实验3 π值可视计算

# 实验目的

学习使用函数

了解可视计算

初步了解OpenCV库的使用

# 实验内容

在实验2中，我们通过统计落入正方形内切圆和正方形中的随机点的比值，来求解pi的近似值。本实验中，我们将把随机点在图像中显示，让求π的过程可视化。要求：

1. 显示一幅边长为R的正方形图像（设置为0，全黑色），利用给出的画圆函数，显示出此正方形边长为R/2的内切圆（设置为255，白色），圆心位于正方形中心；
2. 通过rand（）函数产生随机数，将产生的随机点调用DrawPixel函数写入图像，并在正方形相应区域显示出来，赋值为255（白色），且每个点的亮度随时间（新的随机点的产生）减弱至0。
3. 用合适的精度求得π值。每轮产生一定数量的点（数量通常要大一些），每次通过前n轮的数据求π值，与前面n-1轮累计数据算得的π比较，当误差小于一定数值时终止，得到最终π值，否则继续下一轮计算。每轮算得的π值都在控制台输出。
4. 完成图像所有像素置为某像素值、在图像中画圆、对图像中非零像素值减去某值等函数。

注意：

1. 在本实验和今后的几个实验中，我们会接触数字图像。我们对图像并不陌生，大家用手机拍摄的照片就是一幅数字图像。图像是由一行行、一列列的像素组成的。为简单起见，我们实验中使用的图像，每个像素只用一个数值表示其亮度，称为像素值，所以图像没有色彩信息。
2. 像素值（针对没有色彩的图像而言也常称为灰度值）在本实验和后面几个实验中以8bit无符号整数存取（unsigned char类型）。这也是大部分图像文件采用的存取像素信息的数据类型。有些高级的图像格式选择使用更高精度的数据类型，这里暂不考虑。
3. 我们使用的图像的行数和列数是从零开始计数的。比如图像有100行，那么行数是从第0行到第99行，而不是从第1行到第100行。在对图像所有像素进行遍历时需要注意。
4. 使用系统库的time函数产生srand函数所需要的随机数种子：srand(time(0))
5. 与本大纲同时发布的还有一个VC6.0的工程，里面已经配置好了OpenCV库，请在此基础上进行代码的编写。你自己创建的工程需要另外配置该库。

#include <ctime> //调用系统时间库，作为随机数种子

#include "cv.h" // OpenCV库的头文件

#include "highgui.h" // OpenCV库的头文件，用来读取图像文件

using namespace std;

//获取图像中某个点的像素值，在本实验中直接调用该函数，无需改动

unsigned char GetPixel(IplImage \*img, int row, int col)

{

unsigned char \*img\_data = (unsigned char \*)(img->imageData);

int width\_step = img->widthStep;

return img\_data[width\_step \* row + col];

}

//将图像中某个点的像素值赋值，在本实验中直接调用，无需改动

void DrawPixel(IplImage \*img, int row, int col, unsigned char v)

{

unsigned char \*img\_data = (unsigned char \*)(img->imageData);

int width\_step = img->widthStep;

img\_data[width\_step \* row + col] = v;

}

// 将图像所有像素的像素值赋值为参数v的值，请完成相应的代码。height和width是需要传入的图像的高度（行）和宽度（列），下同。

void DrawAll(IplImage \*img, int height, int width, unsigned char v)

{

// 在该函数内利用循环语句，并调用DrawPixel函数，将图像所有的像素都置为参数v的值

}

// 在图像中绘制圆的函数。圆心为(row,col)，半径为radius, 圆周上的像素值为v，请完成相应的代码

void DrawCircle(IplImage \*img, int height, int width, int row, int col, int radius, unsigned char v)

{

// 遍历图像所有的像素点的坐标，比较其到圆心的距离是否“等于”半径。

// 注意，虽然像素点的坐标是整数，但计算的距离应该设置为浮点型。这时距离和半径相比，应该使用浮点型数比较的语句。

// 利用课上所讲的浮点数的比较方法，设定一个“合适”的阈值，在图像中画出完整的圆周

// 调用DrawPixel函数对相应的像素点进行赋值

}

// 设计函数并完成相应的代码，将图像中非零的像素值减去某个数值，从而在产生随机点“渐渐消失”效果时调用

// 提示：调用GetPixel函数获取某像素点的像素值

int main(int argc, char\* argv[])

{

// 下面的函数为你创建一个OpenCV的图像，图像的宽度（width）和高度（height）由你设定。

// 本实验中的图像为正方形，所以应该设置为相同的数值，也即内切圆的直径

int width = ???;

int height = ???;

IplImage\* img = cvCreateImage(cvSize(width, height), IPL\_DEPTH\_8U, 1); // IplImage是OpenCV中存储图像的结构体

// 调用DrawAll函数，将图像所有像素的像素值都赋值为0（全黑）

// 调用DrawCircle函数，用白色（255）画内切圆，圆心（x,y）半径r

cvNamedWindow("Image", CV\_WINDOW\_AUTOSIZE);*//创建窗口*

cvShowImage("Image", img);*//显示图像*

cvWaitKey(0); *//等待按键，开始采样随机点*

srand(time(0)); *// 设置随机数种子*

// 开始你的求pi循环

// 在你的循环体内

{

// 调用你设计的函数，使图像中所有非零的像素值减去某个固定的常数，以产生动态的渐变效果，如果图像尺寸较小，则减去的数值应该适当大一些

// 产生一个随机点

// 根据点的坐标找到对应图像中的像素点，将该像素值赋值为255。

// 重新绘制内切圆

cvShowImage("Image", img);*//显示图像*

cvWaitKey(50); *//延时50毫秒*

// 同时在控制台输出当前产生的点的个数、落入圆内的点的个数、当前轮次计算得到的pi值

}

cvWaitKey(0); *//等待按键，结束程序*

cvDestroyWindow("Image");*//销毁窗口*

cvReleaseImage(&img); *//释放图像*

return 0;

}

# 实验要求

完成上述代码，利用图像显示动态的求解过程。