



Project Report

Study Project(CS F266) | Lab Project(CS F366)
On
Speaker Detection Using Multiple Kinects

Supervising Instructor: Dr. J.L. Raheja, Senior Scientist, CEERI Pilani

Prepared by:

Aditya Raisinghani	2011A7PS044P
Utkarsh Verma	2011A7PS137P
Viraj Prabhu	2011A7PS044P

(Students pursuing B.E.(Hons.), Computer Science at BITS Pilani, Pilani Campus)

Objective:

To create a software for speaker detection using multiple Microsoft Kinect sensors, and to display the face of the current speaker.

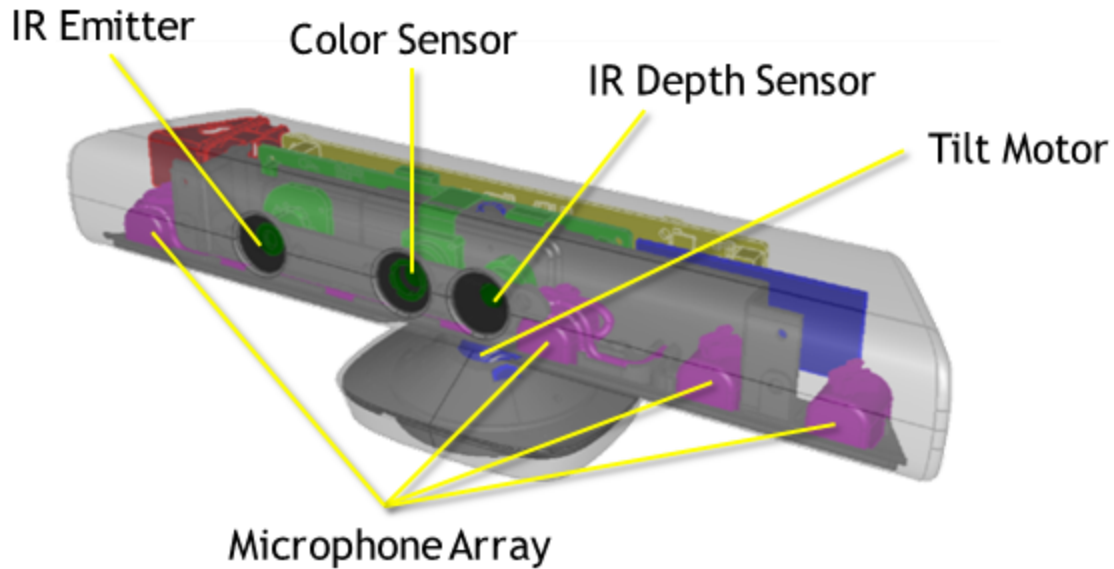
Description:

Multiple Microsoft Kinect sensors are positioned centrally, pointed in different outwards directions to cover a wide field of vision. When multiple speakers seated around this arrangement are speaking, the software can detect the speaker having maximum sound intensity by using the microphone array of the Microsoft Kinect, and also detect his lip movement using the Depth, Color and Skeletal stream based tracking to verify that he is the current speaker. Then his face is to be displayed on the screen.

Implementation Details:

The Microsoft Kinect is a depth/motion sensing input device by Microsoft for Xbox 360 and Xbox One video game consoles and Windows PCs. It enables users to control and interact with their console through a natural user interface using gestures and spoken commands.

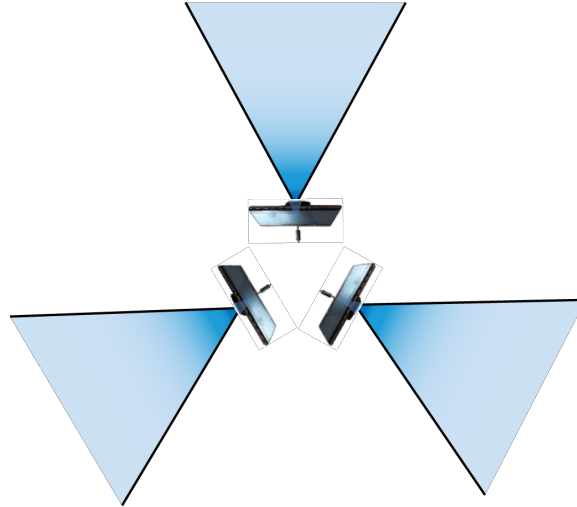
The device features an RGB camera, Infrared based depth sensor and multi-array microphone with vertical tilt motors. These support provide full-body 3D motion capture, facial recognition and voice recognition capabilities.



A diagram of the Kinect for XBOX 360

A Kinect Software Development Kit is freely available from Microsoft. We have used Kinect SDK version 1.8. We have developed the code in C# using Microsoft Visual Studio 2012 as the Integrated Development Environment.

The software can be used to cover a wide field of vision by placing multiple Kinect sensors centrally, pointing outwards. We plan to use 3 devices each in directions 120° apart. Each Kinect sensor has a 57° wide horizontal field of view. The speakers are generally assumed to be sitting around the sensors. This can be applied to situations of such as a conference or a meeting.



A top view of 3 Kinects setup in the described manner

The code has been written to run with a variable number of Kinect sensors. The number of Kinect sensors can be changed by changing the `MaxKinect` field in file `MainWindow.xaml.cs`, in class `AudioKinect`:

```
private const int MaxKinect;
```

This variable is initialized to 3 currently. Support for more sensors can be added/removed simply by changing this value.


“Important Note: Please keep in mind that Windows only supports 1 Kinect device per USB hub. Often low power devices like laptops, and some desktops have multiple USB ports over a shared common USB hub. Connecting multiple Kinects to such common hub ports will fail to work successfully.”

The code is organized in two main classes- AudioKinect and InitializeKinect:

A) InitializeKinect:

This class simply initializes a Kinect sensor. An object of this class is instantiated for every Kinect sensor that has been connected, at runtime. Each object of this class also starts a separate thread: 'AudioReadingThread' to capture audio from its microphone array. This thread handles the polling of the audio stream and updates the on screen audio intensity visualization at every tick. The class also handles the application exit, stopping the reading thread and the sensor on exit.

B) AudioKinect:

This class instantiates an object of the 'InitializeKinect' class for every Kinect sensor that has been connected in an array: 'audioKinect', and adds event handlers to it. Conditional code is used within the 'AudioSourceSoundSourceAngleChanged' event handler of this class , which responds to events of change in audio source angle.

The intensity level has to be above a fixed background noise intensity level, to ensure that random background noise does not interfere. This is controlled by the variable, currently set to 0.2):

private const double BackgroundNoiseIntensity;

It can be set to a higher value to account for noisier conditions.

To ensure that the AllFrameReady handler for the different sensors are attached and removed properly, a variable firstCheck is used (0 for first time, 1 when handler is attached, 2 for after that):

private int firstCheck = 0;

This code enables and renders the ColorStream of the Kinect sensor with the maximum sound intensity. This happens only after a particular sensor records the maximum count for a specific number of frames, which is currently set to 3, after this code snippet:

if (voiceDetectionArr[maxIntensityDevice] == 3)

Within the event handler for change in audio source angle, whenever a particular sensor records the maximum sound intensity for three or more frames (not necessarily consecutively), the color stream of that sensor is rendered and displayed on the screen. This check is performed for every change in audio source angle in the processed frames. This ensures that a sensor's colour stream is rendered only when it has been established that the person speaking is in front of that particular sensor, and practically a frame count of 3 has given us satisfactory results.

For face tracking and lip movement, we are using the Kinect Toolkit which provides specific attributes for the detected faces using depth, color and skeletal stream data. The 'facetrackers' array is a list of FaceTracker objects. Its declared size is equal to MaxKinect value, and each detected Kinect initialises its own facetracker object supplying it with its own multiple stream input data.

On successful run, the tracker object provides 'Animated unit coefficients' which have the following values for *JawLower* unit.

$-1 \leq x \leq 0$	\Rightarrow <i>mouth is closed</i>
$0 < x < 1$	\Rightarrow <i>mouth is opening</i>
$x == 1$	\Rightarrow <i>mouth is fully opened</i>

We are using these values to find the speaking motion of a mouth. This part did not execute as planned within the time constraints, so we have

disabled all the related face tracking code by commenting them. The commented final version falls back to the detection of speakers using speech sound intensity levels for camera switching.

Building and running the code:

The code can be built and run directly in visual studio, in debug mode by pressing <F5>, after connecting any number of Kinect Sensors upto the value of 'MaxKinect'.

It will load an output window, displaying a sound intensity meter showing intensity and direction of sound for the sensor recording the maximum intensity. On startup, the output window will show the color stream of the Kinect sensor which records the sound of the maximum intensity for 3 frames. This process and switching between cameras is automated and does not require any user intervention.

Finally, closing the window will cause the program to complete and end execution.

Conclusion:

This software is capable of detecting the current speaker accurately. It has been tested on an arrangement of two sensors, but can be scaled up to a larger number. Checks have been included to ensure quick and lag-free switching between the cameras, and to filter out background disturbances causing possibly erroneous detection. However, code written to verify lip movement by the current speaker is not working as expected at the moment. Barring this, it can be used to detect and display the current speaker using sound intensity and location and can be applied in scenarios such as conferencing and meetings.