Kinect

C. Andújar

Contents

- Introduction
- Hardware
- Demos
- Software
  - Freenect
  - OpenNI + NITE + SensorKinect
- More demos
- Projects using Kinect
- Upcoming sensors

INTRODUCTION

3D input competitors

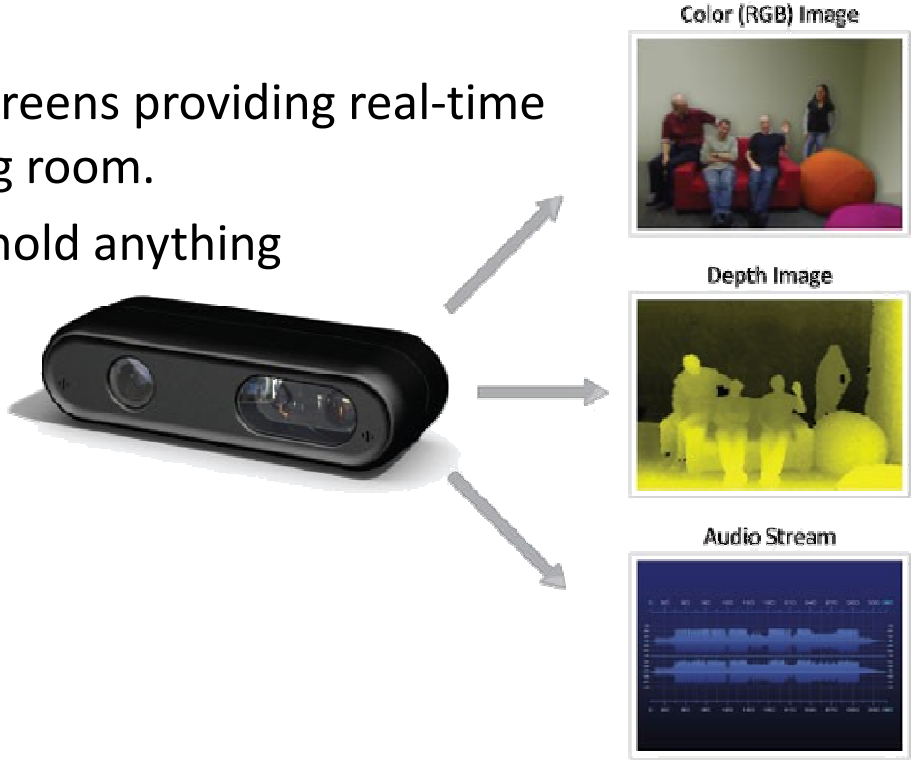
Kinect origins

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

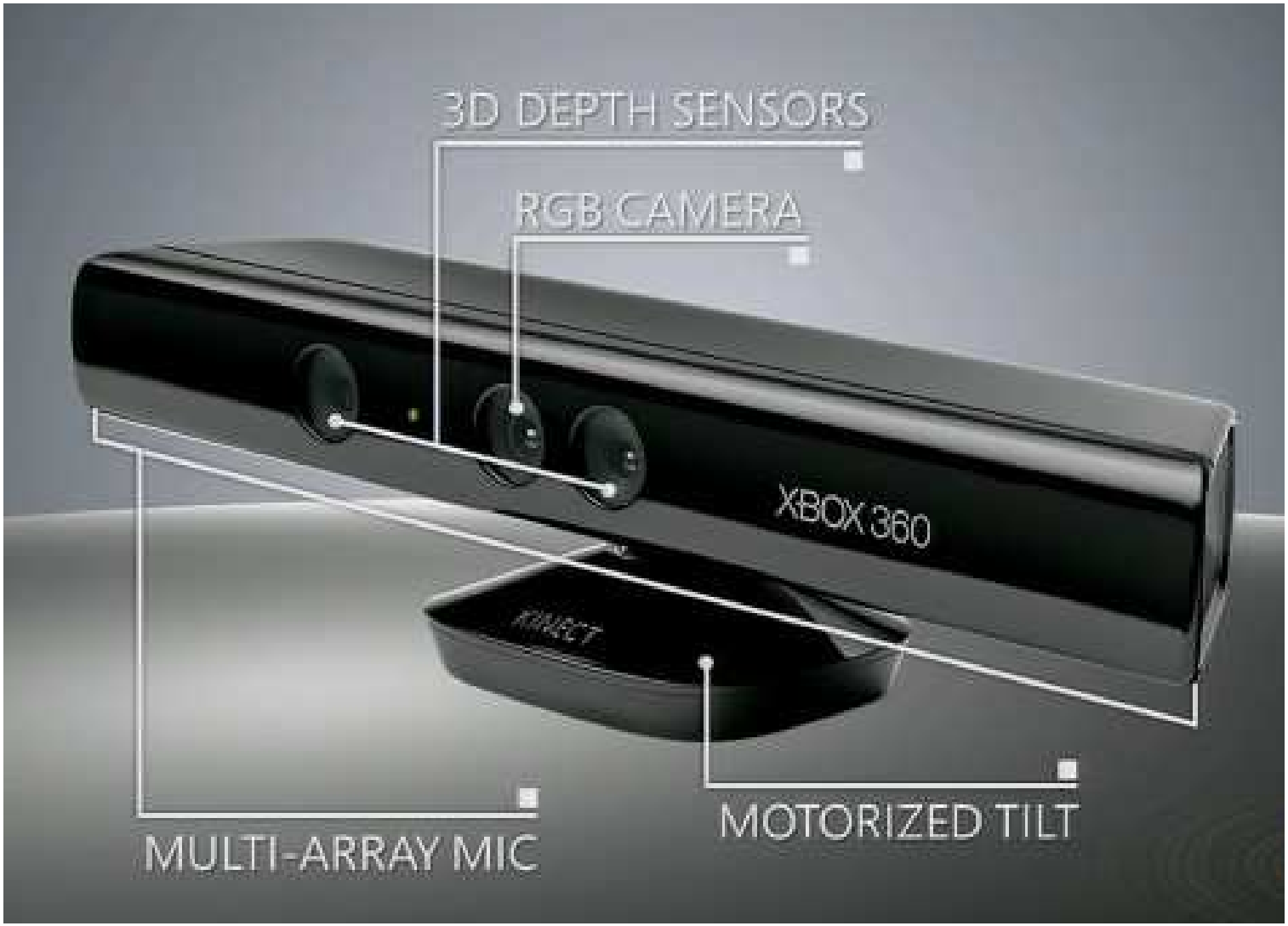


Microsoft Kinect

- 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



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

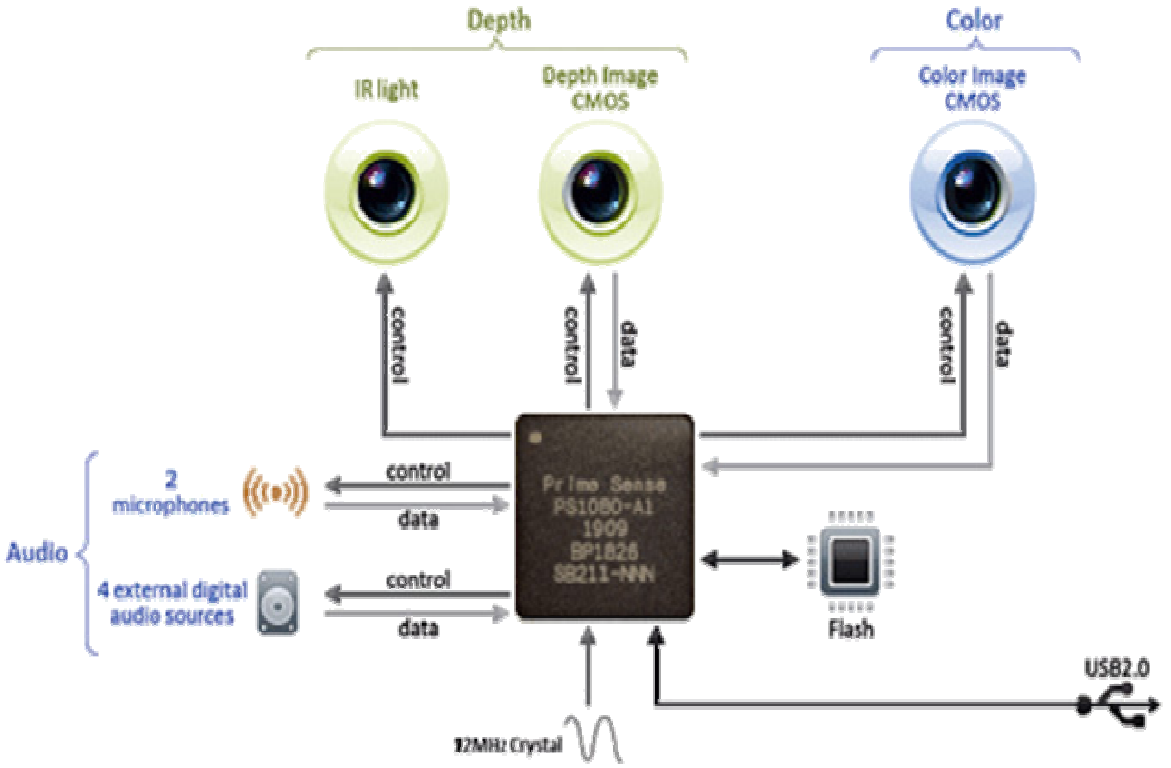


Image sensors

- 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

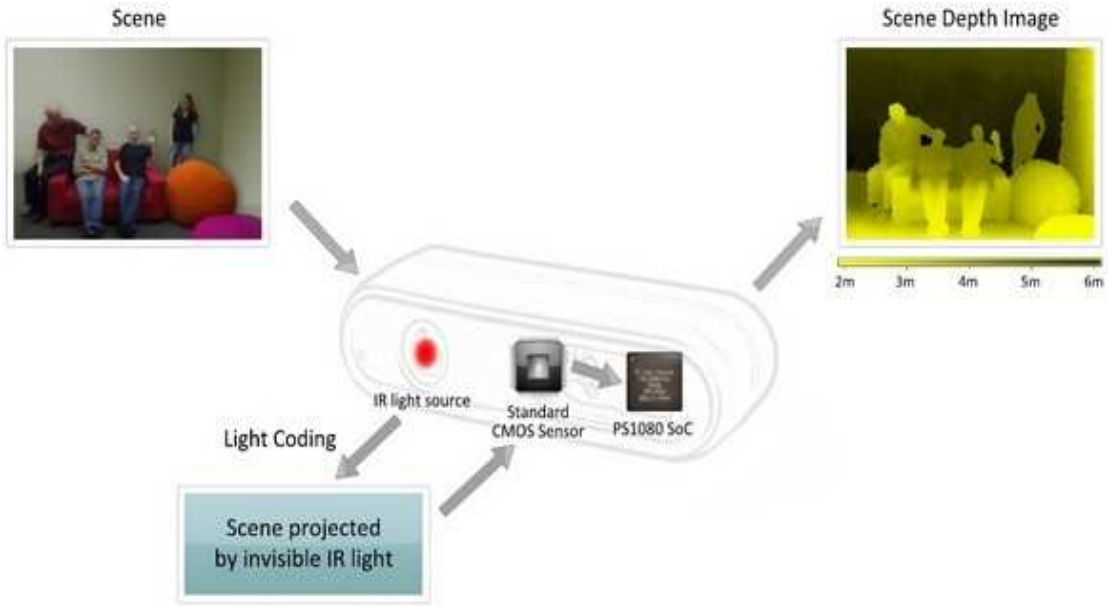
Image sensors specs

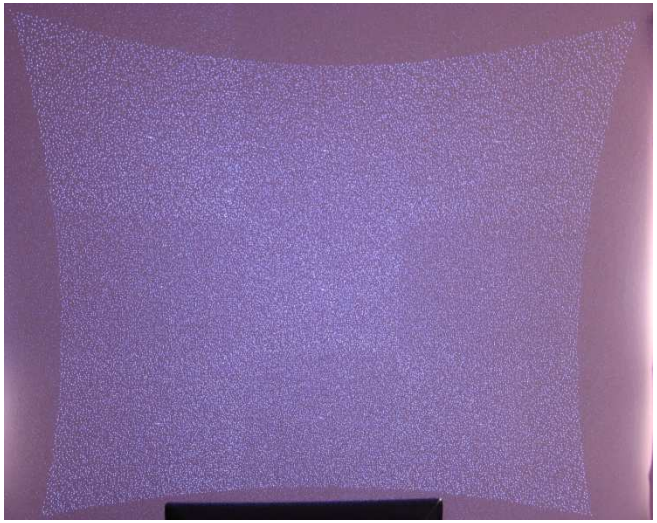
Parameter	typical Value
Optical Format	1/6-inch (4:3)
Active Imager Size	2.30mm(H) x 1.73mm(V) 2.88mm Diagonal
Active Pixels	640H x 480V
Pixel Size	3.6µm x 3.6µm
Color Filter Array	RGB Bayer Pattern
Shutter type	Electronic Rolling Shutter (ERS)
Maximum Data Rate/ Master Clock	12 MPS–13.5 MPS/ 24 MHz–27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

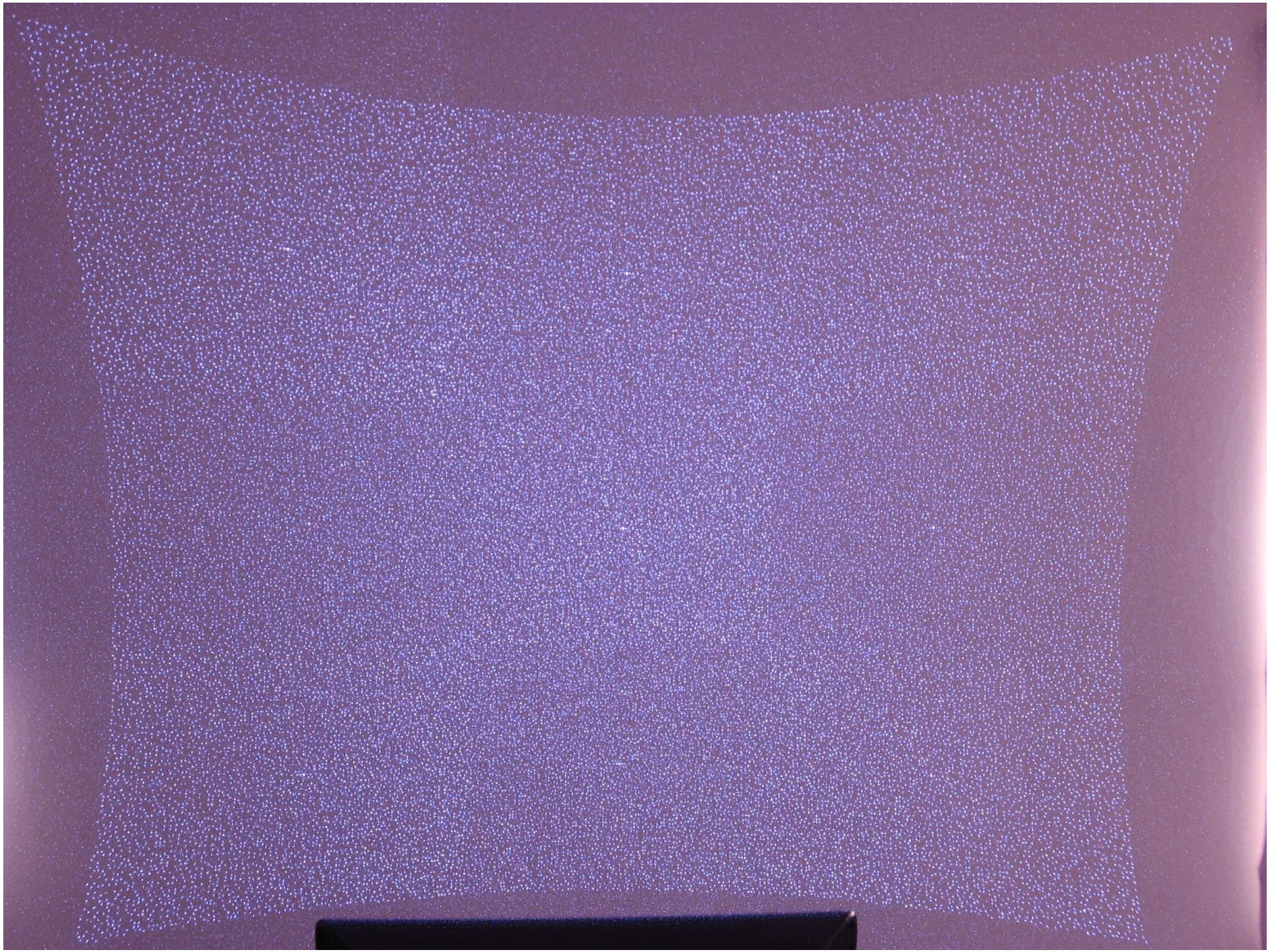
Parameter		Value
Optical format		1/2-inch (5:4)
Active imager size		6.66mm(H) x 5.32mm(V)
Active pixels		1,280H x 1,024V
Pixel size		5.2µm x 5.2µm
Shutter type		Electronic rolling shutter (ERS)
Maximum data rate/ master clock		48 MPS/48 MHz
Frame rate	SXGA (1280 x 1024)	30 fps progressive scan; programmable
ADC resolution		10-bit, on-chip
Responsivity		2.1 V/lux-sec
Dynamic range		68.2dB
SNR_{MAX}		45dB

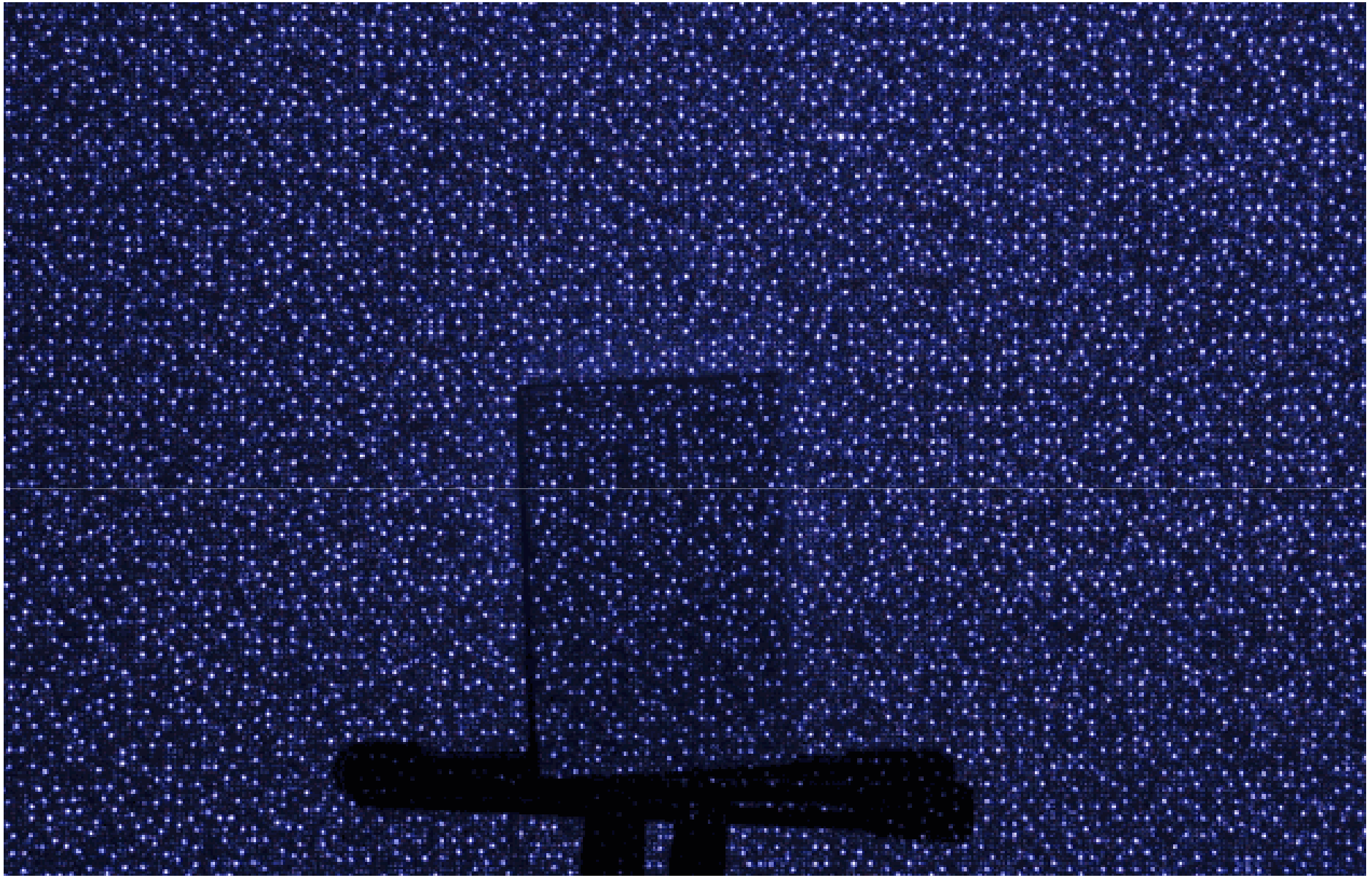
Depth sensing

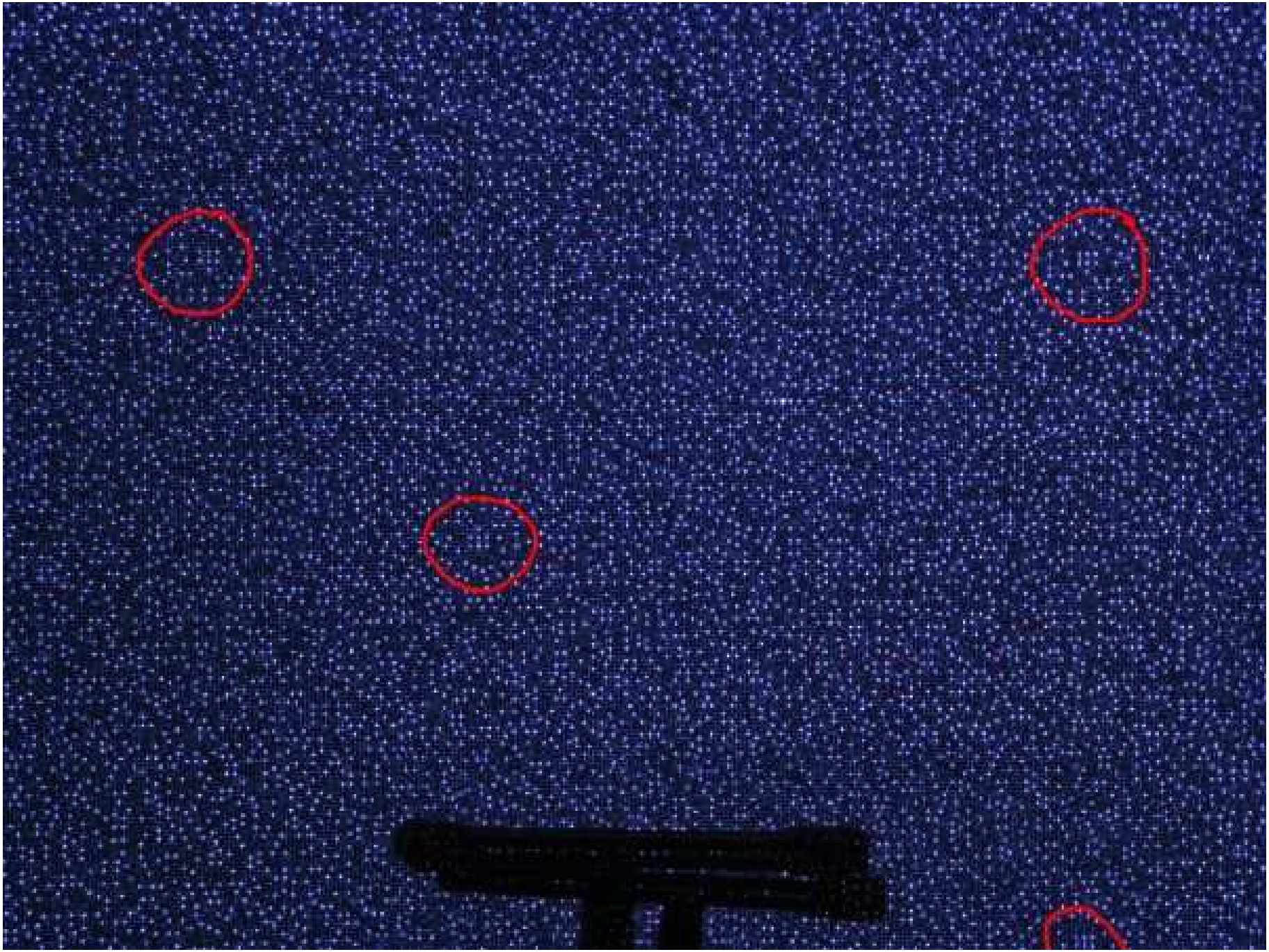
- The IR emitter projects an irregular pattern of IR dots of varying intensities.
- The IR camera reconstructs a depth image by recognizing the distortion in this pattern.

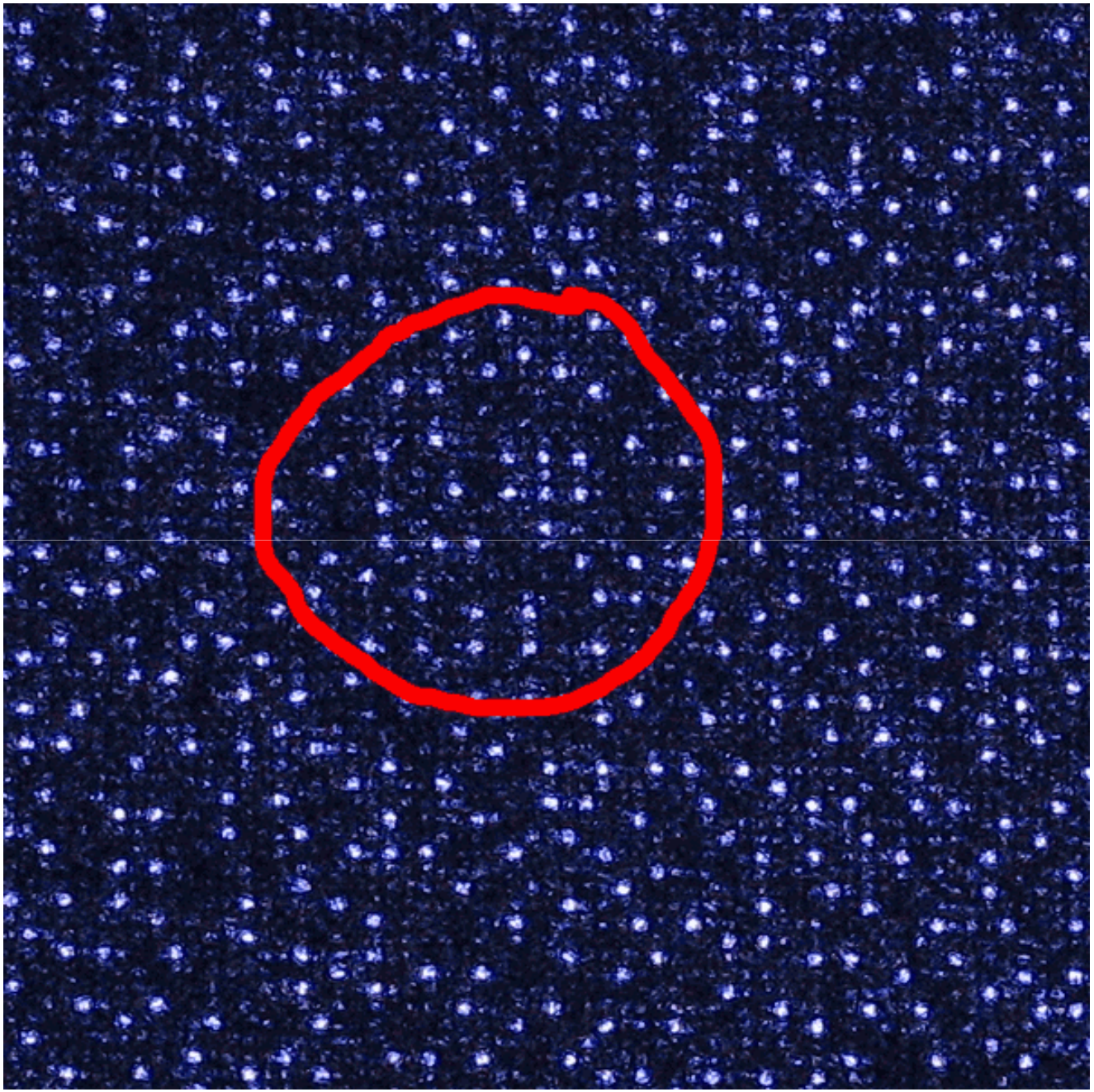












Connecting the Kinect to a PC

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/

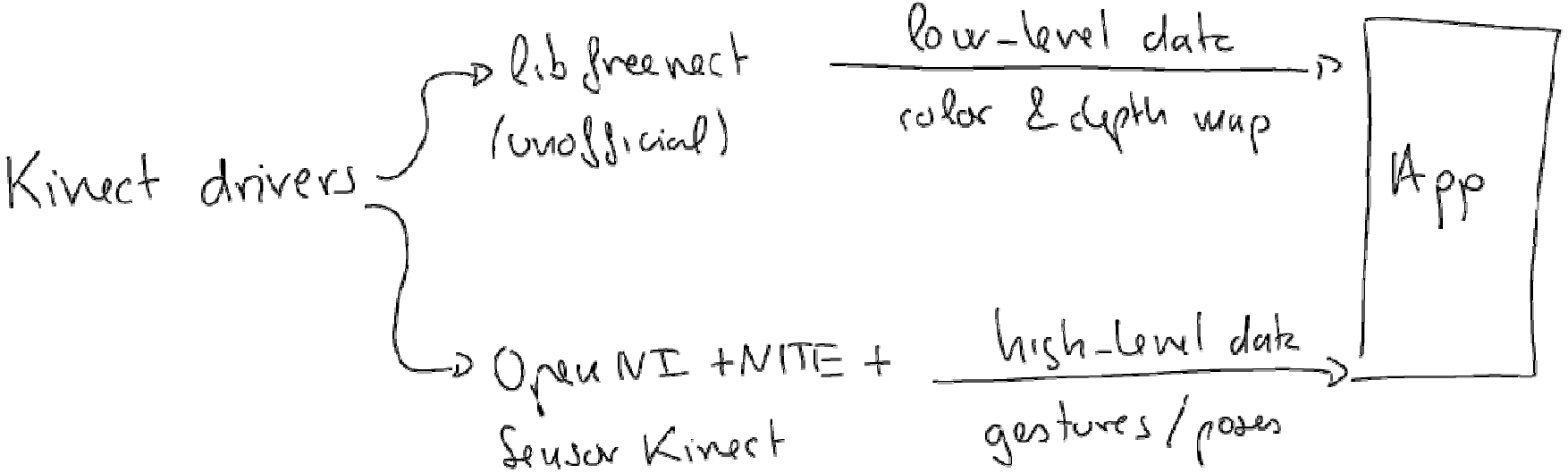
http://www.freepatentsonline.com/7433024.pdf

Viewer (color map, IR map, depth map); skeletor

DEMOS

SOFTWARE

Drivers


Kinect drivers
libfreenect
(unofficial)
low-level data
color & depth map
App
OpenNI +NITE +
Sensor Kinect
high-level data
gestures / poses

libfreenect

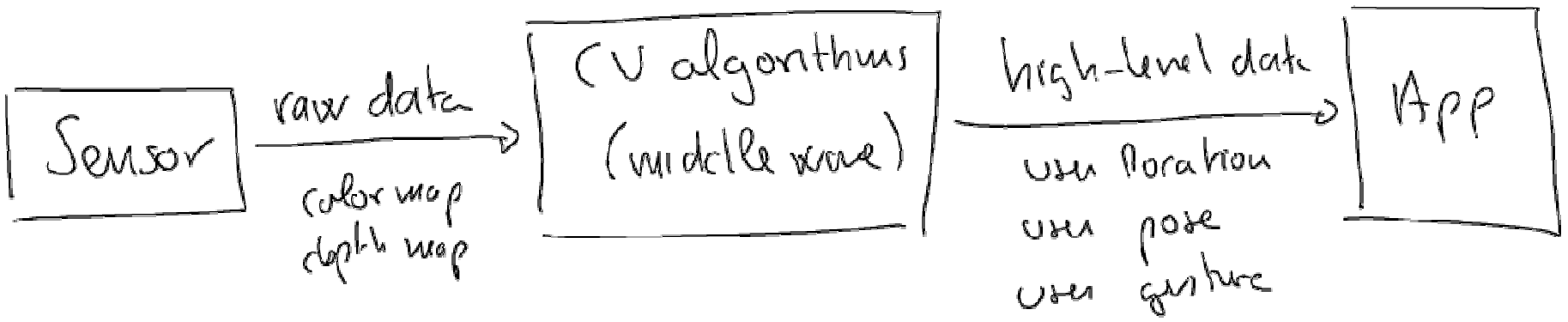
- www.openkinect.org
- Unofficial, open source driver (GLP2)
- Linux, Windows, OS X
- Raw data (color map, IR map, depth map)
- Installation: trivial

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

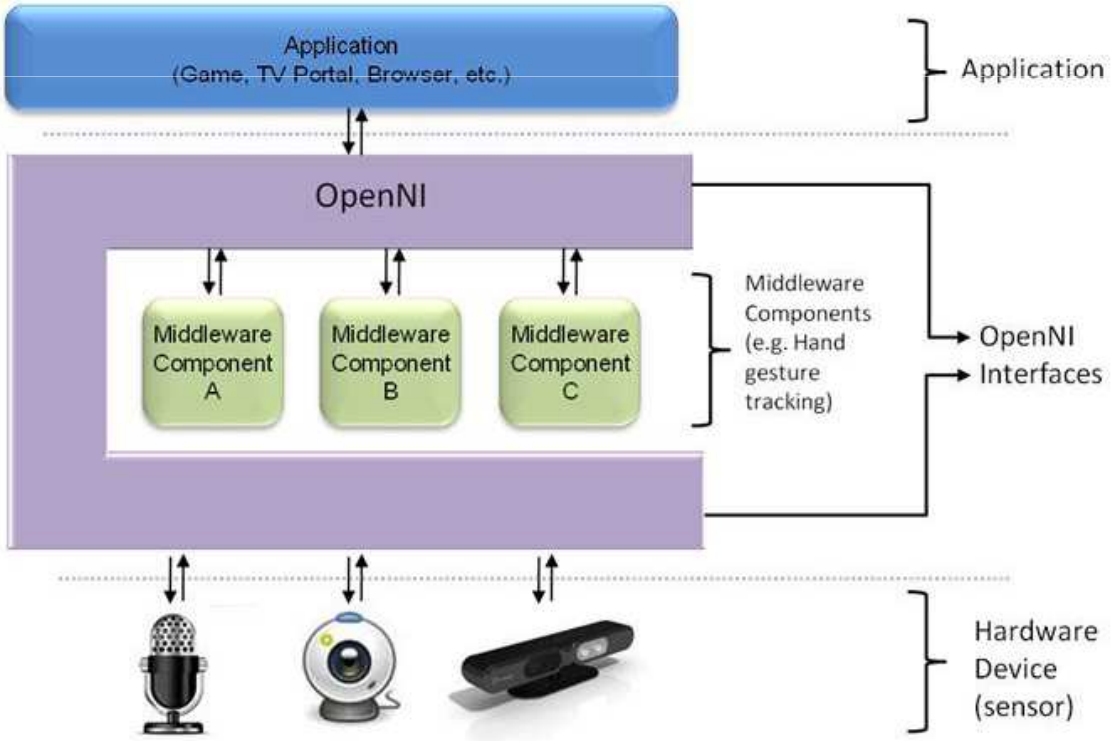
OpenNI

- Framework for modules/sensors providing **depth maps**, **color maps, scene maps, gesture recognition, user pose (skeleton)**
- **www.openni.org**

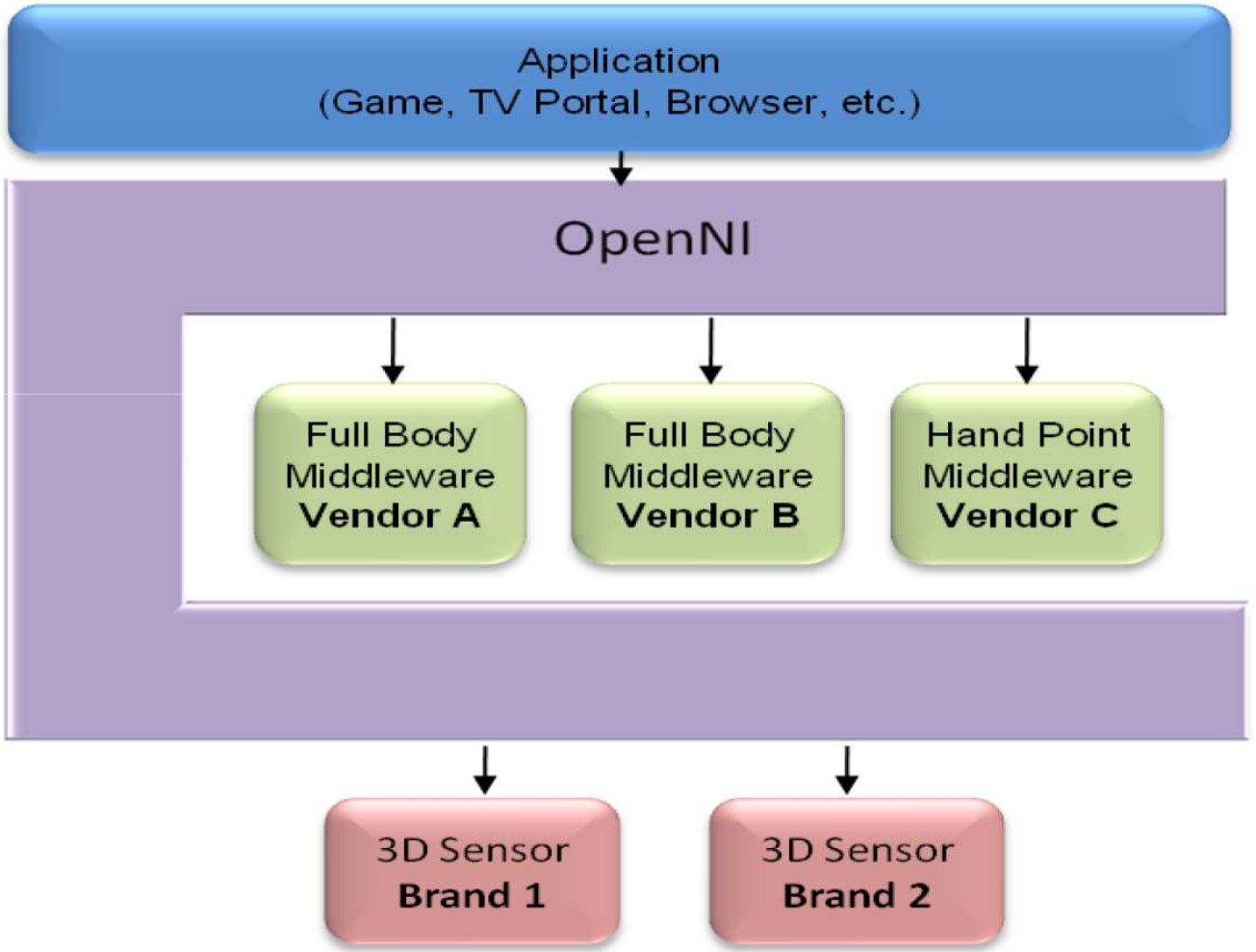


OpenNI

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

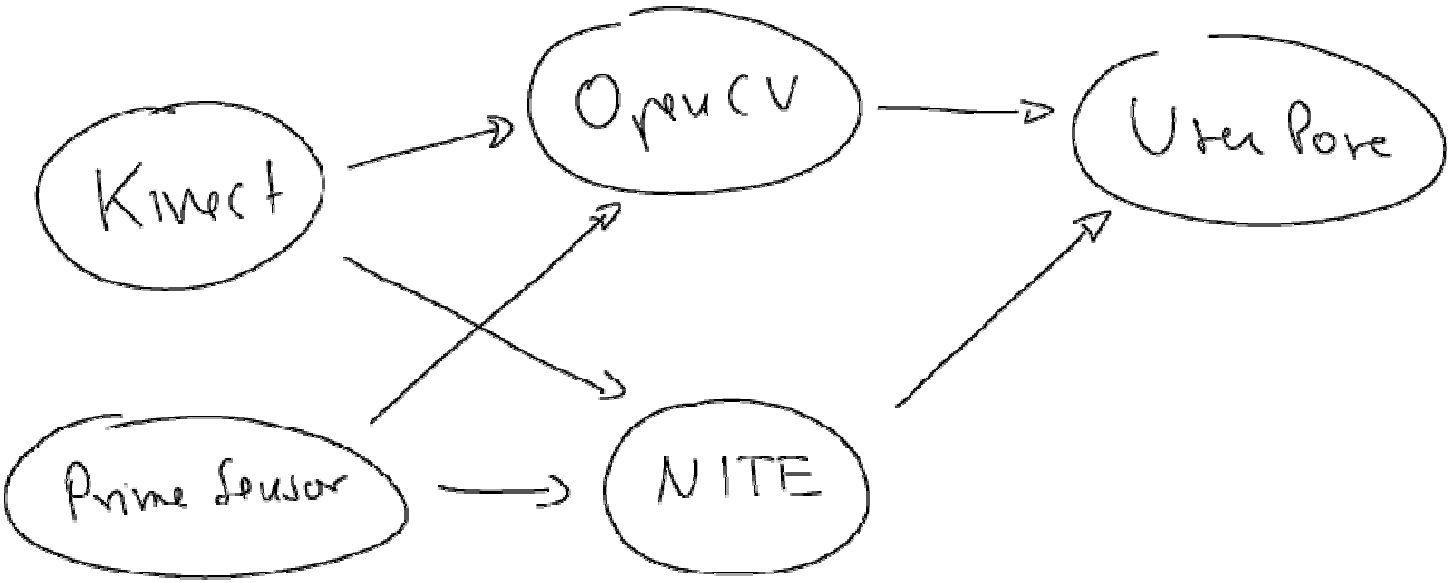


OpenNI


Application
(Game, TV Portal, Browser, etc.)
OpenNI
Full Body
Middleware
Vendor A
Full Body
Middleware
Vendor B
Hand Point
Middleware
Vendor C
3D Sensor
Brand 1
3D Sensor
Brand 2

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	read data from a file and replays it
Codec	used for compression and decompression of data in recordings

OpenNI nodes

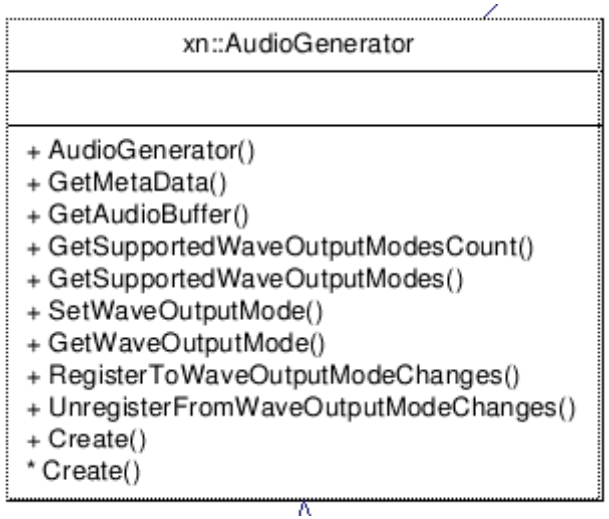
- **Audio** generates an audio stream

- **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** read 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

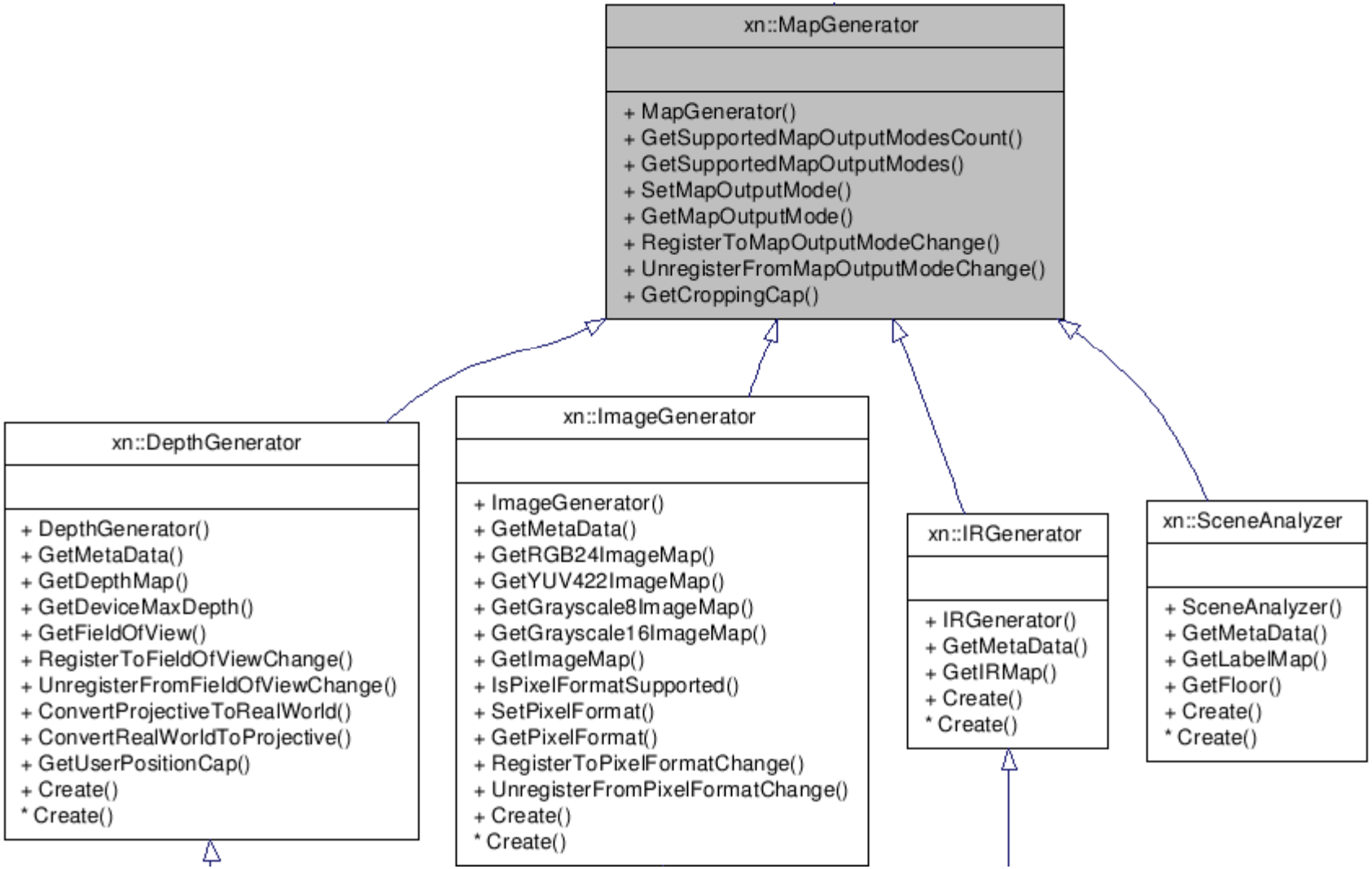
- **Audio** generates an audio stream

- **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



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

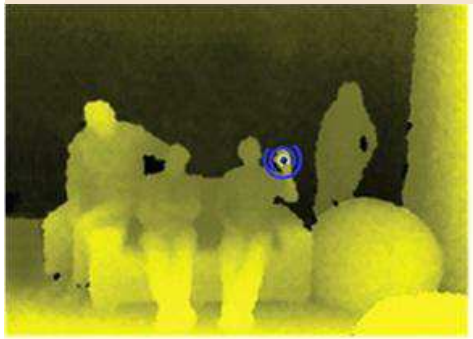
An

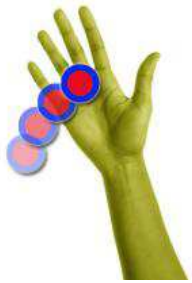
Hand Point Generator

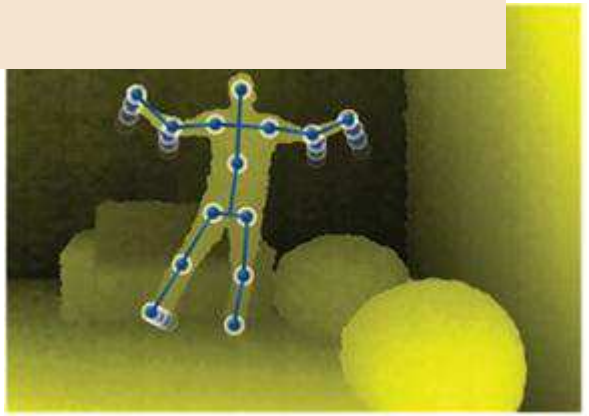
Ma

A **User Generator**

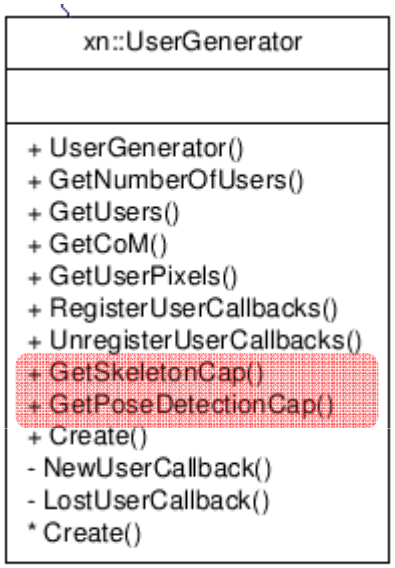
- N An object that generates data relating to a figure in the scene.
- Main Functionalities:
  - **Get Number of Users**: Provides the number of users currently detected in the scene
  - **Get Users**: Provides the current users
  - **Get User CoM**: Returns the location of the center of mass of the 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



	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** &smd)
XnStatus	**RegisterUserCallbacks** (**UserHandler** NewUserCB, **UserHandler** LostUserCB, void *pCookie, **XnCallbackHandle** &hCallback)
void	**UnregisterUserCallbacks** (**XnCallbackHandle** hCallback)
SkeletonCapability	**GetSkeletonCap** ()
PoseDetectionCapability	**GetPoseDetectionCap** ()
XnStatus	**Create** (**Context** &context, **Query** *pQuery=NULL, **EnumerationErrors** *pErrors=NULL)

OpenNI nodes - Skeleton

XN_SKEL_**HEAD**
XN_SKEL_**NECK**
XN_SKEL_**TORSO**
XN_SKEL_**WAIST**
XN_SKEL_**LEFT_COLLAR**
XN_SKEL_**LEFT_SHOULDER**
XN_SKEL_**LEFT_ELBOW**
XN_SKEL_**LEFT_WRIST**
XN_SKEL_**LEFT_HAND**
XN_SKEL_**LEFT_FINGERTIP**
XN_SKEL_**RIGHT_COLLAR**
XN_SKEL_**RIGHT_SHOULDER**
XN_SKEL_**RIGHT_ELBOW**
XN_SKEL_**RIGHT_WRIST**
XN_SKEL_**RIGHT_HAND**
XN_SKEL_**RIGHT_FINGERTIP**
XN_SKEL_**LEFT_HIP**
XN_SKEL_**LEFT_KNEE**
XN_SKEL_**LEFT_ANKLE**
XN_SKEL_**LEFT_FOOT**
XN_SKEL_**RIGHT_HIP**
XN_SKEL_**RIGHT_KNEE**
XN_SKEL_**RIGHT_ANKLE**
XN_SKEL_**RIGHT_FOOT**

```
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

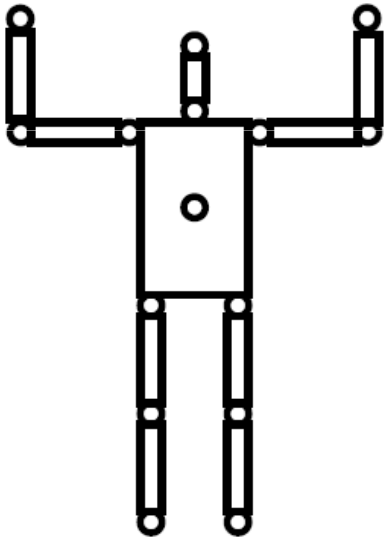
```
// Create a DepthGenerator node
xn::DepthGenerator depth;
depth.Create(context);

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

Sample applications

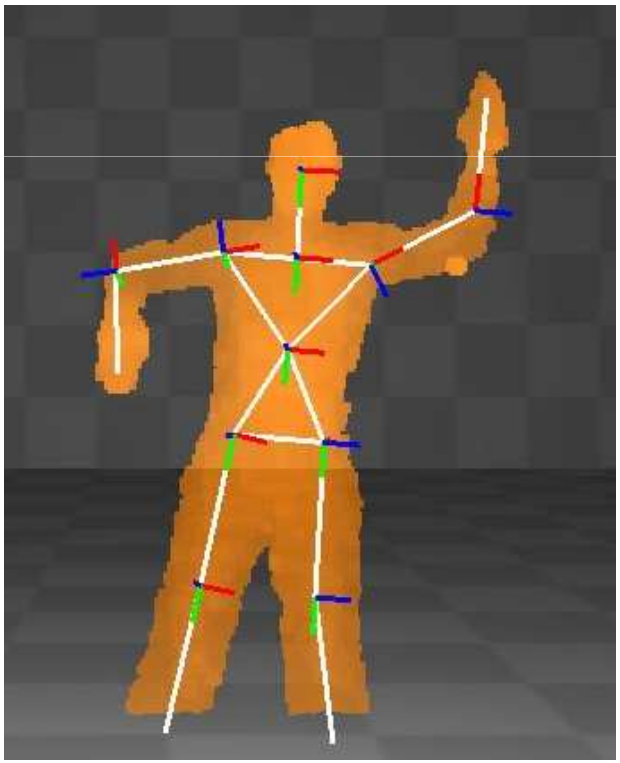
- **NiSimpleRead** – takes a depth generator node from an XML config file and prints out the depth value of the middle pixel.
- **NiSimpleViewer** - OpenGL application which draws the depth maps and the color maps to the screen.
- **NiSampleModule** - sample for writing a module implementing a depth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

NITE

Middleware for pose/gesture recognition

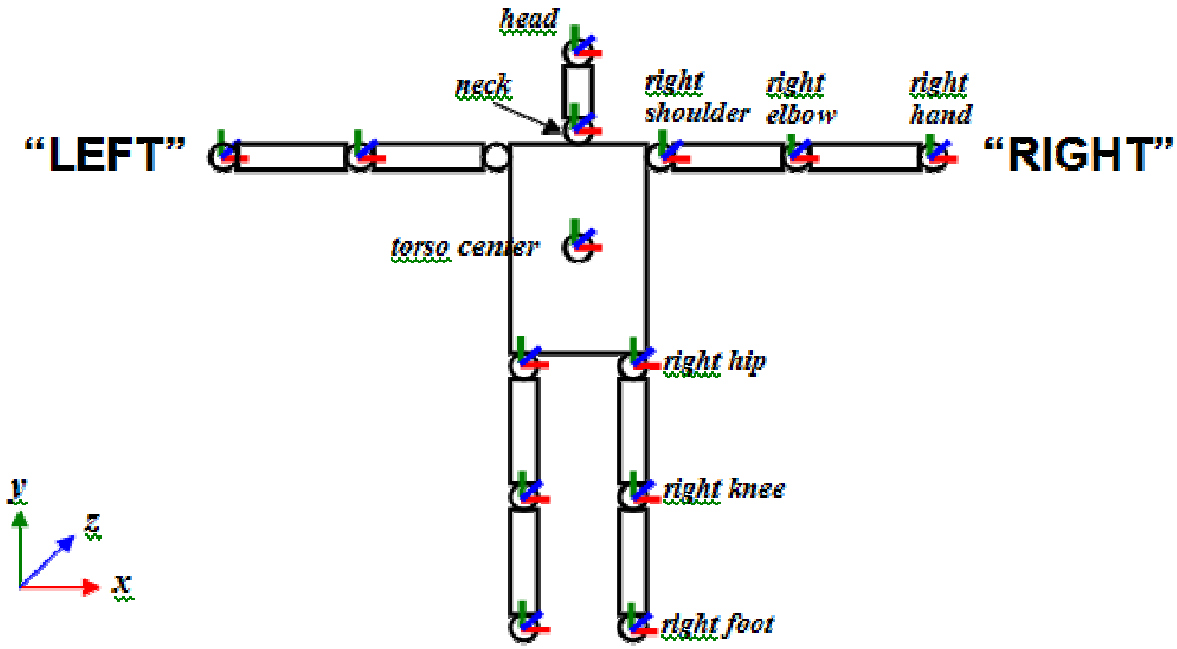


Calibration pose



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).



NOTE 1: Skeleton's front side is seen in this figure
NOTE 2: Upper arm is twisted such that if elbow is flexed the lower arm will bend forwards towards sensor.

Kinect quick start (Windows)

- Install OpenNI binary
- Install SensorKinect binary
- Install NITE – key 0KOIk2JeIBYClPWVnMoRKn5cdY4=
- Connect Kinect to PC
- Try demos

Sample-PointViewer, reconstructor, Miku^2

MORE DEMOS

Projects using Kinect

- About 420 projects in www.kinecthacks.net

Upcoming sensors