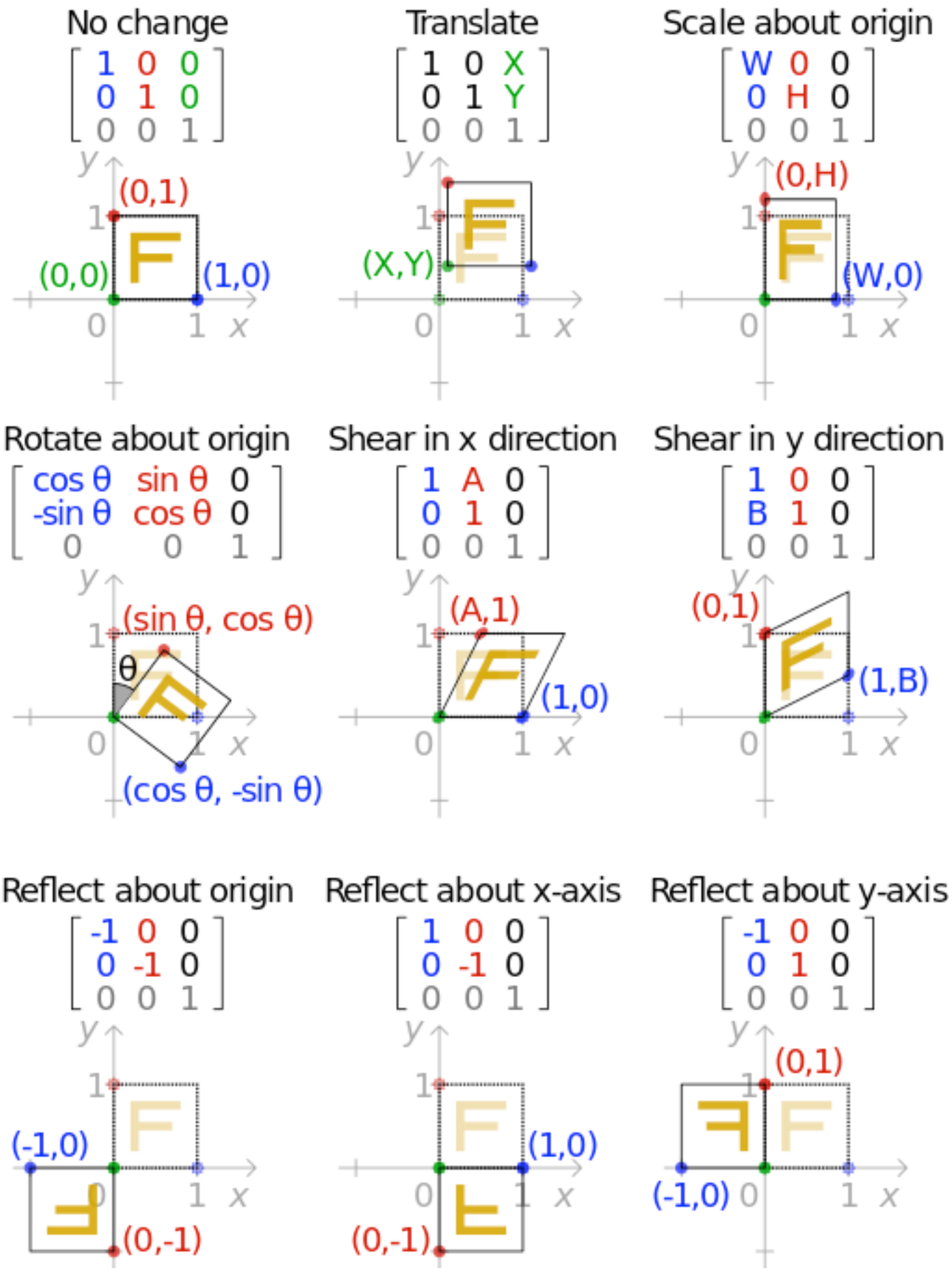
Agenda

- EC Quiz Review

- PA06 out

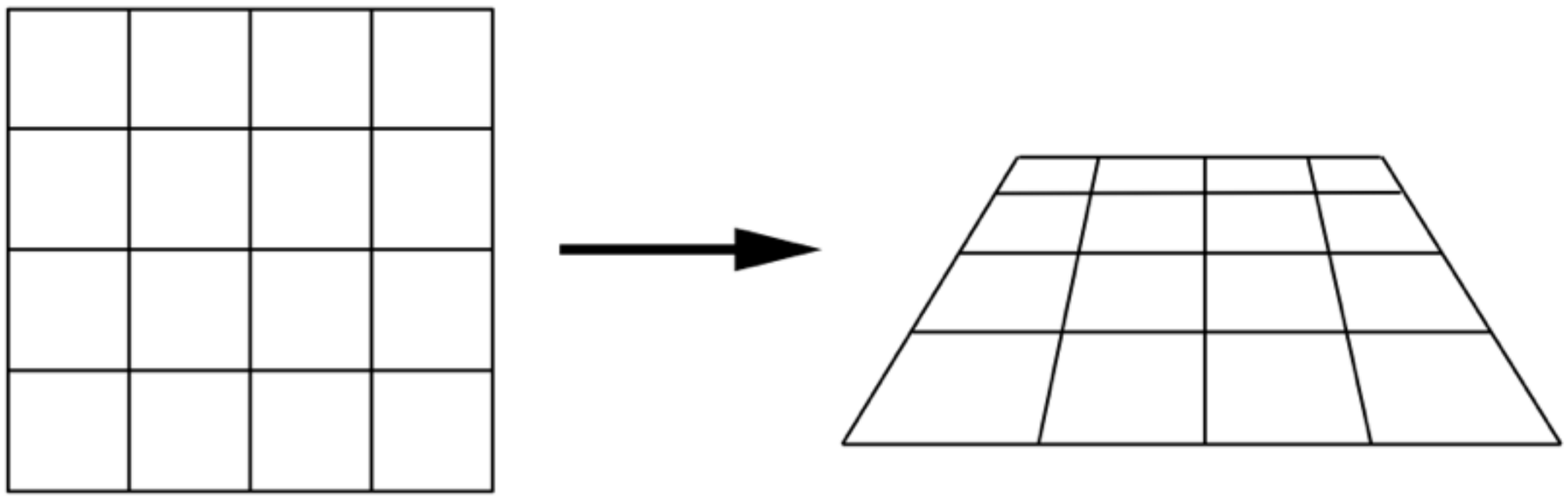
Refresher from Lec18

https://en.wikipedia.org/wiki/Affine_transformation



Perspective Warping

- This form of warping stretches and squishes in different places
- Straight lines stay straight, but not necessarily parallel



Normalizing Coordinates

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

- Rewrite as:

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

This is no longer linear

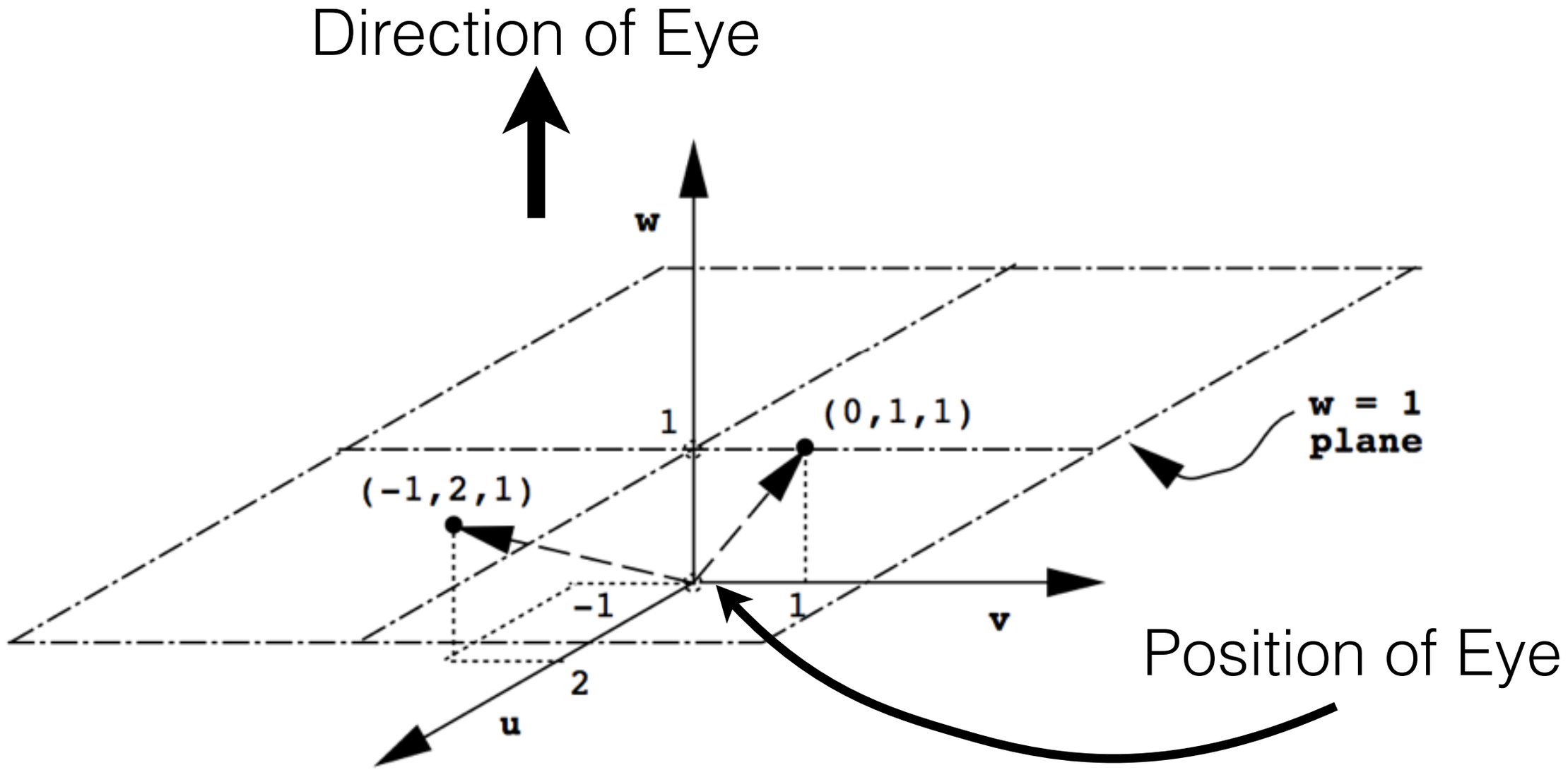


Figure 9.10: Homogeneous Coordinates Lie on the Plane $w = 1$

- Eye is located at (0,0,0), looking down w axis
- Dividing by w shortens vector to lie in w = 1 plane
- Given (x,y,w) and (x/w,y/w,1), the point still lies on the same ray exiting from the eye passing through (0,0,0) and (x,y,w)

Matrix-Based Perspective Warping Algorithm

- Given an input image, IN, of width W and height H as well as a 3x3 matrix M:

1. Using the forward map, M, compute W2, H2 to store the width and height of the bounding box of OUT
2. Allocate output image OUT
3. Compute inverse matrix invM
4. Use the inverse map (invM) to fill in all of pixels of OUT from their respective pixels in IN

Projective Warping

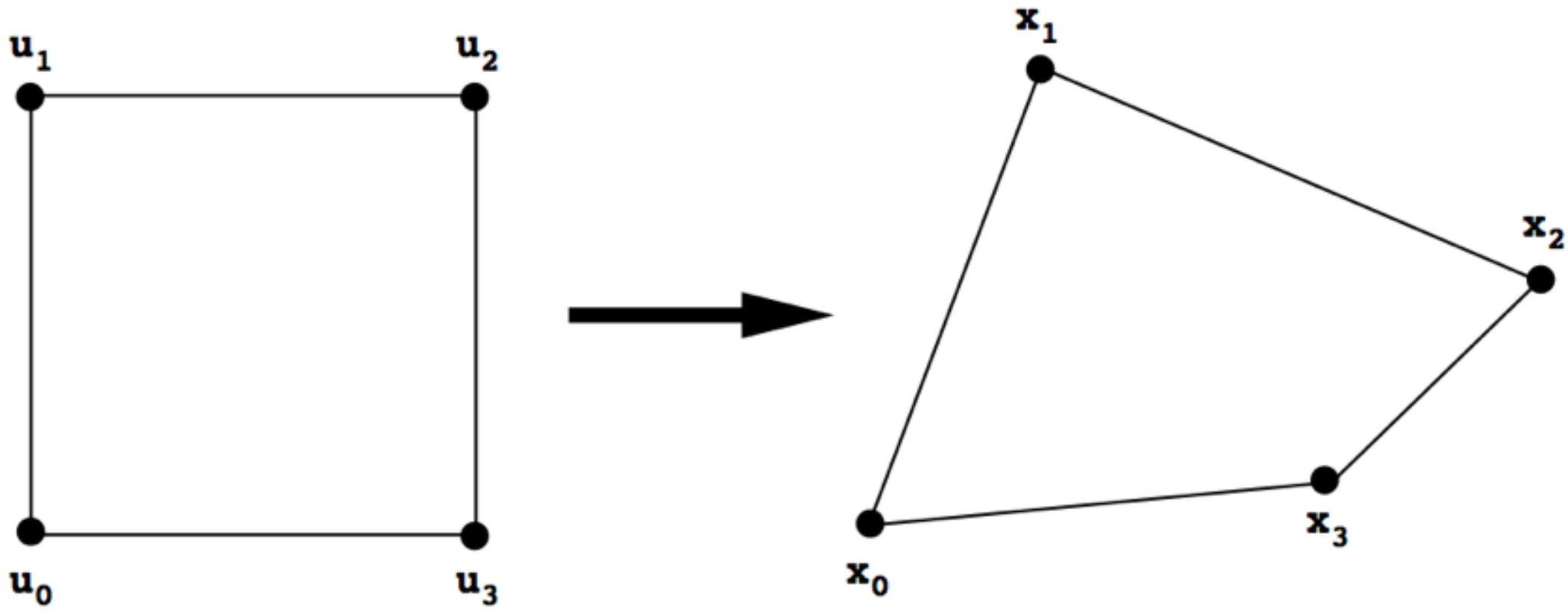
User Driven Warping

- Specifying a matrix can be non-intuitive
- Instead of specifying a matrix to warp, a more useful mechanism would be allowing the user to move the corners

- We can express this as a matrix warp!

Projective Warps

- What is the matrix?
- What the knowns? Unknowns? How many?



Algebra

- A general 3x3 matrix can express this entire class of warps (9 unknowns)
  - a_{33} acts as a global scale parameter, so we can always set it to 1 without losing generality
- The remaining 8 unknowns can be solved by 4 pairs of equations using the 8 known x_i, y_i values and the 8 known u_i, v_i values
- Solving these 8 equations gives the 8 remaining a_{ij} unknowns

$$x_i = \frac{a_{11}u_i + a_{12}v_i + a_{13}}{a_{31}u_i + a_{32}v_i + 1}$$

$$y_i = \frac{a_{21}u_i + a_{22}v_i + a_{23}}{a_{31}u_i + a_{32}v_i + 1}$$

Inverting Projective Matrices

Matrix Inverse

- It's clear that if M is a projective warp, then invM is also a projective warp.
    - Existence is intuitive: we're warping from one quadrilateral to another in either direction
- Transformation happens with the same process:
    1. Promote (x,y) to (x,y,1)
    2. Compute (u,v,w) = invM * (x,y,1)
    3. Normalize by dividing by w.

Summary: Forward Projective Warp

$$(u, v) \longrightarrow \begin{bmatrix} u \\ v \end{bmatrix} \Longrightarrow \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

1. Promote
2. Multiply
3. Normalize

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

$$1/w' \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} u'/w' \\ v'/w' \\ 1 \end{bmatrix} \Longrightarrow \begin{bmatrix} u'/w' \\ v'/w' \end{bmatrix} \longrightarrow (x, y)$$

Same Thing: Inverse Projective Warp

$$(x, y) \longrightarrow \begin{bmatrix} x \\ y \end{bmatrix} \Longrightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

1. Promote
2. Multiply
3. Normalize

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

$$1/w'' \begin{bmatrix} x'' \\ y'' \\ w'' \end{bmatrix} = \begin{bmatrix} x''/w'' \\ y''/w'' \\ 1 \end{bmatrix} \Longrightarrow \begin{bmatrix} x''/w'' \\ y''/w'' \end{bmatrix} \longrightarrow (u, v)$$

Inverse Projective Warps

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

Determinant

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

Adjoint

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

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

Can skip dividing by the determinant, since we're going to do a global scale w., but this will leave $a_{33} \neq 1$

Bilinear Warping

Problem: Foreshortening

- Projective warps tilt the image plane, so things which move further from the eye get closer together

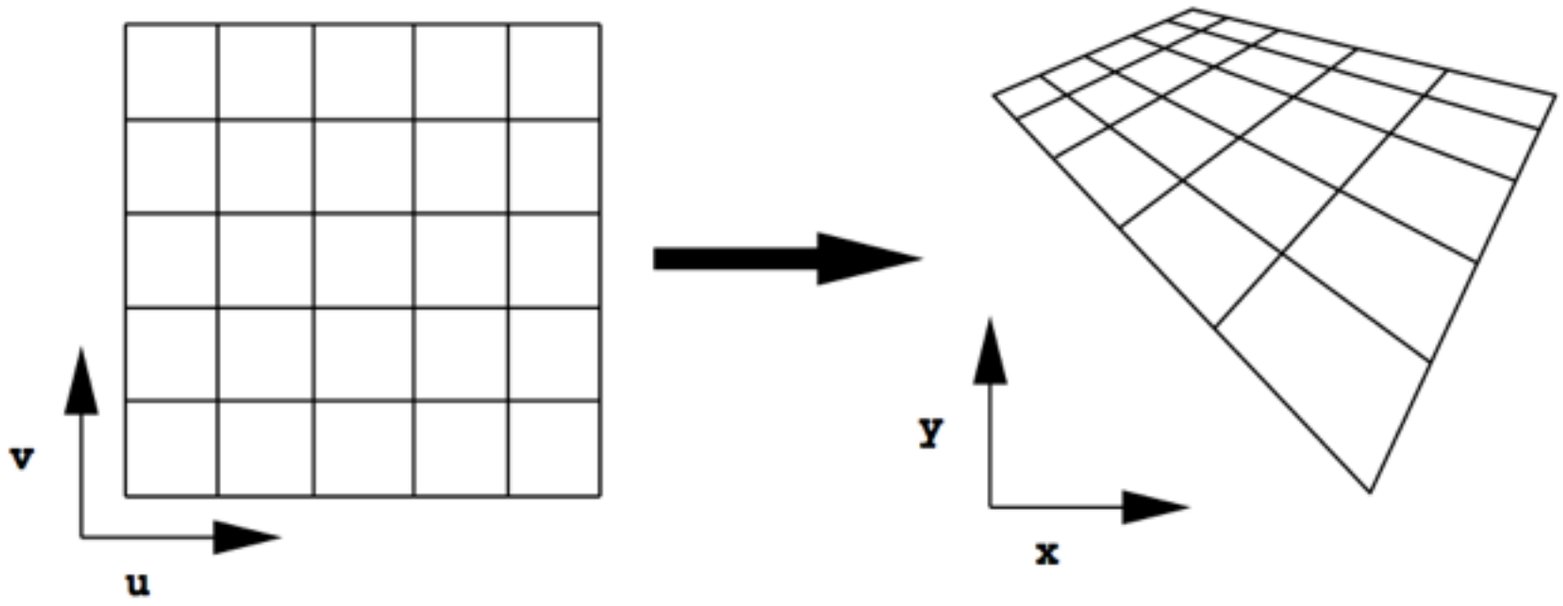


Figure 10.3: Foreshortening due to Perspective

Equal Spacing With Bilinear Warp

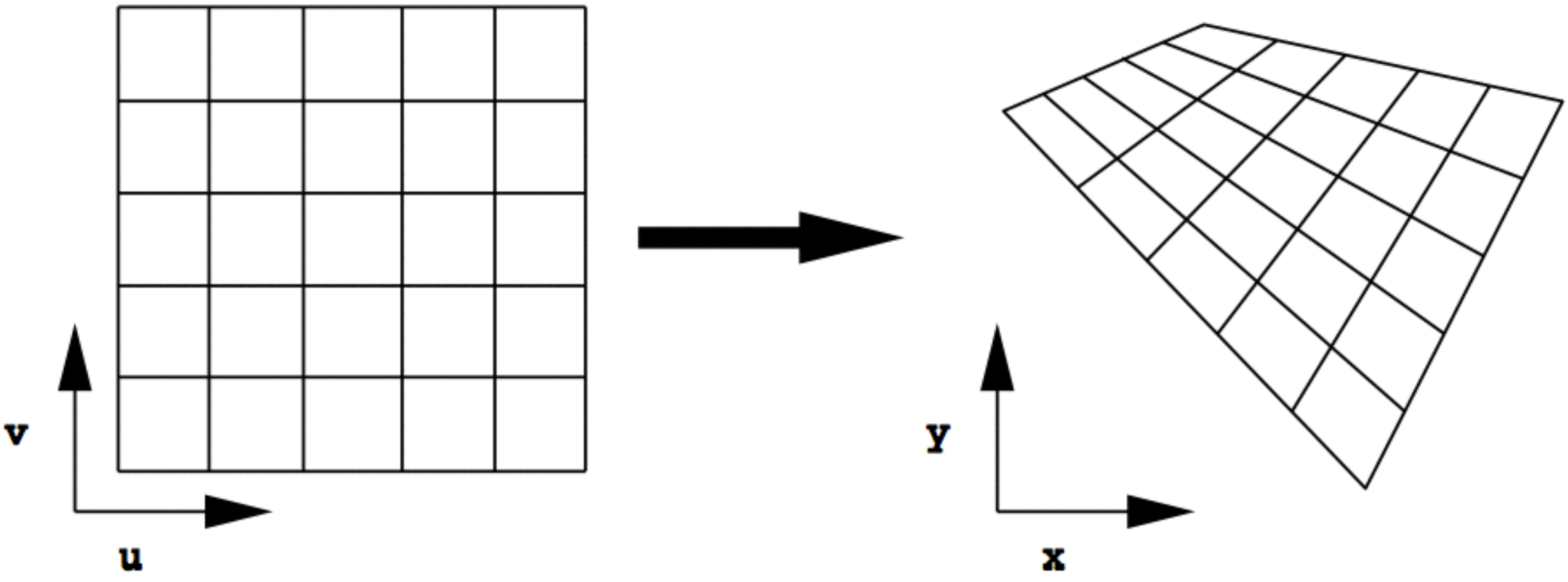
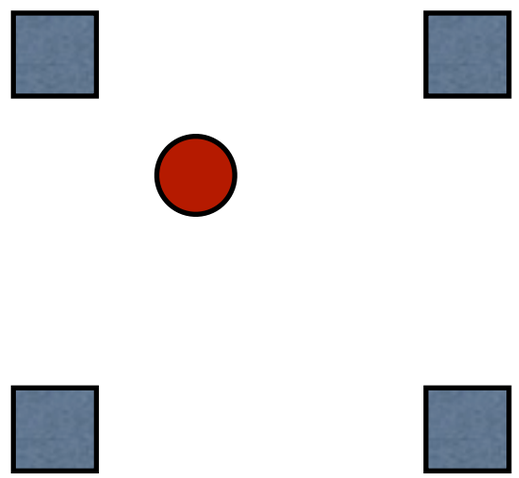


Figure 10.4: Even Spacing under Bilinear Warp

Recall: Bilinear Interpolation

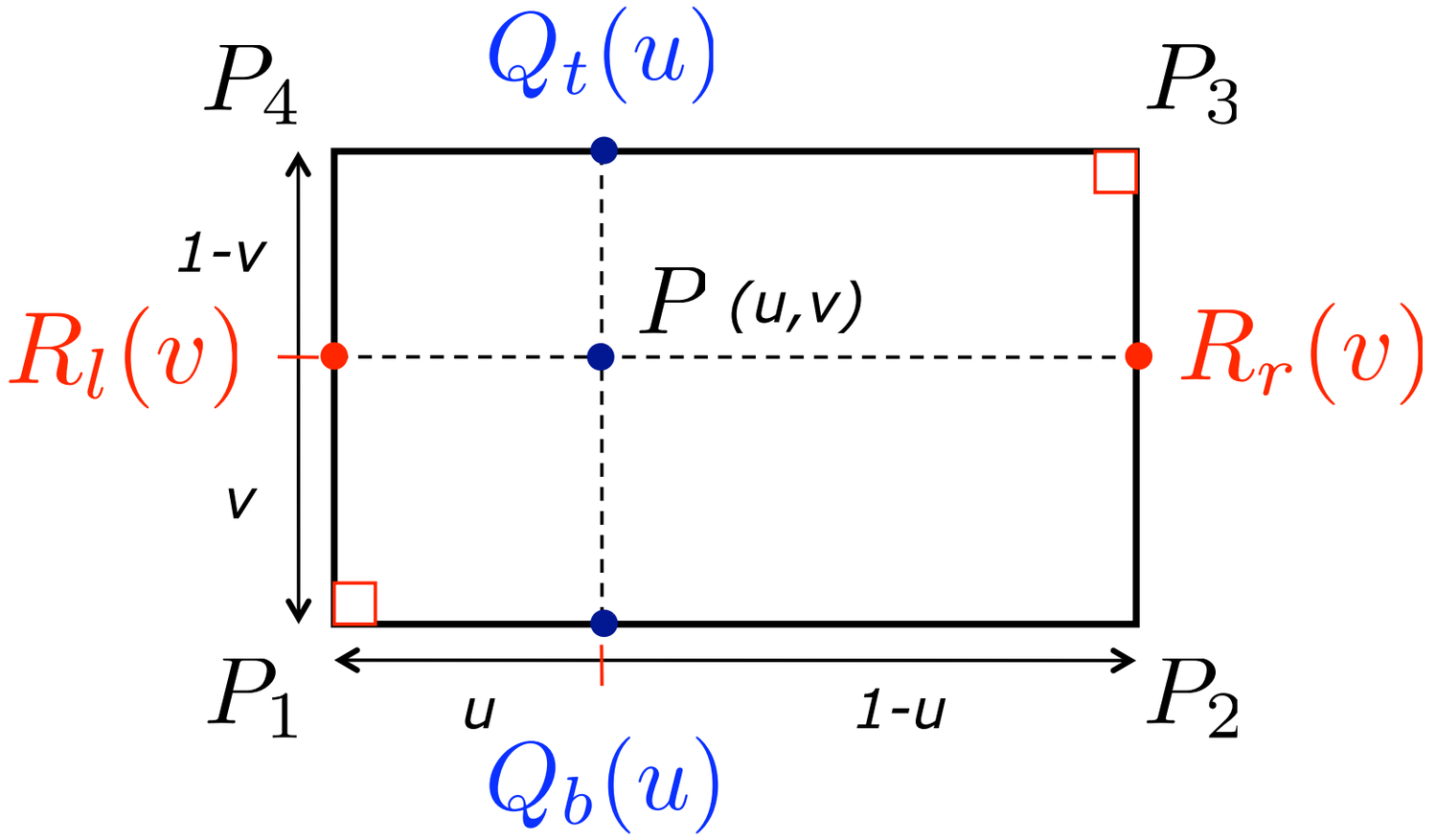
- Examine the four nearest neighbors, corners of the box which contain a sample
- Weight according to horizontal and vertical offsets relative to the box
- We can express this as linear interpolations in both the x and y direction



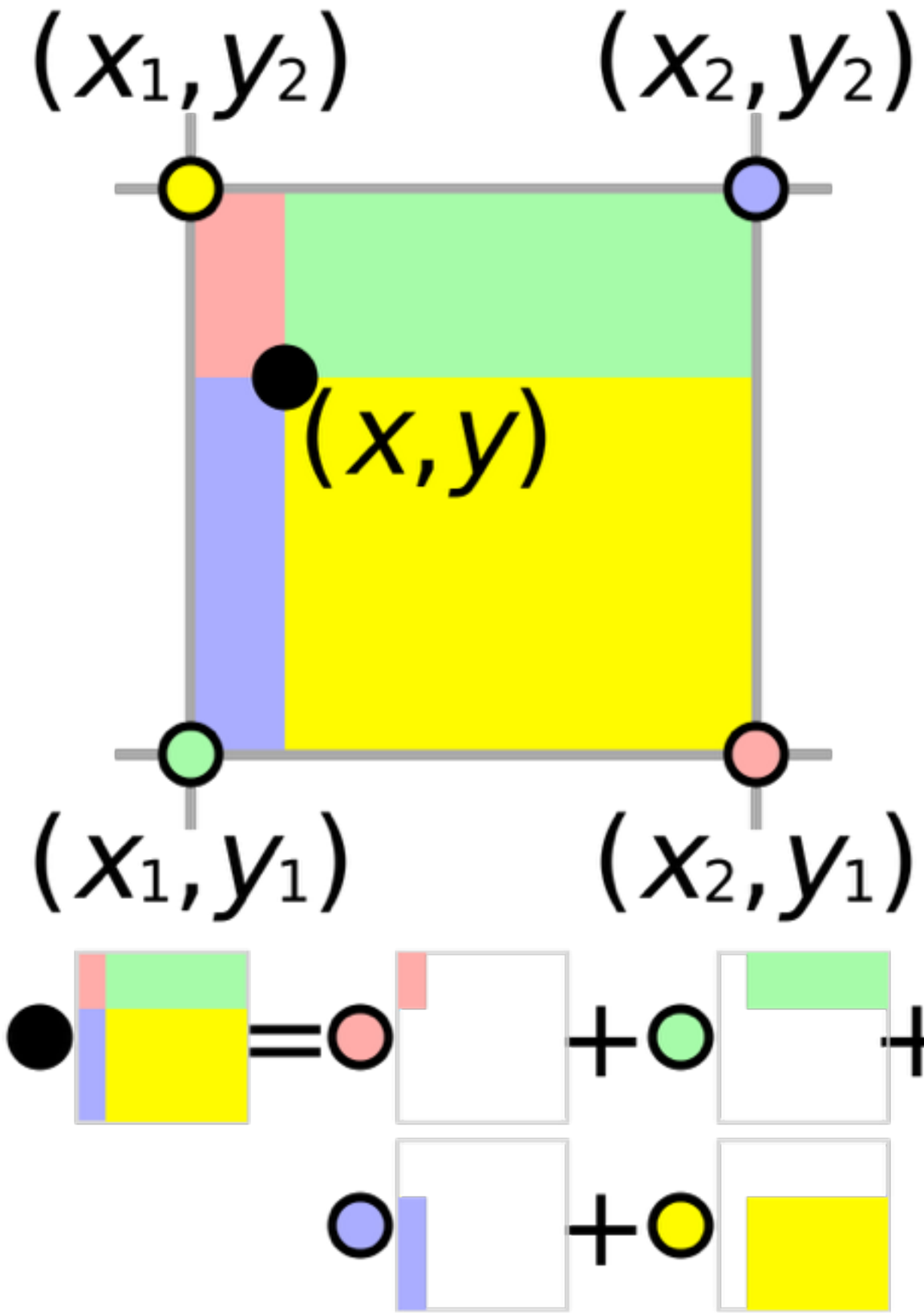
Bilinear Interpolation

- In rectangle

$$P = (1-v)Q_b(u) + vQ_t(u)$$
$$= (1-u)R_l(v) + uR_r(v)$$



Bilinear Interpolation



- Alternate interpretation is a weighted sum of the four pixel values

- Weights defined by the area opposite each corner

http://en.wikipedia.org/wiki/Bilinear_interpolation

Recall: Bilinear Interpolation

- Bilinear interpolation is a weighted average where pixels closer to the backward mapped coordinate are weighted proportionally heavier than those pixels further away.

- Bilinear interpolation acts like something of a rigid mechanical system

1. Two rods vertically connect the four samples surrounding the backward mapped coordinate.

2. A third rod is connected horizontally which is allowed to slide vertically up and down the fixture.

3. A ball is attached to this horizontal rod and is allowed to slide freely back and forth across the central rod, determining the interpolated sample value wherever the ball is located.

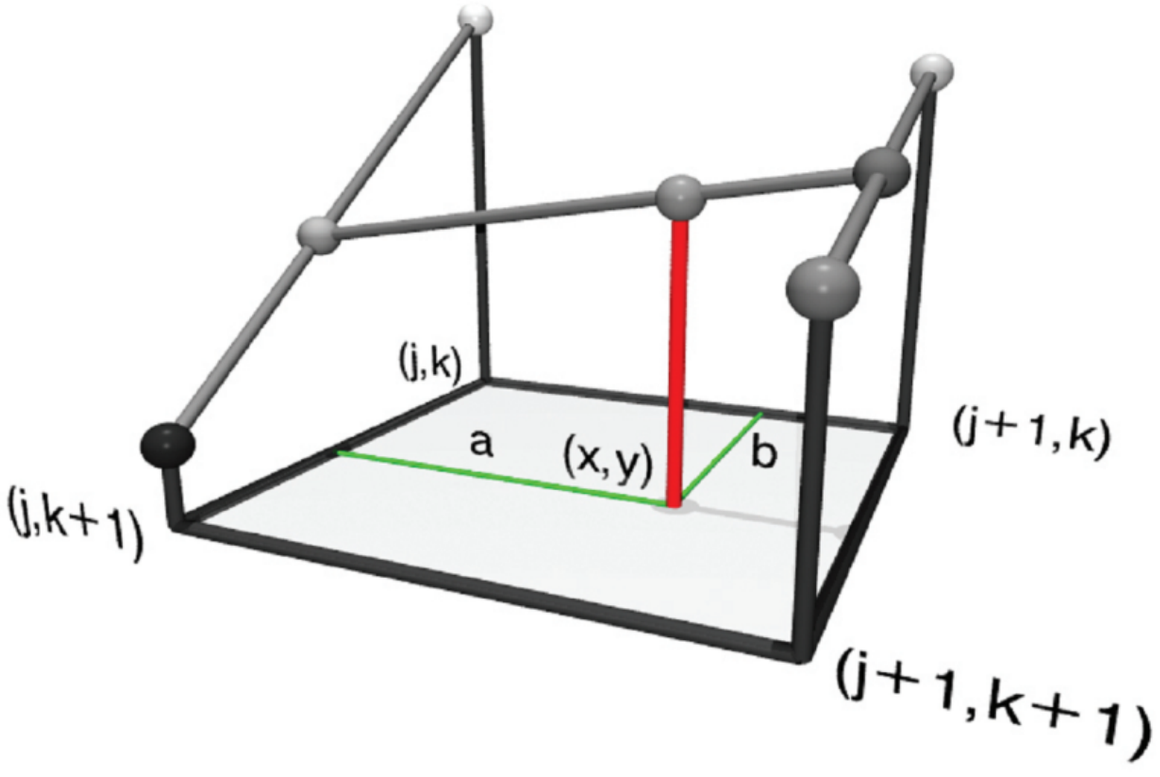


Figure 7.5. Bilinear interpolation.

Bilinear Warping

- Key Idea: Instead of using bilinear interpolation for pixel color values, we can use it to interpolate the positions in the warped image

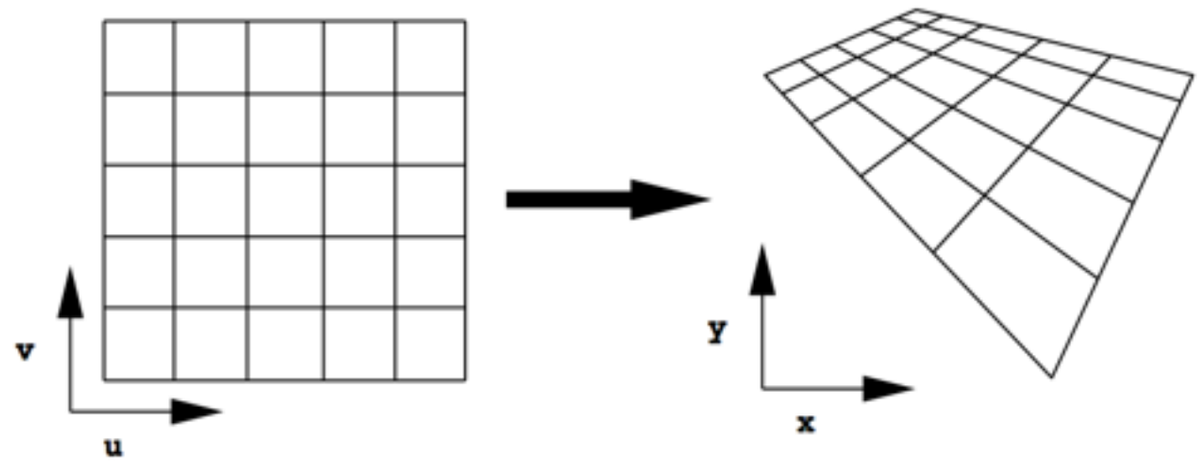


Figure 10.3: Foreshortening due to Perspective

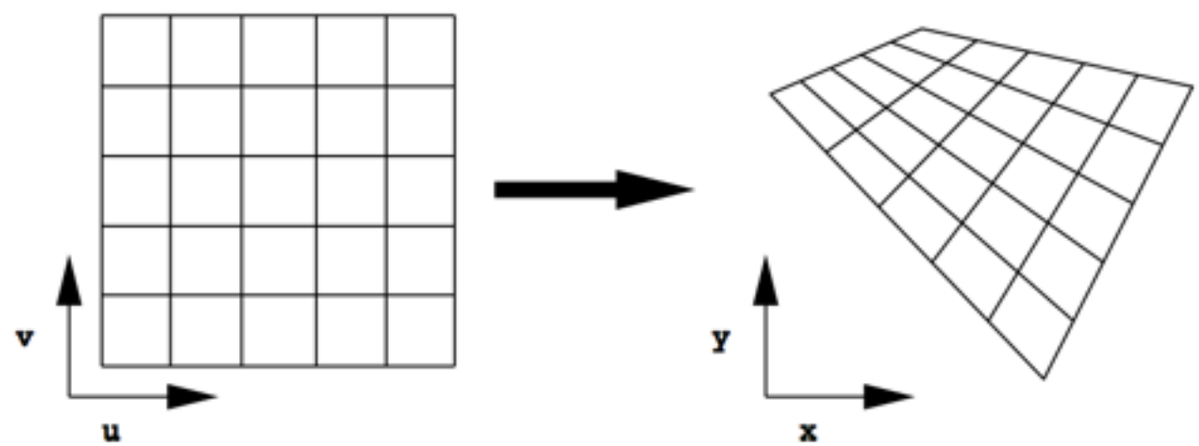


Figure 10.4: Even Spacing under Bilinear Warp

Bilinear Warping

1. Establish correspondence between each corner of the input and output image

2. Using horizontal offsets in u space, identify crossing points x_{03} and x_{12} — the intermediate linear interpolants

3. Using vertical offset in v space, identify point x along the line connecting x_{03} and x_{12}

Step 1: Identify Correspondences

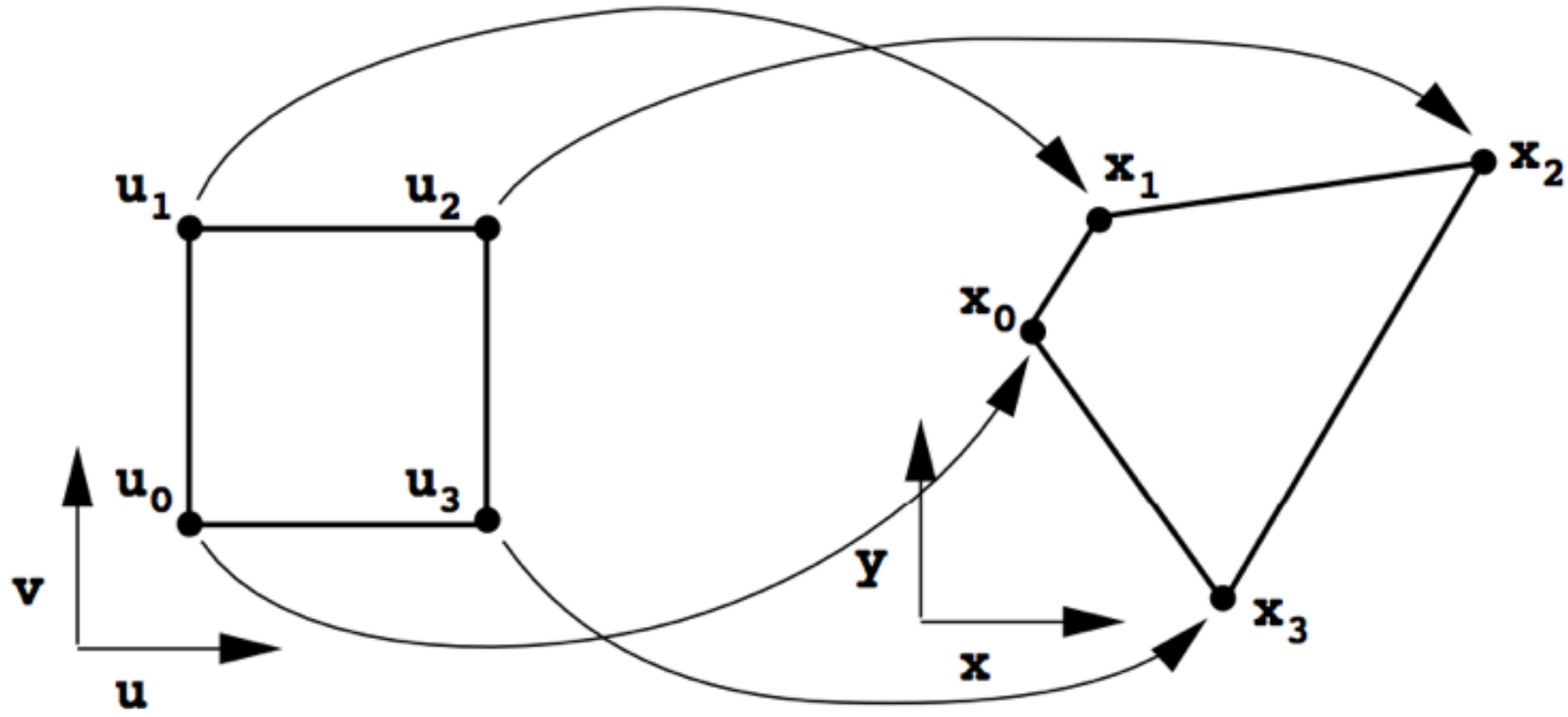
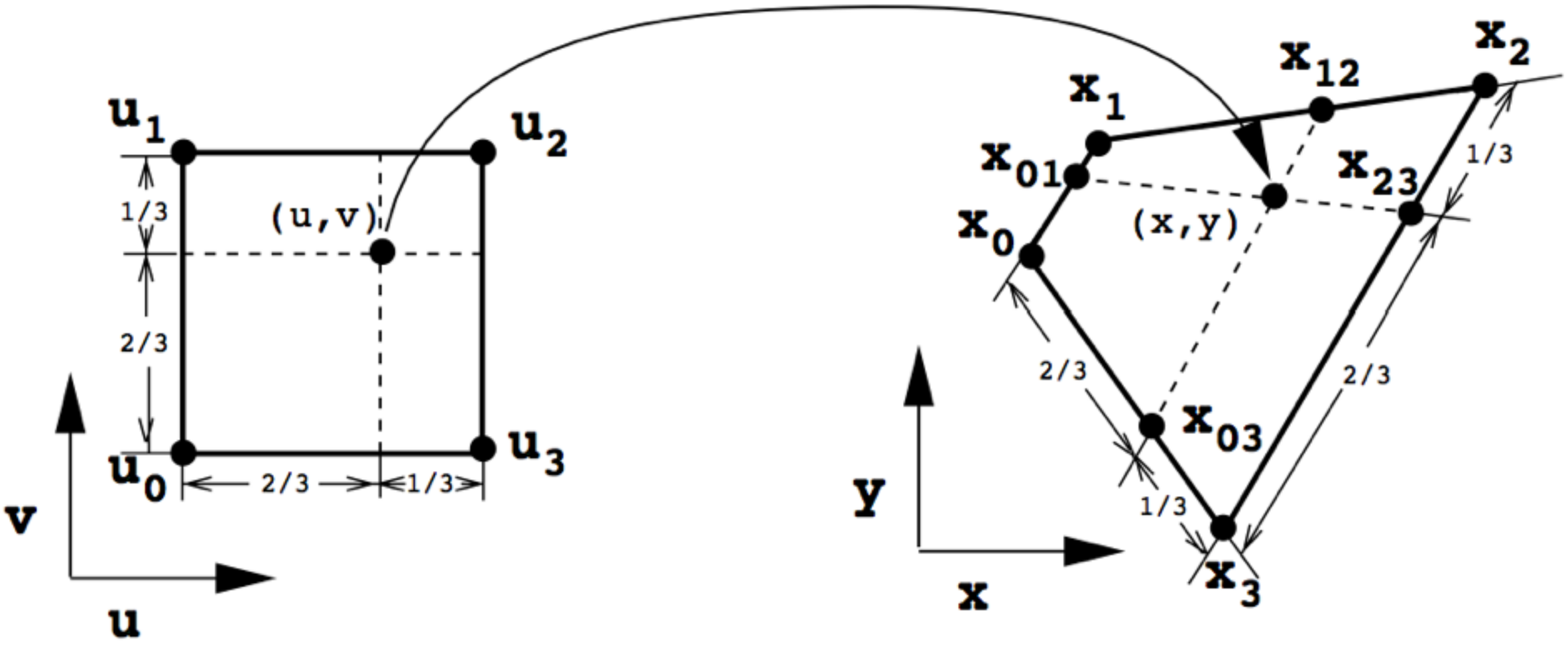


Figure 10.5: Establishing Correspondence Between Corners

Step 2: Forward Warp with u,v offsets



Inverse Bilinear Warp Can Be Computed from the Forward Warp

- Forward warp for a pixel (s, t) is equivalent to the following equations:

$$\overline{x} = (\overline{x_{12}} - \overline{x_{03}})v + \overline{x_{03}}$$

$$\overline{x_{03}} = (\overline{x_3} - \overline{x_0})u + \overline{x_0}$$

$$\overline{x_{12}} = (\overline{x_2} - \overline{x_1})u + \overline{x_1}$$

- where (s_0, t_0) is the lower left of the image, (s_1, t_1) is the upper right of the image, and (effectively normalizing)

$$u = \frac{s - s_0}{s_1 - s_0} \qquad v = \frac{t - t_0}{t_1 - t_0}$$

Inverse Bilinear Warp Can Be Computed from the Forward Warp

- Substituting x_{03} and x_{12} in

$$\overline{x_{03}} = (\overline{x_3} - \overline{x_0})u + \overline{x_0}$$

$$\overline{x_{12}} = (\overline{x_2} - \overline{x_1})u + \overline{x_1}$$

$$\overline{x} = (\overline{x_{12}} - \overline{x_{03}})v + \overline{x_{03}}$$

- And replacing $a_0=x_0$, $a_1=x_3-x_0$, $a_2=x_1-x_0$, $a_3=x_2-x_1-x_3+x_0$, $b_0=y_0$, $b_1=y_3-y_0$, $b_2=y_1-y_0$, $b_3=y_2-y_1-y_3+y_0$

- Leads to the bilinear form of the forward map:

$$x = a_0 + a_1u + a_2v + a_3uv$$

$$y = b_0 + b_1u + b_2v + b_3uv,$$

Inverse Bilinear Warp Can Be Computed from the Forward Warp

- Taking the inverse of

$$x = a_0 + a_1 u + a_2 v + a_3 uv$$

$$y = b_0 + b_1 u + b_2 v + b_3 uv,$$

- Leads to

$$v = \frac{-c_1}{2c_2} \pm \frac{1}{2c_2}\sqrt{{c_1}^2 - 4c_2 c_0}$$

$$u = \frac{x - a_0 - a_2 v}{a_1 + a_3 v},$$

- Where 0<u<1, 0<v<1, and

$$c_0 = a_1(b_0 - y) + b_1(x - a_0),$$

$$c_1 = a_3(b_0 - y) + b_3(x - a_0) + a_1 b_2 - a_2 b_1,$$

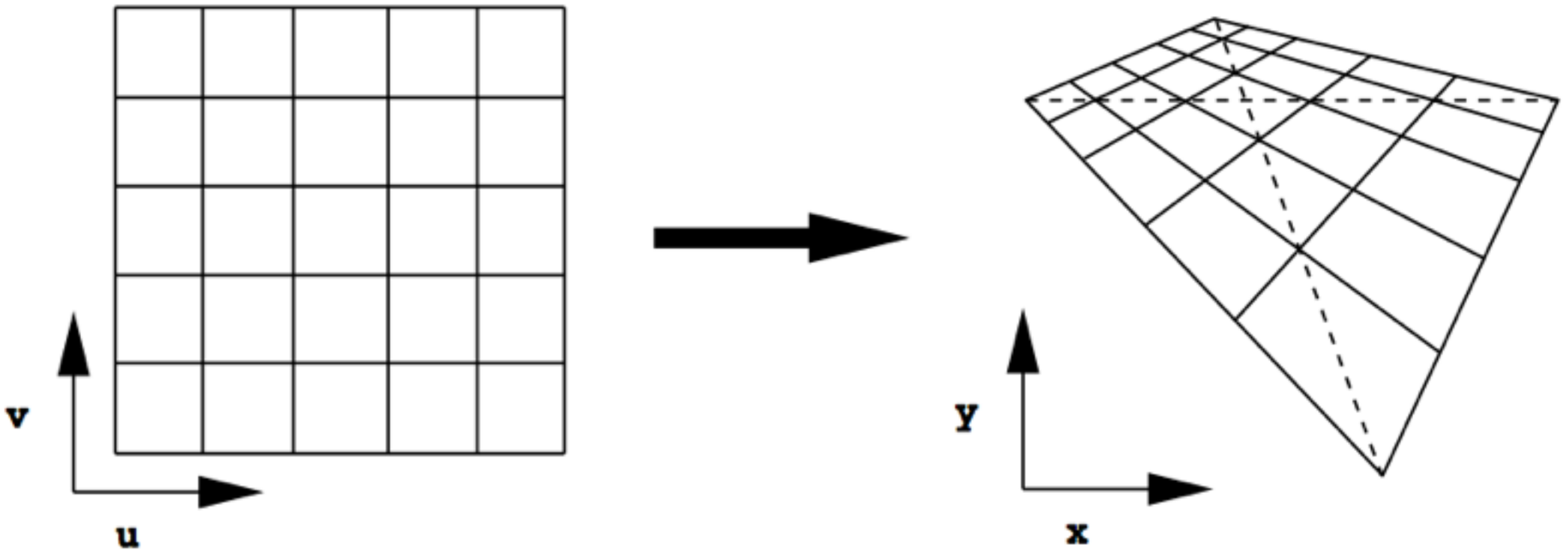
$$c_2 = a_3 b_2 - a_2 b_3.$$

What Did We Learn?

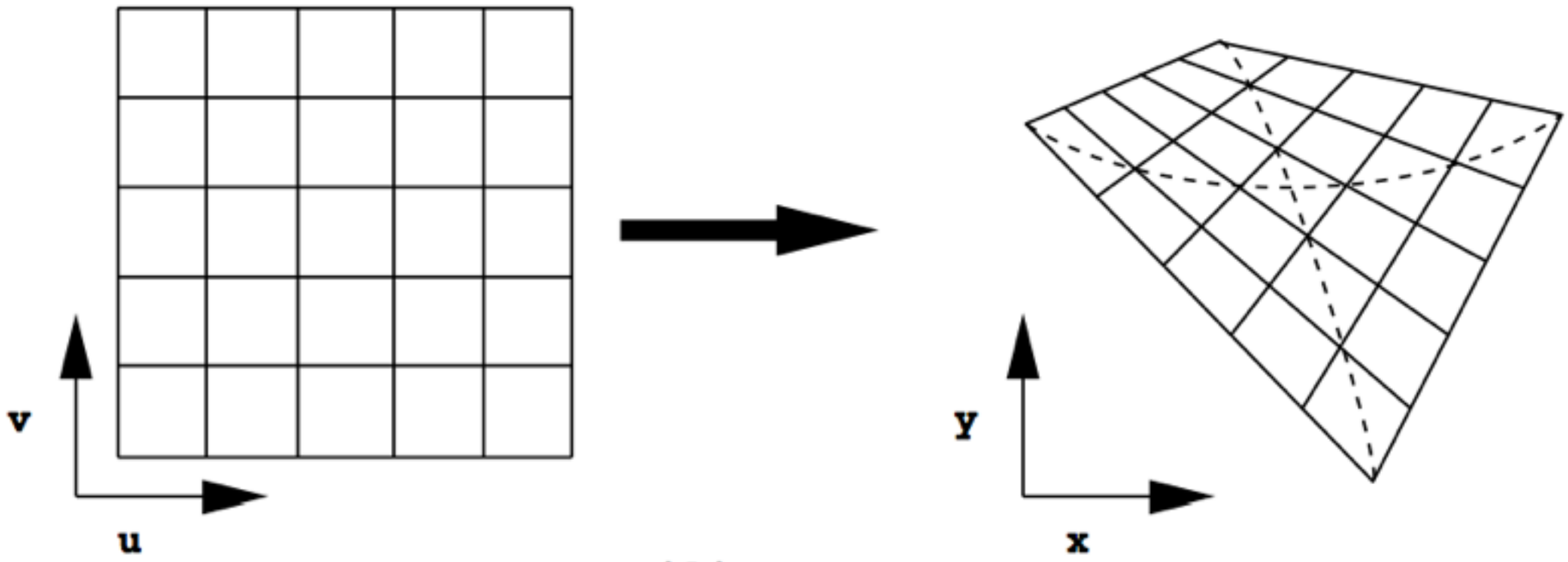
- This exercise shows there is an inverse
- While bilinear warps seem superior in terms of spacing, this spacing is gained by bending the middle of the image.
- Things necessarily get squished (there is no free lunch)
- The equations (the uv term in the bilinear form) shows how

$$x = a_0 + a_1 u + a_2 v + a_3 uv$$

$$y = b_0 + b_1 u + b_2 v + b_3 uv,$$



Perspective Warp



Bilinear Warp

Removing Warping Artifacts

Causes of Warping Artifacts

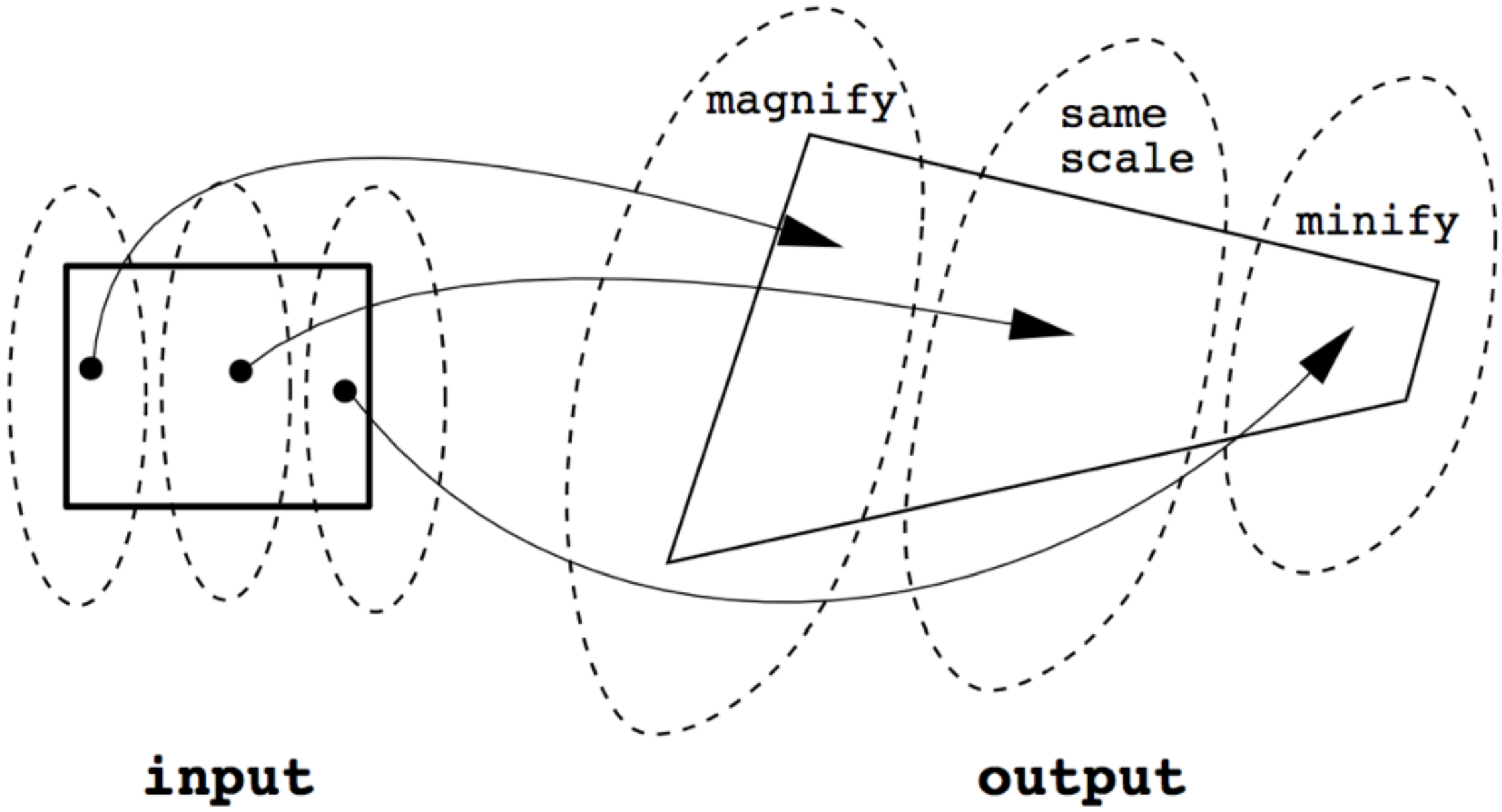


Figure 11.1: Magnification and Minification in the Same Warp

Recall: Types of Artifacts

- Even with inverse mapping, two big problems occur with warping

- Have to do with the fact we are resampling color information at a different rate.

- Two types:

  - **Aliasing**: sampling the input image too coarsely. Occurs when the image is being minified.

  - **Reconstruction**: sampling the input image too crudely. Occurs when the image is magnified.

Sampling and Reconstruction

Recall: Continuous vs. Discrete

- Key Idea: An image is either (both?) in the
  - Continuous domain: where light intensity is defined at every (infinitesimally small) point in some projection
  - Discrete domain, where intensity is defined only at a discretely sampled set of points.

Recall: Converting Between Image Domains

- When an image is acquired, an image is taken from some continuous domain to a discrete domain.
- **Reconstruction** converts digital back to continuous through interpolation.
- The reconstructed image can then be **resampled** and **quantized** back to the discrete domain.

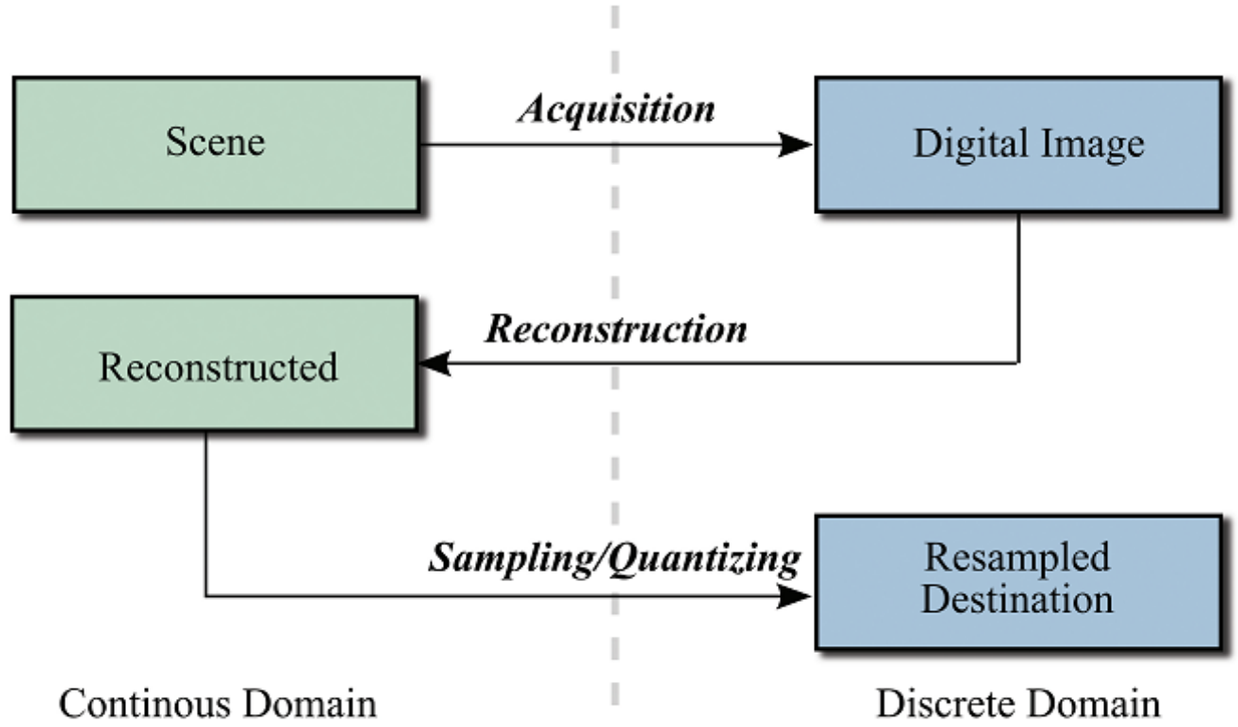


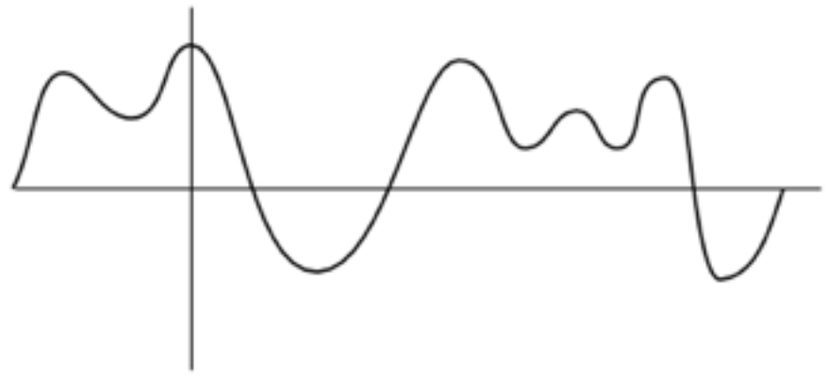
Figure 7.7. Resampling.

Digital Image Processing

- When implementing operations that move pixels, we must account for the fact that digital images are sampled versions of continuous ones

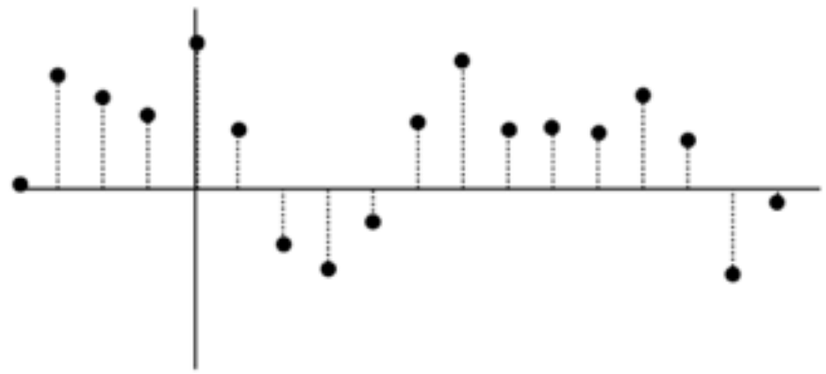
One Dimensional Example

Original



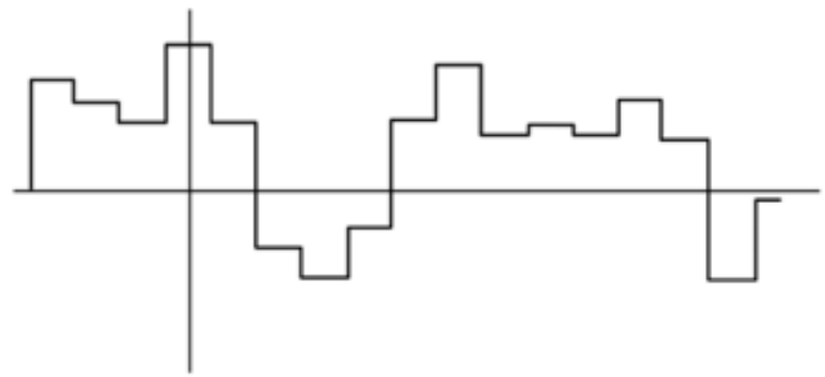
a) brightness along a scanline across the original scene

Sampled



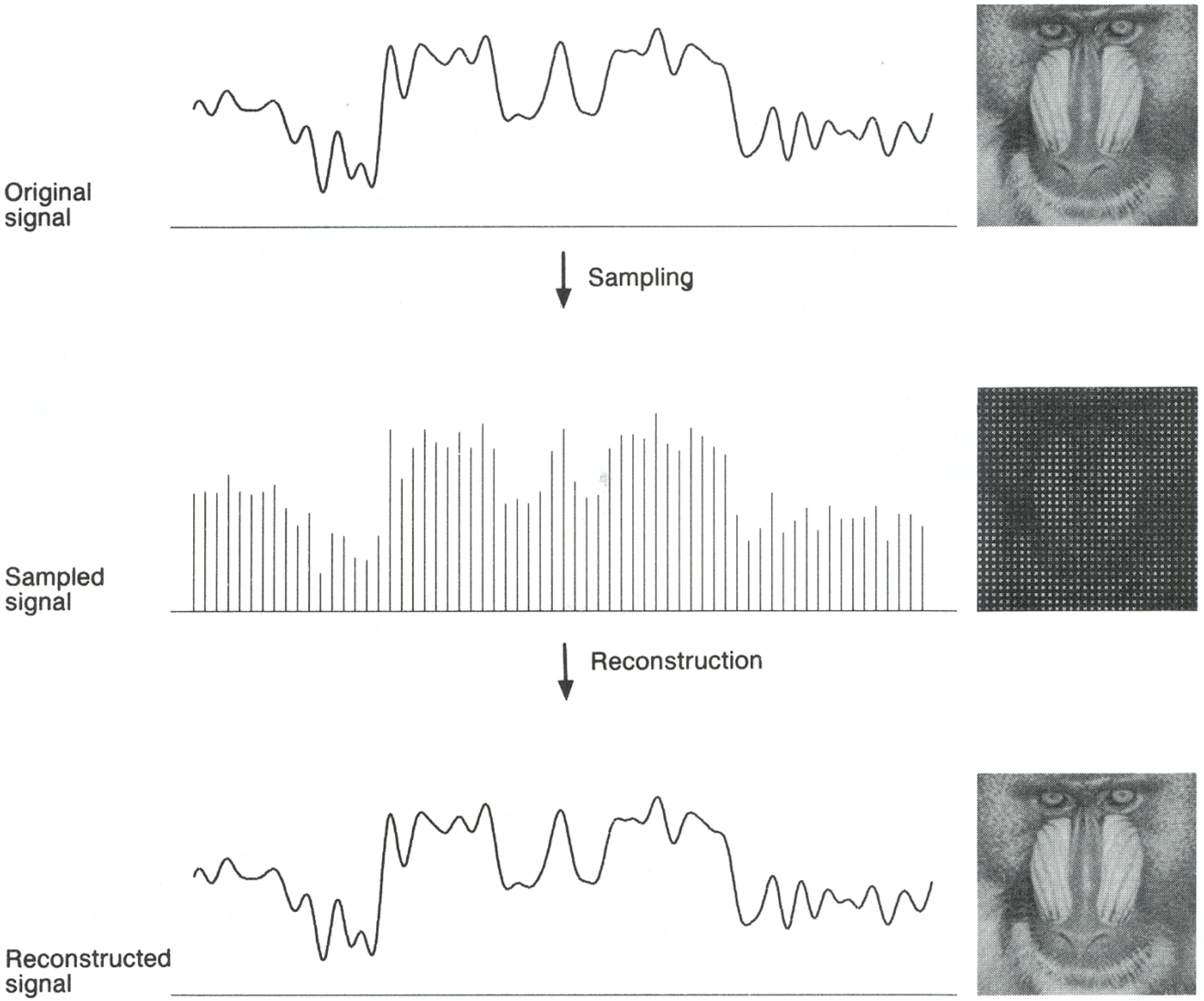
b) brightness samples along the same scanline

Reconstruction



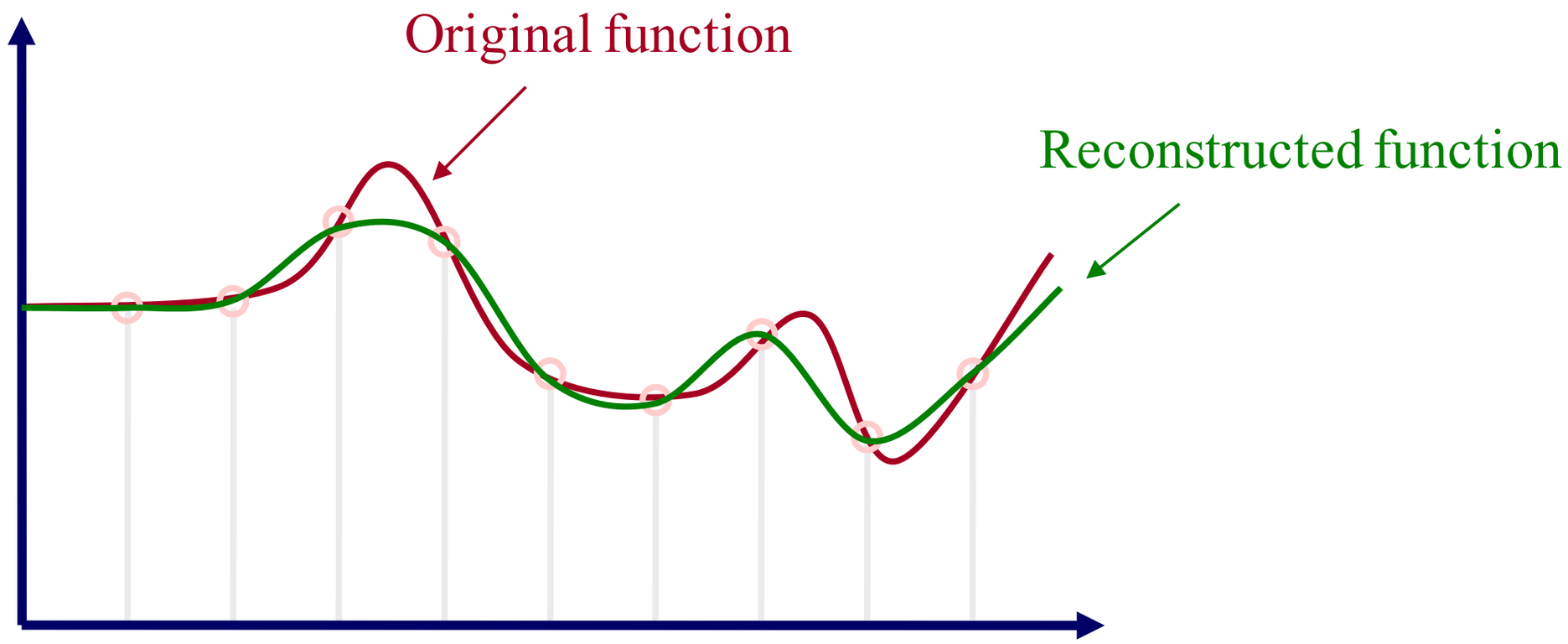
c) pixel-like reconstruction of original line from the samples

Sampling and Reconstruction



Sampling Theory

- How many samples are enough?
  - How many samples are required to represent a given signal without loss of information?
  - What signals can be reconstructed without loss for a given sampling rate?

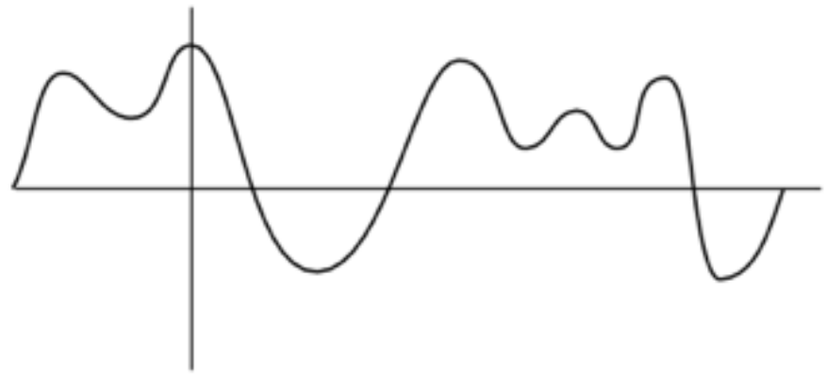


Recall: Fixing Warping Artifacts

- Fixing aliasing artifacts: By carefully smoothing the input first.
    - Idea: We know we are going to lose information samples, so choose the right information samples to lose.
- Fixing reconstruction artifacts: Do a better job of interpolation color values (instead of just nearest neighbor).
    - Idea: We know we are going to create information new samples, so try to use as much data from the image as possible.

One Dimensional Example

Original

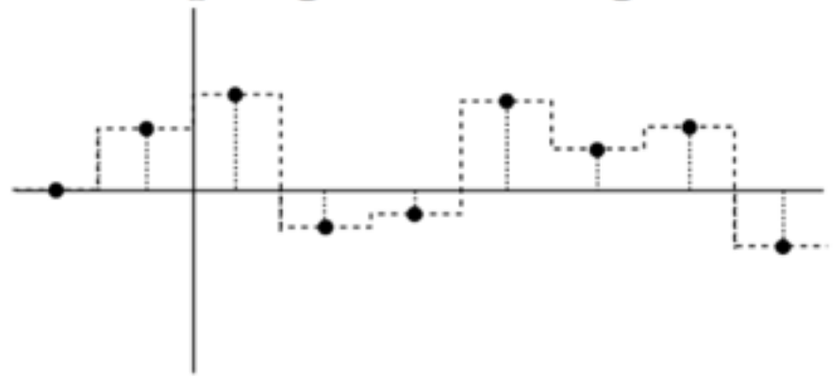


a) brightness along a scanline across the original scene

Resampled

d) resampling under magnification

Antialiased



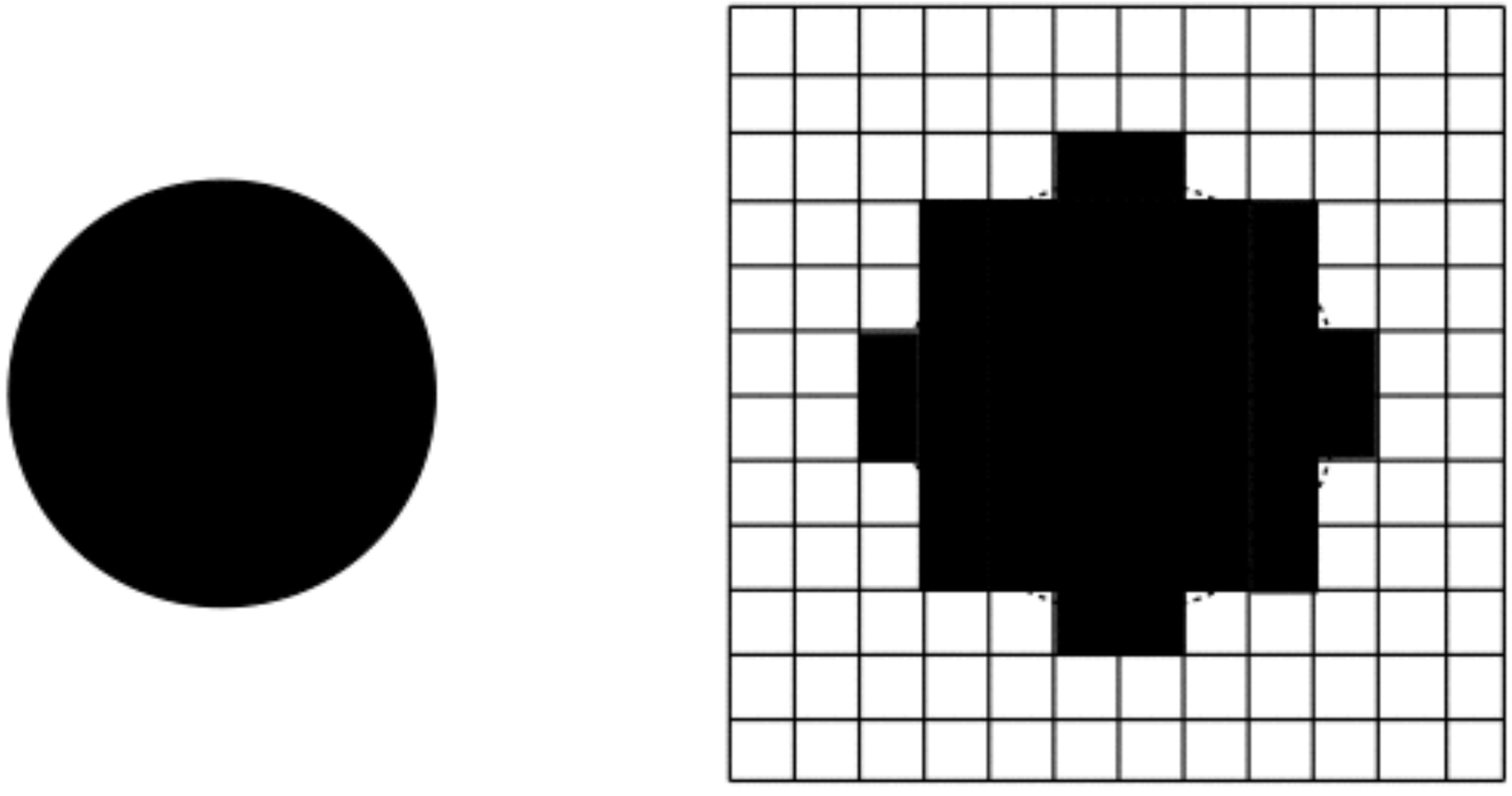
e) resampling under minification

Lesson 1: To reduce magnification artifacts we need to do a better job of reconstruction.

Fixing Jaggies / Magnification Artifacts

Reconstruction Artifacts

- Leads to staircasing or “jaggies”



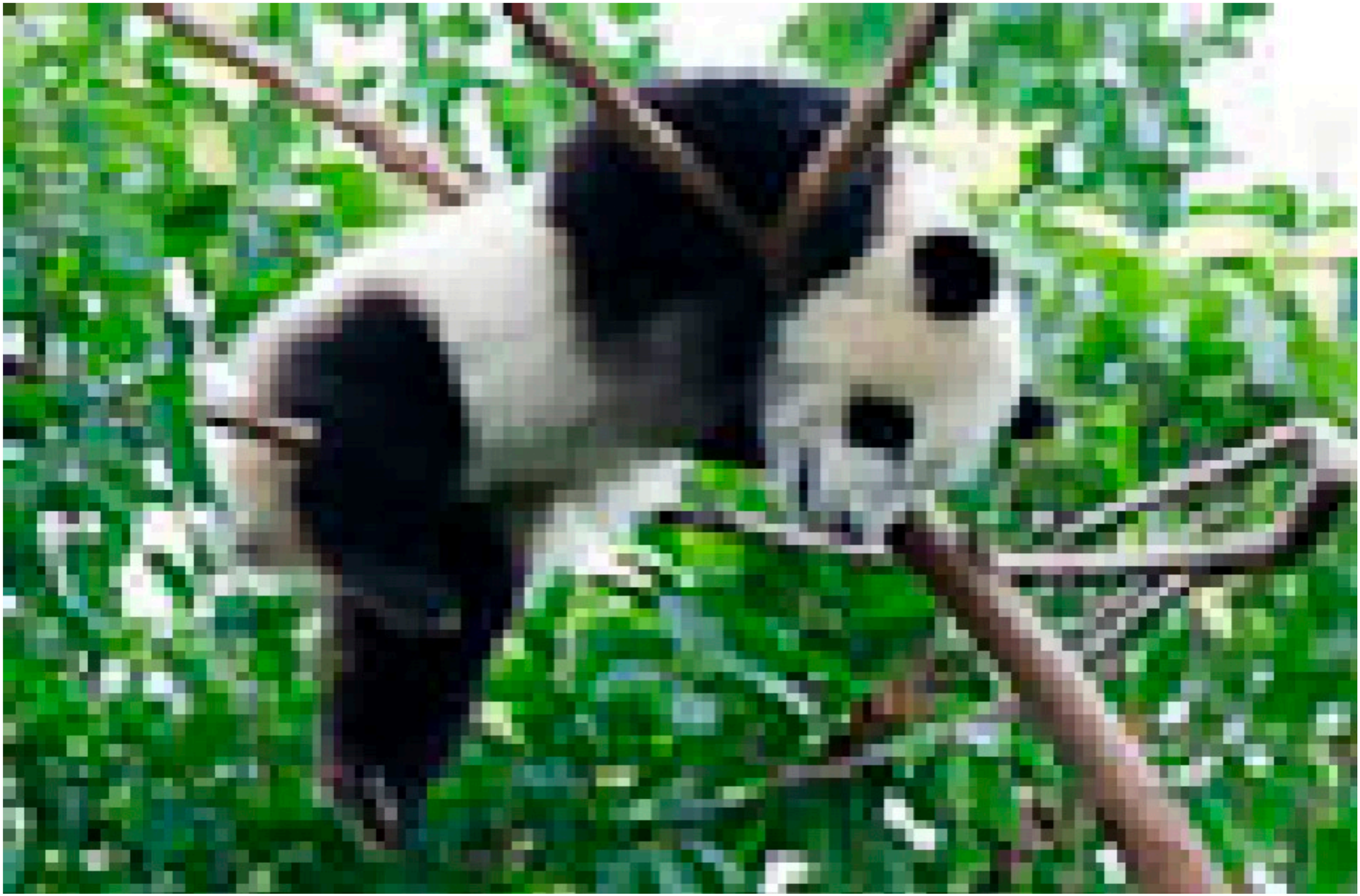
b) reconstruction artifacts

Do a Better Reconstruction?

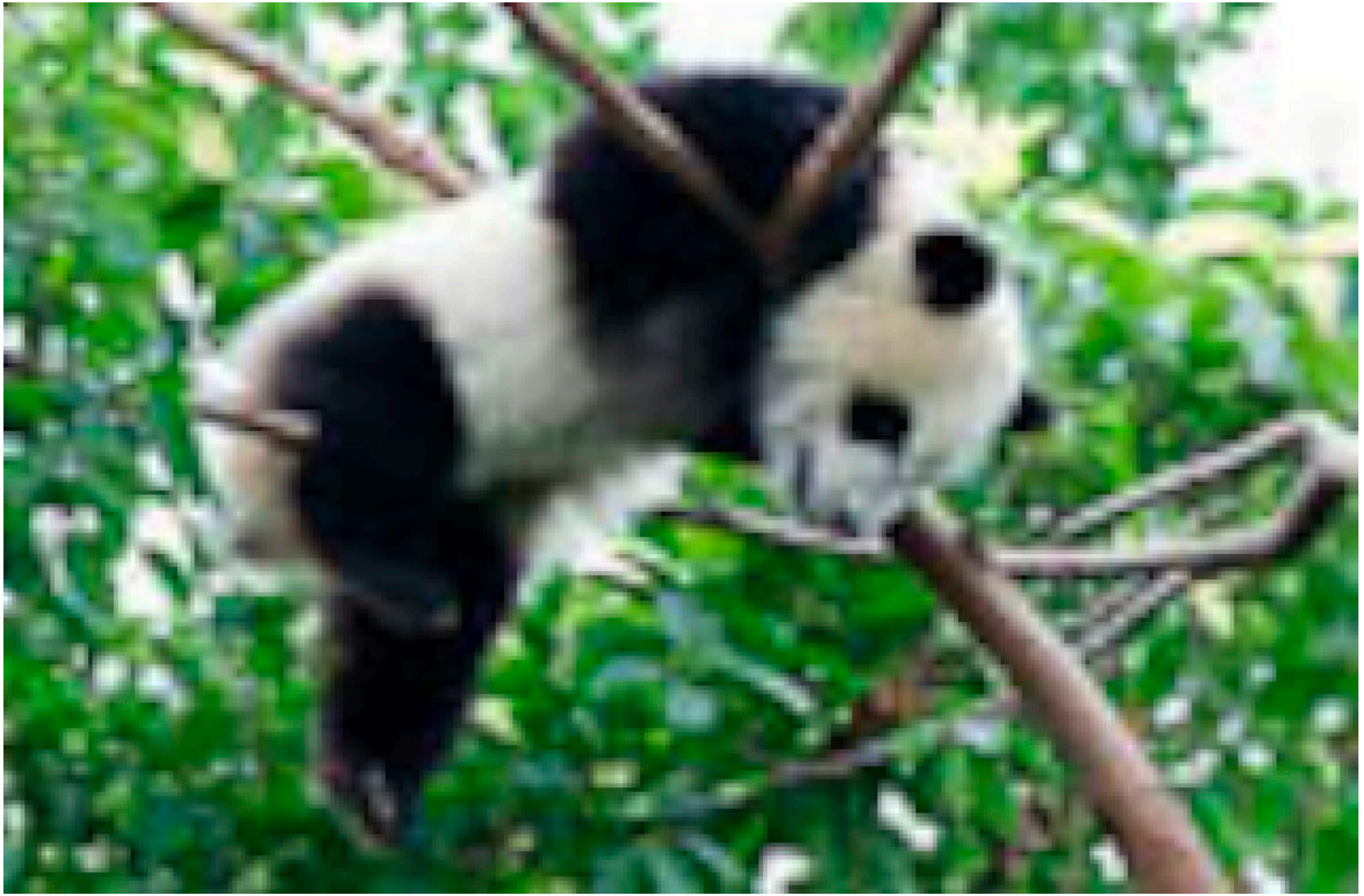
- Basic Idea: If we interpolate the data samples better we will have a superior reconstruction

- How? Bilinear Interpolation, Bicubic, etc.

Recall: Nearest Neighbor



Recall Bilinear Example



Recall Bicubic (from Photoshop)

Lec20 Required Reading

- House, Ch. 11