Physical Arrangement Detection Technique for Networked Devices through Ambient-Sound Awareness

Final Report

Version 1.0

Team Pandaa

Final Report

for

Physical Arrangement Detection Technique for Networked Devices through Ambient-Sound Awareness

Prepared By

TEAM NAME:	Team Pandaa
TEAM MEMBERS:	Dilip Gudlur - dilip.gudlur@west.cmu.edu
	Divya Vavili - divya.vavili@west.cmu.edu
	Fabian Popa - fabian.popa@west.cmu.edu
	Qi Zheng - qi.zheng@west.cmu.edu

Approved By

CLIENT:	Trevor Pering – grafnu@gmail.com
ADVISOR:	Pei Zhang – peizhang@cmu.edu

SUBMISSION DATE: December 9, 2011

Table of Contents

List of Figures

Introduction

This document provides a high level overview of the evolving technical architecture for the PANDAA system. The document also provides in-depth insight into the design details for the system. The goals for the PANDAA system and also the constraints under which the system has been designed have also been discusses briefly in this document.

Purpose

The purpose of this document is to depict the different aspects of the system with respect to its architecture and its design. It is intended to capture and convey the significant architectural and design decisions which have been made on the system. It outlines the different architectural components of the system to provide a better understanding of the data and process flow through the system. This document provides a fundamental platform to allow further changes in the architecture of the system.

Scope

The scope of this document pertains to but is not limited to describing the architecture, design and data structures for the PANDAA system. The document provides technical details for the implementing and expanding the current work of PANDAA. This document can be used as a technical reference for the PANDAA system but may not provide a complete view of the system as it is out of scope of this document.

Definitions and Acronyms

Definitions

Localization	Process of detecting the absolute or relative location of a sensor in a network of other sensors
AudioFrame	Array of raw audio samples
ImpulseFrame	List of impulse amplitude/time-offsets
DistanceFrame	Pairwise distances between devices
DissimilarityFrame	One dimensional matrix of distances of the devices with each other
GeometryFrame	Two Dimensional matrix consisting of the x, y coordinates of all devices

Acronyms

PANDAA	Physical Arrangement Detection Technique for Networked Devices through Ambient Sound Awareness
MDS	Multi-Dimensional Scaling
TDOA	Time Distance of Arrival

Software and Hardware Platforms

The current system of PANDAA has been completely implemented in Java. The system is now platform independent and can be ported across a variety of platforms. The earlier system had a component of MATLAB, which constrained it to be run on software platforms consisting of necessarily of MATLAB software. The current system is more scalable and extensible across all software and hardware platforms.

Architectural Goals and Constraints

The PANDAA system has been thoroughly researched, prototyped and has been demonstrated to work with an accuracy of 0.17m. However, the implementation which existed previously was not platform-independent, which made it difficult to be used for commercial applications. This also constrained other aspects of the system like scalability.

Goals

Our goal would be make the current system platform-independent and demonstrate the accuracy of the new system with respect to the proven prototype. The project would enable us to the prospect of integrating PANDAA into products such as Google+, where sound location technology could augment social interaction/networking features with exciting new functionality.

Constraints

The current system has been tried and tested with devices like laptops but hasn't been tested with mobile devices even though the mobile devices were used for audio capture. So, currently, mobile devices running the processing algorithm have not been explored as an option and that remains as a constraint for the system. Though the algorithm with minor adjustments can be made up and running for the mobile devices as well.

Architecture Overview

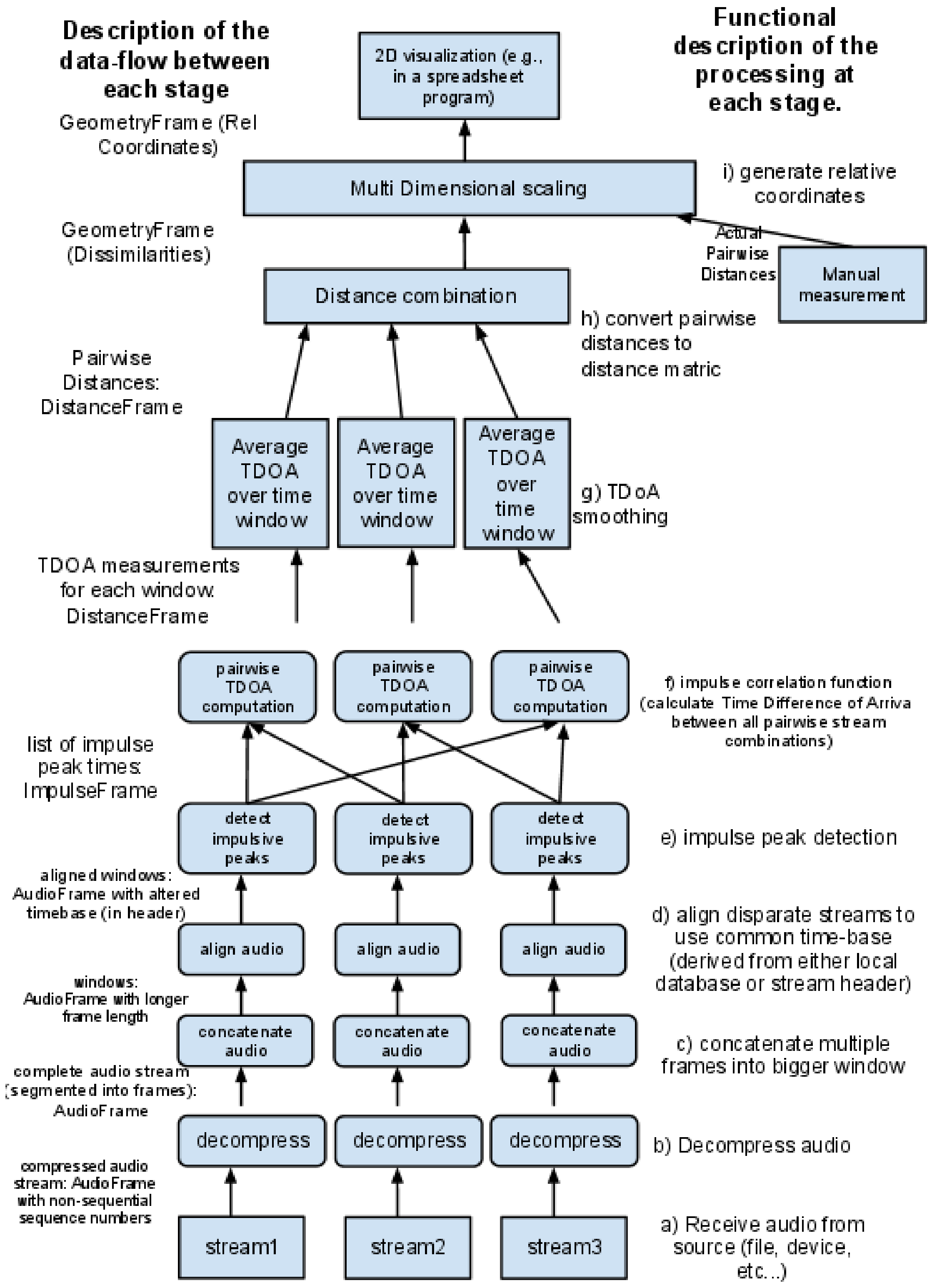
Description of the data-flow between each stage
Functional description of the processing at each stage.
2D visualization (e.g., in a spreadsheet program)
GeometryFrame (Rel Coordinates)
Multi Dimensional scaling
i) generate relative coordinates
GeometryFrame (Dissimilarities)
Actual Pairwise Distances
Manual measurement
Distance combination
h) convert pairwise distances to distance matric
Pairwise Distances: DistanceFrame
Average TDOA over time window
Average TDOA over time window
Average TDOA over time window
g) TDoA smoothing
TDOA measurements for each window: DistanceFrame
pairwise TDOA computation
pairwise TDOA computation
pairwise TDOA computation
f) impulse correlation function (calculate Time Difference of Arriva between all pairwise stream combinations)
list of impulse peak times: ImpulseFrame
detect impulsive peaks
detect impulsive peaks
detect impulsive peaks
e) impulse peak detection
aligned windows: AudioFrame with altered timebase (in header)
align audio
align audio
align audio
d) align disparate streams to use common time-base (derived from either local database or stream header)
windows: AudioFrame with longer frame length
concatenate audio
concatenate audio
concatenate audio
c) concatenate multiple frames into bigger window
complete audio stream (segmented into frames): AudioFrame
decompress
decompress
decompress
b) Decompress audio
compressed audio stream: AudioFrame with non-sequential sequence numbers
stream1
stream2
stream3
a) Receive audio from source (file, device, etc...)

Figure 1: Architecture overview of the system

Architecture Components

PANDAA is a step-wise algorithm. As a whole, it takes in the audio recordings of all the devices in vicinity and outputs their spatial arrangement relative to each other. It can do so by identifying those peaks in the individual waveforms that can be matched to the same sound event in the environment. The resulting time offsets between the same sound event occurring at separate devices (aka. TDOA, Time Difference of Arrival) is telling of the distance between them.

The algorithm is divided into "modules" (processing units), each one with a precise task (such as audio sync, impulse detection, and TDOA computation). They are self-contained and take in specific input and output data types called "frames". The accuracy of the system as a total is directly traceable to the accuracy of each individual module. That's because modules are chained, the output of one feeding into the input of the next.

A succession of modules is a "pipeline", and there are three types. The "single pipeline" runs all the modules specific to one device, namely the impulse detection from synchronized raw audio and the consolidation of several impulses into a larger batch. There is a single pipeline for each device. At the point of TDOA, the impulse batches from a pair of devices are passed on to a "dual pipeline". It performs TDOA computation and smoothing, thereby matching peaks occurring at the two devices to the same sound events. There is a dual pipeline for each pair of devices. Finally, the results of all dual pipelines are combined by the "merge pipeline". It gathers all pair-wise peak distances into a comprehensive distance matrix and performs multi-dimensional scaling to produce the final spatial mapping.

In between modules (and therefore in between pipelines), there are "frame streams". These are structures that either buffer frames in memory (to be further processed locally) or transport them over the network (to be further processed on the server). Frame streams also pair up two frames into a MultiFrame for the dual pipeline and, subsequently, for the merge pipeline.

Finally, there are "frames", the single units of data corresponding to the actual segments of audio. These are passed into the first module of the single pipeline, the impulse detection module, and make their way up the stream. They start out as instances of RawAudioFrame, then evolve into several forms of ImpulseFrame, DistanceFrame, and finally GeometryFrame, as they are processed in the chain modules.

Frames are generated to the specifications of their respective "header". There is a type of header for each type of frame (e.g., RawAudioHeader for RawAudioFrame) and it contains meta-information (for RawAudioHeader, it's sampling rate, number of channels, audio format, etc.) Headers are also used to initialize modules and pipelines.

Design

Class Diagrams

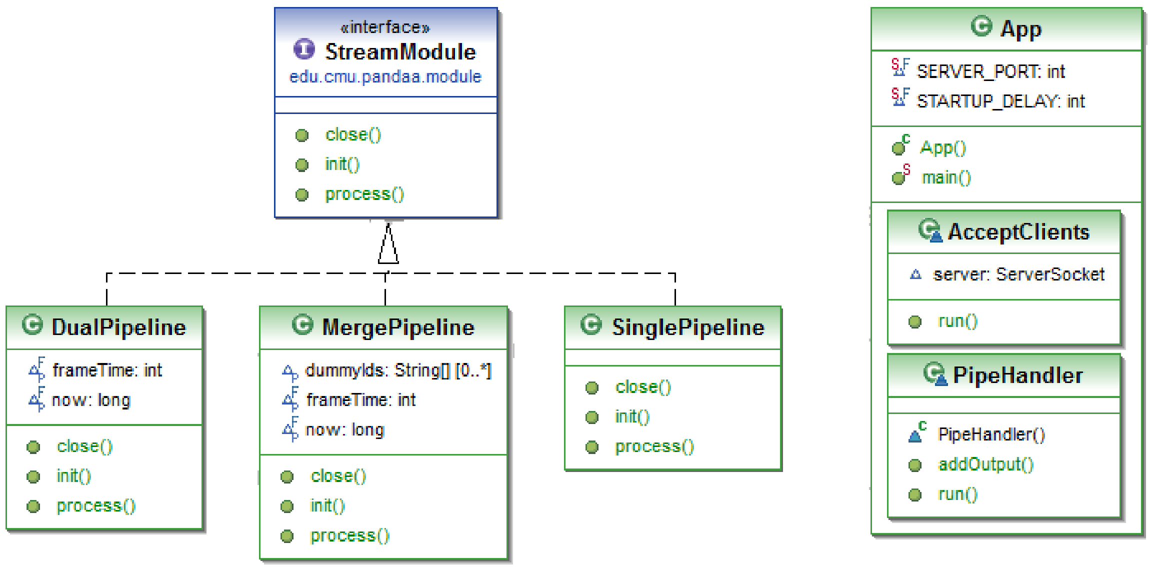

Figure 2: Framework class diagram

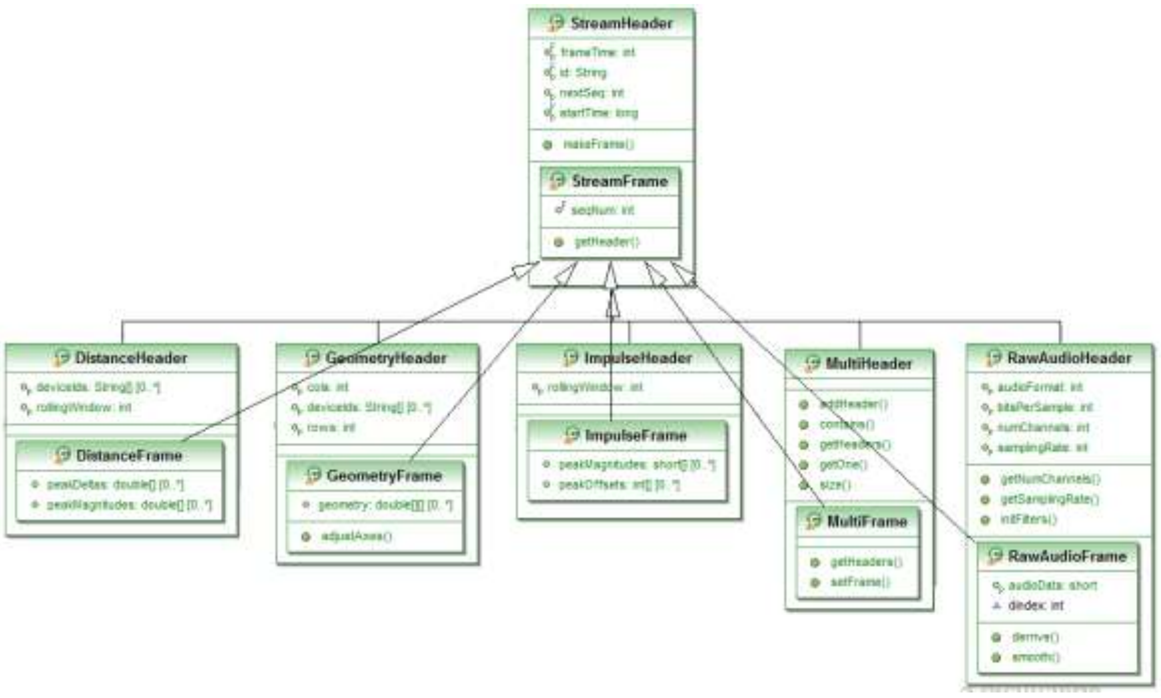

Figure 3: Header class diagram

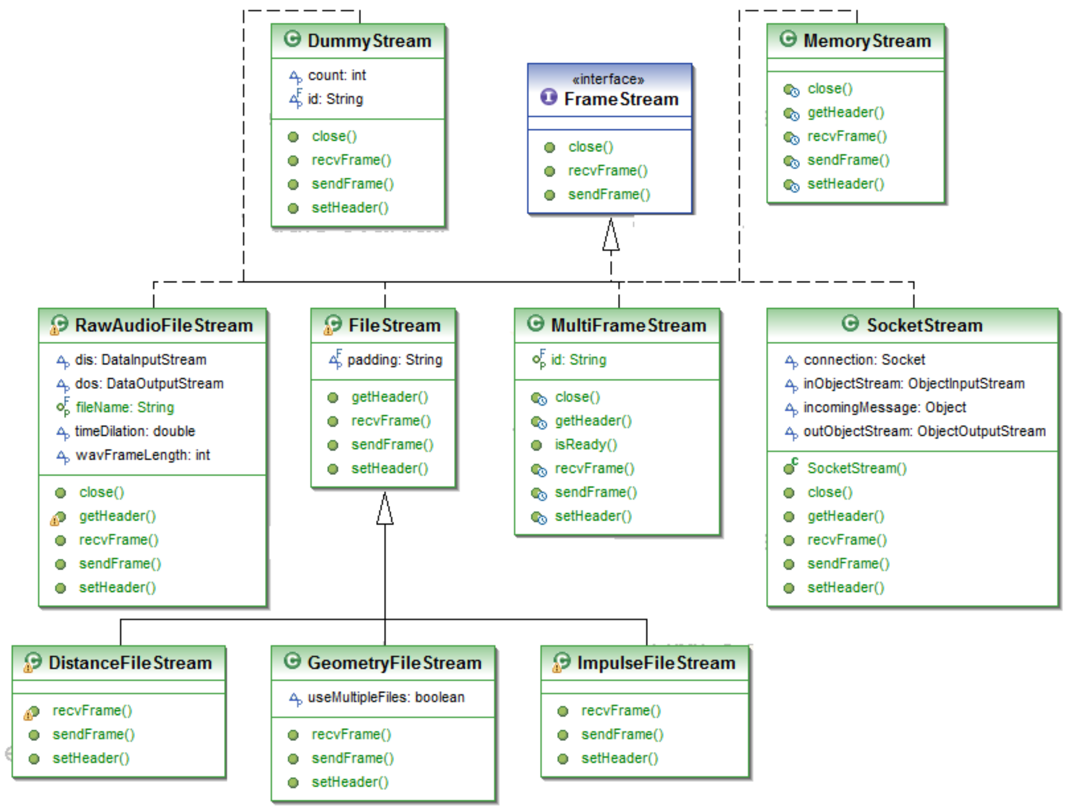

Figure 4: Stream class diagram

Modules

Synchronization

Input: Unsynchronized Raw Audio (.wav) file

Output: Synchronized Raw Audio (.wav) file

Module: AudioSynchronizationModule

Description: The purpose of this module is to synchronize the audio across all devices. It takes the audio files of the devices and then synchronizes it based on the file impulse detected in the file. This serves as the reference for all other impulses to be detected using further steps.

Ambient Sound Event (Impulse) Detection

Input: RawAudioHeader

Output: ImpulseHeader

Module: ImpulseStreamModule, DbImpulseStreamModule, FeatureStreamModule

Description: Ambient Sounds may vary significantly in signal-to-noise ratio (SNR). So we decide to detect impulsive sounds. Impulsive sounds are short duration sounds with relatively higher amplitude, such as human cough and claps. They are distinct and have high signal-to-noise ratios (SNR). Since one frame lasts 100ms, it is small possibility that two sound events starts in one frame. It is assumed that at most one peak is detected during a frame. In conclusion, the first peak of each impulsive sound is detected to represent the sound event. The root mean square (RMS), which represents the average power of the sound during a period of time, can be used to look for a sound event. The slow-window RMS, which is RMS of 1000 samples, is an indicator of current local loudness from the microphone. The fast-window RMS, the RMS of 50 samples, is looking for sound events that are louder than the noise floor. By comparing the slow-window and fast-window RMS, we can know when the sound event starts.

TDOA

Input: MultiHeader (composed of two ImpulseHeader)

Output: DistanceHeader

Module: TDOACorrelationModule, TDOACrossModule (alternative)

Description: The objective of TDOA (Time Difference of Arrival) is to match impulses in corresponding frames (synchronized time intervals) at two devices to the same originating sound event. Only then does the time difference between the peaks hold information about the distance between devices. Our implementation first identifies all plausible peak pairs. That means the time-wise closest pairs with a maximum delta of 29ms (the speed of sound over 10m, a reasonably sized room). All potential matches are then sorted on their time difference. Finally, in decreasing order, matches are validated if none of the two peaks have yet been matched. The final list of matches is a subset of the plausible ones and contains all the peak pairs that most plausibly describe the same event.

Distance

Input: DistanceHeader

Output: GeometryHeader

Module: DistanceMatrixModule

Description: The input pairwise distances from the TDOA Module are converted into a comprehensive pairwise distance matrix. This resultant matrix is a two dimensional matrix and represents the distances between each of the devices. For n devices, the TDOA module generates NC2 outputs(pairs). These pairs become the input for DistanceMatrixModule. The module parses through these inputs and generates a single output which contains the distances between all the pairs of devices available in the system.

Geometry

Input: GeometryHeader

Output: GeometryHeader

Module: GeometryMatrixModule

Description: The resultant matrix from the DistanceMatrixModule contained distances between each pair of the devices. The goal of the GeometryMatrix Module is to convert these distances into corresponding coordinates for the devices. The process of conversion from the distances to coordinates is called Multi Dimensional Scaling or MDS. This algorithm uses "similarities" or "dissimilarities" between objects and maps these into two dimensional coordinates. The GeometryMatrix Module uses an open source Java Library for MDS and converts the distance matrix into a coordinate matrix. This coordinate matrix actually consists of the 2D coordinates for each devices which can then be plotted on a graph to show the locations (relative) of these devices.

Conclusion

The PANDAA system is now platform independent. The system is robust and scalable and can be expanded upon for further architecture and design enhancements. The project has been a great learning experience for all the team members. The source code for this system is being released under Apache 2.0 license for further use. This documentation shall serve as a technical reference for future work and integration into a variety of applications.

References

[1] Zheng Sun, Aveek Purohit, Kaifei Chen, Shijia Pan, Trevor Pering, and Pei Zhang. "PANDAA: Physical Arrangement Detection of Networked Devices through Ambient-Sound Awareness" 13th International Conference on Ubiquitous Computing (Ubicomp 2011) (2011).
Available at: http://works.bepress.com/zhengs/7

[2] Zheng Sun, Aveek Purohit, Philippe De Wagter, Irina Brinster, Chorom Hamm, and Pei Zhang. "PANDAA: A Physical Arrangement Detection Technique for Networked Devices through Ambient-Sound Awareness" ACM SIGCOMM 2011 (2011).