The Kinect Sensor

Simone Zenzaro

30/04/2014

Table of contents

The MS Kinect

- Motion sensing input devices
- Developed by Microsoft
- Based on Prime Sense hardware
- Released in USA on November 4, 2010
- Sold more than 8 million units in the first two months.

Name Origin

- First known as *Project Natal*
  - Alex Kipman, who incubated the project and is from Brazil, chose Natal, a city along the northeastern coast of Brazil, as a tribute to his country.
- Kinetic + Connect
  - From Greek *kīnētikós* (moving)
  - From Latin *nexus* (to bind, link)

Applications

- 3d environment reconstruction
- Healthcare
- NUIs and gesture recognition
- Sign language
- Behavioural research
- Security surveillance
- Virtual reality
- ...

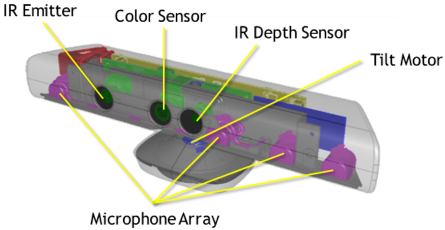



Device Capabilities

- Full body 3d motion capture
- Facial recognition
- Voice recognition

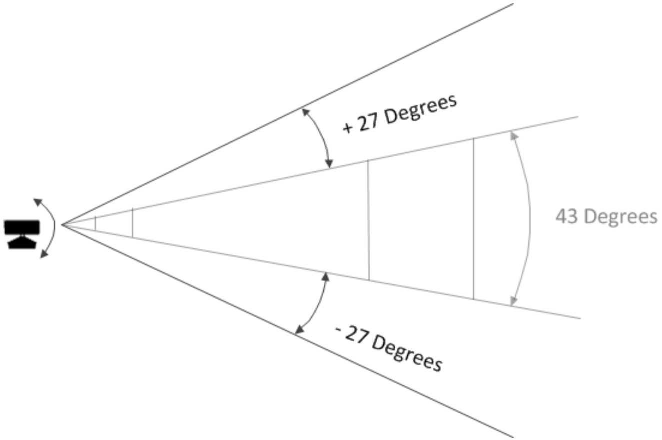
Main Components

- RGB camera
- Infrared laser
- Monochrome CMOS sensor (Active Pixel Sensor)
- Multi-array microphone
- Tilt Motor
- Accelerometer

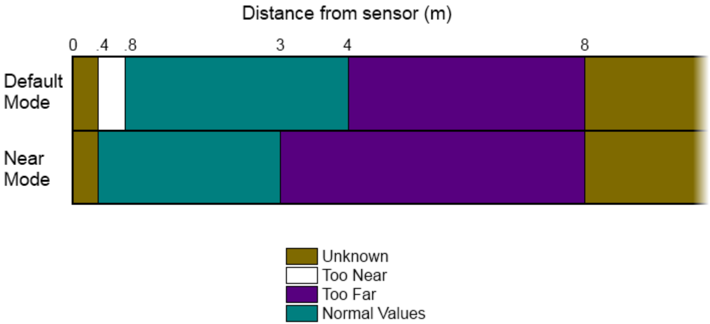


Interaction Space

The field of view of the Kinect cameras



Depth Space Range



RGB Camera

Data type, resolution and framerate:

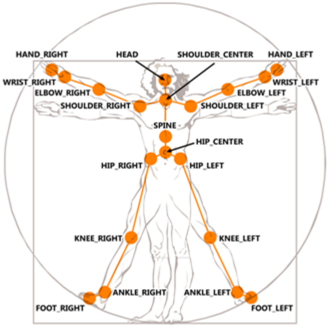
- InfraredResolution640x480Fps30
- RawBayerResolution1280x960Fps12
- RawBayerResolution640x480Fps30
- RawYuvResolution640x480Fps15
- RgbResolution1280x960Fps12
- RgbResolution640x480Fps30
- YuvResolution640x480Fps15

Depth Sensor

Resolutions and framerate:

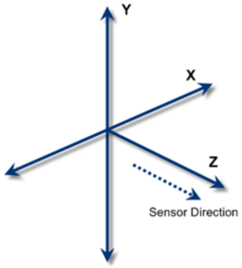
- 320x240, 30 Fps
- 640x480, 30 Fps
- 80x60, 30 Fps

Skeletons



- Up to 6 person recognized
- Up to 2 skeleton tracked
- 20 joints tracked
- Users facing the sensor are recognized better
- Default and seated Mode

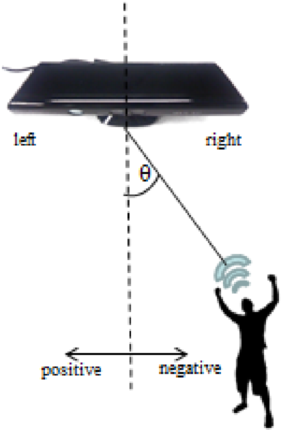
Accelerometer



- 3-axis accelerometer
- 2g range
- 3d vector pointing in the direction of gravity
- Right-handed coordinate system centred on the sensor

Microphone

- 4 microphones are used to detect the position of the sound source
- Supports noise suppression and echo cancellation
- Speech recognition via Microsoft.Speech API



Similar Devices

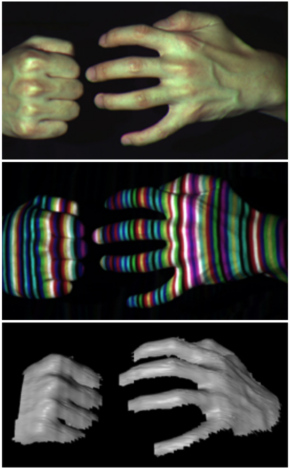

- Leapmotion
- Intel Perceptual
- Kinect 2
  - 1080p HD video
  - Wider/expanded field of view
  - Improved skeletal tracking

Recognizing Human Figure

Two stage process:

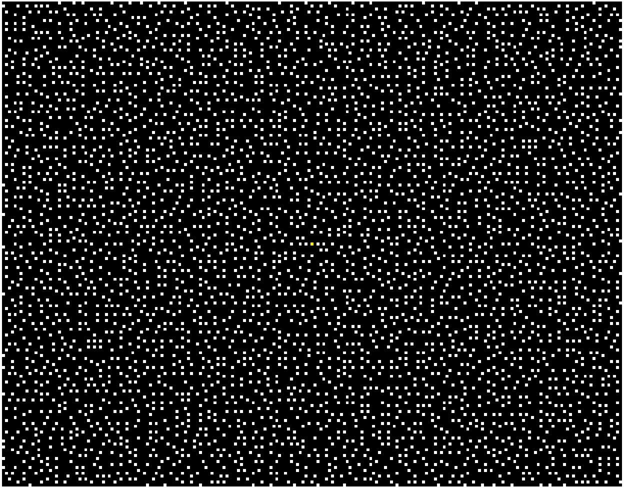
1. Compute depth map (structured light analysis)
2. Infer body position (machine learning)

Structured Light

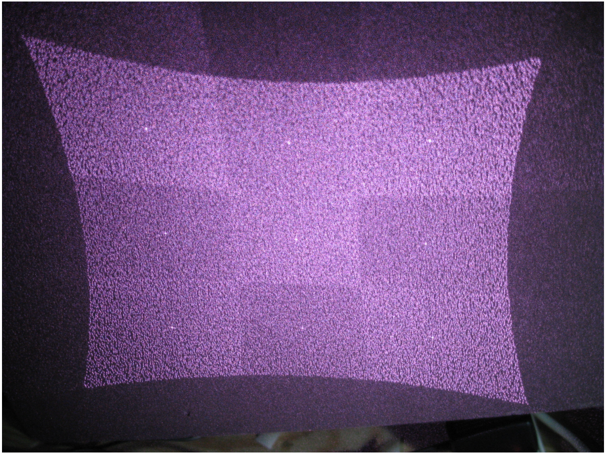


- The technique of analysing a known pattern
- Project a known pattern and infer depth from the deformation of that pattern

IR Pattern

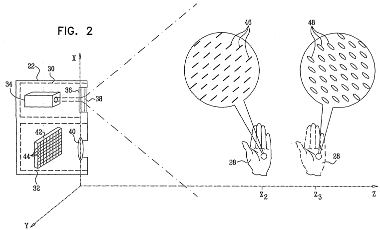


Kinect IR Pattern



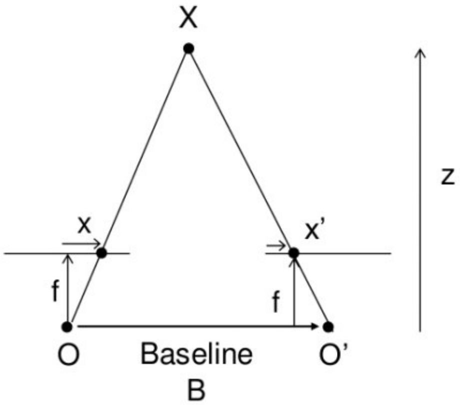
Frame From Focus

- More blur means more distance
- The kinect uses special "astigmatic" lens
- A projected circle then becomes an ellipse whose orientation depends on depth

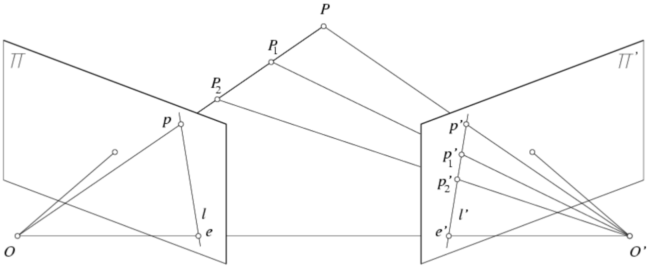


Stereo Images

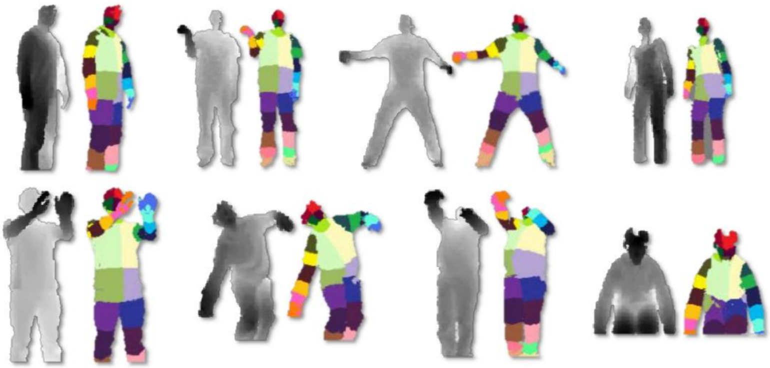
Recover depth by finding image coordinate x' that corresponds to x



Stereo Images From One Image



Body parts recognition

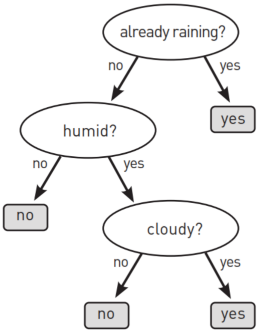


Segmentation

- Starts with 100,000 depth images with known skeletons
- For each real image, render dozens more using computer graphics techniques
- Learn a randomized decision forest, mapping depth images to body parts

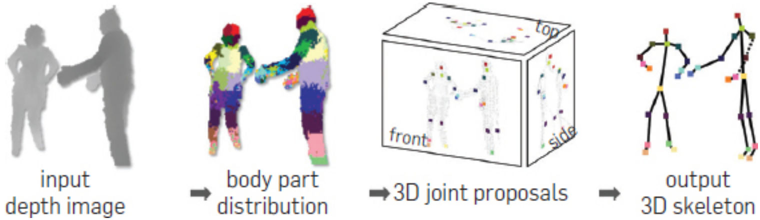
Classification

- Is that pixel "background"?
- How does the (normalized) depth at that pixel compare to this pixel?
- Outputs are actually probability distributions
- Distributed algorithm



The Skeleton

- Mean shift clustering to find center of mass
- Propose skeleton joints



Programming With The Kinect

A lot of APIs:

- OpenNI (R.I.P. Thanks to Apple since April)
- OpenKinect (http://openkinect.org/)
- Kinect SDK (http://www.microsoft.com/en-us/download/details.aspx?id=40278)

Enumerate Kinect sensors

```
private KinectSensor sensor;
...
  foreach (var potentialSensor in
     KinectSensor.KinectSensors)
  {
    if (potentialSensor.Status ==
       KinectStatus.Connected)
    {
      this.sensor = potentialSensor;
      break;
    }
  }
```

Enable Data Streaming

```
if(this.sensor != null)
{
this.sensor.ColorStream.Enable(
 ColorImageFormat.RgbResolution640x480Fps30);
this.sensor.DepthStream.Enable(
 DepthImageFormat.Resolution640x480Fps30);
this.sensor.SkeletonStream.Enable();
}
```

Starting the sensor

```
if (this.sensor != null)
{
  this.sensor.Start();
}
```

Registering and handling sensor stream

```
private byte[] colorPixels;
this.colorPixels = new
   byte[this.sensor.ColorStream
   .FramePixelDataLength];
...
if (this.sensor != null){
    this.sensor.ColorFrameReady +=
        this.SensorColorFrameReady;
}
```

Saving raw data

```
using (ColorImageFrame colorFrame =
    e.OpenColorImageFrame()){
        if (colorFrame != null){
          colorFrame.CopyPixelDataTo(
           this.colorPixels);
          ...
        }
    }
```

Getting a bitmap

```
private WriteableBitmap colorBitmap;
this.colorBitmap = new WriteableBitmap(
 this.sensor.ColorStream.FrameWidth,
 this.sensor.ColorStream.FrameHeight,
    96.0, 96.0, PixelFormats.Bgr32, null);
// Write the pixel data into our bitmap
if (colorFrame != null) {
      this.colorBitmap.WritePixels(
        new Int32Rect(0, 0,
           this.colorBitmap.PixelWidth,
           this.colorBitmap.PixelHeight),
        this.colorPixels,
        this.colorBitmap.PixelWidth *
           sizeof(int),0); }
```

Get Depth Pixel

```
private void DepthImageReady(object
   sender, DepthImageFrameReadyEventArgs
   e) {
     using (DepthImageFrame depthFrame =
        e.OpenDepthImageFrame()){
         if (depthFrame != null){
             depthFrame.
              CopyDepthImagePixelDataTo(
              this.depthPixels);
         }else{
             // depthFrame is null because
                the request did not arrive
                in time
         }
 }}
```

From Depth to RGB

```
int minDepth = depthFrame.MinDepth;
int maxDepth = depthFrame.MaxDepth;

int colorPixelIndex = 0;
for (int i = 0; i <
   this.depthPixels.Length; ++i) {
// Get the depth for this pixel
short depth = depthPixels[i].Depth;
byte intensity = (byte)(depth >= minDepth
   && depth <= maxDepth ? depth : 0);
```

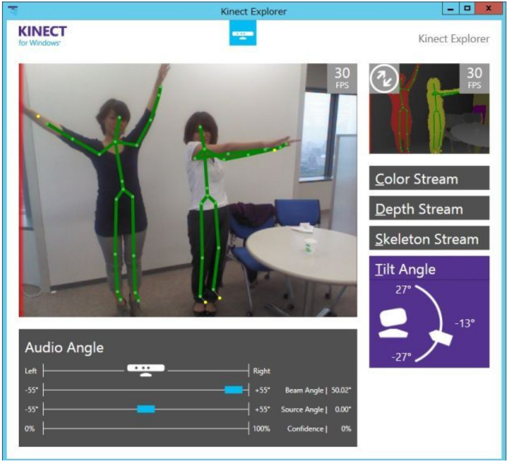
From Depth to RGB (2)

```
        this.colorPixels[colorPixelIndex++]
           = intensity;
        this.colorPixels[colorPixelIndex++]
           = intensity;
        this.colorPixels[colorPixelIndex++]
           = intensity;
        ++colorPixelIndex;
    }
```

Get Skeleton Data

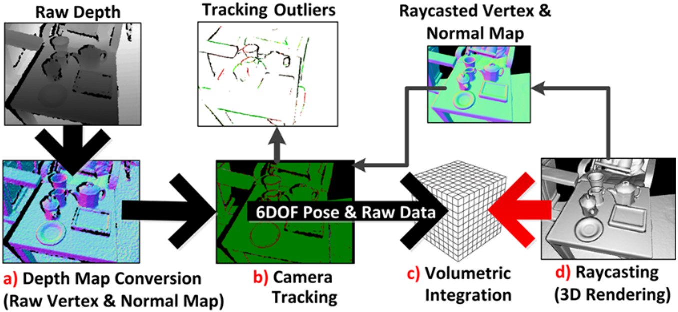
```
private void
   kinect_SkeletonFrameReady(object
   sender, SkeletonFrameReadyEventArgs e){
   // Open the Skeleton frame
      using (SkeletonFrame skeletonFrame =
         e.OpenSkeletonFrame()){
        // check that a frame is available
         if (skeletonFrame != null &&
            this.skeletonData != null) {
         // get the skeletal information in
            this frame
           skeletonFrame.CopySkeletonDataTo(
            this.skeletonData);
         }}}
```

Kinect Explorer

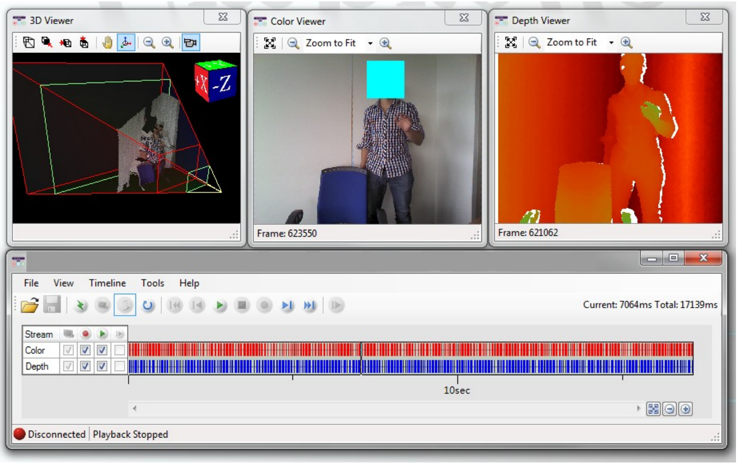


Fusion

3d object scanning and model creation using a Kinect



Kinect Studio



More...

Live Demo