图像处理算法的理论与实践

1. 图像的点运算

点运算指的是对图像中的每个像素依次进行同样的灰度变换运算，通常用于改变图像的灰度范围及分布。设r和s分别是输入图像$f(x,y)$和输出图像$g(x,y)$在任一点的灰度值，则点运算可以使用下面的公式定义：

$$S = T(r)$$

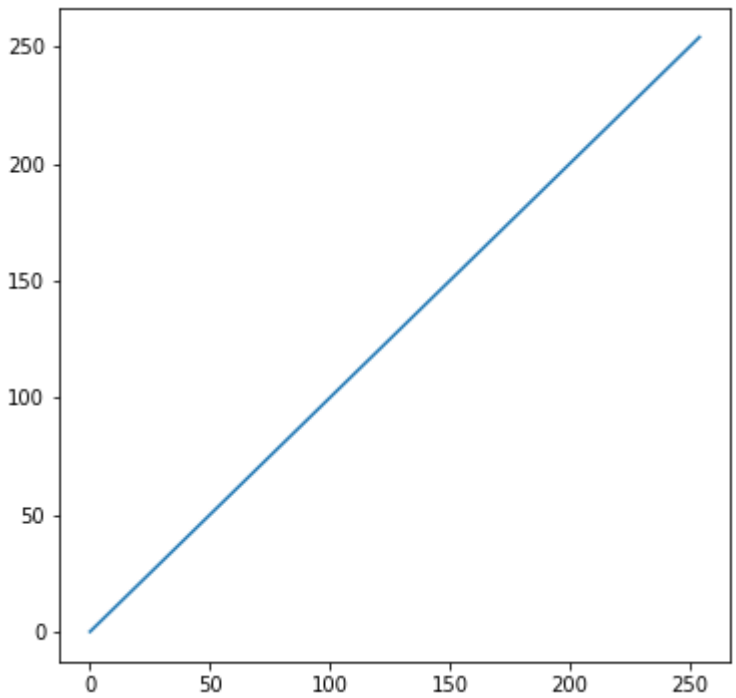
其中，T为采用的点运算算子，表示在原始图像和输出图像之间的某种灰度级映射关系。

1.1 灰度线性变换

灰度线性变换函数$f(x,y)$是一个一维线性函数：

$$D_B = f(D_A) = kD_A + b$$

其中参数k为线性函数的斜率，b为线性函数在y轴的截距，D_A表示输入图像的灰度，D_B表示输出图像的灰度。当斜率大于1时，输出图像的对比度将增大；反之减小。特殊情况下，当$k=-1, b=255$时，输出图像的灰度正好反转，这种反转处理适用于增强暗色图像中亮度较大的细节部分。



OpenCV示例程序如下：

```
#include <opencv2/opencv.hpp>
using namespace cv;
int main()
{
    Mat srcImage = imread("opencv_test.jpg");
    Mat grayImage;
    cvtColor(srcImage, grayImage, CV_BGR2GRAY);
    Mat dstImage = 0.6*grayImage - 5;

    imshow("before", grayImage);
    imshow("after", dstImage);
    waitKey();
}
```

1.2 灰度对数变换

对数变换的一般表达式为：

$$d = c * log(1 + s)$$

其中，c为尺度比例常数，s为源灰度值，d为变换后的目标灰度值。

由对数函数曲线可知，对数变换可以增强图像中较暗部分的细节，从而可以用来扩展被压缩的高值图像中的较暗元素，因此对数变换被广泛用于频谱图像的显示中。

OpenCV代码示例如下：

```
#include <opencv2/opencv.hpp>
using namespace cv;
int main()
{
    Mat srcImage = imread("test.jpg");
    Mat dstImage(srcImage.size(), CV_32FC3);
    float pixels[256];
    for (int i = 0; i < 256; i++)   pixels[i] = log(1 + i);

    for (int x=0; x<srcImage.rows; x++)
    {
        for (int y=0; y<srcImage.cols; y++)
        {
            dstImage.at<Vec3f>(x, y)[0] = pixels[srcImage.at<Vec3b>(x, y)[0]];
            dstImage.at<Vec3f>(x, y)[1] = pixels[srcImage.at<Vec3b>(x, y)[1]];
            dstImage.at<Vec3f>(x, y)[2] = pixels[srcImage.at<Vec3b>(x, y)[2]];
        }
    }
    normalize(dstImage, dstImage, 0, 255, CV_MINMAX);
    imshow("before", srcImage);
    imshow("after", dstImage);
    waitKey();
}
```

1.3 指数变换

指数变换是另一种常用的灰度非线性变换，其表达式为：

$$y = (x + exp)^b$$

其中，x与y的取值范围均为$[0, 1]$，exp为补偿洗漱，b则为伽马系数。伽马系数取值决定了输入图像和输出图像之间的灰度映射方式，即决定是增强低灰度还是增强高灰度。

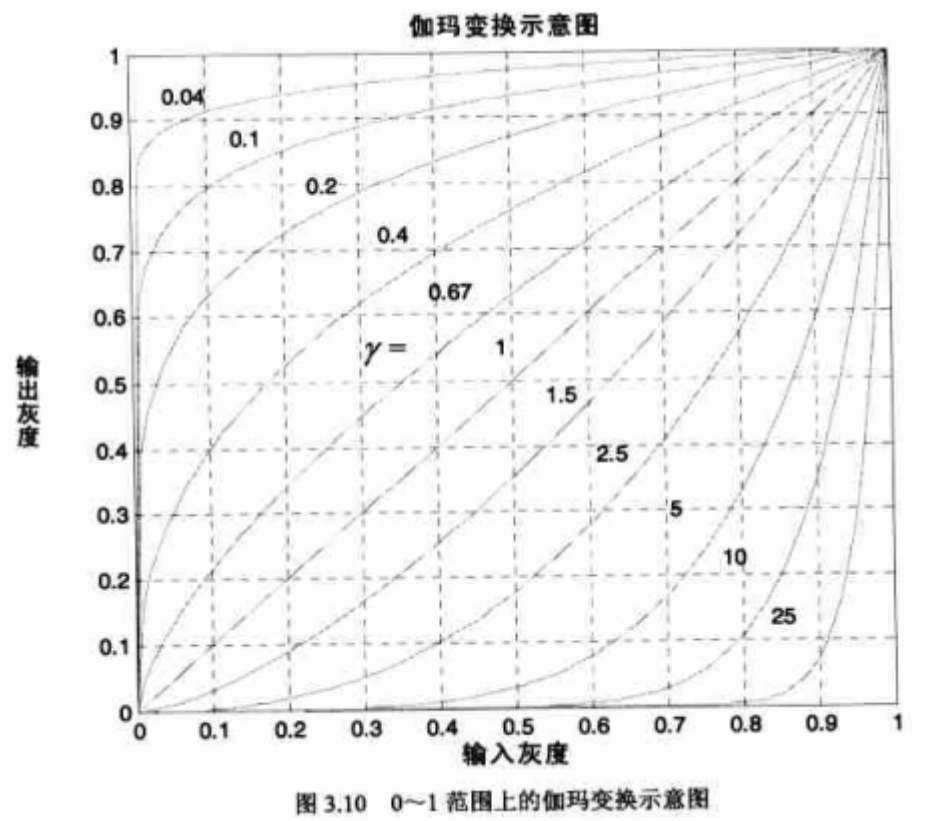


图 3.10　0～1 范围上的伽玛变换示意图

OpenCV代码示例：

```
#include <opencv2/opencv.hpp>
using namespace cv;
int main()
{
    Mat srcImage = imread("test.jpg");
    Mat grayImage;
    cvtColor(srcImage, grayImage, CV_BGR2GRAY);
    Mat dstImage(grayImage.size(), grayImage.type());

    float pixels[256];
    for (int i = 0; i < 256; i++)   pixels[i] = sqrt(float(i)/255) * 255;

    auto iter = grayImage.begin<uchar>();
    auto iter_dst = dstImage.begin<uchar>();
    while (iter != grayImage.end<uchar>())
    {
        *iter_dst = pixels[*iter];
        iter++; iter_dst++;
    }
```

```
    imshow("before", srcImage);
    imshow("after", dstImage);
    waitKey();
}
```

1.4 灰度阈值变换

灰度阈值变换可以将灰度图像装换成黑白的二值图像。用户指定一个灰度值作为阈值，如果图像中像素的灰度值小于该阈值，则将该像素灰度值设为0；否则设为1。其表达式为：

$$f(x) = \begin{cases} 0, x < T \\ 255, x \geq T \end{cases}$$

OpenCV代码示例：

```
#include <opencv2/opencv.hpp>
using namespace cv;
int main()
{
    Mat srcImage = imread("test.jpg");
    Mat grayImage;
    cvtColor(srcImage, grayImage, CV_BGR2GRAY);
    Mat dstImage(grayImage.size(), grayImage.type());

    auto iter = grayImage.begin<uchar>();
    auto iter_dst = dstImage.begin<uchar>();
    while (iter != grayImage.end<uchar>())
    {
        *iter_dst = (*iter) < 100 ? 0 : 255;  //阈值处理
        iter++; iter_dst++;
    }

    imshow("before", grayImage);
    imshow("after", dstImage);
    waitKey();
}
```

1.5 直方图均衡化

灰度直方图描述了一幅图像的灰度级统计信息，表示图像中各个灰度级出现的次数或概率。主要应用于图像分割和图像灰度变换处理中。

直方图均衡是指通过某种灰度映射使输入图像转换为在每一灰度级上都有近似相同的像素点数的输入图像。在经过均衡化处理后的图像中，像素将占有尽可能多的灰度级并且均匀分布。因此，均衡化之后的图像具有较高的对比度和动态范围。

为便于分析，我们首先考虑灰度范围为$[0, 1]$且连续的情况，此时图像的归一化直方图为概率密度函数：

$$p(x), 0 \leq x \leq 1$$

由概率密度函数的性质，有：

$$\int_{x=0}^{1} p(x) = 1$$

设转换前的概率密度函数为$p_r(r)$，转换后图像的概率密度函数为$p_s(s)$，转换函数（灰度映射关系）为$s = f(r)$。则有：

$$p_s(s) = p_r(r) \cdot \frac{\mathrm{d}r}{\mathrm{d}s}$$

如果转换后图像的直方图是均匀分布的，那么其概率密度函数$p_s(s) = 1$，则上式可以转化为：

$$p_r(r) = \frac{\mathrm{d}s}{\mathrm{d}r}$$

等式两边对r积分，可得：

$$s = \int_{u=0}^{r} p_r(u)du$$

上式被称为图像的累积分布函数，将其映射到$[0, 255]$，并进行离散化，得到转换公式：

$$D_B = f(D_A) = \frac{255}{A_0} \cdot \sum_{i=0}^{D_A} H_i$$

其中，D_B为转换后的灰度值，D_A为转换前的灰度值，H_i为第i级灰度的像素个数，A_0为图像像素总数。

OpenCV代码示例如下：

```
#include <opencv2/opencv.hpp>
using namespace cv;

int main()
{
    Mat srcImage = imread("test.jpg");
    cvtColor(srcImage, srcImage, CV_BGR2GRAY);
    Mat dstImage;
    equalizeHist(srcImage, dstImage);   //均衡化函数
    imshow("before", srcImage);
    imshow("after", dstImage);
    waitKey();
}
```

2. 图像的几何变换

图像几何变换（或空间变换），是将图像中的坐标位置映射到另一幅图像中的新坐标位置。其不改变图像的像素值，只是将图像平面上进行像素的重新排列。常见的几何变换有平移、旋转和镜像。由于图像像素可能会被映射到非整数坐标上，所以有事需要使用灰度差值算法进行处理。