XKin - eXtendable Hand Pose and Gesture Recognition Library for Kinect

Fabrizio Pedersoli, Stud.
pedersoli.fabrizio@gmail.com

Nicola Adami, PhD.
nicola.adami@ing.unibs.it

Sergio Benini, PhD.
sergio.benini@ing.unibs.it

Riccardo Leonardi, Prof.
riccardo.leonardi@ing.unibs.it

Department of Information Engineering
University of Brescia
Brescia, Italy

ABSTRACT

In this work we provide an open-source framework for Kinect enabling more natural and intuitive hand-gesture communication between human and computer devices. The software package is endowed with useful tools for training the system to work with user-defined postures and gestures. The XKin project is fully implemented in C and freely available at `https://github.com/fpeder/XKin` under FreeBSD License. Our goal is to encourage contributions from other researchers and developers in building an open and effective system for empowering a natural modality for human-machine interaction.

Categories and Subject Descriptors

I.4.8 [**Scene Analysis**]: Object recognition; I.5.4 [**Pattern Recognition**]: Computer Vision

Keywords

Kinect, Computer Vision, Gesture Recognition

1. INTRODUCTION

During the last years, research in Human Computer Interaction (HCI) has introduced innovative technologies that empower users to interact with computers in increasingly natural and intuitive ways. The main objective of these studies is to enable human-machine interaction protocols which resembles human-human communication. In this sense, gesture recognition is an extremely important element for building advanced HCI systems. The ability to automatically understand, not only hand gesture specifically defined by communication protocols, but also hand and body natural movements, will open to a completely new set of applications.

MM'12, October 29–November 2, 2012, Nara, Japan.

Recently a great interest has been devoted to gesture recognition especially in the field of video-games. The introduction of new sensor devices, such as the Kinect originally designed for the Microsoft Xbox, has allowed to overcome the limitations of computer-vision classical 2D vision algorithms [11], with particular regards to the difficulties of segmenting objects in complex scenes and automatically recognizing hand gestures. This low cost device consisting of one infrared projector, one infrared camera and one regular camera, provides two video streams: depth and color, respectively. Together with the device, Microsoft released a Software Development Kit (SDK) [3], which is mainly oriented to skeleton tracking rather than hand movement recognition. Beyond being limited to Windows environment, this package is not open source.

As an alternative multi-platform solution, PrimeSense [6] has created NiTE [4], a middleware which provides almost the same features as the Microsoft SDK, but again constitutes a closed source software. In addition to this, NiTE is not a complete solution: in order to work it first needs to be included in OpenNI, an open source framework which provides Application Programming Interfaces (APIs) for natural interaction applications. Then NiTE also requires ad-hoc drivers to communicate with other devices; since PrimeSense produces also hardware, drivers are usually released by the same company as open source software.

Despite these proprietary high-level solutions, the open source community has been recently developing a low-level module for acquiring raw data stream from the Kinect: libfreenect [2]. Libfreenect is developed by the OpenKinect community and exclusively targets Kinect hardware. It represents a simple, clean and easy to use solution that provides the possibility to use the device with PCs running either GNU/Linux, MacOS or Windows. It comes also with many wrappers towards programming languages such as: C++, C#, Python, Java, Matlab and Ruby.

Embracing the open source philosophy, in this work we provide an open source package for hand gesture recognition using Kinect sensor. By acquiring raw Kinect data streams via libfreenect we offer a reliable and real-time solution to hand gesture recognition. The developed Application Program Interfaces (APIs) allow for building up intuitive and personalized applications based on hand gesture interaction.

As an add-on, the employed solution for gesture recognition does not encumber the user with the need to use addi-

tional devices such as for example, wearing gloves, bracelets or special clothes, thus respecting the "come as you are" paradigm of Human Computer Interaction.

The developed libraries are part of a public project that can be downloaded from github [7]. We encourage ad support not only open source software, but also an open source way to conduct research: we believe that providing the possibility to share techniques and cooperating in improving performance in one effective way of achieving challenging goals.

The rest of the paper is organized as follows. Section 2 supplies an overview of the developed system. Section 3 provides a detailed description of the principal components of the package, its architecture and also illustrates how to use it for developing application. Section 4 presents some scenarios where the proposed technologies can provide effective solutions. Finally in Section 5 conclusions are gathered.

2. OVERVIEW

The package hereafter presented is written in plain C. Figure 1 depicts the package structure, the relationships between its components, and architectural dependencies with other libraries.

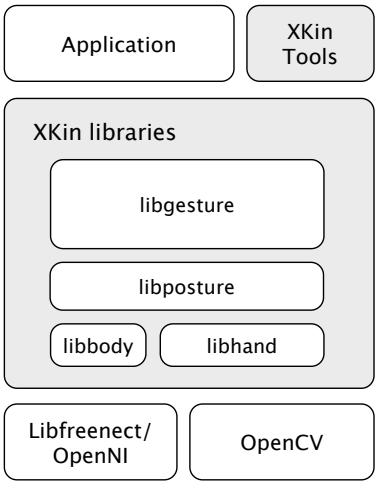

Figure 1: XKin libraries and tools: architecture.

The package is built on top of libfreenect; however it is also possible to smoothly replace the low-level interface with another one such as, for example, the OpenNI/PrimeSense. Concerning related signal processing, we adopted OpenCV libraries [5] to handle digital matrices, images and videos. Clearly the image processing technology underlying the here proposed framework represents one of the main innovations of this work. However the description of these techniques is outside the scope of this document and will be object of other incoming publications.

XKin package is composed of four main libraries: `libbody`, `libhand`, `libposture` and `libgesture`.

`Libbody` and `libhand` are related to the preprocessing stage which is structured in two phases: body detection and hand detection. Through the joint use of `libbody` and `libhand` it is possible to extract hand contours in a robust and fast manner. While `libbody` provides more general functionalities, `libhand` includes highly specialized hand features extraction and analysis. The rationale behind the separation of hand and body detection is not only architectural but it reflects the actual hand detection chain which is performed in two steps, as shown in Figure 2.

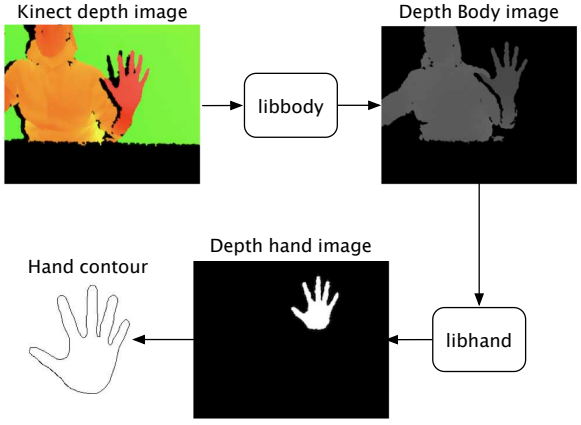

Figure 2: Two step procedure for hand contour detection.

Libraries `libposture` and `libgesture` deal with the proper hand posture and gesture recognition. In particular with *hand posture* we intend a form of non-verbal communication in which a static pose of the hand communicates a particular message, such as numbers or other conventional messages. Example of different postures are given in Figure 3.

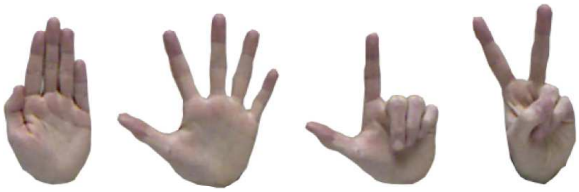

Figure 3: Examples of different postures.

Hand gestures instead include visible hand movements. In the current implementation, in order to enable a reliable and fast detection, a gesture is meant to be composed by three different phases: an initial posture, which is followed by a hand trajectory, and a final posture, as shown in Figure 4.

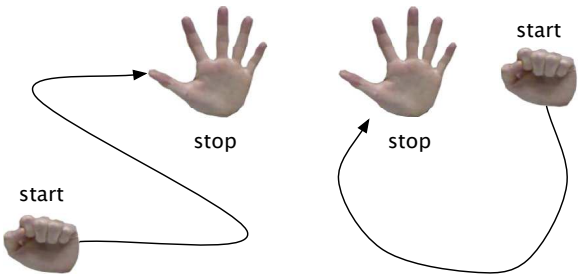

Figure 4: Examples of different gestures.

Nevertheless this choice provides a robust detection phase, an ongoing future extension of our framework is considering the inclusion of a less constrained gesture definition, so that to enable the recognition of more natural hand interactions.

Finally, on top of the XKin libraries we provide some additional support tools for gesture training and recognition (XKin tools).

The design of all libraries and tools has been done targeting speed, modularity and expandability. The recognition speed is a crucial aspect since video stream provided by Kinect is normally at 30fps. Hence it is very important to perform all recognition tasks in a small amount of time in order not to jeopardize real time use feeling.

The code behind these APIs is structured in a strong modular way. The possibility to easily modify any procedure of interest is a key factor since hand gesture recognition is nowadays an actively evolving research field. Therefore the possibility of adding newly designed features is ensured. As an example of future extension, the architectural design allows for easily addition of other specialized libraries dealing, for example, with head and legs movements.

3. DESCRIPTION

This section illustrates the API provided in XKin 1.0, describing the main functions of the different libraries.

3.1 XKin Libraries

libbody

This library has the purpose to detect the body of a person in the acquired depth scene. The most important function is `body_detection` which processes the depth image to isolate the area corresponding to the body and setting to zero all pixels belonging to the background.

libhand

This library handles hand segmentation. It provides a function for hand detection and one for hand contour extraction. With `hand_detection` it is possible to obtain a binary mask representing the hand shape and the corresponding depth image.

Rough hand contours can be extracted by using the `get_hand_contour_basic` which processes only the depth image corresponding to the hand. Beside, `get_hand_contour_advanced` uses a more sophisticated approach to achieve a precise contour definition of the hand. In this case, the depth image is used as initial information to quickly locate the hand in the color image. The correspondences between pixels in the color image and pixels in the depth image can be determined thanks to the knowledge of the relative position of the associated camera models. The provided functions do make use of the intrinsic model parameters provided by [1]; however they can be alternatively estimated through camera calibration procedures.

libposture

This library supplies functions to classify postures, that means static hand poses. We developed two types of posture classification methods. The basic one is intended to be extremely fast and robust limiting the classification to only two postures: open hand and closed hand. A more advanced classification procedure distinguishes between a wider set of postures even if, it is less robust if compared to the previous technique. The first type of classification is performed by `basic_posture_classification` function, which takes as input the basic hand contour and makes a recognition based on the evaluation of the convex hull and of the convexity defects of the contour.

The `advanced_posture_classification` function uses the advanced contour and classifies based on the minimum distance between Fourier shape descriptors [9] that can be calculated by the `get_fourier_descriptors` function. The classification provided by both (basic and advanced) posture functions, is related not only to the current frame, but is determined as the most recurrent posture within a classification time window. This procedure is employed to make the classification more stable in time and less affected from noise.

libgesture

This library provides the classification of hand trajectory movement. It implements a flexible structure for handling Hidden Markov Models (HMMs) [10] and the typical algorithms for models training and testing, such as forward, backward and the Baum-Welch. This library also provides a simple interface for storing and loading HMM models through yaml [8] files.

The defined functions `cvhmm_get_gesture_sequence` and `cvhmm_classify_gesture` have a key role. The first is used to segment the gesture sequence by analyzing the video stream. A state machine is used to extract the hand trajectory by detecting the hand posture associated to the beginning and to the end of a gesture sequence. The second function compares the retrieved trajectory with all HMMs used to model different gestures according to the maximum-likelihood criterion.

Other relevant functions are: `cvhmm_loglik`, that given an HMM and an observation vector computes the log-likelihood of that sequence; `cvhmm_reestimate` which allows for adjusting the HMM parameters, according to expectation maximization (EM) algorithm with respect to an observation sequence. Storing and loading operations of HMM models are handled by `cvhmm_write` and `cvhmm_read`. The first function writes on a text file an array of arbitrary length of HMMs using the yaml standard, while the second one preforms the opposite operations. This library also implements an interface for efficiently handling sequences of points. It optimizes memory management and eases insert/removal operations.

3.2 XKin Tools

On top of XKin libraries we also provide some useful tools to facilitate training and testing processes, both for postures and gestures. These are mainly provided as command line tools that can also interact through graphic extensions with the user.

The `trainposture` program creates a corresponding model for each posture the user wants to insert in the system. The user must specifies the total number of postures and the number of samples that will be provided. Training can be quickly realized by the user that reproduces the desired set of postures directly in front of the Kinect. The procedures produces a `yaml` file, which includes all the information needed in the classification stage. `testposture` takes in input the training file and allows for testing the previously learned postures by providing a drawing of the hand with different colors.

Contrarily to the definition of new postures, the input of new gestures does not require the user to directly interact with the Kinect. In this case the used approach requires to

input the sequence of points describing the movement with the mouse in a dedicated graphical window. In fact, while for the posture detection the knowledge of the real shape is very important, for gesture classification the relevant information is in the trajectory.

More specifically, `genproto` allows for the generation of gesture prototype sequences by writing them into a yaml file; Before inserting them the user must specify the number of states of the correspondent HMM which is equal to the number of piecewise segments in the trajectory. Once all the prototypes are produced, the `trainmodels` program estimates the best HMM parameters for each given gesture. This is done automatically because we represent the trajectory in a way that it results invariant to translation and scaling.

Concerning classification, with `testmodels` it is possible to test HMM gesture models in a graphical window where sequences of points to be classified are drawn. Finally by means of `testgesture` program it is possible to test the trained model in the "online" operational mode, according to which the user performs the desired gesture in front of the Kinect, while the classification results are represented by drawing the gesture sequence with different colors in a dedicated window.

3.3 How to use

Hand gesture recognition is a difficult problem, especially when one embraces the "come as you are" paradigm. Therefore to obtain a well working solution the usage environment must follow some not too restrictive guidelines. The user must be the closest object to the Kinect: this means that no other thing should interpose between him/her and the device. Finally, in particular with respect to the posture recognition, background should be not too complex in order to reduce false detections.

Once met these requirements, creating a gesture/posture based recognition application with XKin is simple and straightforward. Our library function calls are placed in a "infinite loop", and at each iteration depth and color images are retrieved from the Kinect by using `freenect_sync_get_depth_cv` and `freenect_sync_get_rgb_cv` respectively. Once the raw data are available the first step performs hand detection, by calling `body_detection` and `hand_detection`.

At this point, depending on the users needs, it is possible to use posture or gesture oriented functions. In the case of generic posture classification the calls are to `get_hand_contour_advanced` and to `advance_posture_classification` functions. For a gesture oriented application instead the first task requires the identification of the start and stop postures which is done entirely considering the depth image by using `get_hand_contour_basic` and `basic_posture_classification`. The sequence of point between start and stop posture is then classified by `cvhmm_classify_gesture`.

4. APPLICATIONS

Since gestures are the most primary and expressive form of human communication, the provided package for gesture-based interface could radically change the way to interact with computer systems.

This package could fit in a large set of real scenarios such as, for example, medical applications, domotics, and videogames. For example, in medicine, a gesture based application can be convenient to the surgeon for browsing tomographic contents or moving particular support instruments, such as wearable haptics or robots. In the domestic environment gesture recognition can be handy for removing remote controllers, such as television ones, or to communicate with other kind of devices like lights, shutters etc. In the game industry, gesture recognition nowadays is one of the main interest: great efforts has been addressed to the development of natural interfaces starting from controlled based solutions, such as Nintendo's Wii towards a completely free interaction such as Microsoft it is trying to do with Kinect. In addition to these, a gesture based interaction system could boost users experience in all computer applications that suffer from the limitation imposed by traditional mouse and keyboard-based interaction, like 3D modeling and CAD/CAM.

5. CONCLUSION

We developed an open source package for hand posture and gesture recognition that uses low cost Kinect sensor device. This package represents a novel solution in the open source community where there was a lack of these opportunities. Moreover this package targets a particular kind of gesture recognition: the hand one, which is not even provided by proprietary closed source solutions.

Despite this project is still in an young development stage, it shows already good performance of recognition and a simple but powerful interface that enables users to create gesture-based application in an easy way. Our plan is to enlarge the set of open source API that enables the development of natural interaction based applications.

6. REFERENCES

[1] Kinect calibration. `http://nicolas.burrus.name/index.php/Research/KinectCalibration`.

[2] Libfreenect. `http://openkinect.org/wiki/Main_Page`.

[3] Microsoft Kinect for Windows. `http://www.microsoft.com/en-us/kinectforwindows`.

[4] NiTE. `http://www.primesense.com/nite`.

[5] OpenCV. `http://opencv.willowgarage.com/wiki/`.

[6] PrimeSense. `http://www.primesense.com`.

[7] XKin. `https://fpeder@github.com/fpeder/XKin.git`.

[8] YAML. `http://yaml.org/`.

[9] S. Conseil, S. Bourennane, and L. Martin. Comparison of Fourier descriptors and Hu moments for hand posture recognition. In *European Signal Processing Conference (EUSIPCO)*, 2007.

[10] L. R. Rabiner and B. H. Juang. An introduction to hidden markov models. *IEEE ASSp Magazine*, 1986.

[11] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan. Vision-based hand-gesture applications. *Commun. ACM*, 54(2):60–71, Feb. 2011.