机器视觉课程设计

报告

**姓名：\_黄继凡\_\_\_何景淞\_\_\_\_\_\_\_\_\_**

**学号：\_200320224\_200320222\_\_\_\_\_**

**学院：\_\_\_\_机电工程与自动化\_\_\_\_\_**

**专业：\_\_\_\_\_\_自动化\_\_\_\_\_\_\_\_\_\_\_\_\_**

目录

[一、 概述 5](#_Toc43282332)

[二、 课程设计任务及要求 5](#_Toc43282333)

[三、 算法设计 5](#_Toc43282334)

[四、 实验及数据分析 5](#_Toc43282335)

[五、 结论 6](#_Toc43282336)

[六、 收获、体会和建议 6](#_Toc43282337)

[七、 参考文献 6](#_Toc43282338)

1. **概述**

该课题为尺寸测量项目，主要目标有计算给出图例中的物体的尺寸，进行1D测量，并达到亚像素精度且计算算法响应耗时。

1. **课程设计任务及要求**

设计算法实现对给出图例中，平行排列的黑色条状目标进行尺寸测量，包括条状目标的长度、宽度以及之间的距离；图例存在不同程度的噪声，需要一定的抗干扰能力。

1. **算法设计**

**1.噪声处理**

考虑到图像包含有噪声，需要首先进行预处理以部分地减轻噪声影响，本算法中将图像转为了灰度图，并进行了滤波处理和闭合运算，突出了黑色目标物体的特征。

|  |
| --- |
| // 根据输入的原始图像，创建对应的灰度图      Mat gray\_image;      cvtColor(original\_image, gray\_image, COLOR\_BGR2GRAY);      // 滤波处理      GaussianBlur(gray\_image, gray\_image, Size(3, 3), 0, 0);      // 二值化，这一步比较离谱      // threshold(gray\_image, gray\_image, 0, 255, THRESH\_BINARY | THRESH\_OTSU);      // 闭合运算      Mat element = getStructuringElement(MORPH\_RECT, Size(5, 5));      morphologyEx(gray\_image, gray\_image, MORPH\_CLOSE, element);      // 开运算      morphologyEx(gray\_image, gray\_image, MORPH\_OPEN, element); |

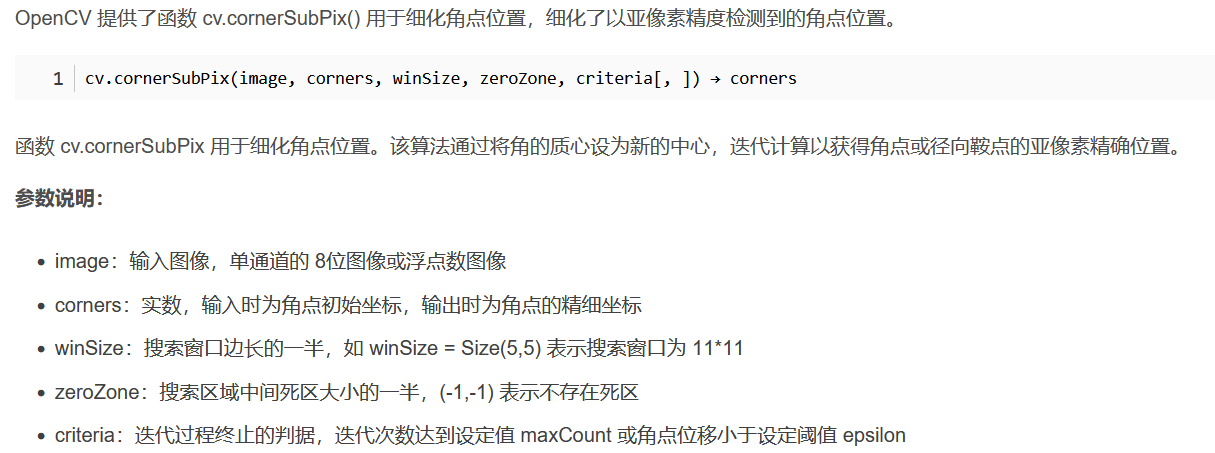
**2.边缘检测**

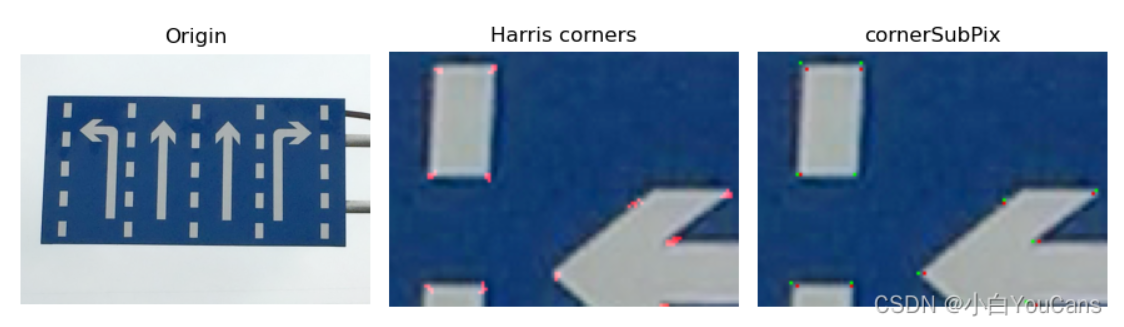
之后当用户选定了测量的线段区域后，在垂直线段方向上进行等间隔采样并进行中值滤波，用以平滑线段上的测量点的灰度值，提高抗干扰能力；之后在平滑后的灰度值中寻找边缘点，通过二阶差分的方式进行计算判断。

|  |
| --- |
| // 寻找边缘点，即灰度值发生跳变的点      // 采用二阶差分的方法      vector<double> gray\_diff(n);      for (int i = 0; i < n; i++) {          // 边界检查          if (i == 0) {              gray\_diff[i] = gray\_smooth[i + 1] - gray\_smooth[i];              continue;          }          if (i == n - 1) {              gray\_diff[i] = gray\_smooth[i] - gray\_smooth[i - 1];              continue;          }          gray\_diff[i] =              gray\_smooth[i + 1] - 2 \* gray\_smooth[i] + gray\_smooth[i - 1];          // [DEBUG]          cout << "gray\_diff[" << i << "]:" << gray\_diff[i] << endl;      }      // 寻找边缘点      vector<Point2f> edge\_points;      vector<int> index\_edge\_points;      for (int i = 0; i < n; i++) {          // 边界检查          if (i == 0 || i == n - 1) {              continue;          }          // 判断是否为边缘点          // 大于规定阈值，且与前一个点的符号不同          if (abs(gray\_diff[i]) < 30) {              continue;          }          // 若前一个点为边缘点，则跳过          if (index\_edge\_points.size() > 0 && i - index\_edge\_points.back() < 5) {              continue;          }          if (gray\_diff[i] \* gray\_diff[i - 1] < 0 ||              gray\_diff[i] \* gray\_diff[i + 1] < 0) {              // 计算边缘点的坐标              double x\_edge = x[i];              double y\_edge = k \* x\_edge + b;              edge\_points.push\_back(Point2f(x\_edge, y\_edge));              index\_edge\_points.push\_back(i);              // [DEBUG]              cout << "index of edge point:" << i << endl;          }      } |

**3.亚像素精度寻点**

随后对边缘点进行亚像素精度化，可以考虑将原先的边缘点作为搜索初始点，向四周方向进行搜索并计算图像梯度，以更新边缘点，使得边缘点更加精确。（后续注意到此函数用于角点优化，对于直线边缘点的优化并不太合适，但是如果是不规则多边形，在寻找角点间距离时，将会比较准确）

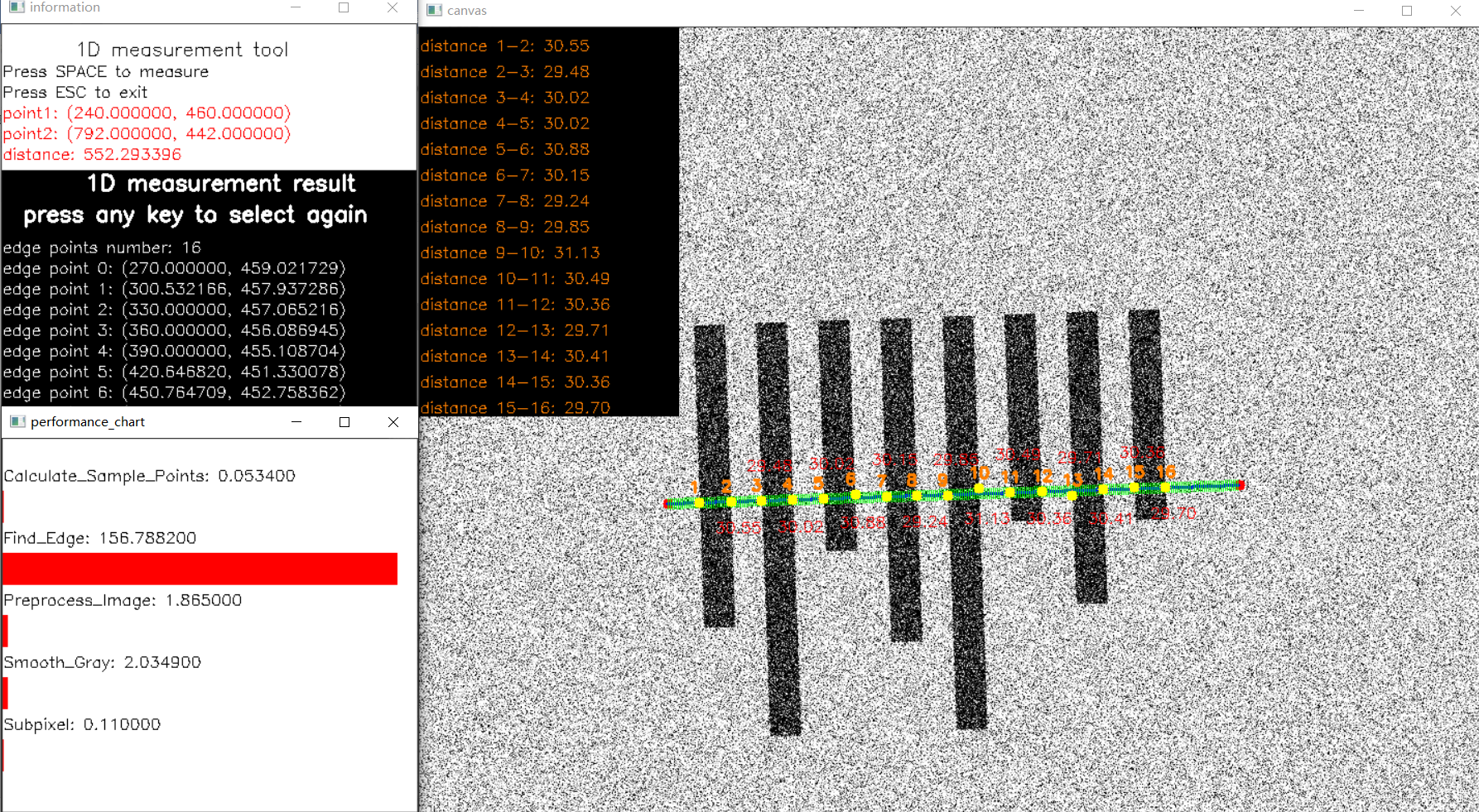




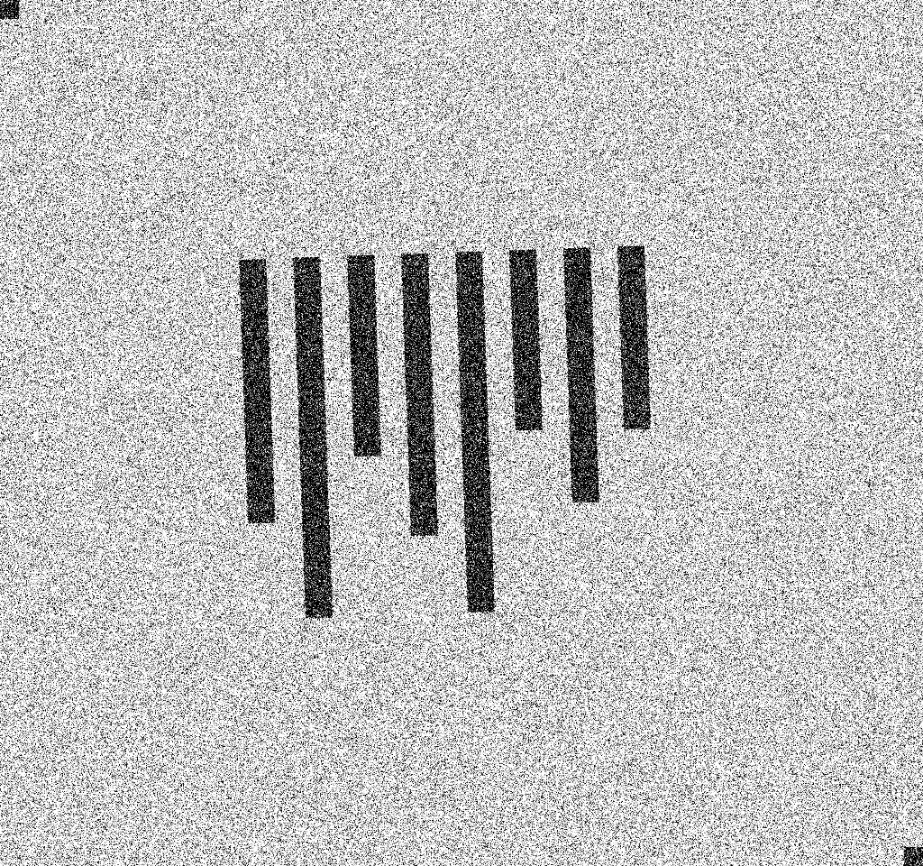
|  |
| --- |
| // 若边缘点个数小于2，则无法进行亚像素精确化      if (edge\_points.size() < 2) {          cout << "edge points number is less than 2" << endl;          return;      } else {          // 亚像素精确化          cornerSubPix(              gray\_image, edge\_points, Size(5, 5), Size(-1, -1),              TermCriteria(TermCriteria::EPS + TermCriteria::COUNT, 40, 0.001));      } |

1. **实验及数据分析**

对包含噪声图像的测量和耗时分析



使用原始图像如下



注意到在包含较严重噪声时，算法依然可以比较准确地识别出目标物体的边缘并计算距离；算法主要耗时在边缘搜索上，大概为130.7ms（边缘搜索耗时与选定边缘长度近乎成正比，因为边缘越长，待搜索的点就越多）

1. **结论**

使用手写实现的卡尺功能类在图像含有较严重的噪声时，依然可以比较准确地识别目标边缘；通过修改边缘差分阈值可以对对比度较弱的图像识别边缘，能够比较稳定的计算图像尺寸信息。

1. **收获、体会和建议**

本次课程设计中，利用1D测量计数对目标物体尺寸信息进行了测量；加深了对于1D测量技术的理解，对于图像边缘的抗干扰处理有了一定认识；为了提高算法处理的灵活性，设计了用户操作的流程，提高了程序的互动性。

经过课程设计的锻炼，对于C++在机器视觉项目中的应用有了进一步的体验和感受，意识到了机器视觉的工程实现的重要性。本次课程设计实现了一个卡尺的功能类，对于边缘测量有较好的效果。

1. **参考文献**

[W FORSTNER. A fast operator for detection and precise location of distincs points, corners and center of circular features](https://cseweb.ucsd.edu/classes/sp02/cse252/foerstner/foerstner.pdf)

https://docs.opencv.org/4.x/dd/d1a/group\_\_imgproc\_\_feature.html#ga354e0d7c86d0d9da75de9b9701a9a87e