ISTA 352

Lecture 8

Finish tutorial on linear algebra and Digital representation of image data

Administrivia

- HW one due tonight
- HW two now posted
- Today includes some material needed for HW two
- Bonus demo today in GS 906 at 1:00

Linear functions (quick recap)

- The usual compact formula defining a linear transformation

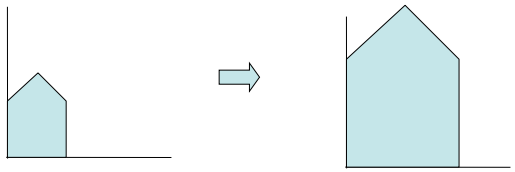
$$f\left(a*\mathbf{L_1}+b*\mathbf{L_2}\right)=a*f\left(\mathbf{L_1}\right)+b*f\left(\mathbf{L_2}\right)$$

- Matrix multiplication implements linear functions on vectors
- Geometrically, linear functions map lines to lines
- Associativity of matrix multiplication means that we can construct complex linear transformations from sequences of elementary ones

Review

Transformation examples in 2D

- Scale (stretch) by a factor of k



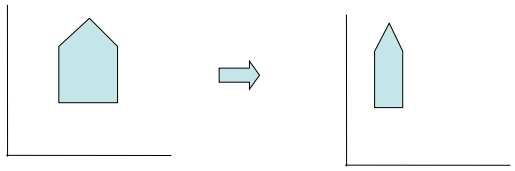
$$M = \begin{vmatrix} k & 0 \\ 0 & k \end{vmatrix}$$

(k = 2 in the example)

Review

Transformation examples in 2D

- Scale by a factor of (S_x, S_y)



$$M = \begin{vmatrix} S_x & 0 \\ 0 & S_y \end{vmatrix}$$

(Above, $S_x = 1/2$, $S_y = 1$)

Review

Orthogonal Transformations

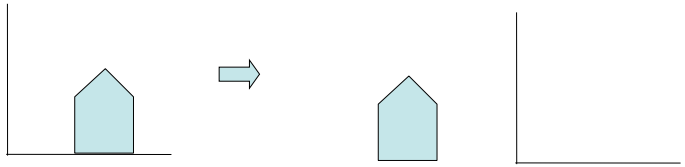
- Orthogonal transformations are defined by $O^T O = I$
- Also have $|\det(O)| = 1$ (*)
- Rigid body rotations and mirror "flip"

* If you are not familiar with determinants, do not worry about it. We will not be using them in this course.

Review

Transformation examples in 2D

- Mirror flip through y axis (Orthogonal)



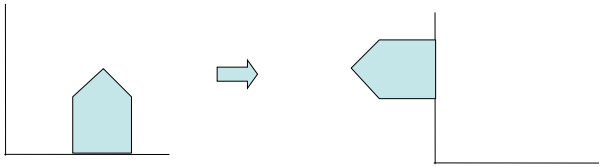
$$M = \begin{vmatrix} -1 & 0 \\ 0 & 1 \end{vmatrix}$$

This looks like a rotation out of the page, but it is actually a bit different because which side is facing you changes in the one case but not the other.

Review

Transformation examples in 2D

- Rotate around origin by θ (Orthogonal)

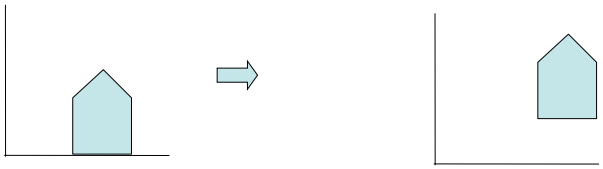


$$M = \begin{vmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{vmatrix}$$

(Above, $\theta = 90^o$)

2D Transformations

- Translation ($\mathbf{P}_{new} = \mathbf{P} + \mathbf{T}$)



M =?

Sneaky trick question!

Recall that translation is not linear!

There is no 2x2 matrix that works!

Homogenous Coordinates

- Represent 2D points by 3D vectors
  - This representation is called homogeneous coordinates
- To create a 3D vector from a 2D point we add "1" as the third coordinate
  - (x,y)-->(x,y,1)
- To convert homogeneous to regular coordinates
  - (x,y,W)-->(x/W, y/W, 1) (and ignore the last coordinate)
- Note that a multitude of 3D points (x,y,W) represent the same 2D point
- Homogeneous coordinates for 3D points work analogously
  - (x,y,z)-->(x,y,z,1)
  - (x,y,z,W)-->(x/W, y/W, z/W, 1)

Homogenous Coordinates (2)

- Basic transformations are easy to derive (examples coming up)
- We can implement translation by matrix multiplication
- More importantly, we can also implement represent perspective projection matrix multiplication
- This means all operations needed for image construction can be chained together into one matrix (recall that matrix multiplication is associative)

2D Scale in H.C.

$$M = \begin{vmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

2D Rotation in H.C.

$$M = \begin{vmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

2D Translation in H.C.

- $\mathbf{P}_{new} = \mathbf{P} + \mathbf{T}$
- $(x', y') = (x, y) + (t_x, t_y)$

$$M = \begin{vmatrix} ? \end{vmatrix}$$

2D Translation in H.C.

- $\mathbf{P}_{new} = \mathbf{P} + \mathbf{T}$
- $(x', y') = (x, y) + (t_x, t_y)$

$$M = \begin{vmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{vmatrix}$$

Digital representation of images

Vector vs raster representation

- Raster representation of an image (very familiar)
  - Array of values indexed by spatial coordinates
    - image(i,j) = intensity encoding (BW/gray scale)
    - image(i,j) = color encoding, e.g, (R,G,B)
  - Makes sense as an encoding of sensor array data
    - Digital cameras, scanned photos
    - Suffers quantization errors, fixed "true" resolution
    - Resolution can be lost during transformation

Vector vs raster representation

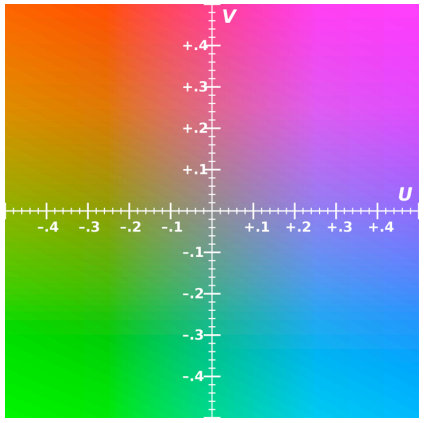
- Vector representation
  - Prescriptive code to generate images
  - Examples
    - A line segment with endpoints (x1,y1) and (x2, y2)
    - Text string to insert at a particular location
  - Can be very compact
  - Resolution is not an issue as image is generated from potentially very accurate data as needed
  - Natural and effective for computer generated material
  - Examples
    - Common modern example is a PDF file (except for embedded raster encoded material)
    - Historical example is the very earliest days of graphics where data to displays was in the form of line segments

How many bits for color images

- 3x256 bits is close to enough
  - Provisos
    - Using a good non-linear transfer function (gamma)
    - Three channels does not quite provide full color fidelity
      - This is not a "number of bits issue"
      - We will understand this ... later!
- Definition of enough
  - If, under the optimal transfer functions, any two colors are barely distinguishable then adding more colors resolution will not help
  - Humans can distinguish about 10 million colors (ignoring brightness) under laboratory conditions
  - So, 2^(3*8) is in the right ball park

Aside on color versus chromaticity

- Color consists of brightness and chromaticity
- In a 3D color space, chromaticity is 2D

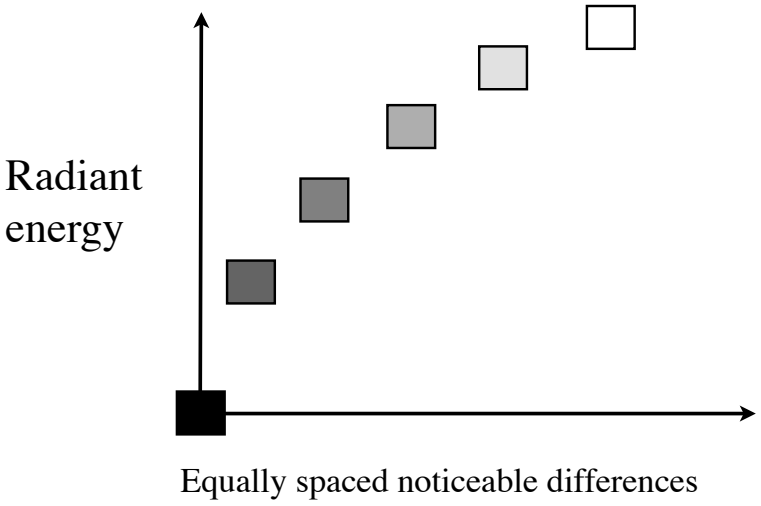


Aside on color versus chromaticity

- Color consists of brightness and chromaticity
- In a 3D color space, chromaticity is 2D
- Brightness is a special dimension in a color space
  - The range in nature is very large
  - Imaging systems adjust the range by controlling the aperture and other means
  - Displays cannot reproduce the range of brightness that our visual system can distinguish (within a given range)
- If we want to store captured colors with a large brightness range, especially if we want purely linear data, more than 8 bits per channel can make sense
  - Typical data format is 16 bit tiff or floating point formats

Concepts to keep separate

- Color range that can be represented or displayed
  - Later we will learn that you can see colors that a typical RGB system does not represent and typical monitors cannot display
- Color resolution (number of colors)
  - Adequate number is driven by "just noticeable differences"
- Optimum step size between neighboring colors
  - As colors become brighter, the energy difference between just noticeable differences decreases.
  - This justifies keeping "gamma" where physical meaning of the values goes through a non-linear function.



Pure vs indexed color

- 3x256 bits used to be a lot when memory and disk space was expensive
- Indexed color works with a color "palette" of a limited number of colors
- Each array element is instead an index into the color map
  - For example, if our palette is 256 colors, then we use one byte per pixel instead of three.
  - 16 colors cuts the memory footprint down to 4 bits per pixel
- Now that memory/disk is cheap should one care?
  - The concept of a color map persists in many places