直方图均衡化(Histogram Equalization) 又称直方图平坦化,实质上是对图像进行非线性拉伸,重新分配图像象元值,使一定灰度范围内象元值的数量大致相等。这样,原来直方图中间的峰顶部分对比度得到增强,而两侧的谷底部分对比度降低,输出图像的直方图是一个较平的分段直方图:如果输出数据分段值较小的话,会产生粗略分类的视觉效果。

直方图是表示数字图像中每一灰度出现频率的统计关系。直方图能给出图像灰度范围、每个灰度的频度和灰度的分布、整幅图像的平均明暗和对比度等概貌性描述。灰度直方图是灰度级的函数, 反映的是图像中具有该灰度级像素的个数, 其横坐标是灰度级r, 纵坐标是该灰度级出现的频率(即像素的个数) pr(r) , 整个坐标系描述的是图像灰度级的分布情况, 由此可以看出图像的灰度分布特性, 即若大部分像素集中在低灰度区域, 图像呈现暗的特性; 若像素集中在高灰度区域, 图像呈现亮的特性。

图1所示就是直方图均衡化, 即将随机分布的图像直方图修改成均匀分布的直方图。基本思想是对原始图像的像素灰度做某种映射变换, 使变换后图像灰度的概率密度呈均匀分布。这就意味着图像灰度的动态范围得到了增加, 提高了图像的对比度。

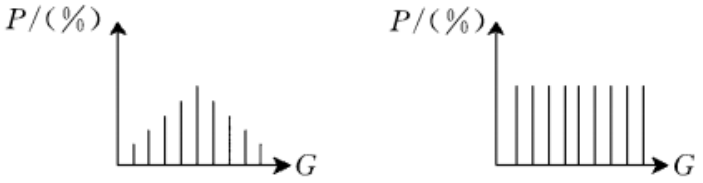

图1 直方图均衡化

通过这种技术可以清晰地在直方图上看到图像亮度的分布情况，并可按照需要对图像亮度调整。另外,这种方法是可逆的，如果已知均衡化函数，就可以恢复原始直方图。

设变量r 代表图像中像素灰度级。对灰度级进行归一化处理，则0≤r≤1，其中r= 0表示黑，r= 1表示白。对于一幅给定的图像来说，每个像素值在[0,1] 的灰度级是随机的。用概率密度函数 $p_r(r)$ 来表示图像灰度级的分布。

为了有利于数字图像处理，引入离散形式。在离散形式下，用 r^k 代表离散灰度级，用 $P_r(r^k)$ 代表 $p_r(r)$ ，并且下式成立: $P_r(r^k)=\frac{nk}{n}$

其中，$0\leqslant r^k\leqslant 1$，k=0，1，2，…，n-1。式中 n^k 为图像中出现 r^k 这种灰度的像素数，n是图像中的像素总数，而 $\frac{nk}{n}$ 就是概率论中的频数。图像进行直方图均衡化的函数表达式为:

$$S_i=T(r_i)=\sum_{i=0}^{k-1}\frac{n_i}{n}$$

式中，k为灰度级数。相应的反变换为:

$$r^i=T^{-1}(S_i)$$

```
#define HDIM 256
#define SRC 0
#define DST 1
intmain(intargc, char** argv)
{
IplImage *src = 0, *dst = 0;
intn[] = {HDIM,HDIM,HDIM};
intr[256] = {0}, g[256] = {0}, b[256] = {0};
if(argc!=2 || (src = cvLoadImage(argv[1],3))== NULL)   return-1;
cvNamedWindow("source",1);
cvNamedWindow("result",1);
```

```
intwidth = src->width;
intheight = src->height;
intsum = width * height;       //图像中的像素点综合
inti,j;
//分别统计直方图的RGB分布
for(i=0; i
for(j=0; j
{
b[((uchar*)(src->imageData+i*src->width))[j*src->nChannels+0]]++;
g[((uchar*)(src->imageData+i*src->width))[j*src->nChannels+1]]++;
r[((uchar*)(src->imageData+i*src->width))[j*src->nChannels+2]]++;
}
////构建直方图的累计分布方程，用于对直方图进行均衡化
doubleval[3] = {0};
for(i=0; i
{
val[0] += b[i];
val[1] += g[i];
val[2] += r[i];
b[i] = val[0]*255/sum;
g[i] = val[1]*255/sum;
r[i] = val[2]*255/sum;
}
dst = cvCreateImage(cvSize(width,height),8,3);
//归一化直方图
for(i=0; i
for(j=0; j
{
((uchar*)(dst->imageData+i*dst->widthStep))[j*dst->nChannels+0]=b[((uchar*)(src-
>imageData+i*src->widthStep))[j*src->nChannels+0]];
((uchar*)(dst->imageData+i*dst->widthStep))[j*dst->nChannels+1]=g[((uchar*)(src-
>imageData+i*src->widthStep))[j*src->nChannels+1]];
((uchar*)(dst->imageData+i*dst->widthStep))[j*dst->nChannels+2]=r[((uchar*)(src-
>imageData+i*src->widthStep))[j*src->nChannels+2]];
}
cvShowImage("source",src);
cvShowImage("result",dst);
cvSaveImage("out.jpg",dst);
cvWaitKey(0);
cvDestroyWindow("source");
cvDestroyWindow("result");
cvReleaseImage(&src);
cvReleaseImage(&dst);
cvReleaseHist(&hist);
return0;
}
```

4.2结果展示