introduction to

kinect

NUI, artificial intelligence
applications and programming

Matteo Valoriani
mvaloriani AT gmail.com
@MatteoValoriani

WHO I AM…

PhD Student at Politecnico of Milano

CEO of Fifth Element

Microsoft Speaker

Consultant

valoriani@elet.polimi.it

@MatteoValoriani

Follow me on Twitter or the Kitten gets it: @MatteoValoriani

Lots of words…

Smart device

Ambient Intelligence

Pervasive Computing

Augmented reality

Human-centered computing

Ubiquitous computing

Internet of Things

Physical Computing

… One concept

No more desktop-centered computation, but **distributed computation(“ubiquitous”)**

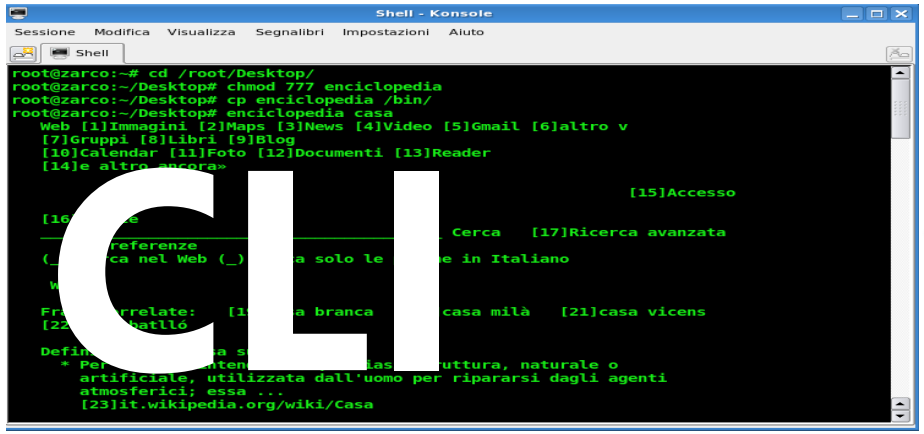
Objects become more “intelligent” and “smart”

New information’s model

New possibility of interaction with information

Machines fit the human environment instead of forcing humans to enter theirs

Interface Evolution



Command Line Interface

Graphical User Interface

Natural User Interface


Facial Recognitionpatial Recognition

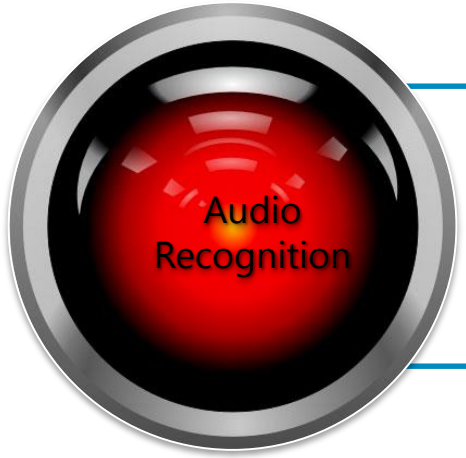

Augmented Reality


Gesture Sensing


Single Touch


Multi-Touch


Pen Input




Voice Command


Natural Speech


Geospatial Sensing


Accelerometers


Biometrics


Ambient Light


Mind control


Mood Recognitio

Kinect

It is a motion sensing input device by Microsoft for the Xbox 360 console (codename Project Natal).

It enables users to control and interact with the Xbox without the need to touch a game controller, through a natural user interface using gestures and spoken commands.

Launched in November 2010, sold 8 million units in 60 days

Kinect's magic

Immersive user experience

=

"Any sufficiently advanced technology is indistinguishable from magic" (Arthur C. Clarke)

Power Comes from the Sum

Any single technology on its own can create good experiences

The sum:
This is where the magic is

Tons of opportunities ahead

Application fields

Videogames

Natural User Interface

Augmented Reality

Medicina e Riabilitazione

Speech & Sign Language recognition

Learning support (also for impaired people)

Real-time conference & collaboration

3D scan & artwork

…

Video and examples available at:
http://www.microsoft.com/en-us/kinectforwindows/discover/gallery.aspx

Videos

http://www.xbox.com/en-US/Kinect/Kinect-Effect (rehabilitation)

http://www.youtube.com/watch?v=id7OZAbFaVI&feature=related (Medical use)

http://www.kinecthacks.com/kinect-interactive-hopscotch/ (game for children)

http://www.youtube.com/watch?v=9xMSGmjOZIg&feature=related (Holographic projection)

http://www.youtube.com/watch?v=1dnMsmajogA&feature=related (Kinect Glasses-less 3D)

http://www.youtube.com/watch?v=s0Fn6PyfJ0I&feature=related (Kinect Virtual Fashion, the Future of Shopping at Home)

http://www.youtube.com/watch?v=4V11V9Peqpc&feature=related (Kinect Projection mapping with box2D physics)

Videos (2)

http://www.youtube.com/watch?v=oALIuVb0NJ4 (Multidevice computing)

http://www.youtube.com/watch?v=-yxRTn3fj1g&feature=related (Kinect Touch wall)

http://www.youtube.com/watch?v=KBHgRcMPaYI&feature=related (Kinect window Bank of Moscow)

http://kinecthacks.net/motion-control-banking-is-so-easy-even-your-pet-can-do-it/ (Kinect Banking App Video)

http://www.youtube.com/watch?v=FMCIO0KNjrs (Art)

http://www.youtube.com/watch?v=g6N9Qid8Tqs&feature=related (Interactive LED Floor)

http://www.youtube.com/watch?v=c6jZjpvIio4 (Kinect in Education)

http://www.youtube.com/watch?v=_qvMHAvu-yc&feature=related (explore universe)

introduction to

kinect

Hardware and sensors

Matteo Valoriani
mvaloriani AT gmail.com
@MatteoValoriani

Hardware:

Depth resolution:
640x480 px

RGB resolution:
1600x1200 px

FrameRate:
60 FPS

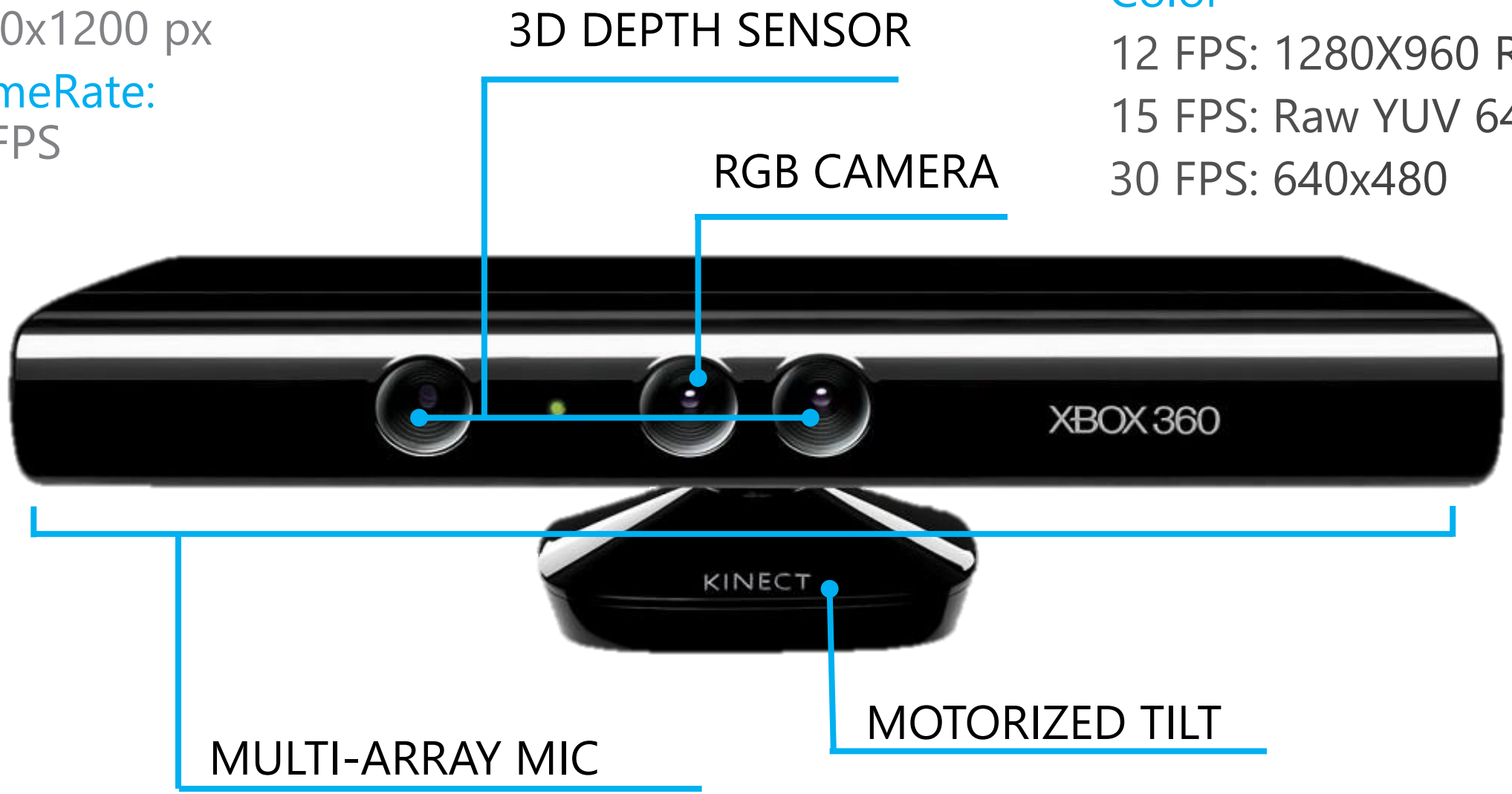
Software

Depth
30 FPS: 80x60, 320x240, 640x480

Color
12 FPS: 1280X960 RGB

15 FPS: Raw YUV 640x480

30 FPS: 640x480



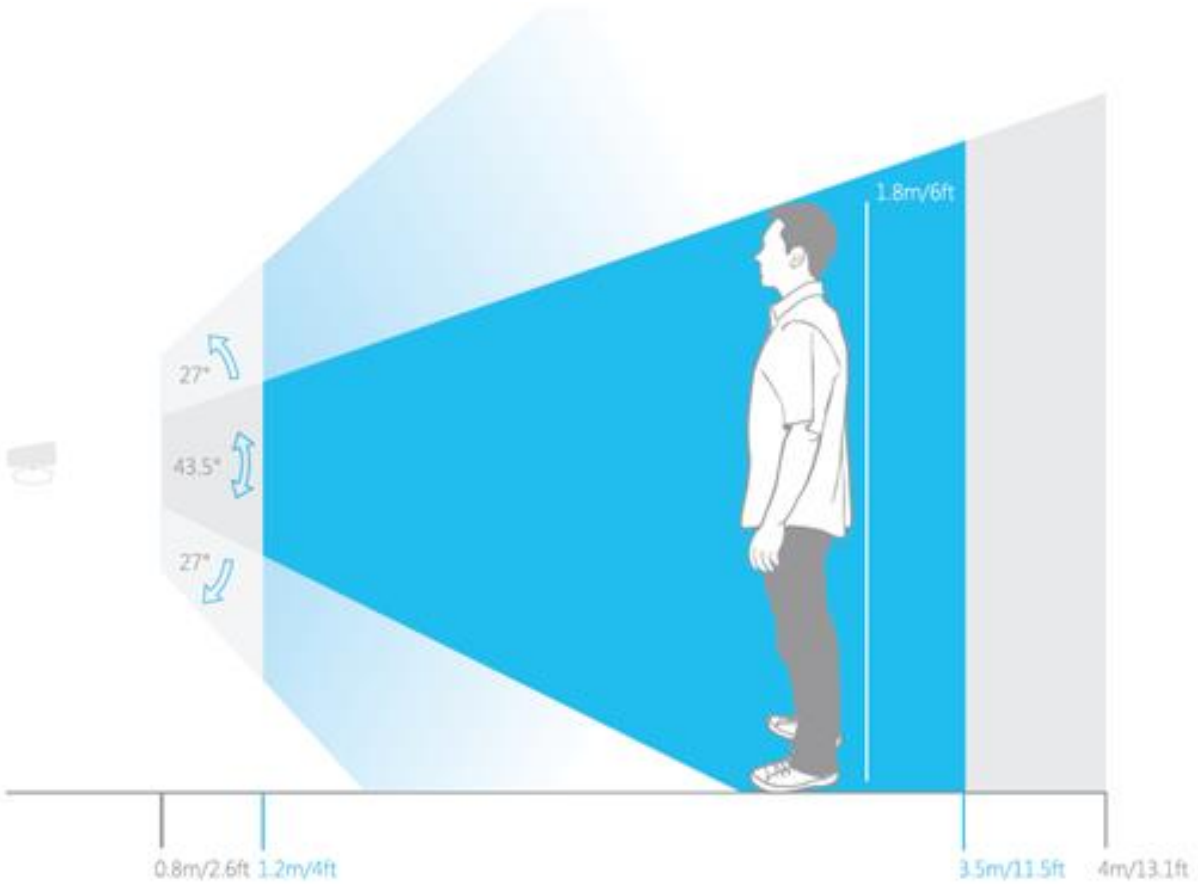
Kinect Sensors

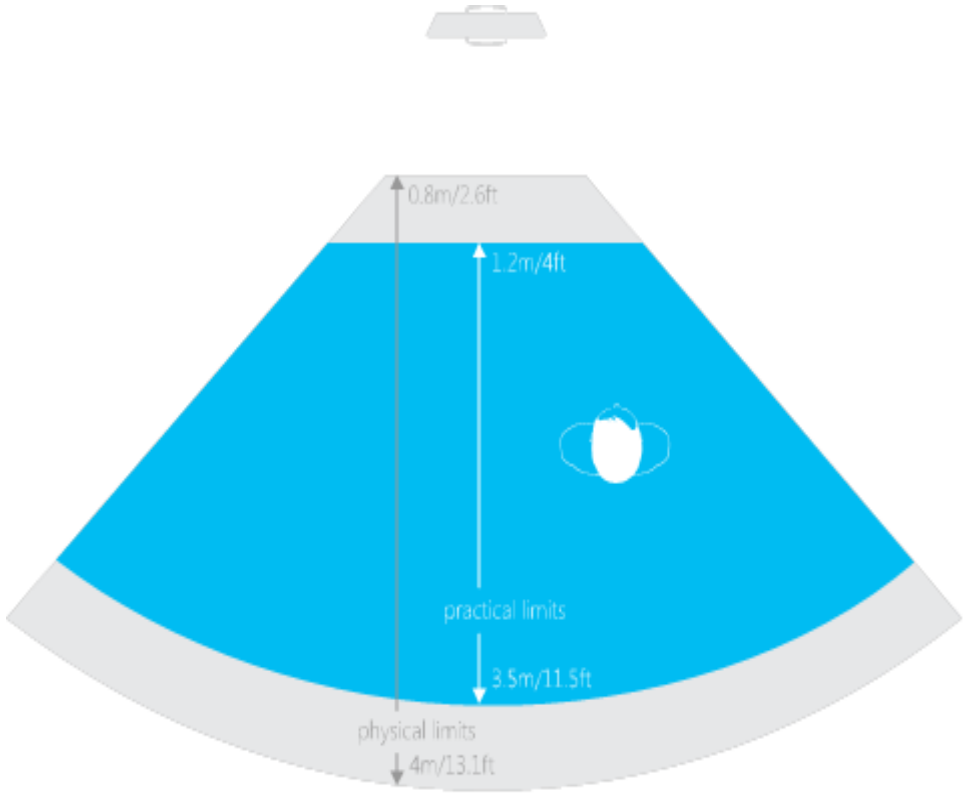
http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/1

Kinect Sensors

Field of View

43° vertical by 57° horizontal field of view



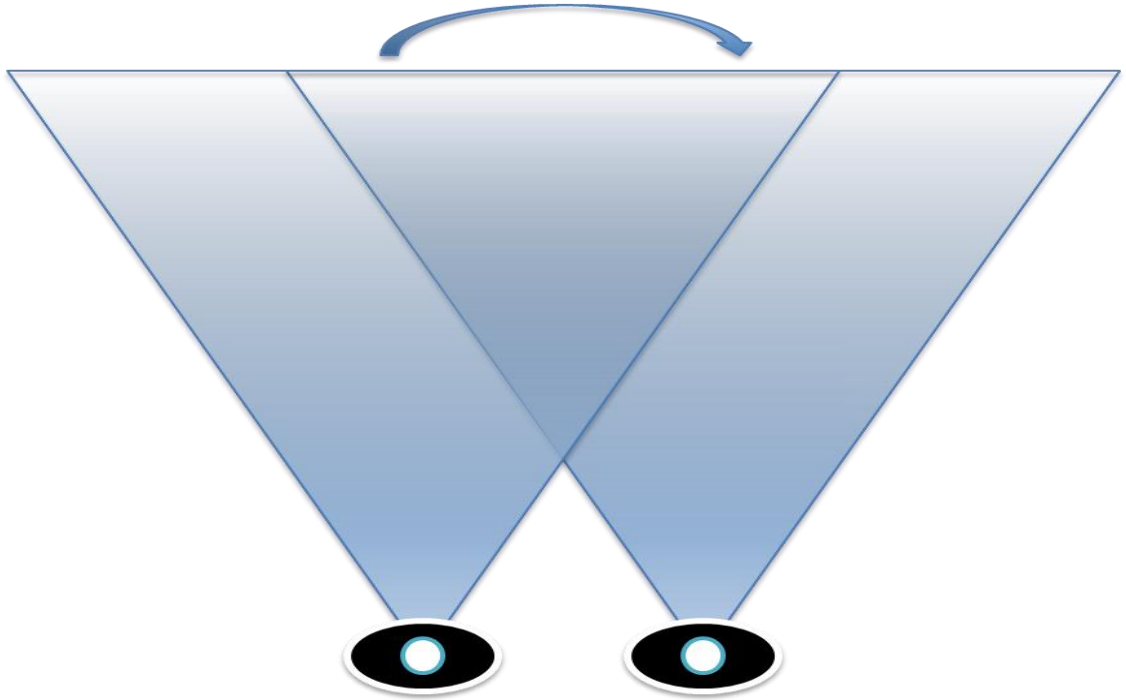


How does it work?

introduction to kinect

Depth Sensing

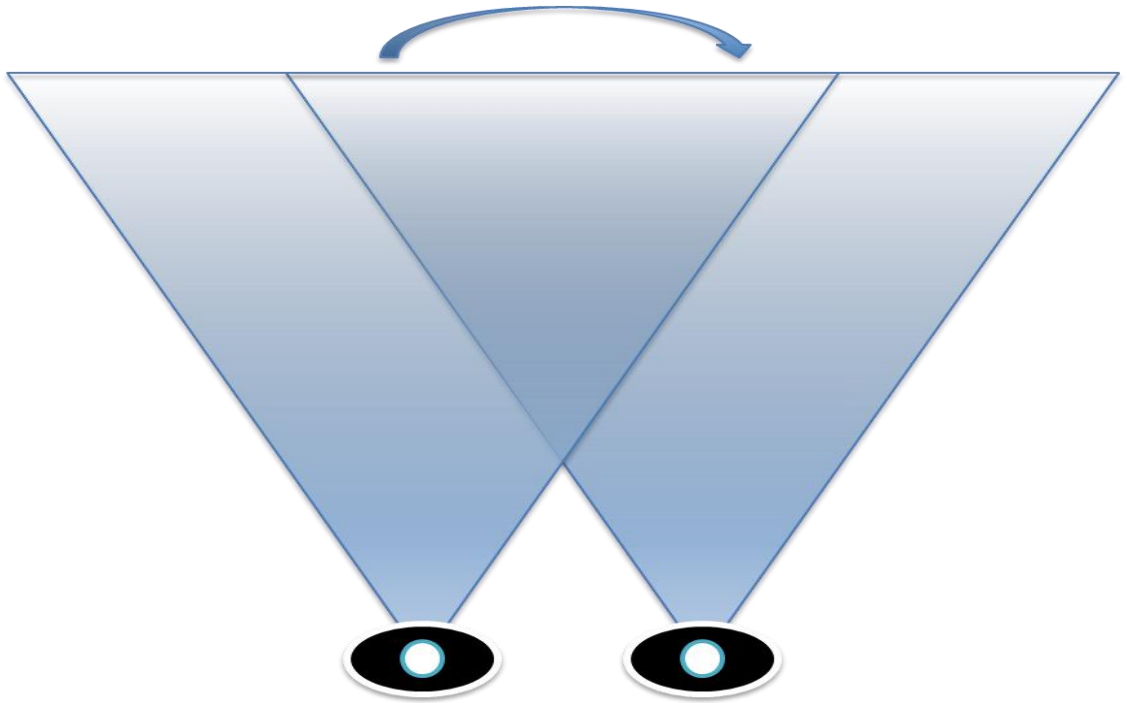
Object pattern similarity
determines disparity



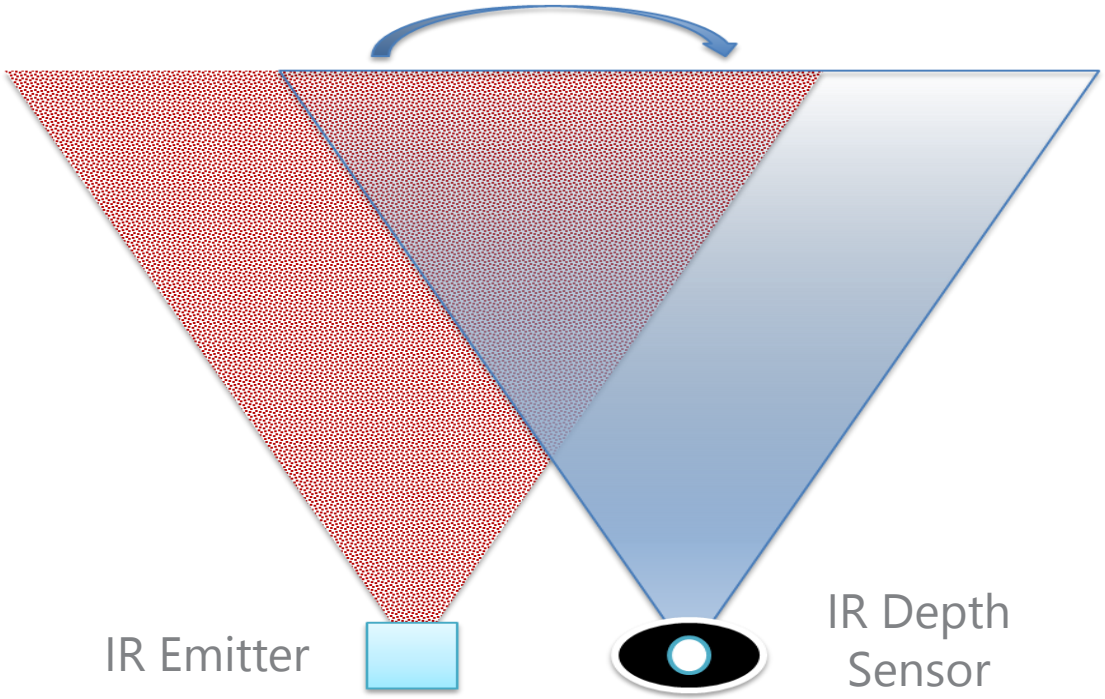

what does it
see?

Depth Sensing

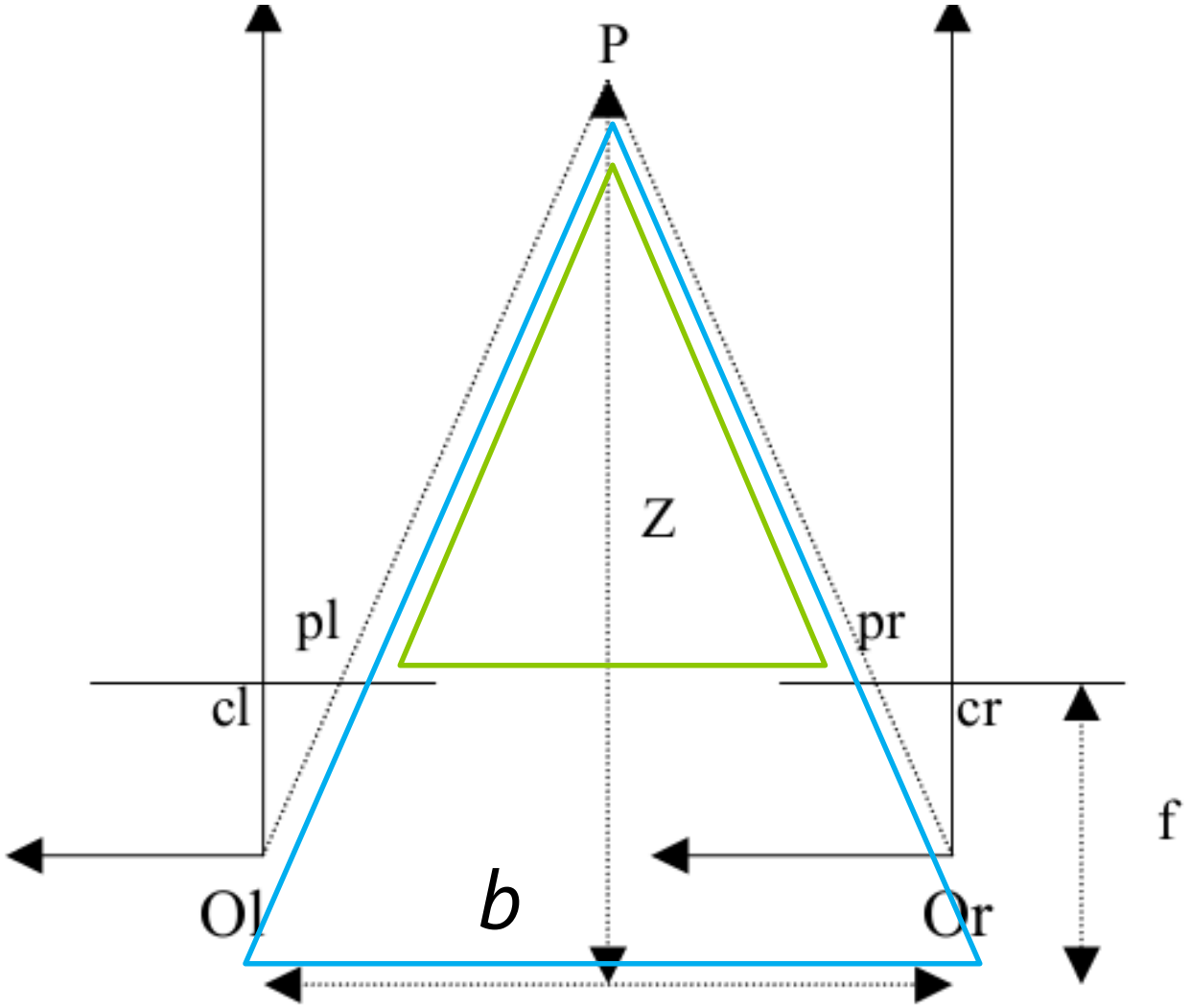
Object pattern similarity determines disparity



IR pattern similarity determines disparity



Mathematical Model



$$\frac{b+x_l-x_r}{Z-f}=\frac{b}{Z}$$

$$d=x_l-x_r$$

$$Z=\frac{b*f}{d}$$

Mathematical Model(2)

Reference plane distance

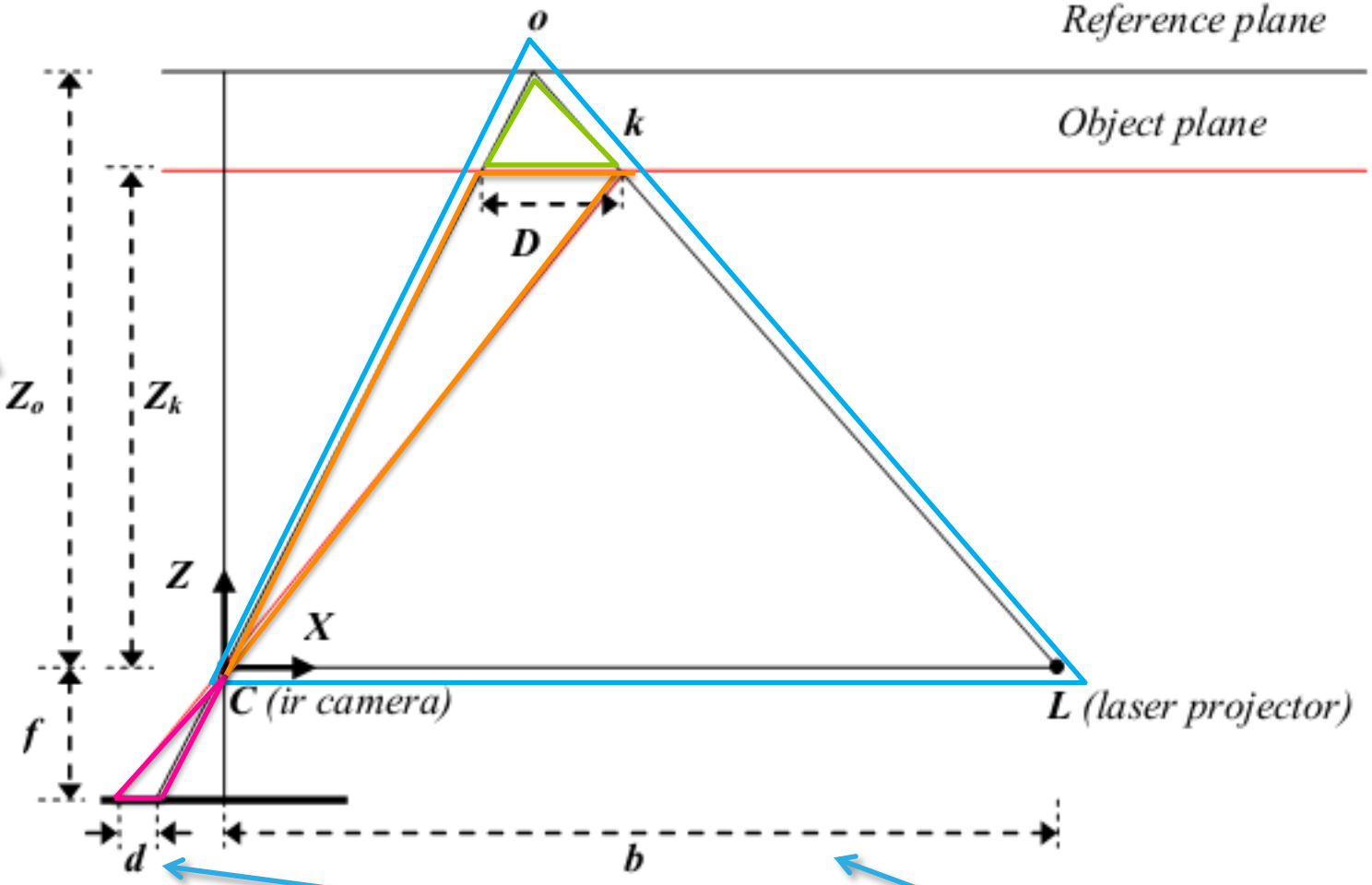
$$\frac{D}{b} = \frac{Z_o - Z_k}{Z_o}$$

$$\frac{d}{f} = \frac{D}{Z_k}$$

$$= Z_k = \frac{Z_o}{1 + \frac{Z_o}{fb} d}$$

$$X_k = -\frac{Z_k}{f}(x_k - x_o + \delta x)$$

$$Y_k = -\frac{Z_k}{f}(y_k - y_o + \delta y)$$



Precision

spatial x/y resolution

3mm @ 2m distance

depth z resolution

1cm @ 2m distance

10 cm @ 4m distance

operation range

0.8m ~ 4m | 0.5m ~ 3m

introduction to

kinect

Microsoft Kinect SDK 1.8

Matteo Valoriani
mvaloriani AT gmail.com
@MatteoValoriani

Kinect SDKs

Nov '10: Linux community release FreeKinect, the first driver that allows the use of the RGB and depth

Dec '10: PrimeSense release OpenNI, their open source SDK with motion tracking middleware called NITE

Jun '11: Microsoft release non-commercial Kinect beta SDK for Windows

Feb '12: Microsoft released first stable commercial Kinect SDK for Windows and specific device (Kinect for Windows)

OpenNI™

KINECT™
for Windows®

Microsoft SDK vs OpenNI

Microsoft SDK

Cons

no gesture recognition system

no support for others devices

only supports Win7/8 (x86 & x64)

SDK does not have events for when new user enters frame, leaves frame etc…

Pros

support for audio

support for motor/tilt

full body tracking:

does not need a calibration pose
includes head, hands, feet, clavicles
seems to deal better with occluded joints

supports multiple sensors

single installer

dedicated Runtime for client

SDK has events for when a new Video or new Depth frame is available

Microsoft SDK vs OpenNI

PrimeSense OpenNI/NITE

Cons

- no support for audio
- no support for motor/tilt (although you can simultaneously use the CL-NUI motor drivers)
  - lacks rotations for the head, hands, feet, clavicles
  - needs a calibration pose to start tracking (although it can be saved/loaded to/from disk for reuse)
  - occluded joints are not estimated
- supports multiple sensors although setup and enumeration is a bit quirky
- three separate installers and a NITE license string (although the process can be automated with my auto driver installer)
- SDK does not have events for when new Video or new Depth frames is available

Pros

- includes a framework for hand tracking
- includes a framework for hand gesture recognition
- can automatically align the depth image stream to the color image
  - also calculates rotations for the joints
  - support for hands only mode
- also supports the Primesense and the ASUS WAVI Xtion sensors
- supports Windows (including Vista&XP), Linux and Mac OSX
- support for record/playback to/from disk
- SDK has events for when new User enters frame, leaves frame etc

GET STARTED

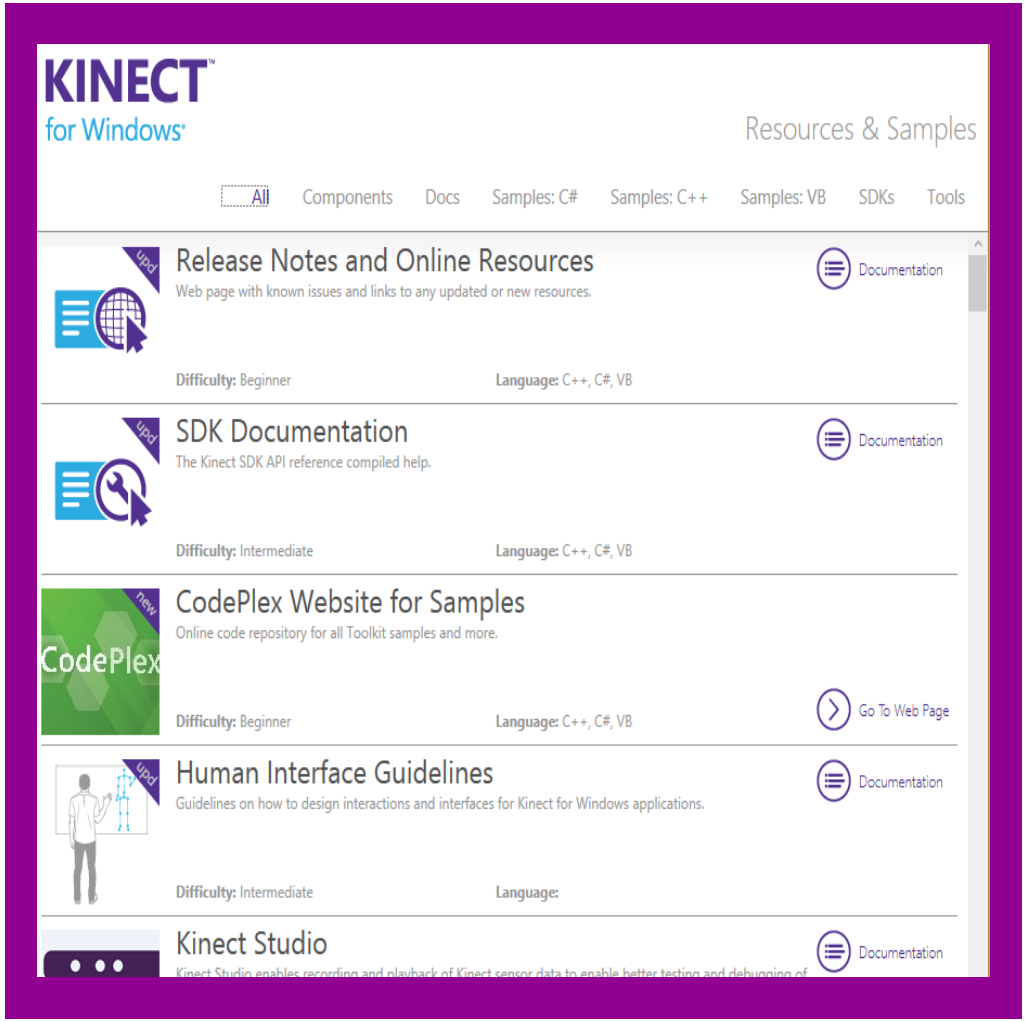
http://kinectforwindows.org

- Order Kinect Hardware
- Download Kinect SDK
- Kinect Toolkit Browser
- Install Kinect Studio
- Coding4Fun Toolkit

C#, C++ or VB languages supported

SDK 1.0 released on February 2013

- minor updates every 2 months
- major updates every 4 months



demo

Kinect Samples

Potential and applications

Depth sensor

Background removal
Object recognition

Skeletal tracking

Multi-user
Easy Gesture Recognition

Microphone array

Sound source detection
Speech recognition

KINECT API BASICS

Manage Kinect state

Connected

Enable Color, Depth, Skeleton

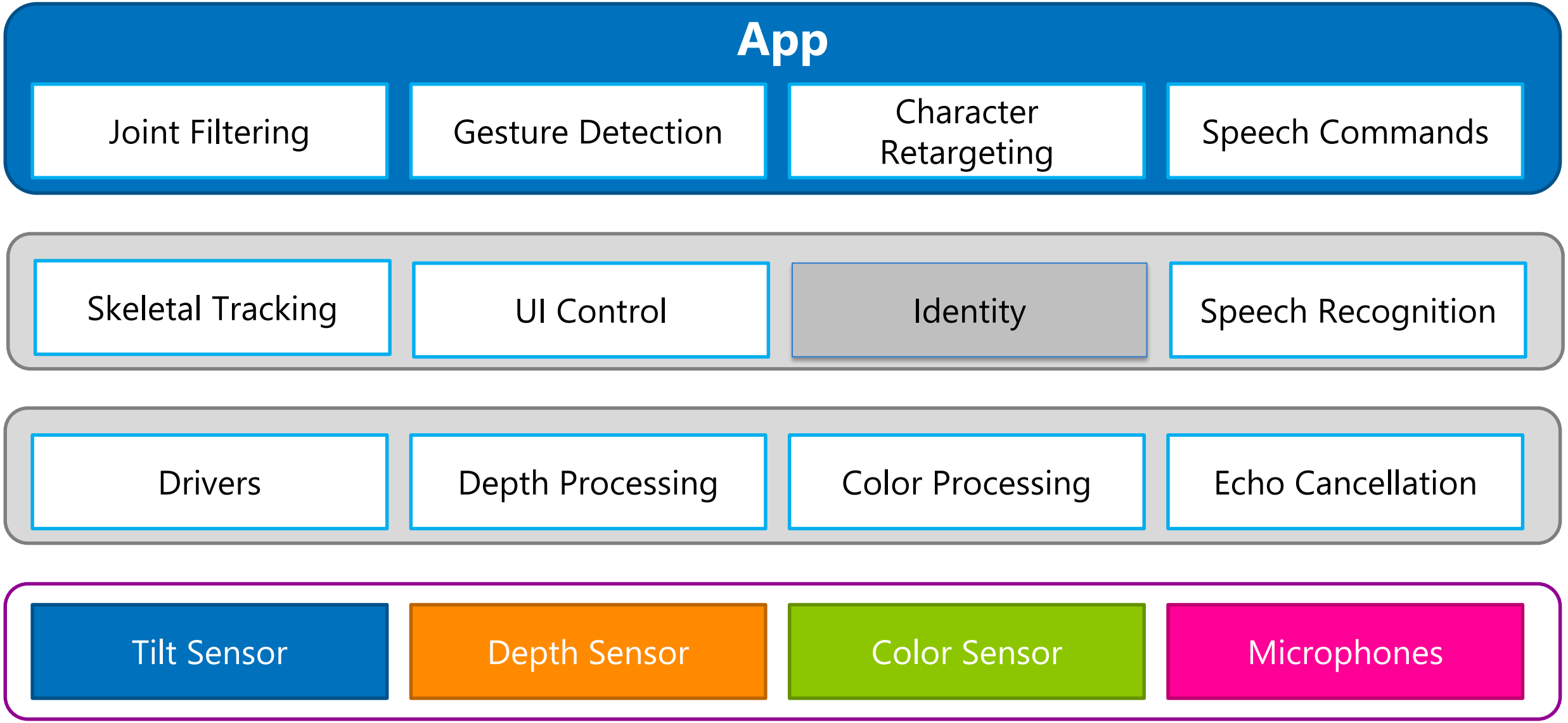
Start Kinect

Get Data

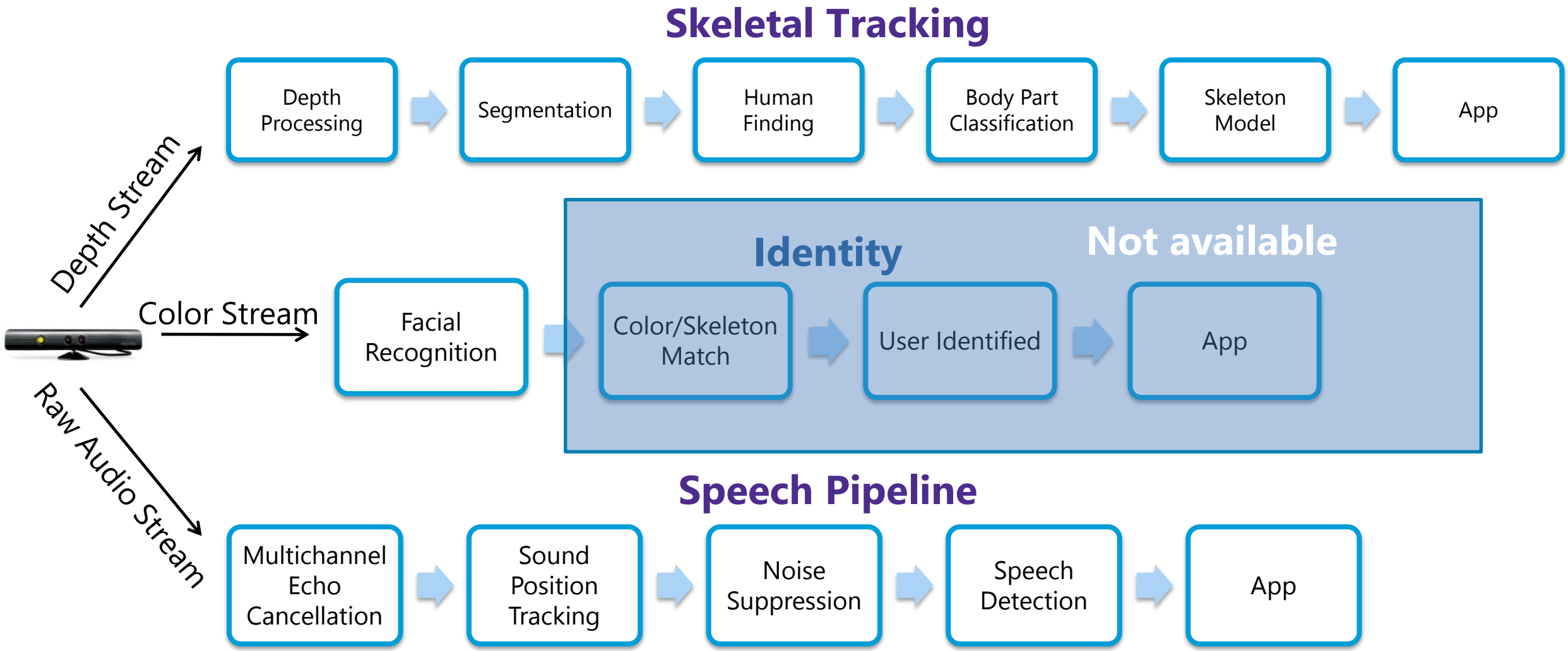
Events – AllFramesReady, ColorFrameReady, SkeletonFrameReady, DepthFrameReady,…

Polling – OpenNextFrame

The Kinect Stack



System Data Flow



code

Detecting a Kinect Sensor

```
private KinectSensor _Kinect;
public MainWindow() {
        InitializeComponent();
        this.Loaded  += (s, e) => { DiscoverKinectSensor(); };

    }

private void DiscoverKinectSensor()     {
        KinectSensor.KinectSensors.StatusChanged +=
                KinectSensors_StatusChanged;
        this.Kinect = KinectSensor.KinectSensors.FirstOrDefault(x =>
                      x.Status == KinectStatus.Connected);

    }
```

```
private void KinectSensors_StatusChanged(object sender, StatusChangedEventArgs e) {
    switch(e.Status) {
            case KinectStatus.Connected:
                if(this.Kinect == null)  {
                    this.Kinect = e.Sensor;
                }
                break;

            case KinectStatus.Disconnected:
                if(this.Kinect == e.Sensor) {
                    this.Kinect = null;
                    this.Kinect = KinectSensor.KinectSensors
                                  .FirstOrDefault(x => x.Status ==
KinectStatus.Connected);

                    if(this.Kinect == null){
                    //Notify the user that the sensor is disconnected
               }
                }
                break;
            //Handle all other statuses according to needs
        }     }
```

```
public KinectSensor Kinect
    {
        get { return this._Kinect; }
        set {
            if(this._Kinect != value) {
                if(this._Kinect != null) {
                    //Uninitialize
                    this._Kinect = null;
                }

                if(value != null && value.Status == KinectStatus.Connected)
{
                    this._Kinect = value;
                    //Initialize
                }
            }
        }
    }
```

KinectStatus VALUES

KinectStatus	What it means
Undefined	The status of the attached device cannot be determined.
Connected	The device is attached and is capable of producing data from its streams.
DeviceNotGenuine	The attached device is not an authentic Kinect sensor.
Disconnected	The USB connection with the device has been broken.
Error	Communication with the device produces errors.
Error Initializing	The device is attached to the computer, and is going through the process of connecting.
InsufficientBandwidth	Kinect cannot initialize, because the USB connector does not have the necessary bandwidth required to operate the device.
NotPowered	Kinect is not fully powered. The power provided by a USB connection is not sufficient to power the Kinect hardware. An additional power adapter is required.
NotReady	Kinect is attached, but is yet to enter the Connected state.

code

Move the camera

Tilt

```
private void setAngle(object sender, RoutedEventArgs e){
    if (Kinect != null) {
            Kinect.ElevationAngle = (Int32)slider1.Value;    }     }
```

```
<Slider Height="33" HorizontalAlignment="Left" Margin="0,278,0,0"
Name="slider1" VerticalAlignment="Top" Width="308" SmallChange="1
IsSnapToTickEnabled="True" />
```

```
<Button Content="OK" Height="29" HorizontalAlignment="Left"
Margin="396,278,0,0" Name="button1" VerticalAlignment="Top"
Width="102" Click="setAngle" />
```

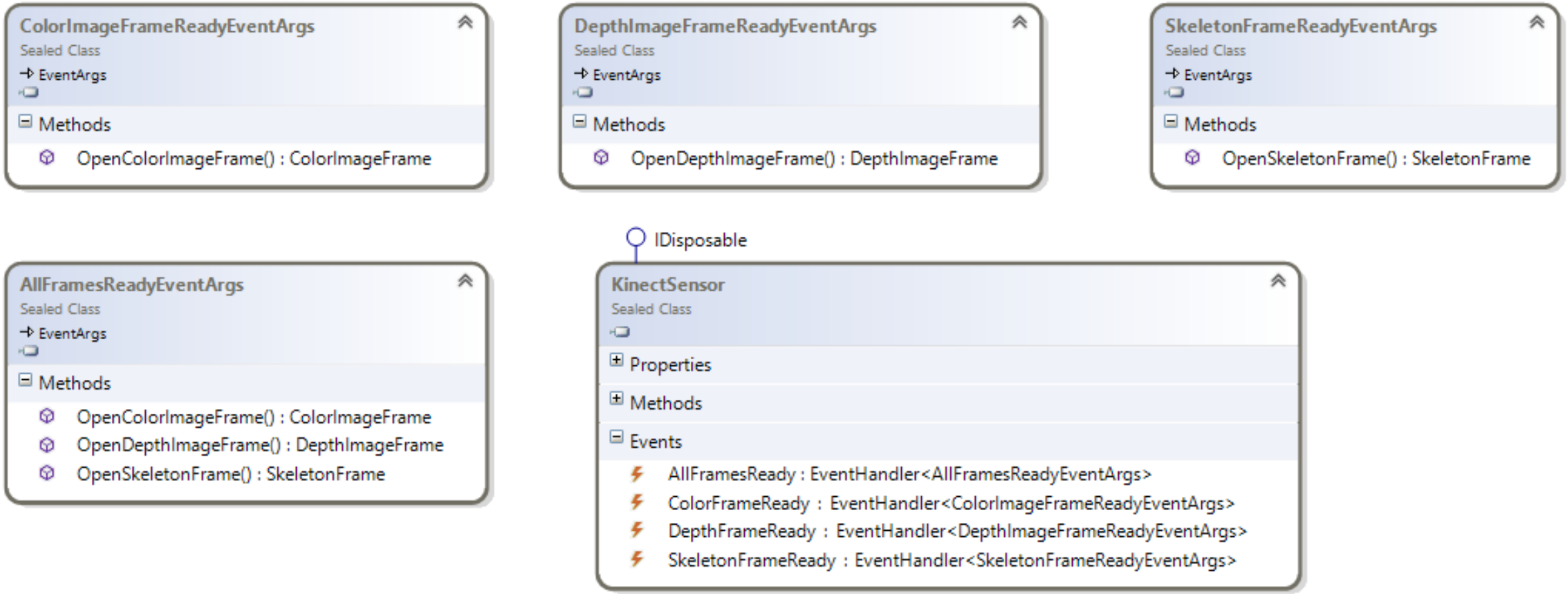
introduction to

kinect

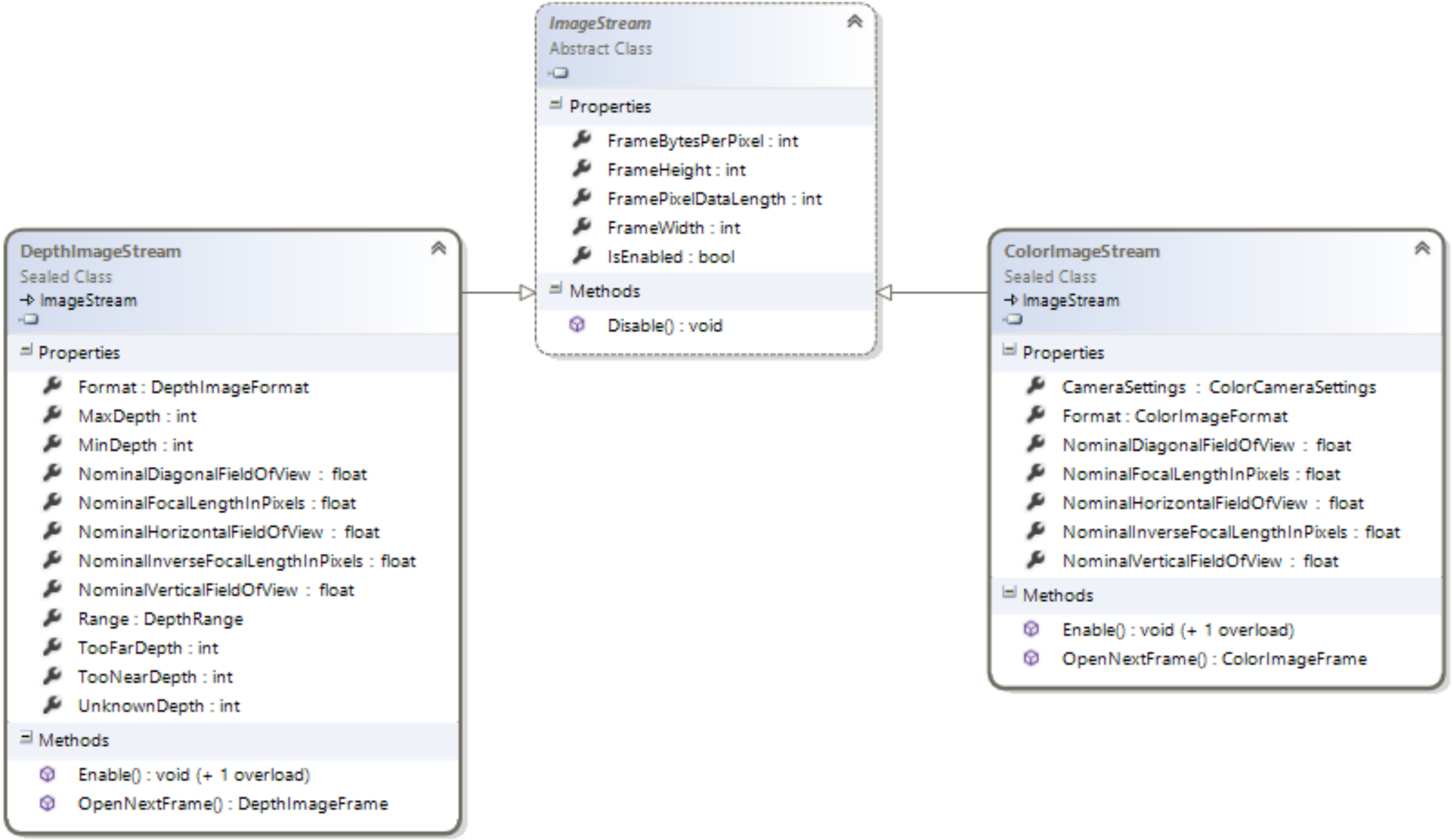
Camera Fundamentals

Matteo Valoriani
mvaloriani AT gmail.com
@MatteoValoriani

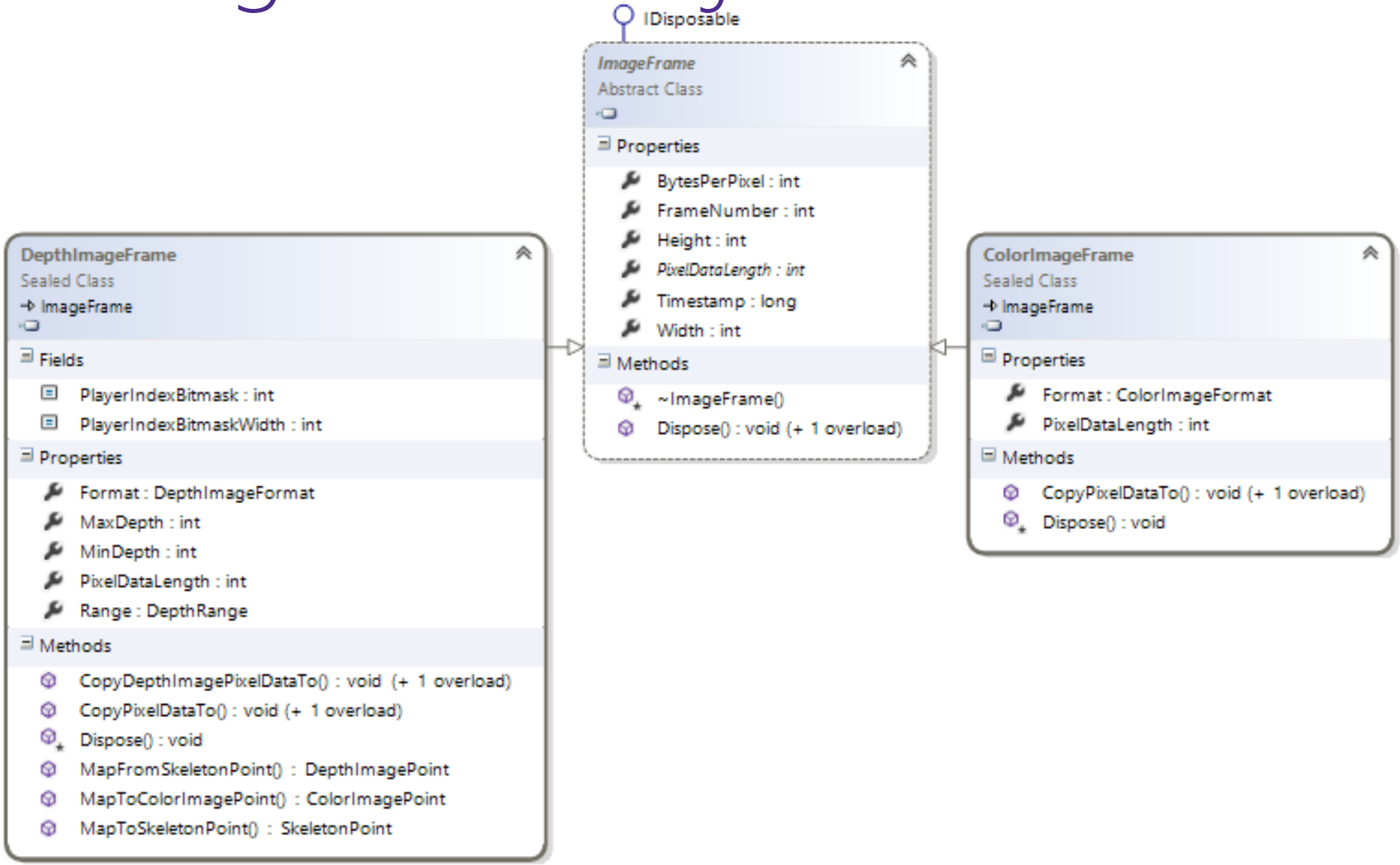
Cameras Events



The ImageStream object model



The ImageFrame object model



ColorImageFormat

Member name	Description
InfraredResolution640x480Fps30	16 bits, using the top 10 bits from a **PixelFormats.Gray16** format (with the 6 least significant bits always set to 0) whose resolution is 640 x 480 and frame rate is 30 frames per second. *Introduced in 1.6.*
RawBayerResolution1280x960Fps12	Bayer data (8 bits per pixel, layout in alternating pixels of red, green and blue) whose resolution is 1280 x 960 and frame rate is 12 frames per second. *Introduced in 1.6.*
RawBayerResolution640x480Fps30	Bayer data (8 bits per pixel, layout in alternating pixels of red, green and blue) whose resolution is 640 x 480 and frame rate is 30 frames per second. *Introduced in 1.6.*
RawYuvResolution640x480Fps15	Raw YUV data whose resolution is 640 x 480 and frame rate is 15 frames per second.
RgbResolution1280x960Fps12	RBG data whose resolution is 1280 x 960 and frame rate is 12 frames per second.
RgbResolution640x480Fps30	RBG data whose resolution is 640 x 480 and frame rate is 30 frames per second.
YuvResolution640x480Fps15	YUV data whose resolution is 640 x 480 and frame rate is 15 frames per second.
Undefined	The format is not defined.

```
colorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
```

DepthImageFormat

Member name	Description
Resolution320x240Fps30	The resolution is 320 x 240; the frame rate is 30 frames per second.
Resolution640x480Fps30	The resolution is 640 x 480; the frame rate is 30 frames per second.
Resolution80x60Fps30	The resolution is 80 x 60; the frame rate is 30 frames per second.
Undefined	The format is not defined.

```
depthStream.Enable(DepthImageFormat.Resolution640x480Fps30);
```

BYTES PER PIXEL

The stream Format determines the pixel format and therefore the meaning of the bytes.

ColorImageFormat.RgbResolution640x480Fps30 → pixel format is Bgra32→ 32 bits (4 bytes) per pixel.

The 1st byte is the blue channel value, the 2nd is green, and the 3th is red. The fourth byte determines the alpha or opacity of the pixel (unused)

640x480 = height * width * BytesPerPixel = 640 * 480 * 4 = 122880 bytes

Stride is the number of bytes for a row of pixels

Stride = width * BytesPerPixel

Depth data

Returns the distance and player for every pixel

Ex: 320x240 = 76,800 pixels

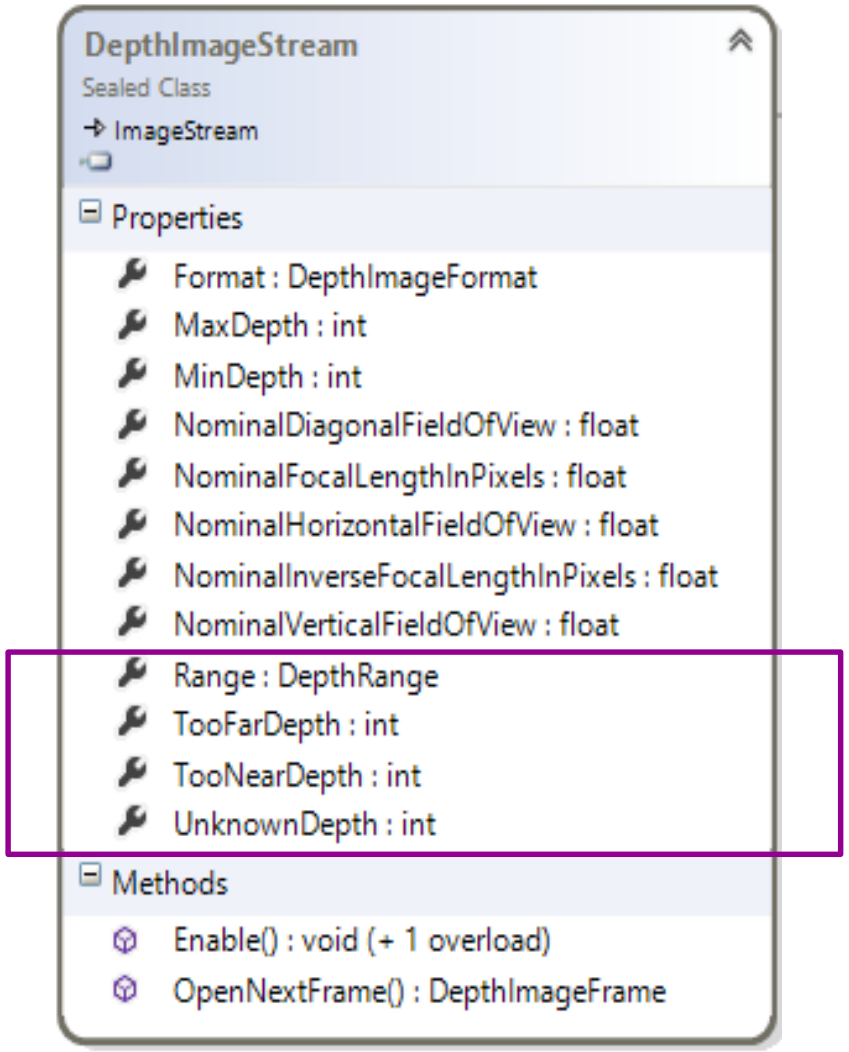
Distance

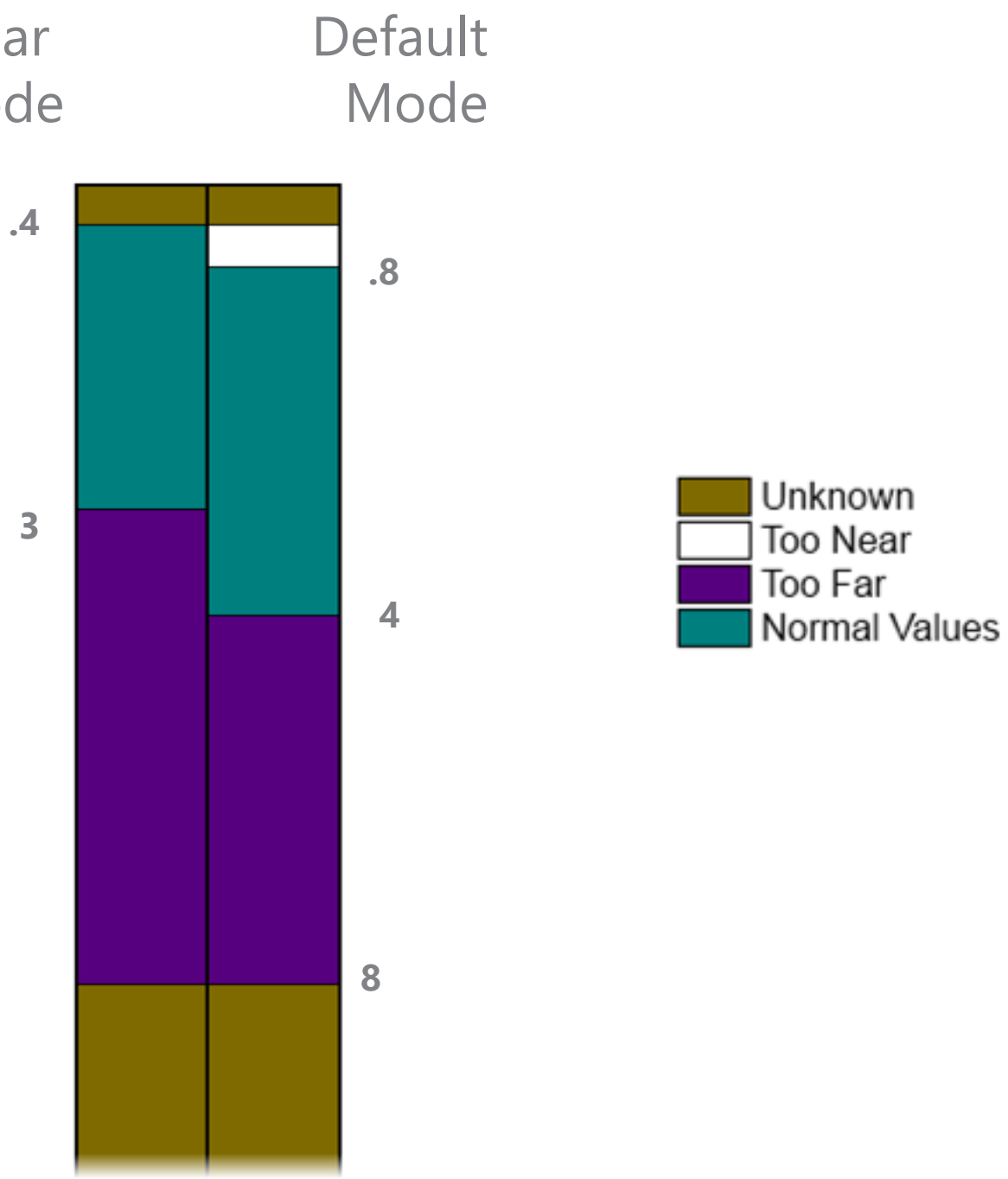
Distance in mm from Kinect ex: 2,000mm

Player

1-6 players

Depth Range


DepthImageStream
Sealed Class
ImageStream
Properties
Format : DepthImageFormat
MaxDepth : int
MinDepth : int
NominalDiagonalFieldOfView : float
NominalFocalLengthInPixels : float
NominalHorizontalFieldOfView : float
NominalInverseFocalLengthInPixels : float
NominalVerticalFieldOfView : float
Range : DepthRange
TooFarDepth : int
TooNearDepth : int
UnknownDepth : int
Methods
Enable() : void (+ 1 overload)
OpenNextFrame() : DepthImageFrame


Near Mode
Default Mode
.4
3
.8
4
8
Unknown
Too Near
Too Far
Normal Values

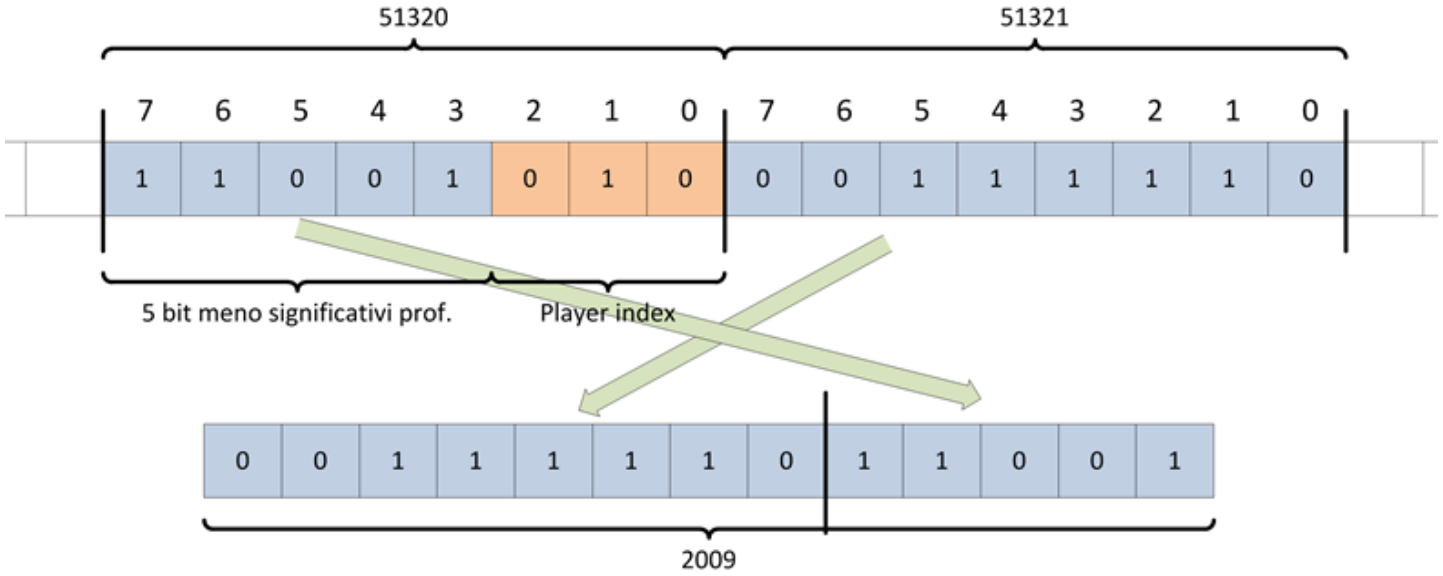
Depth data

Distance Formula

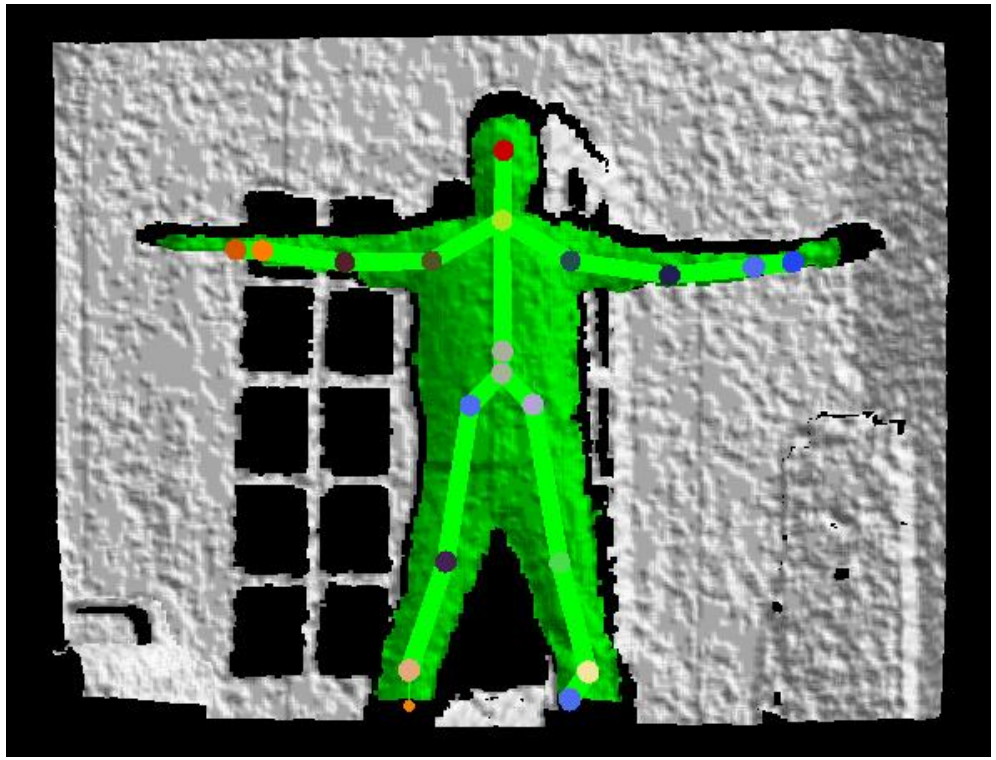
```
int depth = depthPoint >> DepthImageFrame.PlayerIndexBitmaskWidth;
```

Player Formula

```
int player = depthPoint & DepthImageFrame.PlayerIndexBitmask;
```



Depth and Segmentation map



code

Processing & Displaying a Color Data

```
private WriteableBitmap _ColorImageBitmap;
private Int32Rect _ColorImageBitmapRect;
private int _ColorImageStride;

private void InitializeKinect(KinectSensor sensor) {

 if (sensor != null){
         ColorImageStream colorStream = sensor.ColorStream;
         colorStream.Enable();

            this._ColorImageBitmap = new WriteableBitmap(colorStream.FrameWidth,
                                                         colorStream.FrameHeight, 96, 96,
                                                         PixelFormats.Bgr32, null);
            this._ColorImageBitmapRect = new Int32Rect(0, 0, colorStream.FrameWidth,
                                                       colorStream.FrameHeight);
            this._ColorImageStride = colorStream.FrameWidth * colorStream.FrameBytesPerPixel;
            ColorImageElement.Source = this._ColorImageBitmap;

            sensor.ColorFrameReady += Kinect_ColorFrameReady;
            sensor.Start();
            }
}
```

```
private void Kinect_ColorFrameReady(object sender, ColorImageFrameReadyEventArgs e)
        {
            using (ColorImageFrame frame = e.OpenColorImageFrame())
            {
                if (frame != null)
                {
                    byte[] pixelData = new byte[frame.PixelDataLength];
                    frame.CopyPixelDataTo(pixelData);

                    this._ColorImageBitmap.WritePixels(this._ColorImageBitmapRect,
                                                       pixelData,
                                                                   this._ColorImageStride, 0);
                }
            }
        }
```

code

Taking a Picture

```
private void TakePictureButton_Click(object sender, RoutedEventArgs e)  {
        string fileName = "snapshot.jpg";

        if (File.Exists(fileName))  {
            File.Delete(fileName);
        }

        using (FileStream savedSnapshot = new FileStream(fileName, FileMode.CreateNew))  {
            BitmapSource image = (BitmapSource)VideoStreamElement.Source;

            JpegBitmapEncoder jpgEncoder = new JpegBitmapEncoder();
            jpgEncoder.QualityLevel = 70;
            jpgEncoder.Frames.Add(BitmapFrame.Create(image));
            jpgEncoder.Save(savedSnapshot);

            savedSnapshot.Flush();
            savedSnapshot.Close();
            savedSnapshot.Dispose();
        }
    }
```

code

Processing & Displaying a DepthData

```
Kinect.DepthStream.Enable(DepthImageFormat.Resolution320x240Fps30);
Kinect.DepthFrameReady += Kinect_DepthFrameReady;

void Kinect_DepthFrameReady(object sender, DepthImageFrameReadyEventArgs e) {

   using (DepthImageFrame frame = e.OpenDepthImageFrame()) {

       if (frame != null)  {
           short[] pixelData = new short[frame.PixelDataLength];
           frame.CopyPixelDataTo(pixelData);

           int stride = frame.Width * frame.BytesPerPixel;
           ImageDepth.Source = BitmapSource.Create(frame.Width,
              frame.Height, 96, 96,
       PixelFormats.Gray16, null, pixelData,                    stride);
       } } }
}
```

introduction to

kinect

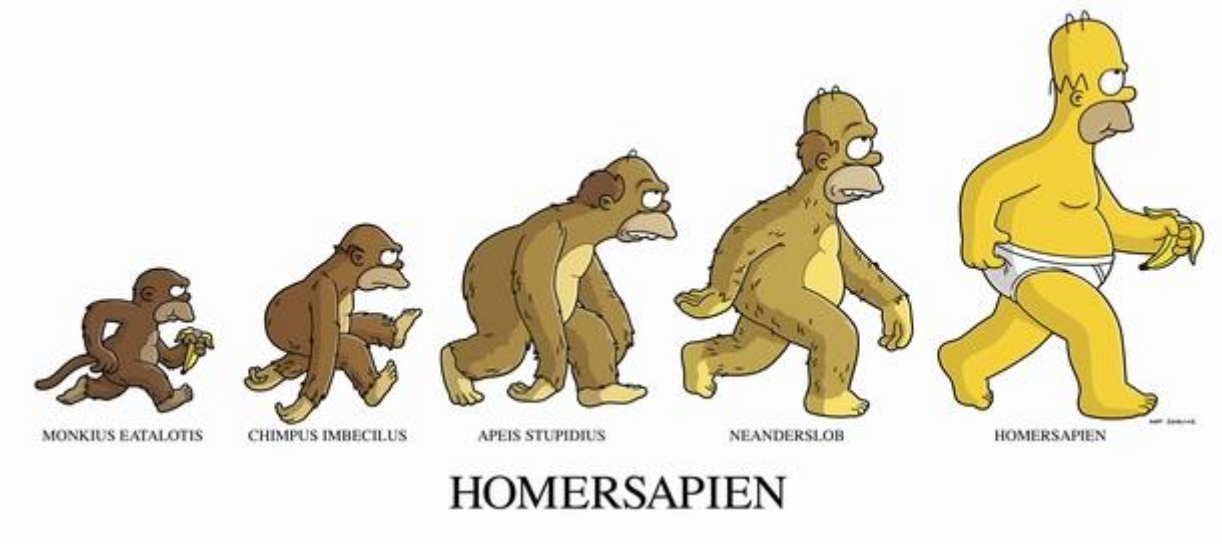
Skeletal Tracking Fundamentals

Matteo Valoriani
mvaloriani AT gmail.com
@MatteoValoriani

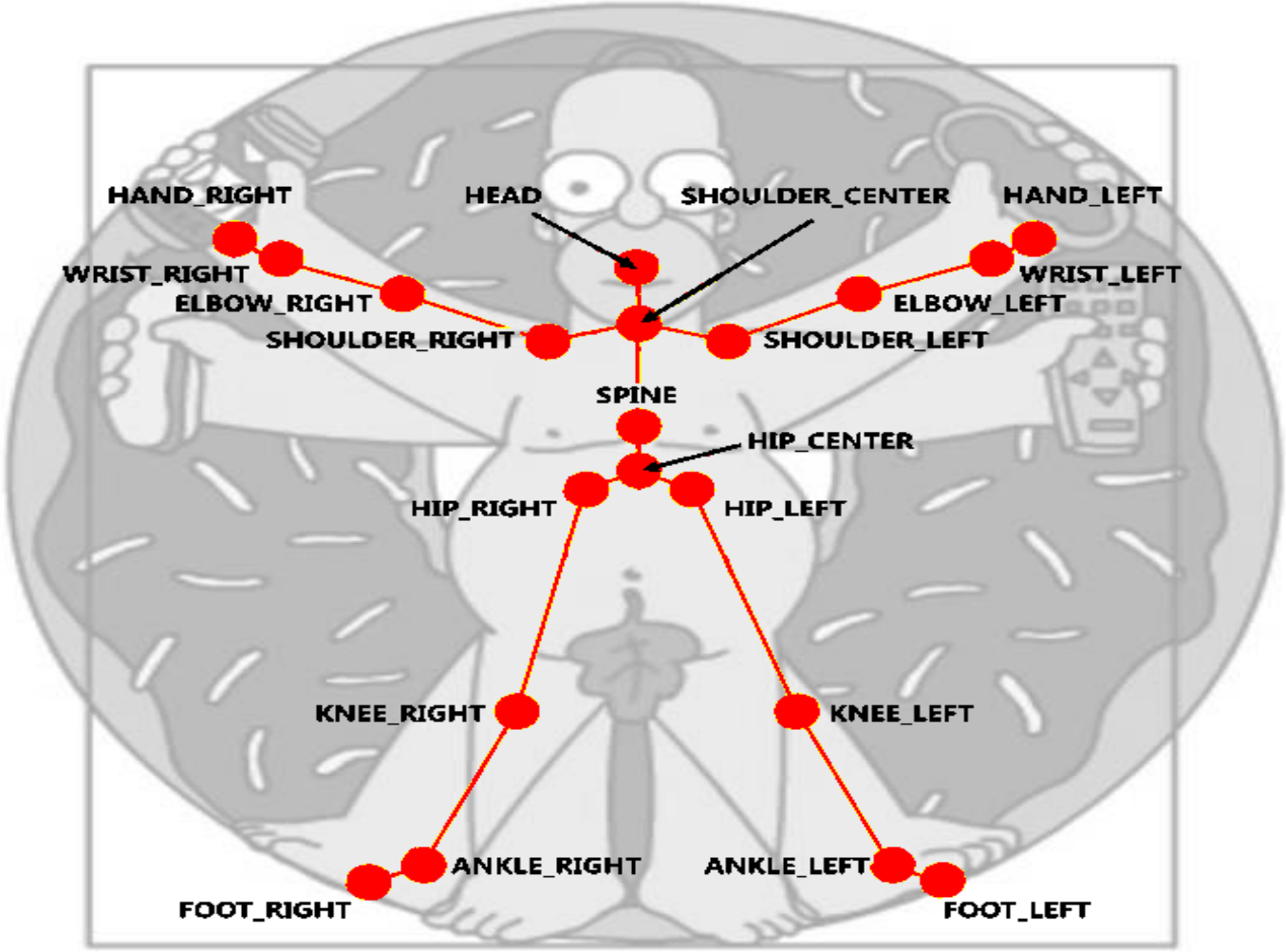
Skeletal Tracking History

First experiments started over 4 years ago:

Started with tracking just hands
Then head and hands
Then added shoulders and elbows
Then built “heavy duty” pipeline
To today’s pipeline that tracks 20 joints and runs fast

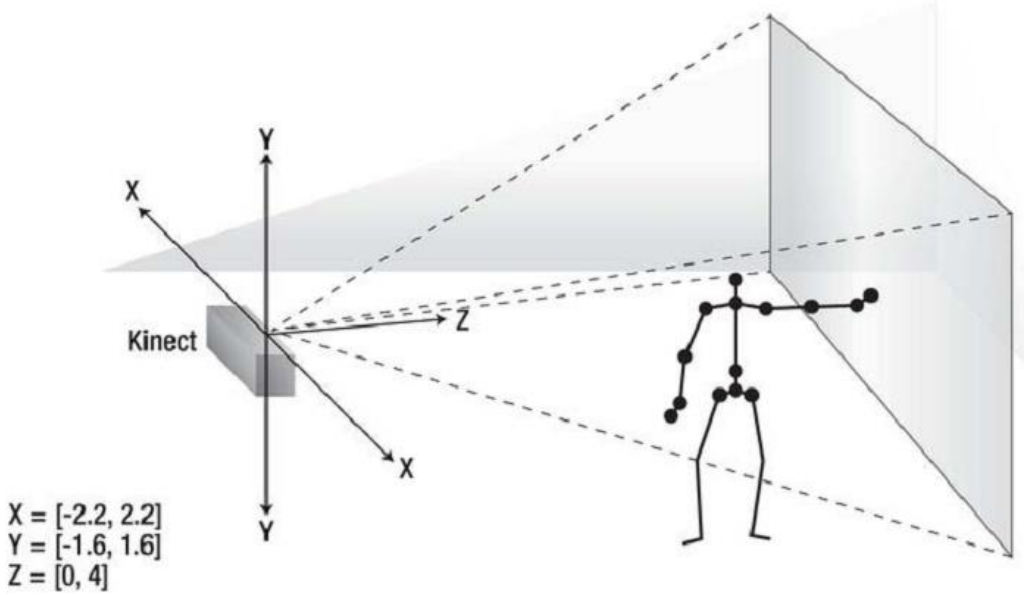


Skeleton Data



Maximum two players tracked at once

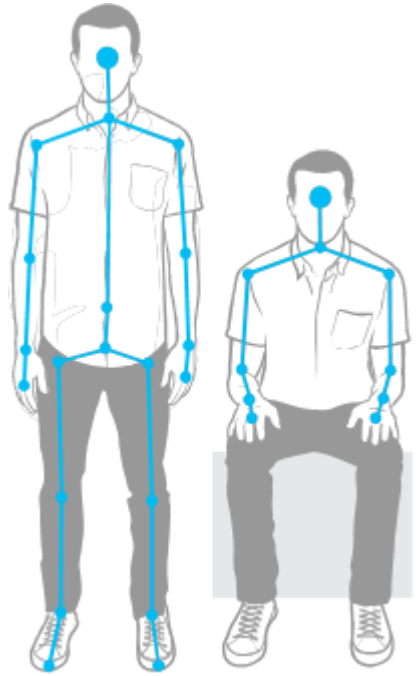
Six player proposals per Kinect



Tracking Modes

Default Mode

- 20 skeleton joints
- Optimized to recognize and track people who are standing and fully visible to the sensor
- Detects the user based on the distance of the subject from the background



Seated Mode

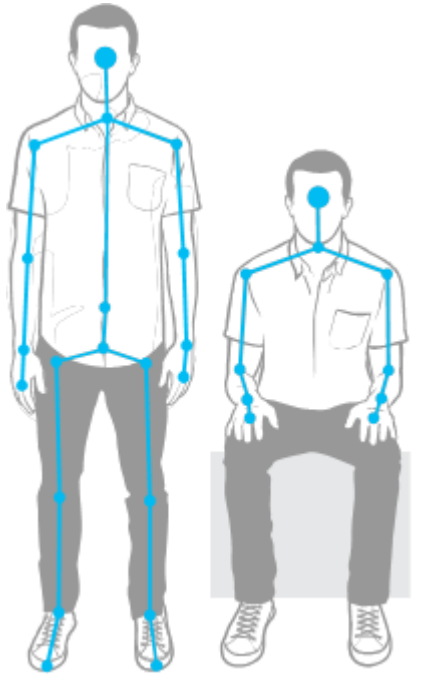
- 10 skeleton joints (upper-body joints)
- designed to track people who are seated or whose lower body is not entirely visible to the sensor
- uses movement to detect the user and distinguish him or her from the background
- uses more resources than the default pipeline
- provides the best way to recognize a skeleton when the Kinect depth sensor is in near range.

Tracking Modes Details

Only one tracking mode can be used at a time.

Different Kinect sensors can use different modes, if they are using different processes.

Toggling between default and seated mode resets skeletal tracking briefly. When the users are relocked, each tracked user will be assigned a new tracking ID.

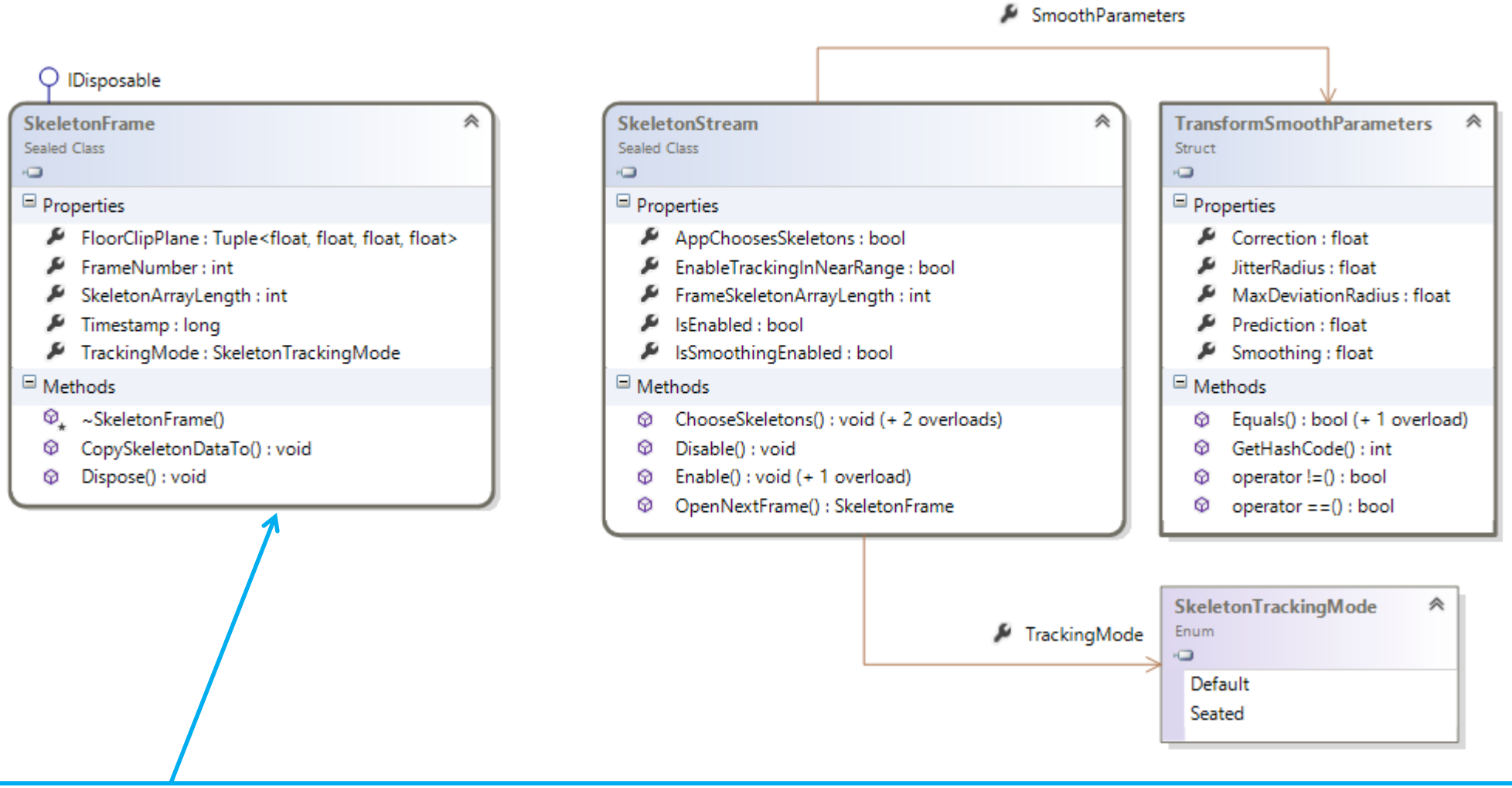


Traking in Near Mode

With SDK 1.5, an application can receive full joint information also in near mode

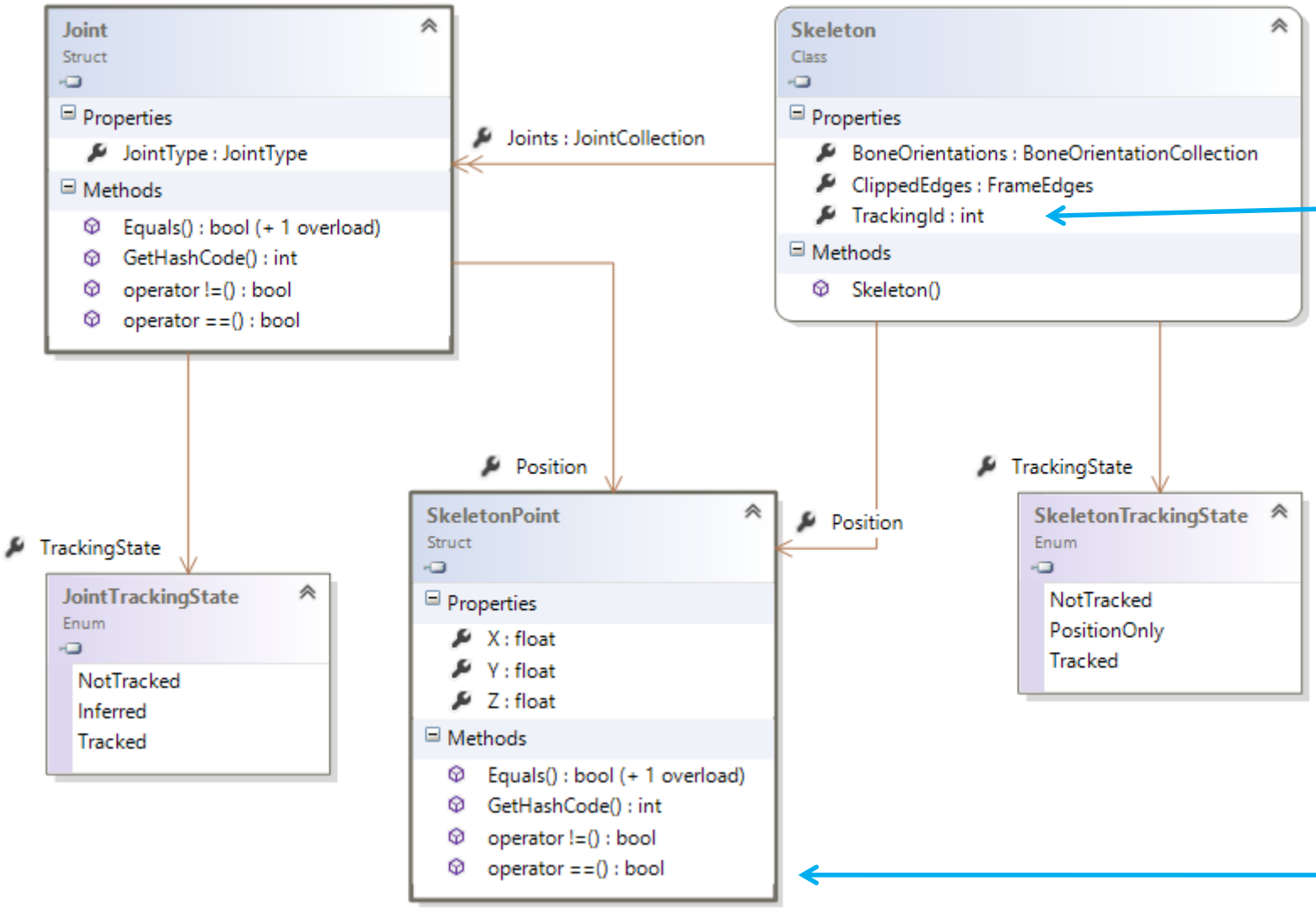
```
// enable returning skeletons while depth is in Near Range
this.kinect.SkeletonStream.EnableTrackingInNearRange = true;


private void EnableNearModeSkeletalTracking() {
 if (this.kinect != null && this.kinect.DepthStream != null && this.kinect.SkeletonStream !=
null) {
 this.kinect.DepthStream.Range = DepthRange.Near; // Depth in near range enabled
this.kinect.SkeletonStream.EnableTrackingInNearRange = true; // enable returning skeletons
while depth is in Near Range
this.kinect.SkeletonStream.TrackingMode = SkeletonTrackingMode.Seated; // Use seated tracking
} }
```

The SkeletonStream object model



AllFramesReady and SkeletonFrameReady Events return a SkeletonFrame which contain skeleton data

The Skeleton object model



Each skeleton has a unique identifier - TrackingID

Each joint has a Position, which is of type SkeletonPoint that reports the X, Y, and Z of the joint.

SkeletonTrakingState

SkeletonTrakingState	What it means
NotTracked	Skeleton object does not represent a tracked user. The Position field of the Skeleton and every Joint in the joints collection is a zero point
PositionOnly	The skeleton is detected, but is not actively being tracked. The Position field has a non-zero point, but the position of each Joint in the joints collection is a zero point.
Tracked	The skeleton is actively being tracked. The Position field and all Joint objects in the joints collection have non-zero points.

JointsTrakingState

JointsTrakingState	What it means
Inferred	Occluded, clipped, or low confidence joints. The skeleton engine cannot see the joint in the depth frame pixels, but has made a calculated determination of the position of the joint.
NotTracked	The position of the joint is indeterminable. The Position value is a zero point.
Tracked	The joint is detected and actively followed.

Use TransformSmoothParameters to smooth joint data to reduce jitter

code

Skeleton V1

```
private KinectSensor _KinectDevice;
private readonly Brush[] _SkeletonBrushes = { Brushes.Black, Brushes.Crimson, Brushes.Indigo,
                                              Brushes.DodgerBlue, Brushes.Purple, Brushes.Pink };
private Skeleton[] _FrameSkeletons;
#endregion Member Variables


private void InitializeKinect()    {

            this._KinectDevice.SkeletonStream.Enable();
            this._FrameSkeletons = new
Skeleton[this._KinectDevice.SkeletonStream.FrameSkeletonArrayLength];
            this.KinectDevice.SkeletonFrameReady += KinectDevice_SkeletonFrameReady;

            this._KinectDevice.Start();
        }

private void UninitializeKinect()
        {
            this._KinectDevice.Stop();
            this._KinectDevice.SkeletonFrameReady -= KinectDevice_SkeletonFrameReady;
            this._KinectDevice.SkeletonStream.Disable();

            this._FrameSkeletons = null;
        }
```

```
private void KinectDevice_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)        {
          using (SkeletonFrame frame = e.OpenSkeletonFrame())  {
              if (frame != null)                  {
              Skeleton skeleton; Brush userBrush;
                 LayoutRoot.Children.Clear();
                  frame.CopySkeletonDataTo(this._FrameSkeletons);

                  for (int i = 0; i < this._FrameSkeletons.Length; i++) {
                      skeleton = this._FrameSkeletons[i];

                      if (skeleton.TrackingState != SkeletonTrackingState.NotTracked)  {
                          Point p = GetJointPoint(skeleton.Position);
                          Ellipse ell = new Ellipse();
                          ell.Height = ell.Width = 30;
                          userBrush = this._SkeletonBrushes[i % this._SkeletonBrushes.Length];
                          ell.Fill = userBrush;

                          LayoutRoot.Children.Add(ell);
                          Canvas.SetTop(ell, p.Y - ell.Height / 2);
                          Canvas.SetLeft(ell, p.X - ell.Width / 2);
}     }  }   }  }
```

Copy SkeletonsData in local variable

Actually, Length is 6

Scale Position

```
private Point GetJointPoint(SkeletonPoint skPoint)
        {
            // Change System 3D ->2 D
            DepthImagePoint point = this.KinectDevice.MapSkeletonPointToDepth(skPoint,
                                                    this.KinectDevice.DepthStream.Format);

            // Scale point to actual dimension of container
            point.X = point.X * (int)this.LayoutRoot.ActualWidth /
                      this.KinectDevice.DepthStream.FrameWidth;
            point.Y = point.Y * (int)this.LayoutRoot.ActualHeight /
                      this.KinectDevice.DepthStream.FrameHeight;

            return new Point(point.X, point.Y);
        }
```

Mapping different Coordinate systems

Smoothing

TransformSmoothParameters	What it means
Correction	A float ranging from 0 to 1.0. The lower the number, the more correction is applied.
JitterRadius	Sets the radius of correction. If a joint position “jitters” outside of the set radius, it is corrected to be at the radius. Float value measured in meters.
MaxDeviationRadius	Used this setting in conjunction with the JitterRadius setting to determine the outer bounds of the jitter radius. Any point that falls outside of this radius is not considered a jitter, but a valid new position. Float value measured in meters.
Prediction	Sets the number of frames predicted.
Smoothing	Determines the amount of smoothing applied while processing skeletal frames. It is a float type with a range of 0 to 1.0. The higher the value, the more smoothing applied. A zero value does not alter the skeleton data.

code

Skeleton V2

```
private void InitializeKinect()       {

            var parameters = new TransformSmoothParameters
            {
                Smoothing = 0.3f,
                Correction = 0.0f,
                Prediction = 0.0f,
               JitterRadius = 1.0f,
               MaxDeviationRadius = 0.5f
            };
            _KinectDevice.SkeletonStream.Enable(parameters);

            this._KinectDevice.SkeletonStream.Enable();
            this._FrameSkeletons = new
        Skeleton[this._KinectDevice.SkeletonStream.FrameSkeletonArrayLength];
            this.KinectDevice.SkeletonFrameReady += KinectDevice_SkeletonFrameReady;

            this._KinectDevice.Start();
        }
```

```
private void KinectDevice_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)        {
          using (SkeletonFrame frame = e.OpenSkeletonFrame())  {
              if (frame != null)                {
              Skeleton skeleton;
              Brush userBrush;
                 LayoutRoot.Children.Clear();
                  frame.CopySkeletonDataTo(this._FrameSkeletons);

                  for (int i = 0; i < this._FrameSkeletons.Length; i++) {
                      skeleton = this._FrameSkeletons[i];
                      if (skeleton.TrackingState != SkeletonTrackingState.NotTracked)  {
                          Point p = GetJointPoint(skeleton.Position);
                          Ellipse ell = new Ellipse();
                          ell.Height = ell.Width = 30;
                    userBrush = this._SkeletonBrushes[i % this._SkeletonBrushes.Length];
                          ell.Fill = userBrush;

                          LayoutRoot.Children.Add(ell);
                          Canvas.SetTop(ell, p.Y - ell.Height / 2);
                          Canvas.SetLeft(ell, p.X - ell.Width / 2);
                    if (skeleton.TrackingState == SkeletonTrackingState.Tracked)         {
                          DrawSkeleton(skeleton, userBrush);
} }     } }   } }
```

```
private void DrawSkeleton(Skeleton skeleton, Brush userBrush)        {
//Draws the skeleton’s head and torso
  joints = new[] { JointType.Head, JointType.ShoulderCenter, JointType.ShoulderLeft, JointType.Spine,
          JointType.ShoulderRight, JointType.ShoulderCenter, JointType.HipCenter, JointType.HipLeft,
          JointType.Spine, JointType.HipRight, JointType.HipCenter };
  LayoutRoot.Children.Add(CreateFigure(skeleton, userBrush, joints));

  //Draws the skeleton’s left leg
  joints = new[] { JointType.HipLeft, JointType.KneeLeft, JointType.AnkleLeft, JointType.FootLeft };
  LayoutRoot.Children.Add(CreateFigure(skeleton, userBrush, joints));

  //Draws the skeleton’s right leg
  joints = new[] { JointType.HipRight, JointType.KneeRight, JointType.AnkleRight, JointType.FootRight };
  LayoutRoot.Children.Add(CreateFigure(skeleton, userBrush, joints));

  //Draws the skeleton’s left arm
  joints = new[] { JointType.ShoulderLeft, JointType.ElbowLeft, JointType.WristLeft, JointType.HandLeft };
  LayoutRoot.Children.Add(CreateFigure(skeleton, userBrush, joints));

  //Draws the skeleton’s right arm
  joints = new[] { JointType.ShoulderRight, JointType.ElbowRight, JointType.WristRight, JointType.HandRight };
  LayoutRoot.Children.Add(CreateFigure(skeleton, userBrush, joints));

  }
```

```
private Polyline CreateFigure(Skeleton skeleton, Brush brush, JointType[] joints)
        {
            Polyline figure = new Polyline();
            figure.StrokeThickness = 4;
            figure.Stroke = brush;

            for (int i = 0; i < joints.Length; i++)
            {
                figure.Points.Add(GetJointPoint(skeleton.Joints[joints[i]].Position));
            }

            return figure;
        }
```

code

Skeleton V3

```
 private void InitializeKinect()        {
     var parameters = new TransformSmoothParameters{
                Smoothing = 0.3f,
                Correction = 0.0f,
                Prediction = 0.0f,
                JitterRadius = 1.0f,
                MaxDeviationRadius = 0.5f };
     _KinectDevice.SkeletonStream.Enable(parameters);


      this._KinectDevice.SkeletonStream.Enable();
      this._FrameSkeletons = new Skeleton[this._KinectDevice.SkeletonStream.FrameSkeletonArrayLength];
      this.KinectDevice.SkeletonFrameReady += KinectDevice_SkeletonFrameReady;

      this._KinectDevice.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
      this._KinectDevice.ColorFrameReady += new
EventHandler<ColorImageFrameReadyEventArgs>(_KinectDevice_ColorFrameReady);

      this._ColorImageBitmap = new WriteableBitmap(_KinectDevice.ColorStream.FrameWidth,
                                                   _KinectDevice.ColorStream.FrameHeight, 96, 96,
                                                       PixelFormats.Bgr32, null);
      this._ColorImageBitmapRect = new Int32Rect(0, 0, _KinectDevice.ColorStream.FrameWidth,
                                      _KinectDevice.ColorStream.FrameHeight);
      this._ColorImageStride = _KinectDevice.ColorStream.FrameWidth *
_KinectDevice.ColorStream.FrameBytesPerPixel;
      ColorImage.Source = this._ColorImageBitmap;


      this._KinectDevice.Start();      }
```

Video Stream initialization

```
private Point GetJointPoint(SkeletonPoint skPoint)
        {
            // Change System 3D ->2D
       this.KinectDevice.DepthStream.Format);

         ColorImagePoint point = this.KinectDevice.MapSkeletonPointToColor(skPoint,
                        this.KinectDevice.ColorStream.Format);

       // Scale point to actual dimension of container
            point.X = point.X * (int)this.LayoutRoot.ActualWidth /
                        this.KinectDevice.DepthStream.FrameWidth;
            point.Y = point.Y * (int)this.LayoutRoot.ActualHeight /
                        this.KinectDevice.DepthStream.FrameHeight;

            return new Point(point.X, point.Y);
        }
```

Mapping on Color Coordinate system

Choosing Skeletons

Using the **AppChoosesSkeletons** property and **ChooseSkeletons** method is possible select which skeletons to track:

AppChoosesSkeletons	What it means
False(default)	The skeleton engine chooses the **first two skeletons** available for tracking (selection process is **unpredictable**)
True	To manually select which skeletons to track call the **ChooseSkeletons** method passing in the **TrackingIDs** of the skeletons you want to track.

The **ChooseSkeletons** method accepts one, two, or no **TrackingIDs**. The skeleton engine stops tracking all skeletons when the ChooseSkeletons method is passed no parameters.

Choosing Skeletons(2)

Call ChooseSkeletons when AppChoosesSkeletons==false throws an InvalidOperationException.

If AppChoosesSkeletons is set to true before the SkeletonStream is enabled, no skeletons are actively tracked until call ChooseSkeletons.

Skeletons automatically selected for tracking before setting AppChoosesSkeletons is set to true continue to be actively tracked until the skeleton leaves the scene or is manually replaced. If the automatically selected skeleton leaves the scene, it is not automatically replaced.

Any skeletons manually chosen for tracking continue to be tracked after AppChoosesSkeletons is set to false until the skeleton leaves the scene.

Gesture Interaction

How to design a gesture?

Matteo Valoriani
mvaloriani AT gmail.com
@MatteoValoriani

Gesture

What is a gesture?

An action intended to communicate feelings or intentions

What is “Gesture Detection” or “Gesture Recognition”?

Computer’s ability to understand human gestures as input

First used in 1963 with pen-based input device

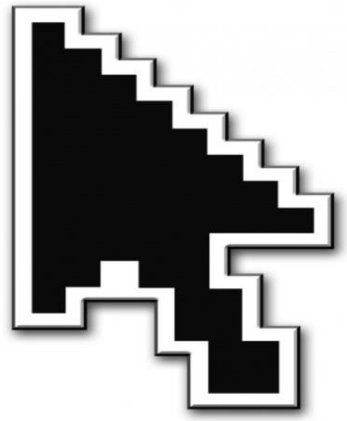
What is it used for?

Mouse movements, Handwriting recognition, Sign language, recognition, Touch screen input, Kinect

Interaction metaphors

Depends on the task

Important aspect in design of UI



Cursors (hands tracking):
Target an object

Avatars (body tracking):
Interaction with virtual space

The shadow/mirror effect

Shadow Effect

I see the back of my avatar
Problems with Z movements

Mirror Effect

I see the front of my avatar
Problem with mapping left/right movements

How to design a gesture?

User Interaction

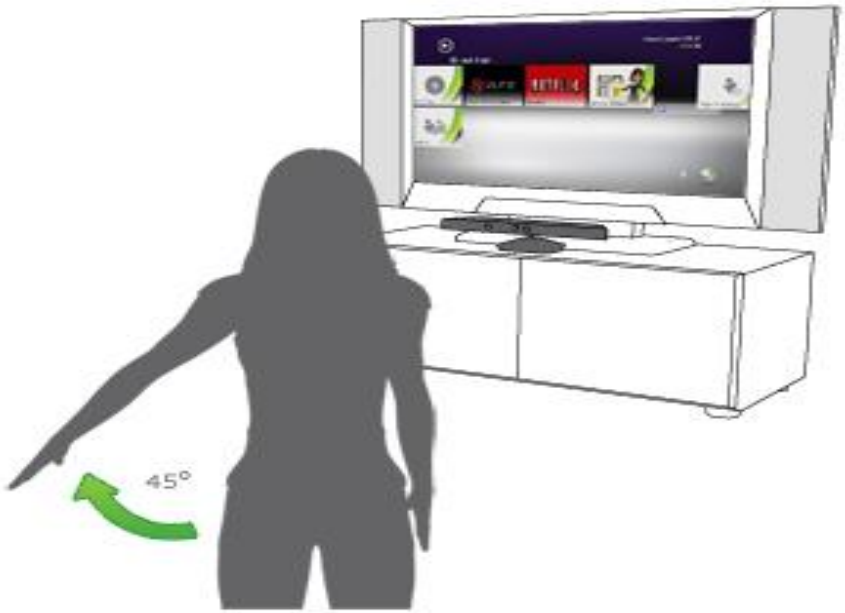
Game mindset ≠ UI mindset

Challenging = fun

Challenging = easy and effective

Gesture semantically fits user task

Abstract



Meaningful

User action fits UI reaction

System's UI feedback relates to the user's physical movement

User action fits UI reaction

System's UI feedback relates to the user's physical movement

Gestures family-up

Each gesture feels related and cohesive with entire gesture set

Handed gestures

Different gesture depending on hand: only left hand can do gesture A

Repeting Gesture?

Will users want/need to perform the proposed gesture repeatedly?

Repeting Gesture?

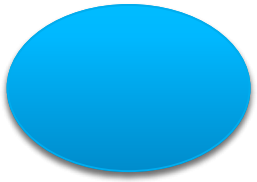
Will users want/need to perform the proposed gesture repeatedly?

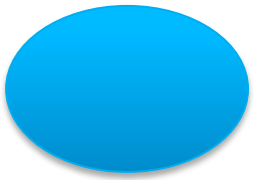
Number of Hands

One-handed gestures are preferred

Symmetrical two-handed gesture

Two hand gesture should be symmetrical



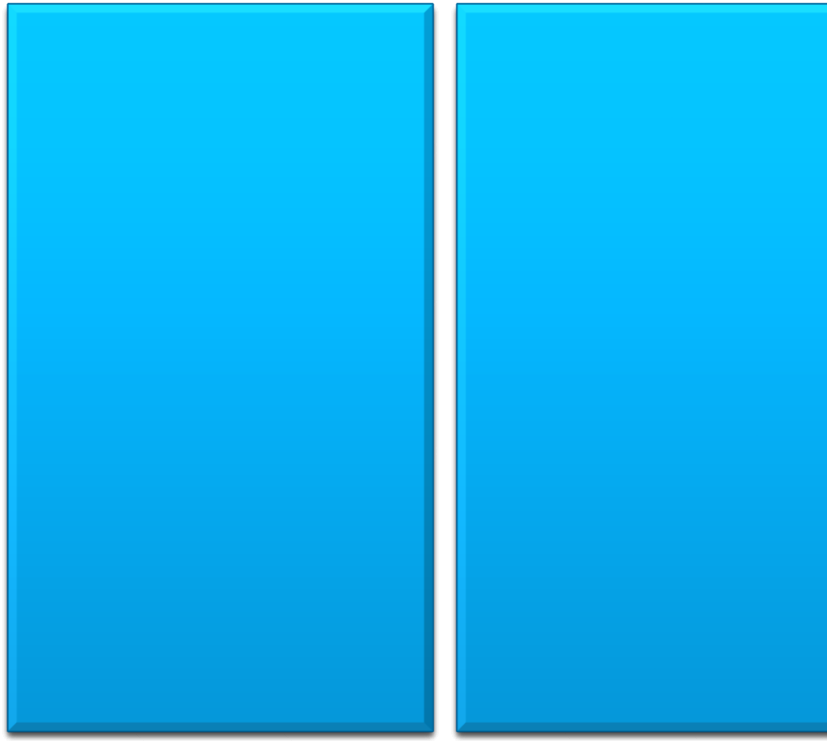


Gesture payoff

Interactions requiring more work and effort should have a higher payoff

Fatigue kills gesture

Fatigue is the start of downward that kills gesture



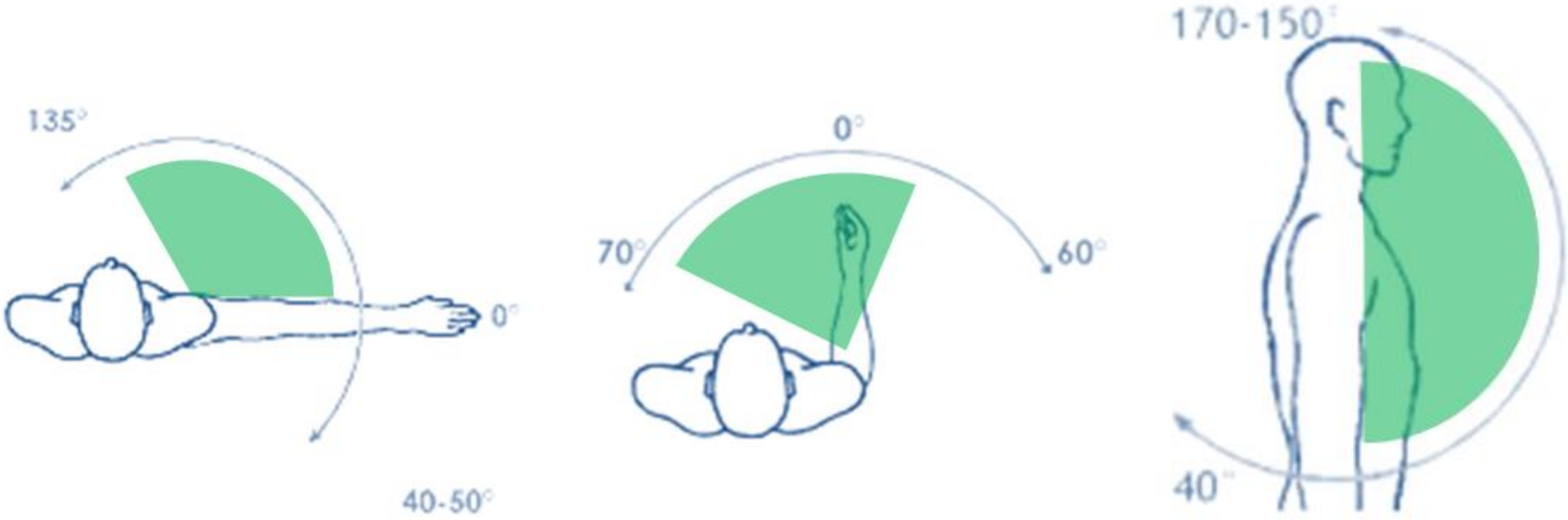
Fatigue increase messiness → poor performance → frustration → bad UX

Gorilla Arm problem

Try to raise your arm for 10 minutes…

☺

Comfortable positions



User Posture

User posture may affect design of a gesture

The challenges

Environment

Input variability

Gesture Recognition

Artificial Intelligence for Kinect

Matteo Valoriani
mvaloriani AT gmail.com
@MatteoValoriani

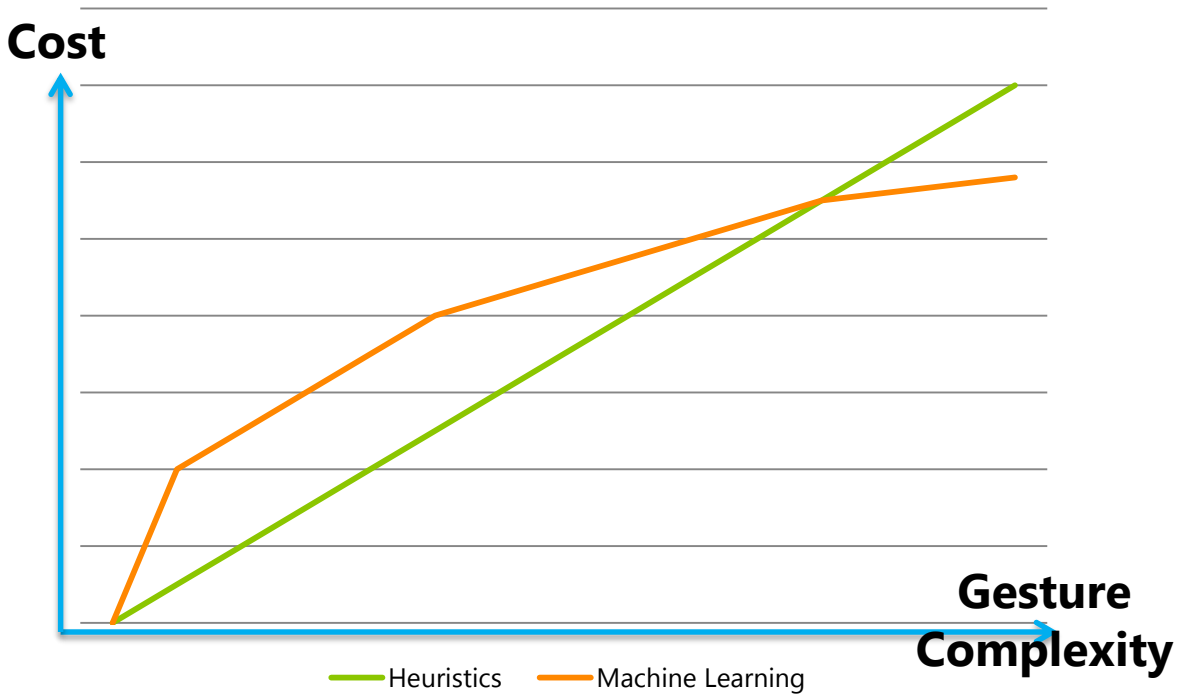
Heuristics

Experience-based techniques for problem solving, learning, and discovery

Cost effective

Helps reconstruct missing information

Helps compute outcome of a gesture



Define What Constitutes a Gesture

Some players have more energy (or enthusiasm) than others

Some players will “optimize” their gestures

Most players will not perform the gesture precisely as intended

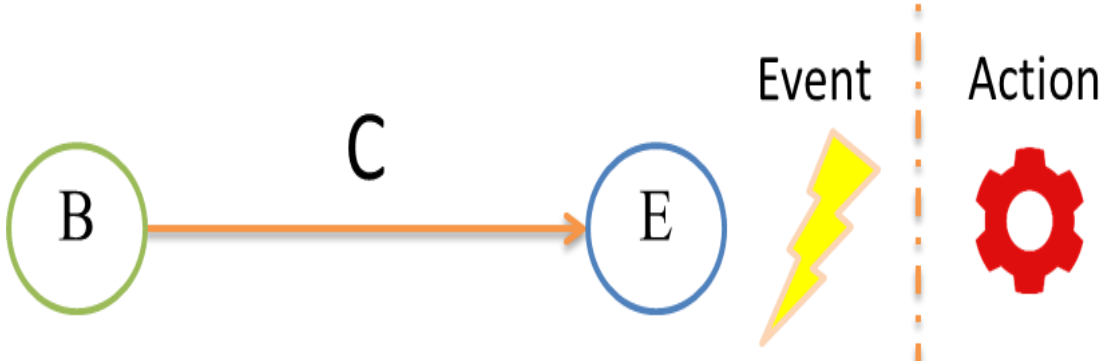
Define Key Stages of a Gesture

Determine

When the gesture begins
When the gesture ends

Determine other key stages

Changes in motion direction
Pauses
…



Definite gesture

Contact or release point
Direction
Initial velocity

Continuous gesture

Frequency
Amplitude

Detection Filter Only When Necessary!

Define clear context for when a gesture is expected

Provide clear feedback to the player

Run the gesture filter when the context warrants it

Cancel the gesture if context changes

Causes of Missing Information

Self Occlusion

Side poses

Player's position in play space

Obstacles

Other players

Furniture

Outside the camera's field of view

Left or right (easy to fix)

Top or bottom (hard to avoid)

Algorithmic Detection

Wave Gesture Detection Sample

Gesture Definition

Define gesture algorithmically:

Hand above elbow

Hand in front of shoulder

Hand moving long enough in same direction

Hand moving at certain velocity

Count number of direction changes

```
if(hand.y - elbow.y > threshold) &&
  (hand.z - shoulder.z < threshold) &&
  (hand.y – precHand.y < threshold) &&
  ((hand.x – precHand.x)/dt < threshold)
```

Implementation Overview

Set threshold

Calculate hand position

Calculate hand movement

Update wave state

code

Static Postures: HandOnHead

```
class GestureRecognizer    {
        public Dictionary<JointType, List<Joint>> skeletonSerie = new Dictionary<JointType, List<Joint>>() {
        { JointType.AnkleLeft, new List<Joint>()},          { JointType.AnkleRight, new List<Joint>()},
        { JointType.ElbowLeft, new List<Joint>()},          { JointType.ElbowRight, new List<Joint>()},
        { JointType.FootLeft, new List<Joint>()},          { JointType.FootRight, new List<Joint>()},
        { JointType.HandLeft, new List<Joint>()},           { JointType.HandRight, new List<Joint>()},
        { JointType.Head, new List<Joint>()},          { JointType.HipCenter, new List<Joint>()},
        { JointType.HipLeft, new List<Joint>()},          { JointType.HipRight, new List<Joint>()},
        { JointType.KneeLeft, new List<Joint>()},          { JointType.KneeRight, new List<Joint>()},
        { JointType.ShoulderCenter, new List<Joint>()},          { JointType.ShoulderLeft, new List<Joint>()},
        { JointType.ShoulderRight, new List<Joint>()},
        { JointType.Spine, new List<Joint>()},
        { JointType.WristLeft, new List<Joint>()},
        { JointType.WristRight, new List<Joint>()}
      };

      protected List<DateTime> timeList;

      private static List<JointType> typesList = new List<JointType>() {
JointType.ElbowLeft, JointType.ElbowRight, JointType.FootLeft, JointType.FootRight, JointType.HandLeft,
JointType.HandRight, JointType.Head, JointType.HipCenter, JointType.HipLeft, JointType.HipRight, JointType.KneeLeft,
JointType.KneeRight, JointType.ShoulderCenter, JointType.ShoulderLeft, JointType.ShoulderRight, JointType.Spine,
JointType.WristLeft, JointType.WristRight };
 //... continue
}
```

Key	Value
AnkleLeft	$<V_{t1}, V_{t2}, V_{t3}, V_{t4}..>$
AnkleRight	$<V_{t1}, V_{t2}, V_{t3}, V_{t4}..>$
ElbowLeft	$<V_{t1}, V_{t2}, V_{t3}, V_{t4}..>$

```
const int bufferLenght=10;

public void Recognize(JointCollection jointCollection, DateTime date)        {
            timeList.Add(date);
            foreach (JointType type in typesList)            {
                skeletonSerie[type].Add(jointCollection[type]);
                if (skeletonSerie[type].Count > bufferLenght)                {
                    skeletonSerie[type].RemoveAt(0);
                }
            }
            startRecognition();
        }

List<Gesture> gesturesList = new List<Gesture>();
private void startRecognition()        {
            gesturesList.Clear();
            gesturesList.Add(HandOnHeadReconizerRT(JointType.HandLeft,
                                        JointType.ShoulderLeft));
            // Do ...
        }
```

```
Boolean isHOHRecognitionStarted;
DateTime StartTimeHOH = DateTime.Now;
private Gesture HandOnHeadReconizerRT (JointType hand, JointType shoulder)       {
        // Correct Position
            if (skeletonSerie[hand].Last().Position.Y > skeletonSerie[shoulder].Last().Position.Y + 0.2f) {
            if (!isHOHRecognitionStarted)     {
                isHOHRecognitionStarted = true;
                StartTimeHOH = timeList.Last();
            }
            else   {
                double totalMilliseconds = (timeList.Last() - StartTimeHOH).TotalMilliseconds;
                // time ok?
                if ((totalMilliseconds >= HandOnHeadMinimalDuration))     {
                  isHOHRecognitionStarted = false;
                  return Gesture.HandOnHead;
                }
            }
        }
        else {//Incorrect Position
            if (isHOHRecognitionStarted)              {
                isHOHRecognitionStarted = false;
            }
        }
        return Gesture.None;        }
```

How to notify a gesture?

Synchronous Solution: Return gesturesList to GUI

Asynchronous Solution: Use Event

```
public delegate void HandOnHeadHadler(object sender, EventArgs e);

public event HandOnHeadHadler HandOnHead;

private Gesture HandOnHeadReconizerRTWithEvent(JointType hand, JointType shoulder) {
            Gesture g = HandOnHeadReconizerRT(hand, shoulder);
            if (g == Gesture.HandOnHead)            {
                if (HandOnHead != null) HandOnHead(this, EventArgs.Empty);
            }
            return g;}
```

code

Swipe

```
const float SwipeMinimalLength = 0.08f;
const float SwipeMaximalHeight = 0.02f;
const int SwipeMinimalDuration = 200;
const int SwipeMaximalDuration = 1000;
const int MinimalPeriodBetweenGestures = 0;

private Gesture HorizzontalSwipeRecognizer(List<Joint> positionList) {
    int start = 0;
    for (int index = 0; index < positionList.Count - 1; index++)    {

        if ((Math.Abs(positionList[0].Position.Y - positionList[index].Position.Y) > SwipeMaximalHeight) ||
                Math.Abs((positionList[index].Position.X - positionList[index + 1].Position.X)) < 0.01f)   {
            start = index;              }


        if ((Math.Abs(positionList[index].Position.X - positionList[start].Position.X) > SwipeMinimalLength))   {
            double totalMilliseconds = (timeList[index] - timeList[start]).TotalMilliseconds;

                    if (totalMilliseconds >= SwipeMinimalDuration && totalMilliseconds <= SwipeMaximalDurati      {

                    if (DateTime.Now.Subtract(lastGestureDate).TotalMilliseconds > MinimalPeriodBetweenGestures)                    {
                        lastGestureDate = DateTime.Now;
                        if (positionList[index].Position.X - positionList[start].Position.X < 0)
                            return Gesture.SwipeRightToLeft;
                        else
                            return Gesture.SwipeLeftToRight;

                    }
                }
            }

    } return Gesture.None;        }
```

Δ_x too small or Δ_y too big → shift start

Δ_x > minimal lenght

Δ_t in the accepted range

```
public delegate void SwipeHadler(object sender, GestureEventArgs e);
public event SwipeHadler Swipe;

private Gesture HorizzontalSwipeRecognizer(JointType jointType) {
        Gesture g = HorizzontalSwipeRecognizer(skeletonSerie[jointType]);
        switch (g) {
            case Gesture.None:
                break;
            case Gesture.SwipeLeftToRight:
                if (Swipe != null) Swipe(this, new GestureEventArgs("SwipeLeftToRight"));
                break;
            case Gesture.SwipeRightToLeft:
                if (Swipe != null) Swipe(this, new GestureEventArgs("SwipeRightToLeft"));
                break;
            default:
                break;
        }

        return g;
    }


...
 public class GestureEventArgs : EventArgs
   {
       public string text;
       public GestureEventArgs(string text) {   this.text = text;       }

   }
```

Personalized EventArgs

demo

Heuristic Based Gesture Detection: FAAST

Pros & Cons

PROs

Easy to understand

Easy to implement (for simple gestures)

Easy to debug

CONs

Challenging to choose best values for parameters

Doesn't scale well for variants of same gesture

Gets challenging for complex gestures

Challenging to compensate for latency

Recommendation

Use for simple gestures (like Hand wave, Head movement, …)

Weighted Networks Based Gesture Detection

Jump Gesture Detection Sample

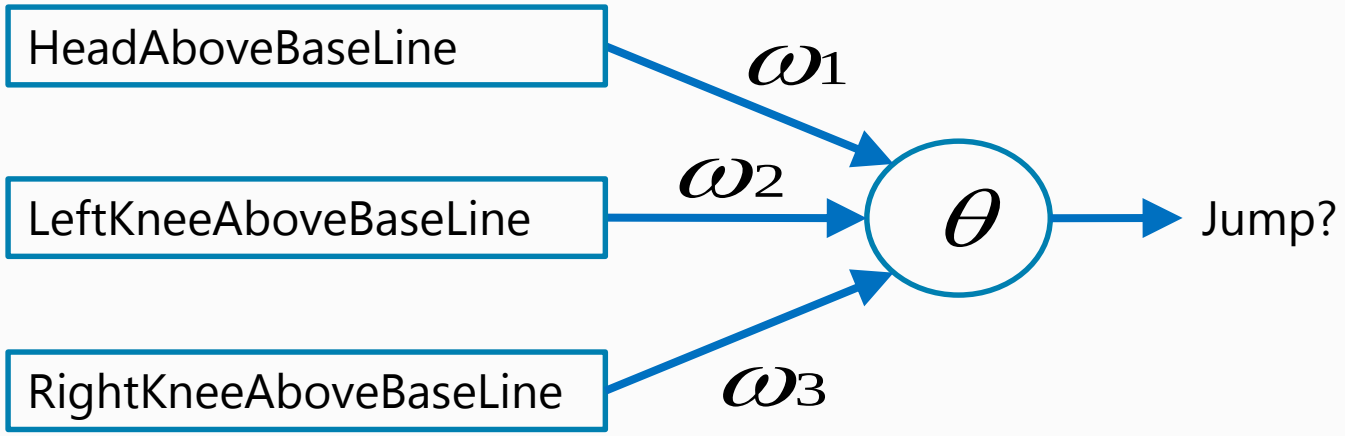
Gesture Definition

Define gesture as weighted network

Simple neural network

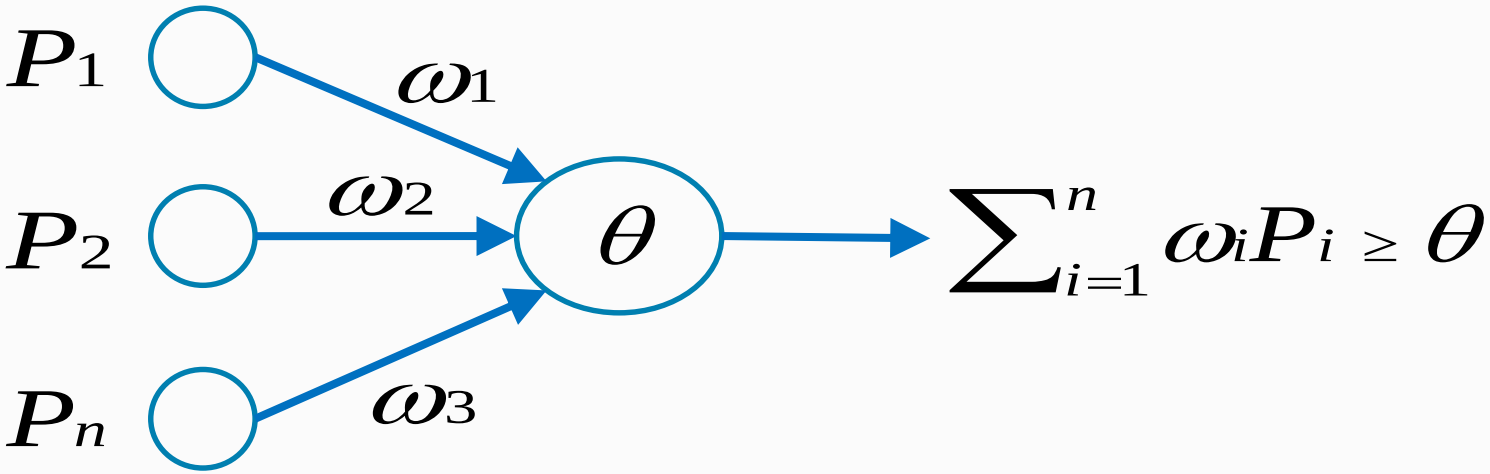
Simple algorithmic gestures as input nodes

Use fuzzy logic, i.e. probabilities, not Booleans



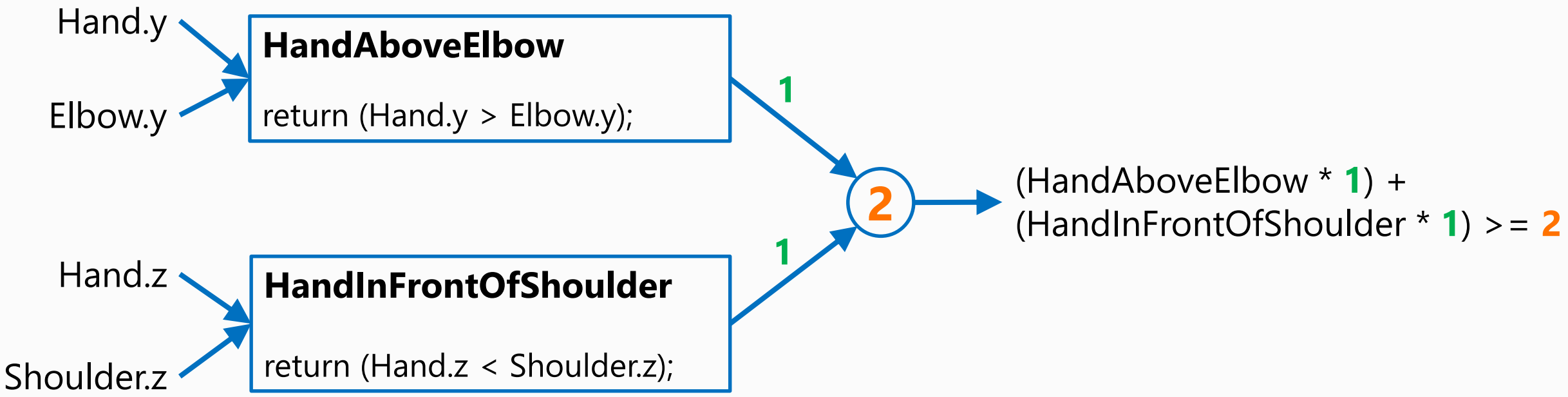
Perceptron

Simple network using weighted threshold elements



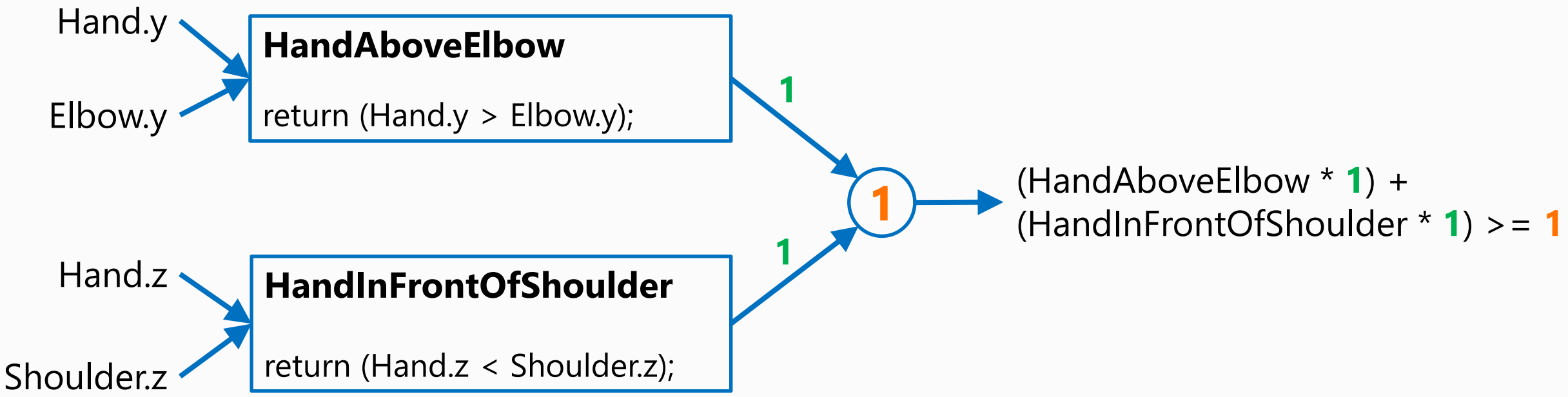
Example

HandAboveElbow **AND** HandInFrontOfShoulder



Example

HandAboveElbow **OR** HandInFrontOfShoulder

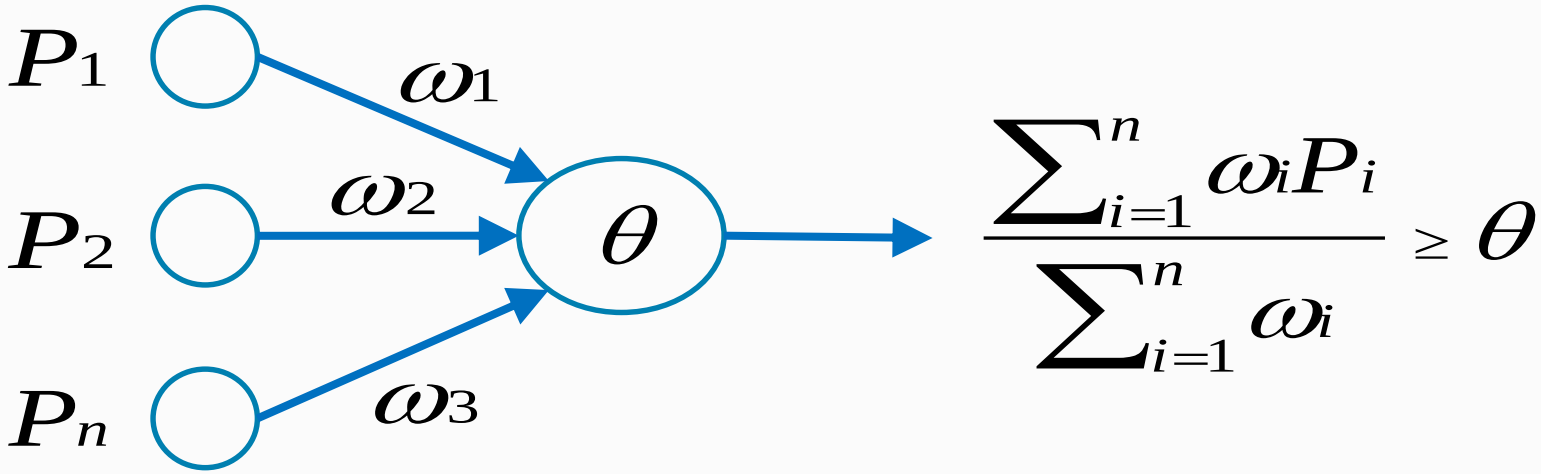


Network Definition for Detector

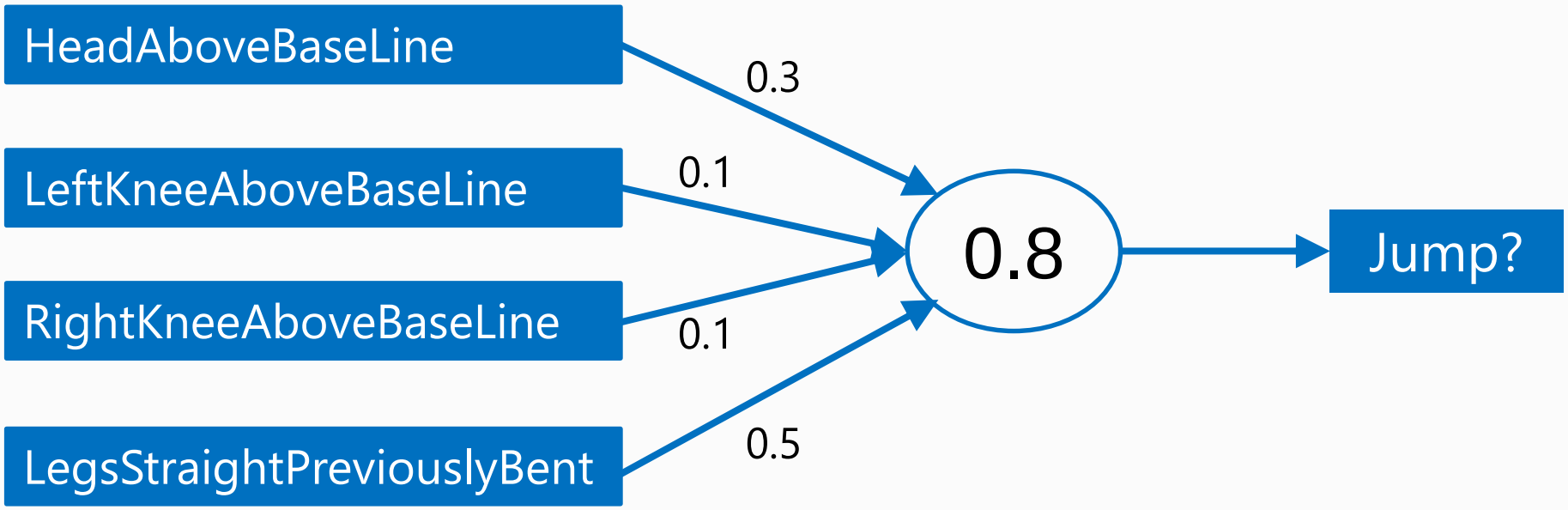
Similar to perceptron

Normalize using weights

Use probabilities, not Booleans



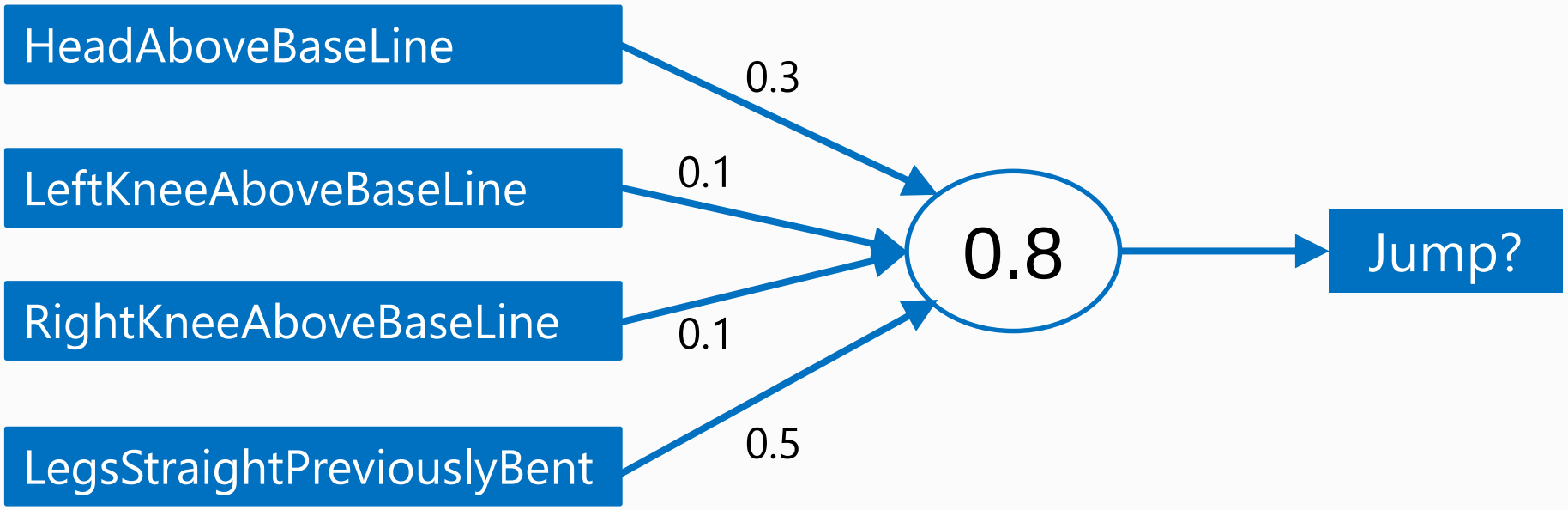
Jump detection



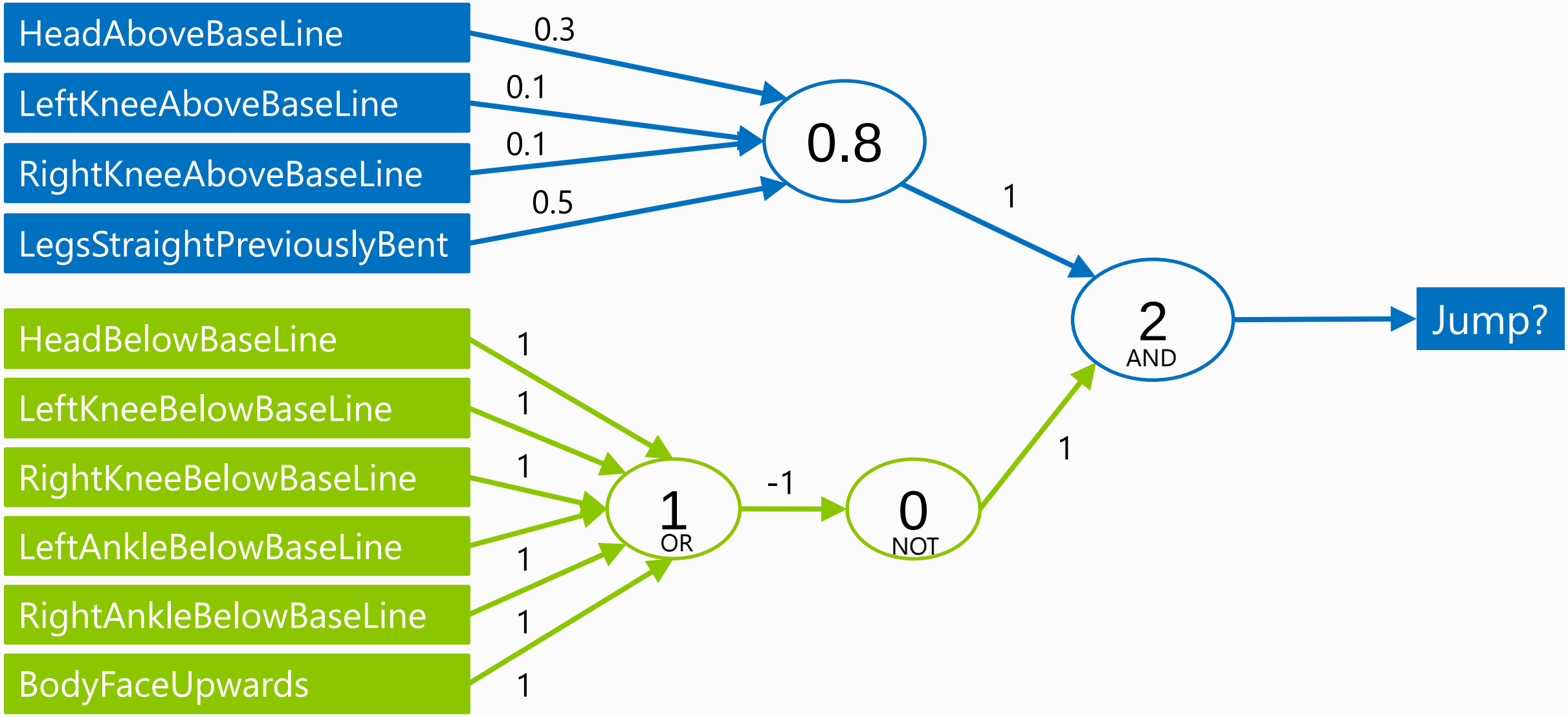
Are we sure this is enough?

Due to noise, still many false positives

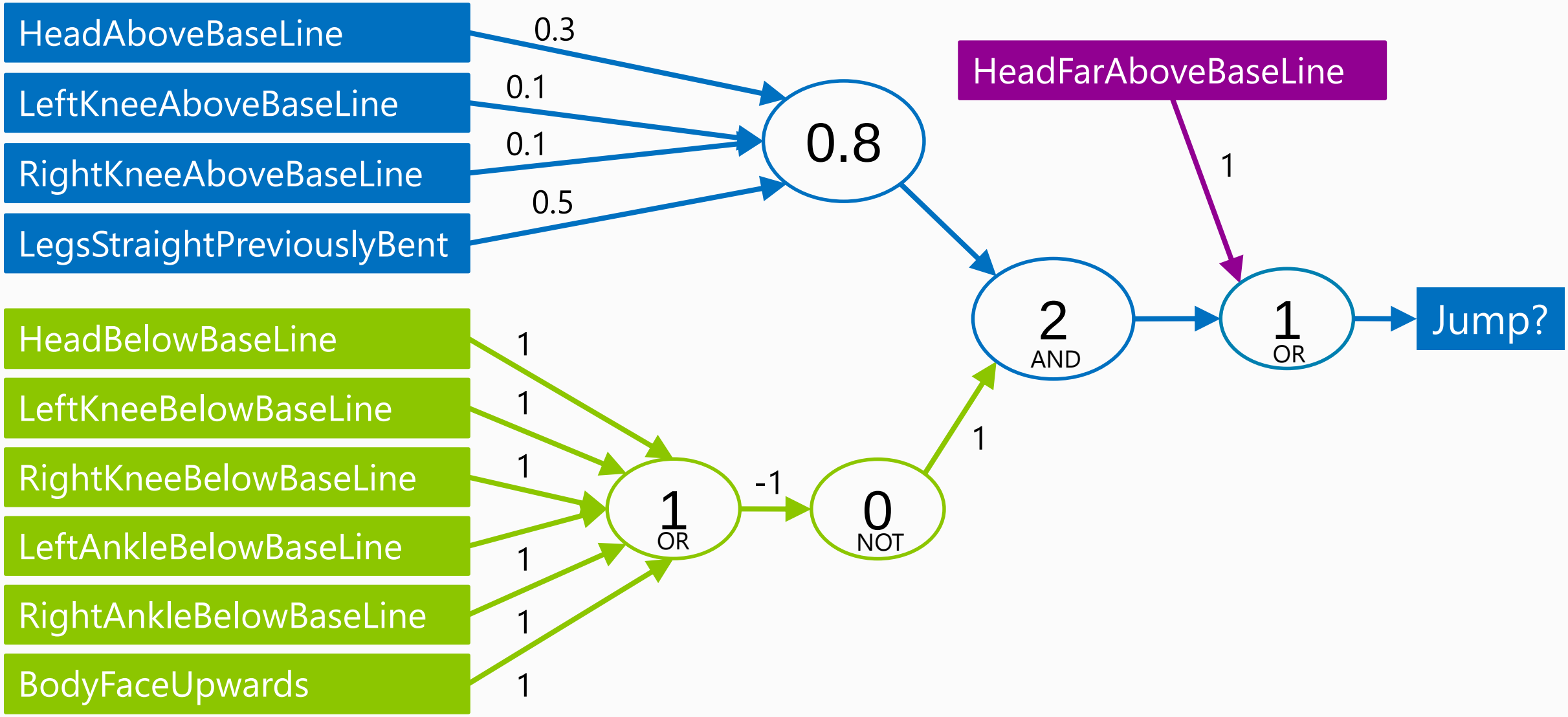
How can we reduce false positives?



And We're Done!



But Wait, If We Know For Sure...



Pros & Cons

PROs

Complex gestures can be detected

Good CPU performance

Scale well for variants of same gesture

Nodes can be reused in different gestures

CONs

Not easy to debug

Challenging to compensate for latency

Small changes in parameters can have dramatic changes in results

Very time consuming to choose manually parameters

Recommendation

Use for composed gestures (Jump, duck, punch,…)
Break complex gestures into collection of simple gestures

Exemplar Matching Based Gesture Detection

Gesture Definition

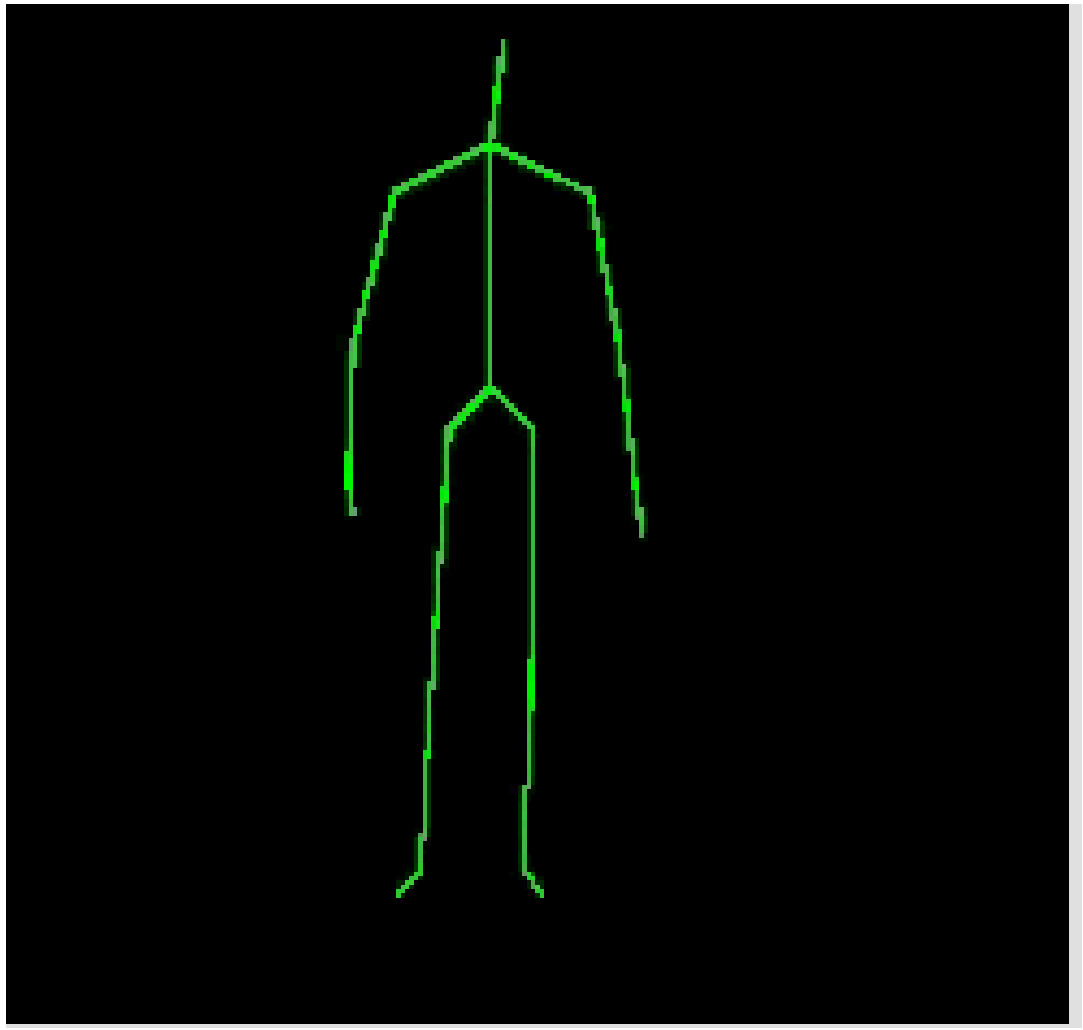
Define gesture as pre-recorded animations

Motion capture animations

Record different people doing same gesture

Each person doing same gesture multiple times

Pre-recorded animations are called Exemplars



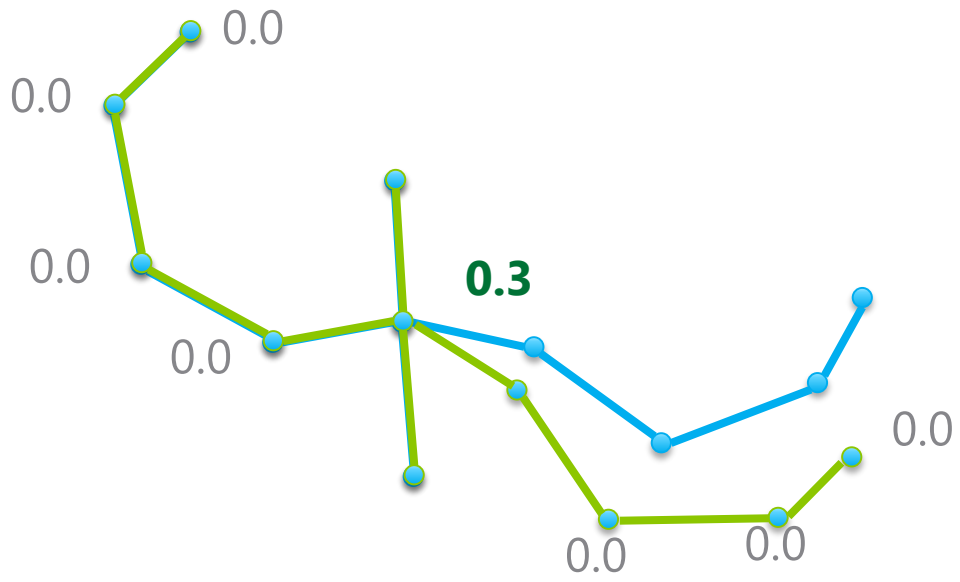
Exemplar Matching

Need to compare skeleton frames

Define error metric for skeleton

Angular difference for each joint in local space

Peak Signal to Noise Ratio for whole skeleton



$$MSE = \frac{1}{N}\sum \text{Distance}_i^2$$

$$PSNR = 10 * \log_{10}(MAX^2 / MSE)$$

Exemplar Matching

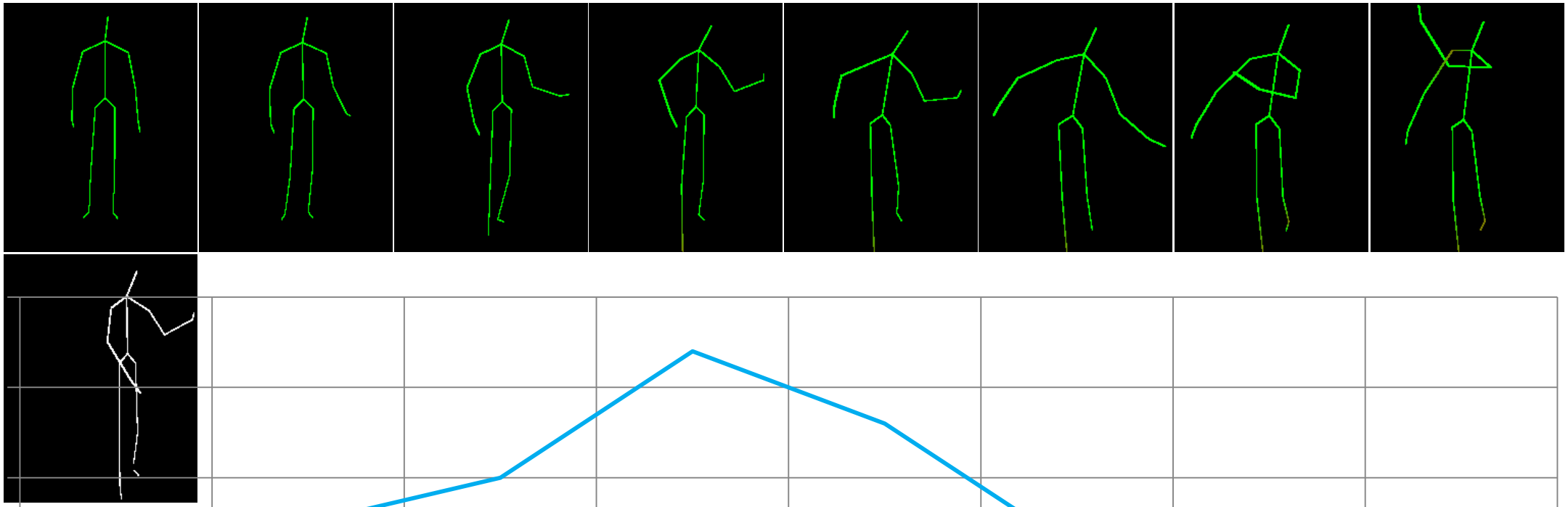
Search for best matching frames

Best matching frame has strongest signal

Different classifiers can be used

K-Nearest Neighbour
Dynamic Time Warping (DTW)
Hidden Markov Models (HMM)

Exemplar Matching



demo

DTW Based Gesture Detection: Swipe

Pros & Cons

PROs

Very complex gestures can be detected

DTW allows for different speeds

Can scale for variants of same gesture

Easy to visualize exemplar matching

CONs

Requires lots of resources to be robust

Multiple recordings of multiple people for one gesture

i.e. requires lots of CPU and memory

Recommendation

Use for complex context-sensitive dynamic gestures

- Dancing, fitness exercises,…

Comparison

10 Gestures, 10 People, 5 times = 500 Exemplars

K-Nearest

46 ms

DTW

156 ms

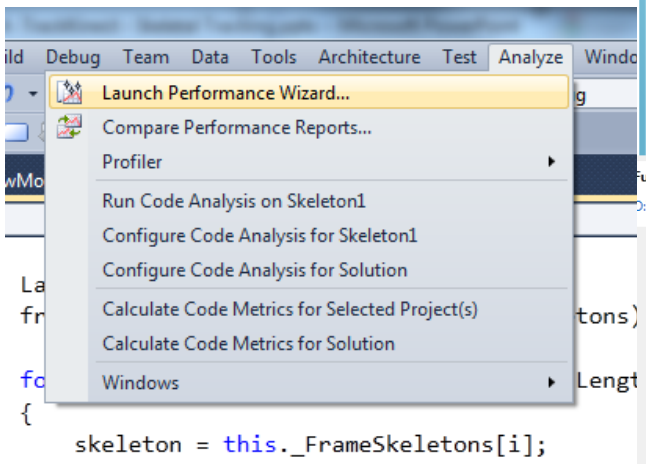
Weighted network

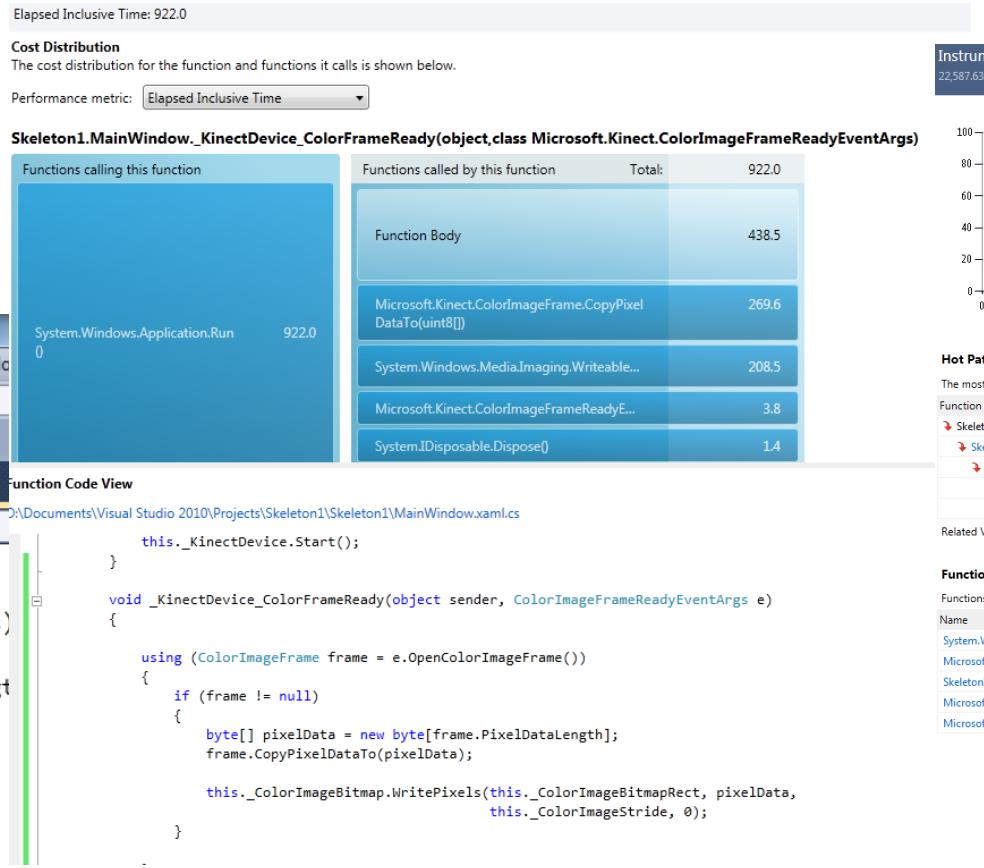
1 ms

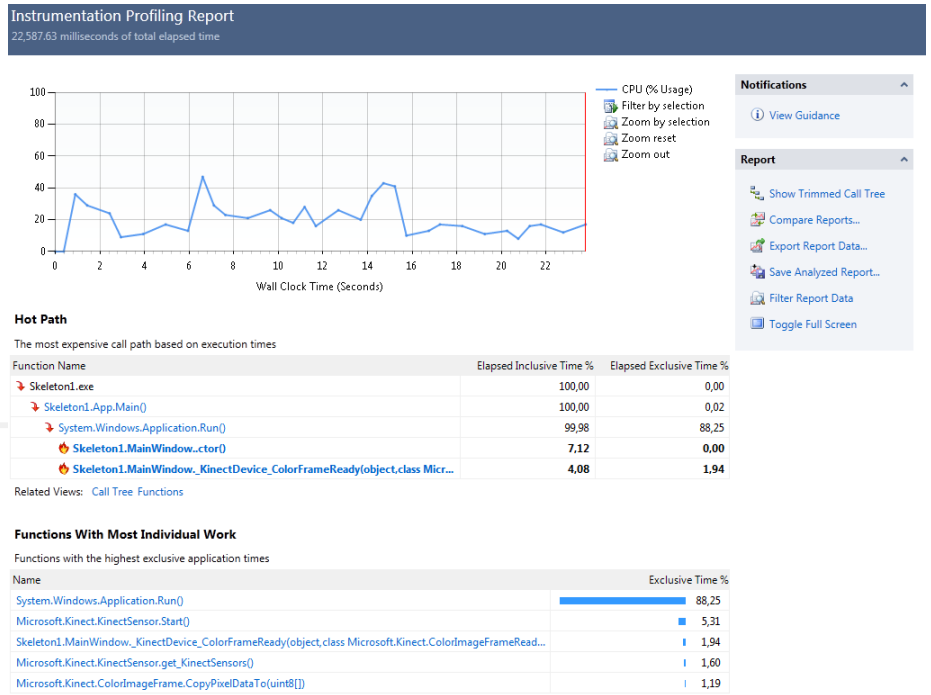
Performance

Skeleton processing is an expensive operation.

Use VS2012 Performance Tool







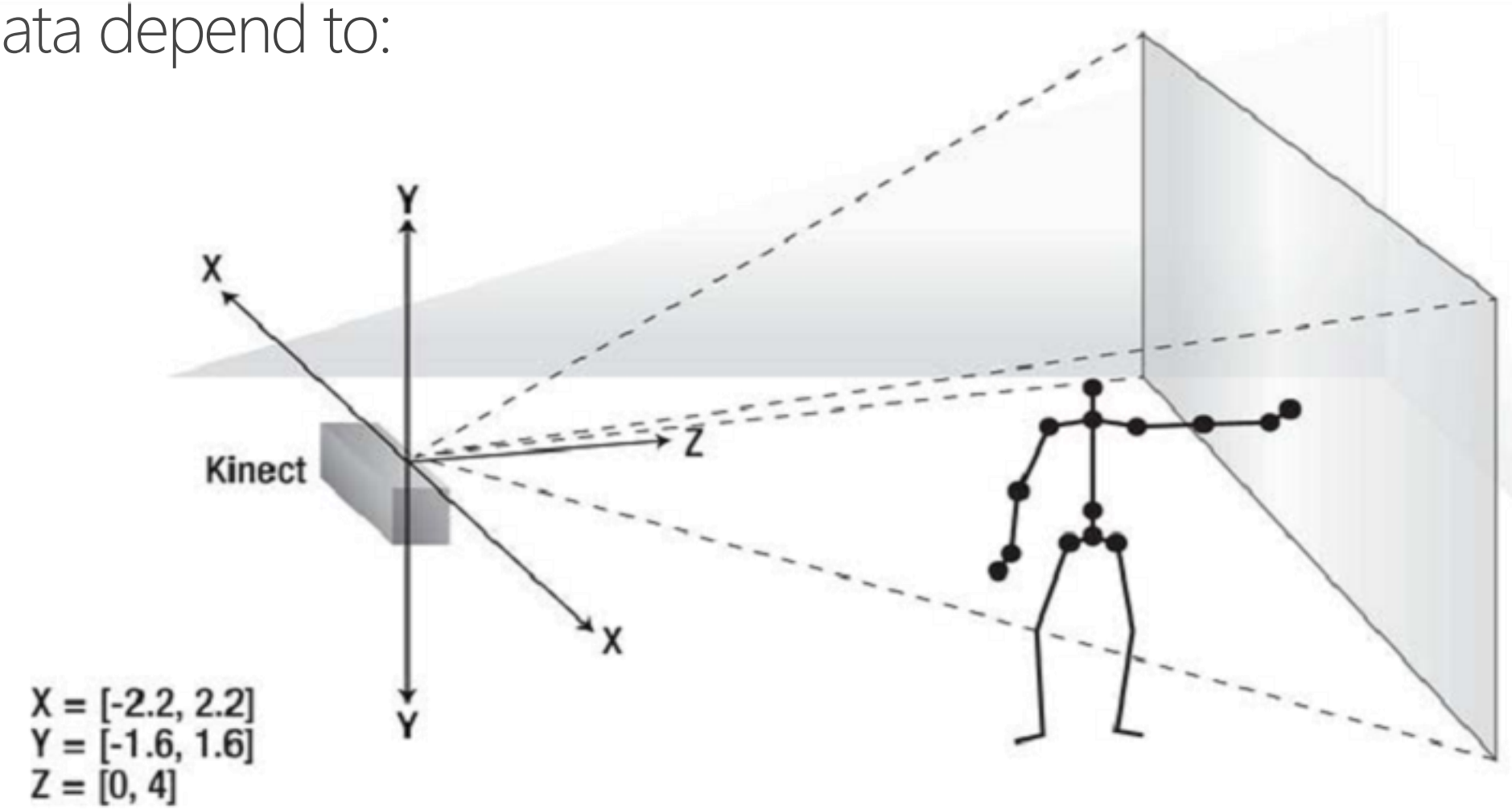
Posture Abstraction and Normailization

Posture Abstraction

Kinect SkeletonData depend to:

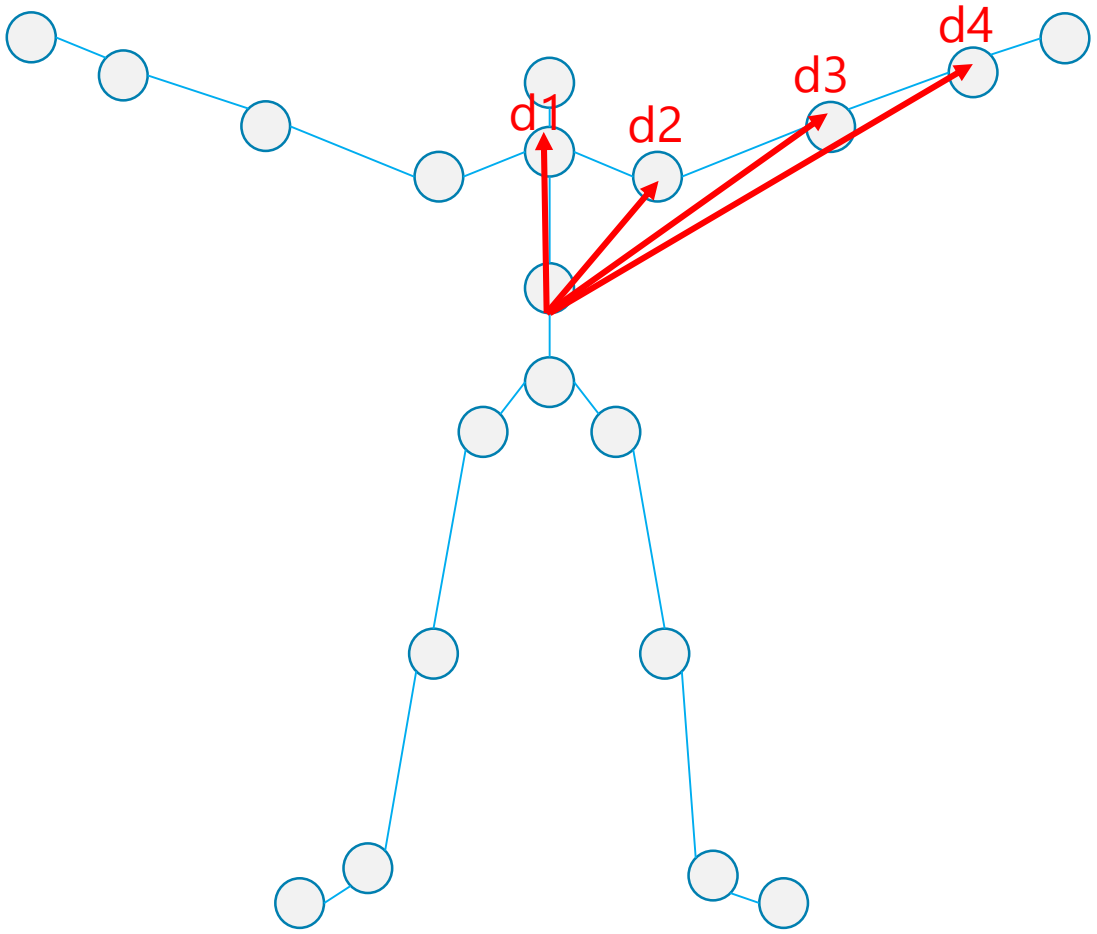
Kinect's location

User location



Distance Model

Use distance between center of body and joints

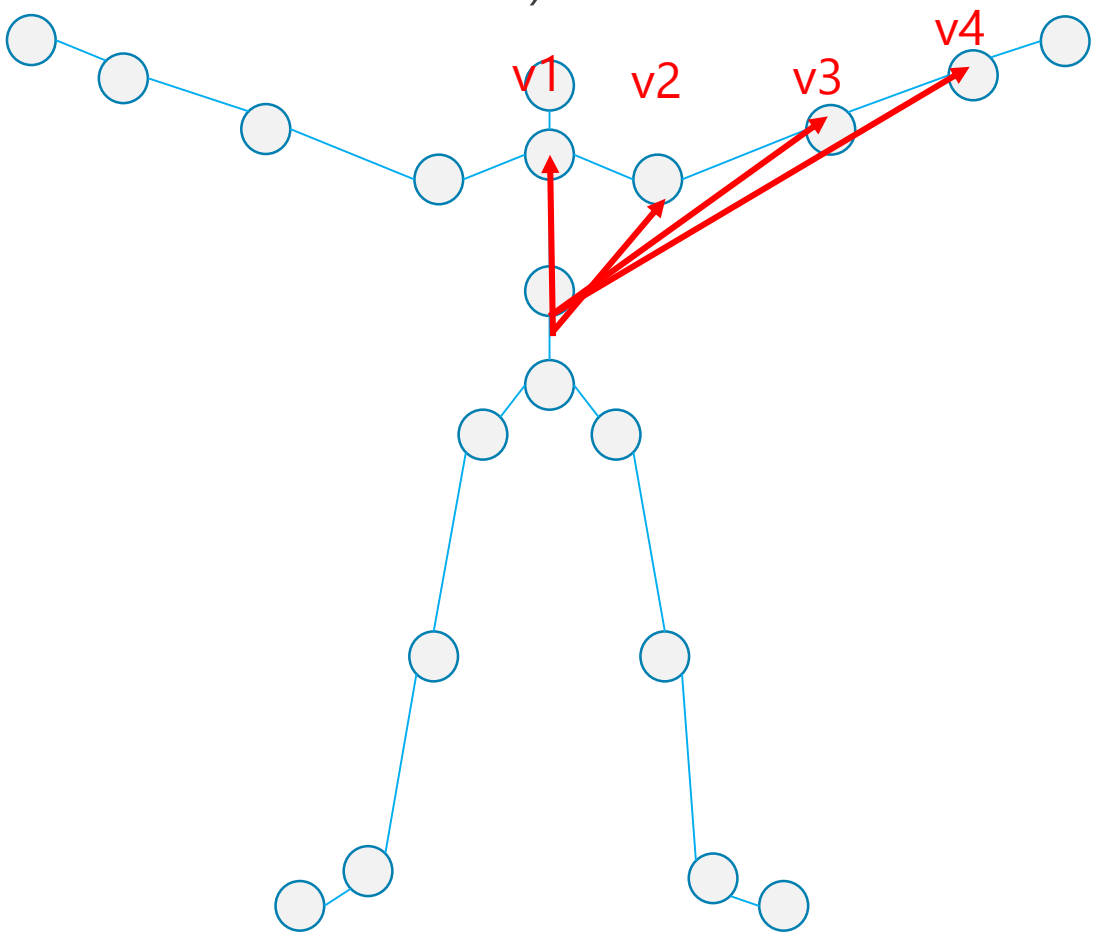


Distances vector:

d1: 33
d2: 30
d3: 49
d4: 53
...

Displacement Model

use displacements between center of body and joints (as distance but using difference of vector).

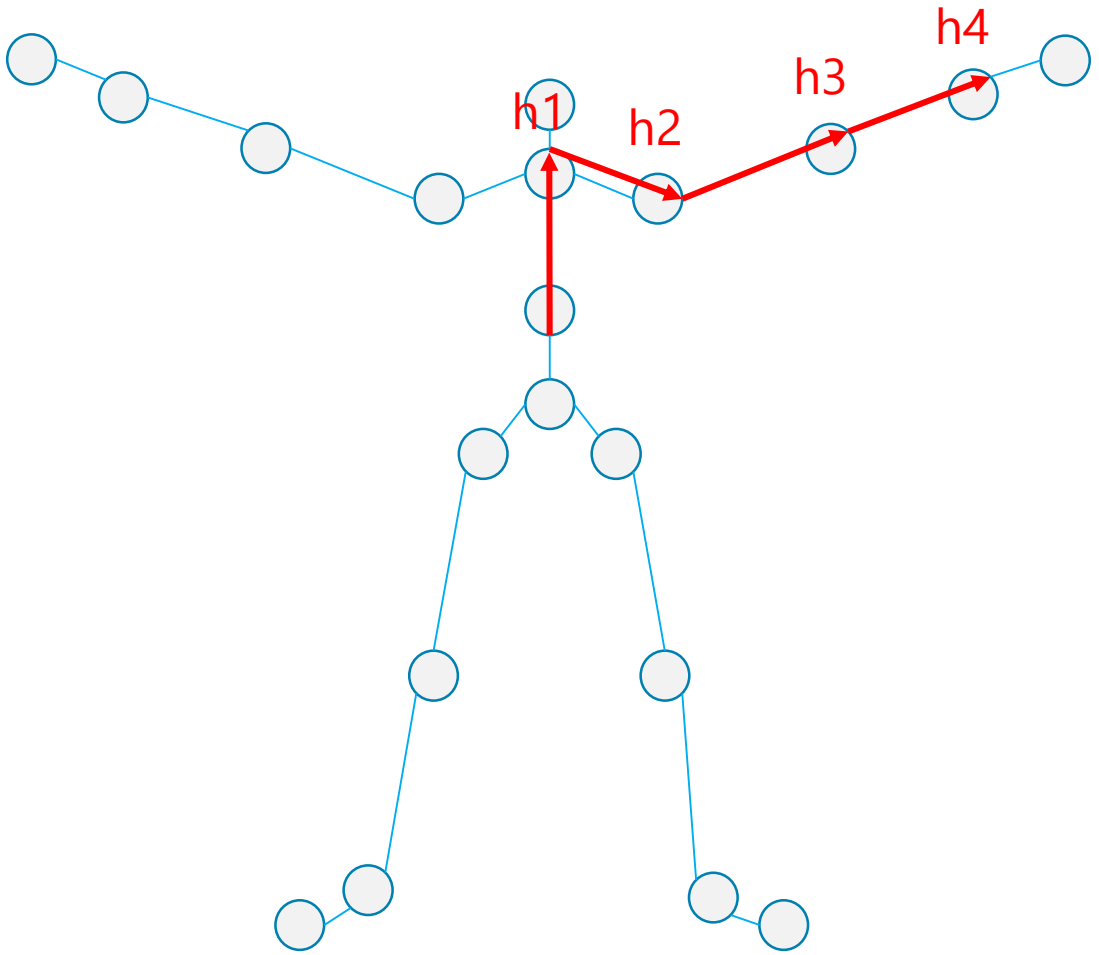


Displacement vector:

v1: 0, 33, 0
v2: 15, 25, 0
v3: 35, 27, 0
v4: 43, 32, 0
...

Hierarchical Model

Skeletal body model as a tree where joints are nodes and the spine joint is the root. A feature represents the displacement between joint and its parent position.



Hierarchical vector:

h1: 0, 33, 0
h2: 15, -7, 0
h3: 20, 9, 0
h4: 18, 9, 0
…

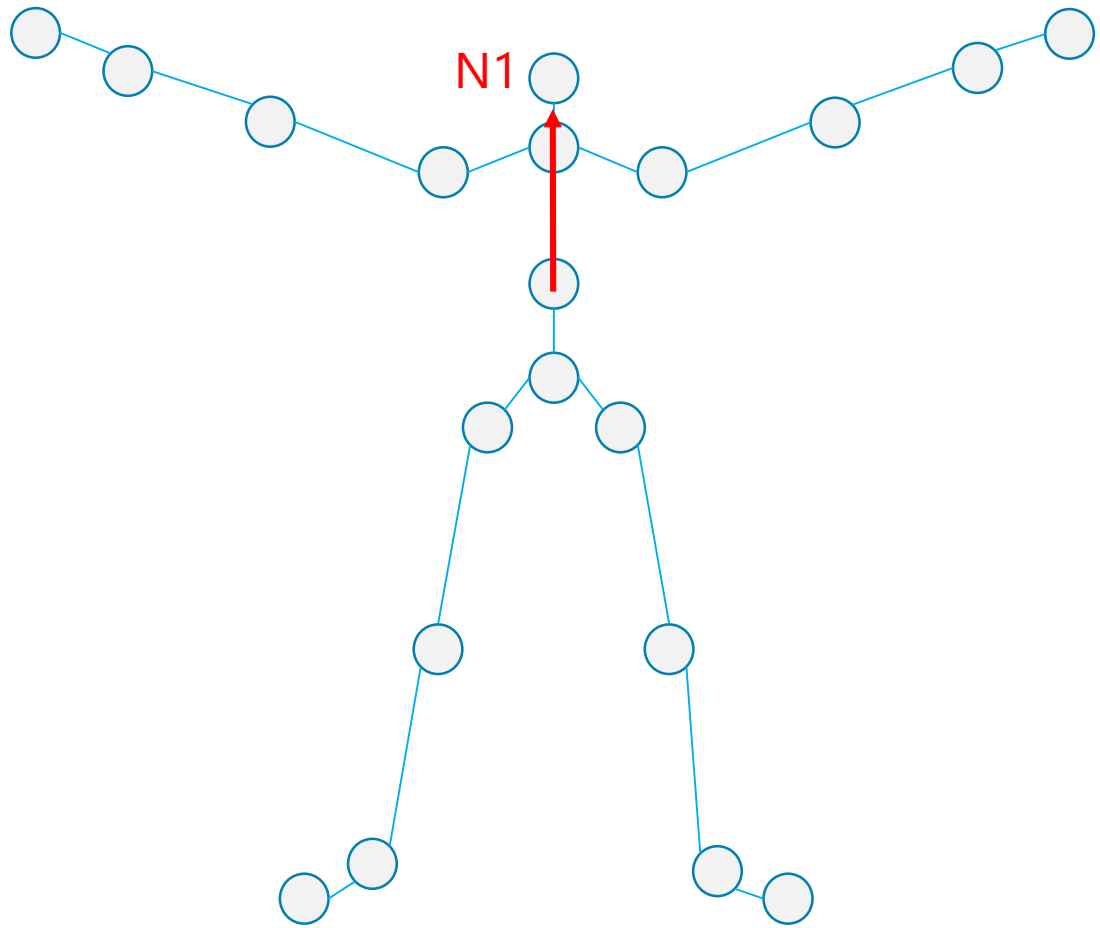
Normalization

One dissimilarity source between the captured data from different individuals is related to their height.

The acquired skeletal data can be scaled properly, simply by dividing all limb lengths by a value that is proportional to a given user’s height.

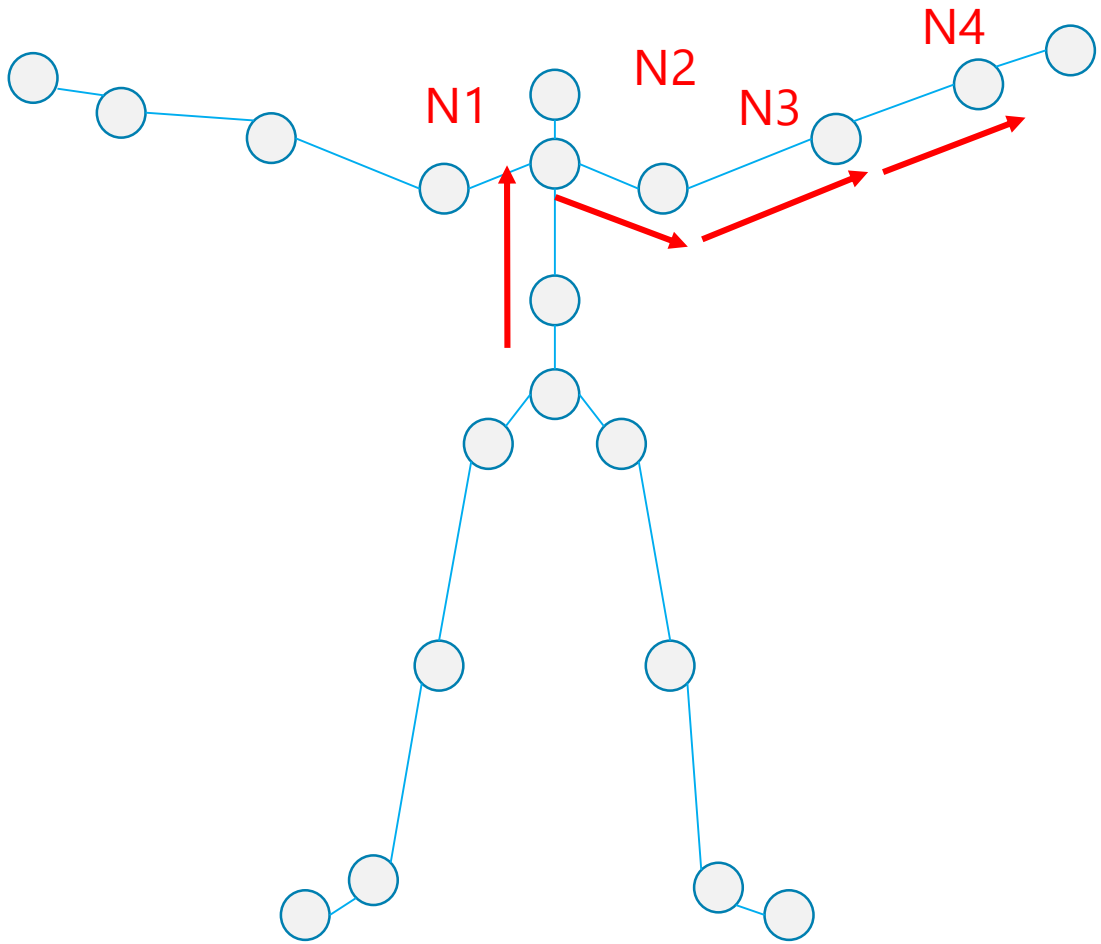
Relative Normalization

Use the distance between spine and head joints to normalize all information



Unit Normalization

Scale all limb segments connecting two joints to unit length before computing the aforementioned features. This way, the vectors lose their length and keep their directions only.



Speech Recognition

Artificial Intelligence for Kinect

Several Info

The Kinect sensor's microphone array

It is an excellent input device for speech recognition-based applications.

Language Support

Improved audio quality

Beamforming and source localization

Noise problem?

Locale	Language
Australia	English
Canada	English
Canada	French
France	French
Germany	German
Ireland	English
Italy	Italian
Japan	Japanese
Mexico	Spanish
New Zealand	English
Spain	Spanish
United Kingdom	English
United States	English

Defining a grammar

```
var directions = new Choices();
directions.Add(new SemanticResultValue("forward", "FORWARD"));
directions.Add(new SemanticResultValue("forwards", "FORWARD"));
directions.Add(new SemanticResultValue("straight", "FORWARD"));
directions.Add(new SemanticResultValue("backward", "BACKWARD"));
directions.Add(new SemanticResultValue("backwards", "BACKWARD"));
directions.Add(new SemanticResultValue("back", "BACKWARD"));
directions.Add(new SemanticResultValue("turn left", "LEFT"));
directions.Add(new SemanticResultValue("turn right", "RIGHT"));

var gb = new GrammarBuilder { Culture = ri.Culture };
gb.Append(directions);

var g = new Grammar(gb);
```

```
<grammar ...>
<rule id="rootRule">
    <one-of>
      <item>
        <tag>FORWARD</tag>
        <one-of>
          <item>forward</item>
          <item>straight</item>
        </one-of>
      </item>
      <item>
        <tag>BACKWARD</tag>
        <one-of>
          <item>backward</item>
          <item>backwards</item>
          <item>back</item>
        </one-of>
      </item>
    </one-of>
  </rule>
</grammar>
```

Let's see some code…

```
RecognizerInfo ri = GetKinectRecognizer();

if (null != ri)
{
   recognitionSpans = new List<Span> { forwardSpan, backSpan, rightSpan, leftSpan };

   this.speechEngine = new SpeechRecognitionEngine(ri.Id);

   using (var memoryStream = new MemoryStream(Encoding.ASCII.GetBytes(Properties.Resources.SpeechGrammar)))
   {
      var g = new Grammar(memoryStream);
      speechEngine.LoadGrammar(g);
   }

   speechEngine.SpeechRecognized += SpeechRecognized;
   speechEngine.SpeechRecognitionRejected += SpeechRejected;

   speechEngine.SetInputToAudioStream( sensor.AudioSource.Start(),
                              new SpeechAudioFormatInfo (EncodingFormat.Pcm,16000, 16, 1, 32000, 2, null));
   speechEngine.RecognizeAsync(RecognizeMode.Multiple);
}
```

Let's see more code…

```
private void SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
{
   const double ConfidenceThreshold = 0.3;

   if (e.Result.Confidence >= ConfidenceThreshold)
   {
      switch (e.Result.Semantics.Value.ToString())
      {
         case "FORWARD": // do something
         case "BACKWARD": // do something
         case "LEFT": // do something
         case "RIGHT": // do something
       }
    }
    . . .
}
```

Let's see more code…

```
private void WindowClosing(object sender, CancelEventArgs e)
{
     if (null != this.sensor)
     {
         this.sensor.AudioSource.Stop();

         this.sensor.Stop();
         this.sensor = null;
     }

     if (null != this.speechEngine)
     {
         this.speechEngine.SpeechRecognized -= SpeechRecognized;
         this.speechEngine.SpeechRecognitionRejected -= SpeechRejected;
         this.speechEngine.RecognizeAsyncStop();
     }
 }
```

demo

Speech recognition:
audio basics

kinect

Application Showcase

Matteo Valoriani
mvaloriani AT gmail.com
@MatteoValoriani

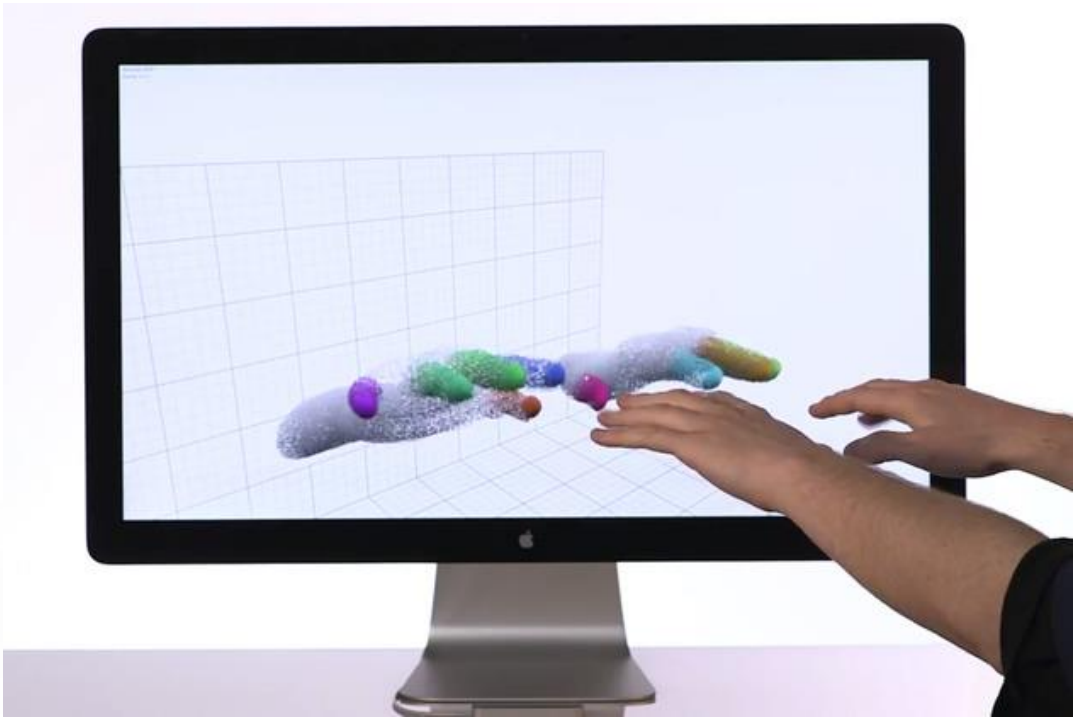
What Next?

Kinect 2, Leap Motion, Intel Perceptual Computing

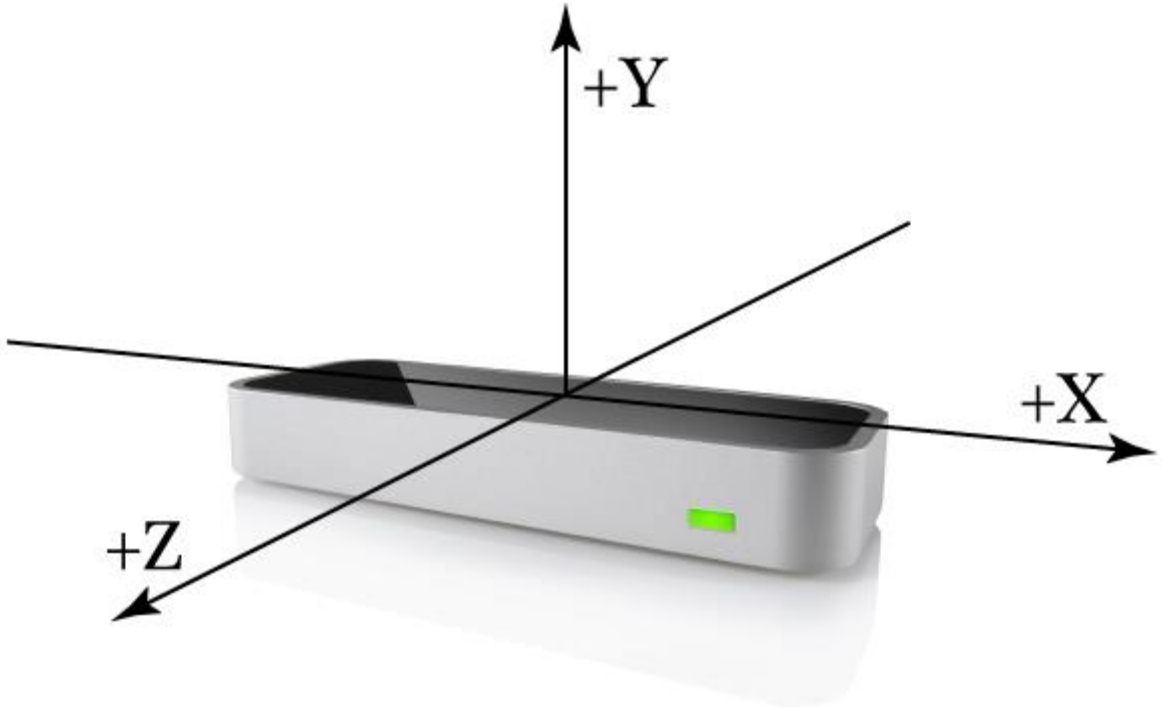
Matteo Valoriani
mvaloriani AT gmail.com
@MatteoValoriani

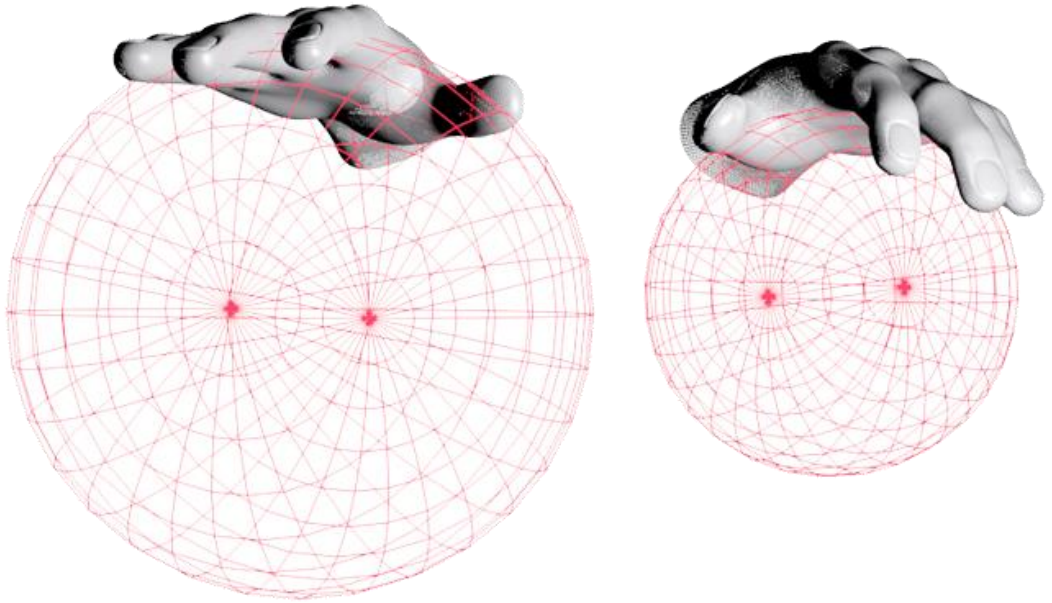
Leap Motion

https://www.youtube.com/watch?v=_d6KuiuteIA



Leap Motion for Developers





C++

C# and Unity

Java

JavaScript

Python

Intel Perceptual Computing

https://www.youtube.com/watch?v=WePIY7svVtg

Xbox One - Kinect 2

Xbox One - Kinect 2

Xbox One - Kinect 2

http://youtu.be/Hi5kMNfgDS4

Which to choose? ALL

Leap Motion Advantages:

Finger tracking is fast and accurate.

Smaller and less expensive: Developer friendly

Framework support: .NET, Processing, Cinder, etc.

Compatible: Mac OS and Windows.

Leap Motion Issues:

Sensing range is fairly limited.

Only fingers are tracked.

There is no skeleton or face tracking.

No access to the raw sensor data.

Best for:
Controlled kiosk environments with a pointing-based UI.
Generally best for general audience desktop apps which can be distributed in the Airspace store.

Which to choose? ALL

Intel Advantages:

Smaller and less expensive

Close-range tracking

Hand posture/gesture recognition

Facial analysis

Speech (built-in support for speech synthesis powered by Nuance).

Raw data

Framework support: Processing, Unity and OpenFrameworks

Intel Issues:

Getting some of the deeper features (like age and gender detection) to work is a bit tricky.

Device and software are in beta.

Due to the close range of the tracking system, hand gestures must be designed such that a user's hand doesn't occlude their own view of the display.

Best for:

Desktop/laptop applications where the user will be seated in front of the PC. Close range applications where features, apart from hand tracking and recognition, are necessary without too much precision or accuracy.

Which to choose? ALL

Microsoft Kinect Advantages:

Skeletal tracking

Face tracking

Multiple sensors:

Raw data

Voice control

Microsoft Kinect Issues:

The device is fairly large

A dedicated power cord is required

The amount of data generated by the sensor also tends to saturate a USB controller

Kinect cannot easily distinguish individual fingers on a hand

Most of the features listed above require the Microsoft SDK, which is only supported for desktop applications on Windows 7 and 8.

Best for:

Kiosks, installations, and digital signage projects where the user will be standing fairly far away from the display.


... TIRED?

Q&A

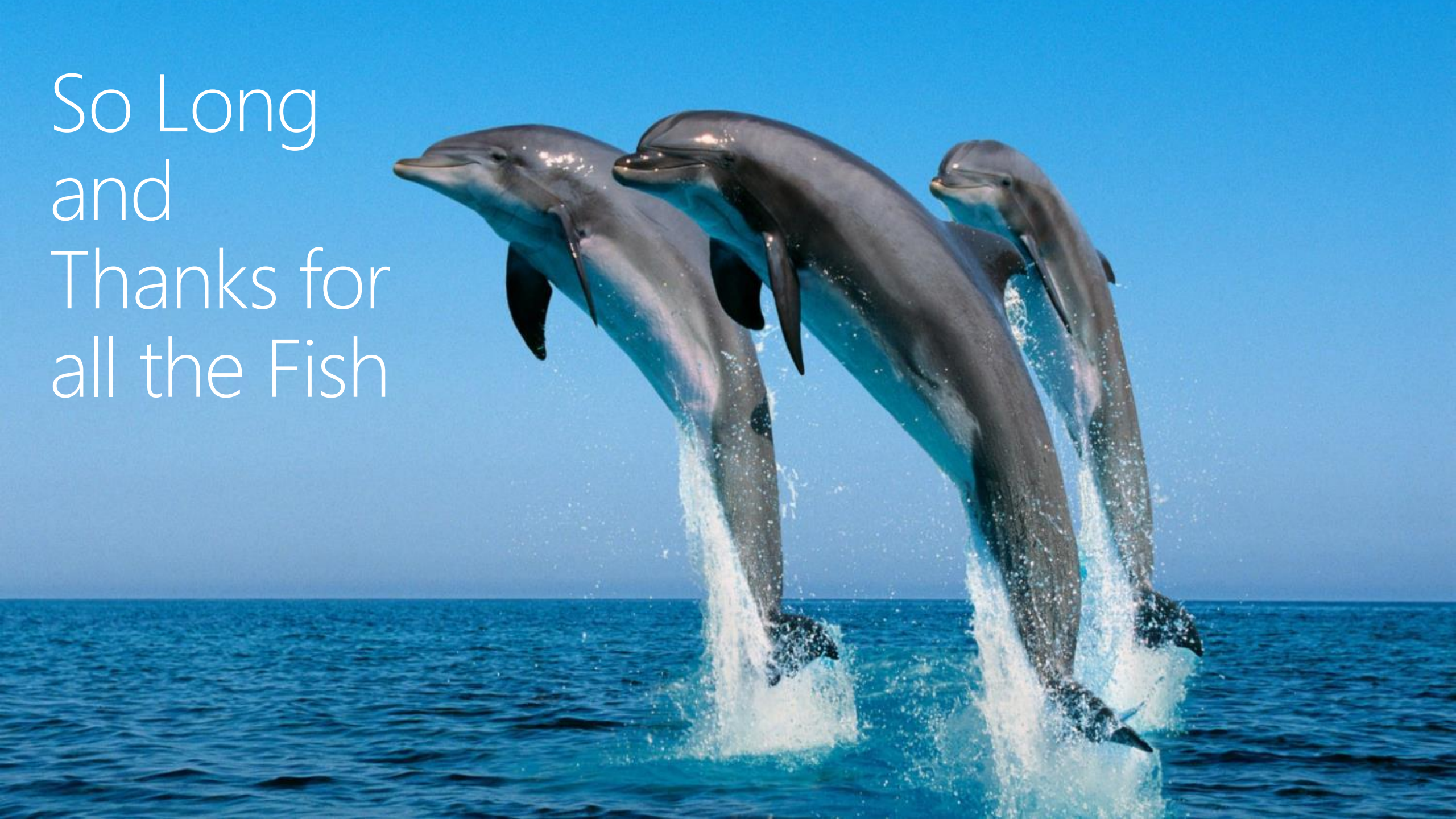
Tutto il materiale di questa sessione su

http://www.communitydays.it/

@MatteoValoriani


FOLLOW ME ON
TWITTER OR THE
KITTEN GETS IT:
@MatteoValoriani

So Long and Thanks for all the Fish



Resources and tools

http://channel9.msdn.com/Search?term=kinect&type=All (Others projects)

http://kinecthacks.net/ (Others projects)

http://www.modmykinect.com (Others projects)

http://kinectforwindows.org/resources/ (Microsoft SDK)

http://www.kinecteducation.com/blog/2011/11/13/9-excellent-programming-resources-for-kinect/ (resources)

http://kinectdtw.codeplex.com/ (gesture recognition library)

http://kinectrecognizer.codeplex.com/ (gesture recognition library)

http://projects.ict.usc.edu/mxr/faast/ (gesture recognition library)

http://leenissen.dk/fann/wp/ (gesture recognition library)

Credits & References

Dan Fernandez, Kinect for Windows Quickstart Series

Zsolt Mathe, The Magic Behind Kinect

27. Freedman, B.; Shpunt, A.; Machline, M.; Arieli, Y. Depth Mapping Using Projected Patterns. U.S. Patent 2010/0118123, 13 May 2010.

28. Fraser, C.S. Digital camera self-calibration. ISPRS J. Photogramm. Remote Sens. 1997, 52, 149–159.

Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications, Kourosh Khoshelham * and Sander Oude Elberink

http://campar.in.tum.de/twiki/pub/Chair/TeachingSs11Kinect/2011-DSensors_LabCourse_Kinect.pdf

References

"A Brief History of Human Computer Interaction Technology"

Brad A. Myers

"Neural Networks – A Systematic Introduction"

Raúl Rojas

"A Gesture Processing Framework for Multimodal Interaction in Virtual Reality"

Marc E. Latoschik

Gamefest 2010 – "Gesture Recognition"

Lewey Geselowitz & J. McBride

Kinect Developer Summit 2011 – "Inside Kinect Skeletal Tracking Deep Dive"

Zsolt Mathe