CM2202: Scientific Computing and Multimedia Applications
Geometric Computing: 4. Transformations and Fitting

Dr. Yukun Lai

School of Computer Science & Informatics

Geometric Transformations

Some Common Geometric Transformations:

2D Scaling: $\begin{pmatrix} x_k & 0 \\ 0 & y_k \end{pmatrix}$

2D Rotation:

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

2D Shear (x axis): $\begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}$

2D Shear (y axis): $\begin{pmatrix} 1 & 0 \\ k & 1 \end{pmatrix}$

3D Scaling: $\begin{pmatrix} x_k & 0 & 0 \\ 0 & y_k & 0 \\ 0 & 0 & z_k \end{pmatrix}$

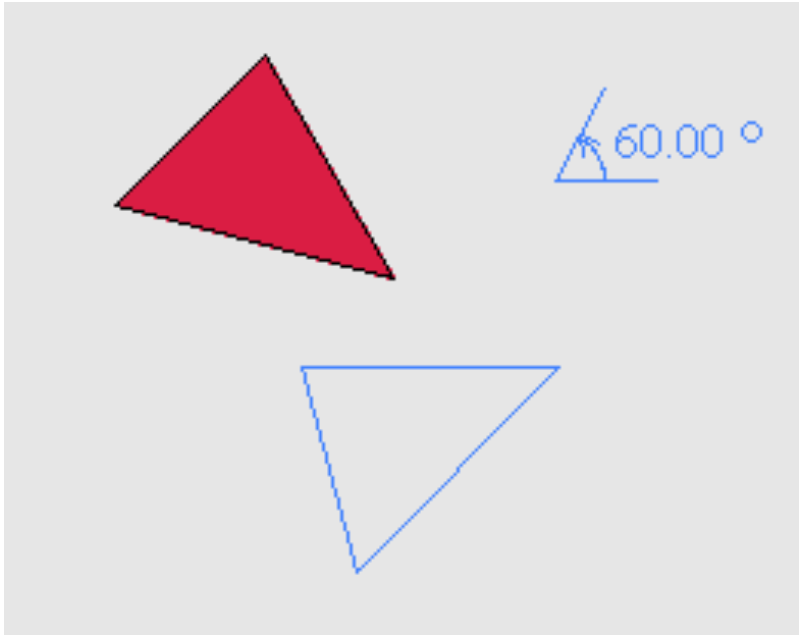
3D Rotation about z axis:

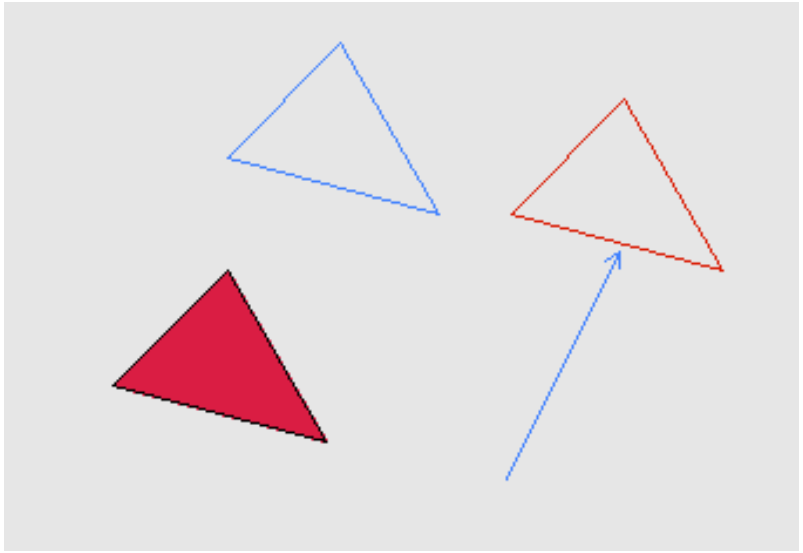
$$\begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2D Translation (Homogeneous Coords):

$$\begin{pmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{pmatrix}$$

Rotation and Translation





$$\begin{bmatrix} X_{rotated} \\ Y_{rotated} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} X_{translated} \\ Y_{translated} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & D_x \\ 0 & 1 & D_y \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Applying Geometric Transformations

When applying geometric transformations to objects

- Rotate the point coordinate and vectors (normals etc.)
  - Simply do matrix-vector multiplication
- For polygons and Boundary representations only need to apply transformation to the point (geometric) data – **the topology is unchanged**

Compound Geometric Transformations

It is a simple task to create compound transformations (*e.g.* A rotation followed by a translation:

- Perform Matrix-Matrix multiplications, *e.g.* $\mathbf{T}.\mathbf{R}_\theta$
  - Note in general $\mathbf{T}.\mathbf{R}_\theta \neq \mathbf{R}_\theta.\mathbf{T}$ (Basic non-commutative law of multiplication)
- One problem exists in that a 2D Translation **cannot** be represented as a 2D matrix — it must be represented as a 3D matrix:

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{pmatrix}$$

- So how do we combine a 2D rotation $\mathbf{R}_\theta. = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$ with a translation?
  We need to define **Homogeneous Coordinates** to deal with this issue.

Homogeneous Coordinates

To work in homogeneous coordinates do the following:

- Increase the dimension of the space by 1, *i.e.* *2D* $\rightarrow$ 3D, 3D$\rightarrow$ 4D. We can now accommodate the translation matrix.
- Convert all standard other 2D transformations via the following:

$$\begin{pmatrix} X & X & 0 \\ X & X & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

  where $\begin{pmatrix} X & X \\ X & X \end{pmatrix}$ is the regular 2D transformation
- Perform compound transformation by matrix-matrix multiplication in homogeneous matrices

Homogeneous Coordinates Example: 2D Rotation

The transformation for a 2D rotation (angle of rotation θ is:

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

The corresponding **homogenous form** is:

$$\begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Compound Geometric Transformations MATLAB Examples

- Simply create the matrices for individual transformations in MATALB (2D, 3D or homogeneous)
- Assemble compound via matrix multiplication.
- Apply compound transform to coordinates/vectors etc. via matrix-vector multiplication.

2D Geometric Transformation MATLAB Demo Code

3D Transformations

3D Rotation about x axis:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix}$$

3D Rotation about y axis:

$$\begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

3D Rotation about z axis:

$$\begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3D Scaling:

$$\begin{pmatrix} x_k & 0 & 0 \\ 0 & y_k & 0 \\ 0 & 0 & z_k \end{pmatrix}$$

3D Translation (Homogeneous Coords (4D):

$$\begin{pmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

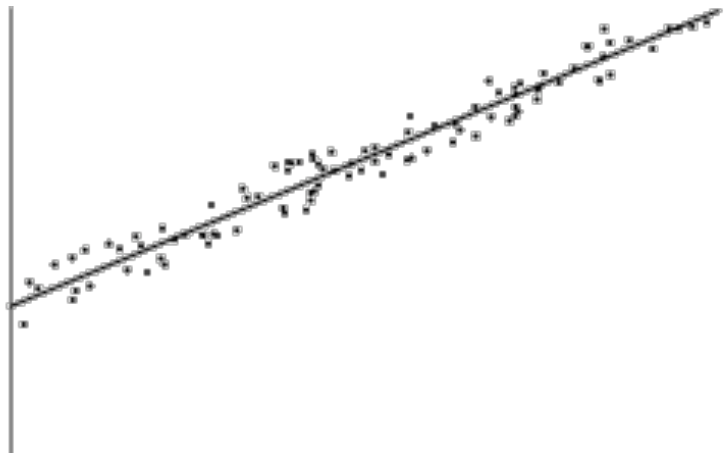
Need to put all 3D transformation into homogenous form for compound transformations.

Least Squares Fitting

We have already seen that we need 2 points to define a line and 3 points to define circle and a plane — the minimum number of free parameters.
However, in many data processing applications we will have many more samples than the minimum number required.

- Errors in the positions of these points mean that in practice they do not all lie exactly on the expected line/curve/surface.
- **Least squares approximation** methods find the surface of the given type which best represents the data points.
    - Minimise some error function

Least Squares Fit of a Line



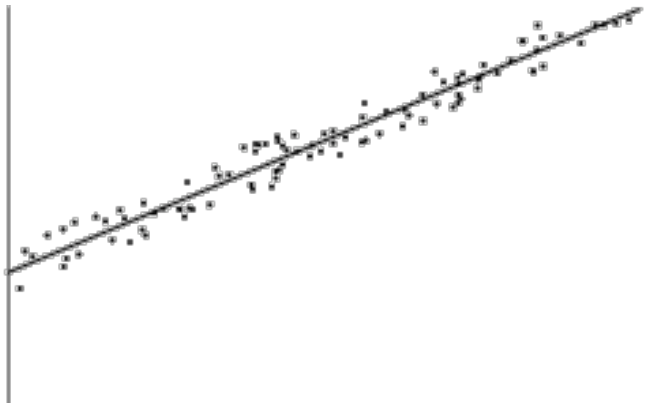
We use the general explicit equation of a line here:

$$y = mx + c$$

We can define an error, σ_i as the distance from the line:

$$\sigma_i = y_i - mx_i - c$$

Least Squares Fit of a Line (Cont.)



The best fitting plane to a set of n points $\{P_i\}$ occurs when

$$\chi^2 = \sum \sigma_i^2 = \sum (y_i - mx_i - c)^2$$

is a minimum, where this and all subsequent sums are to be taken over all points, $P_i, i = 1, \ldots, n$.

This occurs when

$$\begin{aligned} \frac{\partial \chi^2}{\partial m} &= 0 \\ \frac{\partial \chi^2}{\partial c} &= 0 \end{aligned}$$

Least Squares Fit of a Line (Cont.)

Now

$$\frac{\partial \chi^2}{\partial m} = -2\sum(y_i - mx_i - c)x_i = 0$$
$$\frac{\partial \chi^2}{\partial c} = -2\sum(y_i - mx_i - c) = 0$$

which leads to

$$\sum y_i - m\sum x_i - nc = 0$$
$$\sum x_i y_i - m\sum x_i^2 - c\sum x_i = 0$$

which can be solved for m and c to give:

$$m = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}$$
$$c = \frac{\sum y_i \sum x_i^2 - \sum x_i \sum x_i y_i}{n\sum x_i^2 - (\sum x_i)^2}$$

Least Squares Fit of a Line (MATLAB Code)

The MATLAB code to perform least squares line fitting is, line_fit.m

```
function [m c] = line_fit(x,y)

[n1 n] = size(x);

sumx = sum(x);
sumy = sum(y);

sumx2 = sum(x.*x);
sumxy = sum(x.*y);

det = n* sumx2 - sumx*sumx;
m = (n*sumxy - sumx*sumy)/det;
c = (sumy*sumx2 - sumx*sumxy)/det;
```

See line_fit_demo.m for example call.

Least Squares Fit of a Plane

Let us suppose we are given a set of points in the form $z = z(x, y)$, *i.e.* we are given z_i values for the points lying at a set of known (x_i, y_i) positions.

The equation of a plane can be written as

$$z = ax + by + c.$$

For a given point $P_i = (x_i, y_i, z_i)$ the error in the fit is measured by

$$\sigma_i = z_i - (ax_i + by_i + c).$$

Least Squares Fit of a Plane (Cont.)

The best fitting plane to a set of n points $\{P_i\}$ occurs when $\chi^2 = \sum \sigma_i^2$ is a minimum, where this and all subsequent sums are to be taken over all points, $P_i, i = 1, \ldots, n$.

This occurs when

$$\frac{\partial \chi^2}{\partial a} = 0 \tag{1}$$

and

$$\frac{\partial \chi^2}{\partial b} = 0 \tag{2}$$

and and when

$$\frac{\partial \chi^2}{\partial c} = 0 \tag{3}$$

Least Squares Fit of a Plane (Cont.)

Now $\frac{\partial \chi^2}{\partial a} = 0$ gives:

$$\sum x_i z_i - a \sum x_i^2 - b \sum x_i y_i - c \sum x_i = 0,$$

and $\frac{\partial \chi^2}{\partial b} = 0$ gives:

$$\sum y_i z_i - a \sum x_i y_i - b \sum y_i^2 - c \sum y_i = 0,$$

and $\frac{\partial \chi^2}{\partial c} = 0$ gives

$$\sum z_i - a \sum x_i - b \sum y_i - nc = 0.$$

Simplifying a Least Squares Fit of a Plane

Now, some of these sums can become quite large and any attempt to solve the equations directly can lead to poor results due to rounding errors.

- One common method (for any least squares approximation) of reducing the rounding errors is to re-express the coordinates of each point relative to the centre of gravity of the set of points, (x_g, y_g, z_g).
- All subsequent calculations are performed under this translation with the final results requiring the inverse translation to return to the original coordinate system.

The translation also simplifies our minimisation problem since now

$$\sum_{\forall i} x_i = \sum_{\forall i} y_i = \sum_{\forall i} z_i = 0.$$

Least Squares Fit of a Plane (Cont.)

Therefore, using the above fact in previous equations and solving for a, b and c gives:

$$a = \frac{\sum x_i z_i \sum y_i^2 - \sum y_i z_i \sum x_i y_i}{\sum x_i^2 \sum y_i^2 - (\sum x_i y_i)^2}, \quad (4)$$

$$b = \frac{\sum y_i z_i \sum x_i^2 - \sum x_i z_i \sum x_i y_i}{\sum x_i^2 \sum y_i^2 - (\sum x_i y_i)^2}, \quad (5)$$

$$c = 0. \quad (6)$$

All we now need to do is to translate back.

Example Applications Visited

- Geographic Information Systems: Point Location
- Geometric Modelling: Spline Fitting
- Computer Graphics: Ray Tracing
- Image Processing: Hough Transform
- Mobile Systems: Spatial Location Sensing

Application-specific techniques are out of the scope of this module but geometric computing is widely used in all of these applications.

Summary

We have discussed

- 2D geometric transformations (scaling, rotation, shearing, translation)
- Homogeneous coordinates
- Compound geometric transformations
- 3D geometric transformations
- Least squares fitting