刘忠瑞, 赵廷玉. 基于 OpenCV 的 4K HDMI 相机上的自动寻边体系的研究[J]. 智能计算机与应用,2024,14(11):156-162.
DOI:10.20169/j.issn.2095-2163.241124

基于 OpenCV 的 4K HDMI 相机上的自动寻边体系的研究

刘忠瑞, 赵廷玉
(浙江理工大学 理学院, 杭州 310018)

摘 要: 目前诸如 HDMI 显微相机等嵌入式设备主要采用肉眼观察+鼠标定位的方式选取物体边缘,存在精度差、耗时久等问题。本文提出一种基于 OpenCV 视觉库的实时显微图像自动寻边体系。首先,从后台采集实时图像,分离出灰度信息并保存于内存缓冲区;然后,截取鼠标点击处为中心的 ROI 区域,并基于直方图梯度幅度自动确定 Canny 边缘检测双阈值获取 ROI 区域边缘信息;最终,根据不同边缘类型计算鼠标点击处最近的边缘。实验结果表明:相较于肉眼观察+鼠标定位的传统边缘测量方式,本文提出的鼠标定位+自动寻边方式平均耗时缩短约 36%,精度从 69.86%提升至 98.59%,具有精度高、效率高和延时低的特点,在显微测量等领域具有实际应用价值。

关键词: OpenCV ; 高清多媒体接口; 工业相机; 自动寻边; 边缘检测
中图分类号: TP311.1 **文献标志码**: A **文章编号**: 2095-2163(2024)11-0156-07

Automatic edge detecting in 4K HDMI cameras based on OpenCV

LIU Zhongrui, ZHAO Tingyu
(School of Science, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: The method of naked-eye observation-mouse positioning is the main way to detect the edge of objects in the embedded devices represented by HDMI microscope cameras. This conventional edge-detection method inevitably has some problems such as poor precision and long time consuming because it is completely dependent on manual operation. This paper proposes a real-time automatic edge detection system for microscopic images based on the OpenCV vision library. Firstly, a gray-scale image, separated from a real-time mocroscopic image, is stored in the memory buffer. Next, the square area of the gray-scale image centered on the mouse click is truncated as the region of interest (ROI). Then, dual thresholds are determined automatically using Canny edge detection of ROI based on histogram of gradient. Finally, the thresholds are applied to return the edge closest to the mouse-click point according to the different types of edges. The experimental results show that the edge detection method proposed in this paper has the advantages of high precision and short time consuming. Specifically, the average time is shortened by about 36%, while the accuracy is increased from 69.86% to 98.59% compared to the conventional method. It is believed that the automatic edge detection method has a wide application prospect in the field of microscopic detection.

Key words: OpenCV; high definition multimedia interface; industrial camera; automatic edge finding; edge detection

0 引 言

随着工业制造业的高速发展,许多领域对于超高清的视频需求也在增加。高清 4K 分辨率的显微相机已成为工业高清相机的主流产品[1],其中 HDMI 相机可以将传统光学镜头下观察的图像在高清分辨率的显示器上显示出来,可以帮助专业人员更加实时高效地进行测量分析、观察记录等。

同时,高分辨的图像显示使显微图像高精度测量成为现实,HDMI 相机脱离电脑进行各类独立、实时的数据标注和检测也愈发普遍,例如在基于 Hi3516A 的 4K 显微相机上,拥有测点的位置、测角、测线段、测矩形、测圆形等丰富测量功能[2],这些测量功能的实现离不开图像边缘的确定。但是现有 HDMI 相机上的边缘确定还依赖于人眼的观察和鼠标的点击,存在随机性误差大、精度无法保证等缺

作者简介: 刘忠瑞(1998—),男,硕士研究生,主要研究方向:图像处理,嵌入式技术。
通信作者: 赵廷玉(1982—),女,博士,副教授,主要研究方向:光机算一体化设计,成像光学系统的 CAD 设计,光学仪器的机械结构设计,数字图像处理算法研究。Email:zhaotingyu@zstu.edu.cn。

收稿日期: 2023-06-18

点。为了提高测量精度，人们往往通过放大图像来选取边缘点，但这并不能从根本上解决精度问题，却大大降低了测量的效率。

针对此问题，本文提出一套高效完整的图像采集流程，并对所采集到的图像进行灰度化处理、图像去噪、边缘检测、确定最近边缘点优化测量方案等一系列操作。边缘检测是该方法体系的重要环节，Canny[3] 提出的 Canny 边缘检测算子，具有非常好的检测效果，并凭借其严格的边缘检测评价标准得到了广泛应用[4]。传统 Canny 算子需要预先人为设定高低阈值，并且十分依赖先验经验，需要多次试验才能找到合适的阈值，且在显微镜下观测的图像还要受到曝光等影响，其高低阈值的比例不能设为固定值。而基于梯度幅度直方图和类内方差最小化自适应的高低阈值的确定方法[5] 不需要人为的设定阈值，可以根据不同的图片，自动地求取适合自身的阈值。

综合上述情况，本文提出一套基于 OpenCV 库的完整自动寻边方法体系，通过对相机获取的实时图像进行采集预处理，对鼠标点击感兴趣区域范围进行自适应阈值的 Canny 边缘检测，得到最近边缘点的精准定位，并使鼠标移动到该点后再进行下一步测量，大大提高了检测精度和效率。

1　方法总体设计

本文以基于海思平台 Hi3516A 芯片的高性能 4K显微相机为例，在软件开发方面以嵌入式Linux 为操作系统，以海思 MPP（Media Process Platform）为媒体处理开发平台，且具有高清图像采集、实时显示、编码存储等功能；采用嵌入式 Qt 开发的一套友好的图形用户界面，支持鼠标操作，可以完全脱离计算机，连接 4K 分辨率的高清显示屏使用。

该方法的整体架构如图 1 所示。由图 1 可知，主要分为图像采集模块、图像边缘检测模块以及自动寻边模块三个部分。

图 1　自动寻边方法体系总体设计框图

Fig. 1　Block diagram of the overall design of the automatic edge detecting methodology

1.1　图像采集模块

Hi3516A 内部视频处理流程如图 2 所示。此款相机利用 Hi3516A 平台提供的视频处理 VPSS（Video Process Sub-System）模块可以处理 2 类数据。一类为来自传感器（sensor）的实时采集后，经由视频输入 VI（Video Input）模块捕获的视频图像数据；另一类为来自外部存储设备（如 SD 卡）、并经视频解码 VDEC（Video Decode）模块解析后的图像数据。视频处理 VPSS 模块既可以将数据经视频编码 VENC（Video Encode）来实现图像编码功能[6]，即将图像数据或视频编码成不同格式；也可以通过视频输出 VO（Video Output）将数据在显示设备上进行展示。

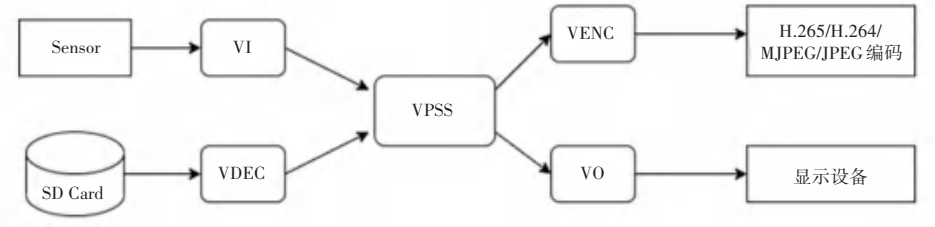

图 2　Hi3516A 内部视频处理流程

Fig. 2　Hi3516A internal video processing processes

本文图像数据格式以 YUV420SP 格式传输为例，其色度信号分辨率是亮度信号分辨率的 1/4，即对于一张 width×height 的彩色图片，数据大小为 width×height×1.5，有 width×height×1 的空间存储的是每个像素点的灰度值信息、即 Y 分量数据，有 width×height×0.5 的空间存储的是每个点的色度值信息[7]。传统边缘检测的一个步骤是将彩色图像灰度化，本文在 YUV420SP 格式视频中分离出 Y 分量数据、即只保留亮度信息，并将 Y 分量数据转化为灰度图像以减少图像数据的计算量。

由于嵌入式设备性能普遍不高，以及 HDMI 的高分辨率图像数据量较大，通过机器视觉方式对观测的显微图像进行边缘检测。首先要采集获取整幅图像信息，如果进行寻边测量时才开始采集会造成主线程卡顿，且速度较慢，不利于实时测量。实时图像采集流程如图 3 所示，从高效率、低延时角度出

发,本文在后台开辟一个线程专用于实时帧的图像解码和采集,在初始化解码参数后,每隔 1 s 抓取一幅实时图像,并将分离得到 Y 分量数据的灰度图像存于内存缓冲区中,内存缓冲区只保存一张灰度图像,新图像会覆盖旧图像。需要自动寻边时,将直接从内存中读取灰度图像,可以明显提高取图速度,避免卡顿。

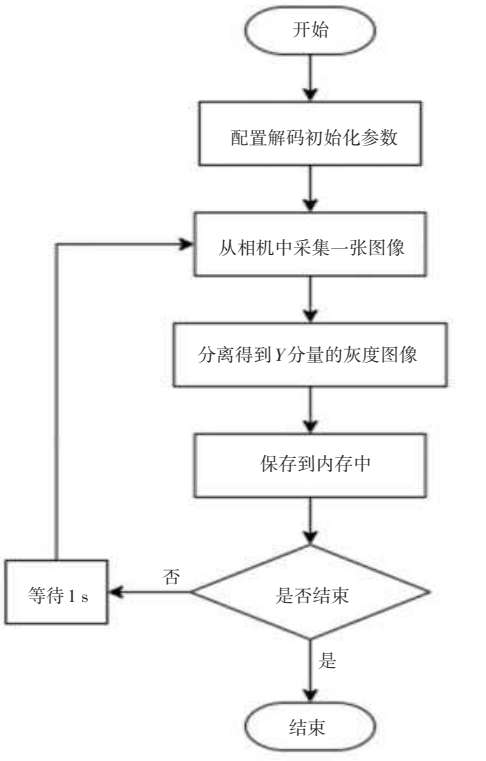

图 3 实时图像采集流程

Fig. 3 Real-time image acquisition process

1.2 图像边缘检测模块

图像边缘检测流程如图 4 所示,图像边缘检测模块分为以下几个步骤:

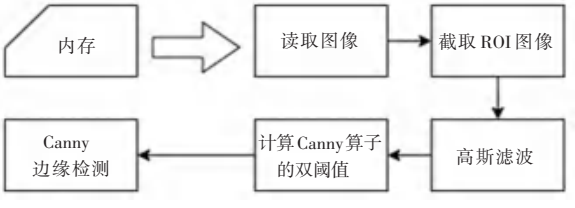

图 4 图像边缘检测流程

Fig. 4 Image edge detection process

(1)从内存区快速读取一张灰度图像,以鼠标单击位置坐标为中心,截取一幅 80×80 正方形图像作为感兴趣区域(Region of Interest, ROI)图像。

(2)对 ROI 图像进行高斯滤波去噪处理。

(3)计算 ROI 图像做 Canny 边缘检测时所需双阈值参数。

(4)利用步骤(3)得到的自适应阈值,进行 Canny 边缘检测,得到边缘图像。

在步骤(1)中,从内存中读取一整幅图像的亮度信息数据,由于嵌入式设备性能有限、采集的图片数据量较大,自动寻边的目标是在鼠标单击点附近寻找最近边缘点,因此使用小面积合适的图像区域计算,可以大大加快图像处理速度。故根据实际情况截取合适边长的正方形,以达到计算速度快和准确检测出边缘的要求。

为了防止噪声像素被检测为虚假边缘,得到图像的准确边缘[8],在步骤(2)中利用高斯滤波对图像进行平滑降噪处理。高斯滤波的原理是距离目标像素越近的像素,其高斯核越大,即距离目标像素越近,则影响越大[9]。高斯滤波可以表示为:

$$I(x,y) = G(x,y,\sigma) \otimes f(x,y) \tag{1}$$

其中,(x,y) 分别表示图像像素点的横坐标和纵坐标;$f(x,y)$ 表示原图;$I(x,y)$ 表示该点经过高斯滤波后的图像;$G(x,y)$ 表示二维高斯函数,可以表示为:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{2}$$

其中,σ 表示高斯滤波器参数,用来控制图像滤波的程度。

在步骤(3)中,边缘检测是该方法研究的重点,Canny 算法提出了 3 个严格的边缘检测标准[10]:好的信噪比、高的定位精度、单边缘响应。根据这 3 个准则,Canny 推导出最优边缘检测算子的一个近似实现,即边界点位于图像被高斯函数平滑后的梯度幅度极大值点上[11]。Canny 算子的基本原理主要包括 4 个部分[12]:平滑图像,上述步骤(2)已经完成;计算梯度的幅值和方向;对梯度幅值进行非极大值抑制;传统的双阈值方法检测和连接边缘。通过对 Canny 算法流程的研究可以得出阈值的选取是进行图像边缘提取的关键。

因此,本文采用梯度幅度直方图的类内特性方差最小化来计算 Canny 算子的双阈值参数方法[10,13]:

将经过非模极大值抑制后的梯度幅值分为 L 级,模极大值分为 3 类:C_0、C_1、C_2,其中 C_0 类为非边缘点像素,C_2 类为边缘点像素,C_1 类里包含需要判断是否为边缘点的点。

设定 n_i 为灰度梯度 i 对应的像素数,N 为图像中总像素数,P_i 为该模级像素数占整个图像像素的比率:

$$P_i = \frac{n_i}{N}, P_i \geqslant 0 \tag{3}$$

则整个区间的梯度幅值期望为：

$$E = \sum_{i=0}^{L-1} iP_i \tag{4}$$

令 C_0 包含模极 $[0,1,\cdots,k]$ 的像素，C_1 包含模极 $[k+1,k+2,\cdots,m]$ 的像素，C_2 包含模极 $[m+1,m+2,\cdots,L-1]$ 的像素。

发生在 C_0、C_1、C_2 类内的梯度幅值期望分别为：

$$E_0(k) = \frac{\sum_{i=0}^{k} i \cdot P_i}{\sum_{i=0}^{k} P_i};\ E_1(k,m) = \frac{\sum_{i=k+1}^{m} i \cdot P_i}{\sum_{i=k+1}^{m} P_i};$$

$$E_2(m) = \frac{\sum_{i=m+1}^{L-1} i \cdot P_i}{\sum_{i=m+1}^{L-1} P_i} \tag{5}$$

并且定义：

$$p(0,k) = \sum_{i=0}^{k} P_i;\ p(k+1,m) = \sum_{i=k+1}^{m} P_i;\ p(m+1, L-1) = \sum_{i=m+1}^{L-1} P_i \tag{6}$$

则可以定义评价函数：

$$\sigma^2(k,m) = [E_0(k) - E]^2 \cdot p(0,k) + [E_1(k,m) - E]^2 \cdot p(k+1,m) + [E_2(m) - E]^2 \cdot p(m+1, L-1) \tag{7}$$

对于已知图像，式(7)中的 i 和 P_i 可以通过它的梯度直方图求出，梯度等级 L 设置为 256，$\sigma^2(k,m)$ 描述了类内特性方差最小化，反映了每类像素之间的差别应当最小，推导求取 $\sigma^2(k,m)$ 最小值化简得到：

$$\begin{cases} 2k - E_0(k) - E_1(k,m) = 0 \\ 2m - E_1(k,m) - E_2(m) = 0 \end{cases} \tag{8}$$

解得的 k,m 的值为 C_0、C_1、C_2 区间的分界点，也就是 Canny 算子的高低阈值。

求得的 k 和 m 双阈值用来对非模极大值抑制后的候选边缘点进行筛选，梯度值大于高阈值 m 的点作为边缘保留；梯度值小于低阈值 k 的点删除；梯度值介于 k 和 m 之间且与边缘点邻接的点作为边缘点保留，否则删除。再判断保留点的 8 个方向中是否存在大于高阈值的边缘像素，如果存在则认为这就是边缘点，否则不是。

在步骤(4)中，使用步骤(3)对 ROI 图像求得的双阈值参数 k 和 m，对图像进行 Canny 边缘检测，得到 ROI 边缘图像。

1.3　自动寻边模块

自动寻边模块流程如图 5 所示。由图 5 可知，自动寻边模块分为以下步骤：

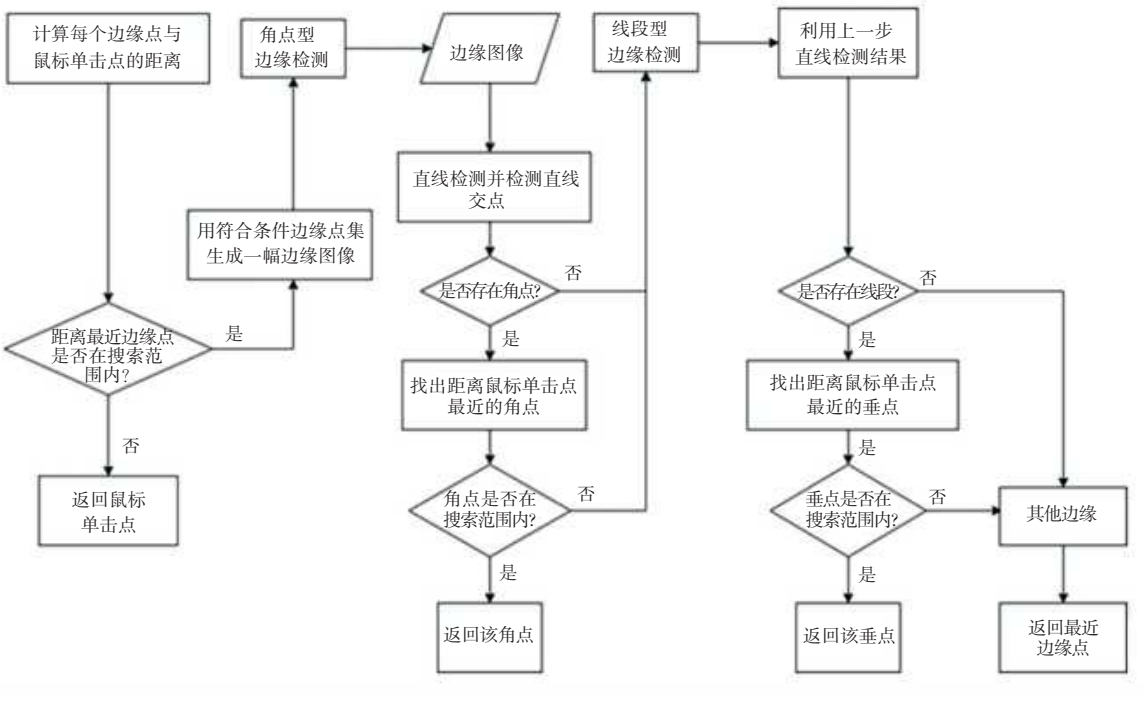

图 5　自动寻边模块流程图

Fig. 5　Flowchart of automatic edge detecting module

(1)提取边缘点信息,判断搜索范围内是否存在边缘信息,如果不存在,则鼠标单击点就是自动寻边结果;如果存在,则计算出离鼠标单击点最近的边缘点集[14]。

(2)在步骤(1)得到的边缘图像基础上,先判边缘是否为角点型。如果是,则搜索范围内与鼠标单击点距离最近的角点就是自动寻边结果;如果不是角点型或者角点不在搜索范围内,进行步骤(3)。

(3)判断边缘是否为线段型。如果是,则搜索范围内鼠标单击点与线段的垂点就是自动寻边结果;如果鼠标单击点恰好位于线段上,则鼠标单击点就是自动寻边结果;如果边缘不是线段型,则接着进行步骤(4)。

(4)如果边缘不是上述 2 种类型之一,则搜索范围内与鼠标单击距离最近的边缘点就是自动寻边结果。

在步骤(1)中,首先使用 OpenCV 中的 *findContours* 函数提取边缘点信息,得到 M 个边缘点集;第 m 个边缘点集包含 N 个边缘点,则所有的边缘点集上的点可以表示为 $P(m,n)(m \in M, n \in N)$,代表第 m 个边缘点集中的第 n 个边缘点。

如果 $M = 0$,则表示不存在边缘点信息,鼠标单击点即是自动寻边结果。

如果 $M > 0$,则计算出所有边缘点 $P(m,n)$ 与鼠标单击点之间的距离 $d_{m,n}$,然后计算出离鼠标单击点最近的距离 $d_{\min} = \min\{d_{m,n} \mid m \in M, n \in N\}$,得到该边缘点 $P(m,n)_{\min}$,判断该点是否在搜索范围内,搜索范围是一个半径为 r 的圆形区域、即寻边精度。如果 $d_{\min} > r$,则说明搜索范围内不存在边缘点,鼠标单击点即是自动寻边结果。

若 $d_{\min} \leqslant r$,则 $P(m,n)_{\min}$ 所在的边缘点集就是最近的边缘,使用此边缘点集重新生成一幅边缘图像,以去除其他边缘的干扰。

在步骤(2)中,首先使用 OpenCV 中的 *HoughLines* 函数对步骤(1)中得到的边缘图像进行直线检测[15],拟合得到 M_l 条直线。如果 $M_l > 1$,则计算直线与直线之间的交点 J,并与步骤(1)中得到的 $P(m,n)_{\min}$ 所在的边缘点集中最接近交点 J 的点 Q 进行对比,在角点检测精度内筛选出合适的交点,由此得到 N_j 个角点。如果 $N_j > 0$,则说明该边缘是角点型,计算得出离鼠标点击点最近的角点,判断该最近角点是否在搜索范围 r 内。若在搜索范围内,则该最近角点为自动寻边结果,若搜索范围内不存在角点,直接进行步骤(3)。

在步骤(3)中,利用步骤(2)直线检测得到的 M_l 条直线结果,步骤(2)中已判断为不满足角点型边缘条件,说明该边缘是线段型,计算得到鼠标点击点与 M_l 条直线垂线距离最短的垂点 T,并与步骤(1)中得到的 $P(m,n)_{\min}$ 进行对比,在满足垂点检测精度下,接下来判断该垂点 T 是否在搜索范围 r 内,若在搜索范围内,则垂点 T 就是自动寻边结果。

若 $M_l = 0$,在该边缘图像内没有检测到直线,则直接进行步骤(4)。

在步骤(4)中,边缘不是上述 2 种类型之一,则步骤(1)中得到的离鼠标单击点最近的边缘点 $P(m,n)_{\min}$ 就是自动寻边结果。

寻边设置示意如图 6 所示。由图 6 看到,可以调整寻边范围(即搜索范围半径 r),r 越小寻边精度越高。根据情况设置边缘检测类型的开启或关闭,可以更准确地获取到需要的边缘点。

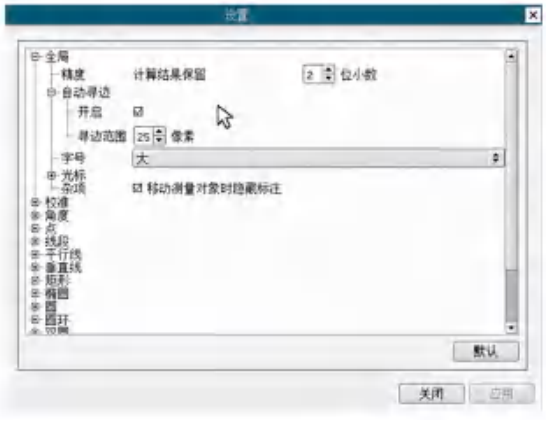

图 6 寻边设置示意图

Fig. 6 Schematic diagram of the edge detecting settings

2 实验结果及分析

本文通过对显微镜拍摄的电路板进行实验,展示了本文提出的边缘自动寻找方法的实际效果。实验结果如图 7~图 9 所示,其中选取了 3 种不同类型的边缘进行测试,分别为角点型边缘、直线型边缘和其他型边缘。图 7~图 9 中的(a)分别是测量的不同区域图像的原图,鼠标点击点为 1,获得以点击点为中心、分辨率为 80×80 的 ROI 区域,寻边范围为 2。图 7~图 9 中的(b)为采用本文方法自适应确定阈值的边缘检测结果图像,其中距离鼠标点击点最近的边缘点集为 3。观察图(b)发现可实现较好效果的边缘检测,图中边缘信息均有检出,整体轮廓完整。图 7~图 9 的(c)为根据最近边缘点集和边缘类型得到的寻边结果,即在 2 所指的寻边范围内计算出离鼠标点击点 1 最近的边缘点 3。

（a）检测区域　（b）边缘检测　（c）寻边结果

图 7　角点型边缘

Fig. 7　Corner point edges

（a）检测区域　（b）边缘检测　（c）寻边结果

图 8　直线型边缘

Fig. 8　Line edges

（a）检测区域　（b）边缘检测　（c）寻边结果

图 9　其他型边缘

Fig. 9　Other type edges

由图 7(b)可知，确定到寻边范围内的离鼠标最近边缘点集是矩形的部分边缘，经过直线检测存在有 2 条直线且在寻边范围内有相交的角点，判断得知在角点精度允许误差之内，确定为角点型边缘，则图 7(c)以该角点作为寻边结果；由图 8(b)得，离鼠标最近边缘点集经过直线检测得到 1 条直线，确定为直线型边缘，由鼠标点击点做直线的垂足，判断得知在直线精度允许误差之内，则图 8(c)以该垂足作为寻边结果；由图 9(b)得，离鼠标最近边缘点集为一段弧线，检测不到直线，确定为其他型边缘，则图 9(c)以该弧线离鼠标点击点最近边缘点作为寻边结果。多次实验结果表明，该方法能准确寻找到角点型、直线型和其他型等各种不同类型的边缘。

表 1 列出了多组图像进行测量时算法的耗时统计。表 1 中，图像边缘检测模块的平均耗时约为 25 ms，而自动寻边模块的平均耗时在 1.5 ms 以内。实验结果表明：这种高效的自动寻边方法非常适合嵌入式设备上的应用，如 HDMI 显微相机等。

表 1　算法检测用时

Table 1　Detection time of the algorithm　ms

边缘检测用时	自动寻边用时	平均总用时
24.92	1.37	26.29

为了进一步对比自动寻边方法和人工手动选取边缘的准确性和耗时性，本文进行了肉眼观察-鼠标点击的传统边缘选取对比实验。在同等条件下，传统边缘选取方式由 6 个人对同一幅图进行边缘选取测试。实验结果和本文提出的鼠标点击-自动寻边方法进行对比，本文认为选取测量点在边缘上即为选中。

对比实验结果见表 2。由表 2 可以看出，不同的测量人员由于观察和测量习惯的不同，选取每个测量点的平均耗时存在差异。肉眼观察和鼠标选点的人工测量平均耗时为 2.32 s，而使用自动寻边方法的测试人员由于不需要精心选择鼠标点击点，平均耗时明显缩短，仅为 1.48 s，即本文提出的自动寻边方法能够将平均耗时缩短约 36%，显著提高了测量的速度。此外，比较各种方法选中边缘的精确度，人工测量的精确度均未超过 75%、平均值仅为 69.86%，而使用本文自动寻边方法的精确度达到 98.59%。因此，本文的自动寻边方法在速度和精确度方面都有很大的提升，相比于人工测量，能够在更短的时间内选取到更精确的边缘测量点。

表 2　人工测量和自动寻边的实验结果

Table 2　Experimental results for manual measurement and automatic edge detecting

测量组别	用时/s	点击个数	平均耗时/(s·个$^{-1}$)	选中边缘点数	精确度/%
No. 1	104.35	59	1.76	40	67.80
No. 2	184.82	62	2.98	43	69.35
No. 3	142.17	62	2.29	46	74.19
No. 4	122.77	60	2.04	38	63.33
No. 5	151.28	63	2.40	44	69.84
No. 6	163.88	67	2.45	50	74.63
平均值	-	-	2.32	-	69.86
本文自动寻边方法	105.16	71	1.48	70	98.59

3 结束语

本文基于 Hi3516A 芯片的 HDMI 显微相机,结合其强大的图像处理能力利用 OpenCV 视觉库首次实现了自动寻边功能。本方法首先开启一个后台线程,每隔 1 s 将相机实时视频流中的图片保存到一块内存中。进行测量操作时,以鼠标单击点为中心,截取一幅 ROI 图像进行边缘检测操作,得到边缘图像。提取边缘点信息,并使用离鼠标单击点最近的边缘点集重新构成一幅边缘图像,进行后续边缘类型判断,并得出最近边缘点。相较于肉眼观察-鼠标定位的传统边缘测量方式,本文提出的鼠标定位-自动寻边方式平均耗时缩短约 37%,精度从 69.86%提升至 98.59%,具有精度高、效率高和延时低的特点。该方法对于未来 HDMI 显微相机等嵌入式设备的智能化测量相关研究具有重要意义。

参考文献

[1] 周芮. 基于 Hi3519A 的 4K 显微相机的设计与实现[D]. 杭州:浙江大学, 2020.

[2] 丁红艳. 基于 Hi3516A 的 HDMI 显微自动对焦相机的设计与实现[D]. 杭州:浙江大学,2018.

[3] CANNY J. A computational approach to edgedetection[J]. IEEE Transactions Pattern Analysis and Machine Intelligence, 1986, 8(6): 679-698.

[4] 唐路路,张启灿,胡松. 一种自适应阈值的 Canny 边缘检测算法[J]. 光电工程,2011,38(5):127-132.

[5] 李牧,闫继红,李戈,等. 自适应 Canny 算子边缘检测技术[J]. 哈尔滨工程大学学报,2007,28(9):1002-1007.

[6] HISILICON. HiMPP V4.0 media processing software development reference[Z]. Shenzhen, China: HiSilicon Technologies Co., Ltd., 2018.

[7] RJSZCB. Yuv422,Yuv420,Yuv444 的区别[EB/OL]. (2021-07-14). https://blog.csdn.net/rjszcb/article/details/118728264.

[8] 王军,林宇航,贾玉彤,等. 一种改进 Canny 算子的图像边缘检测算法[J]. 小型微型计算机系统,2024,45(6):1413-1417.

[9] 陈方升,何纯玉,陈亚飞. 改进 Canny 算法在中厚板边缘检测中的应用[J]. 轧钢,2021,38(5):81-85.

[10] 朱秋林,石银涛,李靖. 一种改进型 Canny 算子边缘检测算法[J]. 地理空间信息,2020,18(1):128-130.

[11] 张启轩,袁明辉. 基于 OpenCV 的物体图像边缘缺陷识别研究[J]. 软件导刊,2021, 20(4): 231-235.

[12] 王小俊,刘旭敏,关永. 基于改进 Canny 算子的图像边缘检测算法[J]. 计算机工程,2012,38(14):196-198.

[13] 罗朝阳,张鹏超,姚晋晋,等. 一种基于形态学的边缘检测算法[J]. 计算机应用与软件, 2020, 37(2):177-181.

[14] GK_2014. 图像处理之计算任意点与轮廓点集中距离最近的点坐标[EB/OL]. (2020-04-12). https://blog.csdn.net/GK_2014/article/details/105477538.

[15] 罗恒, 曹乐乐, 金宗毅,等. 基于 OpenCV 的车道线检测算法改进研究[J]. 北方交通, 2023(2):69-73.