```
/*********************手动实现，通过 HOG (Histogram-of-Oriented-Gradients)比较图像相似度************************/
int compareImages()
{

//读入图像
cv::Mat refMat = imread("../testImages\\hogTemplate.jpg",0);
cv::Mat plMat = imread("../testImages\\img1.jpg", 0);
cv::Mat bgMat = imread("../testImages\\img2.jpg", 0);

if (refMat.empty() || plMat.empty() || bgMat.empty()) {
std::cout << "failed to read image!:" << std::endl;
return -1;
}

//参数设置
int nAngle = 8;
int blockSize = 16;
int nX = refMat.cols / blockSize;
int nY = refMat.rows / blockSize;

int bins = nX*nY*nAngle;


float * ref_hist = new float[bins];
memset(ref_hist, 0, sizeof(float)*bins);
float * pl_hist = new float[bins];
memset(pl_hist, 0, sizeof(float)*bins);
float * bg_hist = new float[bins];
memset(bg_hist, 0, sizeof(float)*bins);

int reCode = 0;
//计算三张输入图片的HOG
reCode = calcHOG(refMat, ref_hist, nAngle, blockSize);
reCode = calcHOG(plMat, pl_hist, nAngle, blockSize);
reCode = calcHOG(bgMat, bg_hist, nAngle, blockSize);

if (reCode != 0) {
return -1;
}

//计算直方图距离
float dis1 = normL2(ref_hist, pl_hist, bins);
float dis2 = normL2(ref_hist,bg_hist , bins);

std::cout << "distance between reference and img1:" << dis1 << std::endl;
std::cout << "distance between reference and img2:" << dis2 << std::endl;

(dis1<=dis2) ? (std::cout << "img1 is similar" << std::endl) : (std::cout << "img2 is similar"<<std::endl);


imshow("ref",refMat);
imshow("img1",plMat);
imshow("img2",bgMat);

waitKey(0);

delete[] ref_hist;
delete[] pl_hist;
delete[] bg_hist;
destroyAllWindows();

return 0;
}

//手动实现 HOG (Histogram-of-Oriented-Gradients)
int calcHOG(cv::Mat src, float * hist,int nAngle,int cellSize)
{

if (cellSize> src.cols || cellSize> src.rows) {
return -1;
}

//参数设置
int nX = src.cols / cellSize;
int nY = src.rows / cellSize;

int binAngle = 360 / nAngle;

//计算梯度及角度
Mat gx, gy;
Mat mag, angle;
Sobel(src, gx, CV_32F, 1, 0, 1);
Sobel(src, gy, CV_32F, 0, 1, 1);
// x方向梯度，y方向梯度，梯度，角度，决定输出弧度or角度
// 默认是弧度radians，通过最后一个参数可以选择角度degrees.

/*************************************************
cartToPolar（） 函数模型：
cartToPolar(InputArray x, InputArray y,
                              OutputArray magnitude, OutputArray angle,
                              bool angleInDegrees = false);
参数介绍：
. InputArray x/InputArray y：输入梯度x/y
.  OutputArray magnitude：输出幅值
. OutputArray angle：输出角度

*************************************************/
cartToPolar(gx, gy, mag, angle, true);

cv::Rect roi;
roi.x = 0;
roi.y = 0;
roi.width = cellSize;
roi.height = cellSize;

for (int i = 0; i < nY; i++) {
for (int j = 0; j < nX; j++) {

cv::Mat roiMat;
cv::Mat roiMag;
cv::Mat roiAgl;

roi.x = j*cellSize;
roi.y = i*cellSize;

//赋值图像
roiMat = src(roi);
roiMag = mag(roi);
roiAgl = angle(roi);

//当前cell第一个元素在数组中的位置
int head = (i*nX + j)*nAngle;

for (int n = 0; n < roiMat.rows; n++) {
for (int m = 0; m < roiMat.cols; m++) {
//计算角度在哪个bin，通过int自动取整实现
int pos =(int)(roiAgl.at<float>(n, m) / binAngle);
//以像素点的值为权重
hist[head+pos] += roiMag.at<float>(n, m);
}
}

}
}

return 0;
}

float normL2(float * Hist1, float * Hist2, int size)
{
float sum = 0;
for (int i = 0; i < size; i++) {
sum += (Hist1[i] - Hist2[i])*(Hist1[i] - Hist2[i]);
}
sum = sqrt(sum);
return sum;
}
```