Beamforming with Kinect

Vanya Avramova, Andrea de Giorgio and Alexander Solsmed

DT2118 Speech and Speaker Recognition
Project Report

May 20, 2014

Abstract

The beam-forming algorithm is highly relevant for speech recognition, because it allows for the identification of the active speaker in natural environments where there may be several conflicting sound sources, and background noise. By performing sound localization, beamforming allows for relevant signals to be amplified, and "noise" suppressed. This makes the task of a speech recognizer simpler, since it can focus on the relevant pieces of the data stream. The Microsoft Kinect device includes an implementation of the beamforming algorithm. In this report, we will investigate how the algorithm is implemented for the Kinect. In addition, we will use the 4-channel audio stream exposed by the device and attempt our own implementation based on the TDOA (time difference of arrival) method of triangulating sound source. Our algorithm may prove useful in situations where Kinect is used in a non-windows environment, or with an open source SDK that does not expose the native beamforming implementation. We will also demonstrate how the algorithm can be used in combination with data from the Kinect's depth sensor to aid speech recognition tasks in gameplay scenarios where the soundtrack of the game competes with the speech signal coming from the players.

1. Introduction

One of the main challenges in Speech Recognition is dealing with signal pollution stemming from background interference, reverberations or competing audio sources. Beamforming (also called sound localization) is the concept of determining the direction of the relevant audio source, by analyzing the audio streams processed by a microphone array, in order to enhance it, while other sources are suppressed, thus reducing noise in the audio input signal.

The aim of this project is to implement a location detector algorithm using the TDOA (time difference of arrival) method. We will also provide a test app that shows the performance of the native Kinect SDK beamforming algorithm side by side with our implementation. Finally, we will demonstrate how the Kinect's microphone array can be used as a directional microphone and pointed at a particular player in scenarios where the speech commands coming from the user conflict with another audio source native to the scenario (games, immersive multimodal environments, etc.).

2. Background

The main objective of beamforming is sound localization. When the sound is perceived by a single microphone, it is challenging to accurately predict the sound origin. A microphone array is a system with multiple, closely positioned microphones. Microphone arrays are superior to signal-channel systems since they can capture the signal from several points, and, with proper algorithms, both spatial and temporal information can be inferred from the sound field. This information is essential for the implementation of more sophisticated noise-reduction and automatic echo cancellation algorithms. By using the fact that sound from the audio source arrives at each microphone in the array at slightly different times, the direction of the audio source can also be determined. The microphone array can then be "steered" as a directional microphone to give special preference to that direction in space, while signals from other location are attenuated.

2.1. The Kinect Sensor

The Kinect sensor includes a 4-element linear microphone array (Fig. 1). The microphone array can supply four channels of 24 bit resolution at 16 kHz sampling rate [7].

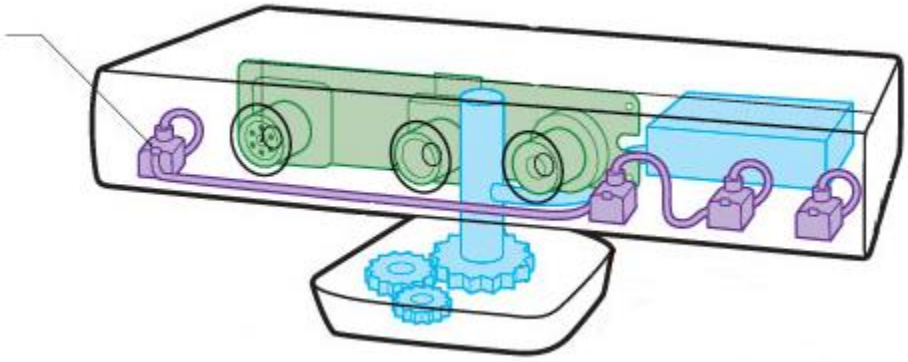

Fig. 1. The Kinect microphone array, shown here in purple [5]

The Kinect sensor captures both video and audio data, and exposes it as 4 data streams through the NUI API [2]:

- Audio Stream
- Color Stream
- Depth Stream
- Skeletal Stream (created by processing depth data)

This data is sent to the subscriber as an event-based data stream. The skeletal stream exposes the skeletal joint positions of each person in view of the sensor, including the head position. This can be used as an extra feedback when localizing the speaker source, in addition to the signals from the microphone array. The Kinect SDK includes a managed audio API that allows operations such as starting, recording, and stopping the audio stream. The stream can be recorded in WAVE format [3], and includes interweaved data from all 4 channels. The Kinect sensor already includes an internal implementation of beamforming, which we will discuss in 2.2. The managed API also exposes events that provide the audio source and beam directions to the application, as an angle defined from the sensor location. Both angles are defined in the x-z plane perpendicular to the z axis of the sensor, and have a range of [-50, 50] degrees (Fig. 2) .

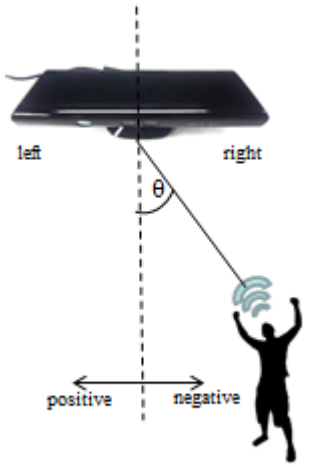

Fig. 2. Range and direction of the sound source and beam angles [5]

- A degree of zero means that the source is centered in front of the sensor.
- A positive value indicates that the source is to the right side of the user (positioned at angle 0), or equivalently, on the left side of the sensor.
- A negative value indicates that the source is to the left side of the user (positioned at angle 0), or equivalently, on the right side of the sensor.

2.2. Kinect's beamforming algorithm

A beamformer algorithm takes advantage of an arbitrary microphone array geometry and makes efficient use of noise models for ambient, instrumental and microphone directivity patterns. Classical geometries can be linear or circular, composed by microphone arrays with four to eight elements. The Microsoft Kinect uses a four-microphone linear array.

Algorithms for beamformer design with optimal noise suppression initially were based on finding of parametric solutions, given the microphone array geometry. To reduce complexity, later designs are based on finding near-optimal solutions for different target criteria; well-known algorithms include constant directivity beamformer, minimum-variance distortionless response (MVDR), linearly-constrained minimum variance (LCMV), general side lobe cancellers and others [8]. In practice, measured parameters of microphone arrays differ from theoretical estimations, leading to a reduction in noise suppression. Better performance can be achieved with adaptive algorithms, which compensate for changes in position of the desired sound and noise sources. In scenarios such as sound capture in small conference rooms, sound sources change rapidly, so that adaptive beamforming algorithms may not converge fast enough. With an efficient algorithm, signals can be processed in real time [8].

A generic beamformer design algorithm based on maximal usage of prior knowledge is well described by Tashev & Malvar (2005) and is very close to what the Kinect actually uses. The near-optimal solution is found by minimizing the noise mean-square value at the beamformer output. This approach is called MVDR (Minimum Variance Distortionless Response). It can be briefly described in the following steps:

- Definition of the target beam shapes.
- Pattern synthesis: find a set of weights that fit the real beam shape into the target via a least-square requirement.
- Normalization: ensures unit gain and zero phase shift for signals originating at the focus point.
- Optimization of the width: one-dimensional search on the target area width with criterion of total noise suppression. It recomputes the optimal normalized weights to find the minimum average noise energy in a certain interval around the work point.
- Calculation of the weight matrices for each beam set.

Assuming that combining the signals from all sensors is just a weighted sum, the algorithm associates a beam shape to each set of weights. Every beam shape represents the beamformer complex gain as function of the sound source position.

3. Our Implementation of Beamforming

While the Kinect SDK already provides the result from the built-in beamforming algorithm, and options to steer the beam automatically or manually, this feature is may not be included on open-source Kinect SDKs, or if someone wishes to use the Kinect as a simple 4-channel microphone array. For the project, we provide our own implementation which pinpoints the angle from which the sound was coming from for each pair of microphones from the array based on the TDOA method. We attempt to solve a full triangulation problem, localizing the position of the sound as well as from what direction it is coming. This problem requires at least three microphones so solve, and is over determined with four microphones. A simplification of the TDOA problem is to consider the incident wave front planar, instead of spherical, as seen described in Fig. 3.

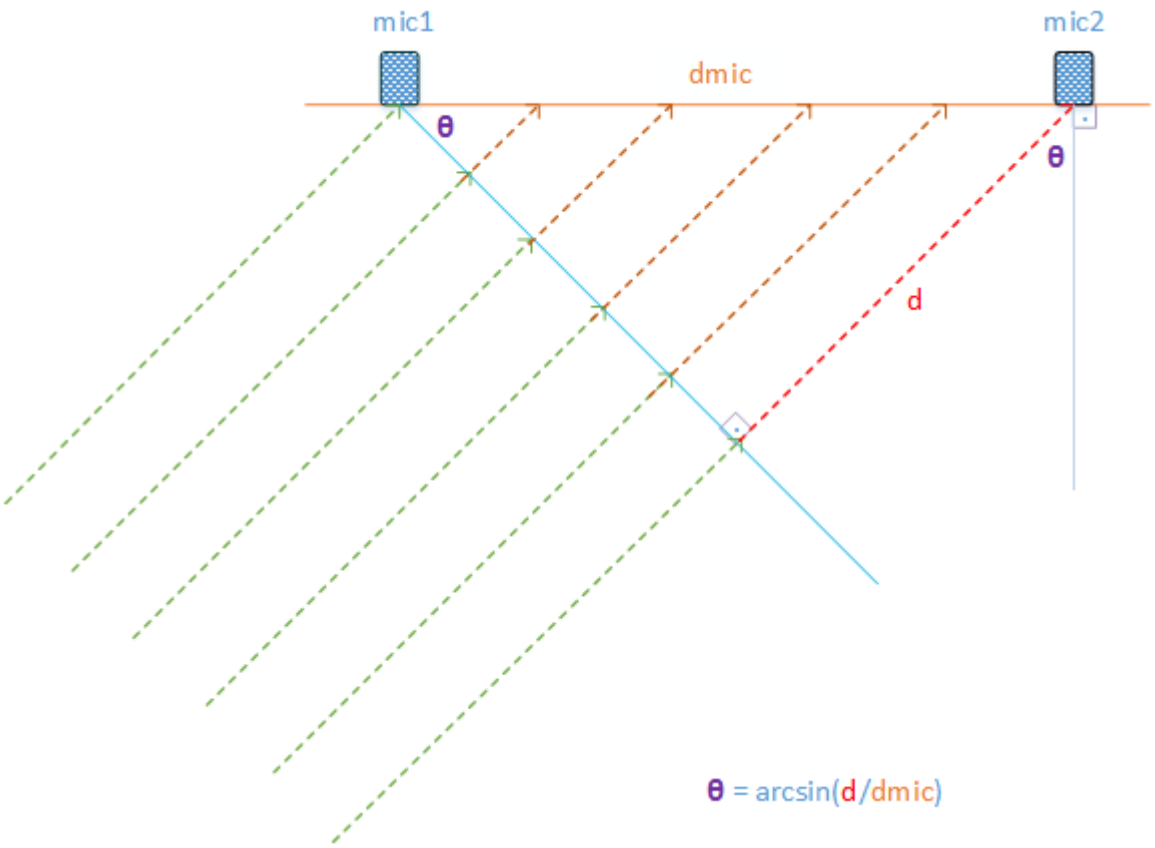

Fig. 3. TDOA (time difference of arrival) using planar wave front triangulation

Many biological organisms, that receive audio signal from two channels, have the ability to pinpoint the location of the audio source. An examination of the biological algorithm used for this is beyond the scope of this paper, but the important point is that two channels make estimation the source possible by using a fairly straightforward trigonometry principle. In real life, audio waves are dispersed from the source in all directions (a sphere in 3D), but for smaller distances we can assume that the waves travel as a plane (Fig. 3). Regardless of the shape of the wave front, the angle of the sound source can be inferred by measuring the time delay between the sound samples recorded by each microphone.

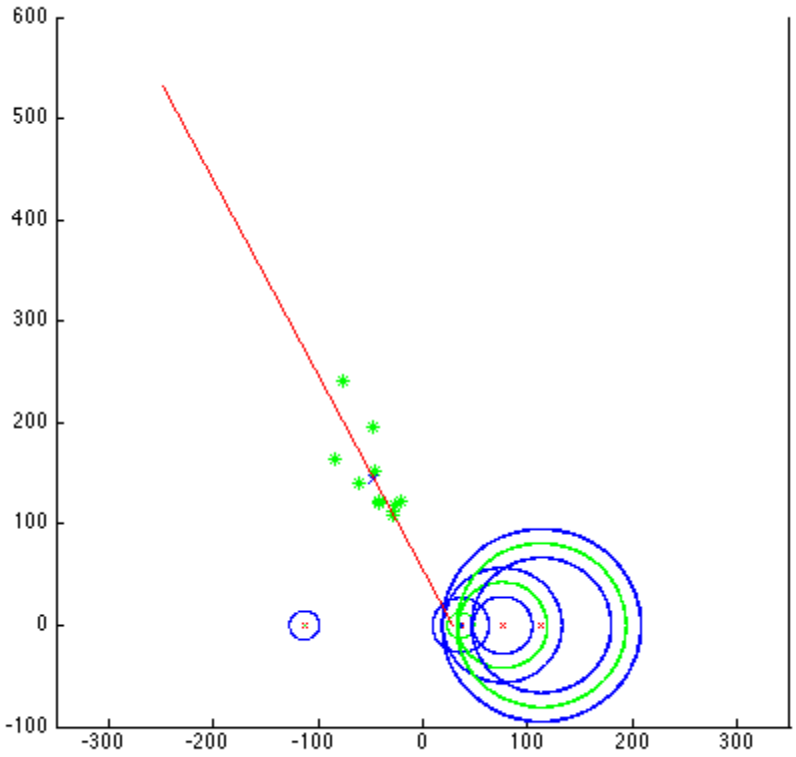

Fig. 4. The geometry of our algorithm. Showing un-discretized delay circles but still with the error margins (blue) of the time discretization. Green dots are source estimations, red line is their mean direction.

Using the time delays we can draw "delay circles" around each microphone, which we then use to trace the sound back to its source by expanding them and looking for a common point of intersection.

The distance between the microphones is known, and we assume that sound travels at constant speed *v* in the environment where the microphone array is located. The sound waves reach the microphones at different times t = [t1, t2, t3, t4]. Using these times, we compute the time delay of each microphone compared to the earliest microphone

dt = t - min(t)

These time delays are then converted to differences in distance, as

dd = dt / v

For each pair of microphones, we compute the points of intersection of their circles. We do not know how far away the sound was absolutely (only relatively between the two microphones), and thus have a free parameter. As one intersection point of two circles (in our case, the point on the front side of the Kinect) moves along a straight line if both circles are expanded by an equal addition to their radii and this addition is exactly our free parameter, each microphone pair gives us a straight line of positions in the solution space. We find this line by solving the intersection problem at two delta radii, namely 700 mm and 1300 mm.

We do this for all six unordered pairs of four microphones, and that procedure is repeated 81 times since we solve the system for the minimum, mean and maximum time delays, in different combinations across the four microphones, caused by the fact that microphones discretize time. In total, the system returns a cloud of possible locations for the sound source. The mean point of the cloud is computed and the direction of the sound is given by the line between that point and the mean location of the microphones.

To determine the time difference of arrival between each pair of microphones, we chose the method of cross-correlation. Cross-correlation can be used as a way to measure the similarity between the two signals by continually shifting ("lagging") one with respect to the other, and taking the sum of products of the corresponding pairs of points. Thus, the cross-correlation xcorr(a, b) between two signals a and b containing N data points can be expressed as:

$$xcorr(a,b) = \sum_{n=1}^{2*N} a(n) * b(n+j)$$

where j represents the number of lags (data points) by which a (or b) has been shifted. The xcorr is computed for a number of different lags over the length of signal in either direction (2*N) in order to establish the largest value of the correlation. Since the microphone array records the same signal in all channels, estimating this shift is possible by "overlaying" the signals and calculating the number of lags at the point at which they exhibit maximum overlap (Fig. 4 and 5). The shift ("lag value") at this point can be converted to time by using the known sample rate of the recording (in Hz). In our implementation, we limit the search space to N=11 since that is the maximum number of samples a correlation can happen thanks to the relatively short interdistance of the microphones. To alleviate the problem of time discretization, we interpolate 20 steps between samples in the sound stream before cross correlating (thus, the xcorr function will be looking at 2*11*20 = 440 samples).

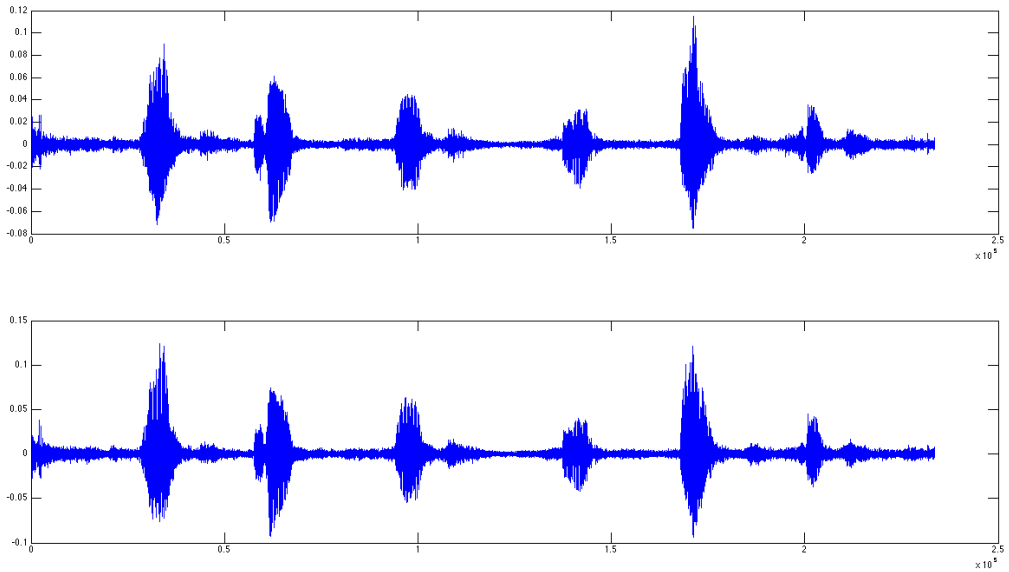

Fig. 5. Example of the same audio signal, recorded on two channels

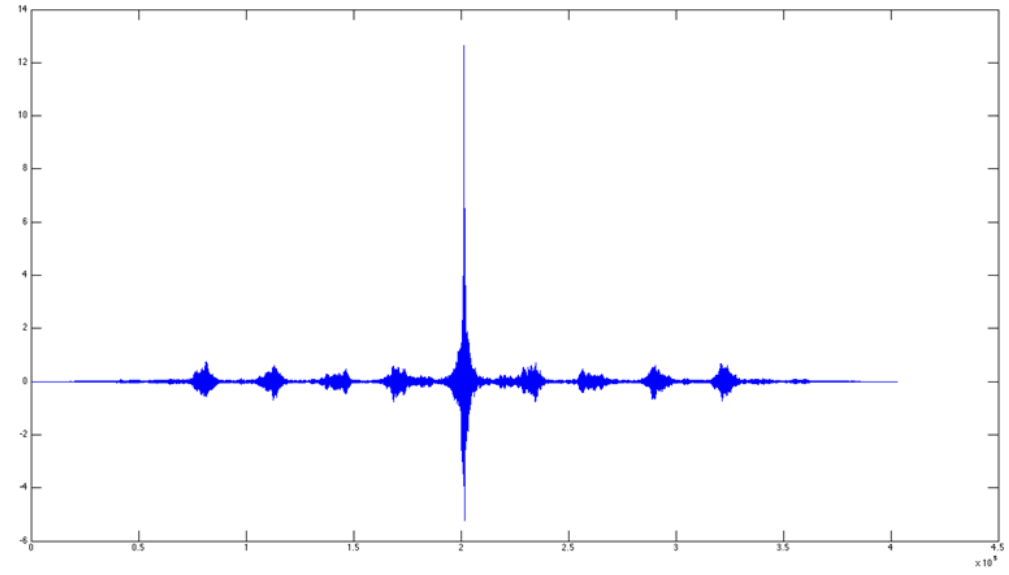

Fig. 6. Plot of the cross-correlation function of two channels, with a clearly distinguishable max overlap point

Given the time t, we can get d from Fig. 3. Once the distance d has been established, we can triangulate theta by using the equation:

$$\theta = \arcsin(d/dmic)$$

Since we have 4 microphones available, we can obtain an estimate for theta from each pair of microphones, and this constitutes our dt vector.

Simulation

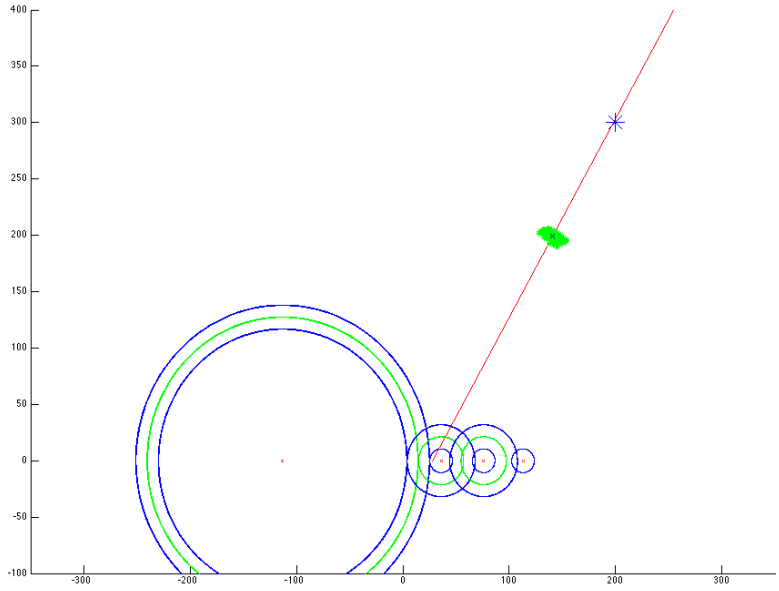

Fig. 7. Internal look at the geometry for a given sample point producing the maps show below (16 kHz sampling rate). The time discretization problem is evident from the fact that two microphones record the sound at the same sample, even though the sound is at an angle. Still though, the estimated angle is accurate.

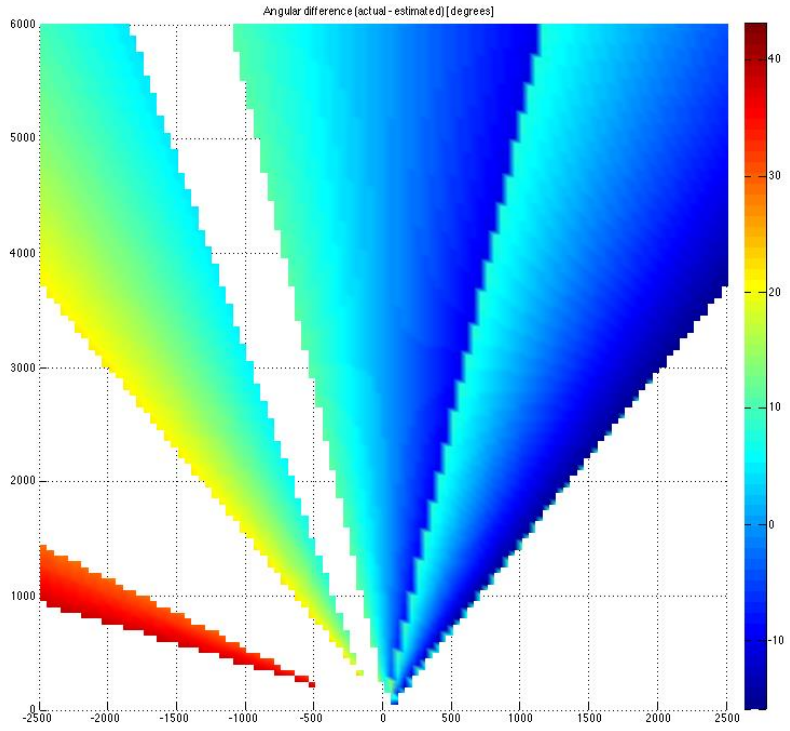

Fig. 8. At 4 kHz sampling rate, only a few principal directions where the algorithm solves correctly (sky blue color).

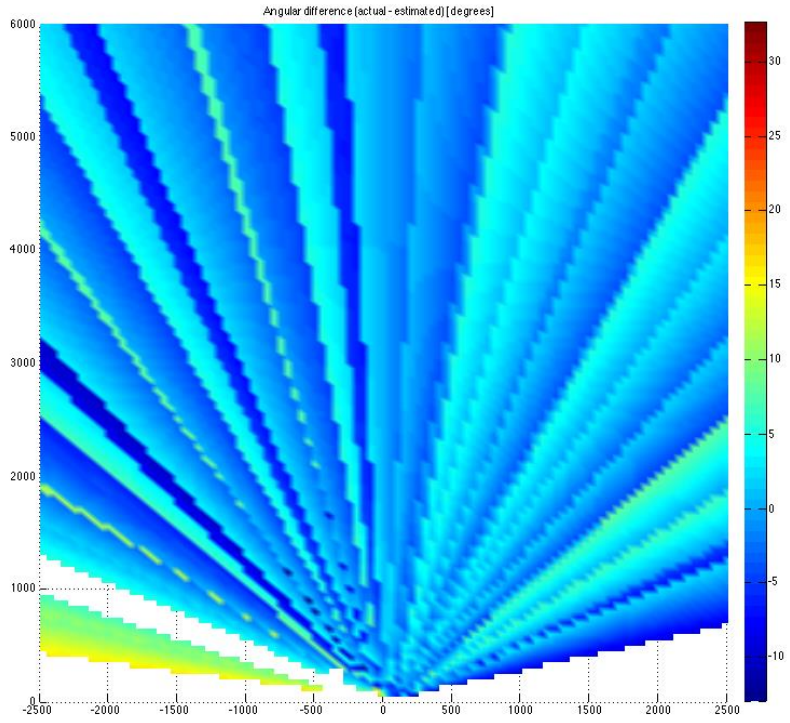

Fig. 9. At 16 kHz sampling rate, the system behaves well and typically estimates the angle correctly within about 5 degrees.

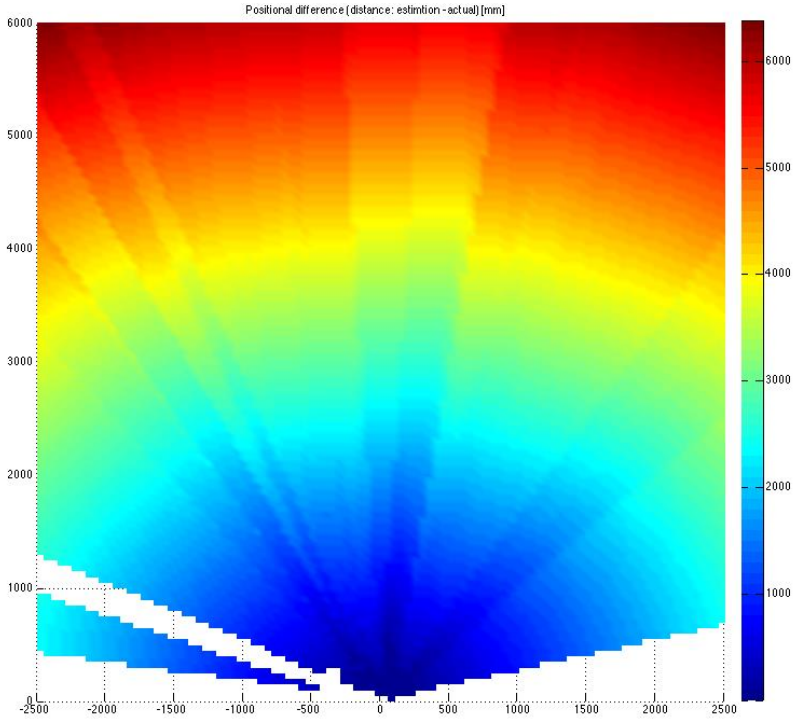

Fig 10. At 16 kHz sampling rate, the positional estimation deteriorates as the distance increases, and is only reasonably correct in the direct vicinity of the microphones.

4. Beamforming in a game with conflicting audio sources

To understand better the effect of beamforming, we will use the Kinect SDK to modify an existing game written for the Kinect, where the user interacts with the application through movement and speech. The game has potentially loud music continually playing in the background, which sometimes interferes with the voice commands. One possible mitigation of this problem is to have the microphone array set up as a directional microphone by continually pointing the beam to the player's head (by using the head's coordinates provided in the skeletal tracking stream).
The skeleton stream exposes the (x, y, z) coordinates of 20 skeletal joints detected by the depth sensor [20]. The (x, y, z) coordinates are expressed in meters, using the following coordinate system:

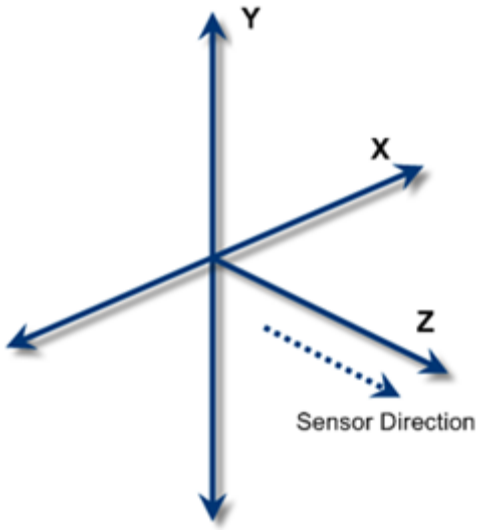

Fig. 11. Kinect skeletal stream coordinate system [9]

The Kinect sensor is positioned at (0, 0, 0), facing the user (z increases the farther away the user is from the Kinect). In order to be able to manually point the beam, we need to find the azimuth angle theta [Fig. 12] between the Kinect and the user.

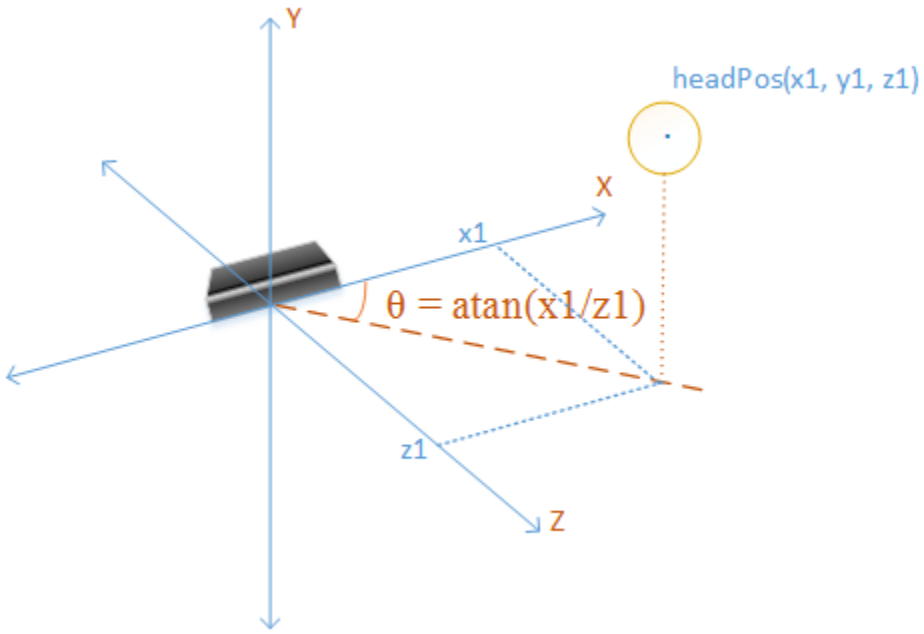

Fig. 12. Estimating the angle theta between the user and the Kinect sensor

This can be accomplished by using the following formula:

θ = atan(headPos.X / headPos.Z),

where headPos is the position of the head joint, given by the skeletal stream data. Then, this angle (converted to degrees) can be used to set the `AudioSource.ManualBeamAngle` property of

the Kinect Sensor (using the .Net Kinect SDK).
We did not perform a formal evaluation of the speech recognition gains garnished by this method, but having the microphone continually pointed to the user in an environment with several conflicting audio sources of approximately equal volume has an obvious benefit.

5. Discussion

We checked the viability of our method by simulating a sound source, the time delays of each microphone and then discretized those delays to simulate the behavior of the Kinect system. The simulation revealed some gaps in the solver, which we also found during live testing. For some angles, our algorithm returns NaN as the angle. This is likely due to some part of the solver not cleaning up a partial NaN result, which in Matlab, spreads like a plague to subsequently computed values.

More interesting though, are the results from the simulation at different sample rates of the microphones. The system (unsurprisingly) suffers greatly in accuracy if the sample rate becomes 4 kHz, which due to SDK issues early in the project, was the sample rate for some of our live tests.

6. Conclusion

We have shown that it is possible to perform TDOA on the Kinect, which is simple in comparison to Kinect's own algorithm for sound localization. Our method carries the benefit of being able to accurately estimate not only the direction of the sound, but also its location in space, if it is in the near vicinity of the Kinect.

We also demonstrated how one can manually point the Kinect's audio source beam to the active user in multimodal scenarios where the speech commands uttered by the user compete with other sounds native to the environment.

7. References

1. Microsoft Kinect, http://www.microsoft.com/en-us/kinectforwindows/, Accessed 2014-05-08
2. Natural User Interface for Kinect for Windows, http://msdn.microsoft.com/en-us/library/hh855352.aspx, Accessed 2014-05-20
3. WAVE file format specifications, http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/WAVE.html, Accessed 2014-05-20
4. Kinect microphone array image credit: http://msdn.microsoft.com/en-us/library/jj131026.aspx, Accessed 2014-05-20

5. Andrew Davidson, *Kinect Open Source Programming Secrets: Hacking the Kinect with OpenNI, NITE, and Java*, chapter 15. *Using the Kinect's Microphone Array*, McGraw-Hill/TAB Electronics; 1 edition (April 18, 2012)
6. Calmes, L. (2009). *Biologically Inspired Binaural Sound Source Localization and Tracking for Mobile Robots*. PhD thesis, RWTH Aachen University, Aachen, Germany.
7. Kinect for Windows Sensor Components and Specifications, http://msdn.microsoft.com/en-us/library/jj131033.aspx, Accessed 2014-06-05
8. Tashev, I. and Malvar, H.S. (2005) A new beamformer design algorithm for microphone arrays, Acoustics, Speech, and Signal Processing. Proceedings. (ICASSP '05). IEEE International Conference on, vol. 3, pp. 101-104.
9. Kinect coordinate spaces, http://msdn.microsoft.com/en-us/library/hh973078.aspx, Accessed 2014-06-05
10. Tracking users with Kinect skeletal tracking, http://msdn.microsoft.com/en-us/library/jj131025.aspx, Accessed 2014-06-05