Three-Dimensional (3D) Transformations

Biswajit Biswas
Ph.D. (C.S.E)

Department of Computer Science
Vivekananda Centenary College
Rahara, Kolkata

March 9, 2025

Geometric Transformations

- Manipulation, viewing, and construction of 3D graphic images requires the use of 3D geometric and coordinate transformations.
- All 3D transformations are formed by composing the basic transformations of translation, scaling, and rotation.
- Each of these **(translation, scaling, and rotation)** transformations can be represented as a matrix transformation.
- This permits more complex transformations to be built up by use of **matrix multiplication or concatenation**.

Geometric Transformations

With respect to some 3D coordinate system, an object **Obj** is considered as a set of points:

$$obj = \mathcal{P}(x, y, z)$$

If the object is moved to a new position, we can regard it as a new object Obj', all of whose coordinate points $\acute{\mathcal{P}}(\acute{x}, \acute{y}, \acute{})$ can be obtained from the original coordinate points $\mathcal{P}(x, y, z)$ of **Obj** through the application of a geometric transformation.

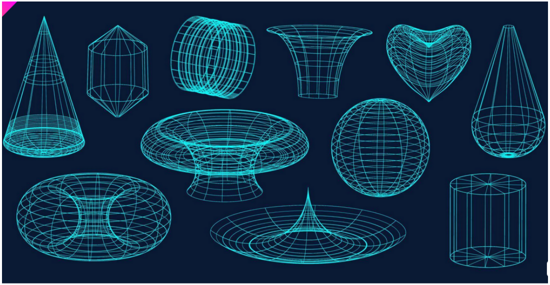


Figure: 3D Objectes

Homogeneous Coordinates

To perform a sequence of transformation such as translation followed by rotation and scaling, we need to follow a sequential process:

- Translate the coordinates,
- Rotate the translated coordinates, and then
- Scale the rotated coordinates to complete the composite transformation.

To shorten this process, we have to use 3x3 transformation matrix instead of 2x2 transformation matrix. To convert a 2x2 matrix to 3x3 matrix, we have to add an extra dummy coordinate W. In this way, we can represent the point by 3 numbers instead of 2 numbers, which is called Homogeneous Coordinate system. In this system, we can represent all the transformation equations in matrix multiplication. Any Cartesian point P$[X, Y]$ can be converted to homogeneous coordinates by $\acute{P}[X_h, Y_h, h]$

Homogeneous Coordinates

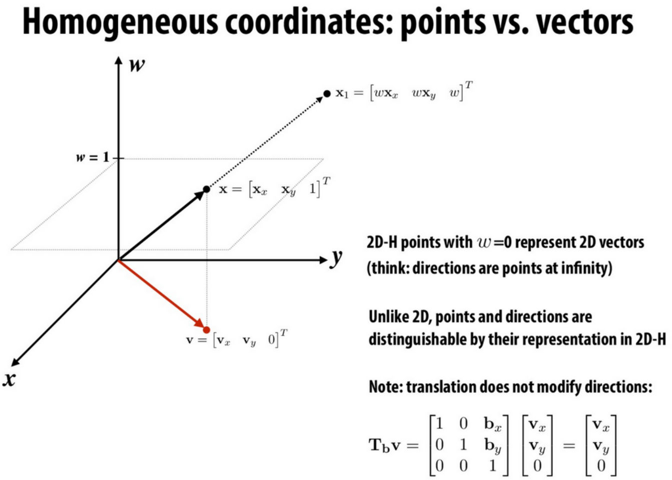


Figure: Homogeneous Coordinates

Translation

An object is displaced a given distance and direction from its original position. The direction and displacement of the translation is prescribed by a vector:

$$\mathcal{V} = a\mathsf{I} + b\mathsf{J} + c\mathsf{K}$$

The new coordinates of a translated point can be calculated by using the transformation:

$$\mathsf{T}_v = \begin{cases} \acute{x} = x + a \\ \acute{y} = y + b \\ \acute{z} = z + c \end{cases}$$

To represent this transformation as a matrix transformation, we use homogeneous coordinates by homogeneous matrix transformation as:

$$\begin{pmatrix} \acute{x} \\ \acute{y} \\ \acute{z} \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Translation

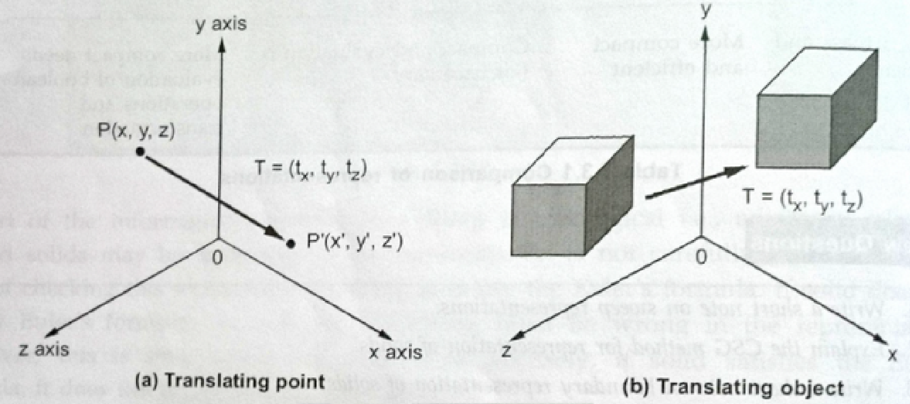


Figure: 3D Translation

Translation

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

$$P' = P \cdot T$$

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

$$= [x + t_x \quad y + t_y \quad z + t_z \quad 1]$$

Figure: 3D Translation

Scaling

The process of scaling changes the dimensions of an object. The scale factor s determines whether the scaling is a magnification, s $>$ 1, or a reduction, s $<$ 1.

$$S_{s_x,s_y,s_z} = \begin{cases} \acute{x} = s_x \cdot x \\ \acute{y} = s_y \cdot y \\ \acute{z} = s_z \cdot z \end{cases}$$

The scaling matrix form as:

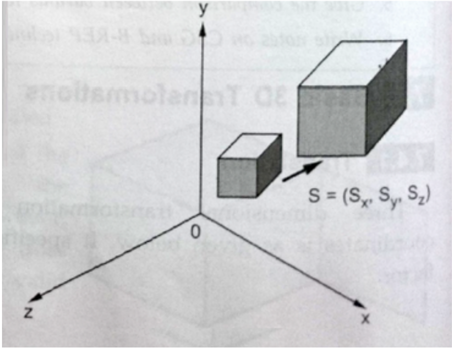
$$S_{s_x,s_y,s_z} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix}$$

A scaling transformation is called **isotropic**, if $s_x = s_y = s_z = 1$. **Isotropic** scaling preserves the similarity of objects (angles).

Scaling

Mirroring about one of the major planes $(xy, xz, oryz)$ can be described as a special case of the scaling transformation, by using $\mathbf{a} = -1$ scaling factor. For the inverse scaling, we can use the inverse of the scaling matrix as:

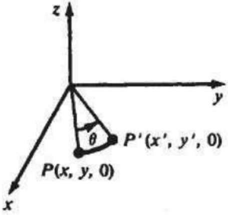
$$\mathrm{S}^{-1}[s_x, s_y, s_z] = \mathrm{S}[\frac{1}{s_x}, \frac{1}{s_y}, \frac{1}{s_z}]$$



Rotation

Rotation in three dimensions is considerably more complex than rotation in two dimensions:

- 3D rotations require the prescription of an angle of rotation and an axis of rotation;
- The canonical rotations are defined when one of the positive **x**, **y**, or **z** coordinate axes is chosen as the axis of rotation;
- The construction of the rotation transformation proceeds just like that of a rotation in two dimensions about the origin;



Rotation about the z Axis

we know that:

$$R_{\theta,\mathbf{K}} = \begin{cases} \acute{x} = x\cos(\theta) - y\sin(\theta) \\ \acute{y} = x\sin(\theta) + y\cos(\theta) \\ \acute{z} = z \end{cases}$$

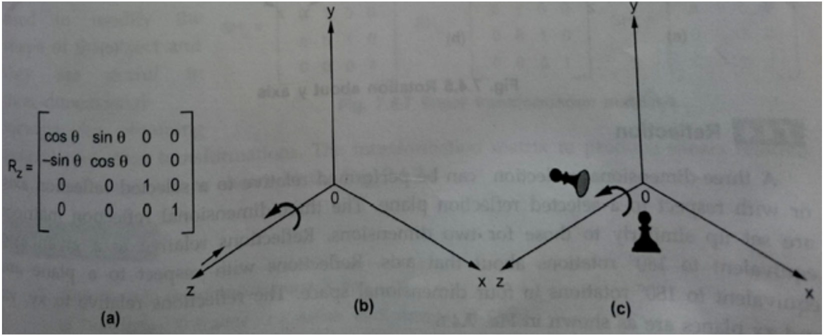


Figure: Rotation about the z Axis

Rotation about the y Axis

we know that:

$$R_{\theta,\mathbf{J}} = \begin{cases} \acute{x} = x\cos(\theta) + z\sin(\theta) \\ \acute{y} = y \\ \acute{z} = -x\sin(\theta) + z\cos(\theta) \end{cases}$$

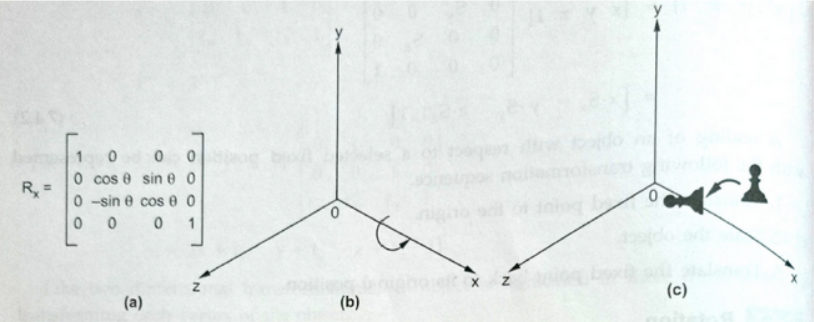


Figure: Rotation about the y Axis

Rotation about the x Axis

we know that:

$$R_{\theta,\mathbf{I}} = \begin{cases} \acute{x} = x \\ \acute{y} = y\cos(\theta) - z\sin(\theta) \\ \acute{z} = y\sin(\theta) + z\cos(\theta) \end{cases}$$

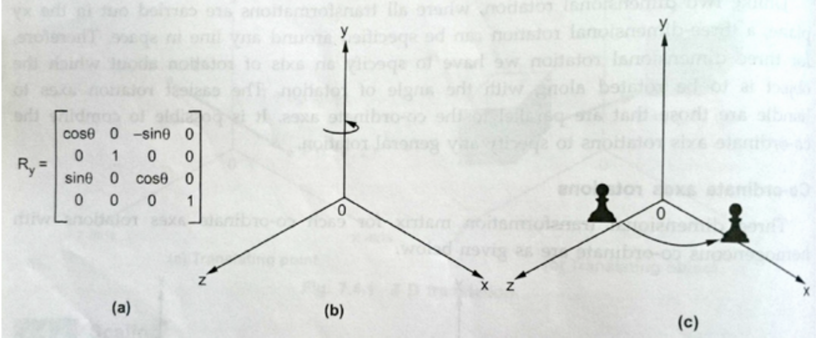


Figure: Rotation about the x Axis

Rotation about the xyz Axis

The corresponding matrix transformations are:

(a) **Rotation about the z Axis**:

$$\mathrm{R}_{\theta,K} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(b) **Rotation about the y Axis**:

$$\mathrm{R}_{\theta,J} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

(c) **Rotation about the x Axis**:

$$\mathrm{R}_{\theta,J} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

Reflection

- A 3D reflection can be performed relative to a Selected reflection axis or with respect to a selected reflection plane.
- The 3D reflection matrices are set up similarly to those for two dimensions.
- Reflections relative to a given axis equivalent to 180° rotations about that axis.
- Reflections with respect to a plane are equivalent to 180° rotations in four dimensional space.

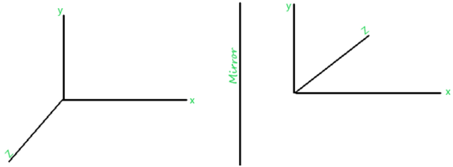


Figure: Reflection X-Y

Reflection X-Y

Reflection in 3D space is quite similar to the reflection in 2D space, but a single difference is there in 3D, here we have to deal with three axes (x, y, z). Reflection is nothing but a mirror image of an object. Three kinds of Reflections are possible in 3D space:

- Reflection along the **X-Y** plane.
- Reflection along **Y-Z** plane.
- Reflection along **X-Z** plane.

Consider, a point $P[x, y, z]$ which is in 3D space is made to reflect along X-Y direction after reflection $P[x, y, z]$ becomes $P'[x', y', z']$.

$$\acute{P}[x, y, z] = P[x, y, z]R_{xy}$$

where

$$R_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

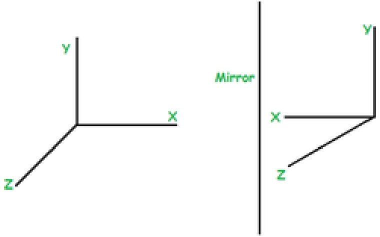
Reflection (Y-Z)

Consider, a point $P[x, y, z]$ which is in 3D space is made to reflect along Y-Z direction after reflection $P[x, y, z]$ becomes $P'[x', y', z']$.

$$\acute{P}[x, y, z] = P[x, y, z]R_{yz}$$

The reflection transformation matrix for Y-Z axes is as follows:

$$R_{yz} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$



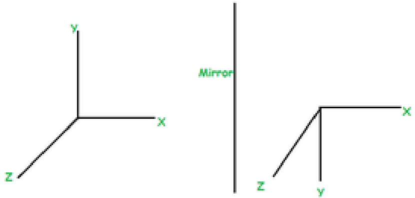
Reflection (X-Z)

Consider, a point $P[x, y, z]$ which is in 3D space is made to reflect along Y-Z direction after reflection $P[x, y, z]$ becomes $P'[x', y', z']$.

$$\acute{P}[x, y, z] = P[x, y, z]R_{yz}$$

The reflection transformation matrix for X-Z axes is as follows:

$$R_{xz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$



Reflection

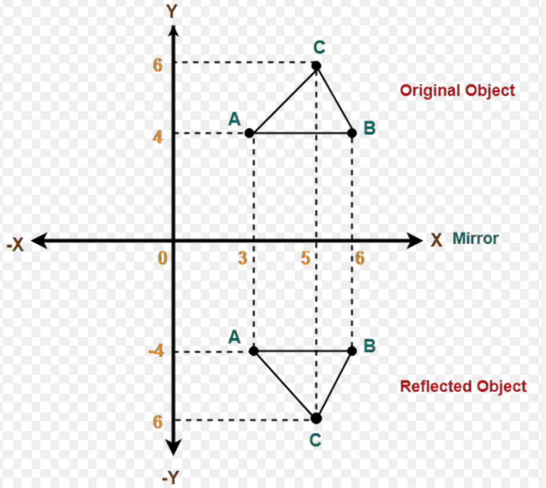


Figure: Reflection

Reflection

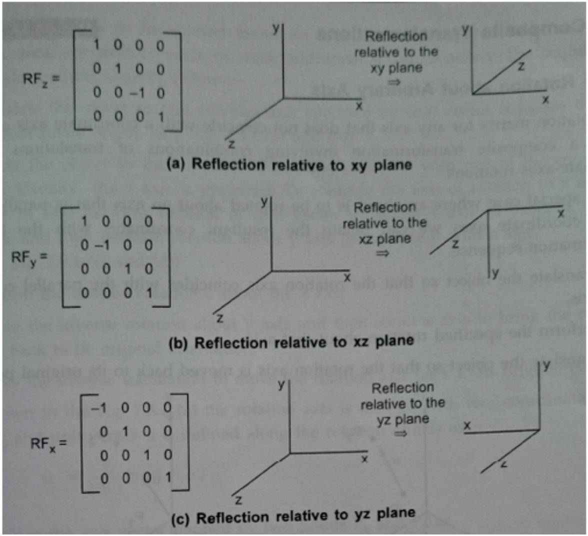
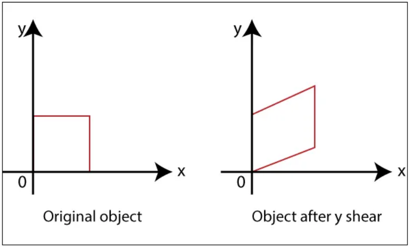


Figure: Reflection

3D Shearing Transformation

- Shearing transformation is the same as we see in 2D space, but here we have to deal with the x, y, and z axes whereas in 2D we deal with the only x and y axes.
- Shearing is the process of slanting an object in 3D space either in x, y, or in the z-direction.
- Shearing changes(or deformed) the shape of the object.
- Given below are the types of shearing transformation.
  - Shearing in X-direction.
  - Shearing in y-direction
  - Shearing in z-direction.



3D Shearing in X-Direction

Shearing in X-Direction: Here the coordinate of X remains unchanged while the coordinate of Y and Z is changed. Shearing is done through the Shearing Transformation matrix, which is represented as follows:

$$\mathrm{S_x} = \begin{bmatrix} 1 & s_y & s_z & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Consider a point $P[x, y, z]$ in 3D space over which we perform the shearing transformation in the X-direction and it becomes $P'[x, y, z]$.

$$\mathrm{P'[x, y, z, 1] = P[x_0, y_0, z_0, 1] \cdot S_x}$$

3D Shearing in Y-Direction

Shearing in Y-Direction: Here the coordinate of Y remains unchanged while the coordinate of X and Z is changed. Shearing is done through the Shearing Transformation matrix, which is represented as follows:

$$\mathrm{S_y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ s_x & 1 & s_z & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Consider a point $P[x, y, z]$ in 3D space over which we perform the shearing transformation in the Y-direction and it becomes $P'[x, y, z]$.

$$\mathrm{P'[x, y, z, 1] = P[x_0, y_0, z_0, 1] \cdot S_y}$$

3D Shearing in Z-Direction

Shearing in Z-Direction: Here the coordinate of Z remains unchanged while the coordinate of X and Y is changed. Shearing is done through the Shearing Transformation matrix, which is represented as follows:

$$\mathrm{S_z} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ s_x & s_y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Consider a point $P[x, y, z]$ in 3D space over which we perform the shearing transformation in the Z-direction and it becomes $P'[x, y, z]$.

$$\mathrm{P'[x, y, z, 1] = P[x_0, y_0, z_0, 1] \cdot S_z}$$

3D Shearing in Z-Direction

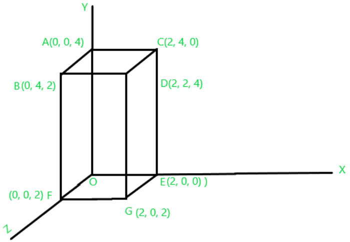


Figure: Before performing the Shearing transformation

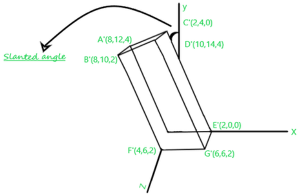
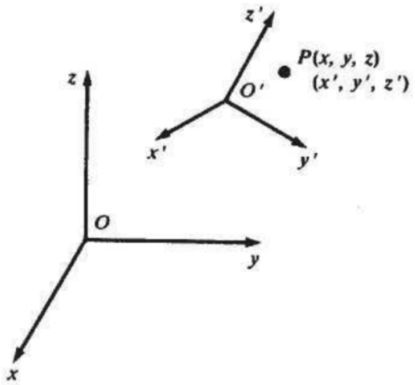


Figure: After performing the Shearing transformation

Coordinate Transformation

We can achieve the effects of translation, scaling, and rotation by moving the observer who views the object and by keeping the object stationary. This type of transformation is called a **coordinate transformation**.
How do it:

- We first attach a coordinate system to the observer and then move the observer and the attached coordinate system;
- Next, we recalculate the coordinates of the observed object with respect to this new observer coordinate system;
- The new coordinate values will be exactly the same as if the observer had remained stationary and the object had moved, corresponding to a geometric transformation;



Coordinate Transformation

If the displacement of the observer coordinate system to a new position is prescribed by a vector $\mathbf{V} = a\mathbf{I} + b\mathbf{J} + c\mathbf{K}$, a point $P(x, y, z)$ in the original coordinate system has coordinates $P(x', y', z')$ in the new coordinate system, and

$$\bar{T}_{\mathbf{v}}: \begin{cases} x' = x - a \\ y' = y - b \\ z' = z - c \end{cases}$$

	Coordinate Transformations	Geometric Transformations
Translation	$\bar{T}_{\mathrm{V}}$	$T_{-\mathrm{V}}$
Rotation	$\bar{R}_{\theta}$	$R_{-\theta}$
Scaling	$\bar{S}_{s_x, s_y, s_z}$	$S_{1/s_x, 1/s_y, 1/s_z}$

Inverse geometric and coordinate transformations are constructed by performing the reverse operation. Thus, for coordinate transformations (and similarly for geometric transformations):

$$\bar{T}_{\mathbf{v}}^{-1} = \bar{T}_{-\mathbf{v}} \qquad \bar{R}_{\theta}^{-1} = \bar{R}_{-\theta} \qquad \bar{S}_{s_x, s_y, s_z}^{-1} = \bar{S}_{1/x_x, 1/s_y, 1/s_z}$$

Affine Transformation

Affine transformation is a linear mapping method that preserves points, straight lines, and planes.

- Sets of parallel lines remain parallel after an affine transformation.
- The affine transformation technique is typically used to correct for geometric distortions or deformations that occur with non-ideal camera angles.

In satellite imagery, affine transformation is a technique used to correct geometric distortions and deformations, preserving parallelism of lines but not necessarily angles or lengths, by applying linear transformations like scaling, rotation, and shearing.
Mainly, satellite imagery uses affine transformations to correct for wide angle lens distortion, panorama stitching, and **image registration**.
Transforming and **fusing** the images to a large, flat coordinate system is desirable to eliminate distortion.

Affine Transformation

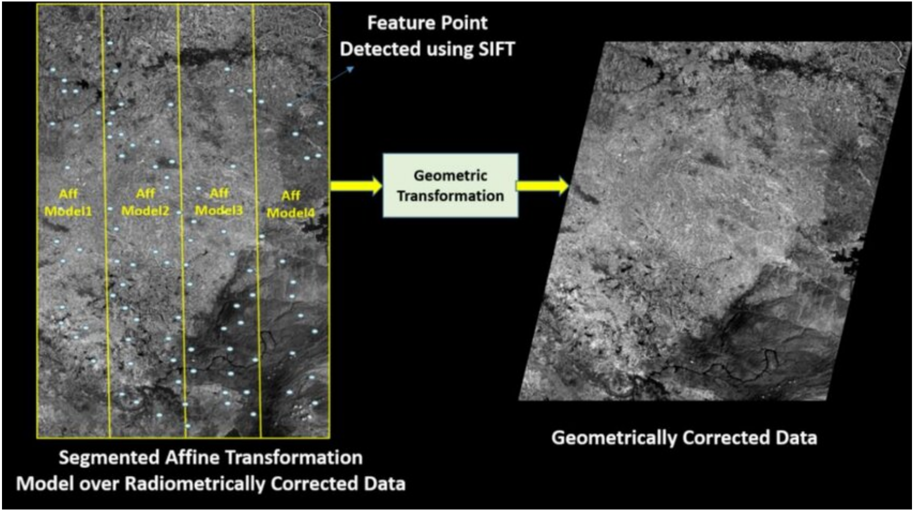


Figure: Affine Transformation

Affine Transformation

The following table illustrates the different affine transformations: translation, scale, shear, and rotation.

Affine Transform	Example	Transformation Matrix	
Translation		$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	t_x specifies the displacement along the x axis t_y specifies the displacement along the y axis.
Scale		$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	s_x specifies the scale factor along the x axis s_y specifies the scale factor along the y axis.
Shear		$\begin{bmatrix} 1 & sh_y & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	sh_x specifies the shear factor along the x axis sh_y specifies the shear factor along the y axis.
Rotation		$\begin{bmatrix} \cos(q) & \sin(q) & 0 \\ -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$	q specifies the angle of rotation.

Figure: Affine Transformation

Composite Transformation

A number of transformations or sequence of transformations can be combined into single one called as composition. The resulting matrix is called as composite matrix. The process of combining is called as concatenation.

Suppose we want to perform rotation about an arbitrary point, then we can perform it by the sequence of three transformations:

- Translation (T_v)
- Rotation (R_θ)
- Scaling (S_{xyz})
- Reverse Translation (T_v^{-1})

The ordering sequence of these numbers of transformations must not be changed. If a matrix is represented in column form, then the composite transformation is performed by multiplying matrix in order from right to left side. The output obtained from the previous matrix is multiplied with the new coming matrix.

Composite Transformation

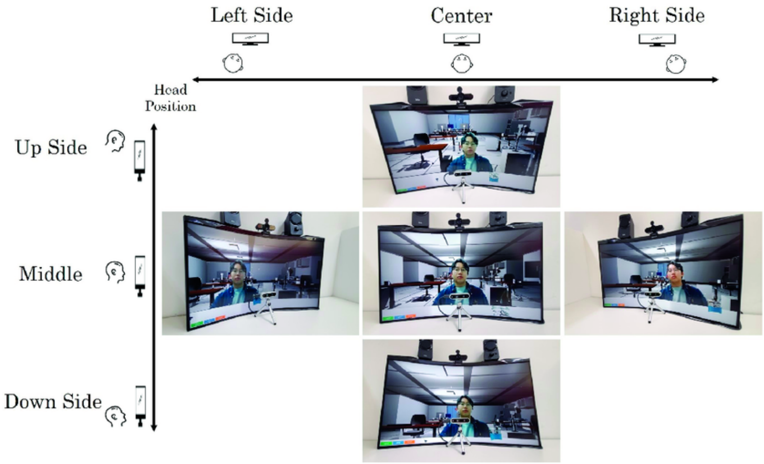


Figure: Composite Transformation

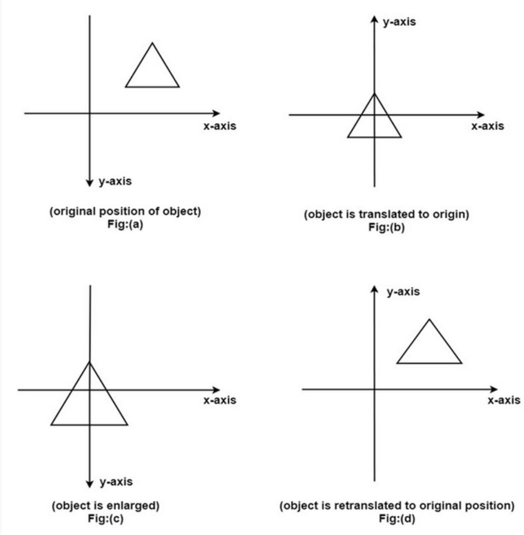
Composite Transformation

Example showing composite transformations: The enlargement is with respect to center. For this following sequence of transformations will be performed and all will be combined to a single one:

1. **Step1:** The object is kept at its position as in fig 35(a);
2. **Step2:** The object is translated so that its center coincides with the origin as in fig 35(b);
3. **Step3:** Scaling of an object by keeping the object at origin is done in fig 35(c);
4. **Step4:** Again translation is done. This second translation is called a reverse translation. It will position the object at the origin location.

Above transformation can be represented as $T_V \cdot S_{xyz} T_V^{-1}$ in fig 35(d).

Composite Transformation



Composite Transformation

Advantage of composition or concatenation of matrix:

- It transformations become compact.
- The number of operations will be reduced.
- Rules used for defining transformation in form of equations are complex as compared to matrix.

Composite Transformations

(A) Translations

If two successive translation vectors (t_{x1},t_{y1}) and (t_{x2},t_{y2}) are applied to a coordinate position P, the final transformed location P' is calculated as: -

$$P' = T(t_{x2},t_{y2}) \cdot \{T(t_{x1},t_{y1}) \cdot P\}$$
$$= \{T(t_{x2},t_{y2}) \cdot T(t_{x1},t_{y1})\} \cdot P$$

Where P and P' are represented as homogeneous-coordinate column vectors. We can verify this result by calculating the matrix product for the two associative groupings. Also, the composite transformation matrix for this sequence of transformations is: -

$$\begin{vmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} \bullet & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} \bullet & 0 & t_{x1}+t_{x2} \\ 0 & 1 & t_{y1}+t_{y2} \\ 0 & 0 & 1 \end{vmatrix}$$

Or, $T(t_{x2},t_{y2}) \cdot T(t_{x1},t_{y1}) = T(t_{x1}+t_{x2}, t_{y1}+t_{y2})$

Which demonstrate that two successive translations are additive.

(B) Rotations

Two successive rotations applied to point P produce the transformed position: -

$$P' = R(\Theta_2) \cdot \{R(\Theta_1) \cdot P\}$$
$$= \{R(\Theta_2) \cdot R(\Theta_1)\} \cdot P$$

By multiplication the two rotation matrices, we can verify that two successive rotations are additive:

$$R(\Theta_2) \cdot R(\Theta_1) = R(\Theta_1 + \Theta_2)$$

So that the final rotated coordinates can be calculated with the composite rotation matrix as: -

$$P' = R(\Theta_1 + \Theta_2) \cdot P$$

(C) Scaling

Concatenating transformation matrices for two successive scaling operations produces the following composite scaling matrix: -

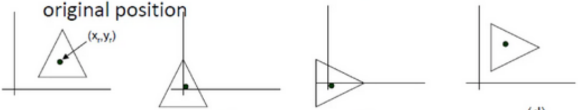
$$\begin{vmatrix} S_{x2} & 0 & 0 \\ 0 & S_{y2} & 0 \\ 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} S_{x1} & 0 & 0 \\ 0 & S_{y1} & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} S_{x1} \cdot S_{x2} & 0 & 0 \\ 0 & S_{y1} \cdot S_{y2} & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

Or, $S(S_{x2}, S_{y2}) \cdot S(S_{x1}, S_{y1}) = S(S_{x1} \cdot S_{x2}, S_{y1} \cdot S_{y2})$

The resulting matrix in this case indicates that successive scaling operations are multiplicative.

General pivot point rotation

- Translate the object so that pivot-position is moved to the coordinate origin
- Rotate the object about the coordinate origin
- Translate the object so that the pivot point is returned to its original position



(a) Original Position of Object and pivot point

(b) Translation of object so that pivot point (x_r,y_r) is at origin

(c) Rotation was about origin

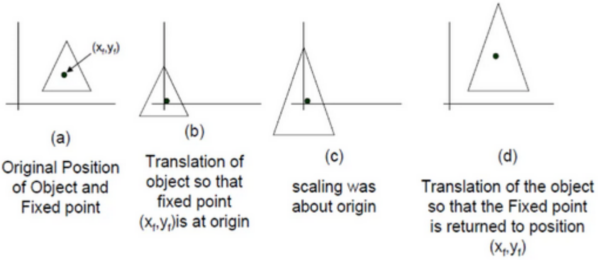
(d) Translation of the object so that the pivot point is returned to position (x_r,y_r)

General fixed point scaling

Translate object so that the fixed point coincides with the coordinate origin

Scale the object with respect to the coordinate origin

Use the inverse translation of step 1 to return the object to its original position



THANK YOU