

基于 Kinect 的实时稳定的三维多手指跟踪算法

晏 浩¹⁾, 张明敏^{1)*}, 童 晶^{1,3)}, 潘志庚^{1,2)}

¹⁾ (浙江大学 CAD & CG 国家重点实验室 杭州 310058)

²⁾ (杭州师范大学数字媒体与人机交互研究中心 杭州 310012)

³⁾ (河海大学物联网工程学院 常州 213022)

(zmm@cad.zju.edu.cn)

摘 要: 针对现有手指跟踪算法的不足, 利用微软 Kinect 设备提出一种实时鲁棒的三维多手指跟踪算法. 首先利用深度图分割出一个粗糙的手区域, 对该区域应用基于像素分类的指尖检测算法得到二维指尖点的位置; 然后在深度图上对二维指尖位置周围的点进行采样, 将均值作为指尖点的 Z 坐标, 再利用卡尔曼滤波器以及帧之间的连续性对指尖点的三维位置进行跟踪. 依据 Kinect 数据特点提出的二维指尖检测算法和利用帧之间连续性的卡尔曼滤波器是文中算法的关键. 实验结果证明, 该算法能够实时、稳定地进行三维多手指跟踪.

关键词: 多手指跟踪; 3D 人机交互; Kinect

中图分类号: TP391.4

Real Time Robust Multi-fingertips Tracking in 3D Space Using Kinect

Yan Hao¹⁾, Zhang Mingmin^{1)*}, Tong Jing^{1,3)}, and Pan Zhigeng^{1,2)}

¹⁾ (State Key Laboratory of CAD & CG, Zhejiang University, Hangzhou 310058)

²⁾ (Digital Media & Interaction Research Center, Hangzhou Normal University, Hangzhou 310012)

³⁾ (College of Internet of Things Engineering, Hohai University, Changzhou 213022)

Abstract: In this paper, we introduce a fast and robust algorithm for tracking 3D multi-fingertips using Kinect. Our method firstly uses Kinect to get a rough hand region, then applies the fingertip detection method based on pixel classification to get the 2D fingertips. Finally, we sample on the depth map around the detected fingertip to get the Z-value of the fingertips and then combine the Kalman filter and the continuity between frames to track the 3D position of fingertips. The fingertip detection method based on the specialty of the data of Kinect and the Kalman filter applied with continuity between frames are the key of our algorithm. The experimental results show that our algorithm can robustly track 3D multi-fingertips in real time.

Key words: multi-fingertips tracking; 3D human computer interaction; Kinect

在人机交互和虚拟现实系统中, 实时、稳定的多手指跟踪能给用户带来友好的交互体验; 稳定的手指跟踪还能提供手的结构信息及其运动轨迹, 因此也能够显著地提高手势识别的准确性. 由于能够跟踪到手指的三维坐标, 三维的多手指跟踪可以使用

户以操作真实世界物体的方法去操作虚拟世界的物体, 这势必大大增强人机交互系统和虚拟现实系统的沉浸感.

基于三维多手指跟踪丰富的应用前景, 它得到了研究者的广泛关注, 其算法主要分为手区域的分割,

收稿日期: 2012-12-28; 修回日期: 2013-09-20. 基金项目: 国家自然科学基金(61173124, 61202284). 晏 浩(1988—), 男, 硕士, 主要研究方向为虚拟现实、计算机视觉等; 张明敏(1968—), 女, 博士, 副教授, 硕士生导师, 论文通讯作者, 主要研究方向为虚拟现实/增强现实、实时渲染、多媒体技术、图像处理; 童 晶(1984—), 男, 博士, 主要研究方向为虚拟现实、增强现实、人机交互; 潘志庚(1965—), 男, 博士, 教授, 博士生导师, 主要研究方向为人机交互、虚拟现实、多媒体技术、数字娱乐.

指尖的检测,指尖三维位置的获取,指尖三维位置及其轨迹的跟踪 4 个步骤。其中,手区域的分割是一个很重要的步骤,它直接决定了指尖检测算法的准确性。手区域分割比较流行的方法是基于肤色的像素分类^[1-3],该方法的缺点是易受灯光等外部环境的影响,在光照条件不好的情况下容易失败。为此,一些研究者利用红外相机来得到可靠的手分割区域^[4-6],然而红外相机价格昂贵。还有一些研究者通过给出一些限制条件,如固定的背景、固定的光照等来得到较好的分割区域^[7-10],但此方法限制了手指跟踪的应用场合。指尖的检测算法主要分为基于轮廓^[11-12]、基于形状^[13]和基于模板^[4-6] 3 类。基于轮廓方法的缺点是要求手区域的轮廓比较准确,否则就会出现错误;基于模板匹配的方法也有基于轮廓方法的缺点,而且速度比较慢;基于形状的方法中,要确定一个合适的 open 操作窗口大小是比较困难的^[13]。

在 Kinect 等类似的深度相机出现之前,指尖三维位置的获取一般是利用多相机来重建指尖的三维位置^[14-16],该方法的优点是由于使用了多个相机,因此能处理在单相机条件下的自遮挡问题;缺点是实验前需要对相机进行定标,而且重建的三维指尖点的精度也不高。

指尖轨迹的跟踪中用得最多的是滤波器算法^[4-6],其中 Oka 等^[4]利用一个红外相机来分割手区域,然后利用模板匹配的方法找到指尖的位置,最后通过卡尔曼滤波器来跟踪多根手指的运动轨迹;将此算法运用于其 Autodesk 交互系统中取得了比较好的效果,但是由于没有深度值,因此只是对手指的二维坐标跟踪,基于该算法也只能进行二维平面上的交互。

随着微软 Kinect 设备的推出,利用这一廉价的硬件设备能够实时捕获场景的彩色和深度信息。一些研究者尝试利用 Kinect 进行多手指跟踪的研究。Raheja 等^[17]利用 Kinect 进行手区域的分割,并利用深度图来寻找指尖点的位置。Ren 等^[18]利用 Kinect 进行手区域的分割,然后利用手区域的时间序列曲线来求出指尖点位置,最后给出了基于 FEMD(finger-earth mover's distance)的手势识别方法。文献^[17-18]都是在 Kinect 给出的深度图的基础上设计更加鲁棒的二维指尖检测算法,但没有在得到二维指尖点之后继续利用 Kinect 给出的深度值去跟踪指尖点的三维坐标和轨迹。Oikonomidis 等^[19]提出了基于模型的手关节跟踪算法,首先利用 Kinect 分割出手区域,然后利用手区域深度图和二值图将问题转化成一个优化问题——找到一个与现有观察值最匹配的手模型;此方法虽然效果比较好,

但是算法复杂且计算量大,在 GPU 加速的情况下跟踪一只手只能达到 15 帧/s。

本文算法利用 Kinect 的深度图来分割手区域,并且根据 Kinect 的数据特点提出一种改进的基于像素分类的指尖检测算法;在此基础上,利用 Kinect 的深度图获得指尖的三维坐标,然后通过应用帧之间连续性的卡尔曼滤波器实现指尖三维位置和轨迹的稳定跟踪。同文献^[17-18]相比,本文算法能稳定地跟踪到指尖的三维位置,不仅能实现二维人机交互,还能有效地进行三维人机交互。同文献^[19]相比,本文算法更加简单,速度更快,在一般 PC 平台同时跟踪 2 只手的所有手指的情况下,也能达到 20+帧/s,跟踪一只手能达到 40+帧/s。

1 本文算法

本文算法主要分为 4 个步骤:手区域的分割,二维指尖检测,指尖三维位置获取和三维指尖轨迹的跟踪。

1.1 手区域的分割

1) 首先获得 Kinect 的深度图(如图 1a 所示),然后利用 NITE 库^①跟踪手点(手中心点)位置(如图 1b 所示)。跟踪和检测到手点的位置以后,根据手点的 Z 坐标对深度图进行分割

$$I(x, y) = \begin{cases} 255, & Z_{\text{hand}} - \theta/2 < Z(x, y) < Z_{\text{hand}} + \theta/2 \\ 0, & \text{其他} \end{cases}$$

其中, $I(x, y)$ 标记像素点是否为手区域; Z_{hand} 表示跟踪到的手点的深度值; $Z(x, y)$ 表示像素点 (x, y) 处的深度值; θ 表示手的最大深度范围,本文取为 160 mm。

2) 将手点投影到二维,根据手的大小再做一次二维分割

$$I(x, y) = \begin{cases} 0, & (x, y) \notin W(Z) \\ 255, & \text{其他} \end{cases};$$

其中 $W(Z)$ 表示以手点为中心的一个包围盒,包围盒的大小与手距离 Kinect 的远近有关。当手离 Kinect 近的时候, $W(Z)$ 大;反之则小。具体的关系可根据实验得出,本文取 $S_{\text{Width}}(W(Z)) = S_{\text{Height}}(W(Z)) = 2 \times \min(\max\{80 \text{ 像素} - 0.2 \times (Z - 640 \text{ mm}), 60 \text{ 像素}\}, 80 \text{ 像素})$, 其中 S_{Width} 与 S_{Height} 分别表示包围盒的宽度与高度,以像素为单位。分割原理如图 1b 所示。

① <http://www.primesense.com/nite>

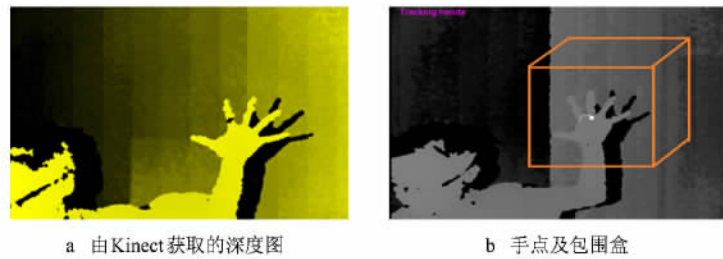


图 1 根据深度信息分割手区域

1.2 指尖的检测

手区域的分割是根据深度值来进行的,由于 Kinect 给出的深度值存在误差,导致手区域的轮廓不是特别准确(轮廓的边缘锯齿比较严重,如图 2 a 所示),在运动时,由于运动模糊的存在,效果就更加差.本文经过多次实验发现,基于轮廓和基于模板的方法对于稍微弯曲的手指的检测效果不好,而基于

形态学操作的算法速度比较慢、且经常性地检测到错误的指尖点.因此,本文根据 Kinect 数据的特点采用基于像素分类的方法^[20],并对这个方法做了改进,不仅重新定义了一些特征,而且在最后求指尖位置时不是对候选指尖区域作聚类得到最终的指尖点,而是对候选手指区域作椭圆拟合得到指尖位置和手指方向.



图 2 获取指尖信息的 3 个步骤

指尖检测算法主要分为 2 步:1)根据预先定义的特征对手区域二值图像进行基于像素的分类;2)对手指区域进行椭圆拟合,得到指尖位置.本文根据指尖的几何特征定义了 4 个特征来对像素进行分类:

特征 1. 点 p 是不是手区域,如图 3 a 所示;

特征 2. 以点 p 为圆心, g_1 为半径的圆区域中手区域所占比例的大小,如图 3 b 所示;

特征 3. 以点 p 为圆心, g_2 为半径的圆与手区域有几个相交点,如图 3 c 所示;

特征 4. 分割点 AB 之间的距离跟手指的大小是否接近,如图 3 d 所示.

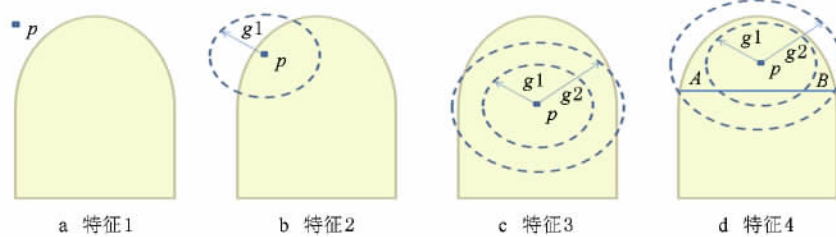


图 3 预先定义的 4 个特征

由于每个人的手指大小是不同的,而且手指的大小也是跟手距离 Kinect 的远近相关,因此 g_1 和 g_2 的选择很重要,直接影响到三维指尖检测的准确性.鉴于以上事实,本文在程序运行开始时设计了一个训练过程:让用户伸出一根手指在 Kinect 前面前后来回移动数次,在移动过程中用基于轮廓方法求得指尖的位置(此时由于移动速度比较慢,没有运动模

糊的存在,而且也不会有手指弯曲的情况,因此基于轮廓的指尖检测算法能够满足要求);求得指尖点的位置之后,以该指尖点为圆心,以某一给定值 r 为半径画一个圆,然后求得该圆与手区域的 2 个交点,交点之间的距离即为手指的大小,如图 4 所示.图 4 a~4 f 依次表示从远到近的 6 个深度区间,细线表示的圆以指尖点为圆心,以 $r=20$ 像素为半径;粗线表示

的直线为圆与手区域的相交区域,其长度即为手指的大小; r 只需大于一般手指的大小且小于手指的长度即可,因此比较好取值,实验中取为 20 像素.考虑到较小的距离(10 cm)对手指大小影响不是很大,本文将深度进行离散化.在实验中,我们将深度离散成为 6 段:[501 mm,600 mm],[601 mm,700 mm],[701 mm,800 mm],[801 mm,900 mm],[901 mm,1000 mm]和[1001 mm,1100 mm].之所从 500 mm 开始,是因为当深度值小于 500 时,Kinect 的深度值变得无效.

在得到每一段的手指大小 L_i 之后, $g1$ 和 $g2$ 的取值按

$$\begin{aligned} g1_i &= L_i/2 - \Delta, \\ g2_i &= L_i/2 + \Delta \end{aligned}$$

获得.其中 Δ 取为 3.

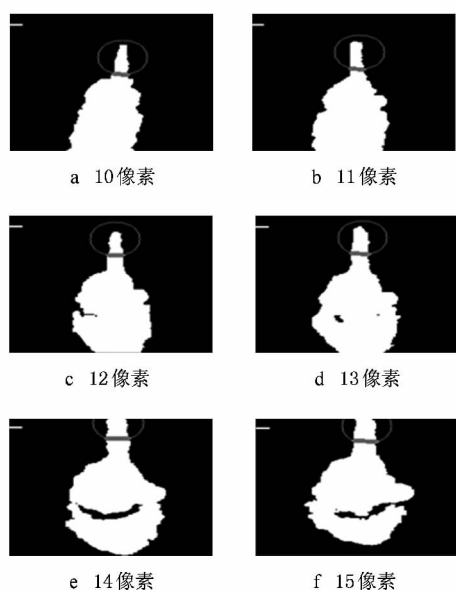


图 4 一组训练序列图

定义好特征之后,本文使用决策树的方法对每个像素进行分类,分类算法流程如图 5 所示.为了提高运算速度,将决策树中检测的特征按照计算量从小到大排序^[20],分类结果如图 1 b 所示.

得到像素的分类结果之后可以直接对指尖区域进行聚类,得到指尖点.但是在实验中我们发现,红色区域的正确性对 $g1$ 和 $g2$ 大小的选取非常敏感,而黄色区域则相对比较稳定,因此本文采用如下的算法求得指尖点:先得到红色和黄色的混合区域 R ,然后对 R 进行形态学操作去掉噪声,再对剩下面积大于某阈值的区域进行椭圆拟合,将该椭圆的长轴端点中距离手点较远的那个作为指尖点,同时得到手指的方向.具体过程如图 6 所示.

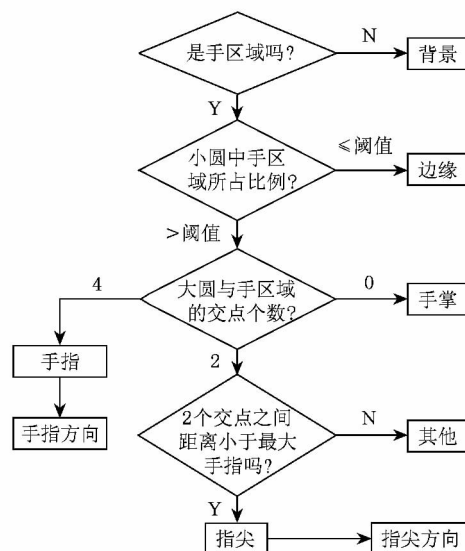


图 5 基于像素分类的指尖检测算法框架

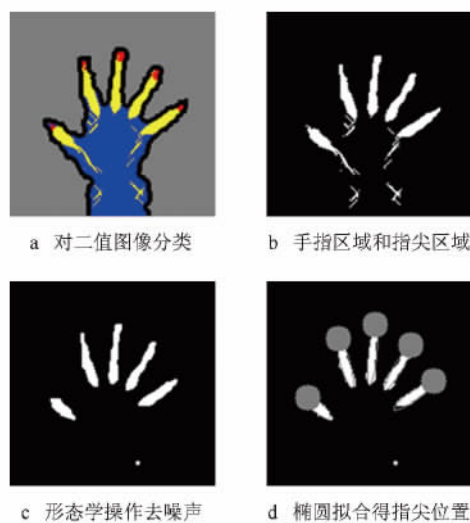


图 6 获取指尖点的过程

1.3 指尖三维位置的获取

得到二维指尖位置 $P_1(x, y)$ 之后,理论上可以直接取 Kinect 深度图上 (x, y) 位置的值作为指尖点的 Z 坐标,然后通过坐标变换得到指尖点的世界坐标.但是,Kinect 的深度值是有误差的,且在运动过程中存在运动模糊,如果采取这个方法指尖的三维位置不稳定.本文的处理方法是:周围采样和范围过滤,如图 2 c 所示.

周围采样指通过在指尖周围的某一个范围之内进行采样,将这些采样点的深度值累加再求其均值.但是在采样的过程中不是所有的采样点都会被用到,只有当采样点的深度值在 NITE 跟踪到的手点的深度值的某个范围时,才认为该点是有效的采样点.

1.4 三维指尖轨迹的跟踪

由于检测出来的二维指尖点以及 Kinect 获得的深度值都存在误差,因此要想稳定地跟踪指尖点的三维位置,必须加上一些稳定的跟踪算法.考虑到在交互系统中手的运动特征,本文利用卡尔曼滤波器来实现;同时,由于 Kinect 的深度值会出现抖动的情况,因此会出现某个指尖是存在的但由于误差原因检测不到该指尖点的情况,导致卡尔曼滤波器频繁的终止与开始,一个频繁终止与开始的滤波器很难达到好的跟踪效果.为了防止卡尔曼滤波器出现这种情况,本文为卡尔曼滤波器引入了帧之间的连续性这一概念.跟踪算法主要分为 3 个步骤:

1) 利用卡尔曼滤波器预测每个指尖的位置

在每一帧中测量指尖点在三维中的位置和速度,定义卡尔曼滤波器的状态向量为

$$\mathbf{x}_t = (x(t), y(t), z(t), v_x(t), v_y(t), v_z(t));$$

其中, $x(t), y(t), z(t)$ 表示指尖在三维中的坐标位置, $v_x(t), v_y(t), v_z(t)$ 表示在每一帧中指尖点的速度.同理,可以定义卡尔曼滤波器的观察向量为

$$\mathbf{y}_t = (x(t), y(t), z(t));$$

即观察向量表示指尖点在每一帧中的三维坐标.根据上面的定义,可以列出卡尔曼系统方程为

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{G}\mathbf{w}_t,$$

$$\mathbf{y}_t = \mathbf{H}\mathbf{x}_t + \mathbf{v}_t;$$

其中, \mathbf{F} 是状态转移矩阵, \mathbf{G} 是驱动矩阵, \mathbf{H} 是观察矩阵, \mathbf{w}_t 是状态向量 \mathbf{x}_t 的系统误差,而 \mathbf{v}_t 是观察误差,也就是测量指尖位置和真实指尖位置的误差,本文算法中表示的就是指尖检测算法和 Kinect 深度检测的误差.

这里需要做如下假设:在 ΔT 很小的情况下,连续两帧之间每个指尖的运动可以近似看做是匀速直线运动.这样, \mathbf{F}, \mathbf{G} 和 \mathbf{H} 定义为

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 & \Delta T & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta T & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta T \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T,$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

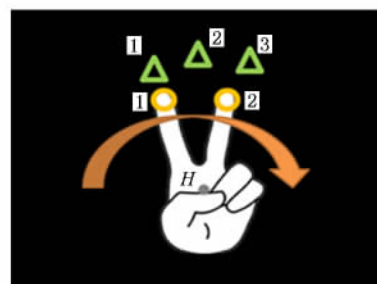
因此,可以利用每个手指的卡尔曼滤波器,通过手指在所有 $t+1$ 帧之前的位置预测它在 $t+1$ 帧时的位置.

2) 预测指尖点和检测指尖点之间的匹配

在当前帧可以利用指尖检测算法检测出当前帧的指尖点;也可以利用卡尔曼滤波器预测出当前帧指尖点所在的位置.一个很重要的问题就是怎样利用当前帧检测出来的指尖点继续卡尔曼滤波器跟踪的过程.本文将预测指尖点位置和检测指尖点位置做组合匹配,找到匹配误差最小的作为最终结果,如图 7a 所示.误差的计算使用指尖点三维位置的欧氏距离,图 7a 也表示由于误差的原因拇指和小指没有检测到,但是由于利用了帧之间的连续性,可以利用卡尔曼滤波器的预测位置来代替指尖的真实位置,因此卡尔曼滤波器可以继续跟踪下去.



a 预测指尖点和检测指尖点的原理



b 指尖点都按顺时针顺序排序

注: Δ 预测指尖位置; \circ 检测指尖位置

图 7 利用卡尔曼滤波器预测指尖点的位置

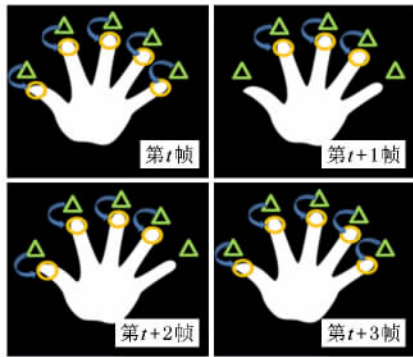
为了减少计算量,同时考虑到操作者很少会将手指进行交叉,本文将检测到的指尖点和对每个指尖建立的卡尔曼滤波器都按以手中心点 H 为中心的顺时针顺序排列,这样在匹配时手指的顺序就是固定的,于是只需搜索 $\circ 1-\Delta 1$ 和 $\circ 2-\Delta 2$; $\circ 1-\Delta 2$ 和 $\circ 2-\Delta 3$; $\circ 1-\Delta 1$ 和 $\circ 2-\Delta 3$ 这 3 组匹配.因此整个搜索空间就从排列级降为组合级,可以较大地减小计算量,如图 7b 所示.

3) 处理指尖数量的变化

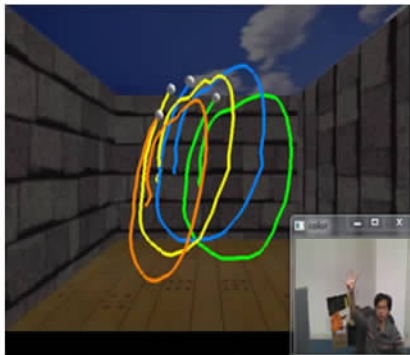
指尖数量的变化有 2 种情况:一是操作者确实

改变了指尖的数量;二是由于误差和运动模糊的原因,在某些帧中某些指尖点检测不到或者误检测到多余的指尖点,如图 7 a 所示.

在第一种情况下,卡尔曼滤波器必须进行相应的变化:当指尖数量增加时,必须开启新的卡尔曼滤波器来跟踪它;当指尖数量减少时,必须终止相应的卡尔曼滤波器.但是对于第二种情况就不能这样处理.为了解决这个问题,必须区分出 2 种指尖数量变化.由于第一种指尖数量变化是一种稳定的变化,因此在多帧之间它将维持这种变化;相反,第二种指尖数量变化只是一种暂时、不稳定的变化,因此在多帧之间它不能维持.根据这个差别,本文在指尖数量开始变化时启动一个计数器,若这个计数器的数值能够大于某个阈值,就认为该变化是第一种情况,否则为第二种情况.本文为数量减少和数量增加各定义了一个计数器,保证了卡尔曼滤波器不会因为指尖的多余检测和漏掉检测而变得不稳定,从而较大幅度地提高了跟踪的稳定性,如图 8 所示.



a 连续帧指尖数量的变化



b 跟踪得到 4 根手指的三维轨迹

图 8 处理手指数量的变化

2 实验分析

本文采用英特尔酷睿 2 双核 2.53 GHz, NVIDIA Quadro600 作为测试系统,跟踪一只手(5 根手指)

的情况下,帧率达到 45 帧/s,跟踪 2 只手(10 根手指)的情况下,帧率达到 20 帧/s. 本文算法中,二维指尖的检测因为是对每个像素单独处理,所以用 GPU 进行了加速.加速之后,每个过程的运行时间所占比例如图 9 所示,可以看出,算法的主要运行时间仍然是基于像素分类的指尖检测,大约需要 20 ms. 其他过程只需要 1~2 ms.

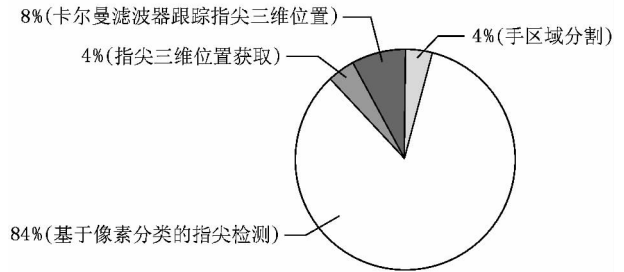


图 9 三维指尖跟踪算法各部分时间分布图

2.1 二维指尖检测算法的准确性分析和比较

本文提出的基于像素分类的指尖检测算法更加鲁棒.基于形状特征的方法最早是由 Hardenberg 等^[21]提出来的,之后 Song 等^[22]和 Letessier 等^[20]对该方法做了改进.本文通过实验来比较本文算法和文献[20-22]3 种算法的正确率:随机采样 180 幅手的二值图像(采样过程中不断变换手的姿态以及在 Kinect 的有效识别范围内改变手离 Kinect 的距离),然后对每幅图像分别运行 4 种算法,最后统计每种算法识别的正确性,比较结果如表 1 所示.可以看出,本文算法的正确率达到 94.4%,比文献[20]的 83.3%和文献[22]的 88.9%都高.对于每一幅图像运行指尖检测算法一般有检测正确、少检测和多检测 3 个结果.

表 1 指尖检测算法正确率比较

	结果	180 帧	平均准确率/%
文献[21]方法	检测正确	83	46.1
	多检测	97	
	少检测	0	
文献[22]方法	检测正确	160	88.9
	多检测	0	
	少检测	20	
文献[20]方法	检测正确	150	83.3
	多检测	17	
	少检测	13	
本文算法	检测正确	170	94.4
	多检测	6	
	少检测	4	

2.2 三维指尖位置跟踪算法的实验分析

2.2.1 准确性和误差分析

由于 Kinect 能够比较准确地跟踪一个手点的位置,而且每个手指尖与手点的相对位置在运动过程中是基本不变的,因此可以把手点的运动轨迹看做是某个指尖点的运动轨迹加上一个偏移向量,这样就能够用手点的运动轨迹近似表示每个手指点的真实运动轨迹。

下面分别比较用本文算法得到的运动轨迹(如图 10 a 中的细线)、不采用本文算法得到的运动轨迹(如图 10 b 中的细线)与真实运动轨迹(如图 10 a, 10 b 中的粗线)之间的误差,由此来评判方法的有效性。在实验中,实验者首先用一根手指运动一段时间(10 s),然后用 2 根手指、3 根手指、4 根手指、5 根手指各运动相同的时间,再分别对每个手指尖的运动轨迹和手点的运动轨迹进行比较。实验采集 2 组数据:一组采用本文算法,一组不采用,然后对轨迹误差做比较。实验结果如图 11 a 所示,图中横、纵坐标分别表示手指数量与误差大小。误差的计算公式如下:假设 $h(t)=(h(x),h(y),h(z)),f(t)=(f(x),f(y),f(z))$,其中 $h(t)$ 表示第 t 帧中手点的位置, $f(t)$ 表示第 t 帧中某个指尖点的位置,则手点和当前指尖点的偏移向量

$$\lambda = (\sum_{t=1}^n (h(t) - f(t))) / n,$$

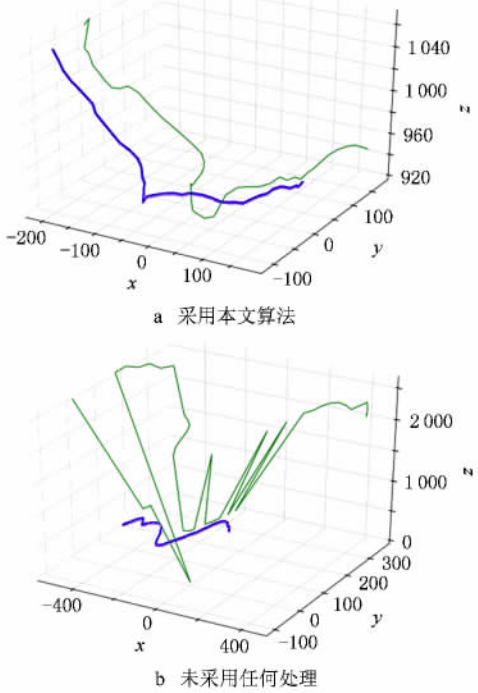


图 10 轨迹的准确性比较

n 表示帧的数量;最后可以得出误差

$$\delta = \sum_{t=1}^n (f(t) + \lambda - h(t)) \times (f(t) + \lambda - h(t)) / n.$$

根据上面的实验数据,可以进一步计算出未采用本文算法得到的误差 δ_{raw} 和使用本文算法后得到的误差 δ 的比值

$$\Theta = \frac{\delta_{\text{raw}}}{\delta};$$

用 Θ 值的大小来评价算法的有效性,如图 11 b 所示。可以看到,当手指数量为 4 时效果最差,这可能跟实验者不习惯伸出 4 根手指的动作有关。

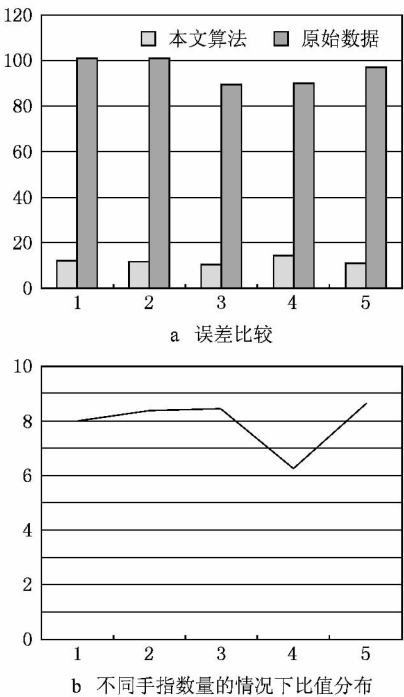
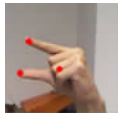
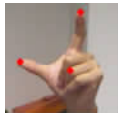


图 11 误差比较及手指个数对算法的影响

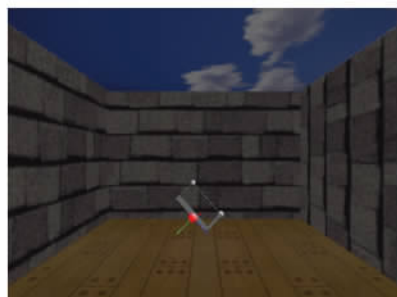
2.2.2 鲁棒性分析

为了分析本文算法的鲁棒性,我们设计了一个应用:在三维空间中有若干个积木,它们的方向被随机扰动过(如图 12 a 所示),要求用户将这些积木的方向修改正确。针对这个应用,本文定义了抓取和释放 2 个手势,如表 2 所示。

表 2 2 个手势的定义及其自由度		
手势	位姿	跟踪的自由度
抓取		位姿(6 个自由度)+2 个指尖之间的距离
释放		位姿(6 个自由度)+2 个指尖之间的距离



a 随机旋转过的积木场景



b 手中心点和另外2个指尖点构成一个局部坐标系

图 12 空中操作积木

抓取手势定义为:三维空间中存在 2 个指尖且两者之间的距离小于某个阈值.释放手势定义为:三维空间中存在 2 个指尖且两者之间的距离大于某个阈值.2 个三维指尖点和一个三维手中心点刚好可以定义三维中的一个局部坐标系即手的位姿,如图 12 b 所示,红色球体表示的手中心点和灰色球体表示的另外 2 个指尖点刚好在手中心处构成一个局部坐标系,其中蓝色直线为 x 轴,红色直线为 y 轴,绿色直线为 z 轴;在用户抓取到物体之后,用手的位姿去修改积木的方向,这样被随机旋转过的积木就能被用户摆放正确.

本文从用户抓取物体时开始统计,直到将所有物体都摆放正确.在这个过程中,设所有帧数为 F_t ,算法检测错误的帧数为 F_e ,则算法的正确率 $\mu = 1 - F_e/F_t$,对 5 个用户进行实验的结果如表 3 所示.

表 3 抓取实验结果

用户	F_t	F_e	正确率/%
1	300	5	98.3
2	320	10	96.9
3	350	12	96.6
4	330	10	97.0
5	380	15	96.0

表 3 中,由于用户 1 是作者本人,因此操作比较熟练,而且错误的帧数也比较少;其他人都是在练习

几次之后进行操作的,正确率基本保持在 96%~97%左右.

尽管如此,由于 Kinect 本身的限制,算法要求手离 Kinect 的距离必须在一定范围之内(50 cm~1.5 m),否则结果会很不准确.手距离 Kinect 小于 50 cm 之后, Kinect 将得不到手的深度值,因此三维位置无法获取;距离 Kinect 大于 1.5 m 之后,由于使用的深度图分辨率为 640×480 ,因此,手指区域的大小会特别小(1~2 像素), g_1 的取值会为负值,算法失效.同时,因为本文只用了一个 Kinect,因此当手指出现自遮挡时算法也会出现问题.当然,由于本文算法使用了三维的卡尔曼滤波器,因此能够处理短时间的自遮挡.

3 结 语

由实验分析可知,采用本文算法不仅能达到实时的要求,而且指尖点的三维位置和轨迹也非常稳定,不会出现抖动特别厉害的现象,为稳定、友好的三维交互提供了基础.不过,由于 Kinect 自身的限制,算法只有在手距离 Kinect 一定范围之内时有效;同时,本文算法只能够处理短时间的自遮挡.

Kinect 除了能给出深度值外,还能给出 RGB 图像,且其 RGB 图像的分辨率能达到 (1280×1024) ,接下来的工作就是考虑利用高分辨率 RGB 图像解决距离超过 1.5 m 时算法失效的情况.同时,因为现在已经能稳定地跟踪到三维指尖的位置,今后也要考虑利用这些结构信息设计新的手势识别算法,以识别更多的手势.

致谢 感谢何辰、陈晨阳同学在本文实验中给予的帮助!

参考文献(References):

- [1] Vezhnevets V, Andreeva A. A survey on pixel-based skin color detection techniques [C] // Proceedings of the 13th International Conference of Computer Graphics and Visualization. Los Alamitos: IEEE Computer Society Press, 2003: 85-92
 - [2] Chen Duansheng, Liu Zhengkai. A survey of skin color detection [J]. Chinese Journal of Computers, 2006, 29(2): 194-207 (in Chinese)
- (陈锻生, 刘政凯. 肤色检测技术综述 [J]. 计算机学报, 2006, 29(2): 194-207)

- [3] Cao Xinyan, Zhao Jiyin, Li Min. Monocular vision gesture segmentation based on skin color and motion detection [J]. Journal of Hunan University: Natural Sciences, 2011, 38(1): 78-83 (in Chinese)
(曹昕燕, 赵继印, 李 敏. 基于肤色和运动检测技术的单目视觉手势分割[J]. 湖南大学学报: 自然科学版, 2011, 38(1): 78-83)
- [4] Oka K, Sato Y, Koike H. Real-time fingertip tracking and gesture recognition [J]. IEEE Computer Graphics and Applications, 2002, 22(6): 64-71
- [5] Oka K, Sato Y, Koike H, *et al.* Real-time tracking of multiple fingertips and gesture recognition for augmented desk interface systems [C] //Proceedings of the 5th IEEE International Conference on Automatic Face and Gesture Recognition. Los Alamitos: IEEE Computer Society Press, 2002: 429-434
- [6] Sato Y, Kobayashi Y, Koike H. Fast tracking of hands and fingertips in infrared images for augmented desk interface [C] //Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition. Los Alamitos: IEEE Computer Society Press, 2000: 462-467
- [7] Crowley J L, Berard F, Coutaz J. Finger tracking as an input device for augmented reality [C] //Proceedings of International Workshop on Automatic Face and Gesture Recognition. Los Alamitos: IEEE Computer Society Press, 1995: 195-200
- [8] Keaton T, Dominguez S M, Sayed A H. SNAP&TELL: a multi-modal wearable computer interface for browsing the environment [C] //Proceedings of the 6th International Symposium on Wearable Computers. Los Alamitos: IEEE Computer Society Press, 2002: 75-82
- [9] Quek F, Mysliwiec T, Zhao M. Finger mouse: a free hand pointing computer interface [C] //Proceedings of International Workshop on Automatic Face and Gesture Recognition. Los Alamitos: IEEE Computer Society Press, 1995: 372-377
- [10] Tomita A, Ishii R. Hand shape extraction from a sequence of digitized grayscale images [C] //Proceedings of the 12th International Conference on Industrial Electronics, Control and Instrumentation. Los Alamitos: IEEE Computer Society Press, 1994: 1925-1930
- [11] Yang D D, Jin L W, Yin J X, *et al.* An effective robust fingertip detection method for finger writing character recognition system [C] //Proceedings of International Conference on Machine Learning and Cybernetics. Los Alamitos: IEEE Computer Society Press, 2005: 4991-4996
- [12] Pan Z G, Li Y, Zhang M M, *et al.* A real-time multi-cue hand tracking algorithm based on computer vision [C] //Proceedings of the IEEE Virtual Reality Annual International Symposium. Los Alamitos: IEEE Computer Society Press, 2010: 219-222
- [13] Nguyen D D, Pham T C, Jeon J W, *et al.* Fingertip detection with morphology and geometric calculation [C] //Proceedings of International Conference on Intelligent Robots and Systems. Los Alamitos: IEEE Computer Society Press, 2009: 1460-1465
- [14] Schlattmann M, Kahlesz F, Sarlette R, *et al.* Markerless 4 gestures 6 DOF real-time visual tracking of the human hand with automatic initialization [J]. Computer Graphics Forum, 2007, 26(3): 467-476
- [15] Shi J H, Zhang M M, Pan Z G. A real-time bimanual 3D interaction method based on bare-hand tracking [C] //Proceedings of the 19th ACM International Conference on Multimedia. New York: ACM Press, 2011: 1073-1076
- [16] Guo Kangde, Zhang Mingmin, Sun Chao, *et al.* 3D fingertip tracking algorithm based on computer vision [J]. Journal of Computer Research and Development, 2010, 47(6): 1013-1019 (in Chinese)
(郭康德, 张明敏, 孙 超, 等. 基于视觉技术的三维指尖跟踪算法[J]. 计算机研究与发展, 2010, 47(6): 1013-1019)
- [17] Raheja J L, Chaudhary A, Singal K. Tracking of fingertips and centers of palm using Kinect [C] //Proceedings of the 3rd International Conference on Computational Intelligence, Modelling and Simulation. Los Alamitos: IEEE Computer Society Press, 2011: 248-252
- [18] Ren Z, Yuan J S, Zhang Z Y. Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera [C] //Proceedings of ACM Multimedia Conference and Co-located Workshops. New York: ACM Press, 2011: 1093-1096
- [19] Oikonomidis I, Kyriazis N, Argyros A A. Efficient model-based 3D tracking of hand articulations using Kinect [C] //Proceedings of the British Machine Vision Conference. Manchester: BMVA Press, 2011: 101.1-101.11
- [20] Letessier J, Bérard F. Visual tracking of bare fingers for interactive surfaces [C] //Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology. New York: ACM Press, 2004: 119-122
- [21] Hardenberg C V, Bérard F. Bare-hand human computer interaction [C] //Proceedings of Workshop on Perceptive User Interfaces. New York: ACM Press, 2001: 1-8
- [22] Song P, Yu H, Winkler S. Vision-based 3D finger interactions for mixed reality games with physics simulation [C] //Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH. New York: ACM Press, 2008: Article No. 7