湖南科技大学课程教案
（章节、专题首页）

授课教师: 王志喜 职称: 副教授 单位: 计算机学院

课程名称	计算机图形图像技术
章节、专题	习题选讲
教学目标及基本要求	
教学重点	
教学难点	
教学内容与时间分配	
习题	

习题选讲

1 基本图元显示

1、使用DDA算法绘制端点为(20, 20)和(28, 26)的线段。

解:

$\Delta x = 8, \Delta y = 6, m = 0.75$

$x_0 = 20, y_0 = 20$	$x_5 = 25, y_5 = y_4 + m = 23.75 \approx 24$
$x_1 = 21, y_1 = y_0 + m = 20.75 \approx 21$	$x_6 = 26, y_6 = y_5 + m = 24.5 \approx 25$
$x_2 = 22, y_2 = y_1 + m = 21.5 \approx 22$	$x_7 = 27, y_7 = y_6 + m = 25.25 \approx 25$
$x_3 = 23, y_3 = y_2 + m = 22.25 \approx 22$	$x_8 = 28, y_8 = y_7 + m = 26$
$x_4 = 24, y_4 = y_3 + m = 23$	

2、使用中点算法绘制端点为(20, 20)和(28, 26)的线段。

解:

$$a=-6, b=8, 2a=-12, (2a+2b)=4$$

k	(x_k, y_k)	p_k
0	$(20,20)$	$2a+b=-4$
1	$(21,21)$	$p_0+(2a+2b)=0$
2	$(22,22)$	$p_1+(2a+2b)=4$
3	$(23,22)$	$p_2+2a=-8$
4	$(24,23)$	$p_3+(2a+2b)=-4$
5	$(25,24)$	$p_4+(2a+2b)=0$
6	$(26,25)$	$p_5+(2a+2b)=4$
7	$(27,25)$	$p_6+2a=-8$
8	$(28,26)$	

2 OpenGL基本图元支持

1、请使用OpenGL和GLUT编写一个简单的图形程序，用于显示一个填充的白色矩形。其中矩形规定为(-0.8, -0.8) ~ (0.8, 0.8)，程序窗口的大小为(200, 200)，标题为“白色矩形”。

```
#include <gl/freeglut.h>
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT);
    glRectf(-0.8, -0.8, 0.8, 0.8);
    glFlush();
}
int main(int argc, char *argv[])
{   glutInit(&argc, argv);
    glutInitWindowSize(200, 200);
    glutCreateWindow("白色矩形");
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

2、请使用OpenGL和GLUT编写一个简单的图形程序，用于显示一个填充的红色三角形。其中三角形的顶点分别是(-0.8, -0.8)、(0.8, -0.8)和(0, 0.8)，程序窗口大小为(200, 200)，标题为“红色三角形”。

解:

```
#include <gl/freeglut.h>
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1, 0, 0);
    glBegin(GL_TRIANGLES);
    glVertex2d(-0.8, -0.8);
    glVertex2d(0.8, -0.8);
    glVertex2d(0, 0.8);
    glEnd();
    glFlush();
}
```

```
int main(int argc, char *argv[])
{   glutInit(&argc, argv);
    glutInitWindowSize(200, 200);
    glutCreateWindow("红色三角形");
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

3、请使用OpenGL和GLUT编写一个简单的图形程序，用于显示一个填充的紫色梯形。其中梯形的4个顶点分别是(-0.9, -0.4)、(0.4, -0.4)、(0.4, 0.4)和(-0.4, 0.4)，程序窗口的大小为(300, 300)，标题为“紫色梯形”。

```
#include <gl/freeglut.h>
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1, 0, 1);
    glBegin(GL_QUADS);
    glVertex2f(-0.9, -0.4);
    glVertex2f(0.4, -0.4);
    glVertex2f(0.4, 0.4);
    glVertex2f(-0.4, 0.4);
    glEnd();
    glFlush();
}
```

```
int main(int argc, char *argv[])
{   glutInit(&argc, argv);
    glutInitWindowSize(300, 300);
    glutCreateWindow("紫色梯形");
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

3　二维图形变换

1、已知旋转角为60°，旋转中心为(1, 2)，请构造该二维旋转变换的变换矩阵M，结果至少保留3位小数（也可使用无理数）。

解:

（1）使基准点与原点重合：$T_1 = T(-1,-2)$

（2）绕原点旋转：$R = R(60°)$

（3）使基准点回到原处：$T_2 = T(1,2)$

完整变换

$$M = T_2 R T_1 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos 60° & -\sin 60° & 0 \\ \sin 60° & \cos 60° & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0.5 & -0.866 & 2.2321 \\ 0.866 & 0.5 & 0.1340 \\ 0 & 0 & 1 \end{pmatrix}$$

2、已知x和y方向的缩放比例为1.5和2，固定点位置为(1, 1)，请构造该二维缩放变换的变换矩阵。

解:

（1）使固定点与原点重合：$T_1 = T(-1,-1)$

（2）以原点为固定点缩放：$S = S(1.5,2)$

（3）使固定点回到原处：$T_2 = T(1,1)$

完整变换

$$M = T_2 S T_1 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1.5 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1.5 & 0 & -0.5 \\ 0 & 2 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

3、已知窗口为(0, 0)~(10, 10)，视区为(1, 1)~(6, 6)，要求将窗口中位于(x, y)的点映像到视区中坐标为(x', y')的点，请构造变换公式和变换矩阵。

解:

为了使视区与窗口中的对象有同样的相对位置，必须满足

$$\begin{cases} \dfrac{x_v - 1}{6 - 1} = \dfrac{x_w - 0}{10 - 0} \\ \dfrac{y_v - 1}{6 - 1} = \dfrac{y_w - 0}{10 - 0} \end{cases}$$

从而，得到如下变换公式和变换矩阵

$$\begin{cases} x_v = 0.5x_w + 1 \\ y_v = 0.5y_w + 1 \end{cases}, \quad \begin{pmatrix} 0.5 & 0 & 1 \\ 0 & 0.5 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

4、已知旋转角为 θ，缩放系数均为 s，旋转中心和固定点位置均为 (x_0, y_0)，请构造该带缩放的旋转变换的变换矩阵（OpenCV中的函数getRotationMatrix2D()就是用来计算这个变换矩阵的）。

解:

① 使旋转中心和固定点与原点重合：$T_1 = T(-x_0, -y_0)$

② 绕原点旋转：$R = R(\theta)$

③ 以原点为固定点缩放：$S = S(s, s)$。

④ 使旋转中心和固定点回到原处：$T_2 = T(x_0, y_0)$

完整变换：

$$
\begin{aligned}
M &= T_2 S R T_1 \\
&= \begin{pmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{pmatrix}
\begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix}
\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}
\begin{pmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} s\cos\theta & -s\sin\theta & x_0(1 - s\cos\theta) + y_0 s\sin\theta \\ s\sin\theta & s\cos\theta & -x_0 s\sin\theta + y_0(1 - s\cos\theta) \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

5、确定反射轴为直线 $y=3x/4$ 的反射变换矩阵的形式。

解:

显然，反射轴通过原点且方向为 $(4,3)$。

（1）使反射轴与 x 轴重合： M_1

$$\vec{U}=(4,3),\ \vec{V}=(-3,4),\ \vec{u}=(0.8,0.6),\ \vec{v}=(-0.6,0.8),$$

$$M_1=\begin{pmatrix}0.8 & 0.6 & 0\\ -0.6 & 0.8 & 0\\ 0 & 0 & 1\end{pmatrix}$$

（2）相对于 x 轴反射:

$$F_x=\begin{pmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{pmatrix}$$

（3）使反射轴回到原处： M_2

$$M_2 = M_1^{-1} = \begin{pmatrix} 0.8 & 0.6 & 0 \\ -0.6 & 0.8 & 0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 0.8 & -0.6 & 0 \\ 0.6 & 0.8 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

完整变换

$$\begin{aligned} M &= M_2 \times F_x \times M_1 \\ &= \begin{pmatrix} 0.8 & -0.6 & 0 \\ 0.6 & 0.8 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.8 & 0.6 & 0 \\ -0.6 & 0.8 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 0.28 & 0.96 & 0 \\ 0.96 & -0.28 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

6、已知新坐标系原点$P(1, 2)$，新y轴方向$(0.8, 0.6)$。请构造该2维坐标系变换，并写出变换矩阵。

解:

① 使新原点与旧原点重合：$T = T(-1,-2)$。

② 使新 y 轴与旧 y 轴重合：R。

因为 $\vec{v} = (0.8, 0.6)$，$\vec{u} = (0.6, -0.8)$，所以

$$R = \begin{pmatrix} 0.6 & -0.8 & 0 \\ 0.8 & 0.6 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

完整变换：

$$M = RT = \begin{pmatrix} 0.6 & -0.8 & 0 \\ 0.8 & 0.6 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0.6 & -0.8 & 1 \\ 0.8 & 0.6 & -2 \\ 0 & 0 & 1 \end{pmatrix}$$

7、已知裁剪窗口为(0, 0)~(5, 5)，要求将裁剪窗口中位于(x, y)的点映像到规范化正方形（坐标范围为[-1, 1]）中坐标为(x', y')的点，请构造变换公式和变换矩阵。

解:

为了使规范化正方形与裁剪窗口中的对象有同样的相对位置，必须满足

$$\begin{cases} \dfrac{x'-(-1)}{1-(-1)} = \dfrac{x-0}{5-0} \\ \dfrac{y'-(-1)}{1-(-1)} = \dfrac{y-0}{5-0} \end{cases}$$

解得

$$\begin{cases} x' = \dfrac{2}{5}x - 1 = 0.4x - 1 \\ y' = \dfrac{2}{5}y - 1 = 0.4y - 1 \end{cases}$$

从而得到变换矩阵为

$$\begin{pmatrix} 0.4 & 0 & -1 \\ 0 & 0.4 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

8、已知屏幕视口为(0, 0)~(200, 200)，要求将规范化正方形（坐标范围为[-1, 1]）中位于(x, y)的点映像到屏幕视口中坐标为(x', y')的点，请构造变换公式和变换矩阵。

解:

为了使规范化正方形与屏幕视口中的对象有同样的相对位置，必须满足

$$\begin{cases} \dfrac{x'-0}{200-0}=\dfrac{x-(-1)}{1-(-1)} \\ \dfrac{y'-0}{200-0}=\dfrac{y-(-1)}{1-(-1)} \end{cases}$$

解得

$$\begin{cases} x'=\dfrac{200}{2}(x+1)+0=100x+100 \\ y'=\dfrac{200}{2}(y+1)+0=100y+100 \end{cases}$$

从而得到变换矩阵为

$$\begin{pmatrix} 100 & 0 & 100 \\ 0 & 100 & 100 \\ 0 & 0 & 1 \end{pmatrix}$$

9、已知线段*AB*的两个端点坐标分别是*A*(-5, 10)和*B*(10, -5)，裁剪窗口为(0, 0)~(10, 10)，请使用Cohen-Sutherland算法计算出裁剪以后剩余的线段。

解:

左边界：$x=0$，右边界：$x=10$，下边界：$y=0$，上边界：$y=10$。

A 区域码：0001，B 区域码：0100。两端点区域码的与为0000。

A 是一外端点，位于窗口左边，AB 与左边界 $x=0$ 求交

$$m=(-5-10)/(10+5)=-1$$

$$y=y_1+m(c-x_1)=10-(0+5)=5$$

得交点 $A'=(0,5)$。舍弃 AA'，保留 $A'B$。

A' 区域码：0000，B 区域码：0100。两端点区域码的与为0000。

B 是一外端点，位于窗口的下边，$A'B$ 与下边界 $y=0$ 求交

$$1/m=-1,\quad x=x_1+(c-y_1)/m=-5-(0-10)=5$$

得交点 $B'=(5,0)$。舍弃 $B'B$，保留 $A'B'$。

A' 区域码：0000，B' 区域码：0000。所以裁剪后剩余线段为 $A'B'$，端点坐标分别为：$A'=(0,5)$，$B'=(5,0)$。

4 三维图形变换

1、已知旋转轴为 AB，其中 $A=(0,0,0)$， $B=(3,4,0)$，请构造绕 AB 旋转 90 度的旋转变换。

解:

（1）使旋转轴与 z 轴重合： R

将 $\overrightarrow{AB}=(3,4,0)$ 单位化，得 $\vec{n}=\overrightarrow{AB}/|\overrightarrow{AB}|=(0.6,0.8,0)$

令 $\vec{u}_x=(1,0,0)$

$$\begin{aligned}\vec{v}&=\vec{n}\times\vec{u}_x/|\vec{n}\times\vec{u}_x|\\&=(0.6,0.8,0)\times(1,0,0)/|(0.6,0.8,0)\times(1,0,0)|\\&=(0,0,-1)\\\vec{u}&=\vec{v}\times\vec{n}=(0,0,-1)\ \times(0.6,0.8,0)=(0.8,-0.6,0)\end{aligned}$$

则

$$R=\begin{pmatrix}0.8 & -0.6 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0.6 & 0.8 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

（2）绕坐标轴（z 轴）完成指定的旋转：$R_z(\theta)$

（3）使旋转轴回到原来的方向：R^{-1}

完整变换

$$M=R^{-1}\times R_z(90^\circ)\times R$$

$$=\begin{pmatrix}0.8 & 0 & 0.6 & 0\\ -0.6 & 0 & 0.8 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}0 & -1 & 0 & 0\\ 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}0.8 & -0.6 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0.6 & 0.8 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

$$=\begin{pmatrix}0.36 & 0.48 & 0.8 & 0\\ 0.48 & 0.64 & -0.6 & 0\\ -0.8 & 0.6 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

2、已知观察参考点为$P(2, 2, 2)$，观察面法向量$\boldsymbol{n}=(0.8, 0.6, 0)$，观察向上向量$\boldsymbol{v}=(-0.6, 0.8, 0)$。请构造从世界坐标到观察坐标的变换，写出变换矩阵。

解:

① 使观察参考点与世界坐标系原点重合：$T = T(-2,-2,-2)$

② 使观察坐标系与世界坐标系重合：R

观察坐标系 z 轴方向：$\vec{n} = (0.8, 0.6, 0)$

观察坐标系 y 轴方向：$\vec{v} = (-0.6, 0.8, 0)$

观察坐标系 x 轴方向：$\vec{u} = \vec{v} \times \vec{n} = (0, 0, -1)$

$$R = \begin{pmatrix} 0 & 0 & -1 & 0 \\ -0.6 & 0.8 & 0 & 0 \\ 0.8 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

完整变换：

$$M = RT = \begin{pmatrix} 0 & 0 & -1 & 0 \\ -0.6 & 0.8 & 0 & 0 \\ 0.8 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & -1 & 2 \\ -0.6 & 0.8 & 0 & -0.4 \\ 0.8 & 0.6 & 0 & -2.8 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3、已知投影向量为 $\overline{V}=(3,4,1)$，投影面为 xy 平面，请根据定义计算该平行投影的变换矩阵。

解:

由 $(x'-x, y'-y, 0-z) = u(3,4,1)$ 可得

$$\begin{cases} x' = x + 3u \\ y' = y + 4u \\ 0 = z + u \end{cases}$$

由 $0 = z + u$ 得 $u = -z$ 。

于是得到该平行影变换的方程为

$$\begin{cases} x' = x - 3z \\ y' = y - 4z \end{cases}$$

原始 z 坐标作为深度信息保存。

从而，变换矩阵为

$$M = \begin{pmatrix} 1 & 0 & -3 & 0 \\ 0 & 1 & -4 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4、求经过平行投影变换后点 $P(1,2,3)$ 的坐标。已知：观察面为 $z=-4$，投影向量为 $(1,1,1)$。

解:

由 $(x'-1,y'-2,-4-3)=u(1,1,1)$ 可得

$$\begin{cases} x'=1+u \\ y'=2+u \\ -4=3+u \end{cases}$$

由 $-4=3+u$ 得 $u=-7$。

于是得到

$$\begin{cases} x'=1-7=-6 \\ y'=2-7=-5 \end{cases}$$

所以经过平行投影变换后点 $P(1,2,3)$ 的坐标为 $(-6,-5,3)$（答 $(-6,-5)$ 也可以）。

5、求经过透视投影变换后点 $P(1,2,3)$ 的坐标。已知：观察面为 $z=-4$，投影中心为 $(0,0,0)$。

解:

由 $(x'-1,y'-2,-4-3)=u(0-1,0-2,0-3)$ 可得

$$\begin{cases} x'=1-u \\ y'=2-2u \\ -7=-3u \end{cases}$$

由 $-7=-3u$ 可得 $u=7/3$。

于是得到

$$\begin{cases} x'=1-\dfrac{7}{3}=-\dfrac{4}{3} \\ y'=2-\dfrac{7}{3}\times 2=-\dfrac{8}{3} \end{cases}$$

所以经过透视投影变换后点 $P(1,2,3)$ 的坐标为 $(-4/3,-8/3)$（答 $(-4/3,-8/3,-4)$ 或 $(-4/3,-8/3,3)$ 也可以）。

6、已知投影中心为原点，投影面为 $z=-5$，请根据定义计算该透视投影的变换矩阵。

解:

由 $(x'-x,y'-y,-5-z)=u(0-x,0-y,0-z)$ 可得

$$\begin{cases}x'=x-xu\\y'=y-yu\\-5=z-zu\end{cases}$$

由 $-5=z-zu$ 可得 $u=(z+5)/z$ 。

于是得到透视变换方程

$$\begin{cases}x'=x-\dfrac{z+5}{z}x=-\dfrac{5}{z}x\\y'=y-\dfrac{z+5}{z}y=-\dfrac{5}{z}y\end{cases}$$

选取 $h=-z$，可得使用齐次坐标的变换方程为

$$\begin{cases} x_h = x' \bullet h = 5x \\ y_h = y' \bullet h = 5y \\ z_h = z' \bullet h = 5z \\ h = -z \end{cases}$$

由该变换方程可得透视变换的变换矩阵为

$$\begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

5　OpenGL图形变换

1、请使用OpenGL、GLU和GLUT编写一个简单的多视口演示程序。要求：在屏幕窗口左下角的1/4部分显示一个红色的填充正三角形；在屏幕窗口右上角的1/4部分显示一个绿色的填充正方形；三角形和正方形的左下角顶点坐标值均为(0, 0)，右下角顶点坐标值均为(1, 0)；裁剪窗口均为(-0.1, -0.1)～(1.1, 1.1)；程序窗口的大小为(200, 200)，标题为“多视口演示”。

```
#include <gl/freeglut.h>
void Viewport(int x, int y, int w, int h)
{   glViewport(x, y, w, h);
    glLoadIdentity();
    gluOrtho2D(-0.1, 1.1, -0.1, 1.1);
}
```

```
void Triangle()
{   glBegin(GL_TRIANGLES);
    glVertex2d(0, 0);
    glVertex2d(1, 0);
    glVertex2d(0.5, 0.8660);
    glEnd();
}
```

```
void Paint()
{   int w = glutGet(GLUT_WINDOW_WIDTH) / 2;
    int h = glutGet(GLUT_WINDOW_HEIGHT) / 2;
    glClear(GL_COLOR_BUFFER_BIT);
    Viewport(0, 0, w, h);
    glColor3f(1, 0, 0);
    Triangle();
    Viewport(w, h, w, h);
    glColor3f(0, 1, 0);
    glRectd(0, 0, 1, 1);
    glFlush();
}
```

```
int main(int argc, char *argv[])
{   glutInit(&argc, argv);
    glutInitWindowSize(200, 200);
    glutCreateWindow("多视口演示");
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

2、请使用OpenGL和GLUT编写一个显示线框椭球体的简单图形程序。其中椭球体的两极方向为上下方向，左右方向的半径为0.98，上下方向的半径为0.49，前后方向的半径为0.6，经线数为48，纬线数为24，使用正投影，裁剪窗口为(-1, -0.5)~(1, 0.5)，程序窗口的大小为(400, 200)，标题为“线框椭球”。

```
#include <GL/freeglut.h>
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    gluOrtho2D(-1, 1, -0.5, 0.5);
    glScaled(0.98, 0.49, 0.6);
    glRotated(-90, 1, 0, 0);
    glutWireSphere(1, 48, 24);
    glFlush();
}
```

```
int main(int argc, char *argv[])
{   glutInit(&argc, argv);
    glutInitWindowSize(400, 200);
    glutCreateWindow("线框椭球");
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

3、请使用OpenGL、GLU和GLUT编写一个三维犹他茶壶程序。其中茶壶的半径为1单位，并远移6.5单位；观察体规定为：视场角=30度，宽高比=1，近=1，远=100；程序窗口的大小为(200, 200)，标题为“尤他茶壶”。

```
#include <gl/freeglut.h>
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    gluPerspective(30, 1, 1, 100);
    glTranslatef(0, 0, -6.5);
    glutSolidTeapot(1);
    glFlush();
}
int main(int argc, char *argv[])
{   glutInit(&argc, argv);
    glutInitWindowSize(200, 200);
    glutCreateWindow("尤他茶壶");
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

4、请使用OpenGL和GLUT编写一个“旋转的尤他茶壶”程序。其中茶壶的半径为0.5；程序窗口的大小为(300, 300)，标题为“旋转的茶壶”。茶壶绕z轴不断旋转，旋转的时间间隔为50毫秒，角度间隔为5度。注意旋转角度必须限定在0 ~ 360度以内。

```
#include <gl/freeglut.h>
int angle = 0;
void Timer(int millis)
{   angle = (angle + 5) % 360;
    glutPostRedisplay();
    glutTimerFunc(millis, Timer, millis);
}
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    glRotatef(angle, 0, 0, 1);
    glutSolidTeapot(0.5);
    glFlush();
}
```

```
int main(int argc, char *argv[])
{   glutInit(&argc, argv);
    glutInitWindowSize(300, 300);
    glutCreateWindow("旋转的茶壶");
    glutDisplayFunc(Paint);
    glutTimerFunc(50, Timer, 50);
    glutMainLoop();
}
```

5、请使用OpenGL和GLUT编写一个“旋转的线段”程序。其中初始线段为(-0.9, 0)~(0.9, 0)，线宽为5；程序窗口的大小为(300, 300)，标题为“旋转的线段”。线段绕z轴不断旋转，旋转的时间间隔为50毫秒，角度间隔为5度。注意旋转角度必须在0~360度以内。

```
#include <gl/freeglut.h>
int angle = 0;
void Timer(int millis)
{   angle = (angle + 5) % 360;
    glutPostRedisplay();
    glutTimerFunc(millis, Timer, millis);
}
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    glRotatef(angle, 0, 0, 1);
    glBegin(GL_LINES);
    glVertex2f(-0.9, 0); glVertex2f(0.9, 0);
    glEnd();
    glFlush();
}
```

```
int main(int argc, char *argv[])
{   glutInit(&argc, argv);
    glutInitWindowSize(300, 300);
    glutCreateWindow("旋转的线段");
    glutDisplayFunc(Paint);
    glutTimerFunc(50, Timer, 50);
    glLineWidth(5);
    glutMainLoop();
}
```

6、请使用OpenGL、GLU和GLUT编写一个三维犹他茶壶程序。其中茶壶的半径为1单位，并远移6.5单位；观察体规定为：视场角=30度，宽高比=1，近=1，远=100；程序窗口的大小为(200, 200)，标题为“旋转的尤他茶壶”。茶壶绕z轴不断旋转，旋转的时间间隔为25毫秒，角度间隔为2度。注意旋转角度必须限定在0～360度以内。

```
#include <gl/freeglut.h>
int angle = 0;
void timer(int millis)
{   angle = (angle + 2) % 360;
    glutPostRedisplay();
    glutTimerFunc(millis, timer, millis);
}
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    gluPerspective(30, 1, 1, 100);
    glTranslatef(0, 0, -6.5);
    glRotated(angle, 0, 0, 1);
    glutSolidTeapot(1);
    glutSwapBuffers();
}
int main(int argc, char *argv[])
```

```
{   glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
    glutInitWindowSize(200, 200);
    glutCreateWindow("旋转的尤他茶壶");
    glutTimerFunc(25, timer, 25);
    glutDisplayFunc(Paint);
    glutMainLoop();
}
```

6 OpenGL的真实感图形

1、使用OpenGL和GLUT编写一个程序，用于模拟一个非常光滑的纯白球面在烈日暴晒下的效果。

```
#include <gl/freeglut.h>
void Hint()
{   glMaterialfv(GL_FRONT, GL_SPECULAR, (float[]) {1, 1, 1, 1});
    glMaterialf(GL_FRONT, GL_SHININESS, 16);
    glLightfv(GL_LIGHT0, GL_POSITION, (float[]) { 1, 1, 1, 0 });
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);
}
void Reshape(int w, int h)
{   glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(30, 1, 1, 1000);
    glTranslatef(0, 0, -4);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
void Paint()
```

```
{   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glutSolidSphere(1, 120, 60);
    glFlush();
}
int main(int argc, char *argv[])
{   glutInit(&argc, argv);
    glutInitWindowSize(300, 300);
    glutCreateWindow("光滑球面");
    glutReshapeFunc(Reshape);
    glutDisplayFunc(Paint);
    glEnable(GL_DEPTH_TEST);
    Hint();
    glutMainLoop();
}
```

2、已知在一个空旷的场景中有一个粗糙的紫色球体，球体的右上角方向放置了一个白色的点光源，请使用OpenGL和GLUT编写一个程序模拟出球面上的光照效果（不考虑环境光）。

```
#include <gl/freeglut.h>
```

```
void Hint()
{   glMaterialfv(GL_FRONT, GL_DIFFUSE, (float[]) {1, 0, 1, 1});
    glLightfv(GL_LIGHT0, GL_POSITION, (float[]) { 20, 20, 0, 1 });
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHTING);
}
```

```
void Reshape(int w, int h)
{   glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(30, 1, 1, 1000);
    glTranslatef(0, 0, -4);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
```

```
void Paint()
{   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glutSolidSphere(1, 120, 60);
    glFlush();
}
```

```
int main(int argc, char *argv[])
{   glutInit(&argc, argv);
    glutInitWindowSize(300, 300);
    glutCreateWindow("紫色的球");
    glutReshapeFunc(Reshape);
    glutDisplayFunc(Paint);
    glEnable(GL_DEPTH_TEST);
    Hint();
    glutMainLoop();
}
```

7　数字图像处理概述

1、随意从互联网下载一幅真彩色图像，使用OpenCV编写一个读入并显示该图像的程序，要求使用真彩色图像和灰度图像两种方式显示。相关函数的使用请参阅OpenCV手册。

```
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   Mat color = imread("lena.jpg", 1);
    Mat gray = imread("lena.jpg", 0);
    imshow("彩色", color);
    imshow("灰度", gray);
    waitKey();
}
```

2、使用OpenCV编写一个程序，该程序首先装入一幅BMP真彩色图像（使用当前目录中的flower.bmp），然后显示该图像，最后将图像另存为JPG格式的图像文件。

```
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   Mat X = imread("flower.bmp", 1);
    imshow("源图像", X);
    imwrite("flower.jpg", X);
    waitKey();
}
```

8 OpenCV核心功能

1、使用OpenCV装入一幅大小至少为300*300的彩色图像，并显示该图像。然后在源图像中指定一个矩形区域（左上顶点和宽高值分别为(64, 128)和(128, 64)），并在结果图像窗口中显示源图像中被选取的部分。

```
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   Mat src = imread("lena.jpg", 1);
    imshow("src", src);
    Rect rect(64, 128, 128, 64);
    Mat dst = src(rect);
    imshow("dst", dst);
    waitKey();
}
```

2、请使用OpenCV编写一个简单的程序，该程序首先从一幅真彩色图像（使用当前目录中的lena.jpg）中选取一个矩形子集，并用白色填充该矩形子集，然后显示图像。其中矩形子集的起始位置为(64, 128)，大小为(128, 64)。

```
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   Mat X = imread("lena.jpg", 1);
    Mat Y = X(Rect(64, 128, 128, 64));
    Y.setTo(Scalar(255, 255, 255));
    imshow("SubRect", X);
    waitKey();
}
```

3、请使用OpenCV编写一个简单的程序，该程序首先读入一幅彩色图像（使用当前目录中的lena.jpg），然后将这幅彩色图像的3个通道分离出来，得到3幅灰度图像，最后显示这3幅灰度图像。

```
#include<opencv2/opencv.hpp>
using namespace cv;
using namespace std;
int main()
{   Mat im = imread("lena.jpg", 1);
    vector<Mat> mv;
    split(im, mv);
    imshow("蓝色图像", mv[0]);
    imshow("绿色图像", mv[1]);
    imshow("红色图像", mv[2]);
    waitKey();
}
```

$$F(u,v)=\sum_{y=0}^{N-1}\sum_{x=0}^{M-1}f(x,y)\omega_x^{ux}\omega_y^{vy}$$

9 图像变换

1、请计算对下列实数矩阵进行傅立叶正变换后的变换结果（不缩放结果）。

1	7	4	0
9	4	8	8

【解答】$\omega_x=e^{-2\pi i/4}=-i,\ \omega_y=e^{-2\pi i/2}=-1$

$$\begin{aligned}F(0,0)&=f(0,0)+f(1,0)+f(2,0)+f(3,0)+\\&\quad f(0,1)+f(1,1)+f(2,1)+f(3,1)\\&=1+7+4+0+9+4+8+8\\&=41\end{aligned}$$

$$\begin{aligned}F(1,0)&=f(0,0)+f(1,0)\omega_x+f(2,0)\omega_x^2+f(3,0)\omega_x^3+\\&\quad f(0,1)+f(1,1)\omega_x+f(2,1)\omega_x^2+f(3,1)\omega_x^3\\&=1+7(-i)+4(-i)^2+0(-i)^3+9+4(-i)+8(-i)^2+8(-i)^3\\&=2-3i\end{aligned}$$

$$
\begin{aligned}
F(2,0) &= f(0,0) + f(1,0)\omega_x^2 + f(2,0)\omega_x^4 + f(3,0)\omega_x^6 + \\
&\quad f(0,1) + f(1,1)\omega_x^2 + f(2,1)\omega_x^4 + f(3,1)\omega_x^6 \\
&= 1 + 7(-i)^2 + 4(-i)^4 + 0(-i)^6 + 9 + 4(-i)^2 + 8(-i)^4 + 8(-i)^6 \\
&= 3
\end{aligned}
$$

$$
\begin{aligned}
F(3,0) &= f(0,0) + f(1,0)\omega_x^3 + f(2,0)\omega_x^6 + f(3,0)\omega_x^9 + \\
&\quad f(0,1) + f(1,1)\omega_x^3 + f(2,1)\omega_x^6 + f(3,1)\omega_x^9 \\
&= 1 + 7(-i)^3 + 4(-i)^6 + 0(-i)^9 + 9 + 4(-i)^3 + 8(-i)^6 + 8(-i)^9 \\
&= -2 + 3i
\end{aligned}
$$

$$
\begin{aligned}
F(0,1) &= f(0,0) + f(1,0) + f(2,0) + f(3,0) + \\
&\quad f(0,1)\omega_y + f(1,1)\omega_y + f(2,1)\omega_y + f(3,1)\omega_y \\
&= 1 + 7 + 4 + 0 - 9 - 4 - 8 - 8 \\
&= -17
\end{aligned}
$$

1	7	4	0
9	4	8	8

$$
F(u,v) = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} f(x,y)\omega_x^{ux}\omega_y^{vy}
$$

$$
\begin{aligned}
F(1,1) &= f(0,0) + f(1,0)\omega_x + f(2,0)\omega_x^2 + f(3,0)\omega_x^3 + \\
&\quad f(0,1)\omega_y + f(1,1)\omega_x\omega_y + f(2,1)\omega_x^2\omega_y + f(3,1)\omega_x^3\omega_y \\
&= 1 + 7(-i) + 4(-i)^2 + 0(-i)^3 - 9 - 4(-i) - 8(-i)^2 - 8(-i)^3 \\
&= -4 - 11i
\end{aligned}
$$

$$
\begin{aligned}
F(2,1) &= f(0,0) + f(1,0)\omega_x^2 + f(2,0)\omega_x^4 + f(3,0)\omega_x^6 + \\
&\quad f(0,1)\omega_y + f(1,1)\omega_x^2\omega_y + f(2,1)\omega_x^4\omega_y + f(3,1)\omega_x^6\omega_y \\
&= 1 + 7(-i)^2 + 4(-i)^4 + 0(-i)^6 - 9 - 4(-i)^2 - 8(-i)^4 - 8(-i)^6 \\
&= -7
\end{aligned}
$$

$$
\begin{aligned}
F(3,1) &= f(0,0) + f(1,0)\omega_x^3 + f(2,0)\omega_x^6 + f(3,0)\omega_x^9 + \\
&\quad f(0,1)\omega_y + f(1,1)\omega_x^3\omega_y + f(2,1)\omega_x^6\omega_y + f(3,1)\omega_x^9\omega_y \\
&= 1 + 7(-i)^3 + 4(-i)^6 + 0(-i)^9 - 9 - 4(-i)^3 - 8(-i)^6 - 8(-i)^9 \\
&= -4 + 11i
\end{aligned}
$$

1	7	4	0
9	4	8	8

$$
F(u,v) = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} f(x,y)\omega_x^{ux}\omega_y^{vy}
$$

2、某研究者在对一个实数矩阵进行傅立叶变换时，将变换结果记录在一张草稿纸上。半小时后，由于意外，有部分数据受到了污损，请根据傅立叶变换的性质帮这位研究者恢复被污损的数据，并说明依据。其中，受到污损后的数据如下（i是虚数单位）。

-1	1+2i	3+4i	5		
1+2i	3+5i	7+9i	11+17i	23-29i	11-17i
3+4i	5+8i	11+14i	23+29i	19-24i	9-14i
5	7+11i	15+19i	35	15-19i	7-11i
	9+14i	19+24i			
	11+17i	23+29i			

【解答】根据二维离散傅立叶变换的性质，如果 $f(x,y)$ 是实数函数，则 $F(M-u,N-v)=\mathrm{conj}(F(u,v))$。由此可得，恢复后的数据如下。

-1	1+2i	3+4i	5	3-4i	1-2i
1+2i	3+5i	7+9i	11+17i	23-29i	11-17i
3+4i	5+8i	11+14i	23+29i	19-24i	9-14i
5	7+11i	15+19i	35	15-19i	7-11i
3-4i	9+14i	19+24i	23-29i	11-14i	5-8i
1-2i	11+17i	23+29i	11-17i	7-9i	3-5i

3、使用OpenCV编写一个演示傅立叶变换和逆变换的程序。该程序首先装入一幅灰度图像并显示该图像，然后对该图像进行傅立叶正变换，对得到的结果进行傅立叶逆变换，显示得到的结果以便与原图像进行比对。

```
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   Mat X = imread("lena.jpg", 0);
    Mat Y;
    X.convertTo(Y, CV_32F, (double)1 / 255);
    dft(Y, Y);
    idft(Y, Y, DFT_SCALE);
    imshow("src", X);
    imshow("dst", Y);
    waitKey();
}
```

4、使用OpenCV编写一个程序，该程序将一幅灰度图像（使用当前目录中的lena.jpg）的灰度值线性地变换到范围[0, 255]。要求分别显示源图像和结果图像。

```
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   Mat X = imread("lena.jpg", 0);
    Mat Y;
    normalize(X, Y, 255, 0, NORM_MINMAX);
    imshow("源图像", X);
    imshow("结果图像", Y);
    waitKey();
}
```

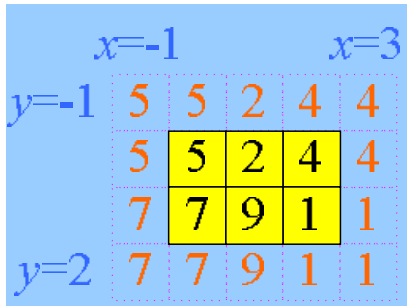
10　图像增强

1、请给出对下列灰度图像采用 3×3 模板进行中值滤波（中值模糊）后的结果（边界外元素当作边界元素处理）。

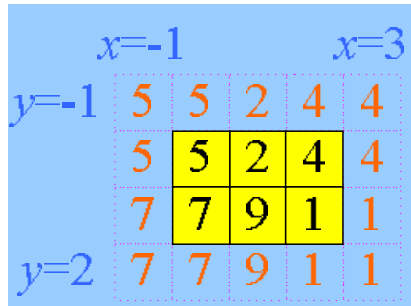
5	2	4
7	9	1

【解答】

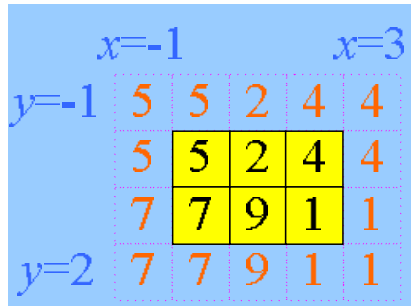
$$
\begin{aligned}
g(0,0) &= \mathrm{Med}\{f(-1,-1), f(0,-1), f(1,-1), f(-1,0), f(0,0), f(1,0), \\
&\quad f(-1,1), f(0,1), f(1,1)\} \\
&= \mathrm{Med}\{f(0,0), f(0,0), f(1,0), f(0,0), f(0,0), f(1,0), \\
&\quad f(0,1), f(0,1), f(1,1)\} \\
&= \mathrm{Med}\{5,5,2,5,5,2,7,7,9\} \\
&= 5
\end{aligned}
$$



$$
\begin{aligned}
g(1,0) &= \mathrm{Med}\{f(0,-1), f(1,-1), f(2,-1), f(0,0), f(1,0), f(2,0), \\
&\quad f(0,1), f(1,1), f(2,1)\} \\
&= \mathrm{Med}\{f(0,0), f(1,0), f(2,0), f(0,0), f(1,0), f(2,0), \\
&\quad f(0,1), f(1,1), f(2,1)\} \\
&= \mathrm{Med}\{5,2,4,5,2,4,7,9,1\} \\
&= 4 \\
g(2,0) &= \mathrm{Med}\{f(1,-1), f(2,-1), f(3,-1), f(1,0), f(2,0), f(3,0), \\
&\quad f(1,1), f(2,1), f(3,1)\} \\
&= \mathrm{Med}\{f(1,0), f(2,0), f(2,0), f(1,0), f(2,0), f(2,0), \\
&\quad f(1,1), f(2,1), f(2,1)\} \\
&= \mathrm{Med}\{2,4,4,2,4,4,9,1,1\} \\
&= 4
\end{aligned}
$$



$$
\begin{aligned}
g(0,1) &= \mathrm{Med}\{f(-1,0), f(0,0), f(1,0), f(-1,1), f(0,1), f(1,1), \\
&\quad f(-1,2), f(0,2), f(1,2)\} \\
&= \mathrm{Med}\{f(0,0), f(0,0), f(1,0), f(0,1), f(0,1), f(1,1), \\
&\quad f(0,1), f(0,1), f(1,1)\} \\
&= \mathrm{Med}\{5,5,2,7,7,9,7,7,9\} \\
&= 7 \\
g(1,1) &= \mathrm{Med}\{f(0,0), f(1,0), f(2,0), f(0,1), f(1,1), f(2,1), \\
&\quad f(0,2), f(1,2), f(2,2)\} \\
&= \mathrm{Med}\{f(0,0), f(1,0), f(2,0), f(0,1), f(1,1), f(2,1), \\
&\quad f(0,1), f(1,1), f(2,1)\} \\
&= \mathrm{Med}\{5,2,4,7,9,1,7,9,1\} \\
&= 5
\end{aligned}
$$



$$
\begin{aligned}
g(2,1) &= \mathrm{Med}\{f(1,0), f(2,0), f(3,0), f(1,1), f(2,1), f(3,1), \\
&\quad f(1,2), f(2,2), f(3,2)\} \\
&= \mathrm{Med}\{f(1,0), f(2,0), f(2,0), f(1,1), f(2,1), f(2,1), \\
&\quad f(1,1), f(2,1), f(2,1)\} \\
&= \mathrm{Med}\{2,4,4,9,1,1,9,1,1\} \\
&= 2
\end{aligned}
$$

由此可知，中值滤波（中值模糊）后的结果为

5	4	4
7	5	2

2、使用OpenCV编写一个程序，该程序对一幅灰度图像进行一次中值模糊，要求分别显示源图像和模糊化以后的图像。其中内核大小为5×5。

```
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   Mat X = imread("lena-n.jpg", 0);
    imshow("src", X);
    medianBlur(X, X, 5);
    imshow("dst", X);
    waitKey();
}
```

3、使用OpenCV编写一个程序，该程序对一幅彩色图像进行Sobel锐化，要求显示锐化以后的图像。其中内核大小为3×3，x和y方向均使用1阶差分。

```
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   Mat X = imread("lena.jpg", 1);
    imshow("src", X);
    Sobel(X, X, -1, 1, 1, 3);
    imshow("dst", X);
    waitKey();
}
```

4、使用OpenCV编写一个程序，该程序对一幅灰度图像（使用当前目录中的lena.jpg）进行Sobel锐化，要求显示源图像和结果图像。其中内核大小为3×3，使用1阶*x*差分计算模版系数。

解：

```
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   Mat X = imread("lena.jpg", 0);
    imshow("src", X);
    Sobel(X, X, -1, 1, 0, 3);
    imshow("dst", X);
    waitKey();
}
```

5、使用OpenCV编写一个程序，该程序对一幅灰度图像进行Laplace锐化，要求显示锐化以后的图像。其中内核大小为3×3。

```
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   Mat X = imread("lena.jpg", 0);
    imshow("src", X);
    Laplacian(X, X, -1, 3);
    imshow("dst", X);
    waitKey();
}
```

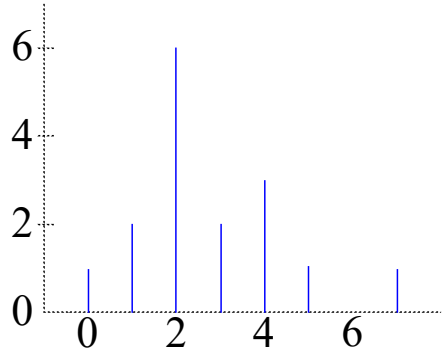
11　图像解析

1、请用线段方式绘制下列8灰度级图像的灰度直方图。

1	3	2	7
2	4	0	2
1	2	3	5
2	4	4	2

【解答】

因为根据图像数据，灰度值为0, 1, 2, 3, 4, 5, 6, 7的像素的频度分别是1, 2, 6, 2, 3, 1, 0, 1，因此在灰度刻度的0~7处分别作一条以该灰度值对应的频度值为长度的直线即可完成该图像的灰度直方图。



2、请对下列8灰度级图像进行直方图均衡化。

1	3	2	7
2	4	0	2
1	2	3	5
2	4	4	2

【解答】

- 计算归一化因子。$s = 7/16$。
- 源图像的直方图 $H = \{1,2,6,2,3,1,0,1\}$。
- 计算直方图积分。由 $H'(i) = \sum_{j=0}^{i} H(j)$ 得 $H' = \{1,3,9,11,14,15,15,16\}$。
- 直方图积分归一化。计算公式为 $H'[i] = H'[i] \times s$（舍入取整）。规范化后 $H' = \{0,1,4,5,6,7,7,7\}$。
- 采用 H' 作为查询表对图像灰度进行变换（$d_i = H'(s_i)$）。也就是将图像灰度按照如下对应关系修改。

源亮度	0	1	2	3	4	5	6	7
结果亮度	0	1	4	5	6	7	7	7

由此可得均衡化以后的图像为

1	5	4	7
4	6	0	4
1	4	5	7
4	6	6	4

3、使用OpenCV编写一个程序，该程序对一幅灰度图像进行直方图均衡化，要求分别显示源图像和均衡化以后的图像。

```
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   Mat X = imread("lena.jpg", 0);
    imshow("Source image", X);
    equalizeHist(X, X);
    imshow("Equalized Image", X);
    waitKey();
}
```

4、使用OpenCV编写一个程序，该程序对一幅灰度图像进行二值化变换，要求分别显示源图像和二值化以后的图像。其中二值化阈值为127，高亮度改为255。

```
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   Mat X = imread("lena.jpg", 0);
    imshow("Source image", X);
    threshold(X, X, 127, 255, THRESH_BINARY);
    imshow("Binary Image", X);
    waitKey();
}
```

5、使用OpenCV编写一个程序，该程序对一幅灰度图像进行Canny边缘检测，要求分别显示源图像和检测到的边缘。其中小阈值为50，大阈值为150，内核大小为3。

```
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   Mat X = imread("lena.jpg", 0);
    imshow("Source image", X);
    Canny(X, X, 50, 150, 3);
    imshow("Canny Image", X);
    waitKey();
}
```