第 3 章

位　　姿

位姿即位置与姿态。一个模型在空间中具有不同的位姿，在背景画面中改变模型的位姿就可以实现模型移动和旋转等效果。

3.1　位姿在不同坐标系下的数学表述

3.1.1　球面角

球面角用于描述物体在三维空间中的旋转角度。在 xOz 球面坐标系中，球面角描述物体的旋转角度的规则如下：

(1) 物体相对于 Oz 轴的旋转角为 colatitude 角或 elevation 角，旋转角度为 θ，范围是 $[0,\pi]$。

(2) 物体相对于 Ox 轴的旋转角为 azimuth 角(也叫 longitude 角)，旋转角度为 φ，范围是 $[0,2\pi)$。

由这两个相对的旋转角度组成的角(θ,φ)即为球面角。例如，球面角$(\pi,1.5\pi)$可以表示物体沿 Oz 轴旋转 π 弧度并且沿 Ox 轴旋转 1.5π 弧度。

注意：colatitude 角和 elevation 角互为余角，二者不是同一个概念。

3.1.2　球面坐标

球面坐标在球面角的基础上增加了距离的概念。物体到球面坐标系原点的距离为 ρ，由球面角的两个相对的旋转角度和物体到原点的距离组成的坐标(θ,φ,ρ)即为球面坐标。例如，球面坐标$(\pi,1.5\pi,1)$ 可以表示物体沿 Oz 轴旋转 π 弧度、沿 Ox 轴旋转 1.5π 弧度并且到球面坐标系原点的距离是 1。

3.1.3　欧拉角

欧拉角用于描述物体在三维空间中的旋转角度。在 XYZ 三维坐标系中，欧拉角描述物体的旋转角度的规则如下：

(1) 物体相对于 x 轴的旋转角为 roll 角,旋转角度为 φ,范围是[−180,180]。

(2) 物体相对于 y 轴的旋转角为 pitch 角,旋转角度为 θ,范围是[−90,90]。

(3) 物体相对于 z 轴的旋转角为 yaw 角,旋转角度为 φ,范围是[−180,180]。

由这 3 个相对的旋转角度组成的角(φ,θ,ψ)(或采用(ψ,θ,φ)等格式)即为欧拉角,其中前者是在几何学中更为常见的格式,而后者是计算机代码支持的其他格式。根据欧拉角的格式不同,欧拉角(10,20,30) 可以表示物体沿 x 轴旋转 30°、沿 y 轴旋转 20°并且沿 z 轴旋转 10°,也可以表示物体沿 x 轴旋转 10°、沿 y 轴旋转 20°并且沿 z 轴旋转 30°等。

3.1.4 RPY 角

RPY 角是一种欧拉角的表述,专指(ψ,θ,φ)格式的欧拉角。RPY 角在计算机中是最常用的欧拉角。

3.2 计算几何相关知识

3.2.1 两点求角度

已知一条直线上的两个点可以求出这条直线的方位角,即这条直线和水平线的夹角。求得角度的范围是[0,2π)。

3.2.2 三点求角度

已知两条直线上的不重合的两个点和重合的第 3 个点可以求出这两条直线的夹角。求得角度的范围是[0,2π)。

3.3 力学相关知识

3.3.1 质点

物体可以被视为由质点代替而成。质点是有一定质量的一个点。当只考虑模型的位置而不考虑模型的姿态时推荐使用质点模型代替物体。

3.3.2 质点系

物体可以被视为由质点系代替而成。许多质点的集合称为质点系。在所讨论的实际问题中,当一个质量连续分布的物体不能被视为质点时,可以将它划为许多微小部分,认为每个微小部分中粒子的运动情况完全相同,从而可以将每个微小部分简化为一个质点,而整个物体可被视为许多质点的集合,所以,质量连续分布的物体也可以被视为质点系。

3.3.3 质心

质心是质量的中心。根据质点系中的每个质点的质量和位置,可以算得质心的质量和

位置。

3.3.4 质心运动定理

质点系的质心运动和一个位于质心的质点的运动相同,该质点的质量等于质点系的总质量,而该质点上的作用力则等于作用于质点系上的所有外力平行地移到这一点上。根据这个定理可得到推论如下:

(1) 质点系的内力不能影响它的质心的运动。例如跳水运动员自跳板起跳后,不论他在空中做何种动作或采取何种姿势,由于外力(重力)并未改变,所以运动员的质心在入水前仍沿抛物线轨迹运动。

(2) 如果作用于质点系上外力的主矢始终为0,则质点系的质心进行匀速直线运动或保持静止。

(3) 若作用于质点系上外力的主矢在某轴上的投影始终为0,则质点系的质心在该轴上的坐标匀速变化或保持不变。

3.3.5 刚体

刚体指的是在运动中和受力作用后,形状和大小不变,而且内部各点的相对位置不变的物体。

绝对刚体实际上是不存在的,只是一种理想模型,因为任何物体在受力作用后都或多或少地变形,如果变形的程度相对于物体本身几何尺寸来讲极为微小,在研究物体运动时变形就可以忽略不计。

把许多固体视为刚体,所得到的结果在工程上一般已有足够的准确度,但要研究应力和应变,则必须考虑变形。由于变形一般总是微小的,所以可先将物体当作刚体,用理论力学的方法求得加给它的各未知力,然后用变形体力学,包括材料力学、弹性力学、塑性力学等的理论和方法进行研究。

注意:有些物体不能视为刚体,例如地球。

3.4 旋转矩阵

3.4.1 旋转矩阵的用法

旋转矩阵用于求得一个点从一个坐标系转换到另一个坐标系后的坐标。将一个模型看作一组点,然后将旋转矩阵左乘这一组点中的每个点即可得到旋转后的点的位置,旋转后的点即可描述旋转后的模型。将点(x,y)旋转到$(2x+3y,4x+5y)$的位置上,旋转矩阵为

$$\begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}$$

将点(x,y,z)旋转到$(2x+3y+4z,5x+6y+7z,8x+9y+10z)$的位置上，旋转矩阵为

$$\begin{bmatrix}2 & 3 & 4\\5 & 6 & 7\\8 & 9 & 10\end{bmatrix}$$

三维空间内的旋转矩阵至少需要3个自由度，而拥有3个自由度的旋转矩阵的尺寸是3×3。将3×3的旋转矩阵写为齐次矩阵后，其尺寸变为4×4。齐次旋转矩阵用于在齐次方程中进行旋转变换，在齐次方程中进行旋转变换可便于求解。

将三维旋转矩阵[2 3 4;5 6 7;8 9 10]和平移向量[11;12;13]写作齐次旋转矩阵为

$$\begin{bmatrix}2 & 3 & 4 & 11\\5 & 6 & 7 & 12\\8 & 9 & 10 & 13\\0 & 0 & 0 & 1\end{bmatrix}$$

可以看出，齐次矩阵不仅可以表示旋转关系，还可以表示平移关系。如果一个齐次旋转矩阵只表示旋转，则平移向量为0向量。将三维旋转矩阵[2 3 4;5 6 7;8 9 10]和平移向量[0;0;0]写作齐次旋转矩阵为

$$\begin{bmatrix}2 & 3 & 4 & 0\\5 & 6 & 7 & 0\\8 & 9 & 10 & 0\\0 & 0 & 0 & 1\end{bmatrix}$$

3.4.2 欧拉角与旋转矩阵的变换

调用eulerAnglesToRotation3d()函数可以将RPY角转换为齐次旋转矩阵。

eulerAnglesToRotation3d()函数需要传入3个参数，此时这3个参数是RPY角的3个分量，然后将返回齐次旋转矩阵。将RPY角(10,20,30)转换为齐次旋转矩阵的代码如下：

```
>> eulerAnglesToRotation3d(10, 20, 30)
ans =

   0.9254   0.0180   0.3785        0
   0.1632   0.8826  -0.4410        0
  -0.3420   0.4698   0.8138        0
        0        0        0   1.0000
```

此外，调用eulerAnglesToRotation3d()函数还可以将其他格式的欧拉角转换为齐次旋转矩阵。例如，指定欧拉角的格式为YZX并将欧拉角(10,20,30)转换为齐次旋转矩阵的代码如下：

```
>> eulerAnglesToRotation3d(10, 20, 30, 'YZX')
ans =
```

```
    0.9254   -0.2049    0.3188         0
    0.3420    0.8138   -0.4698         0
   -0.1632    0.5438    0.8232         0
         0         0         0    1.0000
```

此外，调用 rotation3dToEulerAngles()函数还可以将齐次旋转矩阵或非齐次旋转矩阵转换为欧拉角。例如，指定欧拉角的格式为 YZX 并将齐次旋转矩阵[0.1 0.2 0.3 0;0.4 0.5 0.6 0;0.7 0.8 0.9 0;0 0 0 1] 转换为欧拉角的代码如下：

```
>> [y, z, x] = rotation3dToEulerAngles([0.1 0.2 0.3 0; 0.4 0.5 0.6 0; 0.7 0.8 0.9 0;
0 0 0 1], 'YZX')
```

3.4.3　根据旋转角度创建旋转矩阵

调用 createRotation3dLineAngle()函数可以在点沿着一个转轴旋转的场景下创建旋转矩阵。createRotation3dLineAngle()函数允许传入两个参数，此时第 1 个参数是转轴的 x 坐标和 y 坐标，第 2 个参数是点沿着转轴旋转的角度，然后将返回齐次旋转矩阵。例如将点沿着转轴的 x 坐标为[1 2 3]且 y 坐标为[4 5 6]并旋转 0.5π 弧度，转换为欧拉角的代码如下：

```
>> createRotation3dLineAngle([[1 2 3], [4 5 6]], 0.5 * pi)
ans =

    0.2078   -0.4240    0.8815   -1.0042
    0.9435    0.3247   -0.0662    0.6058
   -0.2581    0.8455    0.4675    0.1646
         0         0         0    1.0000
```

3.4.4　根据旋转矩阵计算转轴或旋转角度

调用 rotation3dAxisAndAngle()函数可以通过齐次旋转矩阵计算转轴或旋转角度。rotation3dAxisAndAngle()函数允许传入一个参数，此时这个参数是齐次旋转矩阵，然后将返回转轴。通过齐次旋转矩阵[0.2078 −0.4240 0.8815 −1.0042; 0.9435 0.3247 −0.0662 0.6058; −0.2581 0.8455 0.4675 0.1646; 0 0 0 1.0000]计算转轴的代码如下：

```
>> rot_matrix = [0.2078 -0.4240 0.8815 -1.0042; 0.9435 0.3247 -0.0662 0.6058;
                 -0.2581 0.8455 0.4675 0.1646; 0 0 0 1.0000];
>> rotation3dAxisAndAngle(rot_matrix)
ans =

    0.7380    1.6725    2.6070    0.4558    0.5698    0.6838
```

此外，如果指定了两个返回参数，则 rotation3dAxisAndAngle()函数将同时计算转轴和旋转角度。通过齐次旋转矩阵[0.2078 −0.4240 0.8815 −1.0042;0.9435 0.3247 −0.0662 0.6058;−0.2581 0.8455 0.4675 0.1646; 0 0 0 1.0000]计算转轴和旋转角度的代码如下：

```
>> rot_matrix = [0.2078 -0.4240 0.8815 -1.0042; 0.9435 0.3247 -0.0662 0.6058; -0.
2581 0.8455 0.4675 0.1646; 0 0 0 1.0000];
>> [rot_axis, angle] = rotation3dAxisAndAngle(rot_matrix)
rot_axis =

   0.7380   1.6725   2.6070   0.4558   0.5698   0.6838

angle = 1.5708
```

3.5 仿射变换

仿射变换也称仿射投影。对一个向量空间进行线性变换并接上一个平移变换，最终变换为另一个向量空间的变换即为仿射变换，所以仿射变换其实也就是如何进行两个向量空间的变换。

仿射变换也需要构造对应的变换矩阵，并且变换矩阵的维数和模型的维数有关。如果对二维模型进行仿射变换，则需要构造一个 3×3 的变换矩阵；如果对三维模型进行仿射变换，则需要构造一个 4×4 的变换矩阵。

对一个模型进行仿射变换的本质是对模型中的所有点均进行仿射变换，进行仿射变换之后的点的集合即为模型进行仿射变换之后的结果。

3.5.1 平移变换

将点(x,y)平移到$(x+\mathrm{tx},y+\mathrm{ty})$的位置上，变换矩阵为

$$\begin{bmatrix}1 & 0 & \mathrm{tx}\\ 0 & 1 & \mathrm{ty}\\ 0 & 0 & 1\end{bmatrix}$$

3.5.2 缩放变换

将点(x,y)缩放为$(\mathrm{sx},\mathrm{sy})$，变换矩阵为

$$\begin{bmatrix}\mathrm{sx} & 0 & 0\\ 0 & \mathrm{sy} & 0\\ 0 & 0 & 1\end{bmatrix}$$

当 sx 等于 sy 时，缩放变换也称为尺度缩放；当 sx 不等于 sy 时，缩放变换也称为拉伸变换。

3.5.3 剪切变换

剪切变换也称为错切变换。一个正方形可以通过剪切变换变换为平行四边形或菱形。其余的几何形状可以用正方形进行参考。

将点(x,y)剪切为(shx,y),横向剪切变换的变换矩阵为

$$\begin{bmatrix}1 & \mathrm{shx} & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

将点(x,y)剪切为(x,shy),纵向剪切变换的变换矩阵为

$$\begin{bmatrix}1 & 0 & 0\\ \mathrm{shy} & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

将点(x,y)剪切为$(\mathrm{shx},\mathrm{shy})$,相当于一个横向剪切与一个纵向剪切的复合,变换矩阵为

$$\begin{bmatrix}1 & \mathrm{shx} & 0\\ \mathrm{shy} & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

3.5.4 旋转变换

将点围绕原点逆时针旋转θ弧度,变换矩阵为

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

将点围绕轴心(x,y)逆时针旋转θ弧度,变换矩阵为

$$\begin{bmatrix}\cos\theta & -\sin\theta & x-x\cos\theta+y\sin\theta\\ \sin\theta & \cos\theta & y-x\sin\theta-y\cos\theta\\ 0 & 0 & 1\end{bmatrix}$$

3.5.5 仿射变换矩阵的尺寸描述

仿射变换矩阵在描述尺寸时有额外的规则。虽然仿射矩阵在数学意义上都是$(m+1)\times(n+1)$的矩阵,但为了更直观地描述仿射变换矩阵的变换因子,在描述仿射变换矩阵的尺寸时可能采用不同于数学意义的描述。

下面的仿射变换矩阵被称为2×2的仿射变换矩阵。

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

这是因为该矩阵的变换因子是一个2×2的矩阵。该矩阵的变换因子如下:

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

下面的仿射变换矩阵被称为2×3的仿射变换矩阵。

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 0 & 0 & 1 \end{bmatrix}$$

这是因为该矩阵的变换因子是一个 2×3 的矩阵。该矩阵的变换因子如下：

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

3.6 Octave 的空间变换函数

空间变换函数包括仿射变换和图像裁剪等函数，常用于更改模型的效果。Octave 的大部分空间变换函数包含在 image 工具箱中，所以如果要使用 Octave 的空间变换函数，则推荐先安装 image 工具箱。

3.6.1 安装 image 工具箱

安装 image 工具箱的命令如下：

```
>> pkg install -forge image
```

3.6.2 实例化仿射变换对象

调用 affine2d()函数可以实例化二维仿射变换对象。调用 affine3d()函数可以实例化三维仿射变换对象。二者需要传入一个参数，此时这个参数是仿射变换的变换矩阵，然后将返回仿射变换对象。以仿射矩阵[1 0 0;0 1 0;0 0 1]实例化二维仿射变换对象的代码如下：

```
>> affine2d([1 0 0; 0 1 0; 0 0 1])
ans =

  affine2d with properties:

             T: [3x3 double]
  Dimensionality: 2
```

以仿射矩阵[1 0 0 0;0 1 0 0;0 0 1 0;0 0 0 1]实例化三维仿射变换对象的代码如下：

```
>> affine3d([1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1])
ans =

  affine3d with properties:

             T: [4x4 double]
  Dimensionality: 3
```

仿射变换对象提供了多个仿射变换方法。二维仿射变换对象提供的仿射变换方法如

表 3-1 所示。

表 3-1 二维仿射变换对象提供的仿射变换方法

方 法 名	用 法	含 义
invert	invert(tform)	进行 affine2d()函数的逆变换
isRigid	isRigid(tform)	检查当前仿射变换是否只代表旋转或平移
isSimilarity	isSimilarity(tform)	检查当前仿射变换是否只代表均匀缩放、旋转、反射或平移
isTranslation	isTranslation(tform)	检查当前仿射变换是否只代表平移
outputLimits	outputLimits(tform,xlims,ylims)	给定以 xlims 和 ylims 代表的边界坐标(左上、右下),然后在仿射变换后返回新的边界坐标
transformPointsForward	transformPointsForward(tform,u,v); transformPointsForward(tform,U)	在 uv 点集(1×n 矩阵)上进行仿射变换; 在 U 点集(2×n 矩阵)上进行仿射变换
transformPointsInverse	transformPointsInverse(tform,u,v); transformPointsInverse(tform,U)	在 uv 点集(1×n 矩阵)上进行仿射变换的逆变换; 在 U 点集(2×n 矩阵)上进行仿射变换的逆变换

三维仿射变换对象提供的仿射变换方法如表 3-2 所示。

表 3-2 三维仿射变换对象提供的仿射变换方法

方 法 名	用 法	含 义
invert	invert(tform)	进行 affine3d()函数的逆变换
isRigid	isRigid(tform)	检查当前仿射变换是否只代表旋转或平移
isSimilarity	isSimilarity(tform)	检查当前仿射变换是否只代表均匀缩放、旋转、反射或平移
isTranslation	isTranslation(tform)	检查当前仿射变换是否只代表平移
outputLimits	outputLimits(tform,xlims,ylims,zlims)	给定以 xlims、ylims 和 zlims 代表的边界坐标(左上前部、右下后部),然后在仿射变换后返回新的边界坐标
transformPointsForward	transformPointsForward(tform,u,v,w); transformPointsForward(tform,U)	在 uvw 点集(1×n 矩阵)上进行仿射变换; 在 U 点集(3×n 矩阵)上进行仿射变换
transformPointsInverse	transformPointsInverse(tform,u,v,w); transformPointsInverse(tform,U)	在 uvw 点集(1×n 矩阵)上进行仿射变换的逆变换; 在 U 点集(3×n 矩阵)上进行仿射变换的逆变换

3.6.3 根据仿射变换对象进行仿射变换

调用 tformfwd()函数可以根据仿射变换对象进行仿射变换。tformfwd()函数允许传入两个参数调用，此时第 1 个参数是仿射变换对象，第 2 个参数是 $n\times1$ 的 x 坐标和 $n\times1$ 的 y 坐标合并而成的 $n\times2$ 的矩阵。

此外，tformfwd()函数允许传入 3 个参数调用，此时第 1 个参数是仿射变换对象，第 2 个参数是 $n\times1$ 的 x 坐标，第 3 个参数是 $n\times1$ 的 y 坐标。

3.6.4 根据仿射变换对象进行仿射变换的逆变换

调用 tforminv()函数可以根据仿射变换对象进行仿射变换的逆变换。tforminv()函数的用法和 tformfwd()函数的用法类似，因此不再赘述。

3.6.5 推断仿射变换矩阵

调用 cp2tform()函数可以根据两个点集推断出仿射变换矩阵。cp2tform()函数需要传入 3 个参数，此时第 1 个参数是仿射变换前的点集，第 2 个参数是仿射变换后的点集，第 3 个参数是转换参数，然后将返回仿射变换矩阵。cp2tform()函数返回的仿射变换矩阵是推断出的仿射变换矩阵，用这个对象可以按照相同的仿射变换方式进行其他点集的仿射变换。

注意：cp2tform()函数只能用于二维仿射变换矩阵的推断。

cp2tform()函数支持多种转换参数，用于返回不同类型的仿射变换矩阵。cp2tform()函数支持的转换参数如表 3-3 所示。

表 3-3 cp2tform()函数支持的转换参数

转换参数	含　义	在指定此转换参数后返回的仿射变换矩阵的用法	在指定此转换参数后返回的仿射变换矩阵的用途
affine	返回仿射变换矩阵和仿射变换的逆变换矩阵(T.tdata.T 和 T.tdata.Tinv)； 变换系数是 3×2 矩阵	IN_CP＝[OUT_CP ones(rows(out_cp),1)]＊T.tdata.Tinv; OUT_CP＝[IN_CP ones(rows(in_cp),1)]＊T.tdata.T	适用于当一个空间中的平行线在另一个空间(例如剪切、平移等)中仍然平行时的仿射变换
nonreflective similarity	T 和 Tinv 具有 Tcoefs＝[a－b; b a; c d]的形式，其他和 affine 转换参数相同	IN_CP＝[OUT_CP ones(rows(out_cp),1)]＊T.tdata.Tinv; OUT_CP＝[IN_CP ones(rows(in_cp),1)]＊T.tdata.T	适用于旋转、缩放和平移； 不适用于反射
similarity	T 和 Tinv 不仅具有 Tcoefs＝[a－b;b a;c d]的形式，还具有 Tcoefs＝[a b; b－a; c d]的形式，其他和 nonreflective similarity 转换参数相同	IN_CP＝[OUT_CP ones(rows(out_cp),1)]＊T.tdata.Tinv; OUT_CP＝[IN_CP ones(rows(in_cp),1)]＊T.tdata.T	适用于反射、旋转、缩放和平移；如果在指定 nonreflective similarity 转换参数后返回的仿射变换矩阵拟合更好，则会输出警告

续表

转换参数	含　义	在指定此转换参数后返回的仿射变换矩阵的用法	在指定此转换参数后返回的仿射变换矩阵的用途
projective	返回仿射变换矩阵和仿射变换的逆变换矩阵(T.tdata.T和T.tdata.Tinv);变换系数是3×3矩阵	[u v w]=[OUT_CP ones(rows(out_cp),1)] * T.tdata.Tinv; IN_CP=[u./w,v./w]; [x y z]=[IN_CP ones(rows(in_cp),1)] * T.tdata.T; OUT_CP=[x./z y./z];	适用于当一个空间中的平行线都向另一个空间的消失点收敛时的仿射变换
polynomial	追加OPT参数作为拟合的阶数; OPT必须为2、3或4,对应的输入控制点数至少为6、10或15; 结构体T中仅包含逆变换(输出空间到输入空间)得到的对应的输入点集; x和y为输出点集,u、v为输入点集	2阶: [u v]=[1 x y x * y x^2 y^2] * T.tdata.Tinv; 3阶: [u v]=[1 x y x * y x^2 y^2 y * x^2 x * y^2 x^3 y^3] * T.tdata.Tinv; 4阶: [u v]=[1 x y x * y x^2 y^2 y * x^2 x * y^2 x^3 y^3 x^3 * y x^2 * y^2 x * y^3 x^4 y^4] * T.tdata.Tinv	适用于求逆变换后得到的点集

以仿射变换前的点集[1 2;10 10]、仿射变换后的点集[2 4;15 15]、转换参数为projective,推断仿射变换矩阵的代码如下:

```
>> cp2tform([1 2; 10 10], [2 4; 15 15], 'projective')
ans =

  scalar structure containing the fields:

    ndims_in = 2
    ndims_out = 2
    forward_fcn = @fwd_projective
    inverse_fcn = @inv_projective
    tdata =

      scalar structure containing the fields:

        T =

           0.192926  -0.017632   0.608814
           0.149892   0.327611  -0.686089
          -0.019438   0.308955   1.000000
```

```
Tinv =

   3.8937   1.4846  -1.3520
  -0.9854   1.4776   1.6137
   0.3801  -0.4276   0.4752
```

3.6.6 裁剪图像函数

调用 imcrop()函数可以裁剪二维图像。imcrop()函数允许不传入参数调用,此时 imcrop()函数使用交互模式裁剪图像。如果不传入参数调用 imcrop()函数,则 Octave 必须先打开一个包含图像对象的窗口,否则 imcrop()函数将报错如下:

```
>> imcrop(image)
error: get: unknown image property currentaxes
error: called from
    imcrop at line 150 column 9
```

此外,imcrop()函数允许传入一个参数,此时 imcrop()函数使用交互模式裁剪图像,这个参数是图像矩阵或图像对象的句柄。

此外,imcrop()函数允许传入两个参数,此时 imcrop()函数使用交互模式裁剪图像,第 1 个参数是索引图像矩阵,第 2 个参数是索引。

在交互模式下,用户需要进入图像所在的画布,单击画布中的一点作为裁剪区域的起点,拖动鼠标以选定裁剪区域,然后松开鼠标以完成图像裁剪操作。

如果已知需要裁剪的区域的具体坐标范围,则可以指定需要裁剪的区域作为参数。imcrop()函数允许追加传入坐标范围参数,此时 imcrop()函数使用非交互模式裁剪图像。指定坐标范围为[1 1 100 100]并裁剪图片的代码如下:

```
>> imcrop(img, [1 1 100 100])
```

在非交互模式下,用户无须用鼠标选定裁剪位置,这是更加精确的裁剪模式。

3.6.7 缩放图像函数

调用 imresize()函数可以缩放图像。imresize()函数至少需要传入两个参数调用,此时第 1 个参数是图像矩阵,第 2 个参数是缩放倍数。当指定缩放的倍数是 1.5 时,表示图像的宽度和高度同时变为原值的 1.5 倍并缩放图像的代码如下:

```
>> imresize(img, 1.5)
```

此外,第 2 个参数还可以是缩放的宽度和高度组成的矩阵。指定缩放的宽度是 100 且高度为 200 并缩放图像的代码如下:

```
>> imresize(img, [100 200])
```

imresize()函数还允许追加传入第 3 个参数,此时这个参数是插值方式。imresize()函数支持的插值方式如表 3-4 所示。

表 3-4　imresize()函数支持的插值方式

插值方式	含　义
nearest	最近邻插值
box	
linear	线性插值
bilinear	双线性插值
triangle	三角插值
cubic	三次插值
bicubic	双三次插值

此外，在 imresize()函数中，bicubic 插值方式实际上等效于 cubic 插值方式，而 linear 插值方式和 triangle 插值方式实际上等效于 bilinear 插值方式。

此外，imresize()函数默认采用 bilinear 插值方式。

imresize()函数还允许追加传入键-值对形式的参数。imresize()函数支持的键-值对形式的参数如表 3-5 所示。

表 3-5　imresize()函数支持的键-值对形式的参数

键参数	默　认　值	含　义
Antialiasing	当插值方式不为 nearest 和 box 时，默认值为 true；否则默认值为 false	如果值为 true，并且图像的宽高比小于 1，则在该方向上启用抗锯齿； 此时插值核被扩展到 1/＜值＞以减少导致混叠效应的频率分量，因此启用抗锯齿将需要使用更多的邻点； 例如以缩放倍数为 0.5、启用双线性插值＋抗锯齿的参数缩放图像就要用到 16 个邻点
Method	—	插值方式； 可以是字符串或插值核
OutputSize	—	表示输出图像的宽度和高度
Scale	—	缩放倍数或缩放因子； 可以是数字或数字矩阵

3.6.8　旋转图像函数

调用 imrotate()函数可以按照给定角度旋转一个二维图像。imrotate()函数需要传入 5 个参数调用，第 1 个参数是黑白图像或灰阶图像的图像矩阵，第 2 个参数是旋转角度，第 3 个参数是插值方式，第 4 个参数是图像结果的裁剪方式，第 5 个参数是变换后的图像的其他部分的值。

imrotate()函数支持的插值方式如表 3-6 所示。

表 3-6 imrotate()函数支持的插值方式

插值方式	含 义	插值方式	含 义
nearest	最近邻插值	pchip	分段三次 Hermite 插值
linear	线性插值	cubic	三次插值
bilinear	双线性插值	bicubic	双三次插值
triangle	三角插值	fourier	傅里叶插值

此外,在 imrotate()函数中,bicubic 插值方式实际上等效于 cubic 插值方式,而 bilinear 插值方式和 triangle 插值方式实际上等效于 linear 插值方式。

此外,imrotate()函数默认采用 nearest 插值方式。

此外,生成的图像在默认情况下包含整个旋转变换后的图像。修改图像结果的裁剪方式可以令生成的图像在裁剪后只包含一部分旋转变换后的图像。imrotate()函数支持的裁剪方式如表 3-7 所示。

表 3-7 imrotate()函数支持的裁剪方式

裁剪方式	含 义
loose	生成的图像包含整个旋转变换后的图像
crop	生成的图像只包含旋转变换后的图像的中心部分

此外,imrotate()函数默认采用 loose 裁剪方式。

此外,第 5 个参数默认为 0;如果使用了 fourier 插值方式,则将忽略第 5 个参数。

3.6.9 快速旋转和缩放图像函数

调用 rotate_scale()函数可以对黑白图像或灰阶图像通过快速双线性插值方式进行快速旋转和缩放。rotate_scale()函数需要传入 4 个参数调用,第 1 个参数是图像矩阵,第 2 个参数是快速旋转和缩放前的特征点,第 3 个参数是快速旋转和缩放后的特征点,第 4 个参数是输出图像矩阵的尺寸。特征点参数是 2×2 的矩阵,其中第 1 行包含特征点的 x 坐标且第 2 行包含特征点的 y 坐标。

3.6.10 透视变换函数

调用 imperspectivewarp()函数可以对二维图像进行透视变换。imperspectivewarp()函数需要传入 5 个参数调用,第 1 个参数是图像矩阵,第 2 个参数是变换矩阵,第 3 个参数是插值方式,第 4 个参数是图像结果的裁剪方式,第 5 个参数是变换后的图像的其他部分的值。变换矩阵 $\boldsymbol{P}$ 必须是 3×3 齐次矩阵、2×2 仿射变换矩阵或 2×3 仿射变换矩阵。

imperspectivewarp()函数支持的插值方式如表 3-8 所示。

此外,在 imperspectivewarp()函数中,bicubic 插值方式实际上等效于 cubic 插值方式,而 bilinear 插值方式和 triangle 插值方式实际上等效于 linear 插值方式。

表 3-8 imperspectivewarp()函数支持的插值方式

插值方式	含 义	插值方式	含 义
nearest	最近邻插值	pchip	分段三次 Hermite 插值
linear	线性插值	cubic	三次插值
bilinear	双线性插值	bicubic	双三次插值
triangle	三角插值	spline	样条插值

此外,imperspectivewarp()函数默认采用 linear 插值方式。

此外,生成的图像在默认情况下包含整个透视变换后的图像。修改图像结果的裁剪方式可以令生成的图像在裁剪后只包含一部分透视变换后的图像。imperspectivewarp()函数支持的裁剪方式如表 3-9 所示。

表 3-9 imperspectivewarp()函数支持的裁剪方式

裁剪方式	含 义
loose	生成的图像包含整个透视变换后的图像
crop	生成的图像只包含透视变换后的图像的中心部分
same	生成的图像只包含原来的坐标轴范围内的图像,并且不在变换后更改坐标轴的参数

此外,imperspectivewarp()函数默认采用 loose 裁剪方式。

3.6.11 高斯金字塔函数

调用 impyramid()函数可以绘制图像上一级或下一级的高斯金字塔。impyramid()函数需要传入两个参数调用,第 1 个参数是图像矩阵,第 2 个参数是变换方向。变换方向必须是 reduce 或 expand,其中 reduce 代表向下采样,用于绘制下一级的高斯金字塔;expand 代表向上采样,用于绘制上一级的高斯金字塔。

注意:如果图像的维数超过 2,则只有图像的前二维会改变尺寸。

绘制图像上一级的高斯金字塔的代码如下:

```
>> impyramid(img, 'expand')
```

3.6.12 重新映射图像函数

调用 imremap()函数可以对二维图像进行透视变换。imremap()函数允许传入 3 个参数调用,第 1 个参数是可以返回图像矩阵的函数 IM(),第 2 个参数是 x 轴方向的映射函数 XI(),第 3 个参数是 y 轴方向的映射函数 YI(),然后 imremap()函数将采用 IM(YI(y,x),XI(y,x))方式重新映射并返回新的图像矩阵。

此外,imremap()函数允许追加传入第 4 个参数,此时这个参数是插值方式。插值方式

默认为 linear。

此外，imremap()函数允许追加传入第 5 个参数，此时这个参数是变换后的图像的其他部分的值。这个值默认为 0。

此外，在 imremap()函数中，bicubic 插值方式实际上等效于 cubic 插值方式，而 bilinear 插值方式和 triangle 插值方式实际上等效于 linear 插值方式。

此外，如果在调用 imremap()函数时指定了 bicubic 插值方式，并且追加传入了第 5 个参数，则此时这个参数是填充方式。imremap()函数支持的填充方式如表 3-10 所示。

表 3-10　imremap()函数支持的填充方式

填充方式	含　义
circular	以圆形区域进行填充
replicate	以复制边界值的方式进行填充
symmetric	以镜面反射的效果进行填充
reflect	和 symmetric 类似，但不使用边界进行填充，因此这种填充方式不适合在 $n\times1$ 图像尺寸场景下的填充

3.6.13　剪切变换函数

调用 imshear()函数可以对二维图像进行剪切变换。imshear()函数需要传入 4 个参数调用，第 1 个参数是图像矩阵或图像对象的句柄，第 2 个参数是剪切方向，第 3 个参数是剪切的坡度，第 4 个参数是图像结果的裁剪方式。剪切方向必须定义为 x 或 y。当剪切方向是 x 时，图像将沿 x 轴方向剪切坡度像素；当剪切方向是 y 时，图像将沿 y 轴方向剪切坡度像素。

注意：剪切的坡度允许不是整数。

此外，生成的图像在默认情况下包含整个剪切变换后的图像。修改图像结果的裁剪方式可以令生成的图像在裁剪后只包含一部分剪切变换后的图像。imshear()函数支持的裁剪方式如表 3-11 所示。

表 3-11　imshear()函数支持的裁剪方式

裁剪方式	含　义
loose	生成的图像包含整个剪切变换后的图像
crop	生成的图像只包含剪切变换后的图像的中心部分
wrap	生成的图像保持原有尺寸不变，但不裁剪坐标轴之外的图像部分，而是将这部分图像包裹回图像中

此外，imshear()函数默认采用 loose 裁剪方式。

3.6.14 平移变换函数

调用 imtranslate()函数可以对二维图像通过快速傅里叶插值方式进行平移变换。

imtranslate()函数需要传入 4 个参数调用,第 1 个参数是图像矩阵,第 2 个参数是沿 x 轴方向的平移量,第 3 个参数是沿 y 轴方向的平移量,第 4 个参数是图像结果的裁剪方式。当第 2 个参数是 1 时,图像将沿 x 轴方向平移 1 像素;当第 3 个参数是 -2 时,图像将沿 y 轴方向平移 -2 像素,即沿 y 轴反方向平移 2 像素。

此外,生成的图像在默认情况下包含整个平移变换后的图像。修改图像结果的裁剪方式可以令生成的图像在裁剪后只包含一部分平移变换后的图像。imtranslate()函数支持的裁剪方式如表 3-12 所示。

表 3-12 imtranslate()函数支持的裁剪方式

裁剪方式	含 义
crop	生成的图像只包含平移变换后的图像的中心部分
wrap	生成的图像保持原有尺寸不变,但不裁剪坐标轴之外的图像部分,而是将这部分图像包裹回图像中

此外,imtranslate()函数默认采用 wrap 裁剪方式。

3.7 ImageMagick 的空间变换命令

3.7.1 -resize 参数

在 convert 命令中指定-resize 参数可以缩放图像。

1. 保持宽高比

将三维机器人模型图片 out.gif 缩放到 64×64 大小并生成新的图片 new_out.gif 的命令如下:

```
$ convert out.gif -resize 64x64 new_out.gif
```

此时,如果模型图片的宽高比不等于指定的宽高比 64∶64,则图片不会被拉伸,而只是将长边缩放到 64 像素并将短边按比例缩放,以保持图片原有的宽高比。

2. 将图片拉伸到指定的大小

将三维机器人模型图片 out.gif 缩放到 64×64 大小,拉伸并生成新的图片 new_out.gif 的命令如下:

```
$ convert out.gif -resize 64x64\! new_out.gif
```

此时,如果模型图片的宽高比不等于指定的宽高比 64∶64,则图片会被缩放到 64×64 大小,改变了图片原有的宽高比。

3. 仅缩小大尺寸的图片

将三维机器人模型图片 out.gif 缩放到 64×64 大小，仅缩小大尺寸的图片并生成新的图片 new_out.gif 的命令如下：

```
$ convert out.gif -resize 64x64\> new_out.gif
```

此时，如果模型图片的宽或高大于 64 像素，则图片会被缩放到 64×64 大小，否则不缩放图片。

4. 仅扩大小尺寸的图片

将三维机器人模型图片 out.gif 缩放到 64×64 大小，仅扩大小尺寸的图片并生成新的图片 new_out.gif 的命令如下：

```
$ convert out.gif -resize 64x64\< new_out.gif
```

此时，如果模型图片的宽或高小于 64 像素，则图片会被缩放到 64×64 大小，否则不缩放图片。

5. 填充整像素区域

将三维机器人模型图片 out.gif 缩放到 64×64 大小，填充整像素区域并生成新的图片 new_out.gif 的命令如下：

```
$ convert out.gif -resize 64×64^ new_out.gif
```

此时，如果模型图片的宽高比不等于指定的宽高比 64：64，则图片不会被拉伸，而只是将短边缩放到 64 像素并将长边按比例缩放，以填充整像素区域。此时允许配合其他参数裁剪掉像素区域之外的像素，而只保留 64×64 像素区域内的像素。

6. 按比例缩放

将三维机器人模型图片 out.gif 缩放 50%并生成新的图片 new_out.gif 的命令如下：

```
$ convert out.gif -resize 50% new_out.gif
```

此时，模型图片的宽度和高度将同时变为 50%并生成新的图片。

7. 限制像素数量

将三维机器人模型图片 out.gif 限制像素数量为 4096 缩放并生成新的图片 new_out.gif 的命令如下：

```
$ convert out.gif -resize 4096@ new_out.gif
```

此时，模型图片将被缩放为至多包含 4096 像素的大小。

8. 在读取图片时缩放

将三维机器人模型图片 out.gif 在读取图片时缩放到 64×64 大小并生成新的图片 new_out.gif 的命令如下：

```
$ convert out.gif '[64x64]' new_out.gif
```

3.7.2 -geometry 参数

在 display 命令中指定-geometry 参数可以控制图像的显示位置和大小。

在 montage 命令中指定-geometry 参数可以作为全局设置，用于控制图块的大小和位置。

1. 控制总图块的大小和位置

将三维机器人模型图片 out.gif 作为图块，从坐标(10,20)处的像素开始放置图块，共放置 3×4 个图块，图块的大小为 50×60，并生成新的图片 new_out.gif 的命令如下：

```
$ montage out.gif -tile 3x4 -geometry 50x60+10+20 new_out.gif
```

此时，模型图片将先缩放为 50×60 的大小，再从坐标(10,20)处的像素开始横向放置 3 次且纵向放置 4 次，最终的图片中含有 12 个图块。

此外，图块的大小允许缺省，此时不会缩放模型图片。将三维机器人模型图片 out.gif 作为图块，从坐标(10,20)处的像素开始放置图块，共放置 3×4 个图块并生成新的图片 new_out.gif 的命令如下：

```
$ montage out.gif -tile 3x4 -geometry 10+20 new_out.gif
```

2. 控制用于覆盖在背景上的图像的大小和位置

在 composite 命令中指定-geometry 参数可以作为全局设置，用于控制覆盖在背景上的图像的大小和位置。

将三维机器人模型图片 out.gif 从坐标(10,20)处的像素开始覆盖在背景上，图像的大小为 50×60，并生成新的图片 new_out.gif 的命令如下：

```
$ composite out.gif -geometry 50x60+10+20 new_out.gif
```

此时，模型图片将先缩放为 50×60 的大小，再从坐标(10,20)处的像素开始覆盖在背景上。

此外，图像的大小允许缺省，此时不会缩放模型图片。将三维机器人模型图片 out.gif 从坐标(10,20)处的像素开始覆盖在背景上，并生成新的图片 new_out.gif 的命令如下：

```
$ composite out.gif -geometry +10+20 new_out.gif
```

3.7.3 -thumbnail 参数

1. 生成缩略图并指定缩略图的大小

在 convert 命令中指定-thumbnail 参数可以生成缩略图并指定缩略图的大小。

将三维机器人模型图片 out.gif 缩放到 64×64 大小并生成缩略图 new_out.gif 的命令如下：

```
$ convert out.gif -thumbnail 64x64 new_out.gif
```

2. 指定缩略图的大小并将原图修改为缩略图

在 mogrify 命令中指定-thumbnail 参数可以指定缩略图的大小并将原图修改为缩略图。

直接将三维机器人模型图片 out.gif 缩放到 64×64 大小的命令如下：

```
$ mogrify -thumbnail 64x64 out.gif
```

注意：mogrify 命令可能损坏原图。

3. 生成缩略图作为图块并控制大小

在 montage 命令中指定-thumbnail 参数可以作为全局设置，用于生成缩略图作为图块并控制图块的大小。

将三维机器人模型图片 out.gif 作为图块，缩放到 64×64 大小并生成缩略图 new_out.gif 的命令如下：

```
$ montage out.gif -tile 3x4 -thumbnail 64x64\> -geometry 64x64\> new_out.gif
```

3.7.4 -sample 参数

在 convert 命令中指定-sample 参数可以指定图片的采样像素。

将三维机器人模型图片 out.gif 的采样像素指定为 64×64 大小并生成缩略图 new_out.gif 的命令如下：

```
$ convert out.gif -sample 64x64 new_out.gif
```

3.7.5 -scale 参数

在 convert 命令中指定-scale 参数相当于指定-resize 参数并使用 box 插值方式。

将三维机器人模型图片 out.gif 指定-scale 参数缩放到 64×64 大小并生成缩略图 new_out.gif 的命令如下：

```
$ convert out.gif -scale 64x64 new_out.gif
$ convert out.gif -scale 50% new_out.gif
```

3.7.6 -filter 参数

在 convert 命令中指定-filter 参数可以指定图片的滤波器。

将三维机器人模型图片 out.gif 指定 box 滤波器缩放到 64×64 大小并生成缩略图 new_out.gif 的命令如下：

```
$ convert out.gif -filter box -resize 64x64 new_out.gif
```

ImageMagick 支持 box、hermite、triangle、gaussian、quadratic、spline、lanczos、hamming、blackman、lagrange、catrom 和 mitchell 滤波器用于插值。