Computer Science 631
Lecture 3: Morphing, Sampling

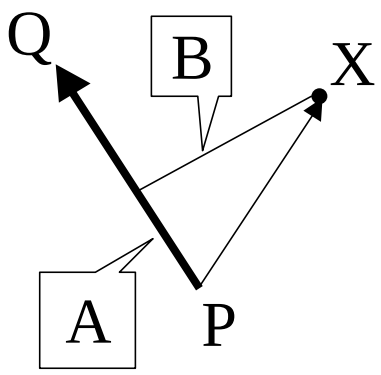
Ramin Zabih
Computer Science Department
CORNELL UNIVERSITY

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)



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

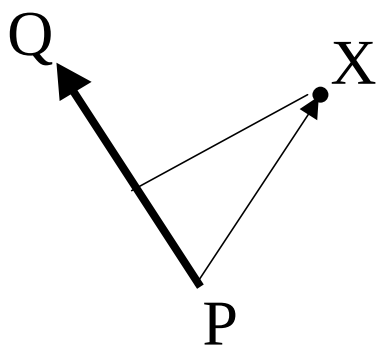
$$B = \frac{(Q-P)^{\perp}}{\|Q-P\|} \cdot (X-P)$$

Changing units

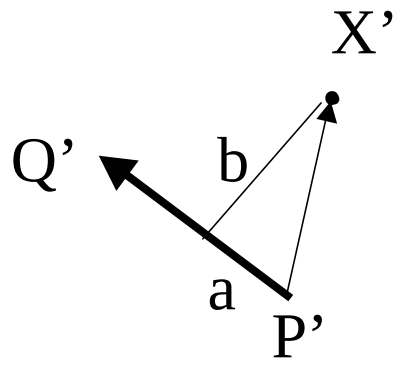
- The problem is that A and B are in units of pixels
  - Need them in percentages of the length of PQ

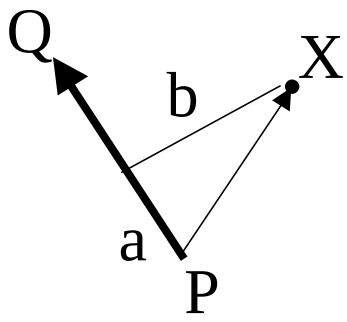
$$u = \frac{A}{\|Q-P\|} = \frac{(Q-P)}{\|Q-P\|^2} \cdot (X-P)$$

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



General formula



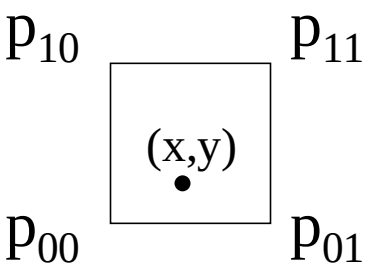


$$X' = P' + a \cdot (Q' - P') + b \cdot (Q' - P')^{\perp}$$

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



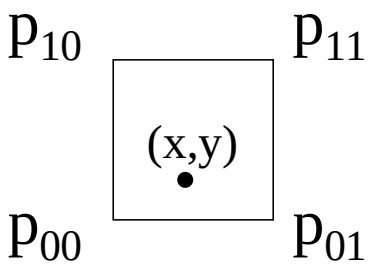
Obvious values where x or y are 0 or 1

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:

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

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*

AC components (detail coefficients)

- The average of two numbers is halfway between them
  - Therefore, a pair *[a b]* can be represented as:

  avg[1 1] + diff*[1 -1]*, where

  avg = (a + b)/2 and *diff = a - avg = avg - b*
  - For the Haar basis, we will use this trick
    - For a pair of numbers, compute the average and difference

Multi-resolution averaging

- Given a signal of size 2^n, we can average the pixels together at various scales
  - Compute a pairwise average, to get size 2^{n-1}
  - Compute a pairwise average on that, too
  - Eventually you get a single number
- Example (n = 2)

scale	average
4	[9 7 3 5]
2	[8 4]
1	[6]

Generating detail coefficients

- Each time we compute the average, we can write down the difference too

scale	average	detail
4	[9 7 3 5]	
2	[8 4]	[1 -1]
1	[6]	[2]

We can reconstruct [8 4] from [6] and [2]

6[1 1 1 1]+2*[1 1 -1 -1]+1*[1 -1 0 0] +-1*[0 0 1 -1]*

Wavelet properties

- We end up with a DC component plus a bunch of details, going from coarse scale to fine scale