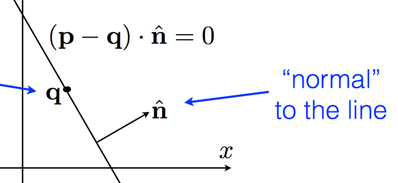
**CS 355 Midterm Notes:**

Raster Images:

Size: number of pixels (height X width)

Bit Depth: bits of precision for each pixel

2-bit gray – 4 shades

8-bit gray - 256 shades

12-bit – X-rays / CT

24-bit color – 8 bits each of R, G, B

Interlacing: Send half screen 60X/second

Color Buffers are faster with Word aligning, so we fill the last slot with ALPHA!

Double Buffering:

Draw to offscreen buffer, then copy buffer

Circles:

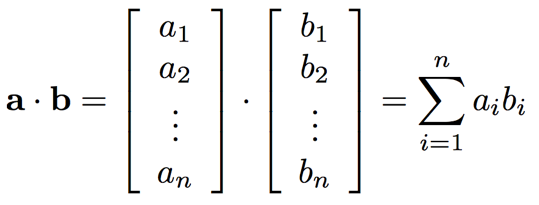
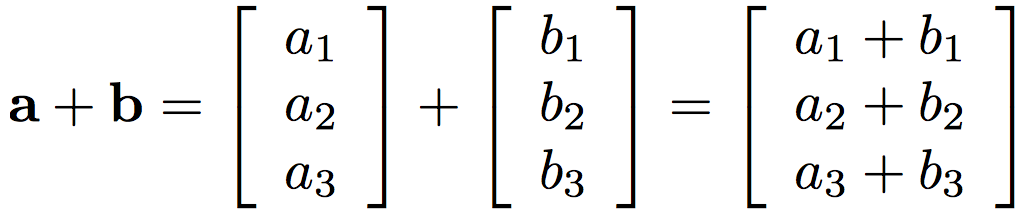
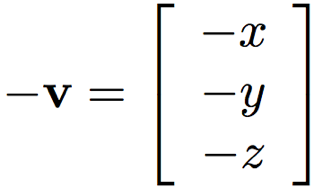
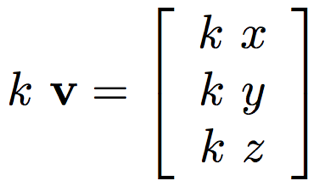
X­­­­2 + Y2 = r2

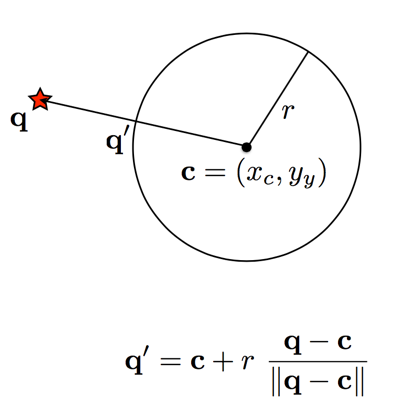
Ellipses:

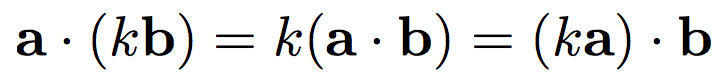
(X/a)2 + (Y/b)2 = r2

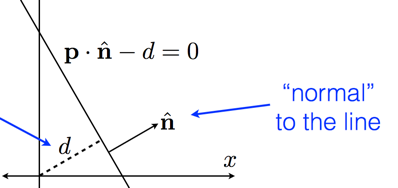
100% saturation is pure color. No white, black, or gray.

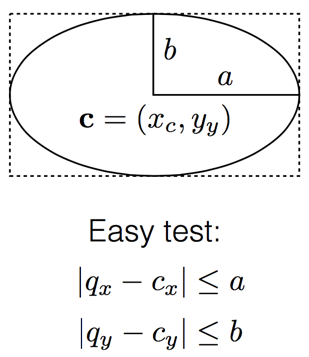
Vector is a directional quantity without a specific location. Direction, and Magnitude (length, norm).

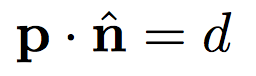
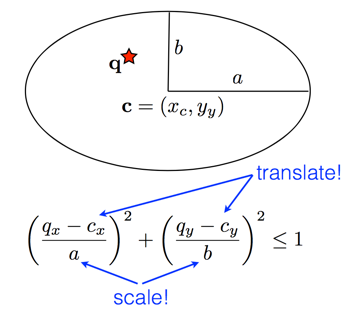


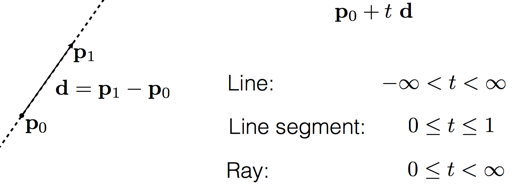
Vectors whose dot product is zero are said to be orthogonal



Implicit:



Normal and distance:

Implicit: Parametric:

World Space:

* Choice of origin and cooridinate system is arbitrary
* Where objects live

Object Space

* Coordinate system used to define objects
* Also arbitrary, but make it simple

Giving an object a Location, Orientation, and Size is what places it into world space (object to world)

Giving the world a location, orientation and size define a world to view transformation

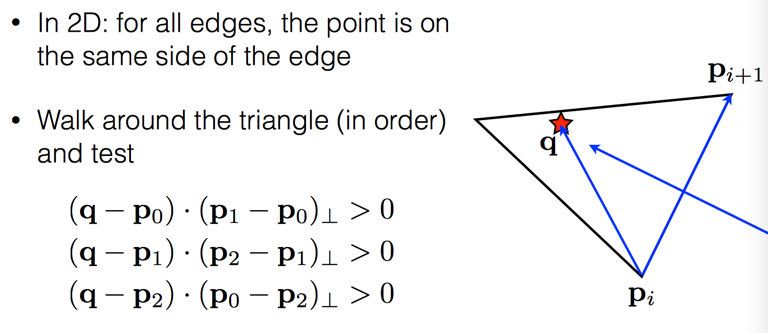
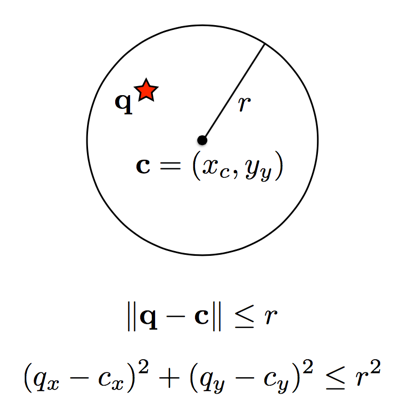
--Object Coord 🡪 World Coord 🡪 View Coord

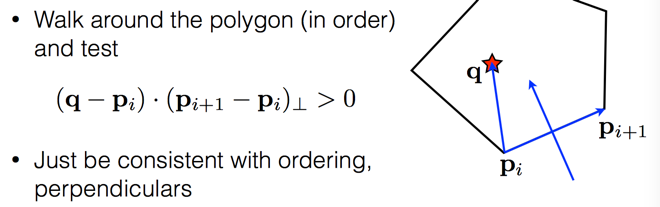
In object to world, First Rotate, Then Translate!

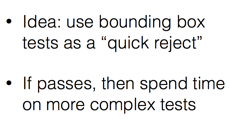
In world to object, First Translate, Then Rotate!

Selection:

Before testing a click, convert to the appropriate space. Screen to world, world to object. Test in object space.

To get a perpendicular vector, swap x and y and negate one of the two.





Texture Mapping:

* Mapping an image onto the surface of geometric model
* Requires image warping

Image Morphing:

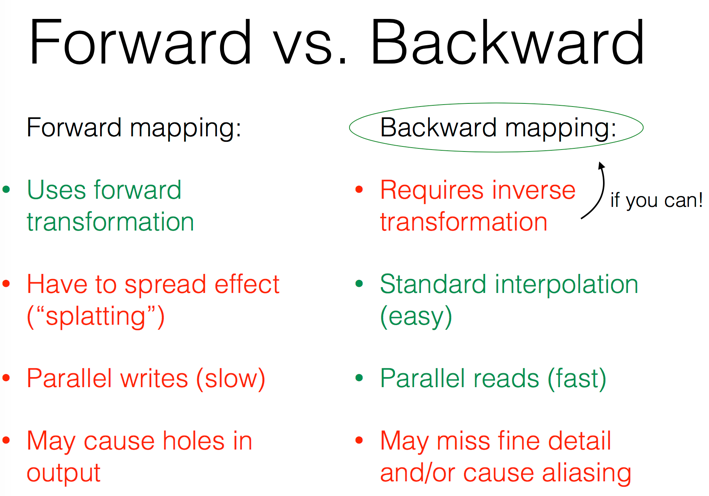
* Combination of warp and cross-dissolve

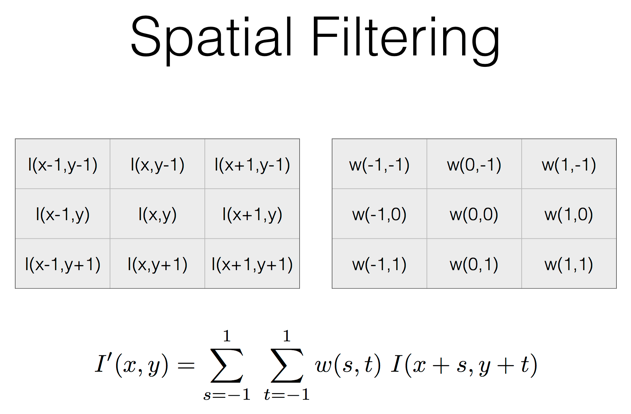
Forward Mapping:

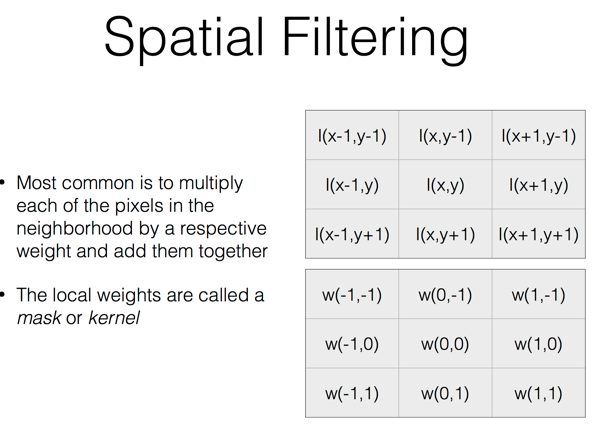
* Doesn’t map to discrete pixel locations
* May produce holes in the output

Backward Mapping:

* Doesn’t map FROM discrete pixel locations
* May miss things in the input



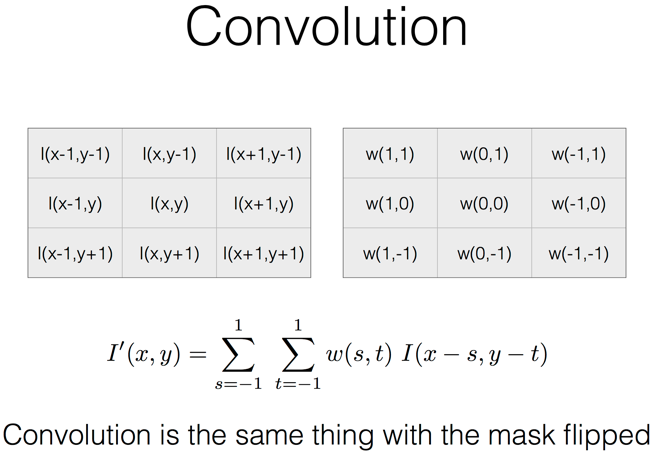




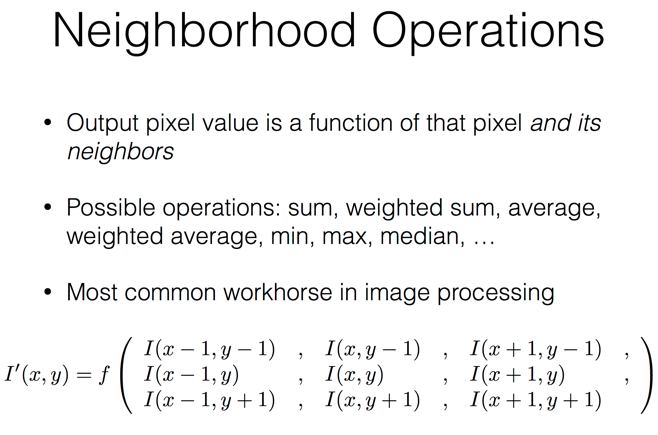
Repeated Warping:

* Spatial transformation requires interpolation which means some error (blurring, jaggies)
* Error will accumulate through repeated transformations

Spatial Filtering:

* What to do outside the image boundaries?
  + Assume Zero (tends to darken)
  + Assume other constant value
  + Wrap around
  + Assume same as closest still in image
  + Or just don’t go there
* Applications
  + Blurring, sharpening, edge detection…

Smoothing:

* Average multiple pixels
* Same as using a larger aperture
* Reduces noise
* Causes blurring
* Any kernel with all positive weights does smoothing/blurring
* Divide by the sum of the weights
* Can be any size, larger = blurrier

Nonlinear Smoothing:

* Spatial is linear, but many operators are not
* Some do noise reduction
  + Trimmed mean, median filter, bilateral

Median Filtering:

* Great for salt and pepper noise
* Respects edges
* Rounds corners

Bilateral Filtering:

* Closer neighbors get more weight
* More similar neighbors get more weight
* Most popular idea
* More computationally expensive

Anisotropic Diffusion:

* Interatively diffuse (blur) based on neighbor similarity

Edge Detection:

* Find strong changes using image derivatives