USER GUIDE

A] Install kinect on PC linux
B]Fall detection algorithm
C]Program explanation

A] Install kinect on PC linux

To make kinect run on a linux PC , you should install three frameworks on your PC:
OpenNI: The OpenNI framework provides a set of open source APIs. These APIs are intended to become a standard for applications to access kinect device.

SensorKinect: SensorKinect is an OpenNI module that allows it to interact with the Kinect. It is the hardware driver.

Nite: NIte is a middleware provide hand-based control and a full-body control capacity, it has functions such as hand locating and tracking; a scene analyzer ; accurate user skeleton joint tracking; various gestures recognition; and more.

1 - Install OpenNI

1.Active terminal, and install some dependencies
sudo apt-get install git-core g++ python libusb-1.0-0-dev freeglut3-dev openjdk-6-jdk doxygen graphviz build-essential

2.Make a directory to store the build, then clone the OpenNI source from Github.
sduo mkdir ~/kinect
cd ~/kinect
sudo git clone https://github.com/OpenNI/OpenNI.git

3.Then run the installation script
cd ~/kinect/OpenNI/Platform/Linux/CreateRedist
sudo ./RedistMaker

cd ~/kinect/OpenNI/Platform/Linux/Redist/OpenNI-Bin-Dev-Linux-x64-v1.5.4.0
sudo ./install.sh

2 - Install SensorKinect

1.Clone the SensorKinect source from Github.

cd ~/kinect

git clone http://github.com/avin2/SensorKinect.git

2.Then run the installation script

cd ~/kiect/SensorKinect/Platform/Linux/CreateRedist/

sudo ./RedistMaker

cd ~/kinect/SensorKinect/Platform/Linux/Redist/Sensor-Bin-Linux-x64-v5.1.2.1

sudo ./install.sh

3 - Install NIte

1.Download the Nite 1.5 from this URL:

http://www.openni.org/openni-sdk/openni-sdk-history-2/#.UfEQeY2Nnms%22%20target=%22_blank%22

2.Extract the contents of the archive and switch to the Data directory contained within.

cd ~/kinect

tar -xvjpf nite-bin-linux-x64-v1.5.2.21.tar.bz2

cd NITE-Bin-Dev-Linux-x64-v1.5.2.21/Data

3.Now modify the license in the files: Sample-Scene.xml, Sample-Tracking.xml, and Sample-User.xml.

Change

<License vendor=”PrimeSense” key=””/>

to:

<License vendor=”PrimeSense” key=”**0KOIk2JeIBYClPWVnMoRKn5cdY4=**”/>

4. Change back to the NITE directory and run the install script.

cd ..

sudo ./install.sh

B]Fall detection algorithm

Our fall detection program apply the 3D bounding box fall detection algorithm. In this section I will explain the Principle of this algorithm, and some modification we made in our programm.

Fall detection algorithm(original version)

```
threshold  Tv_H , Tv_WD
threshold  Ti_H
counter    a, b=0
counter    falling =N (frames)
counter    inactivity =M (frames)
boolean    activityDetection=false
boolean    inactivityDetection=false
boolean    fallDetected=false

While  run  do
    if  v_H  > Tv_H  and v_WD > Tv_WD then
          if a=falling then
             set activityDetection = true
           end if
       set a++
     end if

  if activityDetection = true and abs(v_H ) < Ti_H then
        if b = inactitivity then
            set inacitivityDetection = true
```

```
            set fallDetected= true
        end if
      set b++
   end if
end while
```

1 - Fall detection algorithm analysis

threshold Tv_H , Tv_WD : these are two thresholds of velocity of height and depth-width, they aim to provide two conditions to detect a motion or activity like falling(reference to the first 'IF')

threshold Ti_H : this is threshold of velocity of height, it aim to provide a condition to detect the inactivity after a fall (reference to second 'IF')

counter a , b : because of a fall and an inactivity after falling are the motion progress, not in instant, that means a fall will happen in several frames. So we need these two counters to record it. Or in other way we could think these are two timers, they record the time of a fall and inactivity.

threshold falling =N (frames) : that's the threshold about time of a fall, in our case , we present it in frame. if the 'counter a' exceed the threshold falling , we consider a motion like falling is detected; if not we consider that's some other motion (maybe sit , jump etc)

threshold inactivity =M (frames): the same way to detect the inactivity after the fall.

boolean activityDetection : present a motion like falling is detected.
boolean inactivityDetection : present a inactivity after falling is detected.
boolean fallDetected : a detection of fall is confirmed if two above booleans are true.

To understand the principle of this algorithm, first of all, we need know how a fall happen. A fall we can decompose in two phases : a motion or activity like falling , and second phase

of inactivity. Only these two conditions are both satisfied, that we consider a real fall is detected.

So in the fall detection algorithm we need to test these two conditions.The first 'IF' is aim to detect a motion like falling is presented. So if in a instant or more specific in a frame , the velocity of height and the velocity of depth-width are both exceed their threshold, we suppose the following motion maybe a fall, and we increment the 'counter a' which record the number of frame of a fall, afterwards we do the same thing. When the 'counter a' arrive the condition falling (N frame), we can say an activity of fall is detected , then we need test the whether there is an inactivity follow it.

So in the second phase, we test the inactivity. In the same way, if it inactive in M frames (condition of inactivity), we consider that's an inactivity.

Now if the two above condition are checked, the fall is detected.

2 - Modification of the fall detection algorithm

After analysis the fall detection algorithm and the experience in testing the program, I found some problems in the fall detection algorithm. In fact, there are two main modification I made in the program.

The first problem is in the first 'IF' to test a motion like falling , which compare the velocity of height and velocity of depth-width to their thresholds. I think the first comparison about height is not correct. Because in their article, they say the v_H= ((Hi) - (Hi-1))/((Ti) - (Ti-1)), and they define the threshold of velocity of height is 1.18 m/s ,and their condition to detect a motion like falling is **v_H > Tv_H .** That means if we want to detect a fall (satisfy the condition **v_H > Tv_H**) the value of Hi must be superior the value of H(i-1), the height must increase, that's a motion like jump , but not a fall, that's illogical. I propose we should compare the absolute of velocity to its threshold , like **abs(v_H) > Tv_H.**

The second problem is about the 'counter a'. For the 'counter a' there is no manner to reinitialize it when it detect a motion not a fall but the 'counter a' had changed. For example, if it detect a sit, the 'counter a' should increment. And in next detection, the beginning value of 'counter a' is not 0. That means the threshold falling we defined before lost its part. That may cause a wrong detection of falling. A simple example, if we just repeat sitting and standing, each time counter a will increment a little, it should arrive

threshold falling which is equal 8 frames, and then the wrong detection of falling would happen.

So what I propose is add a manner to reinitialize ‘counter a’. The principle is write below, in each loop we compare the counter a and its value before, if they are equal, we should increment the counter_same_a; and then if it arrive 8 (counter a never change in 8 frames) we can reinitialize ‘counter a’ .

```
if(old_a == a ){
      counter_same_a ++;
 }
 if (counter_same_a == 8){
       a=0;
      counter_same_a =0;
 }
   old_a=a ;
```

Then I will verify it in different case to check this section will not affect to the fall detection :

case 1: when it detect nothing and ‘counter a’ always equal zero. this section of code will have no effect.

case 2: when it detect a motion not a fall , for example it detect a motion like siting. And maybe counter change to 5, and then it rest in this value 5. So after a 8 loops it never change we can say that’s not a fall and we need reinitialize ‘counter a’ for avoid wrong detection afterwords.

case 3: when it detect a motion like falling, in this case counter a should equal 8 or value superior, and then we will set boolean activityDetection and entry to next section (the second ‘IF’), even if we reinitialize ‘counter a’ after then, there is no influence to the whole algorithm.

Fall detection algorithm(modified version)

```
threshold  Tv_H , Tv_WD
threshold  Ti_H
counter    a, b=0
counter    falling =N (frames)
counter    inactivity =M (frames)
boolean    activityDetection=false
boolean    inactivityDetection=false
boolean    fallDetected=false
threshold  Tcounter_same_a

While  run  do
    if  abs(v_H) > Tv_H  and v_WD > Tv_WD then
          if a=falling then
             set activityDetection = true
           end if
       set a++
     end if

  if activityDetection = true and abs(v_H ) < Ti_H then
        if b = inactitivity then
            set inacitivityDetection = true
            set fallDetected= true
        end if
     set b++
  end if

  If old_a == a then
       counter_same_a ++;
  end if

  if counter_same_a == Tcounter_same_a then
       a=0;
      counter_same_a =0;
  end if

  old_a=a ;
```

end while

The algorithm above is the modified version and the version i implement in the program. The author of the 3D bounding box fall detection algorithm propose a set of thresholds they used, and i just add Tcounter_same_a equal 8.

```
#define TvH 1.18
#define TvWD 1.20
#define TiH 0.1
#define falling 8
#define inactivity 10
#define Tcounter_same_a 8
```

NOTE: this set of threshold are not unique and definite , these can be changed in the father test.

C]Program explanation

The program fall detection has two parts, one part for fall detection, another for drawing scene.
For the first part I named it **falldetecion**, it modified from the sample NiSimplesketlon, and the second part **drawScene** I used the sample sceneDrawer .

The fall detection part, it inspire from the sample NiSimpleSkeleton. The program need first read an pre-config xml file, and then it will create two generator :Usergenerator and

Depthgenerator. After that it will try to track user, if it find user and finish calibration , we can start our fall detection.

The fall detection process three steps: first get all of joint point did by function **get getJointInfo()**, the second calcul some parameter like velocity of height , velocity of depth-width, did by function **get_XYZ(),** and last step is apply the fall detection algorithm to detect fall - **detectFalling().**

The second part of the program is draw scene, that i used the sample sceneDrawer and did a little modification on it. It will draw scene in real time,and when detected a fall it will display on screen.

For the more detail about the program, you can see the source code, I write the comment to explain how it work.

Execute the Program

-repertory Bin : all program executable and the file “DataOut.txt”
-repertory Conf : the pre-configuration xml file “SamplesConfig.xml”
-repertory NiDrawScene : source code draw scene
-repertory NiFallDetection : source code fall detection

All the data necessaire will save in the file “DataOut.txt”.

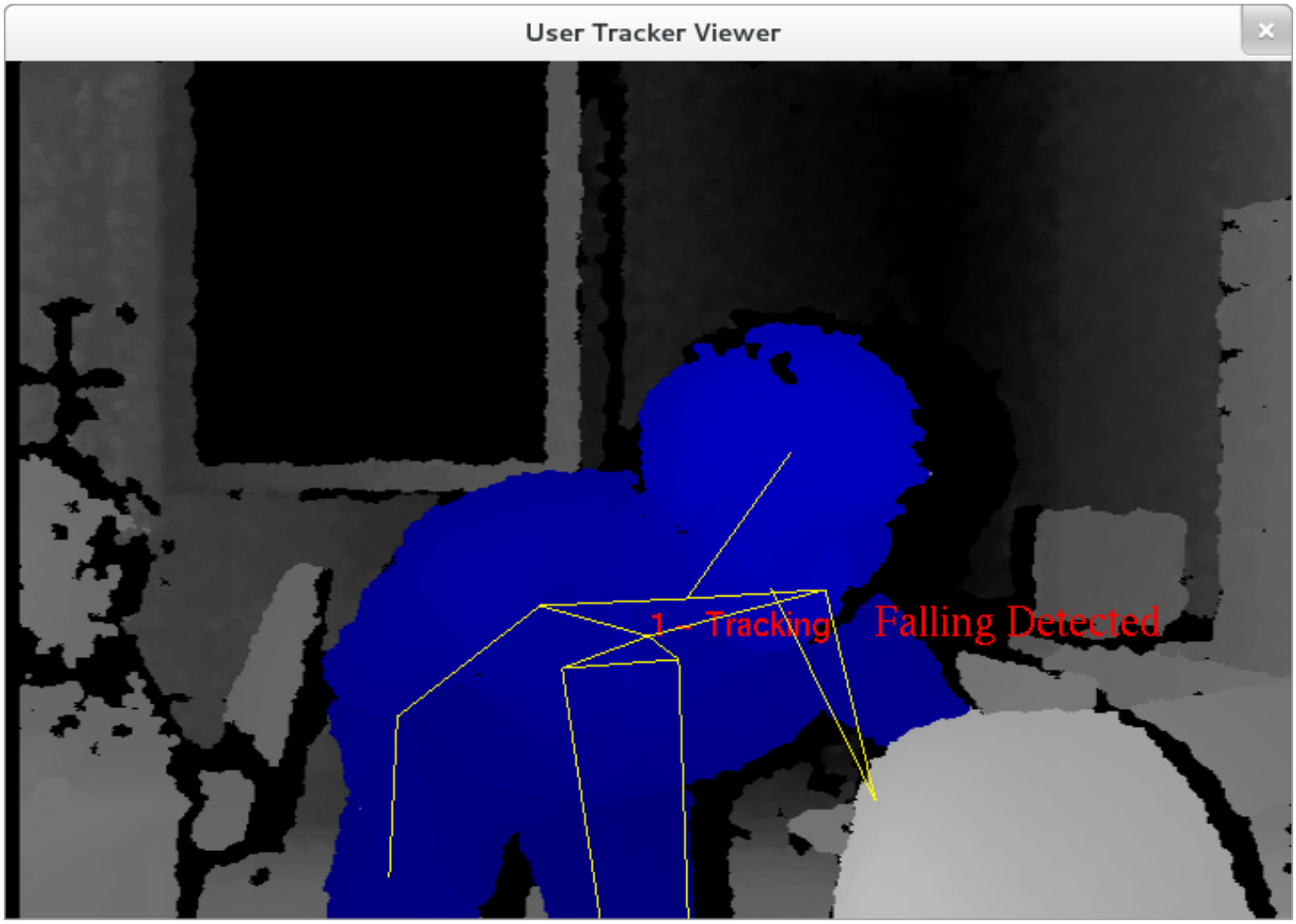
1- compile the first part - fall detection:
cd NiFallDetection/
make

2 - compile the second part - draw scene:
cd NiDrawScene/
make

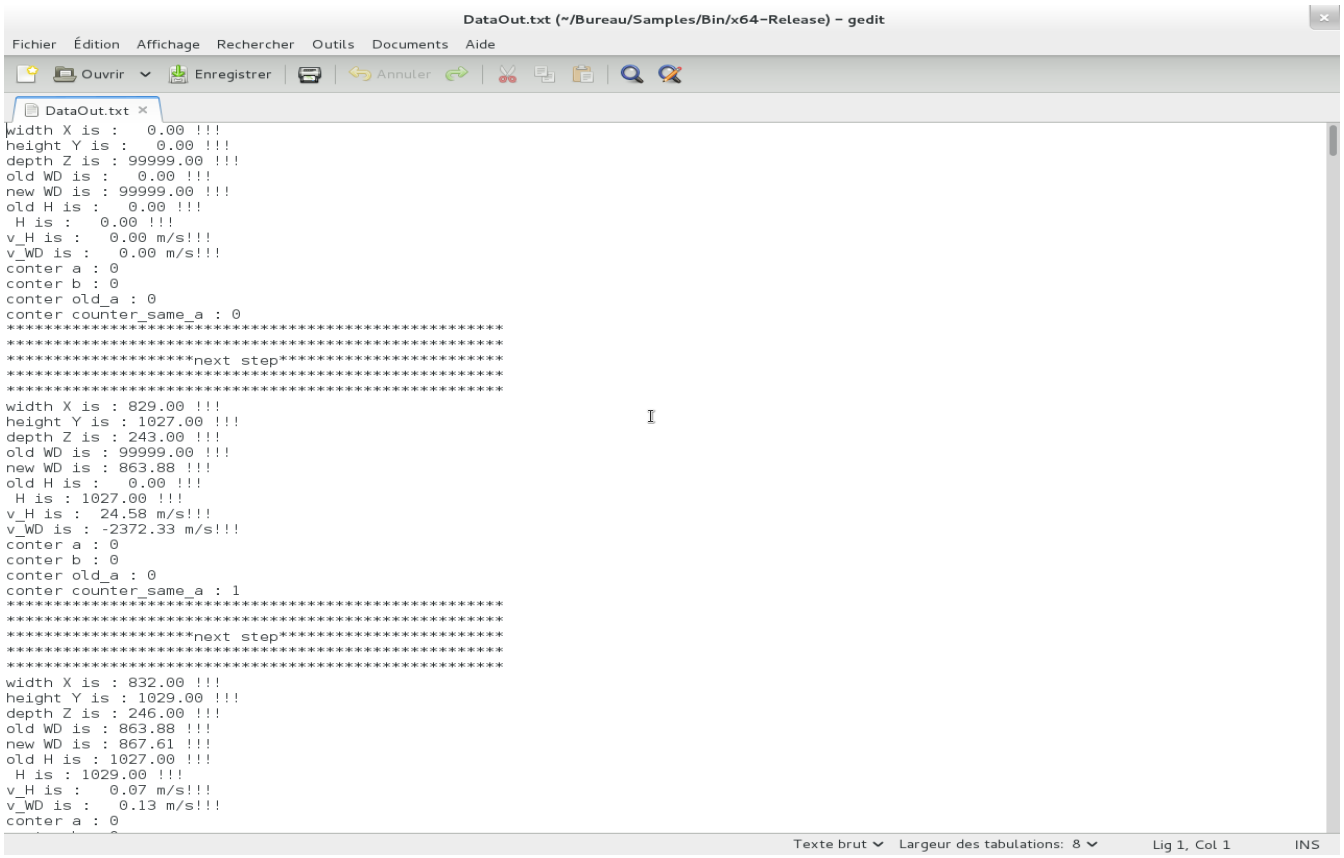
3- execute the program fall detection
cd BIn/x64-Release
./Sample-NiFallDetection

(if you do it on a x86 PC:
cd BIn/x86-Release
./Sample-NiFallDetection)

Draw Scene



DataOut.txt


DataOut.txt (~/Bureau/Samples/Bin/x64-Release) – gedit
Fichier Édition Affichage Rechercher Outils Documents Aide
Ouvrir
Enregistrer
Annuler
DataOut.txt
width X is : 0.00 !!!
height Y is : 0.00 !!!
depth Z is : 99999.00 !!!
old WD is : 0.00 !!!
new WD is : 99999.00 !!!
old H is : 0.00 !!!
H is : 0.00 !!!
v_H is : 0.00 m/s!!!
v_WD is : 0.00 m/s!!!
conter a : 0
conter b : 0
conter old_a : 0
conter counter_same_a : 0
****
****
**********next step**********
****
****
width X is : 829.00 !!!
height Y is : 1027.00 !!!
depth Z is : 243.00 !!!
old WD is : 99999.00 !!!
new WD is : 863.88 !!!
old H is : 0.00 !!!
H is : 1027.00 !!!
v_H is : 24.58 m/s!!!
v_WD is : -2372.33 m/s!!!
conter a : 0
conter b : 0
conter old_a : 0
conter counter_same_a : 1
****
****
**********next step**********
****
****
width X is : 832.00 !!!
height Y is : 1029.00 !!!
depth Z is : 246.00 !!!
old WD is : 863.88 !!!
new WD is : 867.61 !!!
old H is : 1027.00 !!!
H is : 1029.00 !!!
v_H is : 0.07 m/s!!!
v_WD is : 0.13 m/s!!!
conter a : 0
Texte brut
Largeur des tabulations: 8
Lig 1, Col 1
INS