2D Geometry and Transformations

CS 417 Lecture 7

Introduction

- So far: Imaging
- Next: Modeling
  - the study of how to define and manipulate the shapes and arrangements of objects
- We'll start with the 2D case, then later move to the more complex 3D case

Mathematical preliminaries

- Vector spaces
- Linear transformations and matrix multiplication
- 2D affine space (Euclidean space)

2D Vector spaces

- Vector
  - an arrow drawn in the plane
  - an offset (position not important)
- Vector operations
  - addition
  - scalar multiplication

Vectors

- Used to represent points and offsets
- Slightly different datatypes
  - point = point = vector
  - point + vector = point
  - point + point = ?

Representing vectors

- Can use scalars
  - in our case, a pair of real numbers
  - x and y coordinates in some coordinate system
  - addition = add componentwise
  - scaling = multiply both components
**The dot product**

- Dot product = inner product
- Notation: \( u \cdot v, u^T v, \langle u, v \rangle \)
- \( v \cdot v = \|v\|^2 \)
- geometric interpretation: projection
- implementation: \( x_u x_v + y_u y_v \)

**Linear independence and bases**

- linear combination: \( au + bv \)
- linear (in)dependence
  - vectors in same direction: dependent
    - linear combination not useful
  - vectors not in same direction: independent
    - linear combinations can represent all vectors
    - said to be a basis for 2D space
- Representing vectors
  - canonical basis: \( \{e_1, e_2\} \)
  - Abbreviation for \( \begin{bmatrix} x \\ y \end{bmatrix} \)

**Defining geometry**

- Subsets of the plane
- Generally curves (1D) and areas (2D)
  - related, because the boundaries of areas are curves
- Explicit (parametric) and implicit forms

**Implicit representations**

- Equation to tell whether we are on the curve
  - \( \{v \mid f(v) = 0\} \)
  - Example: line
    - \( \{v \mid v \cdot u + k = 0\} \)
  - Example: circle
    - \( \{v \mid (v - p) \cdot (v - p) + r^2 = 0\} \)
  - Always define boundary of region
    - (if \( f \) is continuous)

**Explicit representations**

- Also called parametric
- Equation to map domain into plane
  - \( \{f(t) \mid t \in D\} \)
  - Example: line
    - \( \{p + tu \mid t \in \mathbb{R}\} \)
  - Example: circle
    - \( \{p + r[\cos t, \sin t]^T \mid t \in [0, 2\pi]\} \)
  - Like tracing out the path of a particle over time
  - Variable \( t \) is the “parameter”

**Parametric vs. implicit**

- Parametric: more direct handle on geometry
  - e.g. easy to generate points on the curve
- Implicit: more global view
  - e.g. easy to tell inside vs. outside
- Many operations equally convenient with either
  - e.g. computing normals or curvature
**Transforming geometry**

- Move a subset of the plane using a mapping from the plane to itself
  - \( S \rightarrow \{ T(v) \mid v \in S \} \)
- Parametric representation:
  - \( \{ f(t) \mid t \in D \} \rightarrow \{ T(f(t)) \mid t \in D \} \)
- Implicit representation:
  - \( \{ v \mid f(v) = 0 \} = \{ T(v) \mid f(v) = 0 \} \)
  - \( \{ v \mid T^{-1}(f(v)) = 0 \} \)

**Translation**

- Simplest transformation: \( T(v) = v + u \)
- Inverse: \( T^{-1}(v) = v - u \)
- Example of transforming circle

**Linear transformations**

- Any transformation with the property:
  - \( T(au + v) = aT(u) + T(v) \)
- Can be represented using matrix multiplication
  - \( T(v) = Mv \)

**Matrix-vector multiplication**

\[
\begin{bmatrix}
  a & b \\
  c & d
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  x_2
\end{bmatrix} =
\begin{bmatrix}
  ax_1 + by_1 \\
  cx_1 + dy_1
\end{bmatrix}
\]

- Interpretations:
  - sum of scalar products of columns
  - list of dot products with rows

**Matrix-matrix multiplication**

- Just M-V multiplication for each column

\[
\begin{bmatrix}
  a & b \\
  c & d
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  y_1
\end{bmatrix} =
\begin{bmatrix}
  ax_1 + by_1 \\
  cx_1 + dy_1
\end{bmatrix}
\]

- Key property: \( (MN)v = M(Nv) \)

- Transpose and column vectors

**Geometry of 2D linear trans.**

- 2x2 matrices have simple geometric interpretations
  - uniform scale
  - non-uniform scale
  - rotation
  - shear
  - reflection
- Reading off the matrix
2D Geometry and Transformations
CONT'D
CS 417 Lecture 8

Programming hint

byte b = 255; // b == -1 == 0b11111111
int i = b; // i == -1 == 0b111...11111111
int i = b & 0xff; // i = 255 = 0b000...01111111

Matrix operations

• Matrix transpose: flip rows and columns

\[
\begin{bmatrix}
  a & b \\
  c & d
\end{bmatrix}^T =
\begin{bmatrix}
  a & c \\
  b & d
\end{bmatrix}
\]

• Identity matrix

\[
I = \begin{bmatrix}
  1 & 0 \\
  0 & 1
\end{bmatrix}
\]

– For all matrices \( M \), \( IM = MI = M \)

Matrix operations

• Matrix inverse: cancels matrix, leaving identity

\[
M^{-1}M = MM^{-1} = I
\]

– inverse undoes whatever \( M \) does
– only exists for nonsingular \( M \) (\( M \) does not flatten things)

Composing transformations

• Want to move an object, then move it some more
  
  - \( \mathbf{p} \rightarrow T(\mathbf{p}) \rightarrow S(T(\mathbf{p})) = (S \circ T)(\mathbf{p}) \)

• We need to represent \( S \circ T \)
  
  – and would like to use the same representation as for \( S \) and \( T \)

• Translation easy

  - \( T(\mathbf{p}) = \mathbf{p} + \mathbf{u}_T; S(\mathbf{p}) = \mathbf{p} + \mathbf{u}_S \)

  - \( (S \circ T)(\mathbf{p}) = \mathbf{p} + (\mathbf{u}_T + \mathbf{u}_S) \)

• commutative!

Composing transformations

• Linear transformations also straightforward

  - \( T(\mathbf{p}) = M_T \mathbf{p}; S(\mathbf{p}) = M_S \mathbf{p} \)

  - \( (S \circ T)(\mathbf{p}) = M_S M_T \mathbf{p} \)

• only sometimes commutative
  
  – e.g. rotations & uniform scales
  – e.g. non-uniform scales w/o rotation
Combining linear with translation

- Need to use both in single framework
- Can represent arbitrary seq. as
  \[ T(p) = Mp + u \]
  \[ S(p) = Ms p + u_s \]
  \[ (S \circ T)(p) = Ms(MTp + u_T) + u_s \]
- e.g. \( S(T(0)) = S(u_T) \)
- This will work but is a little awkward

Homogeneous coordinates

- A trick for representing the foregoing simply
- Extra component \( w \) for vectors, extra row/column for matrices
  - for affine, can always keep \( w = 1 \)
- Represent linear transformations with dummy extra row and column

\[
\begin{bmatrix}
a & b & 0 \\
c & d & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1 \\
\end{bmatrix}
= \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}
\]

Homogeneous coordinates

- Represent translation using the extra column

\[
\begin{bmatrix} 1 & 0 & t \\ 0 & 1 & s \\ 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
= \begin{bmatrix} x + t \\ y + s \\ 1 \end{bmatrix}
\]

Affine transformations

- The set of transformations we have been looking at is known as the “affine” transformations
  - straight lines preserved; parallel lines preserved
  - ratios of lengths along lines preserved (midpoints preserved)

Affine transformation gallery

- Translation

\[
\begin{bmatrix} 0 & 0 & t_x \\ 0 & 0 & t_y \\ 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 0 & 0 & 2.15 \\ 0 & 0 & 0.85 \\ 0 & 0 & 1 \end{bmatrix}
\]
Affine transformation gallery

- Translation

- Uniform scale

- Nonuniform scale

- Rotation

- Reflection
  - can consider it a special case of nonuniform scale

- Shear
**General affine transformations**

- The previous slides showed “canonical” examples of the types of affine transformations
- Generally, transformations contain elements of multiple types
  - often define them as products of canonical transforms
  - sometimes work with their properties more directly

**Composite affine transformations**

- In general **not** commutative: order matters!

  rotate, then translate
  translate, then rotate

**Composite affine transformations**

- In general **not** commutative: order matters!

  rotate, then translate
  translate, then rotate

**Composite affine transformations**

- In general **not** commutative: order matters!

  rotate, then translate
  translate, then rotate

**Composite affine transformations**

- Another example

  scale, then rotate
  rotate, then scale
**Composite affine transformations**

- Another example

```
    scale, then rotate
    rotate, then scale
```

**Composite affine transformations**

- Another example

```
    scale, then rotate
    rotate, then scale
```

**Composite affine transformations**

- Another example

```
    scale, then rotate
    rotate, then scale
```

**Rigid motions**

- A transform made up of only translation and rotation is a **rigid motion** or a **rigid body transformation**

- The linear part is an orthonormal matrix

```
R = \begin{bmatrix} Q & u \\
0 & 1 \end{bmatrix}
```

- Inverse of orthonormal matrix is transpose
  - so inverse of rigid motion is easy:

```
R^{-1}R = \begin{bmatrix} Q^T & -Q^Tu \\
0 & 1 \end{bmatrix} \begin{bmatrix} Q & u \\
0 & 1 \end{bmatrix}
```

**Composing to change axes**

- Want to rotate about a particular point
  - could work out formulas directly...

- Know how to rotate about the origin
  - so translate that point to the origin

```
M = T^{-1}RT
```

**Composing to change axes**

- Want to rotate about a particular point
  - could work out formulas directly...

- Know how to rotate about the origin
  - so translate that point to the origin

```
M = T^{-1}RT
```
**Composing to change axes**

- Want to rotate about a particular point
  - could work out formulas directly…
- Know how to rotate about the origin
  - so translate that point to the origin

\[ M = T^{-1}RT \]

---

**Composing to change axes**

- Want to rotate about a particular point
  - could work out formulas directly…
- Know how to rotate about the origin
  - so translate that point to the origin

\[ M = T^{-1}RT \]

---

**Composing to change axes**

- Want to scale along a particular axis and point
- Know how to scale along the y axis at the origin
  - so translate to the origin and rotate to align axes

\[ M = T^{-1}R^{-1}SRT \]

---

**Composing to change axes**

- Want to scale along a particular axis and point
- Know how to scale along the y axis at the origin
  - so translate to the origin and rotate to align axes

\[ M = T^{-1}R^{-1}SRT \]
Composing to change axes

- Want to scale along a particular axis and point
- Know how to scale along the $y$ axis at the origin
  - so translate to the origin and rotate to align axes

\[ M = T^{-1}R^{-1}SRT \]

Affine change of coordinates

- Six degrees of freedom

\[
\begin{bmatrix}
  a_1 & a_2 & a_3 \\
  a_4 & a_5 & a_6 \\
  0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
  u \\
  v \\
  p \\
\end{bmatrix}
\]

Affine change of coordinates

- Coordinate frame: point plus basis
- Interpretation: transformation changes representation of point from one basis to another
- "Frame to canonical" matrix has frame in columns
  - takes points represented in frame
  - represents them in canonical basis
  - e.g. $[0 \ 0 \ \ 0 \ 1]$

Seems backward but bears thinking about

Affine change of coordinates

- When we move an object to the origin to apply a transformation, we are really changing coordinates
  - the transformation is easy to express in object's frame
  - so define it there and transform it

\[
T_e = FT_F F^{-1}
\]

- $T_e$ is the transformation expressed wrt. $\{e_1, e_2\}$
- $T_F$ is the transformation expressed in natural frame
- $F$ is the frame-to-canonical matrix $[u \ v \ p]$ 

This is a similarity transformation

Orthonormal frames

- If the frame matrix $F$ is a rigid motion, then the frame is an orthonormal frame
  - this just means $u$ and $v$ are perpendicular and unit length
- This is actually the common case, because orthonormal frames make things convenient
  - computationally: easy to invert
  - intellectually: easier to think about
Coordinate frame summary

- Frame = point plus basis
- Frame matrix (frame-to-canonical) is

\[ F = \begin{bmatrix} u & v & p \\ 0 & 0 & 1 \end{bmatrix} \]

- Move points to and from frame by multiplying with \( F \)

\[ p_e = F p_F \quad p_F = F^{-1} p_e \]

- Move transformations using similarity transforms

\[ T_e = F T_F F^{-1} \quad T_F = F^{-1} T_e F \]