从数码技术的角度来说，图像的缩小是通过减少像素个数来实现的，因此，需要根据所期望缩小的尺寸数据，从原图像中选择合适的像素点，使图像缩小之后可以尽可能保持原有图像的概貌特征不丢失。下面介绍两种简单的图像缩小变换。

（1）基于等间隔采样的图像缩小方法

这种图像缩小方法的设计思想是，通过对画面像素的均匀采样来保持所选择到的像素仍旧可以保持像素的概貌特征。该方法的具体实现步骤为：设原图为$\mathbf{F}(i，j)$，大小为 $M\times N$（$i=1，2，\cdots，M$；$j=1，2，\cdots，N$）。缩小后的图像为$\mathbf{G}(i，j)$，大小为$k_1M\times k_2N$（当$k_1=k_2$时为按比例缩小，当$k_1\neq k_2$时为不按比例缩小，且$k_1<1$，$k_2<1$）（$i=1，2，\cdots，k_1M$；$j=1，2，\cdots，k_2N$）。则有：

$$\Delta i=1/k_1 \qquad \Delta j=1/k_2 \tag{2.3.3}$$

$$g(i,j)=f(\Delta i\cdot,\Delta j\cdot j) \tag{2.3.4}$$

下面举一个简单的例子来说明图像是如何缩小的。设原图像为：

$$\mathbf{F}=\begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} & f_{15} & f_{16} \\ f_{21} & f_{22} & f_{23} & f_{24} & f_{25} & f_{26} \\ f_{31} & f_{32} & f_{33} & f_{34} & f_{35} & f_{36} \\ f_{41} & f_{42} & f_{43} & f_{44} & f_{45} & f_{46} \end{bmatrix} \tag{2.3.5}$$

图像矩阵的大小为 4×6，将其进行缩小，缩小的倍数$k_1=0.7$，$k_1=0.6$ 则缩小图像的大小为 3×4。由式（2.3.3）计算得$\Delta i=1/k_1=1.4$，$\Delta j=1/k_2=1.7$。由式（2.3.5）可得到缩小后的图像矩阵为：

$$\mathbf{G}=\begin{bmatrix} f_{12} & f_{13} & f_{15} & f_{16} \\ f_{32} & f_{33} & f_{35} & f_{36} \\ f_{42} & f_{43} & f_{45} & f_{46} \end{bmatrix} \tag{2.3.6}$$

（2）基于局部均值的图像缩小方法

从前面的缩小算法可以看到，算法的实现非常简单。但是采用上面的方法，对没有被选取到的点的信息就无法反映在缩小后的图像中。为了解决这个问题，可以采用基于局部均值的方法来实现图像的缩小，该方法的具体实现步骤如下：

用式（2.3.3）计算采样间隔，得到Δi 和Δj；求出相邻两个采样点之间所包含的原图像的子块，即为：

$$\mathbf{F}_{(i,j)}=\begin{bmatrix} f_{\Delta i\cdot(i-1)+1,\Delta j\cdot(j-1)+1} & \cdots & f_{\Delta i\cdot(i-1)+1,\Delta j\cdot j} \\ \vdots & & \vdots \\ f_{\Delta i\cdot i,\Delta j\cdot(j-1)+1} & \cdots & f_{\Delta i\cdot i,\Delta j\cdot j} \end{bmatrix} \tag{2.3.7}$$

利用 $g(i，\ j)=\mathbf{F}(i，\ j)$ 的均值，求出缩小的图像。

同上例一样，设原图像为：

$$\mathbf{F}=\begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} & f_{15} & f_{16} \\ f_{21} & f_{22} & f_{23} & f_{24} & f_{25} & f_{26} \\ f_{31} & f_{32} & f_{33} & f_{34} & f_{35} & f_{36} \\ f_{41} & f_{42} & f_{43} & f_{44} & f_{45} & f_{46} \end{bmatrix} \tag{2.3.8}$$

大小为 4×6，将其进行缩小，缩小的倍数为缩小的倍数 $k_1=0.7$，$k_1=0.6$ 则缩小图像的大小为 3×4。由式（2.3.3）计算得 $\Delta i=1/k_1=1.4$，$\Delta j=1/k_2=1.7$。由式（2.3.7）可以将图像 F 分块为：

$$\mathbf{F}=\left[\begin{array}{cc:cc:cc} f_{11} & f_{12} & f_{13} & f_{14} & f_{15} & f_{16} \\ f_{21} & f_{22} & f_{23} & f_{24} & f_{25} & f_{26} \\ f_{31} & f_{32} & f_{33} & f_{34} & f_{35} & f_{36} \\ f_{41} & f_{42} & f_{43} & f_{44} & f_{45} & f_{46} \end{array}\right] \tag{2.3.9}$$

再由 $g(i，\ j)=\mathbf{F}(i，\ j)$ 的均值，得到缩小的图像为：

$$\mathbf{G}=\begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \end{bmatrix} \tag{2.3.10}$$

式中：$g(i，\ j)$ 为式（2.3.9）各子块的均值，如 $g_{21}=(f_{21}+f_{22}+f_{31}+f_{32})/4$。

若图像为：

$$\mathbf{F}=\begin{bmatrix} 31 & 35 & 39 & 13 & 17 & 21 \\ 32 & 36 & 10 & 14 & 18 & 22 \\ 33 & 37 & 11 & 15 & 19 & 23 \\ 34 & 38 & 12 & 16 & 20 & 24 \end{bmatrix} \tag{2.3.11}$$

按照上例缩小的比例，采用等间隔采样和采用局部均值采样得到缩小的图像分别为：

$$\mathbf{G}=\begin{bmatrix} 35 & 39 & 17 & 21 \\ 37 & 11 & 19 & 23 \\ 38 & 12 & 20 & 24 \end{bmatrix} \tag{2.3.12}$$

$$\mathbf{G}=\begin{bmatrix}33 & 39 & 15 & 21\\ 35 & 11 & 17 & 23\\ 36 & 12 & 18 & 34\end{bmatrix} \tag{2.3.13}$$

2 图像的比例放大变换

图像在缩小操作中，是在现有的信息里如何挑选所需要的有用信息。而在图像的放大操作中，则需要对尺寸放大后所多出来的空格填入适当的像素值，这是信息的估计问题，所以较图像的缩小要难一些。由于图像相邻像素之间的相关性很强，可以利用这个相关性来实现图像的放大。与图像缩小相同，按比例放大不会引起图像的畸变，而不按比例放大则会产生图像的畸变。图像放大一般采用最近邻域法、线性插值法、三次卷积法。

（1）最近邻域法

一般地，按比例将原图像放大 k 倍时，如果按照最近邻域法，则需要将一个像素值添在新图像的 $k\times k$ 的子块中。式（2.3.14）为图像 $\mathbf{F}$ 的矩阵，该图像放大 3 倍得到图像 $\mathbf{G}$ 的矩阵用式（2.3.15）表示。图 2.3.3 为放大 5 倍的示意图。显然，如果放大倍数太大，按照这种方法处理会出现马赛克效应。

$$\mathbf{F}=\begin{bmatrix}f_{11} & f_{12} & f_{13}\\ f_{21} & f_{22} & f_{23}\\ f_{31} & f_{32} & f_{33}\end{bmatrix} \tag{2.3.14}$$

$$G=\left[\begin{array}{ccc:ccc:ccc}
f_{11} & f_{11} & f_{11} & f_{12} & f_{12} & f_{12} & f_{13} & f_{13} & f_{13}\\
f_{11} & f_{11} & f_{11} & f_{12} & f_{12} & f_{12} & f_{13} & f_{13} & f_{13}\\
f_{11} & f_{11} & f_{11} & f_{12} & f_{12} & f_{12} & f_{13} & f_{13} & f_{13}\\
\hdashline
f_{21} & f_{21} & f_{21} & f_{22} & f_{22} & f_{22} & f_{23} & f_{23} & f_{23}\\
f_{21} & f_{21} & f_{21} & f_{22} & f_{22} & f_{22} & f_{23} & f_{23} & f_{23}\\
f_{21} & f_{21} & f_{21} & f_{22} & f_{22} & f_{22} & f_{23} & f_{23} & f_{23}\\
\hdashline
f_{31} & f_{31} & f_{31} & f_{32} & f_{32} & f_{32} & f_{33} & f_{23} & f_{23}\\
f_{31} & f_{31} & f_{31} & f_{32} & f_{32} & f_{32} & f_{23} & f_{23} & f_{23}\\
f_{31} & f_{31} & f_{31} & f_{32} & f_{32} & f_{32} & f_{23} & f_{23} & f_{23}
\end{array}\right] \tag{2.3.15}$$

（2）双线性插值法

为了提高几何变换后的图像质量，常采用双线性插值法。该方法的原理是：当求出的分数地址与像素点不一致时，求出周围四个像素点的距离比，根据该比率，由 4 个邻域的像素灰度值进行双线性插值，如图 2.3.4 所示。

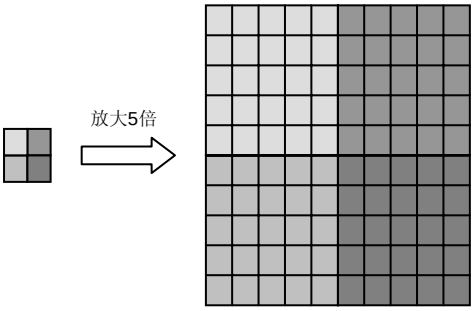


图 2.3.3　按最近邻域法放大 5 倍的图像

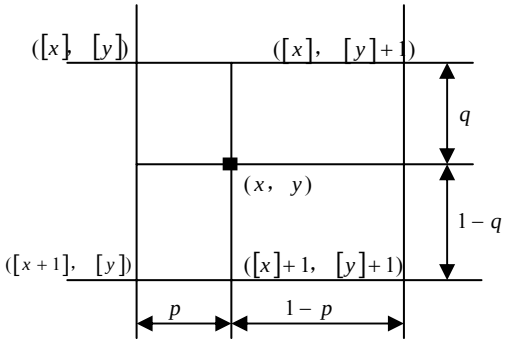


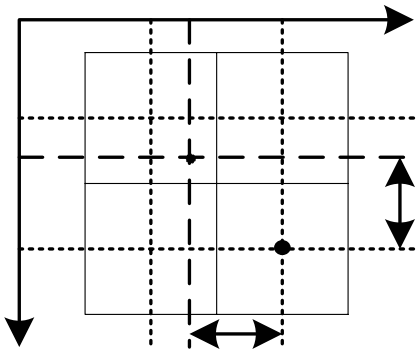
图 2.3.4　双线性插值算法示意图

简化后的灰度值的计算式如下：

$$g(x,y)=(1-q)\{(1-p)\times g([x],[y])+p\times g([x],[y]+1)\}+ q\{(1-p)\times g([x]+1,[y])+p\times g([x]+1,[y]+1)\} \quad (2.3.16)$$

式中：$g(x，y)$为坐标$(x，y)$处的灰度值；$[x]$和$[y]$为分别为不大于x和y的整数。

具体推证：



中心像元 s 坐标（i, j），则像元 t 的坐标为（$i-u$，$j-v$）。根据双线性插值算法原理，t 像元的灰度值 $f(t)$ 是由周围的四个像元灰度值决定，这四个像元坐标分别是（i-1, j-1），（i-1, j），（i, j-1），（i, j）。根据下述公式进行双线性内插求解像元 t 的属性值：

$f(i-u, j-1)=(1-u)\times f(i-1, j-1)+u\times f(i, j-1)$

$f(i-u, j)=(1-u)\times f(i-1, j)+u\times f(i, j)$

$f(i-u, j-v)=(1-v)\times f(i-u, j-1)+v\times f(i-u, j+1)$。

当然，也可先纵向求解 $f(i-1, j-v), f(i, j-v)$；再求解 $f(i-u, j-v)$。

（3）三次卷积插值法　《遥感数字图像处理教程》

三次卷积插值是用(x, y)周围的 16 个格网点（像元）灰度按三次多项式进行内插的高

精度算法，即

$$f(x,y)=\sum_{k}\sum_{l}f(x_k,y_l)\omega(x_k-x)\omega(y_l-y) \qquad (2.3.17)$$

式中，(x_k,y_l)表示(x,y)周围的格网点，$\omega(t)$为内插函数。最理想的采样内插函数为辛克 sinc 函数，但使用不方便，计算时间也长，实践中常用多项式逼近该函数，如图 2.3.5 所示。

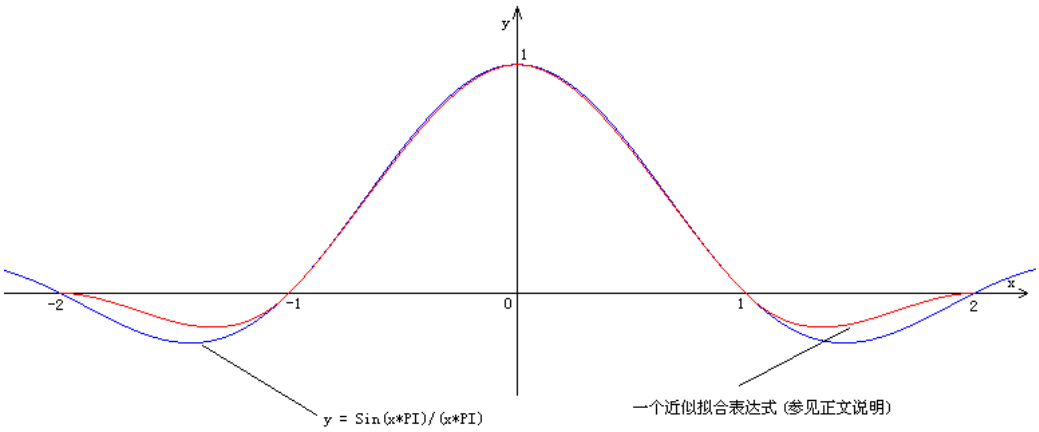
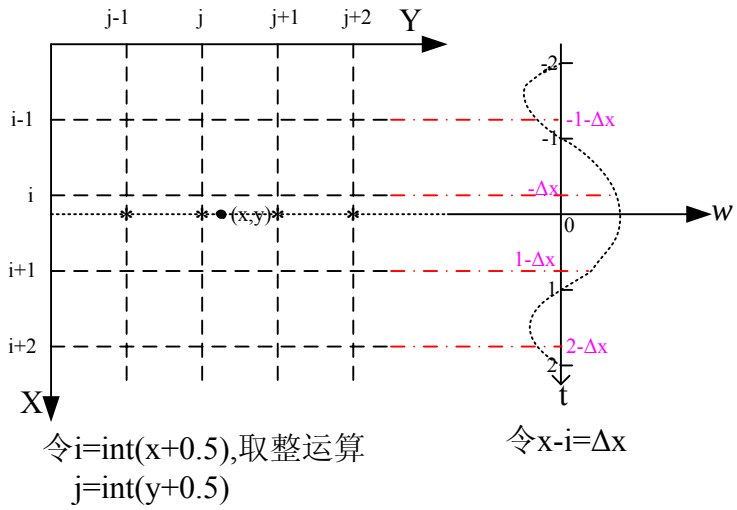


图 2.3.5 多项式逼近辛克 sinc 函数

三次卷积插值是利用多项式$\omega(x)$逼近最佳插值函数（辛克函数），多项式$\omega(x)$表示如下：

$$\omega(x)=\begin{cases}1-2x^2+|x|^3 & 0\le|x|\le1\\ 4-8|x|+5x^2-|x|^3 & 1\le|x|\le2\\ 0 & 2\le|x|\end{cases}$$

三次多项式$\omega(x)$近似表示灰度内插时周围像元的灰度值对内插点灰度值的贡献大小。可先在某一方向上内插，如先在X方向上，每4个值依次内插4次，求出 $f(x,j-1)$，$f(x,j)$，$f(x,j+1)$， $f(x,j+2)$。



其中，$$\begin{aligned} f(x,j-1)=&\omega(-1-\Delta x)\times f(i-1,j-1)+\omega(\Delta x)\times f(i,j-1)\\ &+\omega(1-\Delta x)\times f(i+1,j-1)+\omega(2-\Delta x)\times f(i+2,j-1)\end{aligned}$$；然后在 Y 方向上，同理。写成矩阵形式为

$$f(x,y)=\begin{bmatrix}\omega(-1-\Delta x) & \omega(\Delta x) & \omega(1-\Delta x) & \omega(2-\Delta x)\end{bmatrix}\times \begin{bmatrix} f(i-1,j-1) & f(i-1,j) & f(i-1,j+1) & f(i-1,j+2)\\ f(i,j-1) & f(i,j) & f(i,j+1) & f(i,j+2)\\ f(i+1,j-1) & f(i+1,j) & f(i+1,j+1) & f(i+1,j+2)\\ f(i+2,j-1) & f(i+2,j) & f(i+2,j+1) & f(i+2,j+2)\end{bmatrix}\times \begin{bmatrix}\omega(-1-\Delta y)\\ \omega(\Delta y)\\ \omega(1-\Delta y)\\ \omega(2-\Delta y)\end{bmatrix} \tag{2.3.18}$$

2.3.2 图像错切变换

图像的错切变换实际上是平面景物在投影平面上的非垂直投影。错切使图像中的图形产生扭变，这种扭变只在一个方向上产生，即分别称为水平方向错切或垂直方向错切。下面分别对其进行简要阐述。

1 水平方向错切

根据图像错切定义，在水平方向上的错切是指图形在水平方向上发生了扭变。如图 2.3.6 所示，当图 2.3.6（a）发生了水平方向的错切之后，图 2.3.6（b）所示矩形的水平方向上的边扭变成斜边，而垂直方向上的边不变。图像在水平方向上错切的数学表达式为：

$$\begin{cases} x'=x+by\\ y'=y\end{cases} \tag{2.3.19}$$

式中：$(x，y)$ 为原图像的坐标；$(x'，y')$ 为错切后的图像坐标。

根据式（2.3.19），错切时图形的列坐标不变，行坐标随原坐标 $(x，y)$ 和系数 b 作线性

变化，$b=\tan(\theta)$。若$b>0$，图形沿x轴正方向作错切；若$b<0$，图形沿x轴负方向作错切。

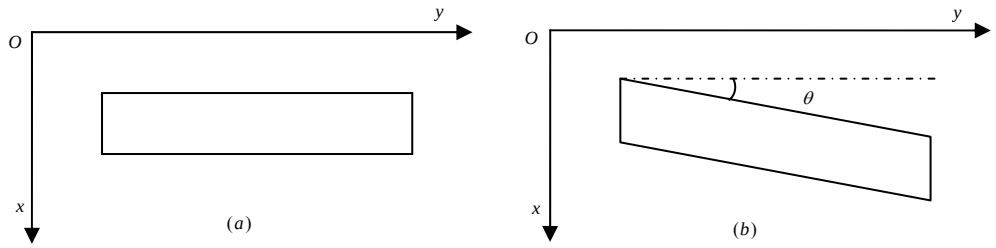


图 2.3.6 水平方向错切示意图

2 垂直方向错切

图像在垂直方向上的错切，是指图形在垂直方向上的扭变。如图 2.3.7 所示，当图 2.3.7（a）发生了垂直方向的错切之后，图 2.3.7（b）所示矩形的水平方向上的边不变，垂直方向上的边扭变成斜边。图像在垂直方向上错切的数学表达式为：

$$\begin{cases} x' = x \\ y' = y + dx \end{cases} \tag{2.3.20}$$

式中：$(x，y)$为原图像的坐标；$(x'，y')$为错切后的图像坐标。

根据式（2.3.20），错切时图形的**行坐标**不变，列坐标随原坐标$(x，y)$和系数d做线性变化，$d=\tan(\theta)$。若$d>0$，图形沿y轴正方向错切；若$d<0$，图形沿y轴负方向错切。

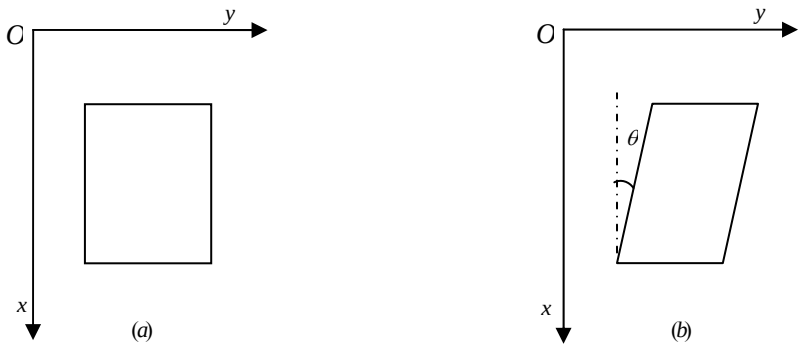


图 2.3.7 垂直方向错切示意图

3 利用错切实现图像的旋转

利用三角函数的性质，可以利用错切来实现图像的旋转。因为：

$$\begin{bmatrix} 1 & -\tan\dfrac{\theta}{2} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ \sin\theta & 1 \end{bmatrix}\begin{bmatrix} 1 & -\tan\dfrac{\theta}{2} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{2.3.21}$$

图像旋转θ角度用矩阵形式表示为：

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.3.22}$$

在 x 方向上和 y 方向上的错切用矩阵形式表示为：

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ d & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.3.23}$$

所以，图像旋转可以分解成三次图像的错切来实现。

从一个简单的例子来体会该方法的思想。设原图像为：

$$\mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \tag{2.3.24}$$

要进行逆时针 30° 的旋转。

首先按照式（2.3.24）进行第 1 次 x 方向上的错切，即 $b = -\tan 15° = -0.268$ 得到第 1 次错切结果为：

$$\mathbf{G_1} = \begin{bmatrix} 0 & 0 & f_{13} \\ f_{11} & f_{12} & f_{23} \\ f_{21} & f_{22} & f_{33} \\ f_{31} & f_{32} & 0 \end{bmatrix} \tag{2.3.25}$$

接下来按照式（2.3.25），进行 y 方向上的错切，即 $d = \sin 30° = 0.5$ 得到第 2 次错切结果为：

$$\mathbf{G_2} = \begin{bmatrix} 0 & 0 & f_{13} & 0 \\ f_{11} & f_{12} & f_{23} & 0 \\ 0 & f_{21} & f_{22} & f_{33} \\ 0 & f_{31} & f_{32} & 0 \end{bmatrix} \tag{2.3.26}$$

最后按照式（2.3.26）再进行一次 x 方向上的错切，即 $b = -\tan 15° = -0.268$，得到第 3 次错切结果为：

$$\mathbf{G}_3 = \begin{bmatrix} 0 & 0 & f_{13} & 0 \\ f_{11} & f_{12} & f_{23} & f_{33} \\ 0 & f_{21} & f_{22} & 0 \\ 0 & f_{31} & f_{32} & 0 \end{bmatrix} \tag{2.3.25}$$

直接旋转的结果为：

$$\mathbf{G}_4=\begin{bmatrix}0 & 0 & f_{13} & 0\\ f_{11} & f_{12} & 0 & f_{23}\\ 0 & f_{21} & f_{22} & 0\\ 0 & f_{31} & f_{32} & 0\end{bmatrix} \tag{2.3.26}$$

与直接旋转的结果相比，像素的排列基本相同（差异为坐标取整时的舍入误差），但由于每次错切总有一个坐标是顺序排列的，因此，最终的结果不存在空穴，无须后续的插值计算。

2.4 图像复合变换

图像的复合变换是指对给定的图像连续施行若干次如前所述的平移、镜像、比例缩放、旋转等基本变换后所完成的变换，图像的复合变换又叫级联变换。

利用齐次坐标，对给定的图像依次按一定顺序连续施行若干次基本变换，其变换的矩阵仍然可以用 3×3 阶的矩阵表示，而且从数学上可以证明，复合变换的矩阵等于基本变换的矩阵按顺序依次相乘得到的组合矩阵。设对给定的图像依次进行了基本变换 F_1，F_2，…，F_n 它们的变换矩阵分别为 T_1，T_2，…，T_n，则图像复合变换的矩阵 T 可以表示为：$T=T_NT_{N-1}\cdots T_1$。

2.4.1 复合平移

设某个图像先平移到新的位置 $P_1(x_1,\ y_1)$ 后，再将图像平移到 $P_2(x_2,\ y_2)$ 的位置，则复合平移矩阵为：

$$\mathbf{T}=\mathbf{T}_1\mathbf{T}_2=\begin{bmatrix}1 & 0 & x_1\\ 0 & 1 & y_1\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & x_2\\ 0 & 1 & y_2\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 0 & x_1+x_2\\ 0 & 1 & y_1+y_2\\ 0 & 0 & 1\end{bmatrix} \tag{2.4.1}$$

由此可见，尽管一些顺序的平移，用到矩阵的乘法，但最后合成的平移矩阵，只需对平移常量作加法运算。

2.4.2 复合比例

同样，对某个图像连续进行比例变换，最后合成的复合比例矩阵，只要对比例常量做乘法运算即可。复合比例矩阵如下：

$$\mathbf{T}=\mathbf{T}_1\mathbf{T}_2=\begin{bmatrix}a_1 & 0 & 0\\ 0 & d_1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}a_2 & 0 & 0\\ 0 & d_2 & 0\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}a_1a_2 & 0 & 0\\ 0 & d_1d_2 & 0\\ 0 & 0 & 1\end{bmatrix} \tag{2.4.2}$$

2.4.3 复合旋转

类似地，对某个图像连续进行旋转变换，最后合成的旋转变换矩阵等于两次旋转角度的和，复合旋转变换矩阵如下：

$$\begin{aligned}\mathbf{T}=\mathbf{T}_1\mathbf{T}_2&=\begin{bmatrix}\cos\theta_1 & \sin\theta_1 & 0\\ -\sin\theta_1 & \cos\theta_1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos\theta_2 & \sin\theta_2 & 0\\ -\sin\theta_2 & \cos\theta_2 & 0\\ 0 & 0 & 1\end{bmatrix}\\&=\begin{bmatrix}\cos(\theta_1+\theta_2) & \sin(\theta_1+\theta_2) & 0\\ -\sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0\\ 0 & 0 & 1\end{bmatrix}\end{aligned}\tag{2.4.3}$$

上述均为相对原点（图像中央）做比例、旋转等变换，如果要相对某一个参考点作变换，则要使用含有不同种基本变换的图像复合变换。不同的复合变换，其变换过程不同，但是无论它的变换过程多么复杂，都可以分解成一系列基本变换。相应地，使用齐次坐标后，图像复合变换的矩阵由一系列图像基本几何变换矩阵依次相乘而得到。