Interaction Analytics of Software Factory Recordings

by

Daniel Pressler

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved October 2018 by the
Graduate Supervisory Committee:

Daniel W. Bliss, Chair
Visar Berisha
Steven Corman

ARIZONA STATE UNIVERSITY

December 2018

ABSTRACT

A human communications research project at Arizona State University aurally recorded the daily interactions of aware and consenting employees and their visiting clients at the Software Factory, a software engineering consulting team, over a three year period. The resulting dataset contains valuable insights on the communication networks that the participants formed however it is far too vast to be processed manually by researchers. In this work, digital signal processing techniques are employed to develop a software toolkit that can aid in estimating the observable networks contained in the Software Factory recordings. A four-step process is employed that starts with parsing available metadata to initially align the recordings followed by alignment estimation and correction. Once aligned, the recordings are processed for common signals that are detected across multiple participants' recordings which serve as a proxy for conversations. Lastly, visualization tools are developed to graphically encode the estimated similarity measures to efficiently convey the observable network relationships to assist in future human communications research.

TABLE OF CONTENTS

Page

LIST OF FIGURES iv

CHAPTER

1 INTRODUCTION 1

1.1 About the Software Factory 2

1.2 Daily Recordings 3

1.3 Recording File Formats 5

2 BACKGROUND 7

2.1 Perceived versus Actual Communication 7

2.2 Social-Network Estimation 8

2.3 Engineering Concepts 8

3 TOOLS AND PROCESS 10

3.1 Cross-Correlation 12

3.2 Time-Alignment 13

3.2.1 Check-in Logs 14

3.2.2 Incomplete Logs 15

3.3 Speech-Detection 17

3.4 Visualization 18

3.4.1 Clip Plots 19

3.4.2 Activity Maps 20

3.4.3 Spring Plots 24

4 ANALYSIS RESULTS 25

4.1 Correlation Normalization 25

4.2 Choosing an Appropriate Segment Length 27

CHAPTER Page

4.3 Alignment Estimation 31

4.4 Alignment Correction 36

4.5 Speech Detection Thresholds and Performance 39

4.6 Analysis of Networks through Force-Directed Graphs 44

4.7 Analysis Summary 49

5 CONCLUSION 51

5.1 Summary 51

5.2 Future Work 54

REFERENCES 56

LIST OF FIGURES

Figure Page

1. Histogram of Recording Durations. 4
2. Bar Graph of Total Recording Durations by User over the Lifespan of the Software Factory. 4
3. Number of Unique Users and Total Recording Durations by Week. 5
4. Diagram of the Process Used for Conversation Detection of the Software Factory Data. The Inputs to Each Step Are Annotated above the Steps, While the Outputs Are below. 11
5. Example Cross-Correlation Output between Two Recordings of a Single Conversation Showing a Peak around Time-Lag 0. 12
6. Example Cross-Correlation Output between Two Unrelated Recordings Showing No Discernible Peak. 13
7. An Example Processing Matrix Showing 10 Recordings from 6 Users Aligned according to the Check-In Log. 15
8. Histogram for the Mismatch between Log Durations and Recording Durations for the Weeks in Which the Check-In Log Was Fully Populated. 16
9. Example 36-Second Clip Plot. 20
10. Example 24-Second Clip Plot Where the Clips Were Originally out of Alignment. 20
11. Example Activity Map for a Day with Four Participants. 22
12. Example Activity Map Demonstrating an Identified Programming Pair (Users 2792 and 4091). 23
13. Spring Plot for the Day Shown in the Programming Pair Activity Map. 24

Figure Page

14.Example of Two Loud Signals Occurring Simultaneously but in Unrelated Conversations. 26

15.Example of Two Softer Signals; the Similar Shapes Present on Each Line Indicate that a Single Conversation Is Being Captured in Each Channel. 27

16.The Unscaled Cross-Correlations from Two Pairs of Signals: a Loud, Unmatched Pair, and a Quiet Matched Set. Note that Each Peak Is at a Similar Level. 27

17.The Cross-Correlations from Two Pairs of Signals after Normalization by the Product of Their RMS Levels. Note that the Matched Peak Is Clearly Distinguished from the Unmatched Pair. 28

18.A 5 Minute Example Where Two Participants Converse with a Brief Pause in between. 29

19.The Cross-Correlation of a Single 5-Minute Segment from the Segment Length Example. 29

20.The Cross-Correlations (Aligned by Segment Center) of Four 36-Second Segments from the Segment Length Example. 29

21.Example Output from the Refined Alignment Processing Step. The Similarity Measure (left) and Time-Lag (right) Are Used to Estimate the Time Offsets between Recordings to Refine the Alignment. 32

22.Refined Alignment Time Offset Output Represented with Time-Lag as a Function of Time. 33

23.Refined Alignment Time Offset Output Gated by a Similarity Measure Threshold. Many of the Noisy Estimates Are Removed However Outliers Still Remain. 33

Figure Page

24.Linear Fit for Refined Alignment Time-Lags from an Example Pair Illustrating Both Ordinary Least Squares and RANSAC. 35

25.Example of Skewed Alignment for Three Participants 37

26.Pairwise Alignment Correction 37

27.Floating Reference Alignment Correction 38

28.Correction Values over Time for Six Participants on 2004-08-19. 40

29.Refined Alignment Output Performed on the Aligned Dataset on 2004-08-19. 40

30.Histogram of RMS Values for All Segments in SF Dataset Plotted with the Thresholds Used for Alignment and Conversation Correlation. 41

31.Sparse Speech Scenario, with Two Participants' RMS Values over Time Showing the Threshold and Detector Decision. 42

32.Dense Speech Scenario, with Two Participants' RMS Values over Time Showing the Threshold and Detector Decision. 43

33.A Participant Pairs' RMS Values Plotted over the Course of a Day. Both Alignment and Conversation Thresholds Are Shown. 43

34.Spring Plots Showing Different Activity between Morning and Afternoon. . 45

35.Spring Plots for a Single Day Using Weights Integrated over 75 Minute Segments..... 45

36.Spring Plots for Each of Three Consecutive Days in July 2004. 46

37.Spring Plots Integrated over Three Consecutive Days in July 2004. 47

38.Example Activity Map for a Day with Eight Participants. 48

39.Snapshot Spring Plots of a Day with Eight Participants. 48

40.Spring Plot Formed from Conversation Weights Aggregated over a Nine-Month Period with Annotated Projects. 53

Chapter 1

INTRODUCTION

In the field of human communication, research has uncovered discrepancies in perceived communication networks, obtained via participant surveys, and observable communication networks as described by trained, outside viewers [1], [2]. This difference underscores the importance of gathering observable data for furthering our understanding of human communication networks, however the process for doing so can be difficult. While perceived networks are measured through self-reports, collecting observable data requires persistent monitoring of the entire network. The problem is exacerbated at scale; small networks may be observed for short periods of time by a single coder, but this becomes impossible as the size of the network grows and costly as the duration increases [3].

One solution is to record all members of a network. Prior studies have used existing network infrastructure to gather indirect metadata such as email headers [4] and cell tower location data [5]. A more direct method was employed in a study at Arizona State University where from 2002 to 2005, employees and customers of a software development and consulting team voluntarily wore constantly-recording microphones. In collecting the network participation data, the problem transforms from data gathering to interpretation and analysis. The raw recordings must be processed to distill the participants' interactions into a usable form so that the observable communication network can be estimated. In this thesis, we aim to apply digital signal processing techniques in the development of a software framework that can be used to analyze the software team recordings. Extracting this information is an essen-

tial step towards characterizing observable communication networks and providing a quantitative counterpart to the perceived networks described by participant surveys.

1.1 About the Software Factory

The Software Factory (SF) was a software engineering consulting team run at Arizona State University from 2002 to 2005. The organization was managed by an experienced software engineer and comprised primarily of student-employees. SF was established for two main goals: first, the team developed software solutions for departments, research groups, and start-ups throughout the university. Over the three-year span, the team completed 33 projects. Roughly one-fifth of these were for internal use, including the group's website and a system for tracking the time and effort expenditures of SF employees. The other projects serviced a variety of external customers - usually other ASU research groups - and varied in size and scope, from a year-long effort to develop programs for entering biochemistry field data to the two-week development of a database for storing applied nano-bioscience patents.

The second motivation for opening the Software Factory was to create a controlled environment for the study and observation of human communication networks. While in the office, all SF employees and clients knowingly and voluntarily participated in a data-gathering research study during work hours. The data was collected using a variety of methods. Self-reports were obtained via periodic surveys probing an individual's effort levels, perceived communication networks, and general feelings. This data was augmented by detailed interviews that were conducted roughly every two weeks and the contents of a metrics database where employees codified the type of work they performed on a given day and logged work hours. To

gather independent observation data, a trained, non-participant observer sporadically took field notes and a webcam was set up to intermittently take pictures to provide visual context. The team's Visual SourceSafe repository also tracked changes on most of the SF codebase, providing quantitative data on the magnitude of changes made to specific projects by programmer and date.

By far, the most persistent independent observation data takes the form of daily recordings captured by individual microphones worn on the person of each employee and client while inside the facility. Upon arrival, a participant would clock-in then don a microphone while a program recorded the participant ID and check-out time. Upon departure, the process was repeated in reverse, creating individual recordings for each participants' visit along with a time log containing coarse time-stamps for each recording's start and end. The near-ubiquitous coverage of this recording dataset affords the unique opportunity to gain insight into the observable networks over a very large time period. The focus of this thesis is in using these daily recordings to estimate the Software Factory's communication networks and in visualizing the extracted results in intuitive ways to aid in their analysis.

1.2 Daily Recordings

The Software Factory dataset contains 3915 files from 57 unique participants totaling over 405 days of recorded time. The recordings vary in length from a few minutes to over nine hours. The histogram of daily recording durations is shown in Figure 1. Recording periods ranging from 20 minutes to 6 hours are most common while durations over six hours are relatively rare.

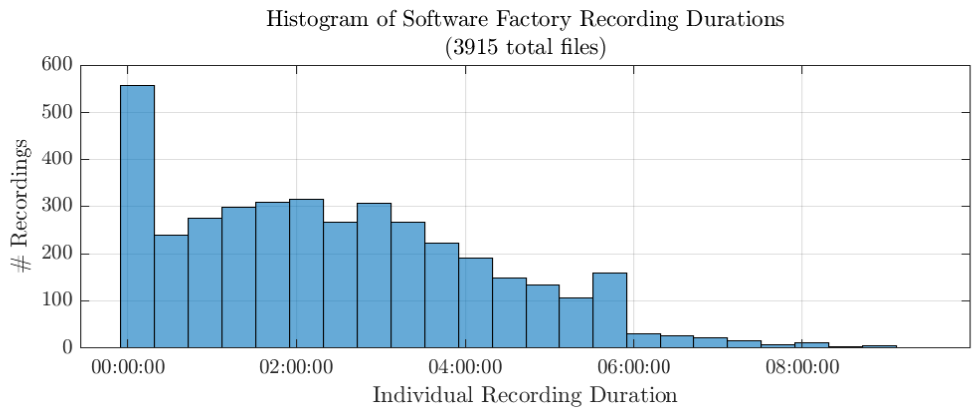

Figure 1. Histogram of recording durations.

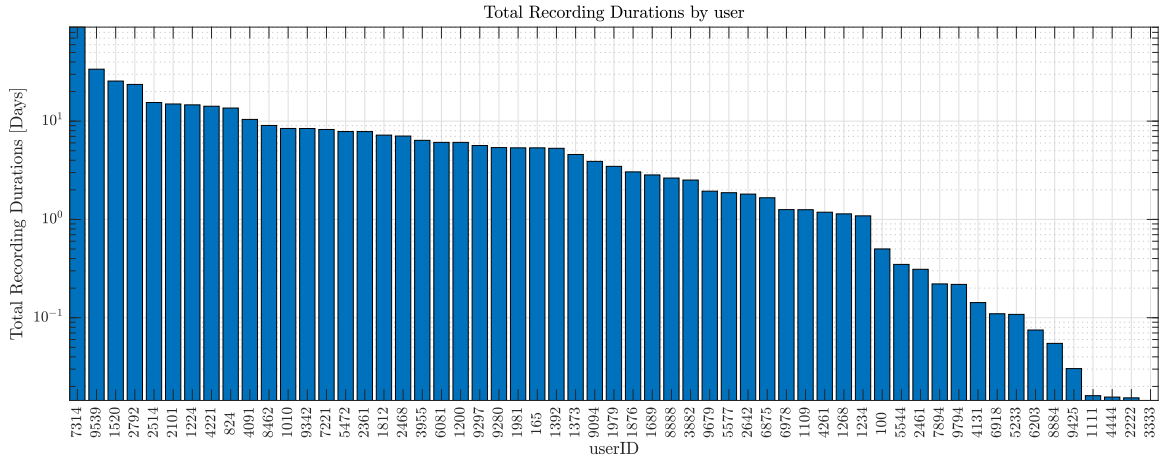

Figure 2. Bar graph of total recording durations by user over the lifespan of the Software Factory.

Figure 2 shows the total recording durations for each user. The data ranges from the SF director (7314) who clocked in 90.5 cumulative days to 20 minute experimental users (1111-4444). The midrange is dominated by programmers while the low-end consists of programmers, researchers, and clients.

The dataset is organized into 130 weekly folders with each week consisting of an average of 74 hours of recordings from 8 unique participants. The actual number of unique users and recording durations per weekly folder are shown in Figure 3.

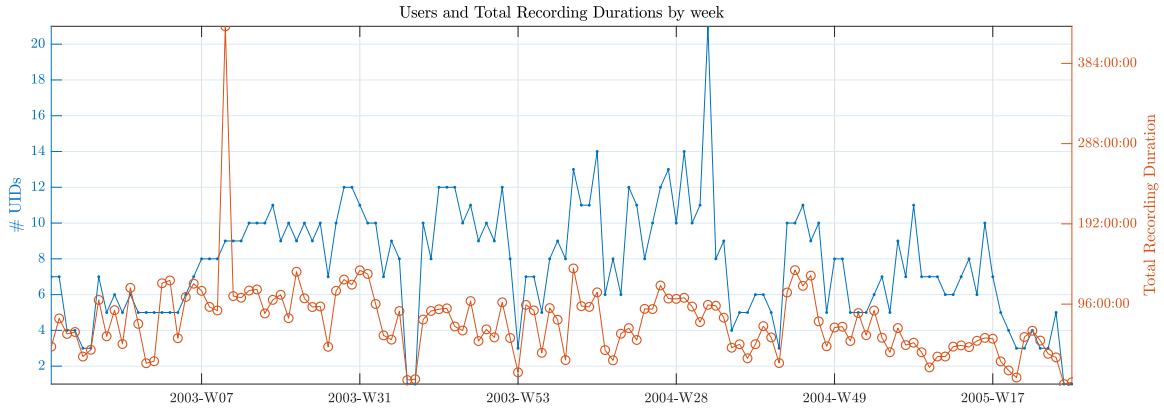

Figure 3. Number of unique users and total recording durations by week.

1.3 Recording File Formats

The interviews and daily recordings were captured into Digital Speech Standard (DSS) files. DSS files are highly-compressed and optimized for voice dictation. The format was designed to accommodate standard recording features - rewind, playback, overwrite - while minimizing file size. As a result, the quality of the recordings varies and is often rather poor. The quality is somewhat counterbalanced by the speakers' proximity: microphones were worn on the person of each participant and were usually able to record moderately strong signals. This too posed challenges, as the noise level and signal level is inconsistent across the dataset. This could be attributed to a number of factors including varying microphone position, low recorder battery, slightly obstructed microphones, and device-to-device variation. Combining these factors, the dataset is characterized by high background noise levels and low-quality but comprehensible speech.

DSS files cannot be processed natively in MATLAB, so format conversion was necessary to process the data. The major built-in functions handle either compressed MP3 or uncompressed WAV formats. Despite their larger size, WAV files were se-

lected due to the potential for time-ambiguity present in the underlying MP3 codec framework MATLAB uses [6]. The Switch Sound Converter program was used to perform the conversion and due to the poor initial quality, the smallest WAV encoding was used. The converted files used for processing were stored as 6 kHz, 8-bit, mono-channel pulse-code modulated WAV files.

Chapter 2

BACKGROUND

The primary goal of this thesis is to extract estimated observable communication networks from a dataset of contextualized recordings collected from the Software Factory. The motivation for this work is rooted in two social science concepts: differences between perceived versus observable behaviors and social network estimation. In order to achieve this goal, this thesis applies several common engineering techniques in order to detect voice activity, identify correspondences, and visualize the estimated networks. This chapter discusses each of these related background topics and places this thesis in their context.

2.1 Perceived versus Actual Communication

Social science research has uncovered discrepancies in perceived versus actual communication [2]. This finding calls into question the idea of relying solely on informants' recollections as the basis for conclusions on social structure and has kicked off an area of research into why these discrepancies exist and how to account for them in anthropological studies [7]. Further work aims to develop a model that links the perceived and actual networks [1]. This concept motivated the formation of the Software Factory experiment and led to the collection of the participants' recordings that are used in this thesis. During the Software Factory experiment, perceived data was collected in the form of surveys and interviews. This thesis aims to estimate the observable communication networks from the participant recordings

as a counterpart to the perceived data; it does not explore the relationship between the two.

2.2 Social-Network Estimation

Social-network estimation is another robust field of research and has wide-ranging applications in social sciences, telecommunications, advertising, cyber security, etc. This field encapsulates the idea of describing or modeling the interactions between individuals in a group. From these networks, further work in this field relies on exploring their dynamics in order to make predictions or explain phenomena [8]. Some studies have used existing network infrastructure to gather indirect metadata such as email headers [4] and cell tower location data [5]. This thesis aims to use the communications themselves, in the form of individual participants' recordings, in order to estimate the social-networks formed in the Software Factory. In addition, visualization tools are used in this thesis to efficiently depict time-dependent network dynamics.

2.3 Engineering Concepts

This thesis does not introduce any novel processing, but instead employs techniques from a variety of engineering fields in order to extract the estimated observable communication networks from the raw recordings and visualize them in ways that facilitate further social science research. Voice-activity detection (VAD) is used in selecting segments of the recordings as candidates for alignment estimation and conversation correlation. VAD is a common step in digital signal processing applica-

tions like digital communication and voice recognition and as such is widely covered in literature. The detectors vary widely based on the application; some are based on statistical models [9] and others use spectral estimators [10]. Different algorithms are used depending on if the processing can be done offline with training data [11] or must be performed real-time in streaming applications [12]. In this thesis, a simple energy detector proves sufficient to select candidate processing segments. From mathematics, graph theory is a natural complement to the study of communication networks. Graph theory encodes pairwise relationships in mathematical structures that can be used to gain insight into the underlying data. Again, this thesis doesn't advance the state of the art, but instead invokes this field to help collect the interaction network data into useful metrics and visualize it using force-directed graphs.

Chapter 3

TOOLS AND PROCESS

Verbal conversations must be identified to estimate observable communication networks. With individual recordings, this can be achieved by detecting the same conversation in the recordings of all its participants. Cross-correlation, as a measure of similarity between signals, is an obvious choice and is the fundamental mechanism used. Due to the size and complexity of the data set, however, additional processing is required to make the computation tractable and suppress false alarms. To shorten the computation time, time-alignment is performed to compare only time-local segments across participants. Speech detection is employed to suppress false alarms caused by the presence of correlated background noise. Once the verbal conversations are recorded, visualization is an important last step in making the processing results useful. Three different styles of graphs are used to illustrate different stages in the analysis process and are essential tools in cross-checking the results and in distilling network relationships into a comprehensible form.

The process used for conversation detection of the Software Factory data is depicted in Figure 4. In the diagram, each stage's primary inputs and outputs can be found above and below the step respectively. Before correlation is performed, the first step is to construct an initial processing timeline by parsing the log files and using the recordings' metadata. Once the clips are coarsely aligned, further alignment is achieved through an initial correlation processing step. Here, the recorded data itself is used to estimate the differential delays between recording files, refining the processing timeline. Once aligned, conversation correlation is performed in a man-

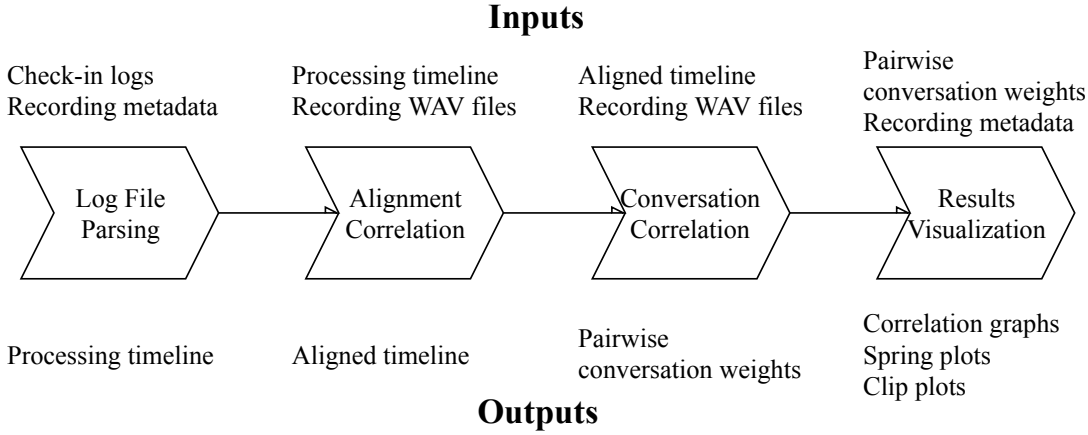

Figure 4. Diagram of the process used for conversation detection of the Software Factory data. The inputs to each step are annotated above the steps, while the outputs are below.

ner very similar to alignment correlation, creating pairwise conversation weights as a function of time for each participant pair. Lastly, the final conversation weights are combined with the recording metadata to create graphs which aid in visualizing the results.

Due to the repetitive nature of the process, this section is organized by technique and feature rather than walking through the procedure from beginning to end. The first section discusses how cross-correlation was used in both aligning and detecting conversations. The next section details how time-alignment of the recordings is achieved, discussing check-in logs, incomplete records, and drift compensation. Following that is an overview of the speech-detection techniques used and why they were necessary. The final section details and demonstrates the resulting graphs and explains their significance.

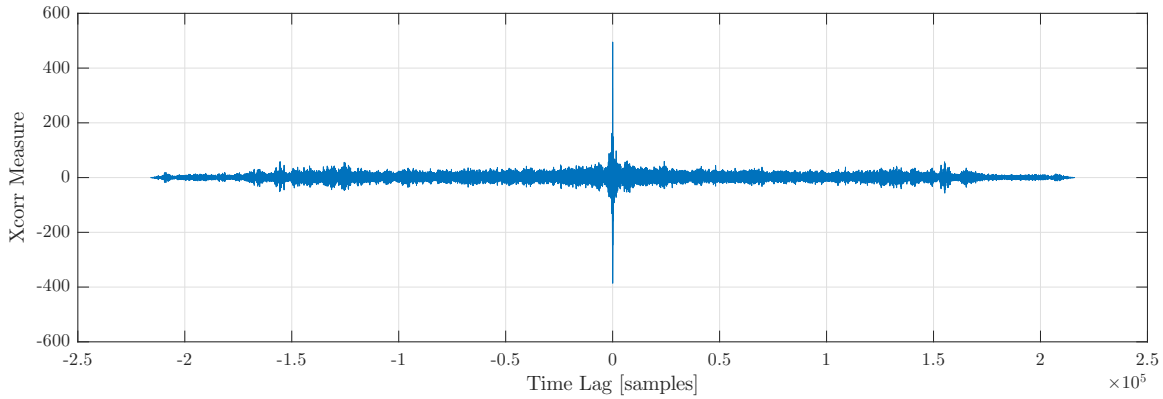

Figure 5. Example cross-correlation output between two recordings of a single conversation showing a peak around time-lag 0.

3.1 Cross-Correlation

Cross-correlation is the fundamental mechanism used to identify the same conversation in two or more recordings. The cross-correlation of two signals returns a similarity measure as a function of the time lags at which it is calculated. Peaks form in the cross-correlation output when the signals are maximally similar at a given time lag. In the Software Factory recordings as in typical speech, the conversations are generally nonrepetitive, so when they occur a single peak is expected in the cross-correlation output as shown in Figure 5. For two uncorrelated signals, the peaks and troughs tend to cancel leaving the formation of a singular peak in the cross-correlation highly unlikely. The cross-correlation output between two unrelated recordings is shown in Figure 6. The presence of cross-correlation peaks is used to determine that the same signal is present in multiple recordings, while the level of the peak itself is used to discriminate conversations from background noise and to quantify the extent of the interaction. The time-lag at which the peaks occur is a key part of time aligning the various recordings and is discussed below in Section 3.2.

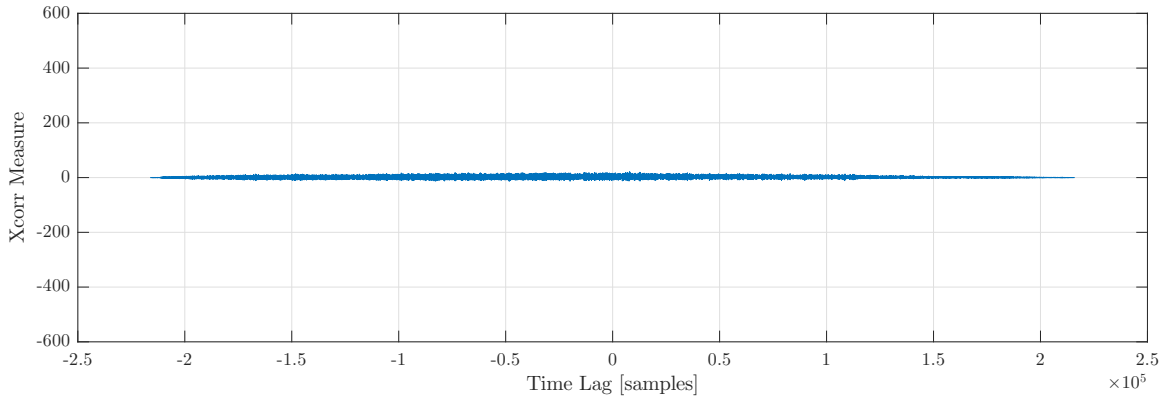

Figure 6. Example cross-correlation output between two unrelated recordings showing no discernible peak.

3.2 Time-Alignment

In order to detect conversations in the cross-correlation functions, the time segments must be time-aligned. The clips must be aligned to within at least one correlation segment length; the better the alignment, the greater the chance that the conversation will be successfully detected. Time-alignment also plays a role in reducing the computation time. When the signals are properly aligned, the processing can be limited to only the recording times during which multiple participants are present. The first level of alignment is performed using the check-in logs. If the logs were consistently populated and the recorders calibrated daily and synchronous, this is the only step that would be necessary. Fifteen percent of the check-in logs are incomplete though and therefore require additional processing. In addition to the missing logs, the recorders lack a common reference that keeps them synchronized. When combined with the sampling loss imparted by the lossy compression and subsequent transcoding, the effect is that even recordings that start out aligned will drift apart over time. Each recorder is unique such that each recording's drift relative to the others must be estimated and compensated for independently.

3.2.1 Check-in Logs

Check-in logs are tab-delimited text files created by the program the Software Factory used to track participant arrivals and departures. A log was created weekly and when properly populated contain:

Direction whether an entry is for a participant coming or going

UID numeric unique participant identifier

Date year, month, day of the log

Time At least HH:MM:SS; sometimes contains milliseconds

Filename The name of the DSS file created at check-in

An example snippet is shown below.

```
Check-In    9539    11/7/2002    11:00:02:365    9539_021107_rec_01.wav
Check-In    7314    11/7/2002    11:15:46:322    7314_021107_rec_01.wav
Check-In    9794    11/7/2002    11:33:49:129    9794_021107_rec_01.wav
Check-Out   7314    11/7/2002    11:37:24
```

These logs are crucial in setting the coarse time relationships between the recordings. For each day, the initial analysis timeline is created spanning from the earliest check-in to the latest check-out. Each recording is placed in the timeline according to its check-in time, generating a coarsely-aligned dataset, like the one shown in Figure 7, that is further processed for conversation detection and refined alignment. Alignment errors introduced during this first step are difficult to ascertain as they can only be measured during detected conversations which typically occur relatively late in the recordings. Anecdotally, initial alignment tended to be off on the order

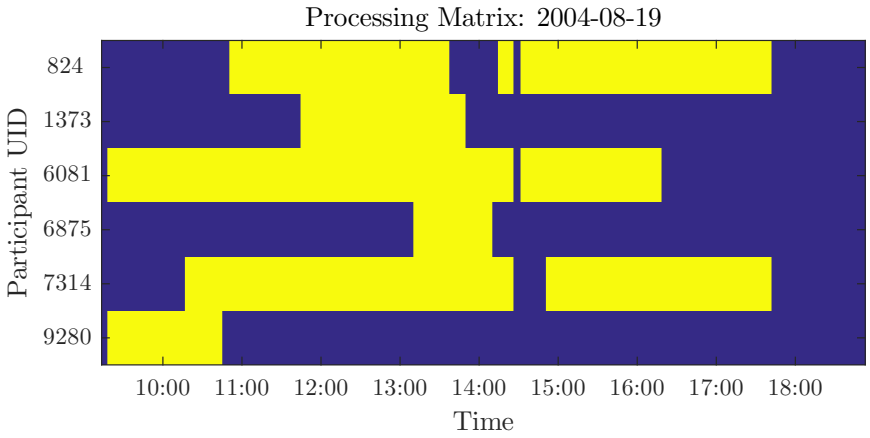

Figure 7. An example processing matrix showing 10 recordings from 6 users aligned according to the check-in log.

of seconds to minutes. It is easy to imagine these errors created with a human in the loop turning on recording devices after check-in.

3.2.2 Incomplete Logs

Twenty of the 130 weekly logs were incomplete and required additional processing to create the coarse timelines used in interaction detection. The errors took two forms: missing filenames, and missing both check-in times and filenames. Under both conditions, the mapping from the list of files in a given folder to a timeline is lost and must be reconstructed. Fortunately, the recording files were named by UID, date, and sequence(e.g. if a user had multiple recordings in a day, they were numbered in chronological order). The filename information can be parsed and cross-checked against the information in the log to restore the mapping. When the number of recordings per user per day is consistent across both the logfile and the file list, the reconstruction is trivial. However, in twelve of the twenty incomplete logs, there was mismatch in these metrics; almost all due to excess log entries. In look-

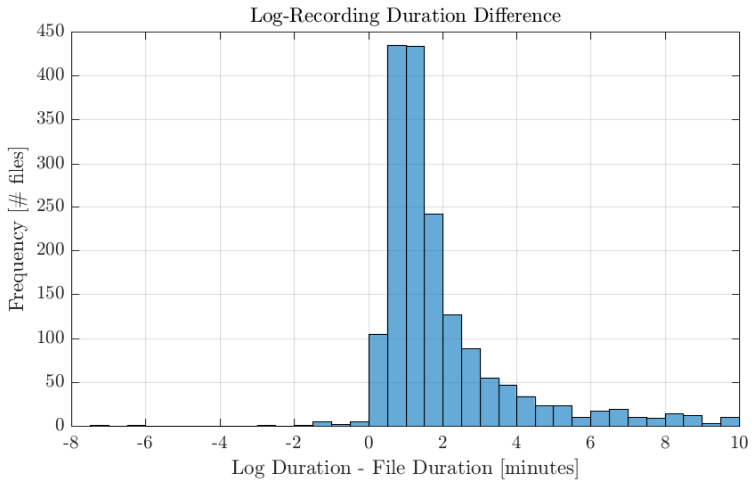

Figure 8. Histogram for the mismatch between log durations and recording durations for the weeks in which the check-in log was fully populated.

ing through the dataset, these are likely the result of brief visitations during which recorders weren't worn. Under these circumstances, a simple sorting algorithm was used to iterate through the metrics to match likely pairs and discard the residual entries. The most likely pairs are chosen based on the similarity of possible recorder durations bounded by the log entries and the actual file lengths.

When the check-in times are also missing, in addition to pairing the files to the log entries, the start times themselves must be reconstructed. A convenient estimate is to subtract the recording's duration from the log file's corresponding check-out time. This introduces an additional error term in the initial alignment as the Software Factory's tracking process did not enforce a constant relationship between the check-out times and the end of the recording. A histogram for the mismatch in between log-file durations and recording durations is shown in Figure 8. The results suggest that most of the check-outs occurred up to five minutes after the recording stopped; however there is a long tail that extends past 10 minutes. Assuming the participants' tracking process was similar during the weeks with incomplete logs, this

provides a good estimate for the initial alignment error we can expect in the timeline reconstruction and demonstrates the rationale for a five-minute correlation segment, as discussed in Section 4.2.

3.3 Speech-Detection

Speech detection is used in both alignment correlation and conversation correlation to lower the false alarm rate and reduce processing time. These effects are both predicated on discriminating recording segments that contain speech from those that don't. By attempting to send only segment pairs where both participants are active, the likelihood of a false positive goes down with the number of trials that contain the false hypothesis. The reduction in processed segment pairs also greatly reduces processing time by only expending resources on segments that likely contain matches.

As a common step in digital signal processing applications, detectors for speech vary widely based on their application and design goals. For the Software Factory conversation detection problem,

- the dataset is large, with roughly 500 days of recording runtime
- the data is processed offline (not realtime)
- the original recordings were highly compressed
- the transcoding is low bitrate

After weighing these traits and trying a few schemes, a simple RMS detector was used to perform speech detection for the Software Factory data. The same detector was used in both correlation steps, though a different threshold was used which changes the operating point between stages. The primary motivation for selecting such a simple detector was processing time. The detections are performed before

both of the cross-correlation steps and must process the entire data set twice, so processing time is an important metric to minimize to keep the analysis tractable. Almost as important were the performance considerations. The dataset is heavily compressed and transcoded to a low bitrate. As a result, some of the VADs that used spectral components seemed to achieve similar performance to their simpler counterparts and took far longer to execute. Lastly, the penalty for a false alarm is low. The speech detector is just an intermediate step that gates which recording segments are further processed. False alarms are passed through the correlator, and if they are true false alarms, the independent noise will create a low similarity measure that will be ignored during conversation detection.

3.4 Visualization

With such a large dataset, finding effective ways to visualize and explore the results is almost as important as the processing itself. Distilling the data from various processing steps into useful graphs provides a powerful mechanism for researchers to efficiently explore the observable communication networks uncovered by the analysis. Surfacing intermediate data can help users judge the fidelity of the estimated networks and gives a way to hear the original recordings and extract attributes that aren't captured in conversation detection, such as conversation content and participants' vocal inflections. In this project, we created three primary graphic types:

1. Clip Plots
2. Activity Maps
3. Spring Plots

Clip plots show time-aligned segments of the recordings from two participants

on a time series plot. Activity maps give an overview of a days' worth of conversation detections and highlight the times of the day that multiple users were in the Software Factory. Spring plots use the extracted conversation weights to display the estimated observable communication networks as a network graph.

3.4.1 Clip Plots

Clip plots, like the one shown in Figure 9, display segments from two participants' recordings on a single axes, allowing a user to see both signals as a function of time, listen to the recordings, and adjust their alignment. The scrollbar on the bottom can adjust the time alignment between the two channels, allowing a user to make corrections on the fly as seen in Figure 10. This clip plot shows two segments before alignment correlation where the two recordings were originally out of alignment. The upper track was shifted 3.4 seconds relative to the lower one to match the signal features. Although you can often tell visually when two clips are aligned, clip plots also allow stereo playback where each recording is sent to a different stereo channel. This provides aural confirmation of both alignment and conversation detection; an important feature that can be used to establish ground truth. The audio playback can also be used to extract what is being said, vocal patterns and characteristics, speaker directionality, and other important features.

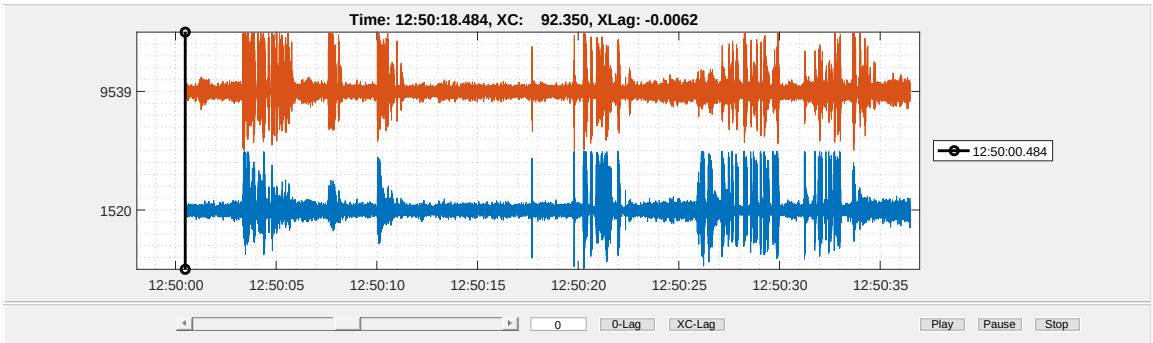

Figure 9. Example 36-second clip plot.

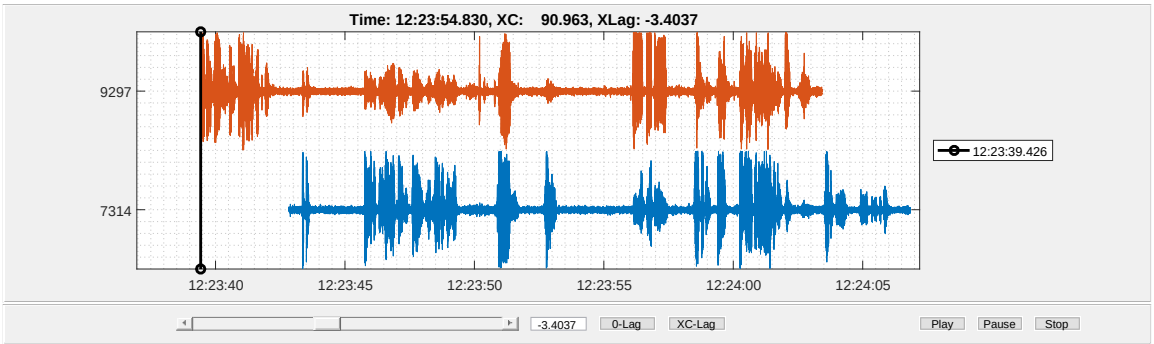

Figure 10. Example 24-second clip plot where the clips were originally out of alignment.

3.4.2 Activity Maps

Activity maps are visual aids for depicting the conversation activity for all participant-pairs on a given day. Activity maps are important because they can efficiently depict daily network traffic as a function of time, and can be used to quickly gauge the time of day, duration, and participant IDs for observable conversations. The similarity measure array from conversation correlation is the backdrop for activity maps; which are then overlaid with track outlines that highlight the segments of time throughout the day where each user has a recording. The track outlines are effective means for conveying the times during which multiple participants are recording and observable communication is possible. The outlines can be used to

discriminate an occasion where two people talk sparsely throughout a long day from a short visit that spans an entire recording session. By default, all 2-combinations of user pairs are displayed; activity maps may optionally be filtered by a specific user. This focuses the graphs on the activity of a single participant and can make it easier to spot relationships and trends.

An example activity map with four participants from August 2003 is shown in Figure 11. The horizontal axes is time and spans from the earliest recorded time (12:00 from participant 9539) to the latest (22:00 from participant 9342). The vertical axes is partitioned into units that represent a pair of participants. The track outlines are the colored lines that border the active regions in blue and create recording gaps filled in gray. During the times in which both participants are active, the similarity measure estimated during conversation correlation is colored in to its corresponding time segment. The range of observed similarities measures is shown in the colorbar on top.

A clear depiction of the participants' comings and goings and the estimated communication network can be read directly from an activity map. In this example, participant 9539 shows up to the Software Factory first, to be joined by 1812 and 9342 about an hour later. Upon their arrival, the three can be heard working together for about an hour before 9539 departs around 14:15. The remaining pair of 1812 and 9342 show very little detected conversation activity until user 1520 shows up and the trio interact for the next hour. User 1812 leaves shortly before 17:00 at which point users 1520 and 9342 are detected to have strong conversation activity. That is the last conversation activity observed during this day; user 9342 leaves to come back alone three hours later, while user 1520 works alone until 19:00.

By painting a picture of daily activity, these maps can help focus research efforts

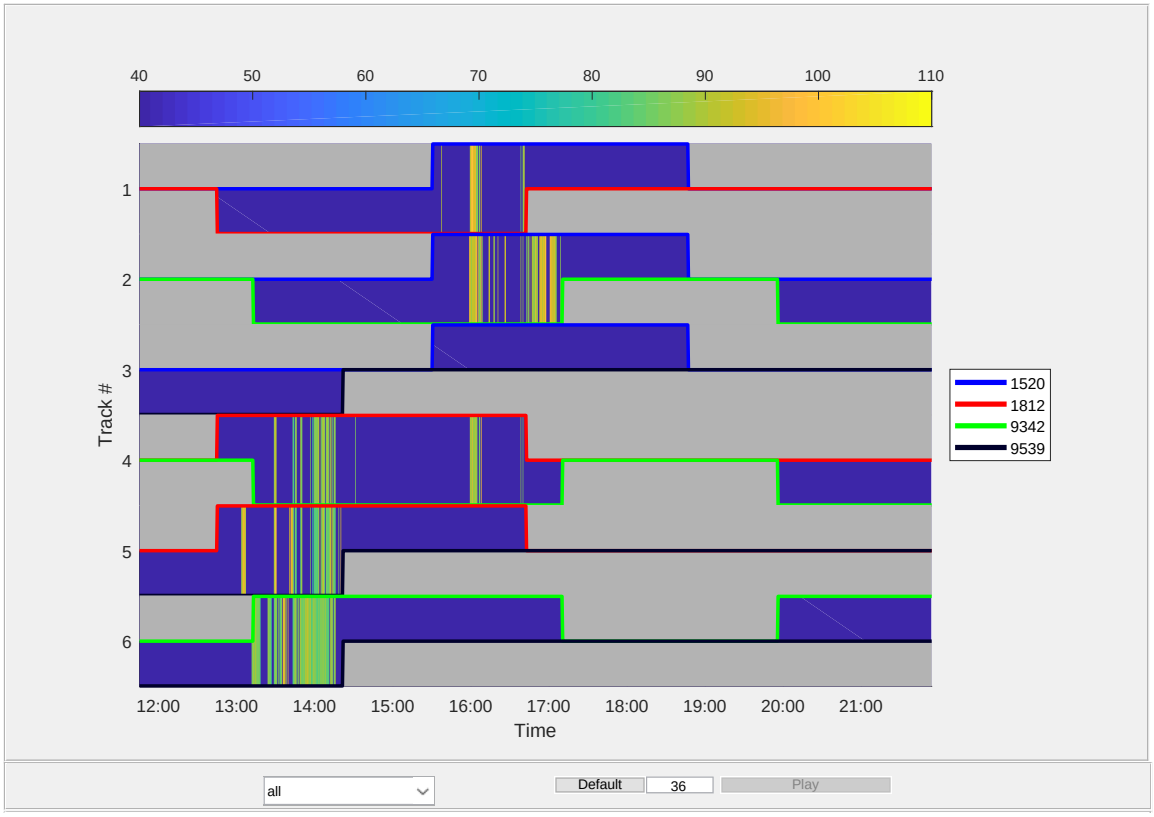

Figure 11. Example activity map for a day with four participants.

by illuminating clips, users, and times that may be of particular interest in building a ground truth of the dataset. This is of particular importance for such a large dataset with minimal metadata. Potential meetings and conversations are easily spotted on an activity map and can be used to decide which segments of the audio to individually analyze. The activity maps also excel at depicting network activity over time, facilitating in the spotting of time-dependent networks such as a group of colleagues that meet regularly in the morning.

Activity maps can also uncover the programming pairs that the Software Factory used as the formal process for much of its software development. In pairwise programming, two users share a common workspace and constantly discuss work

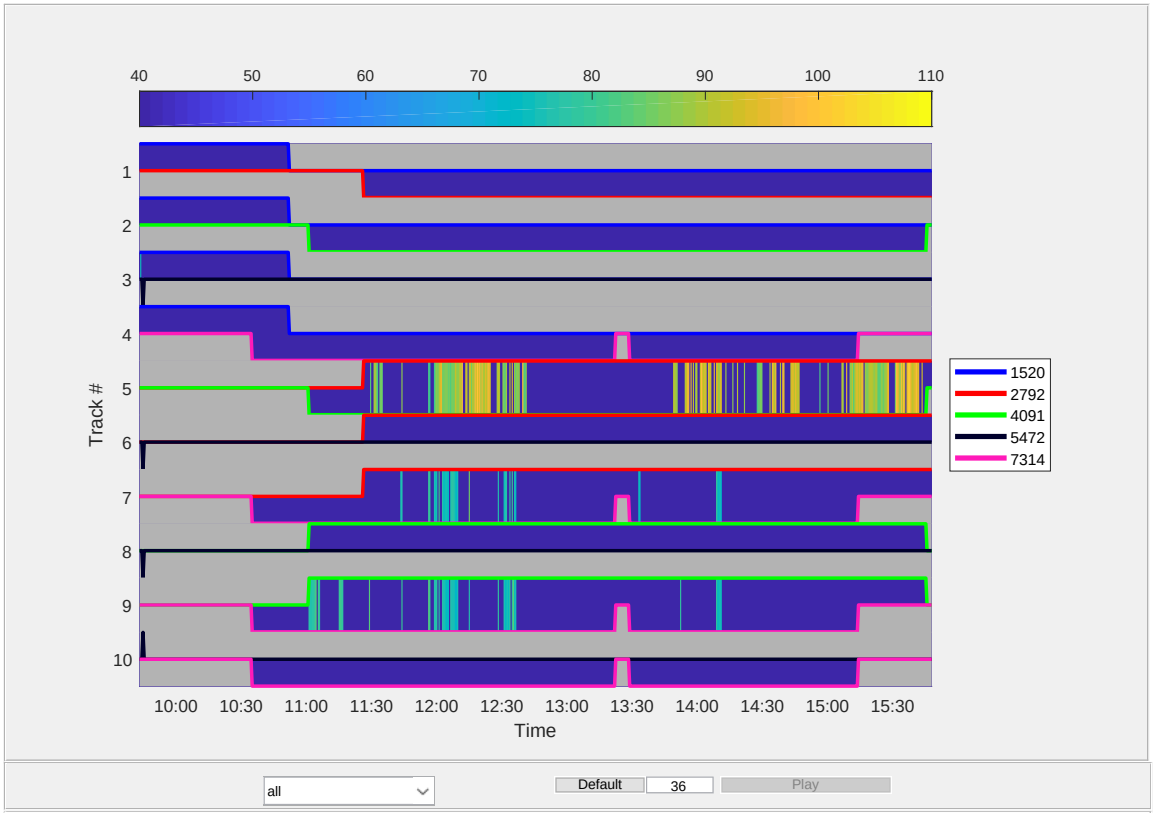

Figure 12. Example activity map demonstrating an identified programming pair (users 2792 and 4091).

products over a single computer. On activity maps, these show up as standalone pairs with very dense detected interactions. The activity map in Figure 12 shows a day with five users, two of which form a programming pair. The pair can be identified by their frequently high similarity measures. This same map shows two users with very low interaction, 5472 only has a blip of a recording, while user 1520 is only there for a few hours before any other colleagues are present. This map also shows about three check-ins with the programming pair and the Software Factory director (7314) lasting about five to ten minutes each.

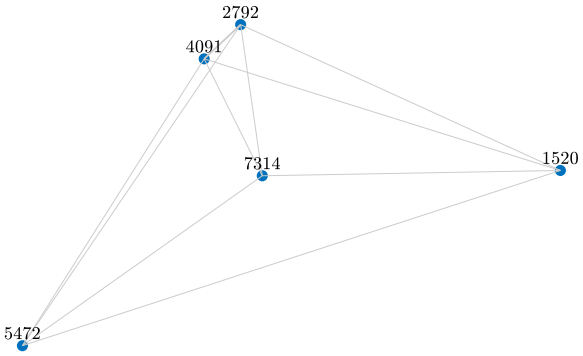

Figure 13. Spring plot for the day shown in the programming pair activity map.

3.4.3 Spring Plots

The last plot type we used were spring plots [13]; network graphs where users are represented as nodes and detected conversation activity is expressed in the widths of the connecting lines and the proximity to other nodes. Because they directly illustrate participant relationships, spring plots are important tools for visualizing observable communication networks. Spring plots are so-called due to the spring-like forces that are modeled to create them. For a given observation period, the nodes representing the participants are randomly arrayed in a space. The conversation weights estimated during conversation correlation are then used to describe spring-like forces between each node pair. To create the graph, the positions of the nodes are re-calculated in discrete steps accordingly to the balance of forces that act on them. Strongly interacting nodes will be pulled together, while weaker links have less influence. After a while, the nodes will arrange in a network that reaches equilibrium, where the forces acting on each node are balanced and the relationships are maintained [14].

Chapter 4

ANALYSIS RESULTS

Using the processing chain described in the previous chapter as a foundation, subsequent analysis and exploration leads to a number of results and conclusions that are described in this chapter. The first two deal with engineering the correlation calculations in a manner that lends itself towards extracting time delay estimates as well as identifying likely conversations. The alignment estimation and correction results indicate that alignment of the recordings is both positive and beneficial in extracting network measures. The speech detection performance section qualitatively assesses the performance of the RMS detector employed in speech detection and concludes that it performs well in marking segments for futher processing. The last section assesses the utility of force-directed graphs in visualizing networks with many users and across different periods of time.

4.1 Correlation Normalization

Normalizing the cross-correlations makes the process more robust to recorder/speaker volume discrepencies.

Each correlation time-lag is calculated as an inner-product, so the resulting values are strongly influenced by the energy of each signal. For this reason, it is necessary to normalize the cross-correlation output to prevent a single strong signal from dominating and creating false alarms or for two signals with strong correlation to be penalized for their low energy. In this analysis, each cross-correlation was normal-

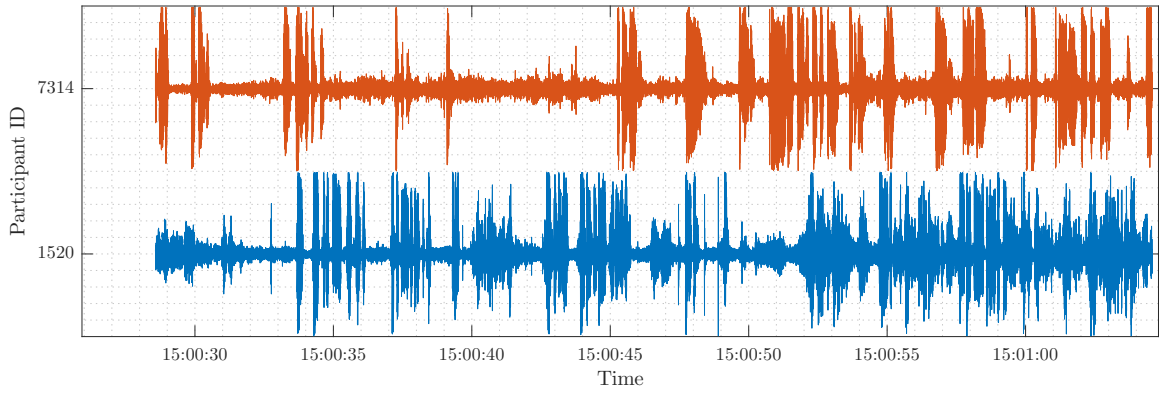

Figure 14. Example of two loud signals occurring simultaneously but in unrelated conversations.

ized by the product of the root mean square of the two signals. The RMS values are a proxy for the energy contained in each signal and are a convenient choice for mitigating the signal level's impact on the similarity measure. This effect is illustrated in the cases below.

Figure 14 shows the time-series data from two time-coincident recordings that are obviously in different conversations but with each containing significant energy. Figure 15 by contrast shows a single quiet conversation captured in two channels. The unscaled cross-correlations for both pairs can be seen in Figure 16. Although the unmatched pair has a much lower peak-to-average ratio, the matched and unmatched peaks are seen at a similar level. Left uncompensated, these cases would stress the detector creating a loss in sensitivity or increase in the false alarm rate. Figure 17 contains the results after normalization showing clearly the matched peak while de-emphasizing the contributions of the strong unmatched signals.

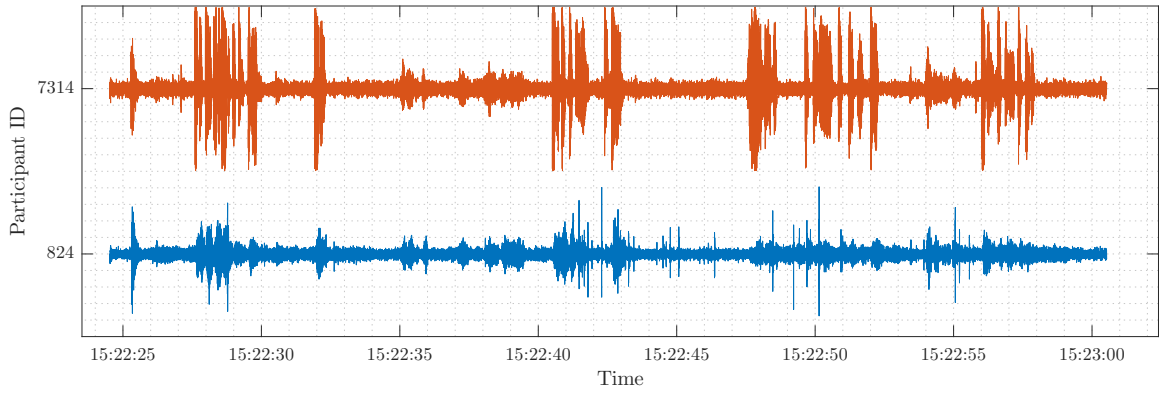

Figure 15. Example of two softer signals; the similar shapes present on each line indicate that a single conversation is being captured in each channel.

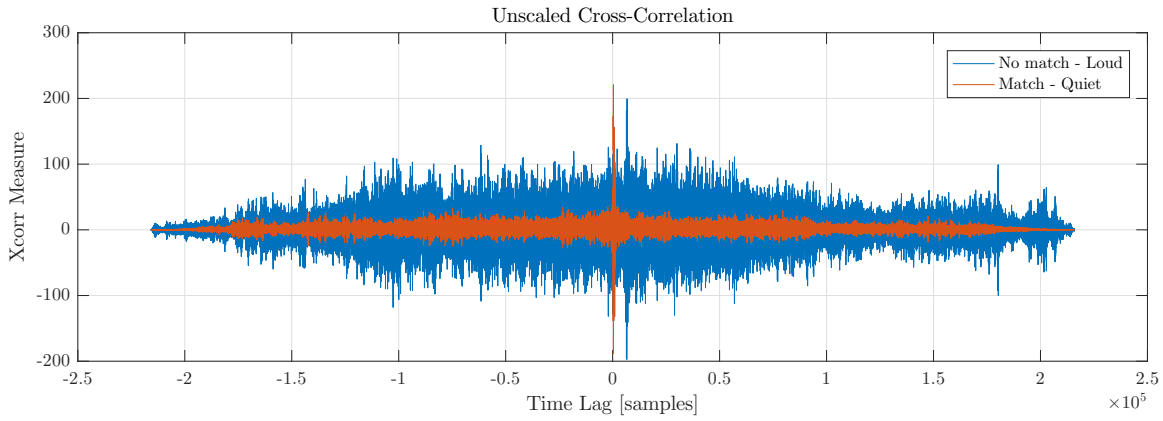

Figure 16. The unscaled cross-correlations from two pairs of signals: a loud, unmatched pair, and a quiet matched set. Note that each peak is at a similar level.

4.2 Choosing an Appropriate Segment Length

The segment length used in correlation is a key driver to the resolution and quality of the results. Choosing a larger segment for time alignment and a shorter segment for conversation detection is beneficial.

Cross-correlating two whole recordings is prohibitively costly from a computational perspective and would at most return a single measure of their similarity during that day. Even if it were possible, the result would be challenged by the inter-

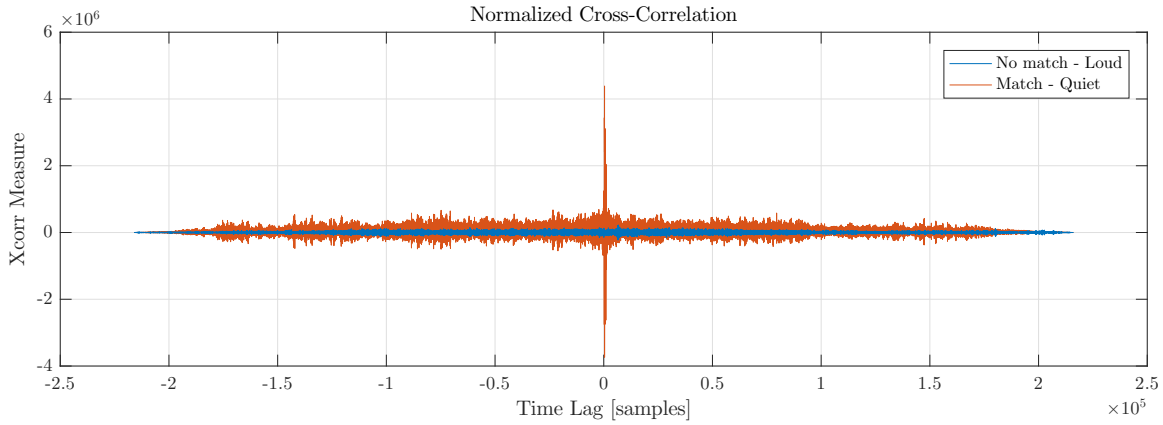

Figure 17. The cross-correlations from two pairs of signals after normalization by the product of their RMS levels. Note that the matched peak is clearly distinguished from the unmatched pair.

recorder skew (discussed further in Section 3.2). To keep the computations feasible and achieve reasonable time-resolution, short segments from signals are operated on creating a downsampled picture of the scene where each segment center creates both a similarity measure and the time-lag at which the peak occurs. In choosing the segment length, there are four primary considerations:

1. The computational resources and processing time go up with the segment length.
2. The resolvable feature-size goes up with the segment length.
3. The tolerable time-alignment mismatch error goes up with the segment length.
4. The SNR varies with segment length depending on the signals.

The clips in Figure 18 illustrate well how both the resolvable feature-size and SNR vary with segment length. In these clips, participants 2792 and 7314 are holding a conversation. At around 13:39, 7314 leaves to make a phone call. After another

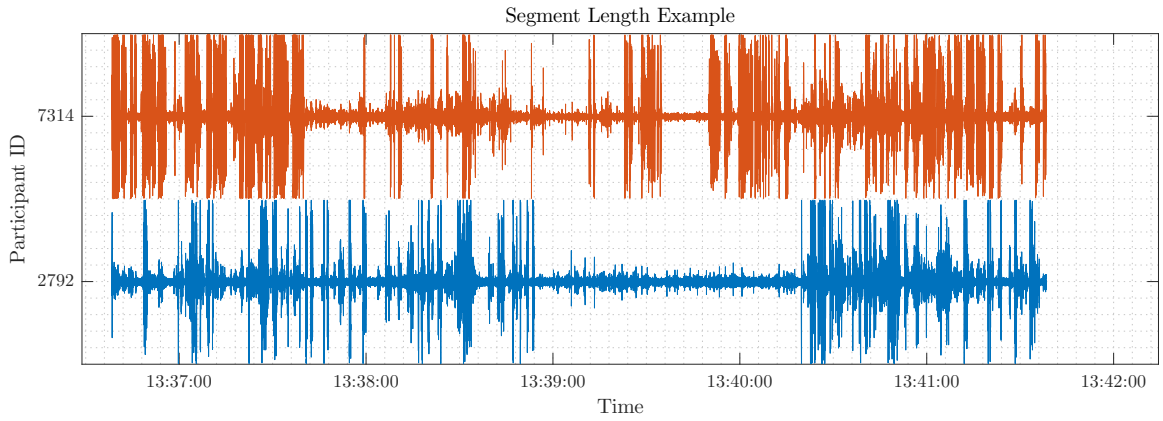

Figure 18. A 5 minute example where two participants converse with a brief pause in between.

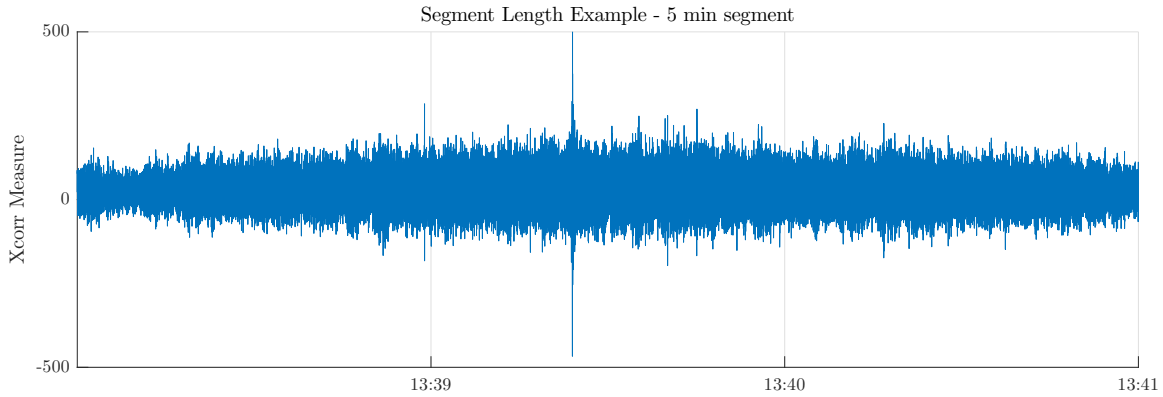

Figure 19. The cross-correlation of a single 5-minute segment from the segment length example.

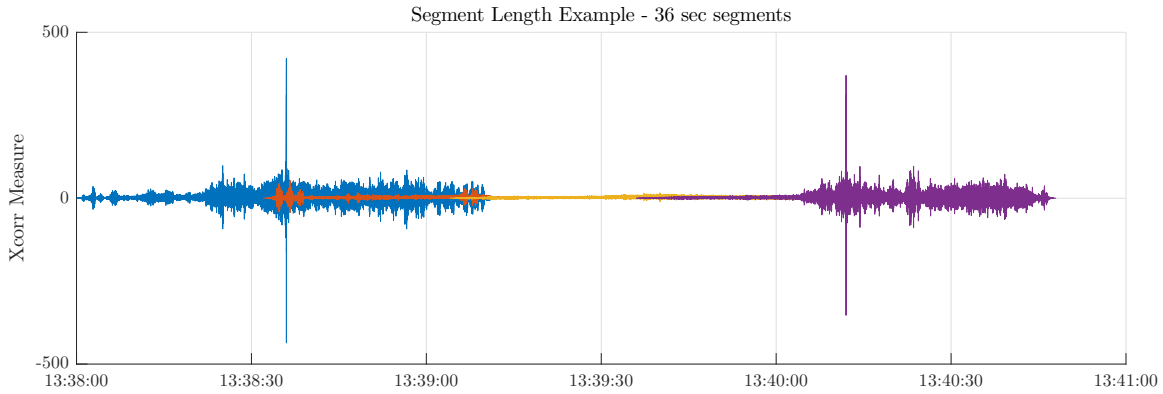

Figure 20. The cross-correlations (aligned by segment center) of four 36-second segments from the segment length example.

two minutes, the pair resume their conversation. In Figure 19, the cross-correlation function using full clips is shown. Because the pair is conversing for most of the clip, a single sizable peak is observed. The lull in conversation doesn't contribute coherently to the peak, but instead raises the average similarity measure everywhere else. By contrast, Figure 20 shows the correlation functions from four pairs of 36-second segments. In two of the segments, clear peaks can be distinguished corresponding to conversations occurring both before and after the interrupting call. During the phone call, the inner orange and yellow segment pairs include mostly noise and barely peak indicating very little similarity. For the shorter segments, less of the interruption is included resulting in a higher observed SNR. The effects are dependent on both the window length as well as the underlying signals.

While the computational burden places a practical limit on the order of minutes, two primary competing factors created the need for a two-pass approach. A coarse sampling of time-alignment errors showed several clips being off by 2-3 minutes, so a first processing pass used five minute segment lengths to accommodate these errors. A second processing pass was then performed on the time-aligned, skew-corrected signal set with a smaller 36 second window. The smaller window processes more quickly, affords better time resolution, and shows a slight sensitivity improvement in areas where communication is sparse. In both the coarse and fine passes, a stride shorter than the segment length was used to step through the data. For the five minute window, a stride of 270 seconds provided 30 seconds of overlap. A stride of 32 seconds was used for the second pass. The overlaps provide slightly better time resolution at the expense of processing time.

4.3 Alignment Estimation

Alignment estimation using the time-lags associated with the peak correlation values is demonstrated by uncovering the inter-recorder drift.

With an initial analysis timelime constructed from the check-in logs, the recordings must go through a further alignment step to correct for calibration errors and recorder drift. In addition to the uncertainty caused by the incomplete logs, calibration errors exist in the complete records as well; likely the result of unsynchronized and/or uncalibrated clocks used to provide timestamps in the logs. There is no global reference to measure skew from or align to. Instead, the mismatch errors must be observed differentially as the time difference of the same signal across multiple recorders and thus the first step to correcting them is to detect these common signals.

The procedure for detecting the signals used for the refined alignment of recordings is similar to that of conversation detection: process each day's timeline from left to right, cross-correlating overlapping segments of the recordings from pairs of participants and retaining both the correlation peak's value and corresponding time lag. With alignment in mind, a larger correlation segment length is used. This increases the correctable offset error while sacrificing time resolution; which is a far less important metric in alignment than in conversation detection. The outputs of this process form [M x N] arrays where:

M: the number of 2-combinations ($\binom{n}{2} = \frac{n(n-1)}{2}$) from the set of participants on that day

N: the number of segments in the day's timeline

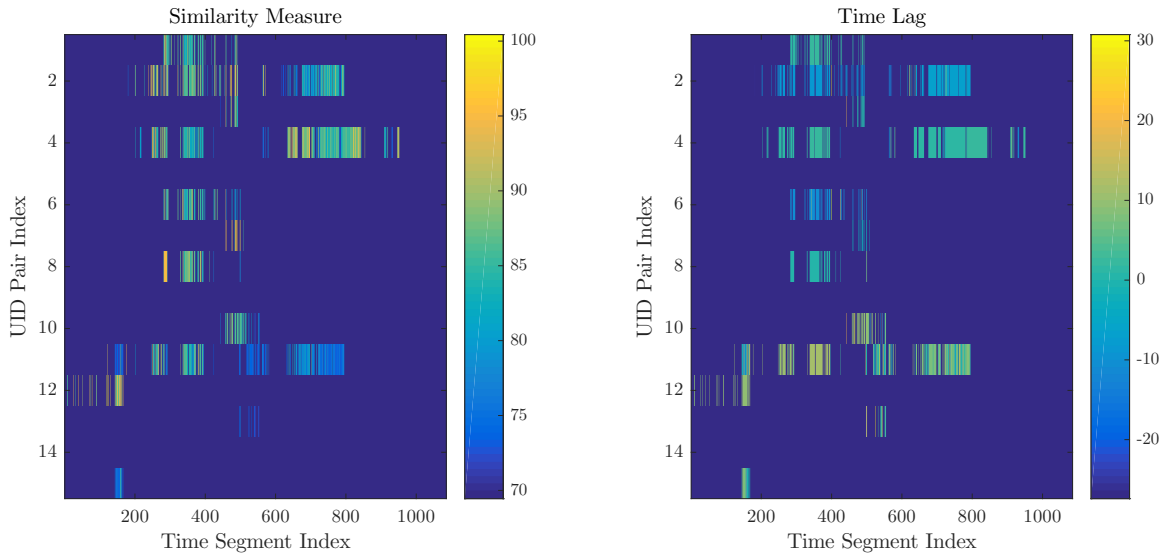

Figure 21. Example output from the refined alignment processing step. The similarity measure (left) and time-lag (right) are used to estimate the time offsets between recordings to refine the alignment.

For the purposes of alignment, the peak's time lag creates the time delay estimate while the correlation peak's value provides a confidence measure on the quality of the estimate. An example output from the alignment correlation stage is shown in Figure 21. Recall the example in Figure 7; on that day there were 6 participants, leading to 15 unique participant pairs. The higher similarity measures on rows 2 and 4 suggest the veracity of the corresponding time lags, which indicate that in pair 2, the first participants track lagged the second's by around 15 seconds while in pair 4, the first participant leads the second by around 5 seconds. Contrast this with the back half of row 11 which has much lower similarity scores that are also reflected in the noisier time lags.

In additional to the initial offset, the time lags created during refined alignment also uncovered drift between participants' recordings during a given day; an effect

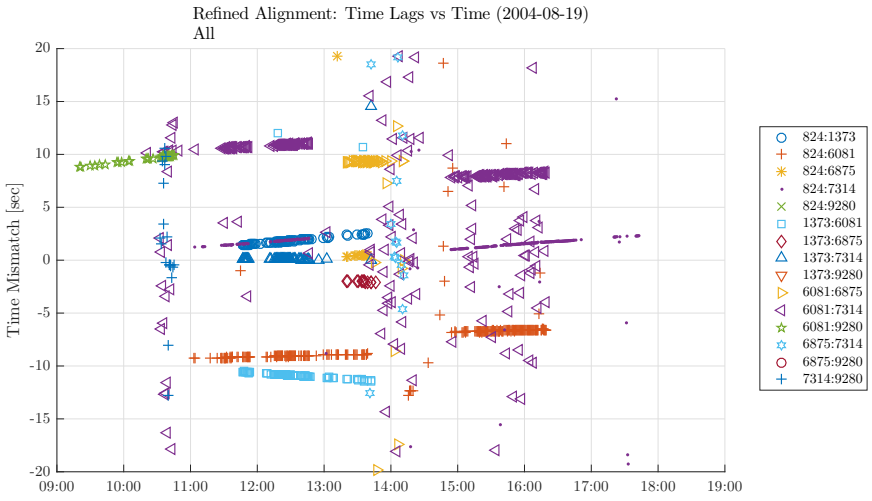

Figure 22. Refined alignment time offset output represented with time-lag as a function of time.

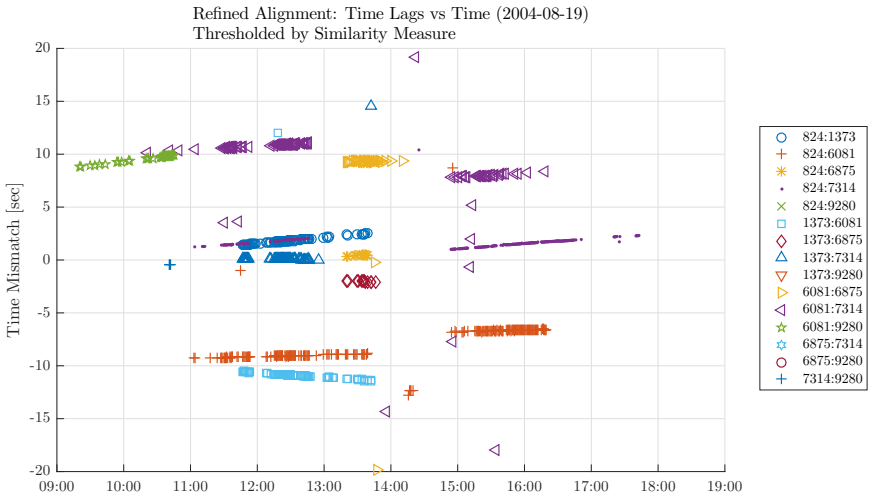

Figure 23. Refined alignment time offset output gated by a similarity measure threshold. Many of the noisy estimates are removed however outliers still remain.

easily seen across much of the dataset when viewing the time lags as a function of time as in Figure 22. This relationship indicates that there is drift between the clocks used to sample the recordings which is likely due to imperfect oscillators on the individual recording devices and is exacerbated by the audio encoding/decoding. The drift's effect is magnified by the long durations of many of the Software Factory recordings; a small mismatch can turn into a large time difference over the course of several hours. To maintain the alignment necessary for interaction detection, the drift between recordings must be estimated and compensated for. If left unaccounted for, clips that are initially aligned could migrate outside of the correlation segment length resulting in a sensitivity loss.

To correct for both the initial offset as well as the drift, the outputs of the refined correlation step are used to fit linear models to the estimated time differences between pairs of recordings. This is made challenging by two competing factors. The time delay estimates used to estimate the linear parameters are only possible when the same signal is present in multiple recordings. As a result, in order to obtain enough samples, the speech-detection thresholds are lowered (see Section 3.3). In addition to raising the probability of detection, this compromise also creates a greater number of false alarms which act as outliers. The outliers, noticeable in the example in Figure 22, can have a lot of leverage against accurate linear regression. As a first line of mitigation, the time delay samples included in the linear regression models are gated by thresholded similarity measures. Only the samples corresponding to pairs with a similarity measure above a heuristically-determined threshold are included. The included samples for the example from Figure 22 are shown in Figure 23; in comparison, the number of outliers is significantly reduced.

Even with similarity measure gating, perfect outlier rejection while simultaneously

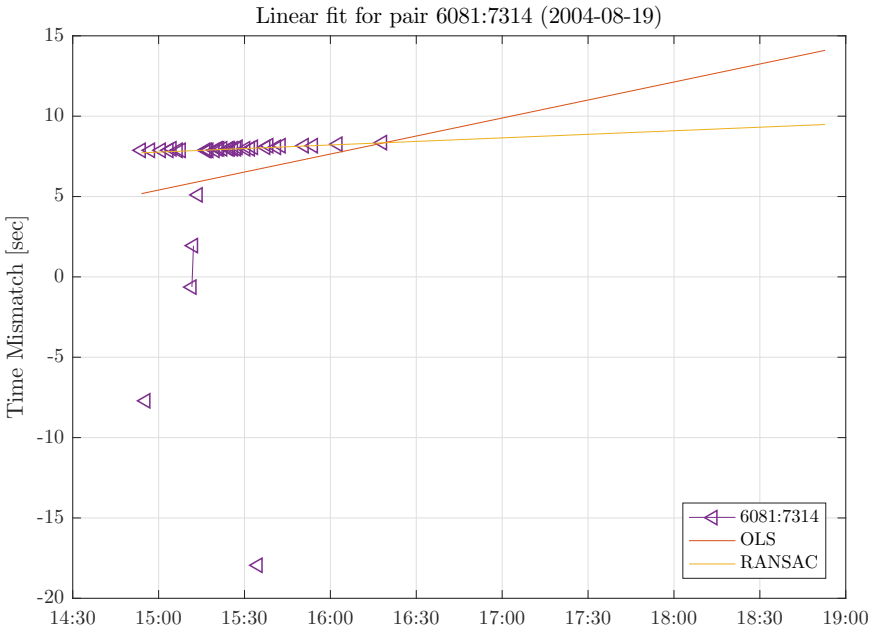

Figure 24. Linear fit for refined alignment time-lags from an example pair illustrating both Ordinary Least Squares and RANSAC.

maintaining good sensitivity is impossible, so a robust linear estimation technique is required to form accurate skew measurements. The ordinary least squares (OLS) estimator is simple but highly susceptible to outlier-induced errors, as demonstrated in Figure 24. Here we see that even with a large number of correlated samples (30) and just a few outliers (5), the line described by OLS is pulled down and if used in the correlation pass would be worse than if no correction were attempted. To handle these outliers, the random sample consensus (RANSAC) method was employed [15]. The line estimated by RANSAC ignores the outliers entirely and fits much more closely to the majority of the data.

RANSAC was selected due to its robustness, simplicity, and configurability. It is a random, iterative algorithm that repeats the following steps:

1. Selects a random subset of the input
2. Form a model for the subset

3. Count the number of samples that fit the model
4. Classify the model based on the consensus score
5. If better than previous, update best model

Many of the steps can contain tunable parameters that affect the robustness and convergence rate of the implementation. With such a large dataset, speed and accuracy are both paramount. The number of iterations and fitness thresholds were used to achieve the desired accurate model probability with minimal computation time.

4.4 Alignment Correction

Correcting the alignment errors is beneficial in creating network estimates that aren't biased by time delays and can be verified by repeating the estimation step.

Once the pairwise recording delays have been estimated, the inter-recording alignment errors can be corrected to improve the conversation correlation performance. Doing so entails changing which samples correspond to a given time in the processing timeline according to the linear models created during refined alignment. The simplest approach for doing so would be to treat the pairs independently and move one participant from each duo according to their linear model. This is the obvious method and won't affect processing time when there are only two active participants. However, when more than two are simultaneously active, treating the pairs independently can result in a single recording being aligned different ways for neighboring models.

Consider the example illustrated in Figure 25 where three simulated signals are

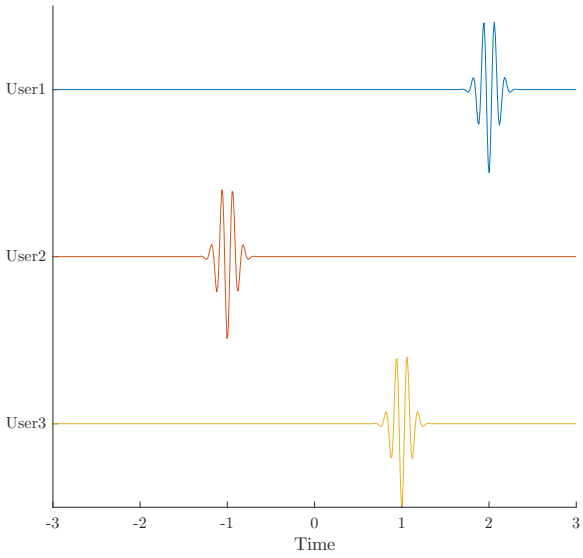

Figure 25. Example of skewed alignment for three participants

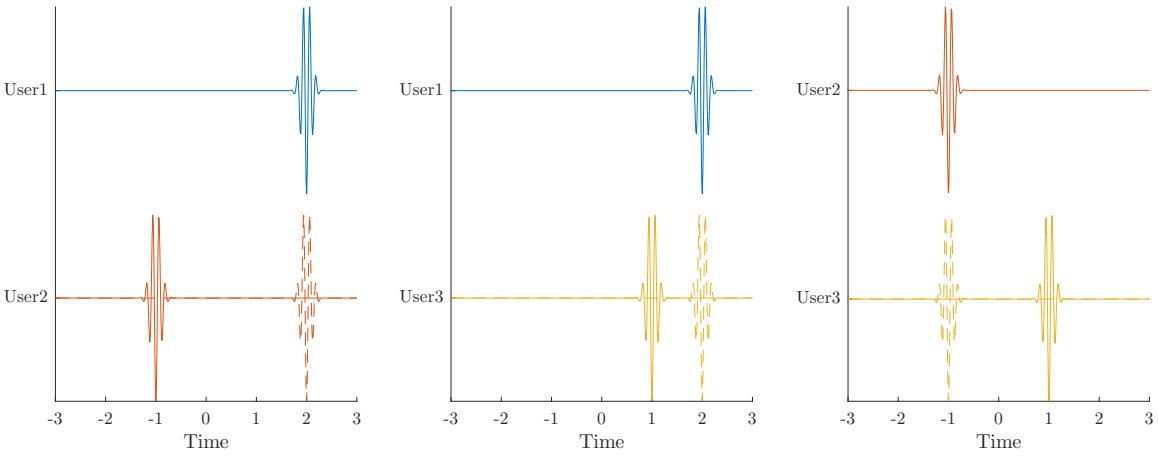

Figure 26. Pairwise alignment correction

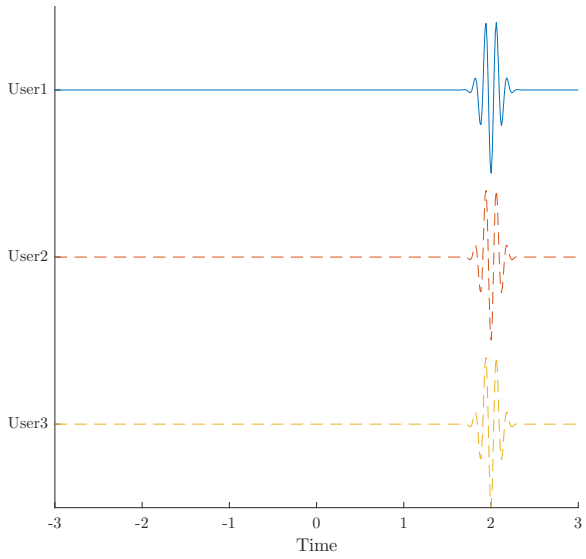

Figure 27. Floating reference alignment correction

offset from each other. An example of the brute-force pairwise correction for this scenario is shown in Figure 26. Note that User 3 is offset differently from Users 1 and 2, which in this approach results in two separate shifts. If, instead, a common reference is realized, then each signal is shifted only once (Figure 27). The pairwise approach takes a large performance penalty; handling unique shifts for a given recording means grabbing a superset of samples that span multiple delays or multiple disc accesses. When using a common reference, each clip is retrieved only once and can then be used in combination with all other active recordings. Applying the alignment correction in this fashion means that conversation correlation can be processed in an identical fashion to alignment correlation: processing each day's timeline sequentially.

Though it greatly reduces the computation time, aggregate alignment correction is complicated by the lack of a complete reference. If there were a measurable signal that spanned the entire day and was present in each participants' recordings, global alignment could be achieved by negating each recording's differential delay with re-

spect to said reference. However, the Software Factory participants did not share common work hours and often would come and go throughout the day creating gaps in the recordings. To reap the benefits of aggregate alignment without a common reference, a floating-reference scheme is used. The floating reference is calculated at each segment as the participant with the highest number of paired time lags. The reference is used as is, while all remaining users' times are adjusted according to their measured delays.

The correction values using the floating-reference scheme for the example set from 2004-08-19 can be seen in Figure 28. The differential measures have been collapsed down to floating references, so instead of having a line for each 2-combination as in the refined alignment output, there is a single line for each participant. Each reference can be spotted along the x-axis, with the other users relative corrections around it. Discontinuities are present at the ends of recordings and whenever a reference is lost. After correction, refined alignment can be repeated to examine the corrections' effect. As seen in Figure 29, a properly calibrated set should create near-zero differential delays across all pairs. The samples from pairs with low similarity measures did not create linear models and were unaffected.

4.5 Speech Detection Thresholds and Performance

Setting a global detection threshold using the statistics of the full dataset sets a passable detector operating point; especially due to the minimal impact of false alarms.

Setting the detection thresholds is challenging in the absence of ground truth for

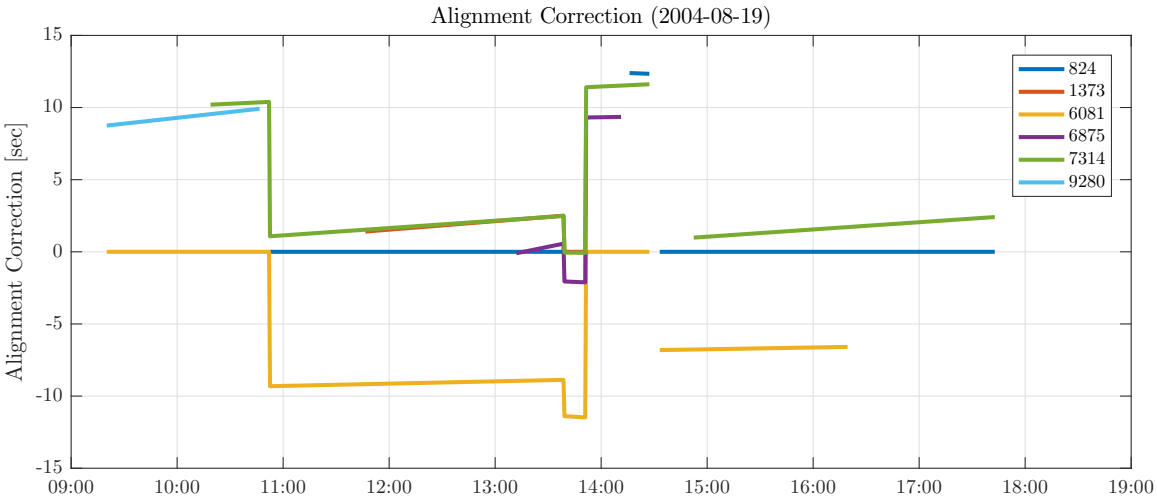

Figure 28. Correction values over time for six participants on 2004-08-19.

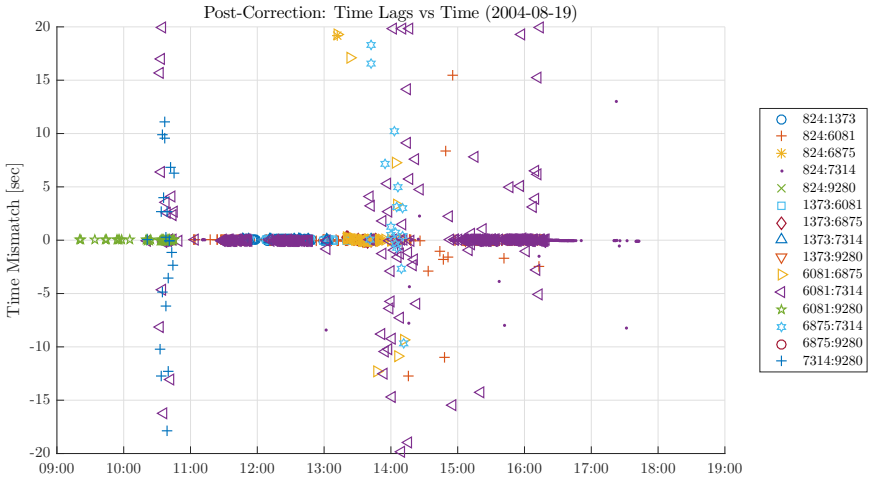

Figure 29. Refined alignment output performed on the aligned dataset on 2004-08-19.

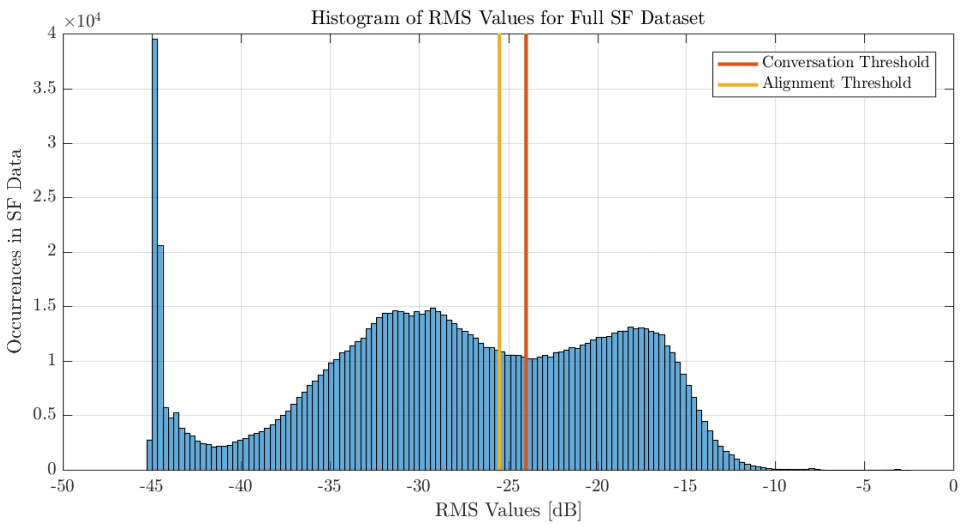

Figure 30. Histogram of RMS Values for all segments in SF dataset plotted with the thresholds used for alignment and conversation correlation.

either the true positive (speech and noise) or true negative (noise-only) conditions. In lieu of traditional likelihood ratios, a heuristic method was used where we randomly sampled the dataset, coded segments manually by listening to the recordings, and then computed the threshold range that would minimize detection errors. After manually coding segments, the resulting threshold range was then judged visually against the entire day and a threshold was qualitatively selected. Because we are processing offline, we can then judge the thresholds against the entire dataset as seen in Figure 30. The histogram shows three main features: a local maxima around -17 with a short upper tail, a local maxima around -31 with a long lower tail, and a spike at -45. Though we lack a fully coded truth, it is likely that upper distribution is speech activity and that the lower distribution is general background noise. The spike at the low end looks like a saturation of the long noise tail likely set by the noise floor of the recorders or compression algorithm.

As the histogram shows, the noise-only and speech distributions cannot be per-

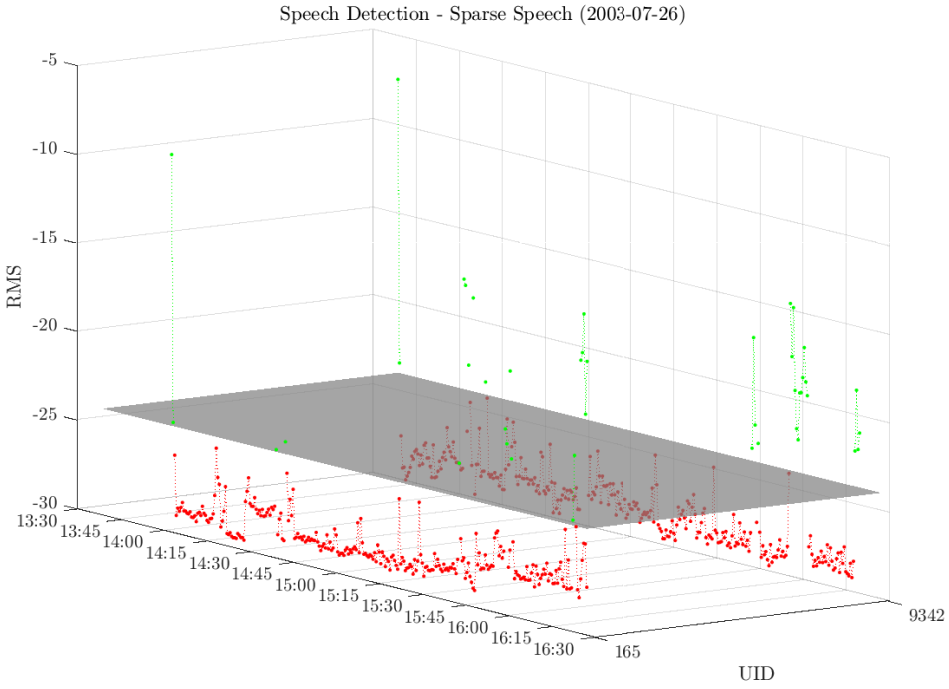

Figure 31. Sparse speech scenario, with two participants' RMS values over time showing the threshold and detector decision.

fectly separated, however reasonable performance can be achieved. Figure 31 shows the detector classifications for a day with two participants with minimal interaction. The conversation threshold is a plane bisecting detected speech activity (displayed in green) and segments without speech (displayed in red). The classifications correspond well to the two distributions seen in the RMS histogram. A similar chart showing an example with denser speech activity is shown in Figure 32. While it is expected that the voice activity should vary widely, it is more surprising that the background noise seems to vary in discrete plateaus. It seems likely that this behavior is caused by the recording compression.

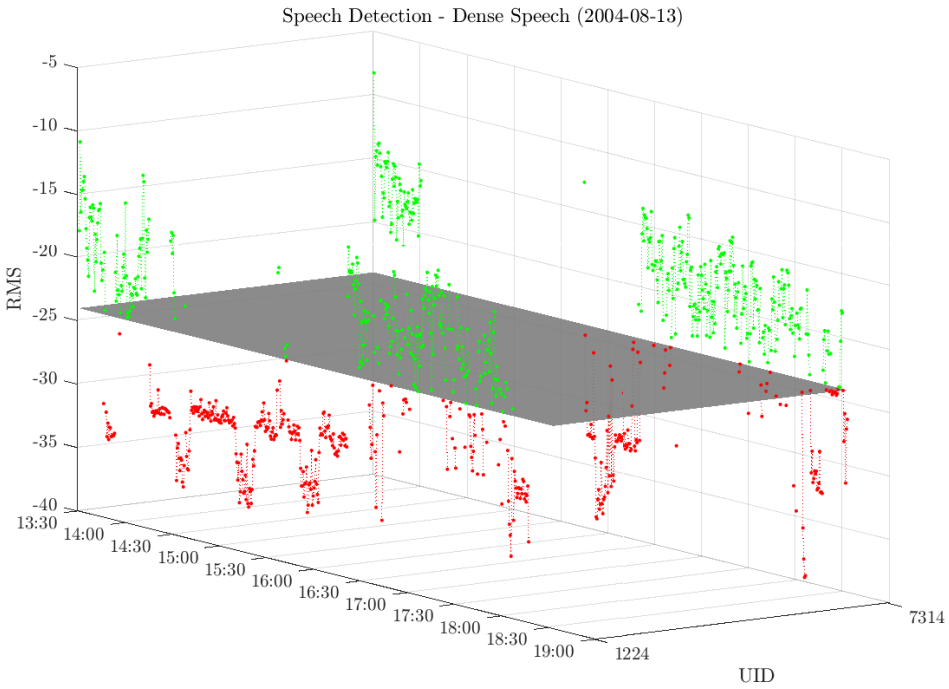

Figure 32. Dense speech scenario, with two participants' RMS values over time showing the threshold and detector decision.

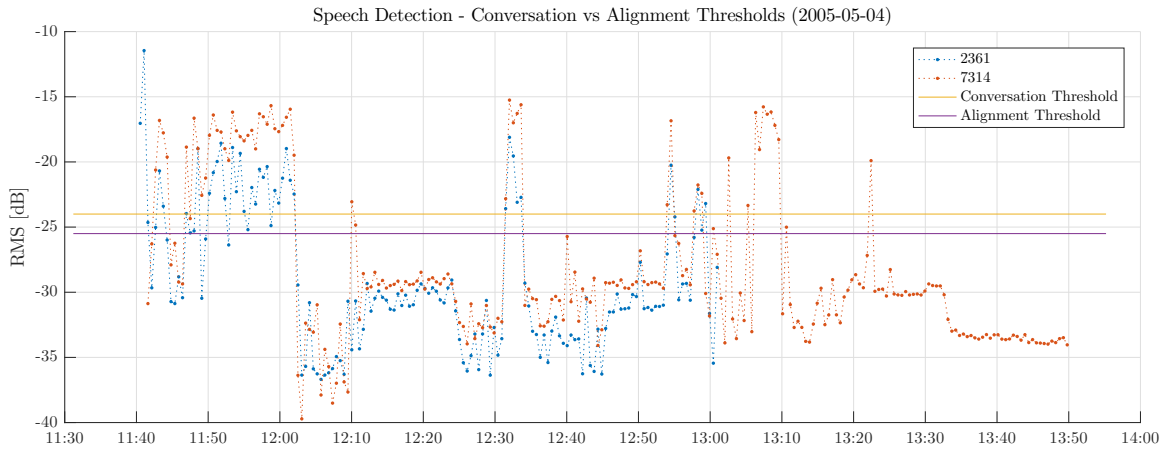

Figure 33. A participant pairs' RMS values plotted over the course of a day. Both alignment and conversation thresholds are shown.

As mentioned in Section 4.3, a slightly lower threshold was used during the alignment stage. This lower threshold was used to take advantage of the lower false alarm penalty; during alignment, false alarms are discarded as outliers in the RANSAC algorithm. As a result, the detector's operating point was shifted to achieve greater sensitivity at a higher false alarm rate. This effect can be seen in the example in Figure 33. By lowering the threshold, twelve additional pairs are sent to the alignment correlation and available for drift estimation.

4.6 Analysis of Networks through Force-Directed Graphs

Force-directed graphs are an efficient way to convey network patterns and can be used in conjuction with network-weights averaged over varying time periods to describe network dynamics.

Unlike activity maps, static spring plots cannot depict a network's changes over time. Instead, a spring plot uses conversation weights aggregated over a period of time to show a snapshot of the average network formed during that span. As recordings are processed by day, daily spring plots are the most common. The spring plot corresponding to the activity map shown in Figure 12 is shown in Figure 13. The programming pair (users 2792 and 4091) are clustered tightly together, with the only other user with minimal conversation detection, 7314, at a moderate distance from the duo. The two users with no detected interaction, 1520 and 5472, don't have any attractive forces and are therefore seen apart from the rest of the network.

Integrating the conversation weights over different durations creates snapshots of the communication networks from which time-dependent dynamics may be ob-

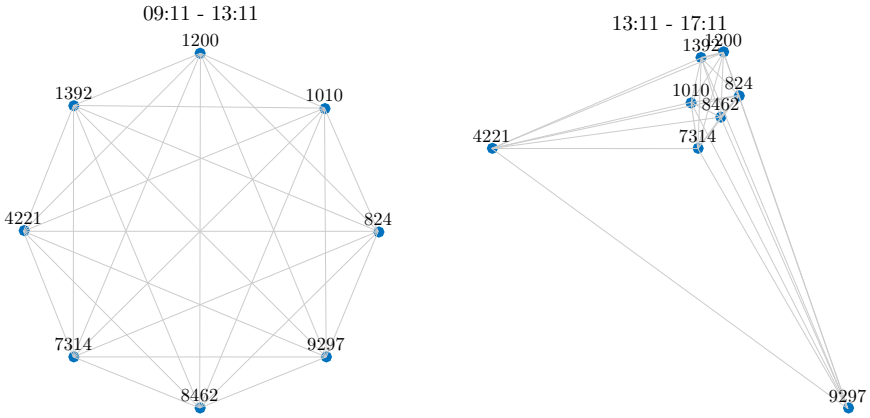

Figure 34. Spring plots showing different activity between morning and afternoon.

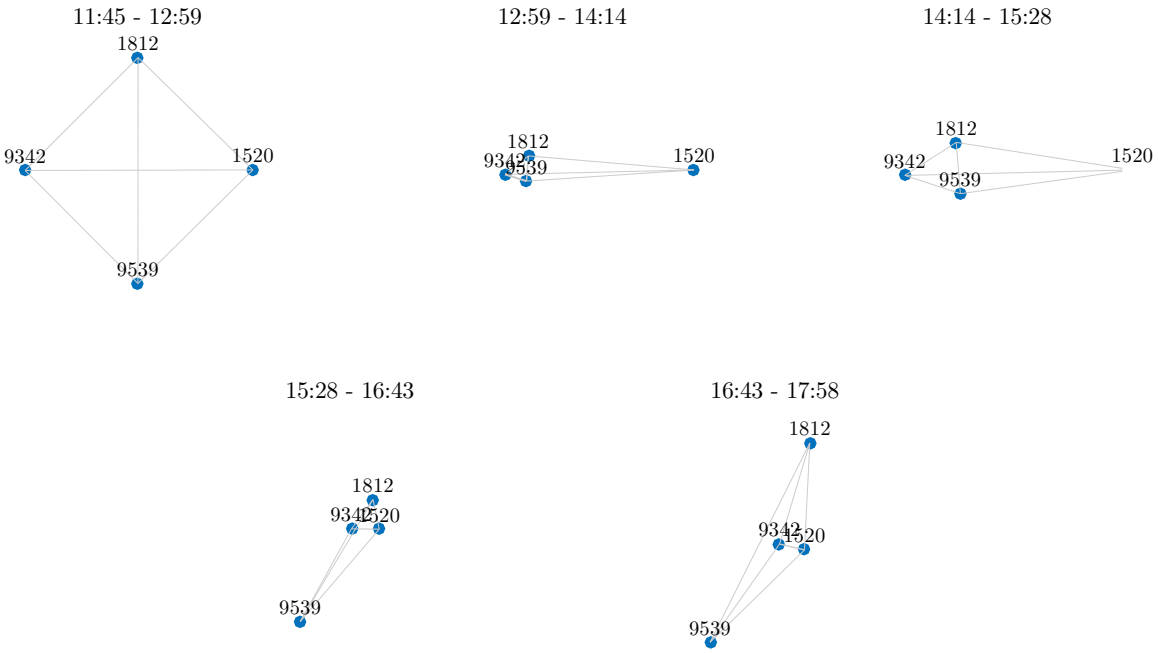

Figure 35. Spring plots for a single day using weights integrated over 75 minute segments.

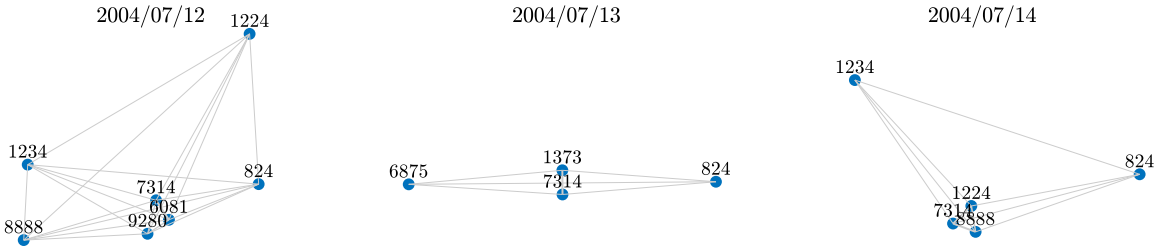

Figure 36. Spring plots for each of three consecutive days in July 2004.

served. Figure 34 illustrates this concept by splitting an 8-hour workday into two 4-hour periods; no activity was detected in the morning, however a six-person afternoon meeting is clearly seen on the right. The sequence of spring plots shown in Figure 35 demonstrates even higher time resolution using the data from the same day as the activity map shown in Figure 11. The conversation weights are aggregated in 75-minute intervals, creating five network graphs spanning from 11:45 to 18:00. The sequence depicts the same transactions that were hinted at from the activity map, but in a more explicit way that condenses all of the individual pairwise observations into a single graph. The four workers start out equidistant with little to no interaction, then form a triplet indicating strong conversation activity. In the next segment, the activity is lessened. The final two networks show changes in participants, where 9539 leaves and 1520 enters, and then 1812 departs.

Conversation weights can also be integrated over multiple days to observe long-term network behavior and trends. Doing so can smooth over anomalous or short-lived communication activity and unearth sparse network connections that don't occur each day. Consider the three-day period depicted as individual spring plots in Figure 36. Taken individually, there are three unique clusters (6081/7314/9280, 1373/7314, and 1224/7314/8888) that are joined each day by participants who share

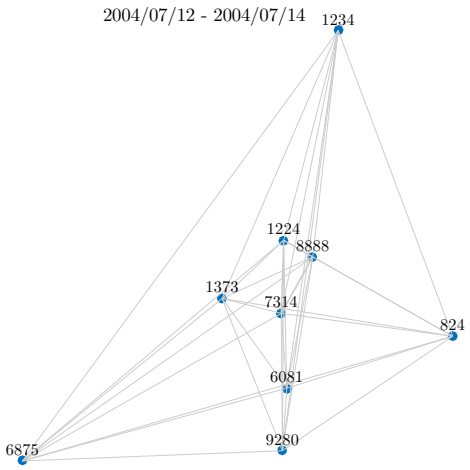

Figure 37. Spring plots integrated over three consecutive days in July 2004.

little interaction with their colleagues. After integrating the activity over the three-day period, however, a different picture emerges. The spring plot in Figure 39 depicts a central figure (7314) surrounded by two rings of users. The inner ring seemingly has close links to 7314 and consists of three groups (1224/8888, 1373, and 6081/9280). The outer ring consists of users who show little interaction over the three days and consists of three users (824, 1234, and 6875). The observed network agrees well with a priori knowledge about the group; participant 7314 is the director of the Software Factory so it comes as no surprise that they are a central figure.

Spring plots are particularly useful as a tool for visually compressing the observed communication activity when the number of participants increases. The pairwise dimension of an activity map with N users grows as the number of 2-combinations: $(N^2 - N)/2$. This scales the activity maps making it increasingly difficult to directly infer communication networks from them. In contrast to the four-user day pictured in Figures 11 and 35, the activity map shown in Figure 38 spans eight users (28 pairs) and is much harder to interpret. The corresponding spring plot, Figure 39,

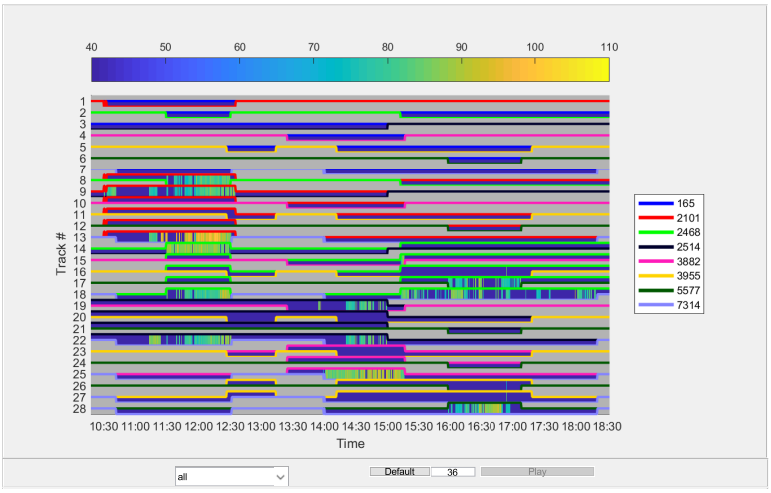

Figure 38. Example activity map for a day with eight participants.

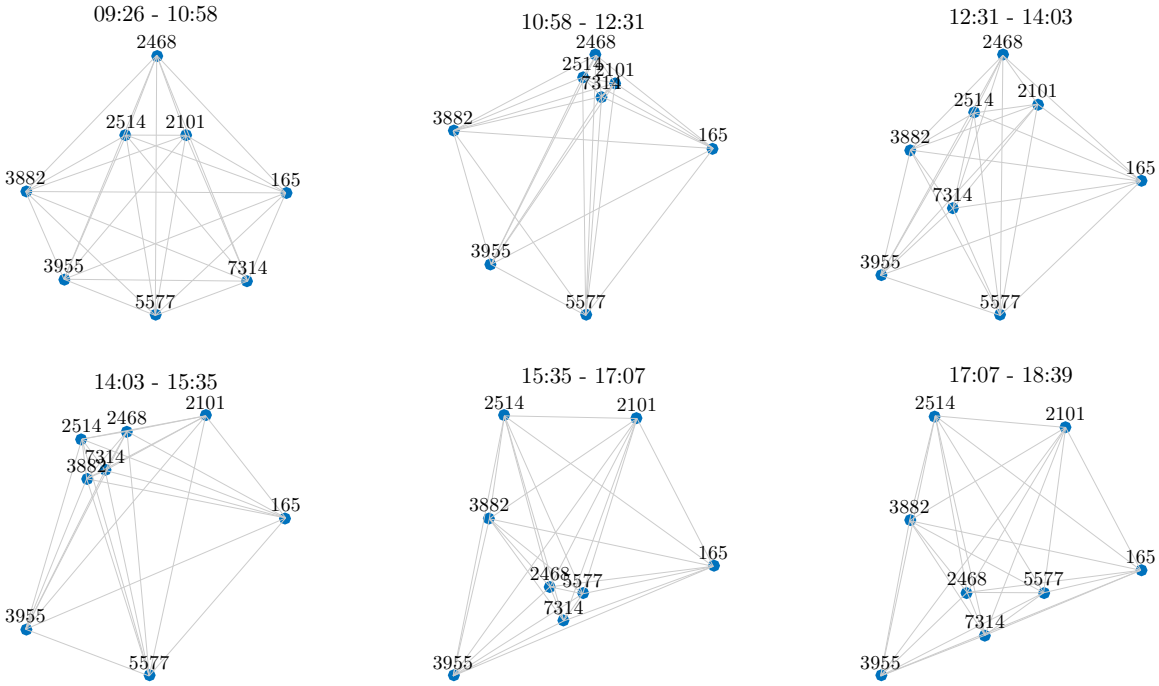

Figure 39. Snapshot spring plots of a day with eight participants.

encodes the same information in a sequence of graphs that clearly distinguish at least three time-local clusters throughout the day. Effectively visualizing the observable communication networks in the large Software Factory dataset is an important step in gleaning information from it for communication research.

4.7 Analysis Summary

By analyzing the individual steps of the network-estimation processing chain, a number of useful results have been demonstrated in the preceding chapter. They are gathered here for convenience:

- Normalizing the cross-correlations makes the process more robust to recorder/speaker volume discrepencies.
- The segment length used in correlation is a key driver to the resolution and quality of the results. Choosing a larger segment for time alignment and a shorter segment for conversation detection is beneficial.
- Alignment estimation using the time-lags associated with the peak correlation values is demonstrated by uncovering the inter-recorder drift.
- Correcting the alignment errors is beneficial in creating network estimates that aren't biased by time delays and can be verified by repeating the estimation step.
- Setting a global detection threshold using the statistics of the full dataset sets a passable detector operating point; especially due to the minimal impact of false alarms.
- Force-directed graphs are an efficient way to convey network patterns and can be used in conjuction with network-weights averaged over varying time periods to describe network dynamics.

In aggregate, these results indicate that a useful processing chain has been developed that can detect speech activity in raw recordings and find correspondences across recordings. It has been demonstrated that these correspondences can be used both to time-align the dataset as well as to estimate pairwise similarity measures. Lastly, force-directed graphs are a good way to visualize the networks and validate the processing chain's performance. As a result, this toolchain demonstrates an important path forward towards providing actual communication network metrics that can be compared against the Software Factory's perceived communication metrics in the pursuit of further social science research.

Chapter 5

CONCLUSION

5.1 Summary

The goal of this thesis was to develop a software toolkit that can be used to estimate the observable communication networks contained in the Software Factory recordings. If successful, the toolkit unearths crucial data for human communication research; providing quantitative observations that can be compared against the qualitative perceived networks and interaction metrics that can be used to form agent-based models. In addition, this work aims to provide visualization tools that can be used both to validate the estimated networks as well as to explore the recordings themselves. The toolkit can aid researchers by providing multi-user aligned audio and highlighting areas of activity that may be of interest.

The central problem is detecting the same voice activity in two or more participants' recordings. This is a common signal processing task and can often be accomplished by a human listening to the files and coding what they observe. This seemingly simple problem is made complicated for the Software Factory data by a number of factors. The quantity of the data - three years worth of recordings - rules out human coding and motivates a signal processing, software-based solution. The data is unaligned and the logfiles that can be used to align them are imperfect, non-homogeneous, and in some cases incomplete. Lastly, there is a lack of ground truth, making it difficult to effectively train an algorithm, set thresholds, and gauge performance.

A four-step process was used to address these problems, process the Software Factory data, and create the toolkit. The first step is parsing the log files; this transforms a flat list of weekly recordings into an approximately aligned timeline. Some of the log files were incomplete and required an additional processing step to link the files to their entries and reconstruct the schedule. After the files are roughly aligned, the next step is alignment correlation. This step uses the recordings themselves to estimate the actual delays between pairs of recordings to be used to refine the time-alignment. During this step, it was noticed that the recording delays between some participant pairs drifted linearly over time. To combat this drift, linear models were developed that were then used to back out the delays and create an aligned timeline.

Conversation correlation is then performed on the aligned data to detect common speech activity. To reduce computation time and false alarms, a speech detector gates which recording segments are processed. After a survey of techniques, a simple threshold detector was employed using the segment's RMS value as the parameter. The resulting pairwise similarity measures form conversation weights which are a proxy for how often a participant pair interacts. These conversation weights are inputs to the final step of results visualization. The toolkit provides three main graphs: clip plots, activity maps, and spring plots. Clip plots display the timeseries recordings from two participants and enable a listener to hear the clips in stereo to validate the similarity measures and gather additional information. Activity maps show an outline of the day for each pair of participants, demarcating the time periods over which each participant pair held overlapping work hours. The similarity measures are shown as the background color at each segment and can be used as as a heat map for inferring a network graph or highlighting regions of activity that may be of interest. Finally, spring plots provide a visualization of the detected networks where

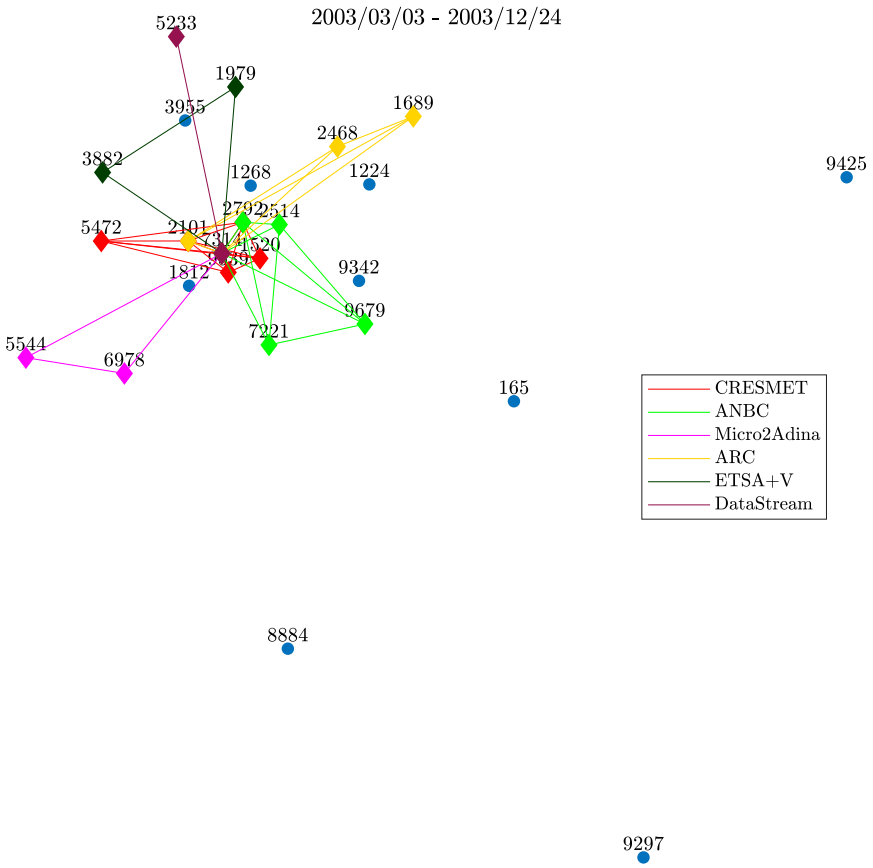

Figure 40. Spring plot formed from conversation weights aggregated over a nine-month period with annotated projects.

the Software Factory employees are represented as nodes and their interactions are encoded in the distance between them and their peers. From the spring plots, the links between the participants can be easily observed and compared against those obtained from their perceived surveys.

For this thesis, the full Software Factory dataset was analyzed, providing time-aligned recordings, estimated conversation weights, and powerful graphs that frame the raw information with a toolkit that can be used to extract and visualize observable networks to further human communication research. The utility of the toolkit is demonstrated by the spring plot shown in Figure 5.1. This plot was generated with

the similarity measures aggregated from March to December 2003. The observable network was then layered with lines connecting known members of projects that occurred in this span. The result depicts a core group centered around the Software Factory director (7314) around which the six projects are arrayed. That the resulting graph forms a coherent image that agrees with surrounding metadata is a testament to the usefulness of the analysis. The Software Factory experiment affords a unique opportunity to observe a group of office workers teaming up on multiple projects over the span of three years. The recordings and associated metadata hold a trove of human communication and network interaction data and the work performed and toolkit developed for this thesis take a step towards unlocking it.

5.2 Future Work

The work performed in this thesis can be extended in three central ways: applying the methods and tools to other datasets, improving the quality of the estimated observable networks, and further exploring the Software Factory dataset network characteristics. Utilizing the developed toolkit on additional datasets could improve the robustness of the algorithms while uncovering the network dynamics of different groups. If a fully coded dataset with ground truth were used, the performance of the toolkit could be more thoroughly quantified and optimized.

A number of techniques can be used to obtain higher fidelity network estimates. Using an adaptive voice activity detection method that estimates the background noise and adjusts the thresholds accordingly should provide increased sensitivity over the heuristically-selected static threshold used in the current RMS detector while maintaining or lowering the false alarm rate [16]. In addition, the SNR of the sig-

nals used in alignment and conversation detection might be improved by using noise suppression or cancellation techniques [17]. Finally, the days with low similarity measures may result from alignment errors exceeding the five minute window used in the current analysis. Extending the alignment search beyond this window may uncover additional conversations.

The Software Factory dataset can be analyzed in different ways to inform future communication research. Graph theory techniques can be employed to understand the make-up and transformation of the uncovered networks at different integration periods and over time [18]. Other features, like hours worked, nationality, gender, age, source lines of code written, and projects could be added to the nodes to expose correlations between features and the observed communication networks. Attempting to identify the participant speaking could discriminate between conversation participants and bystanders and inform the participation rate of the members in conversations [19]. Lastly, the internals of the communications could be analyzed to explore characteristics like tone, pitch, and keywords. Doing so might entail speech-to-text software and/or manual researcher coding [20].

REFERENCES

[1] S. R. Corman and C. R. Scott, "Perceived networks, activity foci, and observable communication in social collectives," *Communication Theory*, vol. 4, no. 3, pp. 171–190, 1994.

[2] P. D. Killworth and H. R. Bernard, "Informant accuracy in social network data," *Human Organization*, pp. 269–286, 1976.

[3] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*. Cambridge university press, 1994, vol. 8.

[4] G. Kossinets and D. J. Watts, "Empirical analysis of an evolving social network," *Science*, vol. 311, no. 5757, pp. 88–90, 2006. [Online]. Available: http://science.sciencemag.org/content/311/5757/88.

[5] N. Eagle, A. (Pentland, and D. Lazer, "Inferring friendship network structure by using mobile phone data," *Proceedings of the National Academy of Sciences*, vol. 106, no. 36, pp. 15 274–15 278, 2009. [Online]. Available: http://www.pnas.org/content/106/36/15274.

[6] *Read audio file - matlab audioread*. [Online]. Available: https://www.mathworks.com/help/matlab/ref/audioread.html#bti4k1b-6.

[7] H. R. Bernard, P. Killworth, D. Kronenfeld, and L. Sailer, "The problem of informant accuracy: The validity of retrospective data," *Annual review of anthropology*, vol. 13, no. 1, pp. 495–517, 1984.

[8] S. Aral and D. Walker, "Identifying influential and susceptible members of social networks," *Science*, p. 1 215 842, 2012.

[9] J. Sohn, N. S. Kim, and W. Sung, "A statistical model-based voice activity detection," *IEEE signal processing letters*, vol. 6, no. 1, pp. 1–3, 1999.

[10] R. Tucker, "Voice activity detection using a periodicity measure," *IEE Proceedings I (Communications, Speech and Vision)*, vol. 139, no. 4, pp. 377–380, 1992.

[11] J. Ramırez, J. C. Segura, C. Benıtez, A. De La Torre, and A. Rubio, "Efficient voice activity detection algorithms using long-term speech information," *Speech communication*, vol. 42, no. 3-4, pp. 271–287, 2004.

[12] M. H. Moattar and M. M. Homayounpour, "A simple but efficient real-time voice activity detection algorithm," in *Signal Processing Conference, 2009 17th European*, IEEE, 2009, pp. 2549–2553.

[13] S. G. Kobourov, "Spring embedders and force directed graph drawing algorithms," *arXiv preprint arXiv:1201.3011*, 2012.

[14] T. M. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software: Practice and experience*, vol. 21, no. 11, pp. 1129–1164, 1991.

[15] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[16] S. G. Tanyer and H. Ozer, "Voice activity detection in nonstationary noise," *IEEE Transactions on speech and audio processing*, vol. 8, no. 4, pp. 478–482, 2000.

[17] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 27, no. 2, pp. 113–120, 1979.

[18] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca, "Network analysis in the social sciences," *science*, vol. 323, no. 5916, pp. 892–895, 2009.

[19] J. P. Campbell, "Speaker recognition: A tutorial," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1437–1462, 1997.

[20] Y. R. Tausczik and J. W. Pennebaker, "The psychological meaning of words: Liwc and computerized text analysis methods," *Journal of language and social psychology*, vol. 29, no. 1, pp. 24–54, 2010.