第三章

图像校正

李静

实例：零件瑕疵检测

标准零件图像

实时采集零件图像

一、图像几何变换

目的：改变图像形状和位置，以便使用变化后的表达方式来获取其中的几何信息。

方法：将一个线性空间中的n维坐标矢量映射到另一个n维坐标矢量。

$$I'(x',y') = I(x,y)$$

（1）正交变换和刚体变换

（2）仿射变换

（3）透视变换

一、图像几何变换

（1）正交变换和刚体变换

正交变换：
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & 0 \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \boldsymbol{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad \boldsymbol{R}\boldsymbol{R}^{\mathrm{T}} = \boldsymbol{I}$$ 为正交矩阵

刚体变换：
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}$ 为任意正交矩阵（角度不同）；

$\boldsymbol{t} = \begin{bmatrix} t_1 & t_2 \end{bmatrix}^{\mathrm{T}}$ 为平移向量。

一、图像几何变换

（2）仿射变换

刚体变换：$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

相似变换：放宽式中R的正交条件，即$\boldsymbol{R}\boldsymbol{R}^{\mathrm{T}} = k\boldsymbol{I}$，k为任意比例系数，变换前后矢量之间的夹角不变，但是矢量的长度变化。

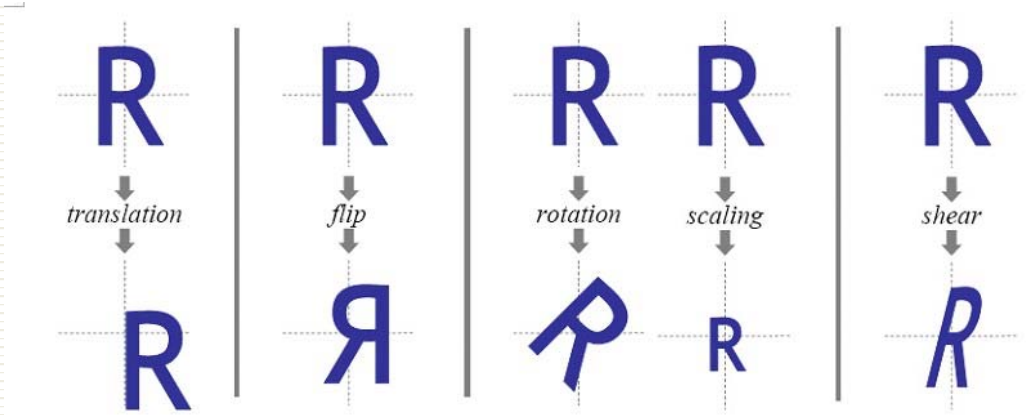
实例：采用不同焦距的摄像机对同一景物采集时所发生的缩放效果。

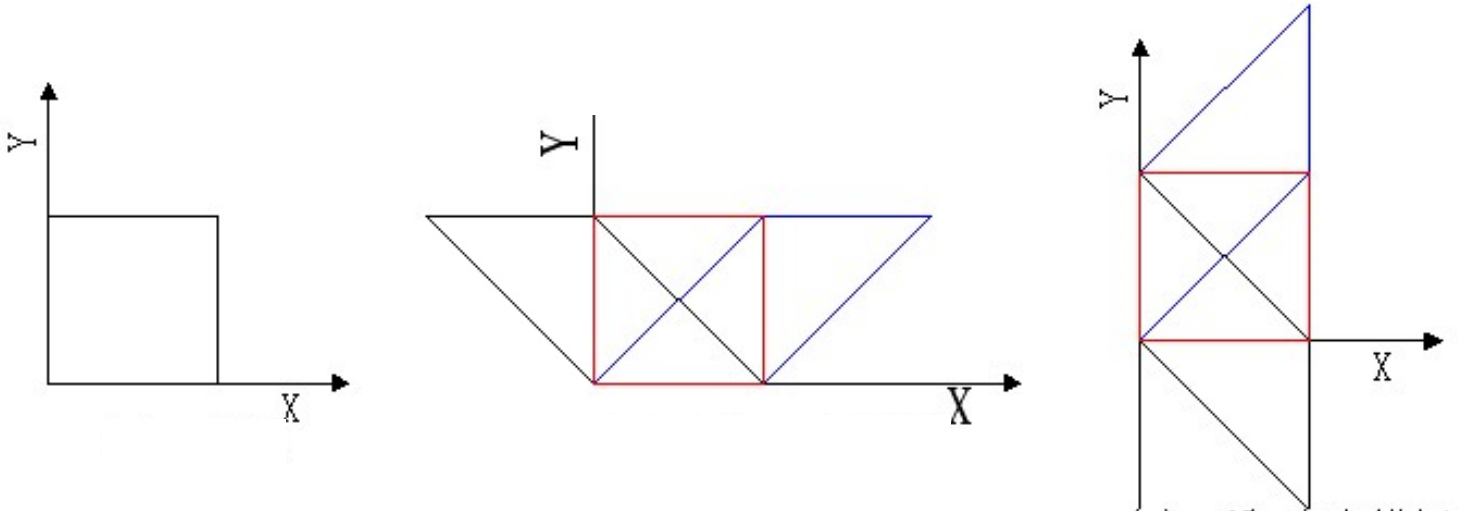
仿射变换：$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_1 \\ a_{21} & a_{22} & t_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{0}^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

一、图像几何变换

可通过一系列原子变换的复合来实现

- 平移（Translation）
- 缩放（Scale）
- 旋转（Rotation）
- 翻转（Flip）
- 剪切（Shear）





（a）原图　　（b）沿x方向剪切　　（c）沿y方向剪切

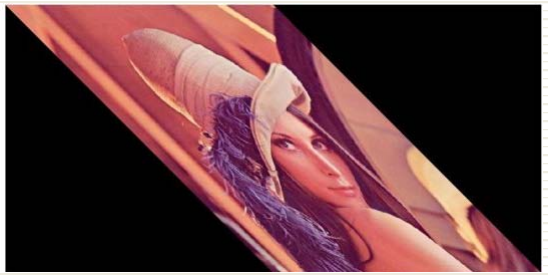
一、图像几何变换

实例

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_1 \\ a_{21} & a_{22} & t_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{0}^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

旋转

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



水平剪切

$$\begin{bmatrix} 1 & \alpha & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{cases} x' = x + y\alpha \\ y' = y \end{cases}$$

竖直剪切

$$\begin{bmatrix} 1 & 0 & 0 \\ \beta & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{cases} x' = x \\ y' = \beta x + y \end{cases}$$

平移

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{cases} x' = x + t_x \\ y' = y + t_y \end{cases}$$

缩放

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{cases} x' = s_x x \\ y' = s_y y \end{cases}$$

一、图像几何变换

warpAffine 对图像做仿射变换

```
void warpAffine(InputArray src, OutputArray dst, InputArray M, Size dsize,
int flags=INTER_LINEAR);
```

getAffineTransform 由三对点计算仿射变换

```
Mat getAffineTransform(InputArray src, OutputArray dst);
```

getRotationMatrix2D 计算二维旋转的仿射变换矩阵

```
Mat getRotationMatrix2D(Point2f center, double angle, double scale);
```

一、图像几何变换

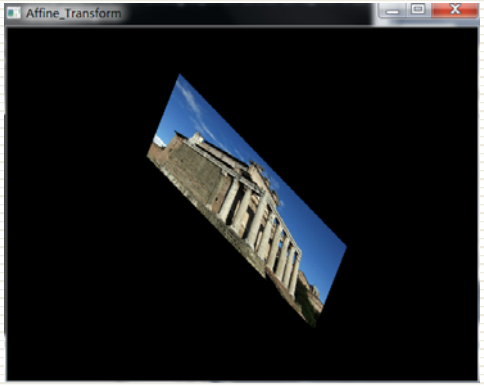
```
Point2f srcTri[3], dstTri[3]; //二维坐标下的点，类型为浮点
Mat rot_mat( 2, 3, CV_32FC1 );  //单通道矩阵
Mat warp_mat( 2, 3, CV_32FC1 );
Mat src, dst;
dst = Mat::zeros( src.rows, src.cols, src.type() );
//计算矩阵仿射变换
srcTri[0] = Point2f (0,0);
srcTri[1] = Point2f(src.cols - 1,0);  //缩小一个像素
srcTri[2]= Point2f(0,src.rows - 1);
//改变目标图像大小
dstTri[0] =Point2f(src.cols * 0.0,src.rows * 0.33);
dstTri[1] =Point2f(src.cols * 0.85,src.rows * 0.25);
dstTri[2] =Point2f(src.cols* 0.15,src.rows* 0.7);
//由三对点计算仿射变换
Warp_mat=getAffineTransform( srcTri, dstTri);
//对图像做仿射变换
warpAffine( src, dst, warp_mat,src.size());
```

$$\begin{bmatrix} 0.85 & 0.2 & 0 \\ -0.06 & 0.37 & 148.5 \\ 0 & 0 & 1 \end{bmatrix}$$

一、图像几何变换

```
//计算旋转仿射变换
 Point2f center = Point2f( src.cols/2, src.rows/2);
double angle = -50.0; //旋转角度，负值表示顺时针
double scale = 0.6;  //各项同性的尺度因子
rot_mat=getRotationMatrix2D( center, angle, scale );
warpAffine( src, dst, rot_mat );//将src仿射变换存入dst
```

原图



顺时针旋转50度

平移缩放

$$\begin{bmatrix} 0.3 & 0 & 0.5w \\ 0 & 0.3 & 0.5h \\ 0 & 0 & 1 \end{bmatrix}$$

特性：仿射变换具有直线性和平行性，即仿射变换前共线的三点在变换后仍然共线；仿射变换前的两条平行线在变换后依然平行。

一、图像几何变换

（3）透视变换

同样的景物在两个不同位置和角度的摄像机上形成的图像之间呈现一种透视关系，将两幅图像进行透视变换具体步骤：

第一步，建立两幅图像的透视变换方程

$$\omega\begin{bmatrix}x'\\y'\\1\end{bmatrix}=\begin{bmatrix}t_{11}&t_{12}&t_{13}\\t_{21}&t_{22}&t_{23}\\t_{31}&t_{32}&t_{33}\end{bmatrix}\begin{bmatrix}x\\y\\1\end{bmatrix}=\boldsymbol{T}\begin{bmatrix}x\\y\\1\end{bmatrix}$$

T为透视变换矩阵

$\begin{bmatrix}t_{11}&t_{12}\\t_{21}&t_{22}\end{bmatrix}$ 为旋转参数

$\begin{bmatrix}t_{13}\\t_{23}\end{bmatrix}$ 为平移参数

$[t_{31}\quad t_{32}]$ 为透视失真参数

t_{33} 为整个图像的比例因子

一、图像几何变换

第二步，从两幅图像获取公共特征点，并通过透视变换方程将后一幅图像的特征点逐一与前一幅图像的对应特征点进行重合配对。

第三步，将透视变换扩展至整个一幅图像，对一幅图像，从左上角像素开始直至右下角的像素为止，除了公共特征点以外的所有像素点逐一进行透视变换，转换成变换后的像素坐标图像。

warpPerspective 对图像进行透视变换

```
void warpPerspective(InputArray src, OutputArray dst, InputArray M,
Size dsize, int flags=INTER_LINEAR);
```

getPerspectiveTransform 由四对点计算透射变换

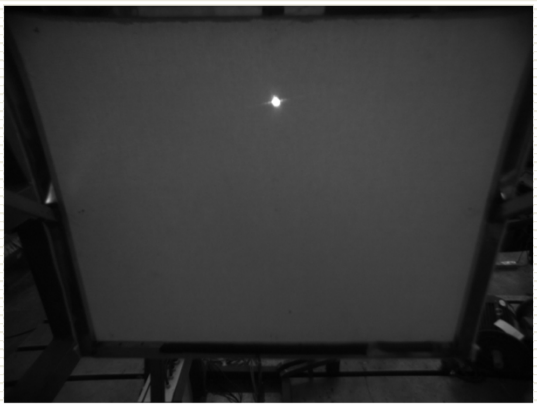
```
Mat getPerspectiveTransform(InputArray src, InputArray dst );
```

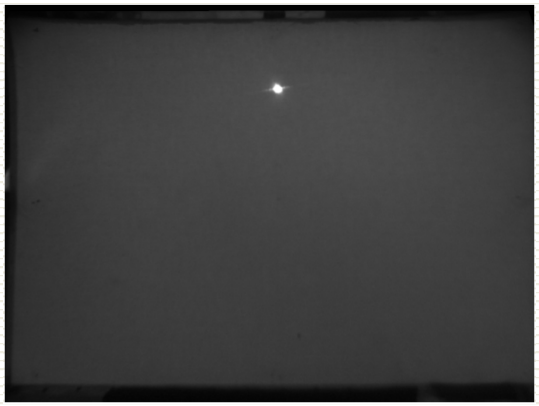
一、图像几何变换

```
Point2f srcQuad[4],dstQuad[4];
Mat warp_matrix(3,3,CV_32FC1);
Matsrc,dst;
dst = Mat::zeros( src.rows, src.cols, src.type());

srcQuad[0]=Point2f(0,0);                    //src top left
srcQuad[1] =Point2f(src.cols -1,0);         //src top right
srcQuad[2]=Point2f(0, src.rows-1);          //src bottom left
srcQuad[3]=Point2f(src.cols -1, src.rows-1);    //src bot right

dstQuad[0]=Point2f(src.cols*0.05,src.rows*0.33); //dst top left
dstQuad[1]=Point2f(src.cols*0.9,src.rows*0.25); //dst top right
dstQuad[2]=Point2f(src.cols*0.2,src.rows*0.7); //dst bottom left
dstQuad[3]=Point2f(src.cols*0.8,src.rows*0.9); //dst bot right

warp_matrix=getPerspectiveTransform(srcQuad,dstQuad);
warpPerspective(src,dst,warp_matrix,src.size());
```

$$\begin{bmatrix} 0.372 & 0.287 & 29.95 \\ -0.16 & 0.6 & 148.5 \\ -0.009 & 0.007 & 1 \end{bmatrix}$$

一、图像几何变换





透视变换前后的图像

特点：透视变换不再保证平行性，即变化前的平行线变换后不再平行，但是直线性仍然被保留，即变换前的直线在变换后仍然是直线。

对比：

仿射变换：平行四边形，三个控点，warp_mat矩阵2*3；

透视变换：任意四边形，四个控点，warp_mat矩阵3*3。

课堂测试5——编写程序：

1、放射变换程序

2、透视变换程序