

# Translation Matrices

A transform matrix is a way to

- Shift a point by the vector (x, y, z)
- Rotate the coordinate frame, and
- Zoom in and out with a scaling factor of w.

Since each point is defined by a 4x1 vector, the transformation matrix needs to be a 4x4 matrix:

$$a_{4 \times 1} = T_{4 \times 4} b_{4 \times 1}$$

T is composed of three parts:

- A 3x3 rotation matrix (identity in this example)
- A 3x1 translation matrix ( [bx, by, bz]<sup>T</sup> )
- A 1x1 scalar (w) defining the zoom in / zoom out factor.

$$\begin{bmatrix} a_x \\ a_y \\ a_z \\ \dots \\ a_w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \vdots & x \\ 0 & 1 & 0 & \vdots & y \\ 0 & 0 & 1 & \vdots & z \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \vdots & w \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \\ \dots \\ b_w \end{bmatrix}$$

Example 1: Shift the point [1,2,3] by [x, y, z] Use a scaling factor of one (w=1).

$$b = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

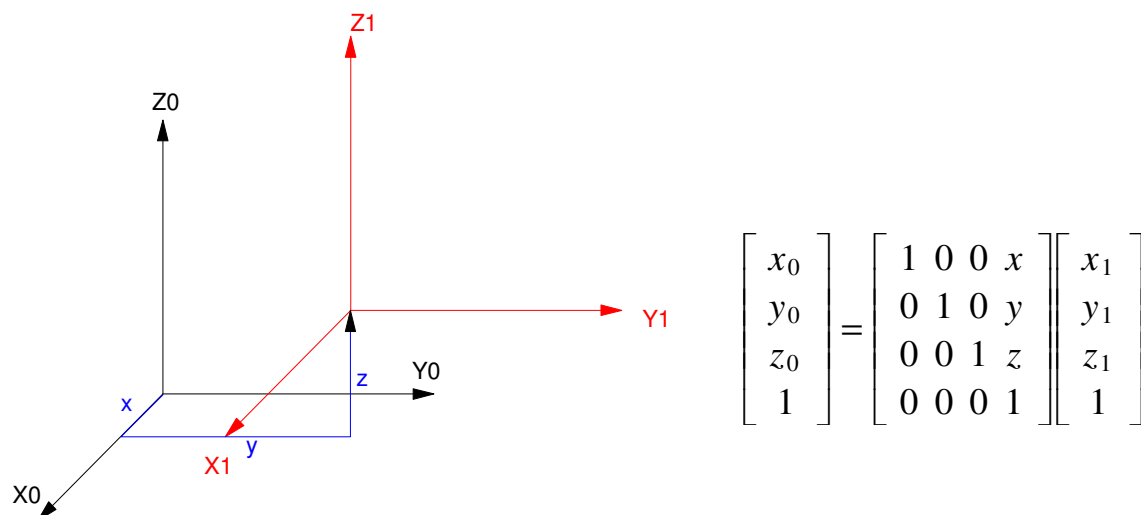
$$a = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1+x \\ 2+y \\ 3+z \\ 1 \end{bmatrix}$$

Point b has been shifted by [x,y,z].

Zoom in with a scaling factor of 2

$$a = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 2 \end{bmatrix}$$

This means if you plot the point (1,2,3), it will be doubled (zoomed in with a factor of 2)



Example 3: Project a 3D image of an arrow on the YZ plane.

The arrow has eight points

-->Arrow

```

0.    0.    0.    0.    0.    0.    0.    0.
- 1.    1.    1.    1.5  0.   -1.5  - 1.   - 1.
0.    0.    1.    1.    2.    1.    1.    0.
1.    1.    1.    1.    1.    1.    1.    1.

```

### The display routine in Matlab

```
function Display3D(DATA, T)

% scaling factor
s = T(4,4);

% draw the X, Y, Z axis
X = [1,0,0,1]';
Y = [0,1,0,1]';
Z = [0,0,1,1]';
O = [0,0,0,1]';

DATA = T*DATA;

T0 = T;
T0(1,4) = 0;
T0(2,4) = 0;
T0(3,4) = 0;

% transform
X0 = T0*X;
Y0 = T0*Y;
Z0 = T0*Z;
Origin = T0*O;

% display is the y-z plane
hold off;
plot([-2,2],[-2,2],'wx');
hold on;

% Project onto the YZ axis

Tx = s*[0,1,0,0];
Ty = s*[0,0,1,0];

plot(Tx*[Origin, X0], Ty*[Origin,X0], 'g');
plot(Tx*[Origin, Y0], Ty*[Origin,Y0], 'r');
plot(Tx*[Origin, Z0], Ty*[Origin,Z0], 'm');

% display the data

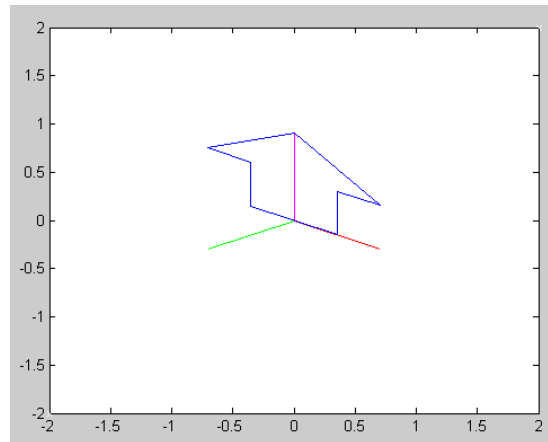
plot(Tx*DATA,Ty*DATA, 'b')

end
```

### To draw this in Matlab

```
c = cos(25*pi/180);
s = sin(25*pi/180);
Ty = [c,0,s,0;0,1,0,0;-s,0,c,0;0,0,0,1];
c = cos(-45*pi/180);
s = sin(-45*pi/180);
Tz = [c,-s,0,0;s,c,0,0;0,0,1,0;0,0,0,1]
Tdisp = Ty*Tz;

Display3D(ARROW,Tdisp);
```



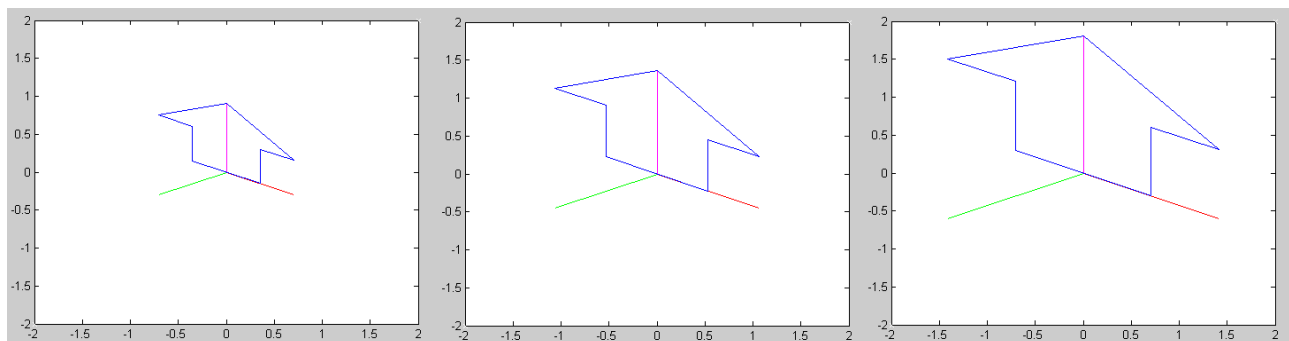
Plot the arrow as you move closer to it (meaning the scaling factor  $w$  changes from 0.1 to 3.0)

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & w \end{bmatrix}$$

```
T = eye(4,4);
for i=0:300
    w = i/100;
    T(4,4) = w;

    Display3D(ARROW,T*Tdisp);
    pause(0.01);
end
```

This shows the arrow getting bigger as you get closer to it



Arrow with scaling factor ( $w$ ) equal to { 1.0, 1.5, 2.0 }

**Translation:**

Shift the data in the X direction

$$T_x = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Shift the data in the Y direction:

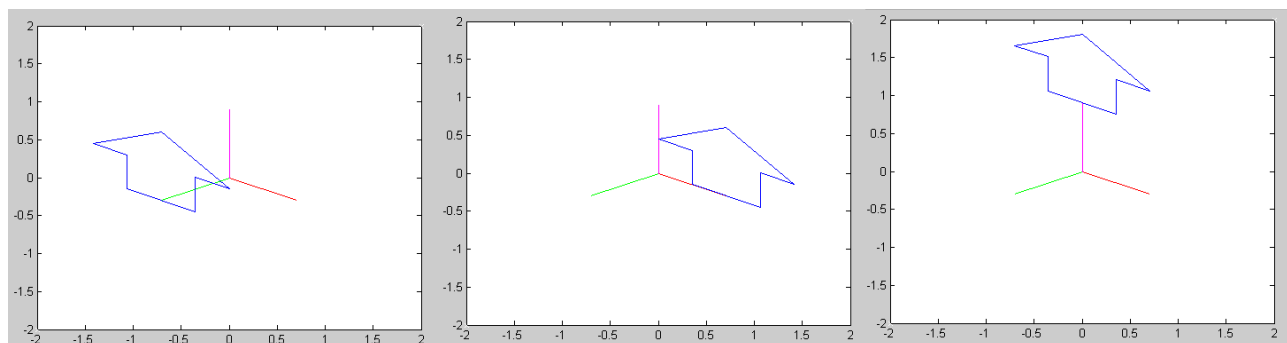
$$T_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Shift the data in the Z direction

$$T_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For example: Translate the arrow in the x, y, and z direction:

```
T = eye(4,4);
for i=0:100
    T(1,4) = i / 100;           % x
    Display3D(T*ARROW,Tdisp);
    pause(100);
end
```



Translation in the x, y, and z direction by one unit

Example: Where is the point (1, 2, 3) if you translate it by (4,5,6)?

```
>> P = [1;2;3;1]

      1
      2
      3
      1

>> T = eye(4);
>> T(1,4) = 4;
>> T(2,4) = 5;
>> T(3,4) = 6;
>> T

      1      0      0      4
      0      1      0      5
      0      0      1      6
      0      0      0      1

>> T*P

      5
      7
      9
      1
```

Answer: The point is now at (5, 7, 9)

## Translation Plus Rotation.

What happens if you combine a translation matrix plus a rotation matrix?

Note that matrix multiplication is not commutative: the order makes a difference. For example, define two matrices:

Tx is a rotation matrix about the X axis by 45 degrees

```
>> Tx = [1,0,0,0;0,c,-s,0;0,s,c,0;0,0,0,1]

      1.0000      0      0      0
      0      0.7071      0.7071      0
      0     -0.7071      0.7071      0
      0      0      0      1.0000
```

Tt is a translation matrix of (4, 5, 6)

```
>> Tt = T

      1      0      0      4
      0      1      0      5
      0      0      1      6
      0      0      0      1
```

If you translate then rotate, the net result is:

```
>> Tx*T
    1.0000    0    0    4.0000
    0    0.7071    0.7071    7.7782
    0   -0.7071    0.7071    0.7071
    0    0    0    1.0000
```

If you rotate then translate, then

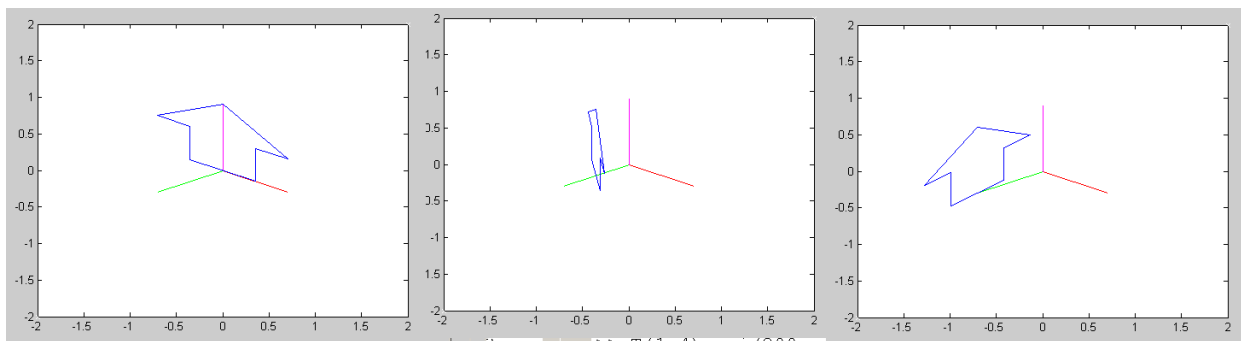
```
>> T*Tx
    1.0000    0    0    4.0000
    0    0.7071    0.7071    5.0000
    0   -0.7071    0.7071    6.0000
    0    0    0    1.0000
```

In Matlab, you can see this effect as follows:

- Rotate about the X axis while
- Translating about the Z axis:

```
c = cos(5*pi/180);
s = sin(5*pi/180);
Tx = [1,0,0,0;0,c,-s,0;0,s,c,0;0,0,0,1];
Ty = [c,0,s,0;0,1,0,0;-s,0,c,0;0,0,0,1];
Tz = [c,-s,0,0;s,c,0,0;0,0,1,0;0,0,0,1];
```

```
for i=1:200
    T = Tz ^ i;
    T(1,4) = i/200;
    Display3D(T*ARROW,Tdisp);
    pause(0.01);
end
```



The arrow spins about its Z-axis ( $T_z$ ) while translating along the x-axis

In other words, when you mix a translation and a rotation matrix:

- You translate (x, y, z) relative to the original axis, and then
- Rotate the object

