Kinect

C. Andújar
Contents

• Introduction
• Hardware
• Demos
• Software
  – Freenect
  – OpenNI + NITE + SensorKinect
• More demos
• Projects using Kinect
• Upcoming sensors
INTRODUCTION
3D input competitors
Kinect origins

PrimeSense – Israeli company (2005) focused on vision and natural interaction
- Natural interface for controlling living-room devices (e.g. TVs)
- Gesture-based browsing of images, video...
- Full-body interaction for games

PrimeSensor – low-cost device for TV screens providing real-time depth, color and audio data of the living room.
- The user is not required to wear or hold anything
- Insensitive to lighting conditions
- It does not require calibration
Microsoft Kinect

- New controller for Microsoft’s Xbox 360
- Full-body tracking, face and voice recognition
- Low cost (99 Euros as of April 2011)
HARDWARE
Main components

- 3D Depth Sensors
- RGB Camera
- Multi-Array Mic
- Motorized Tilt
Main components

Video
- Color CMOS camera
- Infrared (IR) CMOS camera
- Infrared projector - 830nm, 60mW laser diode.

Audio
- Four microphones

Tilt control
- Motor
- Accelerometer (3-axes)

Processors & memory
- PrimeSense chip PS1080-A2
- 64 MB DDR2 SDRAM
Image sensors

- Color camera: 640x480 sensor, 640x480@30fps output
- IR camera: 1280x1024 sensor, 640x480@30fps output
- Operation range (depth sensor) = 0.8m - 3.5m
- FOV = 58° H, 45° V, 70° D
- Spatial resolution (@ 2m distance) = 3mm
- Depth resolution (@ 2m distance) = 1cm
## Image sensors specs

<table>
<thead>
<tr>
<th>Parameter</th>
<th>typical Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optical Format</td>
<td>1/6-inch (4:3)</td>
</tr>
<tr>
<td>Active Imager Size</td>
<td>2.30mm(H) x 1.73mm(V) 2.88mm Diagonal</td>
</tr>
<tr>
<td>Active Pixels</td>
<td>640H x 480V</td>
</tr>
<tr>
<td>Pixel Size</td>
<td>3.6μm x 3.6μm</td>
</tr>
<tr>
<td>Color Filter Array</td>
<td>RGB Bayer Pattern</td>
</tr>
<tr>
<td>Shutter type</td>
<td>Electronic Rolling Shutter (ERS)</td>
</tr>
<tr>
<td>Maximum Data Rate/</td>
<td>12 MPS–13.5 MPS/24 MHz–27 MHz</td>
</tr>
<tr>
<td>Master Clock</td>
<td></td>
</tr>
<tr>
<td>Frame Rate (VGA 640H x 480V)</td>
<td>30 fps at 27 MHz</td>
</tr>
<tr>
<td>ADC Resolution</td>
<td>10-bit, on-chip</td>
</tr>
<tr>
<td>Responsivity</td>
<td>1.0V/lux-sec (550nm)</td>
</tr>
<tr>
<td>Dynamic Range</td>
<td>71dB</td>
</tr>
<tr>
<td>SNR&lt;sub&gt;MAX&lt;/sub&gt;</td>
<td>44dB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optical format</td>
<td>1/2-inch (5:4)</td>
</tr>
<tr>
<td>Active imager size</td>
<td>6.66mm(H) x 5.32mm(V)</td>
</tr>
<tr>
<td>Active pixels</td>
<td>1,280H x 1,024V</td>
</tr>
<tr>
<td>Pixel size</td>
<td>5.2μm x 5.2μm</td>
</tr>
<tr>
<td>Shutter type</td>
<td>Electronic rolling shutter (ERS)</td>
</tr>
<tr>
<td>Maximum data rate/</td>
<td>48 MPS/48 MHz</td>
</tr>
<tr>
<td>master clock</td>
<td></td>
</tr>
<tr>
<td>Frame rate</td>
<td>SXGA (1280 x 1024)</td>
</tr>
<tr>
<td>ADC resolution</td>
<td>10-bit, on-chip</td>
</tr>
<tr>
<td>Responsivity</td>
<td>2.1 V/lux-sec</td>
</tr>
<tr>
<td>Dynamic range</td>
<td>68.2dB</td>
</tr>
<tr>
<td>SNR&lt;sub&gt;MAX&lt;/sub&gt;</td>
<td>45dB</td>
</tr>
</tbody>
</table>
Depth sensing

- The IR emitter projects an irregular pattern of IR dots of varying intensities.
- The IR camera reconstructs a depth image by recognizing the distortion in this pattern.
Connecting the Kinect to a PC

Trivial for “old” Kinect models (those sold separately): USB-2 standard connector
Further info on Kinect’s hardware

http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/1
http://openkinect.org/
Viewer (color map, IR map, depth map); skeletor

DEMOS
SOFTWARE
Drivers

Kinect drivers

- Libfreenect (official)
- OpenNI + NITE + Sensor Kinect

low-level data

- color & depth map

high-level data

- gestures/poses

App
libfreenect

- [www.openkinect.org](http://www.openkinect.org)
- Unofficial, open source driver (GLP2)
- Linux, Windows, OS X
- Raw data (color map, IR map, depth map)
- Installation: trivial
OpenNI + NITE + SensorKinect

• OpenNI
  – SDK for natural interfaces; open source
  – www.openni.org

• NITE
  – OpenNI Plugin for gesture/pose recognition; closed source
  – www.openni.org

• SensorKinect
  – Driver for the Kinect; fork from the PrimeSensor driver (open source)
  – github.com/avin2/SensorKinect
OpenNI

- Framework for modules/sensors providing depth maps, color maps, scene maps, gesture recognition, user pose (skeleton)
- [www.openni.org](http://www.openni.org)
OpenNI

OpenNI aims at abstracting developers from sensor hardware (Kinect, PrimeSensor...) and CV algorithms (scene analysis, gesture recognition, pose recognition...).
OpenNI basic concepts

- Production node: device/map/CV algorithm abstraction
- Production graph: a particular combination of nodes
- Module: plug-in (.dll, .so file) implementing nodes
OpenNI nodes

- **Audio**: generates an audio stream
- **Depth**: generates depth-maps
- **Image**: generates color maps
- **IR**: generates IR maps
- **SceneAnalyzer**: generates a label map (e.g. segmenting objects)
- **Gestures**: recognizes gestures and calls specific callbacks
- **Hands**: recognizes hands and calls specific callbacks when a hand appears/is moved/disappears
- **User**: computes user 3D info (including center of mass, pose detection, skeleton) and calls specific callbacks
- **Recorder**: saves node data to a file
- **Player**: reads data from a file and replays it
- **Codec**: used for compression and decompression of data in recordings
OpenNI nodes

- **Audio**
  - Generates an audio stream

- **Depth**
  - Generates depth-maps

- **Image**
  - Generates color maps

- **IR**
  - Generates IR maps

- **SceneAnalyzer**
  - Generates a label map (e.g. segmenting objects)

- **Gestures**
  - Recognizes gestures and calls specific callbacks

- **Hands**
  - Recognizes hands and calls specific callbacks when a hand appears/is moved/disappears

- **User**
  - Computes user 3D info (including center of mass, pose detection, skeleton) and calls specific callbacks

- **Recorder**
  - Saves node data to a file

- **Player**
  - Reads data from a file and replays it

- **Codec**
  - Used for compression and decompression of data in recordings
OpenNI nodes - Audio

**Audio Generator**

An object that generates Audio data.

Main Functionalities:

- Get Audio Buffer
- **Wave Output Modes property**: Configure the audio output, including sample rate, number of channels and bits-per-sample
**OpenNI nodes**

- **Audio** generates an audio stream

- **Depth** generates depth-maps
- **Image** generates color maps
- **IR** generates IR maps
- **SceneAnalyzer** generates a label map (e.g. segmenting objects)

- **Gestures** recognizes gestures and calls specific callbacks
- **Hands** recognizes hands and calls specific callbacks when a hand appears/is moved/disappears
- **User** computes user 3D info (including center of mass, pose detection, skeleton) and calls specific callbacks

- **Recorder** saves node data to a file
- **Player** read data from a file and replays it
- **Codec** used for compression and decompression of data in recordings
# OpenNI nodes - Maps

## Depth Generator
An object that generates a depth map.

Main Functionalities:
- **Get depth map**: Provides the depth map
- **Get Device Max Depth**: The maximum distance available for this depth generator
- **Field of View property**: Configures the values of the horizontal and vertical angles of the sensor
- **User Position capability**

## Image Generator
A Map Generator that generates a color image map.

Main Functionalities:
- **Get Image Map**: Provides the color image map
- **Pixel format property**

## IR Generator
A map generator that generates an IR map.

Main Functionality:
- **Get IR Map**: Provides the current IR map

## Scene Analyzer
A map generator that gets raw sensory data and generates a map with labels that clarify the scene.

Main Functionalities:
- **Get Label Map**: Provides a map in which each pixel has a meaningful label (i.e. figure 1, figure 2, background, and so on)
- **Get Floor**: get the coordinates of the floor plane
OpenNI nodes

• Audio generates an audio stream
• Depth generates depth-maps
• Image generates color maps
• IR generates IR maps
• SceneAnalyzer generates a label map (e.g. segmenting objects)

• Gestures recognizes gestures and calls specific callbacks
• Hands recognizes hands and calls specific callbacks when a hand appears/is moved/disappears
• User computes user 3D info (including center of mass, pose detection, skeleton) and calls specific callbacks

• Recorder saves node data to a file
• Player read data from a file and replays it
• Codec used for compression and decompression of data in recordings
OpenNI nodes - Recognition

*Gesture Generator*

An **Hand Point Generator**

A **User Generator**

- An object that generates data relating to a figure in the scene.

- Main Functionalities:
  - **Get Number of Users**: Provides the number of users currently detected in the scene
  - **Get Users**: Provides the current users
  - **Get User CoM**: Returns the location of the center of mass of the user
  - **Get User Pixels**: Provides the pixels that represent the user. The output is a map of the pixels of the entire scene, where the pixels that represent the body are labeled User ID.
  - **Register/Unregister user callbacks**: The following actions will generate user callbacks:
    - When a new user is identified
    - When an existing user disappears
OpenNI nodes - User node

```c
UserGenerator (XnNodeHandle hNode=NULL)
XnUInt16 GetNumberOfUsers () const
XnStatus GetUsers (XnUserID aUsers[], XnUInt16 &nUsers) const
XnStatus GetCoM (XnUserID user, XnPoint3D &com) const
XnStatus GetUserPixels (XnUserID user, SceneMetadata &md)
XnStatus RegisterUserCallbacks (UserHandler NewUserCB, UserHandler LostUserCB, void (*)(Cookie, XnCallbackHandle &hCallback)
    void UnregisterUserCallbacks (XnCallbackHandle hCallback)
SkeletonCapability GetSkeletonCap ()
PoseDetectionCapability GetPoseDetectionCap ()
XnStatus Create (Context &context, Query *pQuery=NULL, EnumerationErrors *pErrors=NULL)
```

OpenNI nodes - Skeleton

**XN_C_API XnStatus xnGetSkeletonJointOrientation ( XnNodeHandle hInstance, XnUserID user, XnSkeletonJoint eJoint, XnSkeletonJointOrientation * pJoint )**

Get a specific joint's orientation.

**Parameters:**
- `hInstance` [in] A handle to the instance
- `user` [in] The ID of the user to which the skeleton belongs
- `eJoint` [in] The interesting joint
- `pJoint` [out] The joint's current orientation

**XN_C_API XnStatus xnGetSkeletonJointPosition ( XnNodeHandle hInstance, XnUserID user, XnSkeletonJoint eJoint, XnSkeletonJointPosition * pJoint )**

Get a specific joint's position.

**Parameters:**
- `hInstance` [in] A handle to the instance
- `user` [in] The ID of the user to which the skeleton belongs
- `eJoint` [in] The interesting joint
- `pJoint` [out] The joint's current position
OpenNI example

// Initialize context object
xn::Context context;
context.Init();

// Create a DepthGenerator node
xn::DepthGenerator depth;
depth.Create(context);

// Make it start generating data
context.StartGeneratingAll();

while (true)
{
    // Wait for new data to be available
    XnStatus status = context.WaitOneUpdateAll(depth);
    if (status == XN_STATUS_OK) {
        const XnDepthPixel* pDepthMap = depth.GetDepthMap();
        // process depth map
        ....
    }
}
OpenNI - Initializing nodes

// Create a DepthGenerator node
xn::DepthGenerator depth;
depth.Create(context);

// Configure node
XnMapOutputMode outputMode;
outputMode.nXRes = 640;
outputMode.nYRes = 480;
outputMode.nFPS = 30;
status = depth.SetMapOutputMode(outputMode);
Sample applications

- **NiSimpleRead** – takes a depth generator node from an XML config file and prints out the depth value of the middle pixel.
- **NiSimpleViewer** - OpenGL application which draws the depth maps and the color maps to the screen.
- **NiSampleModule** - sample for writing a module implementing a depth node with mirror capability.
- **NiViewer** – displays depth, image and IR maps, plays audio, etc.
NITE

Middleware for pose/gesture recognition

Calibration pose
NITE

- Joint positions: given in real world coords (mm)
- Joint orientations: 3x3 rotation matrix representing the rotation of joint’s local coords w.r.t world coords (the first column is the direction of the joint’s +X axis in world coordinates, and so on).
Kinect quick start (Windows)

- Install OpenNI binary
- Install SensorKinect binary
- Install NITE – key 0KOIk2JeIcICIPWVnMoRKn5cdY4=
- Connect Kinect to PC
- Try demos
MORE DEMOS

Sample-PointViewer, reconstructor, Miku²
Projects using Kinect

• About 420 projects in www.kinecthacks.net
Upcoming sensors

Xtion PRO
The World's First Professional Pack
FOR Developers

Controller-free
Real time motion capture &
Body movement tracking system

Create your own applications with ASUS Xtion Pro!

Specification:
Effective distance: between 0.8m and 3.5m
Effective angle: 70 degree
Interface/power: USB 2.0
Sensor: IR transmitter / IR receiver
System Requirement: Windows 7,Windows Vista or
Windows XP (x86 and x64)
Programming Language: C++, C#