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：计算机科学与工程学院

课 程 名 称	计算机图形图像技术
章 节 、 专 题	图像的基础变换
教 学 目 标 及 基 本 要 求	掌握图像的线性变换、仿射变换和透视变换的基本方法，熟悉OpenCV对线性变换、仿射变换和透视变换的支持。
教 学 重 点	图像的线性变换
教 学 难 点	图像的归一化，插值方法
教 学 内 容 与 时 间 分 配	（1）图像的线性变换及OpenCV中的相关支持（0.6课时） （2）图像的仿射变换及OpenCV中的相关支持（0.8课时） （3）图像的透视变换及OpenCV中的相关支持（0.6课时） 共计2课时。
习 题	第13.5.1节（基础知识题）和13.5.2（程序设计题）。

第13章　图像的基础变换

13.1　图像变换的主要内容

图像变换是对图像的像素位置或像素值进行的某种变换，主要内容通常包括对像素位置的变换、对单一像素的变换和对像素块的变换等方面。

（1）对像素位置的变换。逐一计算每个像素的新位置，用于改变像素的位置、图像的大小和形状等，通常使用图像的仿射变换和透视变换实现。

（2）对单一像素值的变换。每个新像素值都通过对应的单一源像素值获得，通常使用图像的线性变换实现。

（3）对像素块的变换。每个新像素值都通过对应的源像素块获得，通常使用图像的空域滤波和频域滤波等变换实现。

本章只介绍图像的线性变换、仿射变换和透视变换等基础变换，图像的空域滤波和频域滤波等变换在“图像增强”一章中介绍。

13.2 图像的线性变换

13.2.1 典型的线性变换

1. 变换方程

线性变换的变换方程为 $g(x,y)=\alpha f(x,y)+\beta$，其中 α 和 β 分别是线性变换的1次项系数和常数项。

2. OpenCV中的实现方法

（1）使用OpenCV矩阵表达式。“dst = alpha * src + beta”或“dst = src * alpha + beta”，其中，alpha和beta是线性变换的1次项系数和常数项。

$$g(x,y)=\alpha f(x,y)+\beta$$

（2）使用Mat对象的convertTo()成员函数。

【函数原型】void convertTo(OutputArray m, int rtype, double alpha = 1, double beta = 0) const;

【功能】对数组进行线性变换和类型转换。

【参数】

- *m*：结果数组，大小和通道数与源数组一致，必要时重建。
- rtype：结果数组的元素类型（位深度），负数表示与源数组一致。
- alpha和beta：线性变换的1次项系数和常数项。

【举例】这里只给出部分代码。

```
Mat2b X(4, 4); // 4×4双通道字节矩阵
Mat W = X * 1.5 + 2; // W(i) = X(i)×1.5 + 2
X.convertTo(W, -1, 1.5, 2);
```

13.2.2 图像的归一化

很多时候需要对图像元素的值进行归一化处理（使像素值位于某个指定的范围），可以根据某种范数或者某个数值范围归一化图像。图像的归一化可以看成是和图像的统计特性相关的一类特殊的线性变换。

1. 计算方法

（1）根据某种范数归一化图像。相当于将源图像 $f(x,y)$ 当作一个向量，将该向量单位化后乘以指定的范数值 α 得到结果图像 $g(x,y)$，即 $g(x,y)=\alpha f(x,y)\Big/\|f\|$。

（2）根据某个指定的数值范围归一化图像。假设源图像 $f(x,y)$ 中元素的最小值和最大值分别为 f_0 和 f_1，指定的数值范围是 $[g_0,g_1]$，即结果图像 $g(x,y)$ 中元素的最小值和最大值分别是 g_0 和 g_1，则归一化后结果图像元素值的计算方法如下。

由

$$\frac{g(x,y)-g_0}{f(x,y)-f_0}=\frac{g_1-g_0}{f_1-f_0}$$

可得

$$g(x,y)=\frac{g_1-g_0}{f_1-f_0}\big(f(x,y)-f_0\big)+g_0=\frac{g_1-g_0}{f_1-f_0}f(x,y)+\frac{f_1g_0-f_0g_1}{f_1-f_0}$$

2. OpenCV中的归一化函数

【函数原型】void normalize(InputArray src, InputOutputArray dst, double alpha = 1, double beta = 0, int norm_type = NORM_L2, int dtype = -1);

【功能】根据某种范数或者数值范围归一化数组。

【参数】

- src和dst：源数组和结果数组，大小和通道数一致，必要时重建结果数组。
- alpha和beta：结果数组的取值范围（alpha, beta）或范数（alpha）。
- norm_type：归一化类型，通常可选NORM_INF（C-范数归一化）、NORM_L1（L_1-范数归一化）、NORM_L2（L_2-范数归一化）和NORM_MINMAX（指定范围归一化）。
- dtype：结果数组类型，负数表示与源数组一致。

【说明】对于指定范数归一化，结果数组的范数由alpha指定。对于指定范围归一化，结果数组的取值范围由alpha和beta指定。

13.2.3 图像的融合

1. 计算方法

图像的融合，也可以称为图像的线性组合，就是计算两图像对应像素的加权和。两个大小和类型一致的图像 $f(x,y)$ 和 $g(x,y)$ 的融合为

$$s(x,y)=\alpha f(x,y)+\beta g(x,y)+\gamma$$

其中，α 和 β 是对应像素的权值，γ 是添加的常数项。

2. OpenCV中的图像融合函数

【函数原型】void addWeighted(InputArray src1, double alpha, InputArray src2, double beta, double gamma, OutputArray dst, int dtype = -1);

【功能】计算两数组对应元素的加权和。

【参数】

- src1和src2：通常是两个通道数和大小一致的源数组。
- alpha和beta：两个对应数组元素的权值。
- gamma：添加的常数项。
- dst：输出数组，大小和通道数与源数组一致，必要时重建。
- dtype：输出数组的类型，负数表示与源数组一致。

【说明】也可由矩阵表达式“dst = alpha * src1 + beta * src2 + gamma”完成。

13.2.4 线性变换应用举例

下列程序演示了图像融合、归一化和线性变换函数的使用，运行结果如图13-1所示。

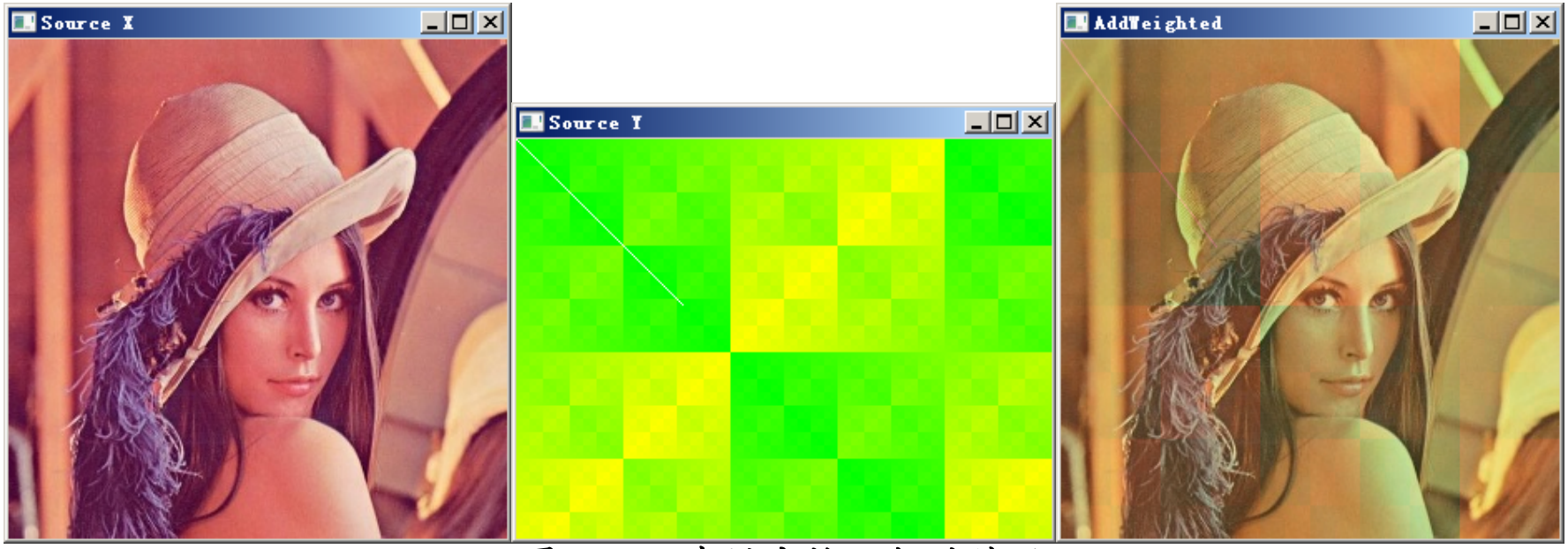


图 13-1 线性变换函数的使用

```
// AddWeighted.Cpp
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   // 读入彩色图像
    Mat X = imread("lena.jpg"), Y = imread("Green.bmp");
    if(X.empty() or Y.empty()) return -1; // 读取失败
    imshow("Source X", X), imshow("Source Y", Y); // 显示源图像
    // 源图像转换为浮点数图像, X(i) = X(i) / 255, Y(i) = Y(i) / 255
    X = Mat3f(X) / 255, Y = Mat3f(Y) / 255;
    X = X * 1.5 + Scalar::all(1); // X(i) = 1.5 * X(i) + 1
    resize(Y, Y, X.size()); // Y与X大小一致
    addWeighted(X, 0.7, Y, 0.3, 1, X); // 或 X = X * 0.7 + Y * 0.3 + 1;
    normalize(X, X, 0, 1, NORM_MINMAX); // 结果归一化到[0, 1]
    X = Mat3b(X * 255); // 结果转换为字节图像, X(i) = X(i) * 255
    imshow("AddWeighted", X); // 显示结果图像(字节图像)
    waitKey();
}
```

13.3 图像的仿射变换

13.3.1 仿射变换的变换公式

1. 变换公式

将像素从源位置 (x,y) 变换到目标位置 (x',y') 的变换公式为

$$\begin{cases} x' = a_{11}x + a_{12}y + a_{13} \\ y' = a_{21}x + a_{22}y + a_{23} \end{cases}$$

$$\begin{cases} x' = a_{11}x + a_{12}y + a_{13} \\ y' = a_{21}x + a_{22}y + a_{23} \end{cases}$$

2. 变换公式的矩阵表示

（1）齐次坐标。用三元组 (x_h, y_h, h) 表示坐标 (x, y)。其中，$x = x_h / h$，$y = y_h / h$。

（2）矩阵表示。如果使用齐次坐标，则变换公式等价于下述矩阵表示。

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

其中，矩阵

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix}$$

称为仿射变换的变换矩阵。

（3）逆变换。合理的仿射变换都是可逆的，且逆变换也是一个仿射变换，其变换公式为

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

（4）变换矩阵的存储。为节省存储空间，可以使用矩阵

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}$$

代替变换矩阵

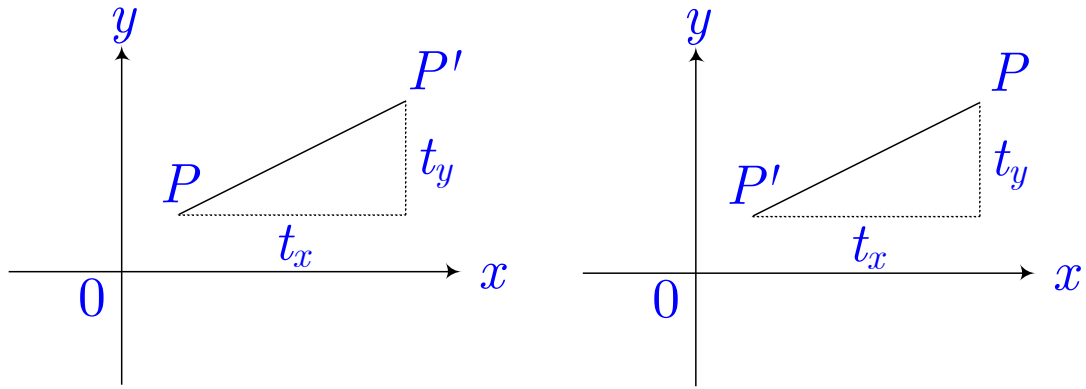
$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix}$$

13.3.2# 三种基本的仿射变换

平移、旋转和缩放是三种基本的仿射变换，其他的仿射变换通常可以通过这三种基本的仿射变换复合得到。

1. 平移

（1）含义。将像素沿直线路径从一个位置移到另一个位置。如图13-2(*a*)所示。



(*a*) 正变换　　(*b*) 逆变换

图13-2　平移

（2）变换方程。由图13-2(*a*)容易看出，该变换的变换方程为

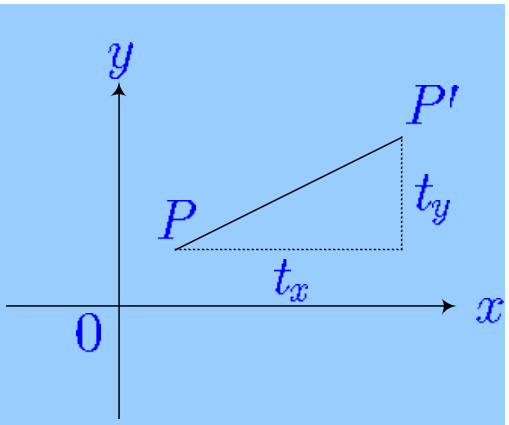
$$\begin{cases} x' = x + t_x \\ y' = y + t_y \end{cases}$$

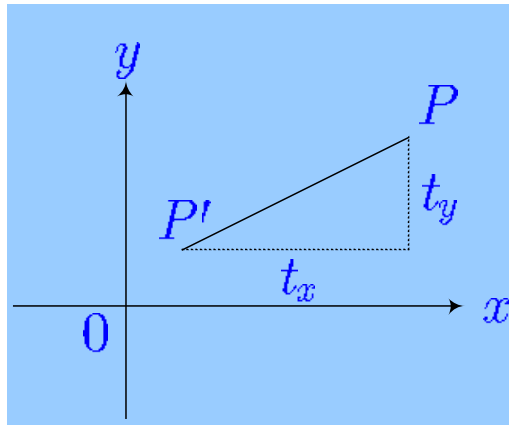
（3）矩阵形式。用 $T(t_x, t_y)$ 表示。将变换方程改写成矩阵形式，可得

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

（4）逆变换。如图13-2(*b*)所示。易知 $T^{-1}(t_x, t_y) = T(-t_x, -t_y)$。变换方程为

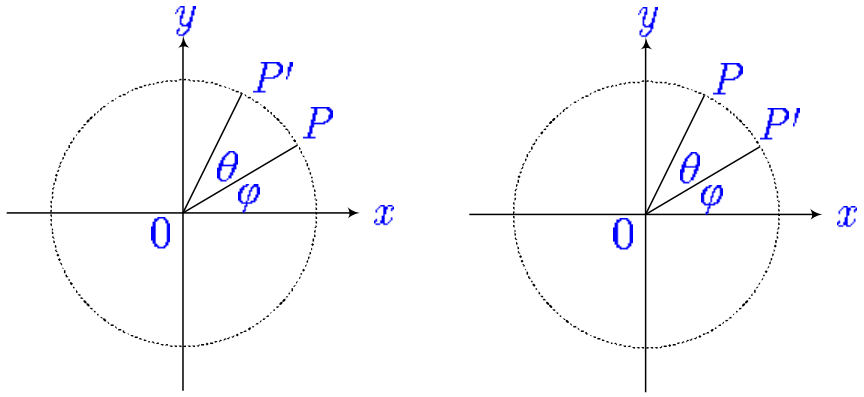
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$





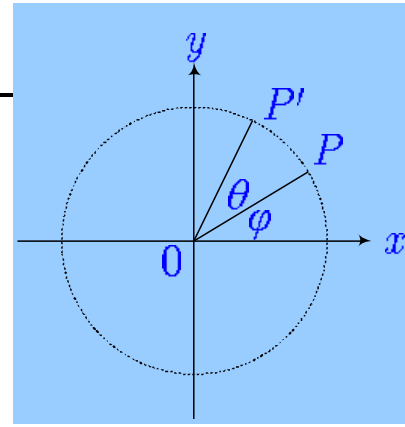
2. 旋转

（1）含义。将像素沿 xy 平面内的圆弧路径重定位。如图13-3(a)所示。



(a) 正变换　　(b) 逆变换

图13-3　旋转



（2）变换方程。规定基准点（旋转中心）为原点。

在极坐标系中，点的原始坐标为 $(x,y)=(r\cos\phi,r\sin\phi)$，变换后的坐标为

$$\begin{cases}x'=r\cos(\phi+\theta)=r\cos\phi\cos\theta-r\sin\phi\sin\theta\\ y'=r\sin(\phi+\theta)=r\cos\phi\sin\theta+r\sin\phi\cos\theta\end{cases}$$

所以变换方程为

$$\begin{cases}x'=x\cos\theta-y\sin\theta\\ y'=x\sin\theta+y\cos\theta\end{cases}$$

（3）矩阵形式。用 $R(\theta)$ 表示。将变换方程改写成矩阵形式，可得

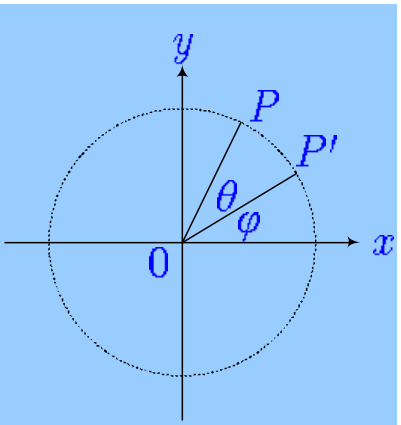
$$\begin{pmatrix}x'\\ y'\\ 1\end{pmatrix}=\begin{pmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}x\\ y\\ 1\end{pmatrix}$$

$$\begin{aligned}e^{i(\alpha+\beta)}&=e^{i\alpha}e^{i\beta}\\ &=(\cos\alpha+i\sin\alpha)(\cos\beta+i\sin\beta)\\ &=(\cos\alpha\cos\beta-\sin\alpha\sin\beta)+i(\sin\alpha\cos\beta+\cos\alpha\sin\beta)\end{aligned}$$

$$e^{i\theta}=\cos\theta+i\sin\theta$$

（4）逆变换。如图13-3(*b*)所示。易知 $R^{-1}(\theta)=R(-\theta)=R^{T}(\theta)$。变换方程为

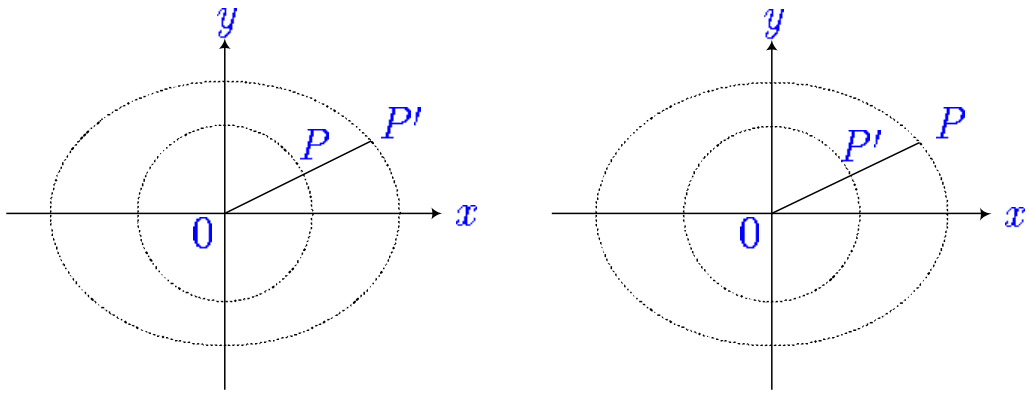
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}=\begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



$$\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}\quad\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}\quad\begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3. 缩放

（1）含义。对 x 和 y 坐标分别乘以一个系数。如图13-4(a)所示。



(a) 正变换　　(b) 逆变换

图13-4　缩放

（2）变换方程。根据含义直接可得

$$\begin{cases} x' = x \times s_x \\ y' = y \times s_y \end{cases}$$

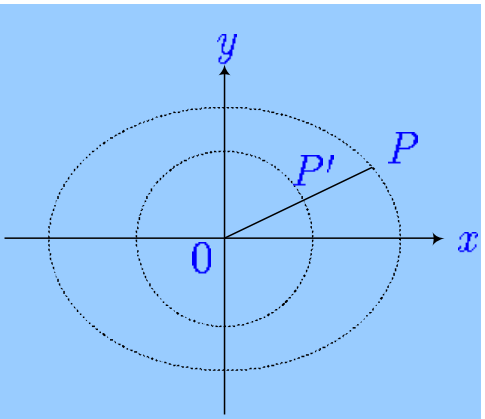
$$\begin{cases} x' = x \times s_x \\ y' = y \times s_y \end{cases}$$

（3）矩阵形式。用 $S(s_x, s_y)$ 表示。将变换方程改写成矩阵形式，可得

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

（4）逆变换。如图13-4(*b*)所示。易知 $S^{-1}(s_x, s_y) = S(1/s_x, 1/s_y)$。变换方程为

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1/s_x & 0 & 0 \\ 0 & 1/s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



$$\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

13.3.3# 变换的复合

1. 变换复合的基本方法

设对位置 P 依次进行变换 M_1 和 M_2，则变换以后的位置 P' 可以使用下列公式计算。

$$P' = M_2 \times (M_1 \times P) = (M_2 \times M_1) \times P = M \times P$$

由此可知，变换复合实际上就是计算各变换矩阵的乘积。

2. 同种变换的复合

同种变换的复合是指两个连续出现的同一类基本变换的复合。

（1）连续旋转。假定两个旋转变换的旋转角度分别是 θ_2 和 θ_1。

$$
\begin{aligned}
R(\theta_1)\times R(\theta_2) &= \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} \cos\theta_1\cos\theta_2 - \sin\theta_1\sin\theta_2 & -\cos\theta_1\sin\theta_2 - \sin\theta_1\cos\theta_2 & 0 \\ \sin\theta_1\cos\theta_2 + \cos\theta_1\sin\theta_2 & -\sin\theta_1\sin\theta_2 + \cos\theta_1\cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= R(\theta_1+\theta_2)
\end{aligned}
$$

$$
\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}
$$

（2）连续平移和连续缩放。使用与连续旋转同样的办法可以证明，两个连续的平移 $T(u_x,u_y)$ 和 $T(t_x,t_y)$ 满足 $T(t_x,t_y)\times T(u_x,u_y)=T(t_x+u_x,t_y+u_y)$，两个连续的缩放 $S(u_x,u_y)$ 和 $S(s_x,s_y)$ 满足 $S(s_x,s_y)\times S(u_x,u_y)=S(s_xu_x,s_yu_y)$。

（3）交换性。由

$$T(t_x,t_y)\times T(u_x,u_y)=T(t_x+u_x,t_y+u_y)=T(u_x,u_y)\times T(t_x,t_y)$$
$$R(\theta_1)\times R(\theta_2)=R(\theta_1+\theta_2)=R(\theta_2)\times R(\theta_1)$$
$$S(s_x,s_y)\times S(u_x,u_y)=S(s_xu_x,s_yu_y)=S(u_x,u_y)\times S(s_x,s_y)$$

可知，两个连续的同种变换在复合时可以交换顺序。

3. 一致缩放与旋转的复合

一致缩放是缩放系数为 (s,s) 的缩放变换。

（1）先缩放后旋转。

$$R(\theta)\times S(s,s)=\begin{pmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}s & 0 & 0\\ 0 & s & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}s\cos\theta & -s\sin\theta & 0\\ s\sin\theta & s\cos\theta & 0\\ 0 & 0 & 1\end{pmatrix}$$

（2）先旋转后缩放。

$$S(s,s)\times R(\theta)=\begin{pmatrix}s & 0 & 0\\ 0 & s & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}s\cos\theta & -s\sin\theta & 0\\ s\sin\theta & s\cos\theta & 0\\ 0 & 0 & 1\end{pmatrix}$$

由此可知，一致缩放与旋转在复合时可以交换缩放与旋转的顺序。

13.3.4# 几种复杂变换的构造

1. 通用基准点旋转

【问题】如图13-5所示。已知旋转角为 θ，基准点位置为 $P_r(x_r, y_r)$，请构造该旋转变换的变换矩阵。

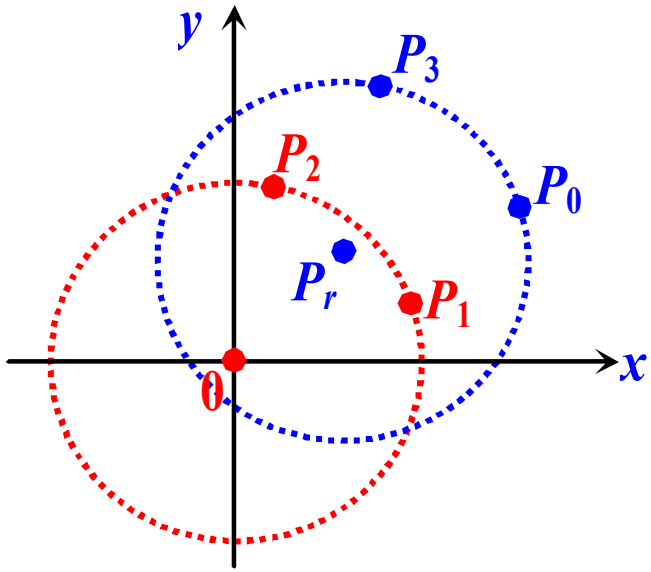
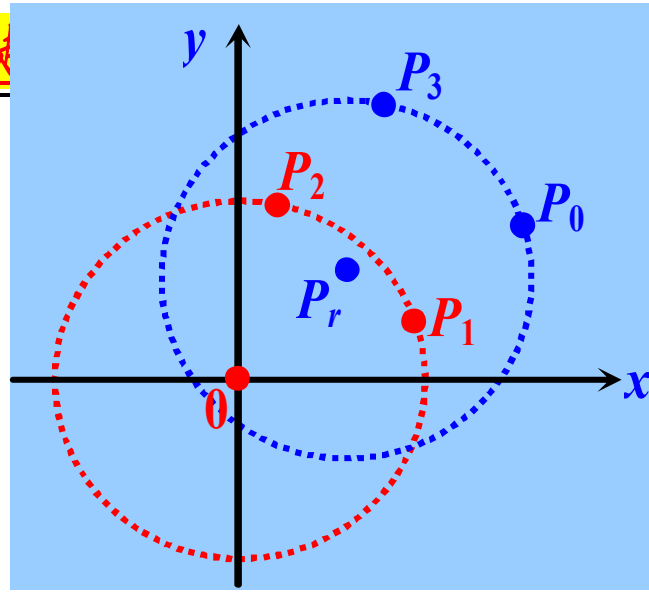


图13-5 通用基准点旋转



【变换的构造】

（1）使基准点与原点重合：$T_1 = T(-x_r, -y_r)$

$$P_1 = T_1 \times P_0$$

（2）绕原点旋转：$R = R(\theta)$

$$P_2 = R \times P_1$$

（3）使基准点回到原处：$T_2 = T(x_r, y_r)$

$$P_3 = T_2 \times P_2$$

由 $P_3 = M \times P_0 = T_2 \times R \times T_1 \times P_0$ 可知，完整变换为

$$
\begin{aligned}
M &= T_2 R T_1 \\
&= \begin{pmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} \cos\theta & -\sin\theta & x_r(1 - \cos\theta) + y_r \sin\theta \\ \sin\theta & \cos\theta & -x_r \sin\theta + y_r(1 - \cos\theta) \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

2. 通用固定点缩放

【问题】已知缩放系数为 s_x 和 s_y，固定点（缩放中心）位置为 $P_f(x_f, y_f)$，请构造该缩放变换的变换矩阵。

【变换的构造】

（1）使固定点与原点重合：$T_1 = T(-x_f, -y_f)$。

（2）以原点为固定点缩放：$S = S(s_x, s_y)$。

（3）使固定点回到原处：$T_2 = T(x_f, y_f)$。

完整变换为

$$
\begin{aligned}
M &= T_2 S T_1 \\
&= \begin{pmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

3. 含缩放的旋转矩阵

【问题】已知旋转中心和缩放中心均为 (x_0, y_0)，旋转角度为 θ，缩放系数为 (s, s)（一致缩放），请构造该带缩放的旋转变换的变换矩阵。

【变换的构造】

（1）使旋转中心与原点重合：$T_1 = T(-x_0, -y_0)$。

（2）以原点为固定点缩放：$S = S(s, s)$。

（3）以原点为旋转中心旋转：$R = R(\theta)$。

（4）使固定点回到原处：$T_2 = T(x_0, y_0)$。

完整变换为

$$
\begin{aligned}
M &= T_2RST_1 \\
&= \begin{pmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{pmatrix}
\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}
\begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix}
\begin{pmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} s\cos\theta & -s\sin\theta & x_0(1-s\cos\theta)+y_0s\sin\theta \\ s\sin\theta & s\cos\theta & -x_0s\sin\theta+y_0(1-s\cos\theta) \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

【注】因为一致缩放与旋转在复合时可以交换缩放与旋转的顺序，所以这里的步骤（2）和步骤（3）可以交换顺序。

13.3.5 关于插值方法

通常，在图像进行仿射变换和透视变换等变换以后，不能保证结果像素与源像素一一对应，所以结果像素的亮度很难直接使用源像素的亮度，必须采用某种方法从源像素的亮度计算出结果像素的亮度。通常采用插值的方法从源像素的亮度计算出结果像素的亮度，最常用的插值方法是最近邻插值和双线性插值。

1. 最近邻插值

假设需要计算结果像素 (x',y') 的亮度，则最近邻插值需要完成下述步骤。

（1）使用逆变换获得位置 (x,y)，这里的 (x,y) 不一定是一个源像素。

（2）令 $x_0=\text{int}(x)$，$y_0=\text{int}(y)$，$x_1=x_0+1$，$y_1=y_0+1$。显然，$(x_0,y_0),(x_0,y_1),(x_1,y_0),(x_1,y_1)$ 是源像素。

（3）在 $(x_0,y_0),(x_0,y_1),(x_1,y_0),(x_1,y_1)$ 中选取距离 (x,y) 最近的位置 (u,v)，将 (u,v) 处的亮度 f 作为结果像素 (x',y') 的亮度。如图13-6左侧所示。

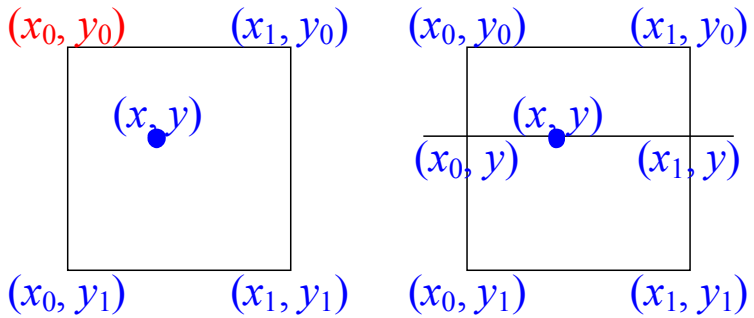


图13-6　最近邻插值与双线性插值

2. 双线性插值

假设需要计算结果像素 (x',y') 的亮度，则双线性插值需要完成最近邻插值的前2个步骤和下述步骤。

（3）设 $(x_0,y_0),(x_0,y_1),(x_1,y_0),(x_1,y_1)$ 处的亮度分别是 $f_{00},f_{01},f_{10},f_{11}$。

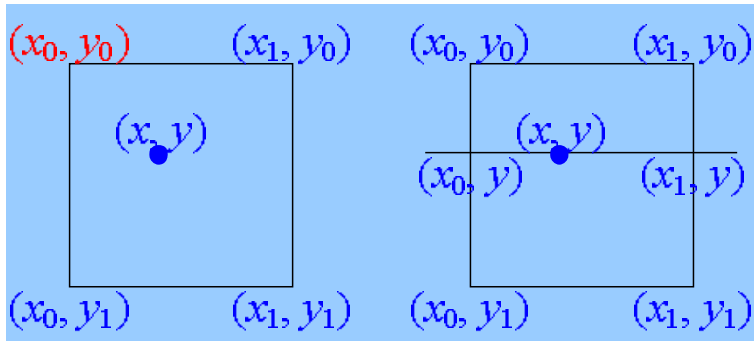
（4）用 $f_0=\dfrac{f_{01}-f_{00}}{y_1-y_0}(y-y_0)+f_{00}$ 计算 (x_0,y) 处的亮度 f_0。

（5）用 $f_1=\dfrac{f_{11}-f_{10}}{y_1-y_0}(y-y_0)+f_{10}$ 计算 (x_1,y) 处的亮度 f_1。

（6）用 $f=\dfrac{f_1-f_0}{x_1-x_0}(x-x_0)+f_0$ 计算 (x,y) 处的亮度 f。

（7）将 (x,y) 处的亮度 f 作为结果像素 (x',y') 的亮度。如图13-6右侧所示。

$$\frac{y-y_0}{x-x_0}=\frac{y_1-y_0}{x_1-x_0}\Rightarrow y=\frac{y_1-y_0}{x_1-x_0}(x-x_0)+y_0$$



13.3.6 OpenCV对仿射变换的支持

1. 相关函数

（1）翻转图像。使用flip()。

【函数原型】void flip(InputArray src, OutputArray dst, int flipCode);

【功能】垂直，水平或既垂直又水平翻转图像。

【参数】

- src：源图像。
- dst：结果图像，大小和类型与源图像一致，必要时重建。
- flipCode：指定怎样翻转图像。0表示垂直翻转，正数表示水平翻转，负数表示既垂直又水平翻转。

（2）改变图像大小。使用resize()。

【函数原型】void resize(InputArray src, OutputArray dst, Size dsize, double fx = 0, double fy = 0, int interpolation = INTER_LINEAR);

【功能】图像大小变换。

【参数】

- src：源图像。
- dst：目标图像，类型与源图像一致，必要时重建，大小由dsize或(fx, fy)指定。
- dsize：目标图像大小，(0, 0)表示由(fx, fy)指定。
- (fx, fy)：水平方向和垂直方向的缩放因子，(0, 0)表示由dsize指定。
- interpolation：插值方法，通常选用INTER_NN（最近邻插值）或INTER_LINEAR（双线性插值）。

【说明】dsize和(fx, fy)不能都等于(0, 0)。

【举例】如果目标图像已经创建，只需变换像素，可以使用如下类似的调用。

```
resize(src, dst, dst.size());
```

如果需要指定缩放系数，可以使用如下类似的调用。

```
resize(src, dst, Size(), 0.5, 0.5);
```

（3）执行仿射变换。使用warpAffine()。

【函数原型】void warpAffine(InputArray src, OutputArray dst, InputArray M, Size dsize, int flags = INTER_LINEAR);

【功能】对图像做仿射变换。

【参数】

- src：输入图像。
- dst：目标图像，类型与源图像一致，必要时会重建。
- M：2×3变换矩阵。
- dsize：目标图像大小，默认与源图像相同。
- flags：插值方法和开关选项WARP_INVERSE_MAP（逆变换）的组合。

【说明】

- 若未指定WARP_INVERSE_MAP，则进行正变换。
- 目标图像中对应输入图像边界外像素通常使用黑色填充。

（4）计算含缩放的旋转矩阵。使用getRotationMatrix2D()。

【函数原型】Mat getRotationMatrix2D(Point2f center, double angle, double scale);

【功能】计算含一致缩放的旋转变换矩阵。

【参数】

- center：输入图像的旋转中心坐标。
- angle：旋转角度（度）。正值表示逆时针旋转。
- scale：一致缩放的缩放系数。

【返回值】获得的变换矩阵（2×3）。

2. 举例说明

这里给出了一个演示程序，用于演示平移、旋转和缩放的效果。将源图像向左上平移1/5的运行结果图像如图13-7所示，将源图像绕中心逆时针旋转60度并缩小到3/4的运行结果图像如图13-8所示。

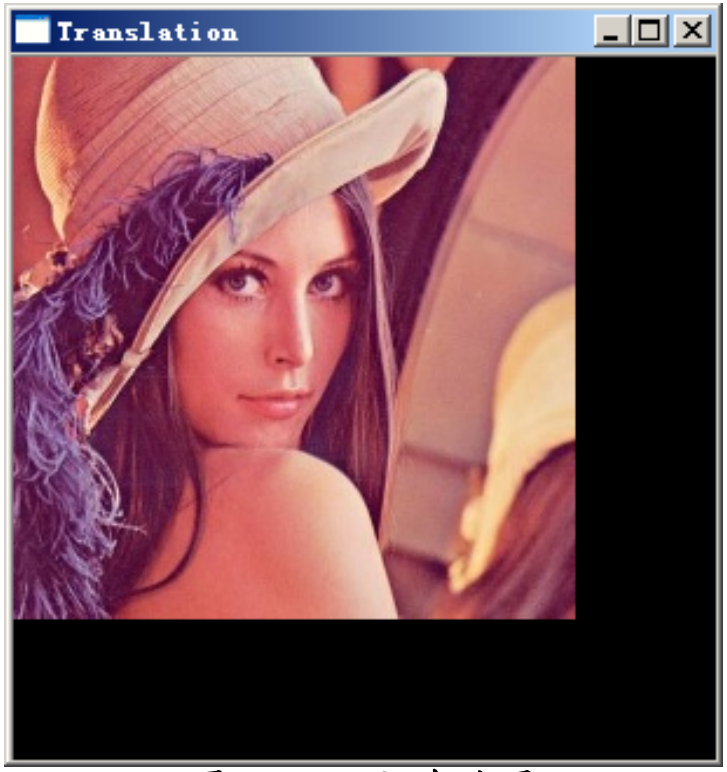


图13-7　平移效果

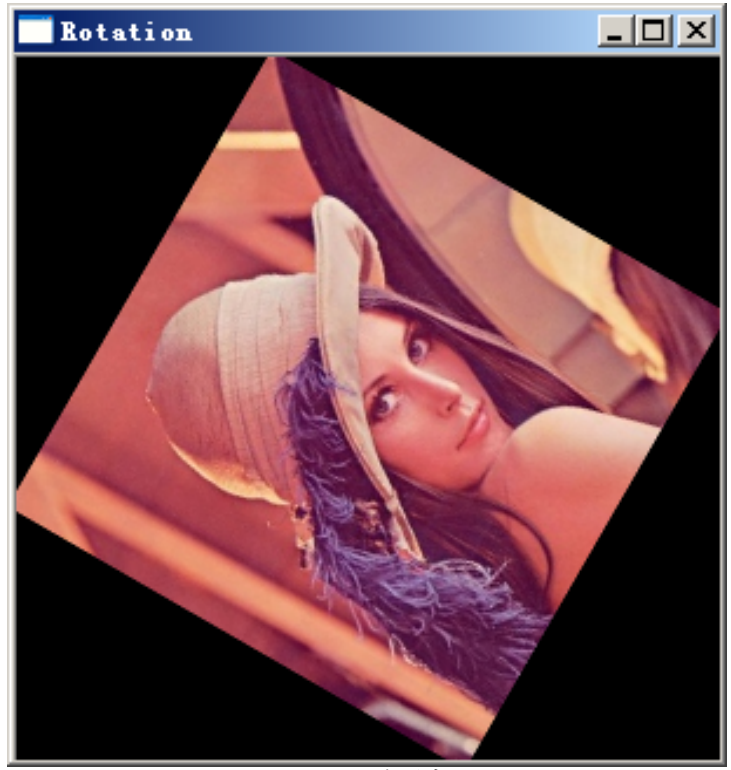


图13-8　旋转效果

```
// WarpAffine.Cpp
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   Mat X = imread("lena.jpg"); // 载入彩色图像
    if(X.empty()) return -1; // 载入图像失败
    Mat Y; // 结果图像
    Matx23f Translation // 平移矩阵
    (1, 0, -0.2 * X.cols,
     0, 1, -0.2 * X.rows
    );
    warpAffine(X, Y, Translation, Size());
    imshow("Translation", Y); // 显示平移结果
    Point2f center(X.cols * 0.5, X.rows * 0.5); // 旋转中心
    // 构造变换矩阵, 旋转中心, 角度, 缩放比例
    Mat Rotation = getRotationMatrix2D(center, 60, 0.75); // 旋转矩阵
    warpAffine(X, Y, Rotation, Size());
    imshow("Rotation", Y); // 显示旋转结果
    waitKey();
}
```

13.4 图像的透视变换♫Ж

13.4.1 透视变换的变换公式

1. 变换公式

将像素从源位置 (x,y) 变换到目标位置 (x',y') 的变换公式为

$$\begin{cases} x' = \left(a_{11}x + a_{12}y + a_{13}\right) / \left(a_{31}x + a_{32}y + 1\right) \\ y' = \left(a_{21}x + a_{22}y + a_{23}\right) / \left(a_{31}x + a_{32}y + 1\right) \end{cases}$$

显然，当 $a_{31} = a_{32} = 0$ 时，透视变换退化为仿射变换。

2. 变换公式的齐次坐标形式

在使用齐次坐标时，将像素从源位置 $(x, y, 1)$ 变换到目标位置 (x_h, y_h, h) 的变换公式为

$$\begin{cases} x_h = a_{11}x + a_{12}y + a_{13} \\ y_h = a_{21}x + a_{22}y + a_{23} \\ h = a_{31}x + a_{32}y + 1 \end{cases}$$

变换后的物理坐标 (x', y') 可通过下述方法获得。

$$\begin{cases} x' = x_h \,/\, h \\ y' = y_h \,/\, h \end{cases}$$

$$\begin{cases} x' = \left(a_{11}x + a_{12}y + a_{13}\right) / \left(a_{31}x + a_{32}y + 1\right) \\ y' = \left(a_{21}x + a_{22}y + a_{23}\right) / \left(a_{31}x + a_{32}y + 1\right) \end{cases}$$

3. 变换公式的矩阵表示

齐次坐标形式的变换公式等价于下述矩阵表示。

$$\begin{pmatrix} x_h \\ y_h \\ h \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

其中，矩阵

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{pmatrix}$$

称为透视变换的变换矩阵。

$$\begin{cases} x_h = a_{11}x + a_{12}y + a_{13} \\ y_h = a_{21}x + a_{22}y + a_{23} \\ h = a_{31}x + a_{32}y + 1 \end{cases}$$

13.4.2 获取变换矩阵

由

$$\begin{cases} x_h = a_{11}x + a_{12}y + a_{13} \\ y_h = a_{21}x + a_{22}y + a_{23} \\ h = a_{31}x + a_{32}y + 1 \end{cases}$$

和

$$\begin{cases} x_h = hx' \\ y_h = hy' \end{cases}$$

$$\begin{cases} x' = x_h / h \\ y' = y_h / h \end{cases}$$

可得

$$\begin{cases} a_{11}x + a_{12}y + a_{13} = a_{31}xx' + a_{32}yx' + x' \\ a_{21}x + a_{22}y + a_{23} = a_{31}xy' + a_{32}yy' + y' \end{cases}$$

整理后可得

$$\begin{cases} x' = a_{11}x + a_{12}y + a_{13} - a_{31}xx' - a_{32}yx' \\ y' = a_{21}x + a_{22}y + a_{23} - a_{31}xy' - a_{32}yy' \end{cases}$$

给定源图像中不共线的四个点（(x_0,y_0)、(x_1,y_1)，(x_2,y_2)和(x_3,y_3)））和目标图像中对应的点（(x_0',y_0')、(x_1',y_1')，(x_2',y_2')和(x_3',y_3')）后，可以获得下述矩阵形式的方程组。

$$\begin{pmatrix} x_0' \\ y_0' \\ x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \end{pmatrix} = \begin{pmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 & x_0x_0' & y_0x_0' \\ 0 & 0 & 0 & x_0 & y_0 & 1 & x_0y_0' & y_0y_0' \\ x_1 & y_1 & 1 & 0 & 0 & 0 & x_1x_1' & y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & x_1y_1' & y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & x_2x_2' & y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & x_2y_2' & x_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & x_3x_3' & y_3x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & x_3y_3' & x_3y_3' \end{pmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{pmatrix}$$

$$\begin{cases} x' = a_{11}x + a_{12}y + a_{13} - a_{31}xx' - a_{32}yx' \\ y' = a_{21}x + a_{22}y + a_{23} - a_{31}xy' - a_{32}yy' \end{cases}$$

上课时请勿吃喝，请勿讲话，请勿使用电话，请勿随意进出和走动。

求解该方程组，可得

$$\begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{pmatrix} = \begin{pmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 & x_0x_0' & y_0x_0' \\ 0 & 0 & 0 & x_0 & y_0 & 1 & x_0y_0' & y_0y_0' \\ x_1 & y_1 & 1 & 0 & 0 & 0 & x_1x_1' & y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & x_1y_1' & y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & x_2x_2' & y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & x_2y_2' & x_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & x_3x_3' & y_3x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & x_3y_3' & x_3y_3' \end{pmatrix}^{-1} \begin{pmatrix} x_0' \\ y_0' \\ x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \end{pmatrix}$$

从而得到透视变换的变换矩阵

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{pmatrix}$$

$$\begin{pmatrix} x_0' \\ y_0' \\ x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \end{pmatrix} = \begin{pmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 & x_0x_0' & y_0x_0' \\ 0 & 0 & 0 & x_0 & y_0 & 1 & x_0y_0' & y_0y_0' \\ x_1 & y_1 & 1 & 0 & 0 & 0 & x_1x_1' & y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & x_1y_1' & y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & x_2x_2' & y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & x_2y_2' & x_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & x_3x_3' & y_3x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & x_3y_3' & x_3y_3' \end{pmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{pmatrix}$$

13.4.3 OpenCV对透视变换的支持

1. 相关函数

（1）执行透视变换。使用warpPerspective()。

【函数原型】void warpPerspective(InputArray src, OutputArray dst, InputArray M, Size dsize, int flags = INTER_LINEAR);

【功能】对图像执行透视变换。

【参数】

- src：输入图像。
- dst：目标图像，类型与源图像一致，必要时会重建。
- M：3×3变换矩阵。
- dsize：目标图像大小，默认与源图像相同。
- flags：插值方法和开关选项WARP_INVERSE_MAP（逆变换）的组合。

【说明】

- 若未指定WARP_INVERSE_MAP，则进行正变换。
- 目标图像中对应输入图像边界外像素通常使用黑色填充。

（2）获取变换矩阵。使用getPerspectiveTransform()。

【函数原型】

- Mat getPerspectiveTransform(InputArray src, InputArray dst);
- Mat getPerspectiveTransform(const Point2f src[], const Point2f dst[]);

【功能】从四对对应点计算透视变换。

【参数】

- src：源图像中四边形顶点的坐标。
- dst：目标图像中相应四边形顶点的坐标。

【返回值】获得的变换矩阵（3×3）。

2. 举例说明

这里给出了一个演示程序，用于透视变换的效果，程序运行结果如图13-9所示。

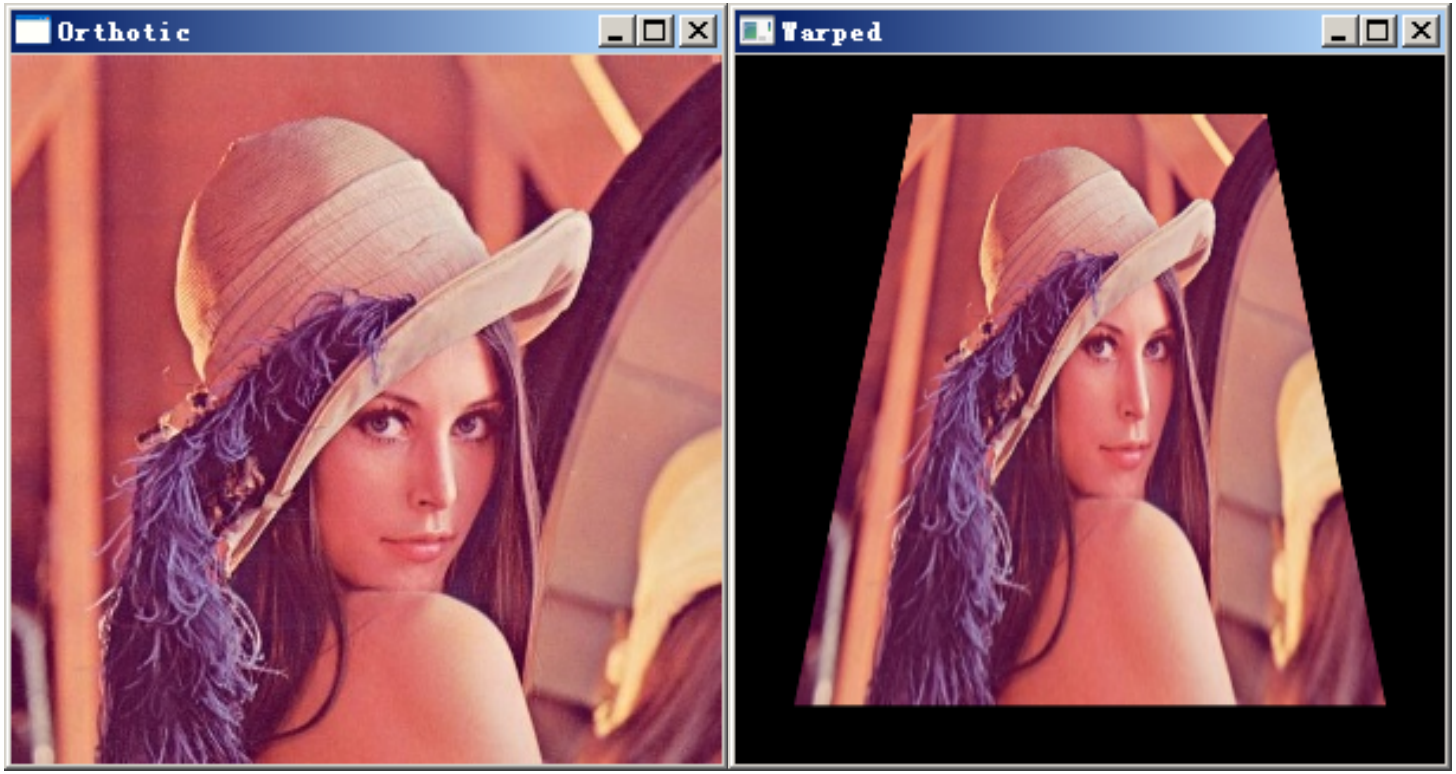


图13-9　透视效果

```
// PerspectiveSimple.Cpp
#include<opencv2/opencv.hpp>
using namespace cv;
int main()
{   Mat im = imread("lena.jpg");
    auto [w, h] = (Size2f)im.size();
    imshow("Orthotic", im); // 正形的图像
    // 由4双对应点(左上, 右上, 左下, 右下)计算变换矩阵
    auto x = w - 1, y = h - 1; // x和y的最大值
    Point2f src[] = {{0, 0}, {x, 0}, {0, y}, {x, y}};
    Point2f dst[] = {{75, 25}, {x - 75, 25}, {25, y - 25}, {x - 25, y - 25}};
    auto m = getPerspectiveTransform(src, dst); // 计算变换矩阵
    warpPerspective(im, im, m, Size()); // 执行变换
    imshow("Warped", im); // 变形的图像
    waitKey();
}
```

13.5 练习题

13.5.1# 基础知识题

1. 通过对$R(\theta_1)$和$R(\theta_2)$矩阵表示的旋转变换合并得到$R(\theta_1)\times R(\theta_2)=R(\theta_1+\theta_2)$，证明两个复合的旋转是相加的。

2. 证明下列每个操作序列对是可以交换的。

（1）两个连续的旋转。

（2）两个连续的平移。

（3）两个连续的缩放。

3. 证明一致缩放和旋转形成可交换的操作对，但一般缩放和旋转不是可交换的操作对。

4. 已知旋转角为θ，旋转中心为(x_0, y_0)，请构造该旋转变换的变换矩阵。

5. 已知旋转角为60度，旋转中心为(1, 2)，请构造该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）。

6. 已知缩放系数为s_x和s_y，固定点位置为(x_0, y_0)，请构造该缩放变换的变换矩阵。

7. 已知旋转中心和缩放中心均为(x_0, y_0)，旋转角度为θ，缩放系数为(s, s)（一致缩放），请构造该带缩放的旋转变换的变换矩阵。

13.5.2 程序设计题

1. 随机生成一幅浮点数灰度图像（大小和亮度都是随机的，大小值位于区间[128, 639]），然后将该图像变换成亮度是0~1的浮点数图像，最后变换成字节图像并显示该图像。

2. 分别使用最近邻和双线性插值将一幅彩色图像变换成另一幅大小不同的图像。例如，结果图像的宽度和高度分别是源图像的1.25倍和0.75倍。

3. 使用OpenCV编写一个演示对原图像进行缩放变换的程序。该程序首先装入一幅真彩色图像并显示该图像，然后对该图像进行缩放变换，显示得到的结果。其中旋转中心位于图像中心，缩放系数为(0.707, 0.707)，旋转角度为45度。

13.5.3 阶段实习题

1. 照片矫正程序。通过该程序在源照片中选取需要矫正的部分后，使用该程序获得矫正后的照片（如图13-10所示）。矫正部分的选取可通过选取待矫正部分四角的4个像素完成。

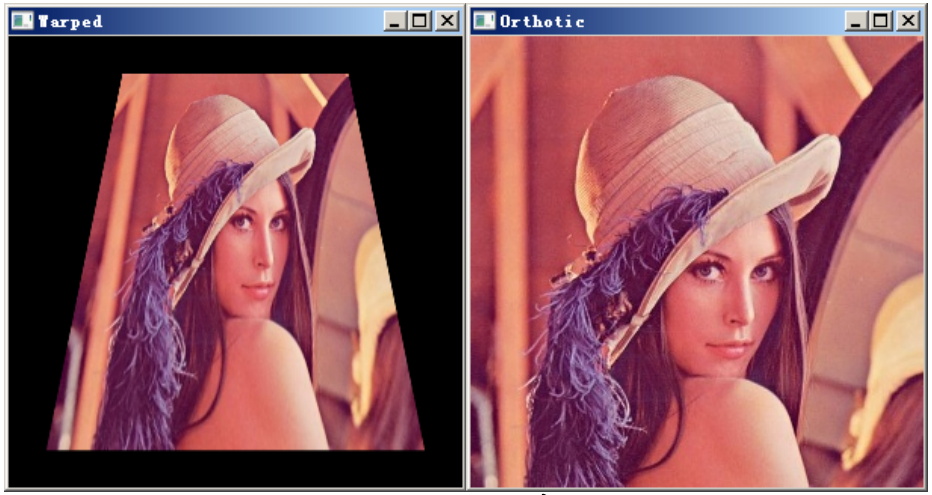


图13-10 照片矫正

2. 智能修复的实现。在一幅彩色图像上有一小块污损，用鼠标选定污损区域，然后使用该程序修复。可以首先考虑污损区域是一个矩形区域，然后考虑污损区域位于一个斜的矩形区域内，最后考虑污损区域位于一个一般的多边形区域内。实现方法是污损区域的像素用周围像素的加权平均或插值代替。