基于手势识别算法的鼠标终端①

李 平, 李允俊

(延边大学 计算机科学与技术学科, 延吉 133002)

摘 要: 提出了一种基于静态手势与动态手势的识别算法, 并结合 Windows API 的鼠标类函数实现鼠标操作. 首先, 通过图像处理技术把从摄像头捕捉的原图像转换为可信度较高的二值图像; 其次, 调用静态手势识别算法识别展开的手指个数, 根据手指个数, 结合 Windows API 的鼠标类函数实现鼠标双击及移动功能; 最后, 当检测到手指个数为 5 时, 调用动态手势识别算法来识别手势的上下左右四个方向, 并结合 Windows API 的鼠标类函数模拟鼠标左右键按下、抬起及滚轮滑动等操作. 实验表明, 该手势识别算法的识别率达到了 94.11%, 对于一些开发平台没有鼠标或在使用鼠标不方便的情况下, 用手势来替代鼠标输入具有一定的研究价值和意义.

关键词: 手势识别算法; 图像处理技术; Windows API; 鼠标

Mouse Terminal Based on Gesture Recognition Algorithm

LI Ping, LI Yun-Jun

(Department of Computer Science and Technology, Yanbian University, Yanji 133002, China)

Abstract: A recognition algorithm based on static gestures and dynamic gestures was proposed in this paper, combined with mouse class functions of Windows API to achieve the mouse operation. Firstly, the original image was captured from the camera was converted to the binary image of high credibility through the image processing technology; and furthermore, calling static gesture recognition algorithm for recognition of spreading fingers number, according to the finger number, combined with mouse class functions of Windows API achieve mouse double-click and move function; finally, when the number of detected fingers was 5, calling dynamic gesture recognition algorithm to recognize gestures up down left right four directions, combined with the mouse class functions of Windows API to simulate the press and lift of left-right mouse button and wheel sliding. The experiment results show that the gesture recognition algorithm recognition rate reached 94.11%, for some development platform without a mouse or inconvenient to use the mouse, using gestures instead of the mouse has a certain value and significance.

Key words: gesture recognition algorithm; image processing technology; Windows API; mouse

人与计算机的交互活动(HCI: Human Computer Interaction)越来越成为人们日常生活中的一个重要组成部分. 目前, 遥控器、鼠标、键盘等作为已经广泛使用的人机交互手段, 虽然已经取得了很大的进步并得到了广泛的认可, 但是从自然、直接、方便的方面考虑, 仍然存在许多不尽人意的地方. 手势分析由于其固有的自然、方便的人际交流特性, 越来越受到广大研究人员的关注, 成为下一代人机交互技术中的一大研究热点[1].

手势分为静态和动态两类. 其中, 静态手势是指一个特定的手势或姿态, 用单幅图像来表示, 而动态手势是指运动的手势, 用一系列连续图像表示[2]. 对手势的识别大体可以分为三类. 第一类, 基于数据手套的手势识别[3]; 第二类, 基于触摸屏的手势识别, 将手在触摸屏上的运动作为输入; 第三类, 基于视觉的手势识别, 它模仿人类接受周围信息的方式, 是最自然的人机交互方式, 也是难度最大的方式.

本文考虑一些开发平台没有鼠标或在使用鼠标不

① 收稿时间:2013-01-02;收到修改稿时间:2013-03-06

通讯作者:李允俊, yjlee@ybu.edu.cn

方便的情况下, 提出了一种静态和动态手势相结合的识别算法, 并借助 Windows API 鼠标类函数来替代鼠标终端输入功能.

1 相关知识介绍

1.1 现有手势识别算法

手势识别分为静态手势识别和动态手势识别. 现有的静态手势识别算法主要是判别展开手指个数[4]. 算法流程如图 1 所示:

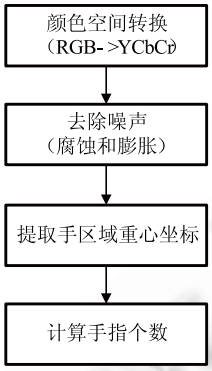


图 1　手指个数识别算法的流程

图 1 中首先通过颜色空间转换把一副 RGB 图像转换为 YCbCr 图像, 然后根据人的肤色在 YCbCr 颜色空间中呈现良好的聚类特性[5-7], 统计分布满足公式(1):

$$77 \le Cb \le 127$$
$$133 \le Cr \le 173 \tag{1}$$

公式(1)中 Cb 和 Cr 为 YCbCr 颜色空间中颜色分量, 根据公式(1)中聚类特点来提取手区域二值图像, 进而通过图像形态学方法去除噪声. 在去除噪声的手区域二值图像中提取手区域重心坐标, 提取手区域重心坐标公式(2)[8]如下:

$$C(x,y)=\frac{\sum_{m=0}^{N} p_{m(x,y)}}{N} \tag{2}$$

在公式(2)中, $P_m(x,y)$为手区域中第 m 次像素值, N 为手区域中像素总数目. 在求得手区域重心坐标后以手区域重心坐标为圆心画一个空心圆, 并统计手区域和圆不相交区域的个数, 以不相交区域的个数作为手指个数. 算法实现过程如图 2 所示.

现有的动态手势识别主要有利用手区域重心坐标的变化量来判别横向和纵向手势[9]以及利用角度来判定手势方向[10]等识别算法. 判定横向和纵向动态手势

图 2　手指个数识别算法图示

的识别过程可以由公式(3)表示:

$$\begin{aligned} Horizontal &= \frac{1}{2}\Delta x \ge \frac{1}{2}\Delta y \\ Vertical &= \frac{1}{2}\Delta y \ge \frac{1}{2}\Delta x \end{aligned} \tag{3}$$
$$\Delta x=\frac{\sum_{i=1}^{n}|x_i-x_{i-1}|}{n} \qquad \Delta y=\frac{\sum_{i=1}^{n}|y_i-y_{i-1}|}{n}$$

其中Δx 代表重心坐标(x,y)中 x 的平均变化量, Δy 代表重心坐标(x,y)中 y 的平均变化量, 连续手重心坐标个数为 n+1 个. 利用角度判定动态手势识别算法如图 3 所示.

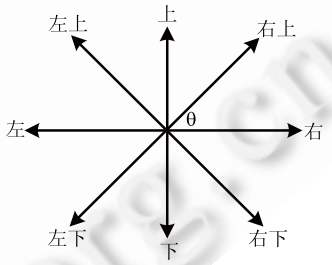


图 3　动态手势识别算法示意图

在图中以起始位置手重心坐标为原点(x_0, y_0), $\theta=\tan^{-1}(\Delta y/\Delta x)$, Δy 为当前点 y 坐标与起始点 y 坐标差值; Δx 为当前点 x 坐标与起始点 x 坐标差值, 并选用一定的角度判定动态手势的方向性.

1.2 Windows API 鼠标类函数

Windows 多任务系统除了协调应用程序的执行、分配内存、管理资源之外, 它同时也提供开启视窗、描绘图形、使用外设等应用程序接口(API: Application Programming Interface).

本文涉及到 Windows API 函数, 主要有:

① BOOL GetCursorPos(LPPOINT　lpPoint); //得到当前鼠标指针的坐标

② BOOL SetCursorPos(int X, int Y);//设置鼠标在屏幕上的坐标

③ VOID mouse_event(

DWORD dwFlags, //鼠标动作标识
DWORD dx, //鼠标水平方向位置
DWORD dy, //鼠标垂直方向位置
DWORD dwData, //鼠标轮子转动的数量
ULONG_PTR dwExtraInfo //关联鼠标动作辅加信息
);

④ BOOL ClipCursor(CONST RECT * lpRect);//限制鼠标活动区域(矩形区域)

2 手势识别算法的设计

2.1 静态手势识别算法

为了模拟鼠标双击及移动操作, 需要使用判别手指个数的识别方法, 由于已存在的手指个数识别算法没有规定圆半径大小, 会给识别带来一定的误差. 为了弥补这点, 本文提出了一种新的手指个数判别算法, 算法流程如图 4 所示:

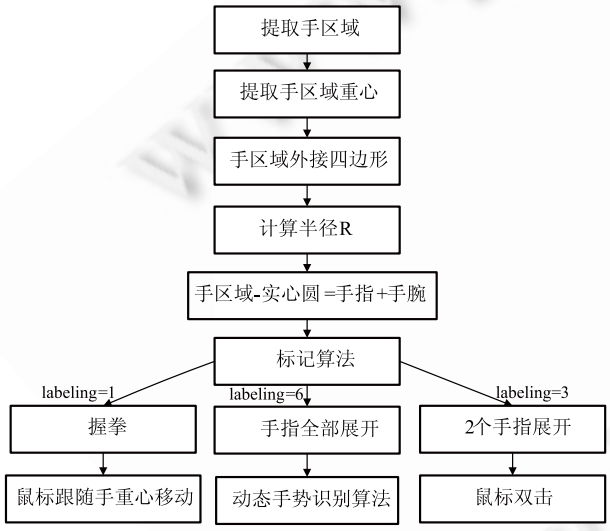


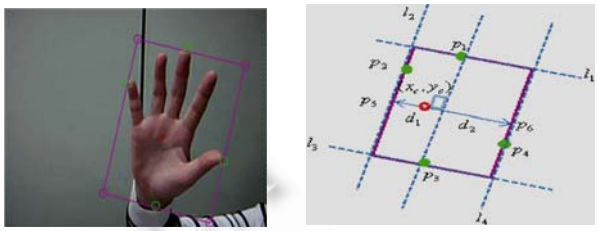
图 4 静态手势识别算法流程

首先通过图像处理技术获取手区域二值图像, 通过公式(2)求得手区域重心坐标; 进而提出一种求手区域外接四边形的方法来限定实心圆半径的大小, 手区域外界四边形及其图示如图 5 所示.

手区域外接四边形按照以下三步求得:

1) 提取手区域轮廓中四点 p_1, p_2, p_3, p_4. p_1 为手轮廓中坐标 y 最小点; p_2 为手轮廓中坐标 x 最小点; p_3 手轮廓中坐标 y 最大点; p_4 手轮廓中坐标 x 最大点.

2) 对于选取 p_1, p_2, p_3, p_4 四点, 两两组合, 有 6 种组合; 进而求得这 6 种组合中长度最长的两点, 假设为$|p_1, p_3|$.



(a) 手区域外接四边形 (b) (a)的图示

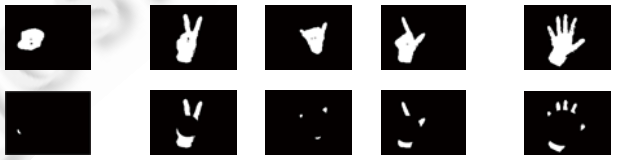
图 5 手区域外接四边形及其图示

3) 已知$p_1(x_1,y_1), p_3(x_3,y_3)$求得 $k_1=(y_3-y_1)/(x_3-x_1)$; 进而求得过点 p_2 和 p_4, 斜率为 k_1 的直线 l_2 和 l_4; 以及过 p_1 和 p_3, 斜率为 $k_2=-(1/k_1)$的直线 l_1 和 l_3; 手区域外接四边形求得.

求得手区域外接四边形后, 求过手区域重心(x_c,y_c), 斜率为 k_2 的直线与分别与直线 l_2 和直线 l_4 的交点 p_5 和 p_6, 进而计算手重心(x_c,y_c)分别到点 p_5 和 p_6 距离 d_1 和 d_2, 通过公式(4)计算实心圆半径. 其中, 半径 R 是选定 d_1 和 d_2 中较小的值乘上实验参数α来计算.

$$R=\begin{cases} d_1\times\alpha, & if \quad d_1\le d_2 \\ d_2\times\alpha, & if \quad d_1\ge d_2 \end{cases} \tag{4}$$

求得半径 R 后, 画一个实心圆; 采用 CvSub 函数(手区域去除实心圆区域)来获取只含有手指和手腕区域的二值图像, 如图 6 所示:



labeling =1 labeling=3 labeling=6

图 6 CvSub 函数实现过程

在只含有手指和手腕的二值图像中调用标记算法, 根据标记值来判断静态手势, 进而调用 Windows API 来模拟鼠标输入功能.

当 labeling=1, 判断手势为握拳状态, 鼠标跟随手重心(x_c,y_c)在计算机屏幕上移动. 手重心坐标转换成计算机屏幕坐标(_x,_y)过程如图 7 所示.

首先通过 GetSystemMetrics 函数获取计算机屏幕分辨率 *dmPelsWidth* 和 *dmPelsHeight*, 已知手区域所在

的图像分辨率为 320*240, 通过公式(5)来实现手重心坐标(x_c,y_c)到屏幕坐标(_x,_y)的转换.

$$\begin{aligned} &\_x : x_c = 320 : dmPelsWidth \\ &\_y : y_c = 240 : dmPelsHeight \end{aligned} \tag{5}$$

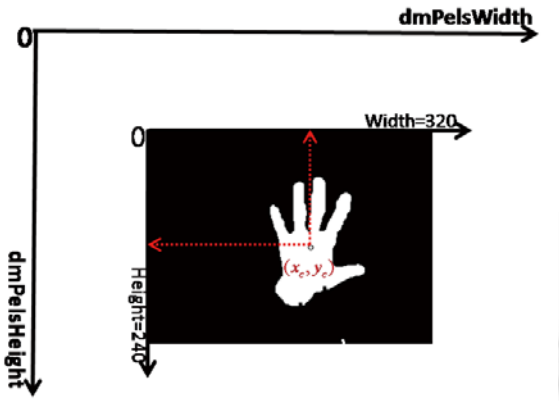


图 7　手重心坐标转换为屏幕坐标示意图

得到屏幕坐标后, 调用 Windows API 函数 SetCursorPos(_x,_y)来设置鼠标在屏幕上的坐标.

当 labeling=3, 判断手势为展开 2 个手指状态, 调用 Windows API mouse_event 函数来模拟鼠标双击操作.

当 labeling=6, 判定手势为手指全部展开状态, 调用动态手势识别算法.

2.2 动态手势识别算法

由于原有动态手势识别算法是基于八方向, 并且是利用角度来判断手势方向, 较容易产生误差. 本文为了弥补这一点, 提出了新的动态手势识别算法来判定手势的上下左右四个方向.

前提: 在手指全部展开的前提下, 我们提取一系列手重心坐标为 Data={t[0],t[1],...,t[n]}.

算法描述:

1) 求经过 t[0], 并且斜率分别为 1 和-1 的两条直线 l_1 和 l_2, 如图 8 所示:

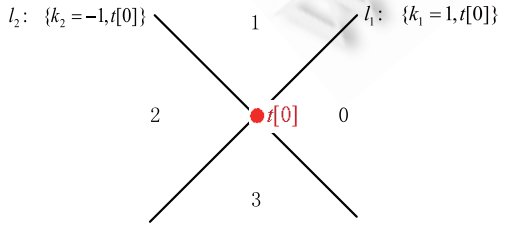


图 8　动态手势识别算法图示

图 8 把整个平面区域划分为四部分, 0 区域代表手势向右; 1 区域代表手势向上; 2 区域代表手势向左; 3 区域代表手势向下.

2) 判断一系列手重心坐标分布(i=1,2,...,n), 判定规则如下:

t[i]在 l_1 下面, 在 l2 上面: 0 区域数 *count*0 加 1;

t[i]在 l_1 上面, 在 l2 上面: 1 区域数 *count*1 加 1;

t[i]在 l_1 上面, 在 l2 下面: 2 区域数 *count*2 加 1;

t[i]在 l_1 下面, 在 l2 下面: 3 区域数 *count*3 加 1;

3) 比较 *count*0~*count*3 值, 把其中值为最大的返回.

return *count*0: Right

return *count*1: Up

return *count*2: Left

return *count*3: Down

判定动态手势方向后, 通过手势的方向来调用相应 Windows API 函数来模拟鼠标输入, 具体操作如下:

向左: 模拟鼠标左键按下;

向右: 模拟鼠标右键按下;

向下: 模拟鼠标滑轮向后滑动;

向上: 模拟鼠标松开左键/松开右键/滑轮向前滑动.

3　实验与结果分析

本文采用实验工具为 Microsoft Visual Studio 6.0, 函数库为 Open CV 1.0, 图像分辨率为 320*240. 由于半径大小直接影响静态手势的识别率, 因此, 为了获得适当半径大小, 本文通过不同人以及距离摄像头远近不同依次进行了多次实验, 确定公式 4 中实验参数α为 1.2.

3.1 静态手势实验与结果分析

静态手势识别算法主要识别握拳、展开 2 个手指、展开 5 个手指. 在展开 2 个手指中, 本文根据大多数人行为习惯采用了 3 种方式(数字手势 2、6、8). 每种手势我们采用正对摄像头和不正对摄像头两种方式, 分为左手和右手分别进行实验, 每种方式进行 20 次; 一种姿势有四种方式, 共 80 次; 一共 5 种姿势, 共进行 400 次实验, 实验结果如表 1 所示.

从表 1 平均识别率可以看出, 握拳状态的手势识别率最低, 原因是在握拳状态二值图像中去除实心圆后, 有时食指部分会有遗留, 这种情况下标记值就会出现值为 2 的情形.

表 1 静态手势实验结果

标记值(labeling)	手	方向	实验次数	识别个数	识别率	平均识别率
标记值=1 握拳	右手	端正	20	18	90%	83.75%
		倾斜	20	16	80%	
	左手	端正	20	17	85%	
		倾斜	20	16	80%	
标记值=3 数字手势 2	右手	端正	20	20	100%	96.25%
		倾斜	20	19	95%	
	左手	端正	20	20	100%	
		倾斜	20	18	90%	
标记值=3 数字手势 6	右手	端正	20	19	95%	91.25%
		倾斜	20	18	90%	
	左手	端正	20	19	95%	
		倾斜	20	17	85%	
标记值=3 数字手势 8	右手	端正	20	20	100%	96.25%
		倾斜	20	19	95%	
	左手	端正	20	20	100%	
		倾斜	20	18	90%	
标记值=6 手指全部展开	右手	端正	20	20	100%	97.5%
		倾斜	20	19	95%	
	左手	端正	20	20	100%	
		倾斜	20	19	95%	

3.2 动态手势实验与结果分析

在动态手势识别算法中, 对于手势四个方向(上下左右), 分为左手和右手分别进行, 一个动作进行 20 次, 一种方向进行 40 次; 四个方向, 共进行 160 次实验. 实验结果如表 2 所示:

表 2 动态手势实验结果

手势方向	手	实验次数	识别个数	识别率	平均识别率
左	右手	20	20	100%	97.5%
	左手	20	19	95%	
右	右手	20	20	100%	100%
	左手	20	20	100%	
上	右手	20	20	100%	97.5%
	左手	20	19	95%	
下	右手	20	19	95%	92.5%
	左手	20	18	90%	

从表 2 可以看出, 从平均识别率来看, 由于人们行为习惯原因导致向右识别率最高; 而向下识别率明显低于其他方向识别率.

3.3 整体分析

整体实验结果如表 3 所示:

表 3 整体实验结果

	左手	右手	静态手势	动态手势	总实验
实验次数	280	280	400	160	560
识别个数	260	267	372	155	527
识别率	92.86%	95.36%	93%	96.88%	94.11%

从表 3 可以看出, 由于人类行为习惯, 右手识别率明显高于左手识别率; 静态手势识别率要低与动态手势识别率, 原于握拳状态识别效果不理想所导致.

4 结语

本文提出了一种基于静态手势与动态手势的识别算法, 并结合 Windows API 鼠标类函数来现鼠标操作. 静态手势识别算法主要判别展开手指个数, 而动态手势识别算法主要判别手势方向(上下左右), 通过已判定的手势结合 Windows API 鼠标类函数实现模拟鼠标输入功能. 实验表明, 总识别率为 94.11%, 其中动态手势识别率为 96.88%, 而静态手势识别率为 93%. 可见静态手手势识别率明显低于动态手势识别率, 其原因是静态手势握拳状态识别效果不理想所导致的, 下一步工作是进一步研究手势特性, 进而提出更好的解决方案.

参考文献

1 柴秀娟.用于视觉交互系统的手势跟踪和识别研究[硕士学位论文].北京:北京邮电大学,2009.

2 江冬梅.基于视觉的手势识别及其在人机交互中的应用[硕士学位论文].济南:山东大学,2004.

3 孙丽娟,张立材,郭彩龙.基于视觉的手势识别技术.计算机科学与发展,2008,18(10):214–217.

4 Choi KM, Na YG, Chae Sb, Jung KH. A Hand Gesture-based Remote Control of Robot. Proc. of KSBE Fall Conference, 2010: 196–199.

5 Yang J, Waibel A. A Real-Time Face Tracker. Proceedings 3rd IEEE Workshop on,1996:142–147.

6 邵林昌.基于肤色分割的人脸检测[硕士学位论文].南京:东南大学,2006.

7 Hsu RL, Mohamed AM, Jain AK. Face Detection in Color

(下转第 71 页)

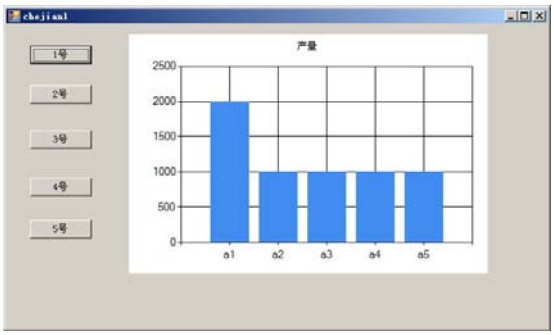


图 7 车间 1 的产品数量的统计

图 8 故障情况

4 结论

本文设计了以 CC2530 为核心, 应用于绕线机上的 ZigBee 模块. 通过 ZigBee 无线通信方式, 将绕线机上的运行参数, 故障统计, 生产产品数量的统计. 并且通过 PC 终端对绕线机进行参数设置. 极大地方便了绕线机的监控和控制, 为管理者的监控提供了便利, 也便于统计绕线机生产的总个数, 大大加强了工作的效率.

参考文献

1 韩华峰,杜克明,孙忠富,等.基于 ZigBee 网络的温室环境远程监控系统设计与应用.农业工程学报,2009,25(7):158–163.

2 江修波.ZigBee 技术及其应用.低压电器,2005(7):27–29,33.

3 方智文,邓启平,蒋良兵,等.基于 zigbee 的矿井人员管理系统设计.中国科技财富,2011(22):199.

4 冯军,宁志刚,阳璞琼,等.基于 ZigBee 的无线抄表系统设计.电力自动化设备,2010,30(8):108–111.

5 Yao XL, Hu NL, Li GQ, Chen DG. Research on mining energy measurement system based on the ZigBee. Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference, 2012, 6: 488–491.

6 王风.基于 CC2530 的 ZigBee 无线传感器网络的设计与实现[学位论文].西安:西安电子科技大学,2012.

7 金纯,蒋小宇,罗祖秋,等.ZigBee 与蓝牙的分析与比较.信息技术与标准化,2004,(6):17–20.

8 Higuera J, Kartsakli E, Valenzuela JL, Alonso L, Laya A, Martinez R, Aguilar A. Experimental Study of Bluetooth, ZigBee and IEEE 802.15.4 Technologies on Board High-Speed Trains. Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th, 2012: 1–5.

9 章伟聪,俞新武,李忠成,等.基于 CC2530 及 Ziee 协议栈设计无线网络传感器节点.计算机系统应用,2011,20(7):184–187,120.

10 刘广亚,姚善化.基于 ARM9 和 ZigBee 的矿井机车运输监控系统的设计.矿山机械,2012,40(1):28–31.

11 王春,廖映华,刘高君,等.基于 ZigBee 和 CAN 总线的多焊机群控系统设计.焊接技术,2012,41(1):34–36.

(上接第 87 页)

Images. IEEE Transactions on Pattern Analysis and Machine Intelligence,2002,24(5):696–706.

8 Lee DS, Kim SD, Lee DW, Yoo J. 3D world space recognition system using stereo camera. Proc. of KSBE Fall Conference, 2008: 215–218.

9 Son HS, Seo IK, Kim JH, Yun TS, Lee DH. A Study on Development of Arcade Game based Gesture Recognition. Proc. of South Korea Entertainment Society, 2010,13(2): 330–333.

10 Ko MS, Lee KH, Kim CW, Ahn JH, Kim IJ. An Implementation of User Interface Using Vision-based Gesture Recognition. Proc. of South Korea Computer Comprehensive Academic Conference, 2008, 35 (1): 507–509.