基于 OpenCV 的仿射变换研究与应用

管焱然[1],管有庆[2]
(1. 东华大学,上海 201620;
2. 南京邮电大学,江苏 南京 210003)

摘 要:仿射变换是一种常用的图像几何变换。使用仿射变换矩阵能够方便地描述图像的线性变换以及平移等非线性变换。介绍了仿射变换的数学原理,并借助 OpenCV 函数库的内部函数对仿射变换的算法进行具体分析。通过在原图像和目标图像上分别找出三个不共线的点,并利用这两组三点间的一一映射构建方程组,并求解得出一个 2×3 的仿射变换矩阵 M;再根据所求得的仿射变换矩阵 M,对图像上每一个点进行变换,在这个过程中可采用最近邻插值算法、线性插值算法、三次样条插值算法和兰索斯插值算法进行实现。给出了一个图像实例对仿射变换进行说明。

关键词:图像处理;仿射变换;矩阵;OpenCV;算法
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2016)12-0058-06
doi:10.3969/j.issn.1673-629X.2016.12.013

Research and Application of Affine Transformation Based on OpenCV

GUAN Yan-ran[1], GUAN You-qing[2]
(1. Donghua University, Shanghai 201620, China;
2. Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: Affine transformation is a kind of frequently used geometric image transformation in the field of computer image processing. An affine transformation matrix can be used to describe any linear transformation and one kind of nonlinear transformation. It gives an introduction to the mathematical principles of affine transformation and makes an analysis of the algorithms of affine transformation basing on the OpenCV function library. If one image is related to another by an affine transformation, three pairs of corresponding points of the two images are needed to solve for the 2×3 affine transformation matrix M, which is used to apply the transformation to the source image. A series of interpolation algorithms can be employed such as nearest neighbor interpolation, linear interpolation, cubic spline interpolation and Lanczos interpolation to implement the affine transformation. An example is given concerning affine transformation of an image.

Key words: image processing; affine transformation; matrix; OpenCV; algorithm

0 引 言

在向量空间中,一个线性变换可以用一个矩阵去乘以一个向量的方式来表示,这种方法在计算机处理图像变换的过程中极为常用。然而一种常见的几何变换却不属于线性变换,那就是平移。

为了扩充线性变换,实现更多的功能,需要引入仿射变换的概念,并找到合适的数学方式来描述这一变换。

仿射变换在计算机图像处理中有着重要的作用,在模式识别、图像配准、图像增强、图像恢复、特征提取、图像编码压缩等方面都需要应用到。文献[1]研究了一种基于仿射变换的形状配准模糊算法;文献[2]通过仿射变换图像间的关系,提出了一种频域分析法;文献[3]利用仿射变换的相关性质,提出了一种用于人脸识别的参数回归算法;文献[4]研究了仿射变换在增强现实中的应用;文献[5]研究了极坐标系下的仿射变换及其在图像拼接和遥感图像配准方面的应用;文献[6-7]研究了仿射变换的目标跟踪算法;文献[8-10]研究了仿射变换的特征和性质;文献[11-12]研究了仿射变换在加密中的应用。

文中在介绍仿射变换数学原理的基础上详细分析了 OpenCV 内部函数处理仿射变换的算法,并调用 OpenCV 内部函数实现了图像的仿射变换。

收稿日期:2016-02-26 **修回日期**:2016-06-08 **网络出版时间**:2016-11-22
基金项目:江苏省高校自然科学研究计划项目(05KJD520146)
作者简介:管焱然(1994-),男,研究方向为计算机应用;管有庆,副研究员,硕士生导师,研究方向为数据库、通信软件和下一代网络等。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20161122.1227.032.html

1　仿射变换

1.1　平面上仿射变换的定义

定义 1(仿射变换的几何定义):平面上的点之间的映射如果满足下列条件:

(1)任何共线的三点的象仍是共线的三点;

(2)任何共线三点的简比(注:共线三点 A,B,C 组成的两个有向线段 AC 和 BC 的量的比 AC/BC 称为 A,B,C 的简比)不变。

则此映射称为平面上的仿射变换。

定义 2(仿射变换的代数定义):R^2 到自身的一个变换 f,如果对于 R^2 中任意向量 $v(x,y)$,与它的象 $f(v)=v'(x',y')$ 之间的关系由式(1)确定,则变换 f 叫做 R^2 上的仿射变换;式(1)称为仿射变换公式。

$$\begin{cases} x' = a_{11}x + a_{12}y + a_{13} \\ y' = a_{21}x + a_{22}y + a_{23} \end{cases}, \quad \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \neq 0 \tag{1}$$

1.2　仿射变换的基本性质

由仿射变换的几何定义可知,仿射变换具有如下基本性质[13]:

(1)保持二维图形的平直性,即共线点经过仿射变换仍为共线点。

(2)保持二维图形的平行性,即平行直线经过仿射变换仍为平行直线,但向量间的夹角可能会改变。

(3)保持共线三点的简比,即保持两平行线段的比值不变。

1.3　用线性变换表示仿射变换

由式(1)可知,在二维平面上,并非所有的仿射变换都是线性的。当式(1)中 a_{13} 或 a_{23} 不为 0 时,该公式所确定的变换 f 就是非线性的,因为此时对于 R^2 中的任意两个向量 $\boldsymbol{v}_1$ 和 $\boldsymbol{v}_2$,$f(\boldsymbol{v}_1+\boldsymbol{v}_2) \neq f(\boldsymbol{v}_1)+f(\boldsymbol{v}_2)$。

然而,可以用线性变换来表示仿射变换。对于 R^2 中的任意点 (x,y),可以用 R^3 中的齐次坐标 $(x,y,1)$ 来表示,这时可以将式(1)表示的仿射变换用矩阵乘法来实现:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2}$$

将式(2) 抽象为分块矩阵的形式:

$$\begin{bmatrix} \boldsymbol{v}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ 1 \end{bmatrix} \tag{3}$$

式(3) 就是计算机图形学中对仿射变换的处理方式,其中 $\boldsymbol{v}' = \begin{bmatrix} x' \\ y' \end{bmatrix}$,$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$,$\boldsymbol{b} = \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix}$,$\boldsymbol{v} = \begin{bmatrix} x \\ y \end{bmatrix}$。通常,使用一个 2×3 的矩阵来表示仿射变换,即:

$$\boldsymbol{M} = [\boldsymbol{A} \quad \boldsymbol{b}] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \tag{4}$$

其中,$\boldsymbol{M}$ 就是一个仿射变换矩阵,在 OpenCV 的内部函数中就是通过计算这样的仿射变换矩阵,并以此为参数来实现二维图形的仿射变换。

2　几种典型的仿射变换

通过观察式(1)可以发现,二维平面上任意一个向量 $\boldsymbol{v}(x,y)$ 到 $\boldsymbol{v}'(x',y')$ 的仿射变换都可以拆分成一个线性变换和一个平移,即乘以一个矩阵再加上一个向量:

$$\boldsymbol{v}' = \boldsymbol{A}\boldsymbol{v} + \boldsymbol{b} \tag{5}$$

其中,$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$;$\boldsymbol{b} = \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix}$。

不难发现,像平移(Translation)、缩放(Scale)、翻转(Flip)、旋转(Rotation)和错切(Shear)等等常见的二维图形的几何变换都是仿射变换的特例。

2.1　平　移

只需令式(5) 中的矩阵 $\boldsymbol{A}$ 为单位矩阵 $\boldsymbol{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$,再确定为平移的方向向量 $\boldsymbol{b} = \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix}$,即可实现二维平面上任意一点按向量 $\boldsymbol{b}$ 的平移。平移的仿射变换矩阵 $\boldsymbol{M} = \begin{bmatrix} 1 & 0 & a_{13} \\ 0 & 1 & a_{23} \end{bmatrix}$。平移变换示例见图 1(a)。

2.2　缩　放

二维平面上图形的均匀缩放可以通过将缩放因子 k 乘以单位矩阵 $\boldsymbol{I}$ 实现,即令式(5) 中的矩阵 $\boldsymbol{A} = \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$,平移的方向向量 $\boldsymbol{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$。均匀缩放的仿射变换矩阵 $\boldsymbol{M} = \begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \end{bmatrix}$。

二维平面上图形的非均匀缩放可以通过向单位矩阵 $\boldsymbol{I}$ 分别乘以 x 轴方向的缩放因子 k_x 和 y 轴方向的缩放因子 k_y 实现,即令式(5)中的矩阵 $\boldsymbol{A} = \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix}$,令平移的方向向量 $\boldsymbol{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$。非均匀缩放的仿射变换矩阵 $\boldsymbol{M} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \end{bmatrix}$。缩放变换示例见图 1(b)。

2.3　翻　转

二维平面上任意一点以 x 轴为对称轴的翻转可以通过令式(5)中的 $\boldsymbol{A} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$、$\boldsymbol{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ 来实现。以 x 轴为对称轴的翻转的仿射变换矩阵 $\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$。

二维平面上任意一点以 y 轴为对称轴的翻转可以通过令式(5)中的矩阵 $\boldsymbol{A}=\begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix}$、$\boldsymbol{b}=\begin{bmatrix}0\\0\end{bmatrix}$ 来实现。以 y 轴为对称轴的翻转的仿射变换矩阵 $\boldsymbol{M}=\begin{bmatrix}-1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}$。翻转变换示例见图 1(c)。

2.4 旋 转

二维平面上任意一点以原点为中心逆时针旋转 θ 弧度的旋转可以通过令式(5)中 $\boldsymbol{A}=\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}$、$\boldsymbol{b}=\begin{bmatrix}0\\0\end{bmatrix}$ 来实现。以原点为中心逆时针旋转 θ 弧度的旋转的仿射变换矩阵 $\boldsymbol{M}=\begin{bmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\end{bmatrix}$。以原点为中心旋转变换示例见图 1(d)。

二维平面上任意一点以点 (x_0,y_0) 为中心逆时针旋转 θ 弧度的旋转可以由两次平移和一次绕原点旋转复合而成,即该点先按 $(-x_0,-y_0)$ 进行平移,然后绕原点逆时针旋转 θ 弧度,最后按 (x_0,y_0) 进行平移。这样,经计算得到实现旋转的矩阵 $\boldsymbol{A}=\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}$,总共进行的平移为 $\boldsymbol{b}=\begin{bmatrix}x_0(1-\cos\theta)+y_0\sin\theta\\ y_0(1-\cos\theta)-x_0\sin\theta\end{bmatrix}$。以点 (x_0,y_0) 为中心逆时针旋转 θ 弧度的仿射变换矩阵

$$\boldsymbol{M}=\begin{bmatrix}\cos\theta & -\sin\theta & x_0(1-\cos\theta)+y_0\cdot\sin\theta\\ \sin\theta & \cos\theta & y_0(1-\cos\theta)-x_0\cdot\sin\theta\end{bmatrix}$$

。以点 (x_0,y_0) 为中心的旋转变换示例见图 1(e)。

2.5 错 切

二维图形的错切指的是图形上的点沿着某一指定方向产生不等量移动而引起图形变形的一种变换。错切变换的一个典型例子就是平行四边形因其不稳定性而产生的形变,即以平行四边形的一条边不动然后拖动另一对边的顶点,这种变换就叫错切。

沿 x 轴方向的错切可以通过令式(5)中的 $\boldsymbol{A}=\begin{bmatrix}1 & \mathrm{sh}_x\\ 0 & 1\end{bmatrix}$、$\boldsymbol{b}=\begin{bmatrix}0\\0\end{bmatrix}$ 来实现。其中,sh_x 为 x 轴方向错切的变换系数,$\mathrm{sh}_x=\tan\alpha$,α 为沿 x 轴方向的错切角度。沿 x 轴方向的错切的仿射变换矩阵 $\boldsymbol{M}=\begin{bmatrix}1 & \mathrm{sh}_x & 0\\ 0 & 1 & 0\end{bmatrix}$。沿 x 轴方向错切变换示例见图 1(f)。

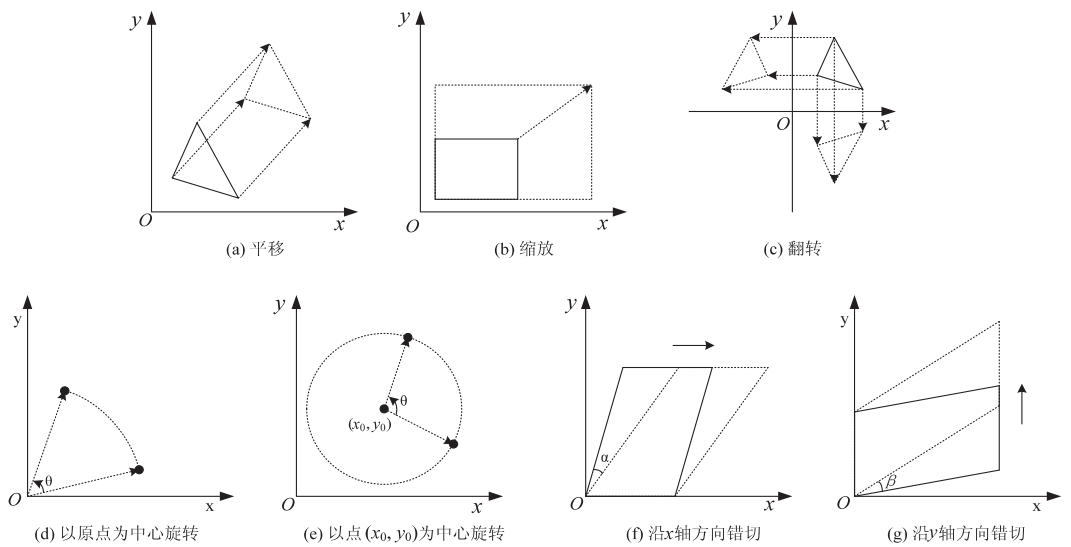
沿 y 轴方向的错切可以通过令式(5)中的 $\boldsymbol{A}=\begin{bmatrix}1 & 0\\ \mathrm{sh}_y & 1\end{bmatrix}$、$\boldsymbol{b}=\begin{bmatrix}0\\0\end{bmatrix}$ 来实现。其中,sh_y 为 y 轴方向错切的变换系数,$\mathrm{sh}_y=\tan\beta$,β 为沿 y 轴方向的错切角度。沿 y 轴方向的错切的仿射变换矩阵 $\boldsymbol{M}=\begin{bmatrix}1 & 0 & 0\\ \mathrm{sh}_y & 1 & 0\end{bmatrix}$。沿 y 轴方向错切变换示例见图 1(g)。

任何一个一般错切均可以由一个沿 x 轴方向的错切和一个沿 y 轴方向的错切复合而成,故一个一般错切可以通过令式(5)中的 $\boldsymbol{A}=\begin{bmatrix}1 & \mathrm{sh}_x\\ \mathrm{sh}_y & 1\end{bmatrix}$、$\boldsymbol{b}=\begin{bmatrix}0\\0\end{bmatrix}$ 来实现。一个一般错切的仿射变换矩阵 $\boldsymbol{M}=\begin{bmatrix}1 & \mathrm{sh}_x & 0\\ \mathrm{sh}_y & 1 & 0\end{bmatrix}$。

可以发现,仿射变换也具有传递性,也就是说任意两个仿射变换相复合的变换仍是一个仿射变换。

3 OpenCV 的内部函数对仿射变换的处理

OpenCV 中对于仿射变换的处理涉及三个函数,分别是 getAffineTransform、getRotationMatrix2D 和 warp



(a) 平移 (b) 缩放 (c) 翻转

(d) 以原点为中心旋转 (e) 以点 (x_0, y_0) 为中心旋转 (f) 沿x轴方向错切 (g) 沿y轴方向错切

图 1 典型仿射变换示例图

Affine。其中,getAffineTransform 和 getRotationMatrix2D 的功能均为返回一个 2×3 的仿射变换矩阵,warpAffine 的功能是以求得的仿射变换矩阵为参数来实现从原图像到目标图像的仿射变换。这里,将探讨 OpenCV 的内部函数如何求得一个仿射变换的变换矩阵以及如何高效地完成仿射变换。

一个仿射变换矩阵大致可以通过两种方法求得,分别如下:

(1)对于一个有着特殊几何性质的变换,可以根据该变换的具体性质直接求出其变换矩阵,就像求平移、缩放、翻转、旋转和错切等变换的变换矩阵一样。函数 getRotationMatrix2D 就是通过这一方法直接求出绕任意点旋转的仿射变换矩阵,其具体算法基本同 2.4 中所述一致。

(2)对于一个一般的仿射变换,并不知道其具体性质,或者该变换没有明显的性质,因此也就无法直接求出其仿射变换矩阵。然而,对于任何一个给定的仿射变换,如果知道原图像和目标图像上点的坐标,并将其代回式(1)中,就可以求解出实现该变换的变换矩阵。使用这种方法时只需在原图像和目标图像上分别找三个不共线的点,并利用这两组三点间的一一映射构建方程组,就足以求解出一个 2×3 的仿射变换矩阵。函数 getAffineTransform 就是利用这种方法求解出了一个一般的仿射变换矩阵。

下面对 getAffineTransform、getRotationMatrix2D 和 warpAffine 进行具体分析。

3.1　getAffineTransform

该函数出现在 imgwarp.cpp 的第 6 275 行。其形参表为:getAffineTransform(const Point2f src[], const Point2f dst[])。其中,src 为 const Point2f 类数组,表示原图像上的三组点;dst 为 const Point2f 类数组,表示目标图像上的三组点。函数的算法流程如图 2 所示。

显然,该函数并没有将原图像和目标图像上的三点坐标直接代入式(1)中并联立方程组求解。下面,描述该函数求解仿射变换矩阵的算法过程:

(1)将仿射变换矩阵 $\boldsymbol{M}=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$ 中的各元素放入一个 6×1 的待求矩阵 $\boldsymbol{X}$ 中,即 $\boldsymbol{X}=\begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{bmatrix}$。

(2)根据数据图像上的三点坐标 (x_1',y_1'),(x_2',y_2'),(x_3',y_3') 构建一个 6×1 的常数项矩阵 $\boldsymbol{B}=\begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \end{bmatrix}$。

(3)根据原图像上的三点坐标 (x_1,y_1),(x_2,y_2),(x_3,y_3) 构建一个 6×6 的系数矩阵 $\boldsymbol{A}=\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{bmatrix}$。

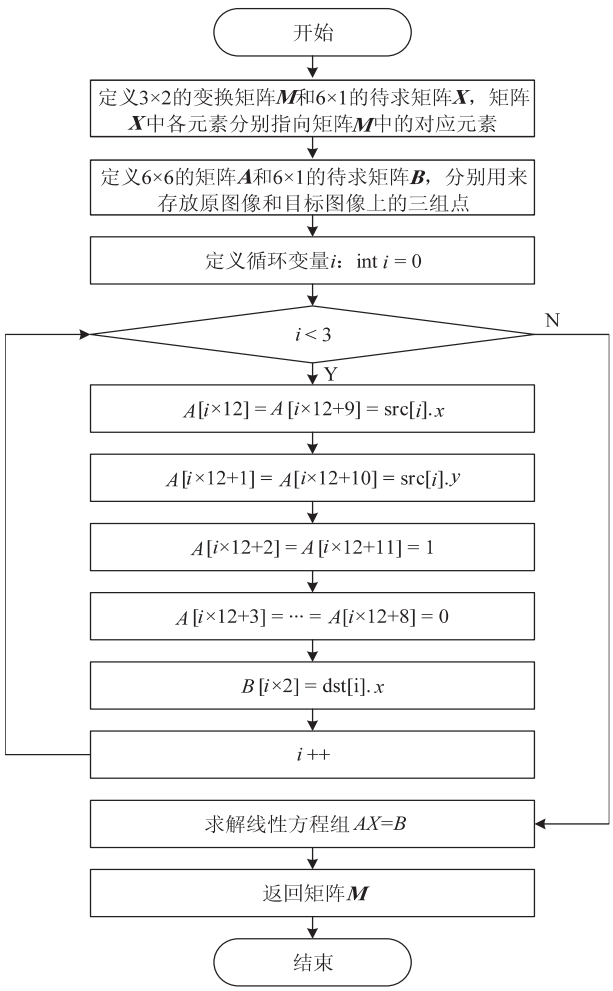


图 2　getAffineTransform 算法流程图

(4)调用 OpenCV 内部函数 solve 求解线性方程组 $\boldsymbol{AX}=\boldsymbol{B}$。这样,求解出的矩阵 $\boldsymbol{X}$ 中的各元素就是仿射变换矩阵 $\boldsymbol{M}$ 中的各元素。

3.2　getRotationMatrix2D

该函数出现在 imgwarp.cpp 的第 6 275 行。其形

参表为:getRotationMatrix2D(Point2f center, double angle, double scale)。其中,center 为 Point2f 类型,表示旋转中心点;angle 为 double 类型,表示旋转角度,其值为正数时表示逆时针旋转;scale 为 double 类型,表示缩放因子。

该函数的功能就是返回一个以 center 为旋转中心逆时针旋转 angle 角度并进行 scale 倍缩放的仿射变换矩阵,其算法流程无需多言。唯一需要注意的是,由于 OpenCV 默认的平面直角坐标系比较特殊,它选取屏幕左上角为原点,屏幕右端为 x 轴正方向,屏幕下端为 y 轴正方向,故该函数计算出的仿射变换矩阵的形式与 2.4 中有所不同,为 $\boldsymbol{M}=\begin{bmatrix} \alpha & \beta & (1-\alpha)\text{center}.x+\beta\cdot\text{center}.y \\ -\beta & \alpha & \beta\cdot\text{center}.x+(1-\alpha)\text{center}.y \end{bmatrix}$。其中,$\alpha=\text{scale}\cdot\cos(\text{angle})$;$\beta=\text{scale}\cdot\sin(\text{angle})$。

3.3 warpAffine

该函数出现在 imgwarp.cpp 的第 5 562 行。其形参表为:warpAffine(InputArray _src, OutputArray _dst, InputArray _M0, Size dsize, int flags, int borderType, const Scalar& borderValue)。其中,_src 为 InputArray 类型,表示原图像;_dst 为 OutputArray 类型,表示目标图像;_M0 为 InputArray 类型,表示 2×3 的仿射变换矩阵;dsize 为 Size 类型,表示目标图像的尺寸;flags 为 int 类型,是插值方法的标识符,可选的插值方法和开关选项包括 INTER_NEAREST(最近邻插值)、INTER_LINEAR(线性插值)、INTER_AREA(区域插值)、INTER_CUBIC(三次样条插值)、INTER_LANCZOS4(兰索斯插值)、CV_WARP_FILL_OUTLIERS(填充所有输出图像的像素,如果部分像素落在输入图像的边界外,那么它们的值设定为 fillval)、CV_WARP_INVERSE_MAP(指定仿射变换矩阵是输出图像到输入图像的反变换,因此可以直接用来做像素插值。否则,函数从仿射变换矩阵得到反变换。),其默认值为 INTER_LINEAR(线性插值);borderType 为 int 型,表示边界像素模式,其默认值为 BORDER_CONSTANT;borderValue 为 const Scalar& 类型,是在恒定边界情况下取的值,默认值为 0。函数算法流程如图 3 所示。

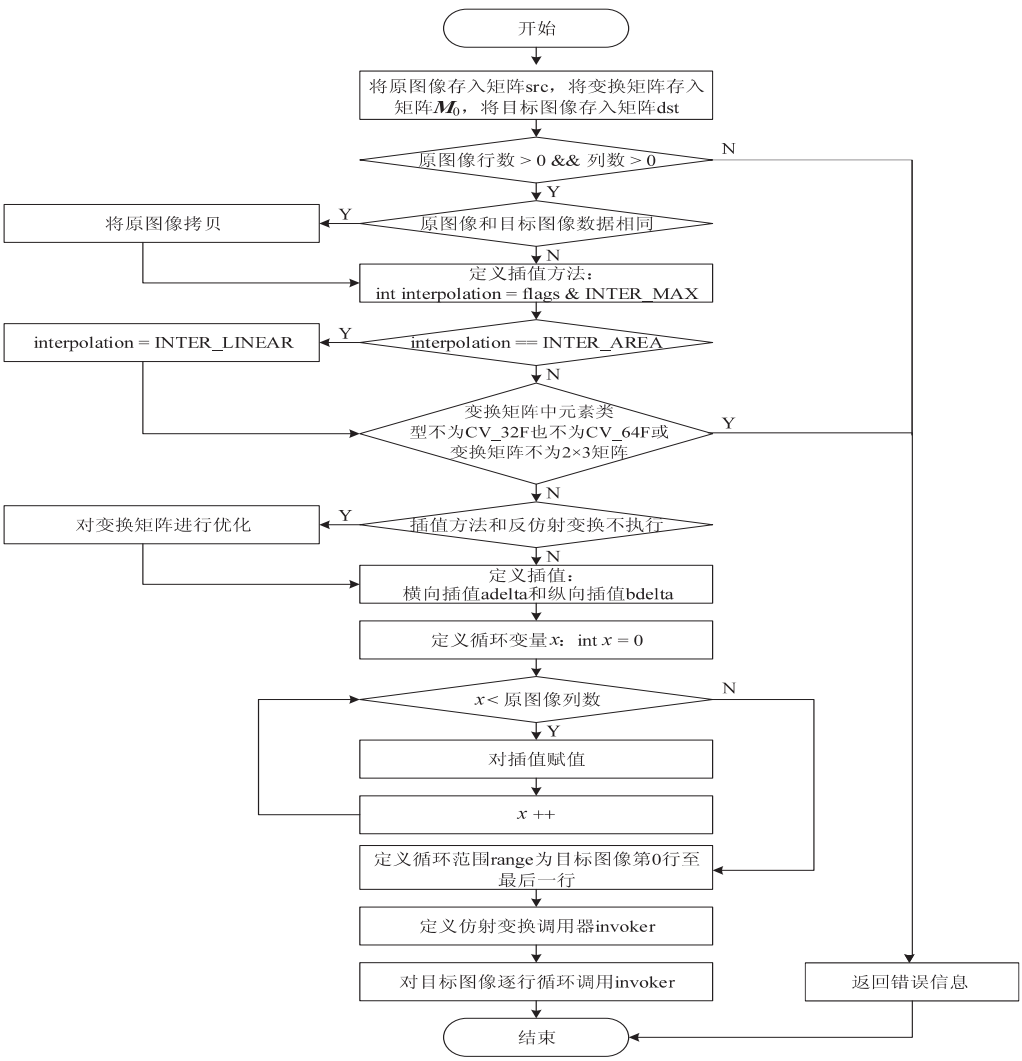


图 3 warpAffine 算法流程图

该函数的功能是应用式(3)实现从原图像到目标图像的仿射变换。在对图像上每一个点进行变换的过程中,该函数支持四种插值算法,分别是最近邻插值算法、线性插值算法、三次样条插值算法和兰索斯插值算法。当插值方法的标识符为区域插值算法时,函数默认按线性插值算法执行。

4　仿射变换的应用实例

通过调用 OpenCV 的内部函数可以很轻松地实现图像的仿射变换。下面给出一个仿射变换的实例。

首先输入一个原图像,如图 4(a)所示。使用 getAffineTransform 函数得到的变换矩阵对原图像进行扭曲(warp),如图 4(b)所示。也可以使用 getRotationMatrix2D 函数得到的变换矩阵对原图像进行旋转(Rotate),如图 4(c)所示。甚至可以在对原图像经过扭曲的基础上再进行旋转,或在对原图像进行旋转的基础上再进行扭曲,如图 4(d)所示。

(a) 原图像　(b) 扭曲

(c) 旋转　(d) 旋转+扭曲

图 4　仿射变换的实例

5　结束语

介绍了仿射变换的数学原理、算法流程以及应用实例,较为详细地对计算机处理仿射变换的方式进行了数学上的推导,并以 OpenCV 函数库为研究对象,详细解读了其中涉及仿射变换的三个函数,最后给出了用 OpenCV 内部函数处理图像仿射变换的实例。事实上,在计算机图形学的实际应用中,仿射变换的用处极为广泛,在一些需要图像进行几何变换的过程中都离不开仿射变换的使用。

参考文献:

[1] Xue Z, Shen D G, Teoh E K. An efficient fuzzy algorithm for aligning shapes under affine transformation[J]. Pattern Recognition, 2001, 34(6): 1171-1180.

[2] Lucchese L. A frequency domain technique based on energy radial projections for robust estimation of global 2D affine transformations[J]. Computer Vision and Image Understanding, 2001, 81(1): 72-116.

[3] Li X, Xu Y D, Lv Qi. Affine-transformation parameters regression for face alignment[J]. IEEE Signal Processing Letters, 2016, 23(1): 55-59.

[4] 明德烈,柳　健,田金文. 仿射变换在增强现实中的应用[J]. 系统仿真学报, 2001, 13(S1): 286-289.

[5] Luca L, Simone L, Guido M C. Estimation of two-dimensional affine transformations through polar curve matching and its application to image mosaicking and remote-sensing data registration[J]. IEEE Transactions on Image Processing, 2006, 15(10): 3008-3019.

[6] 张召悦,魏孝强,杨　晗. 基于 SIFT 和仿射变换的航空器起飞图像跟踪方法[J]. 航空计算技术, 2016, 46(1): 82-84.

[7] 李培华,肖莉娟. 基于 MeanShift 的相似性变换和仿射变换目标跟踪算法[J]. 中国图象图形学报, 2011, 16(2): 258-266.

[8] 柏　森,曹长修. 亚仿射变换的性质及其应用[J]. 计算机辅助设计与图形学学报, 2003, 15(2): 205-208.

[9] 汪文英,张冬明,张勇东,等. 利用仿射变换的快速空间关系验证[J]. 计算机辅助设计与图形学学报, 2010, 22(4): 625-631.

[10] 葛　娟,曹伟国,周　炜,等. 一种颜色仿射变换下的局部特征描述子[J]. 计算机辅助设计与图形学学报, 2013, 25(1): 26-33.

[11] 何　冰. 基于仿射变换的图像置乱改进新算法[J]. 计算机与数字工程, 2011, 39(3): 121-124.

[12] 文昌辞,王　沁,丁　华,等. 基于三维仿射变换的数字图像置乱算法[J]. 北京科技大学学报, 2012, 34(12): 1478-1482.

[13] Lay D C. Linear algebra and its applications[M]. 3rd ed. [s. l.]: Pearson Education, 2002.