Computer Science 631
Lecture 3: Morphing, Sampling

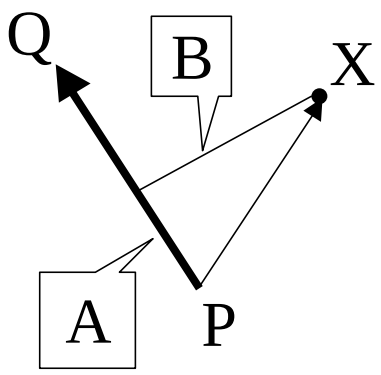
Ramin Zabih
Computer Science Department
CORNELL UNIVERSITY

Outline

- Ray-based coordinates recap
- Bilinear interpolation
- Rotations
- Multiple rays
- Aliasing and sampling

Ray-based coordinates

- Compute the position of the pixel X w.r.t. an oriented ray PQ
  - Coordinates are A (along PQ) and B (perpendicular to PQ)



$$A = \frac{(Q-P)}{\|Q-P\|} \cdot (X-P)$$

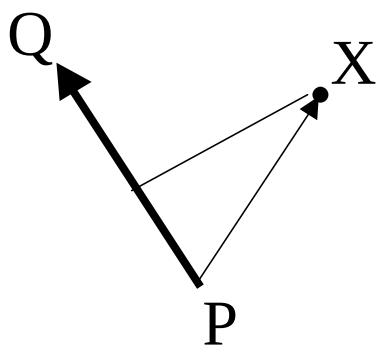
$$B = \frac{(Q-P)^{\perp}}{\|Q-P\|} \cdot (X-P)$$

Changing units

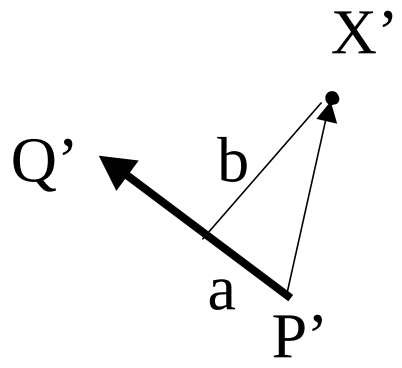
- The problem is that A and B are in units of pixels
  - Need them in percentages of the length of PQ

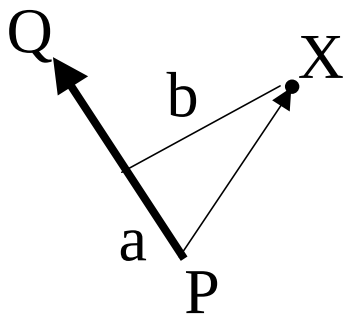
$$u = \frac{A}{\|Q-P\|} = \frac{(Q-P)}{\|Q-P\|^2} \cdot (X-P)$$

$$v = \frac{B}{\|Q-P\|} = \frac{(Q-P)^{\perp}}{\|Q-P\|^2} \cdot (X-P)$$



General formula



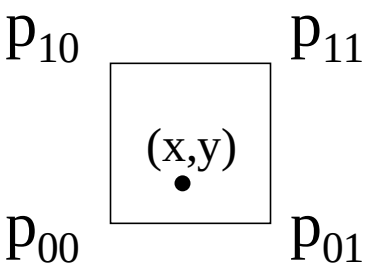


$$X' = P' + a \cdot (Q' - P') + b \cdot (Q' - P')^{\perp}$$

Note that X’ (as well as X) is a *point*, not a *pixel*

Bilinear interpolation

- We will need to estimate the intensity at a point, from data which is defined on pixels
  - Consider a 2-by-2 square of pixels
  - Let's see how to interpolate the value at some point in their midst



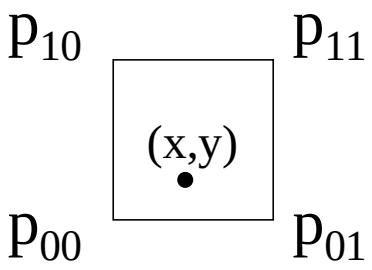
Obvious values where x or y are 0 or 1

Want to interpolate linearly in between

Fast bilinear interpolation

- The value at the interior point (x,y) is

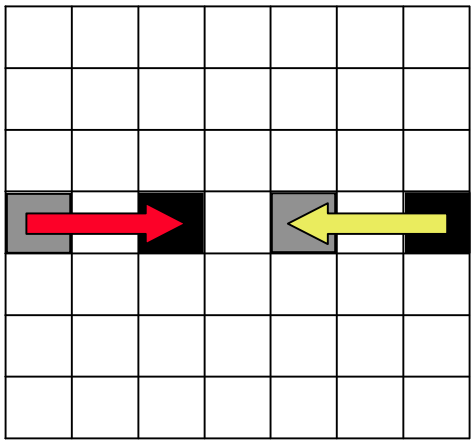
$$p_{xy} = (1-x)(1-y)\cdot p_{00} + x(1-y)\cdot p_{10} + (1-x)y\cdot p_{01} + xy\cdot p_{11}$$



- To compute this fast:

$$p_{x0} = p_{00} + x(p_{10} - p_{00})$$
$$p_{x1} = p_{01} + x(p_{11} - p_{01})$$
$$p_{xy} = p_{x0} + y(p_{x1} - p_{x0})$$

Subtlety: rotations



- What happens if we interpolate the endpoints?
  - In general the segments can get very small
  - There is no well-defined solution
  - Can interpolate center point, orientation, length

Subtlety: multiple rays

- Multiple rays are a necessity
  - Want to shrink the nose, but not the eyes
  - Kai's Power Goo is a nice example of this
- Multiple rays will in general give conflicting ideas about the right intensity
  - For a given point X, each ray will specify a point on the source image that X's intensity should come from
  - How to resolve?

Weighting with distance

- The closer X is to the ray PQ, the more PQ's opinion "matters"
  - We take a weighted average
  - Let X_i be the point that the ith ray believes that X's intensity should come from

$$X' = X + \frac{w_1 D_1 + w_2 D_2 + \cdots}{w_1 + w_2 + \cdots}$$

$$D_i = X_i - X$$

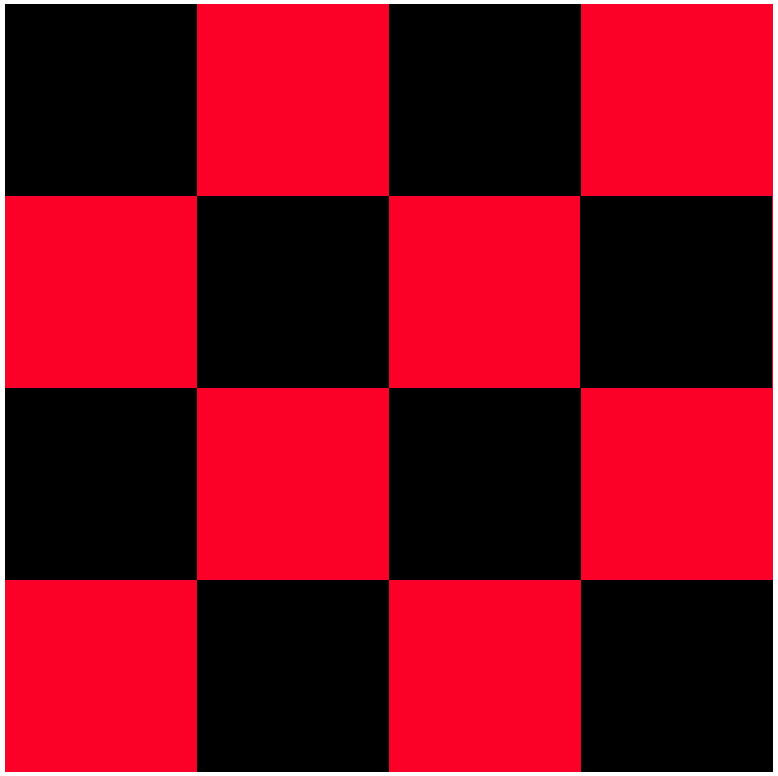
How to do the weighting?

- Beier and Neely use $w_i = \left(\frac{\|Q_i - P_i\|^p}{a + dist} \right)^b$
  - dist is the distance from X to this ray
  - a,b,p are constants that tune the function
    - If a=0, points near the line matter are hugely influenced by that line
    - If p=0, line length doesn't matter
    - b determines how fast weight decays with distance
  - In practice, p in in [0,1], b in [.5,2]

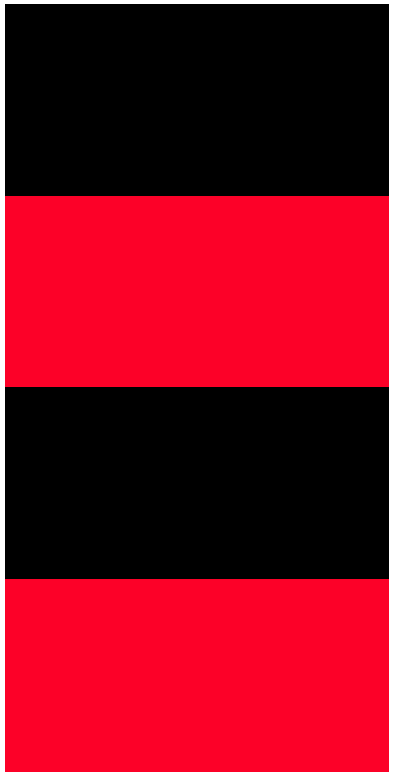
Subtlety (and next topic): aliasing

- Suppose that we shrink the input image
  - We will only examine a subset of input pixels!
  - This can introduce an artifact called *aliasing*
    - Patterns in the output that aren't in the input
  - Example: output is 1/2 width of input image
    - Destination scanning will ignore half the input pixels!
    - Only examine input pixels where *x* is even
    - Suppose that the input is a checkerboard, where every square is a single pixel

Mutilating a checkerboard

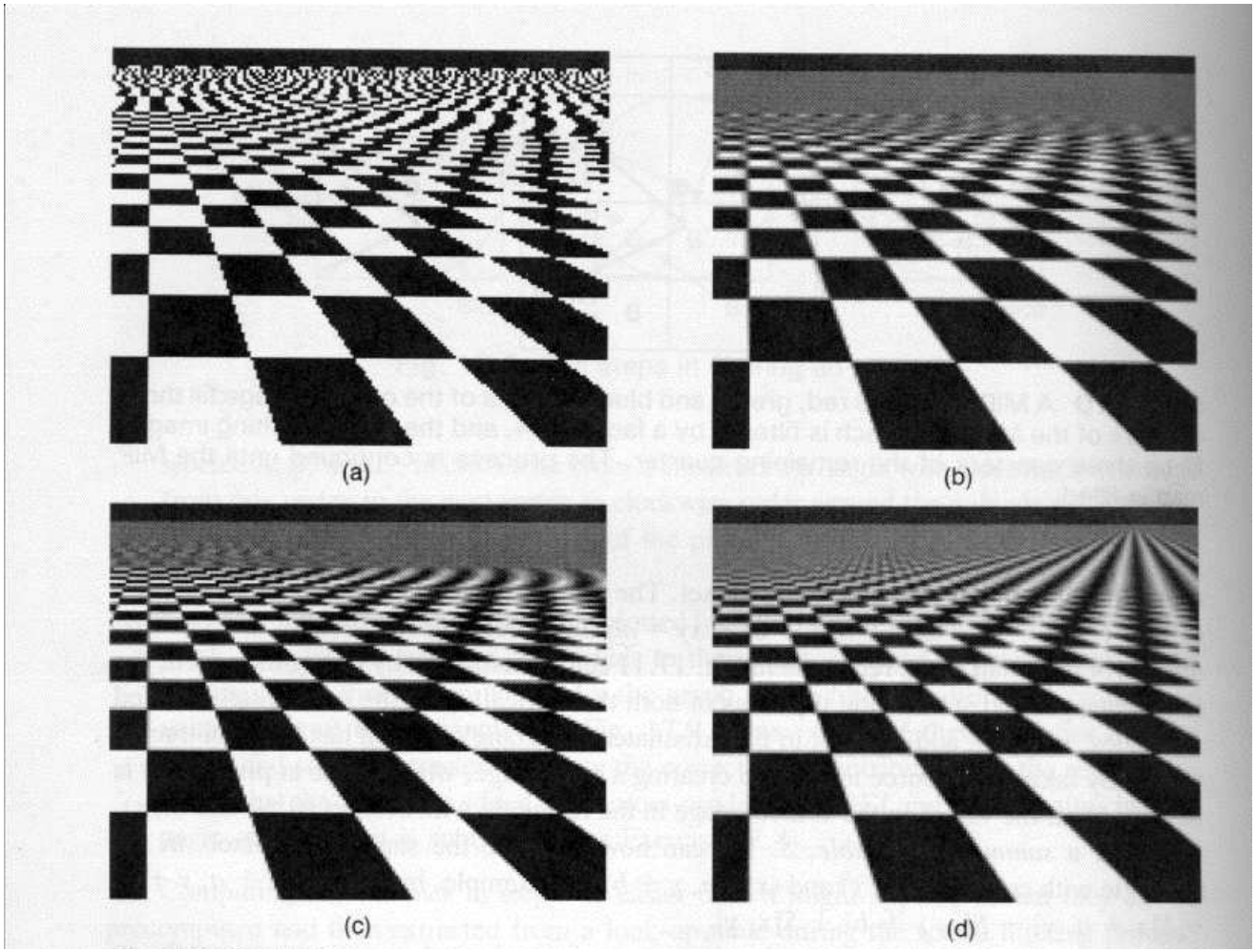


Input



Output

Aliasing example



Aliasing issues

- These issues can arise any time the image size changes (which is almost always)
- When is aliasing not a problem?
  - If the input image changes “slowly” relative to how much we shrink it, this isn’t a problem
    - Consider a black image, or a uniform ramp
    - Or we can “blur” the image (we’ll cover this)

How to think about aliasing

- There is a lot of material on this topic
  - DSP courses, especially EE302 or EE425
  - Any computer graphics course
- Basic idea: represent an image as a sum of parts that change “slowly” and parts that change “fast”
  - This is a change of basis from the standard representation in terms of pixels

Frequency decomposition of an image

- An image can be described in several ways
  - So far, in terms of pixels (= spatial domain)
  - The frequency decomposition is very useful
  - Low-frequency components change slowly
  - High-frequency components change rapidly
- If the image has no (little) high-frequency components, then aliasing is not a problem

Why use the frequency domain?

- By describing an image in terms of the frequency domain, many things become clear
  - The image formation process itself removes really high-frequency components
    - What happens when we take a picture of a checkerboard where a pixel contains 100 squares?
    - Many image operations are naturally viewed in terms of their effects on various frequency components
      - Local averaging removes high-frequency components
    - Image compression is best viewed in this way

Choice of basis

- The canonical way to describe the frequency of an image is in terms of its Fourier transform
  - This involves a number of issues that we don't have time to cover in depth
- Instead, we will use a *Wavelet* representation called the *Haar basis*
  - Same basic idea, but easier and more intuitive
  - For example, our basis vectors will be mostly 0
    - By contrast, the Fourier basis is sine waves

Image representations

- Consider a 1-D image (signal) with four elements: $I = [9\ 7\ 3\ 5]$
  - Spatial representation:

$I = 9*[1\ 0\ 0\ 0]+7*[0\ 1\ 0\ 0]+3*[0\ 0\ 1\ 0]+5*[0\ 0\ 0\ 1]$

  - The basis elements are 1-D (in this case) vectors, each with a single 1
    - What are they called for an image?

Wavelets and DC components

- Our basis vectors will have a *scale*, which intuitively means how many non-zero elements
  - To begin with, we will subtract the average value of *I*, which is 6 in our example

    I - 6 = [3 1 -3 -1] I = 6[1 1 1 1] + [3 1 -3 -1]*
  - [1 1 1 1] is our first (dull) new basis
    - No zeros, so coarsest possible scale
  - The average value of an image is referred to as its *DC component*, others are *AC components*