Chapter 6
Digital Mock-Up and 3D Animation for Robot Arms

6.1 Basic Surface Drawing and Data Structure in MATLAB™

In order to draw a multi-joint robot arm, the first thing that we must do is to know how to create surfaces of the 3D basic parts in MATLAB™. The 3D basic parts may include but are not limited to a cylinder and its variations, a sphere or an ellipsoid, a rectangular block, and a torus. Since in MATLAB™, each 3D part drawing is always initiated at the base coordinate frame, we need to be not only aware of how to create it, but also familiar of its data structure in order to modify it and/or send it to a desired destination (both position and orientation). The detailed instructions, procedures and examples can be found either in the MATLAB™ online help manual, or refer to the literature [1]–[4]. Let us here start with a surface of cylinder drawing.

MATLAB™ offers an internal function \[x, y, z\] = cylinder(r,n), where the first input dummy element \(r\) can be specified either by a single radius for creating a regular cylinder, or by an array for drawing a cylinder with variable radii. The second dummy element \(n\) is to specify the number of planar faces to form the entire side surface of the cylinder. The user is not allowed to specify a height of the cylinder and it is always 1 as default. The output coordinates \([x, y, z]\) are an \(m\) by \(n+1\) array for each of the \(x, y\) and \(z\), where \(m\) is the dimension of the radius array \(r\), which is at least 2 if \(r\) is a single constant for a regular cylinder. For example, \([x, y, z]\) = cylinder(1,10) creates a set of coordinate arrays \(x, y\) and \(z\) of 2 by 11 each for a cylinder side surface of radius equal to 1 and formed by 10 equal-size planar faces with a unity height, as shown in Figure 6.1. To draw this side surface, it must call a 3D surface drawing internal function surf(···) as follows:

\[
\begin{align*}
>> [x,y,z] &= \text{cylinder}(1,10); \\
>> \text{surf}(x,y,z,\text{'FaceColor','c','Facealpha',0.6, ...} \\
&\quad \text{'EdgeColor','r','AmbientStrength',0.4}),
\end{align*}
\]
grid, axis([-1.5 1.5 -1.5 1.5 0 2]), daspect([1 1 1]), grid, view([1 1 0.4]), light('Position',[1 -0.2 1],'Style','infinite'),

The internal function `surf(···)` in addition to accepting all the output coordinate arrays \( x, y, \) and \( z \) from the cylinder function, allows us to specify many drawing properties, including the face color, the degree of transparency in 'Facealpha' between 0 and 1 with smaller number more transparent, the edge color and the ambient strength. Under certain definitions of axis, view and lighting condition, a resulting semi-transparent cylindrical surface that is sitting on the base frame is given in Figure 6.1.

**Fig. 6.1** Data structure of a cylinder drawing in MATLAB\textsuperscript{T,M}

The data structure in each array of \( x, y, \) and \( z \) is defined in MATLAB\textsuperscript{T,M} by a certain counting rule: the number of rows is counted as a latitude of the cylindrical side surface with its bottom corresponding to the first row and its top corresponding to the last row in each of the \( x, y, \) and \( z \) arrays. The columns of each array count the vertices counterclockwise starting at the point \((1,0,0)\) and ending to the same point after a round trip to form the first row of each coordinate array as a longitudinal measure of the cylinder. Then, starting at \((1,0,1)\) and ending at the same point to form the second row for each of the 2 by 11 arrays \( x, y, \) and \( z \).

In Figure 6.1 the number without parenthesis indicates the counting order of the vertices as a longitude, and the number with parenthesis counts the latitude from bottom up. For instance, if the radius \( r = 1 \) is replaced by an array \( r = [0.5, 1, 0.5, 0] \) and \( n \) remains 10 inside the cylinder function, then the