INTERACTIVE AUDITORY DISPLAY OF URBAN SPATIO-ACOUSTICS

Michael Musick, Johnathan Turner, Tae Hong Park

Music and Audio Research Lab (MARL)
Steinhardt School
New York University
New York, NY 10012 USA
{musick, jmt508, thp1}@nyu.edu

ABSTRACT

This paper presents an interactive exploration platform and toolset for spatial, big-data auditory display. The exploration platform is part of the *Citygram* project, which focuses on geospatial research through a cyber-physical system that automatically streams, analyzes, and maps urban environmental acoustic energies. Citygram currently concentrates on dynamically capturing geo-tagged, low-level audio feature vectors from urban soundscapes. These various feature vectors are measured and computed via Android-based hardware, traditional personal computers, and mobile computing devices that are equipped with a microphone and Internet connection. The low-level acoustic data streams are then transmitted to, and stored in, the Citygram database. This data can then be used for auditory display, sonification, and visualization by external clients interfacing with the Citygram server. Client users can stream data bi-directionally via custom software that runs on Cycling '74's Max and SuperCollider, allowing for participatory citizen-science engagement in auditory display.

1. INTRODUCTION

Citygram, started in 2011, is a project that seeks to capture, map, and explore non-ocular spatial energies. The current iteration of the project focuses primarily on acoustic energies with an emphasis on urban noise. Subsequent iterations will be expanded to other forms of non-ocular energies such as temperature, humidity, and electromageticism. The project is composed of two main parts: (1) a central server that serves as a hub for collecting and analyzing acoustic energy and (2) robust, powerful, flexible, and low-cost remote sensing devices (RSDs). The RSDs collect, analyze, and stream acoustic feature vectors to the Citygram server and contribute to our system's distributed computing architecture. Two types of data streaming mechanisms are available through our RSD designs: *fixed* RSDs and *crowdsourced* RSDs. Fixed RSDs are installed in *fixed* locations and are maintained/managed through the project infrastructure for uninterrupted capture and streaming of spatial-acoustic signals. The crowdsourced RSDs allow the general public and citizen-scientists to contribute to the project via their desktop, laptop, or mobile computing hardware and software.

One of the key design philosophies in Citygram is the embracing of simplicity for hardware and software requirements. For example, contributing as a *streamer* – becoming a *node* on the Citygram system – simply requires a hardware configuration of a laptop computer with a microphone and an Internet connection. For the software part, popular applications like Max use the custom Citygram objects to analyze and extract audio-features that are streamed to the Citygram system. Likewise, accessing the data from these streamers is as simple as using these same Max objects to receive the real-time audio data and drive a synthesis engine via the low-level acoustic descriptors such as spectral centroid and acoustic energy levels. Other application examples include possibilities for telematic performance, where groups of performers could simultaneously contribute and interact with soundscapes in multiple locations, which would be captured by deployed RSDs and used by performers elsewhere. The resulting soundscapes would then be accessible by users via the Citygram infrastructure, which could then be strategically exploited in a compositional context for various performance and auditioning formats including concert settings and personal studio settings.

Another key component of our research and auditory display efforts is to address noise pollution, as it is one of the most important quality-of-life issues in dense urban areas such as New York City (NYC). This collaborative research with NYU's Center for Urban Science and Progress (CUSP) takes to heart data from the last few years which show that excessive and unwanted noise has been the top complaint registered with NYC [1]. Beyond the issue of annoyance, noise pollution contributes to cardiovascular disease, cognitive impairment, sleep disturbance, tinnitus, and has also been shown to have a negative impact on learning [2].

One of the primary goals of Citygram is to provide researchers, artists, and the general public an interactive spatio-temporal platform to explore non-ocular dynamic energies that occupy physical spaces. In effect, Citygram provides an expansive and participatory – encouraging citizen-science engagement – spatio-temporal exploration platform for auditory display, sonification, artistic endeavors, and research. We place importance in community involvement for a number of reasons including: (1) public awareness, education, and transparency for potentially sensitive issues involving privacy,

(2) encourage awareness of well-being for city-dwellers, and (3) providing a mechanism to scale our cyberphysical sensor network.

2. CITYGRAM FOR AUDITORY DISPLAY

The current version of Citygram contains a number of tools for the research and exploration of urban spatio-acoustics and audio-visual display. Included in the current version (soon to be made public) are a server platform, an online exploration portal, desktop based streaming and pulling software, and a network of fixed RSDs all at the disposal of the general public.

Prior work in the area of soundscape sonification has often suffered from a narrow scope of applicability caused by over-specialization, expensive sensor network deployment methods that lead to poor scalability, and an inability for the public to effectively access or collect data. In the era of "big data science," which we currently live in, open data paradigms and massive user collaboration capabilities are becoming increasingly ubiquitous and important [3]. Citygram is able to embody many of these design philosophies, and benefits from them in regards to offering an accessible and scalable general spatio-acoustic exploration platform.

A related project by Fink et al. [4] is the *Soundwalks* project. This project is generally concerned with collecting soundscape recordings, automatically classifying the recordings based on types of audio events and locations, and curating a “soundwalk” for a user through a web-based portal. As with most sonification platforms, the Soundwalks project is based on an “audio-snapshot and upload” model that requires its users to record off-line, followed by manually uploading audio files to a server. In contrast, Citygram automatically streams all captured audio related data to its server in real-time without manual intervention – this process happens in the background and is unobtrusive to the user. Citygram thus provides a simpler interface and will hopefully encourage citizen-scientists to participate and contribute to the project. A comparable issue, emblematic of many similar platforms, was Soundwalks’ limited distribution mechanism to potential users: it focused on a single web-based platform that created machine-curated walks, based on a user-chosen location, limiting the free distribution and use of the collected data. Citygram provides an alternative approach through the active development of multiple deployment platforms, including desktop, mobile, and web-based interaction applications. Although the work in [3] was used as an example, other similar projects do have related issues with respect to creating an infrastructure for citizen-science participation, and a means of interaction with data for the general public [5].

Citygram presents a wide range of opportunities for engagement by the auditory display community. These opportunities encompass the use of the raw collected data including low-level acoustic feature vectors, as well as machine learning analysis that is currently under way. Both of the aforementioned outputs could be used for novel research or to build on previous projects, such as the proposed framework for soundscape re-synthesis by Valle et al. in [6]. A fairly rudimentary, though potentially fruitful, application of the raw data would be to serve as an interactive *soundmap* for the visually impaired. By using the real-time streaming data collected from the RSDs, both sighted and visually impaired users would be able to experience an aural representation of the perceptually salient acoustic feature vectors in an urban setting. A similar example would be a soundmap of traffic patterns as classified by machine learning algorithms that are currently in development. Such sonifications might provide a user (joggers, drivers, etc.) with audible cues for the best “acoustic route” to take on their daily commute. Additionally, the acoustic event detection (AED) algorithms being developed will allow users to aurally parse acoustic event densities on an audio-visual map, providing a dynamic and multi-sensory representation of urban spaces. An experimental feature we are also looking into is mood detection based on the urban spatio-acoustic features. While this data also naturally lends itself to visualization – as a color-coded map of an area's mood, for instance – the auditory display community could easily map it to *earcons* [7] that could be utilized by a user on their smartphone, for example. Many of the examples are applicable in the context of sonification for their dual focus on both practical application and aesthetics while empowering users to explore their world in unique ways.

Figure 1: Citygram visualization interface showing heatmap visualization of RMS data, near Washington Square Park in Manhattan.

2.1. Website Portal and Visualization Tool

The main interface for interacting and viewing data streams from RSDs is the Citygram visualization web interface, which allows users to explore the spatio-acoustic data, listen to specific RSD audio streams, and search for RSDs. The primary means of exploring the data is through heatmap visualizations of a particular feature type from all active RSDs, overlaid on the Google map API. This is available as a real-time display, in which the current time of the Citygram server is used to control and display all RSDs that are actively streaming features to the database. These heatmap visualizations are also available for historical data displays, in which a user can control start/end date/time, with additional loop playback options. Additionally, when accessing historical data playback speed can be controlled, which allows the user to view broad representations of large time periods, or the minutia of change for a short period.

The current visualization page serves two purposes: (1) provide real-time heatmap representations of urban spaces and

(2) serve as an interface for RSD discovery in the larger Citygram infrastructure. All RSDs are shown on this web interface, regardless of their active status. When heatmap visualization is enabled, RSD nodes that are actively streaming are represented by solid black dots whereas inactive ones are represented by grey dots. In addition to the visual node representation, active RSDs will have an associated "heat ring," mirroring the relative value of the selected visualization feature. For example, for RMS, the ring can be interpreted as acoustic energy captured by the RSD as shown in Figure 2.

Figure 2: Visualization interface showing active and non-active remote sensing devices

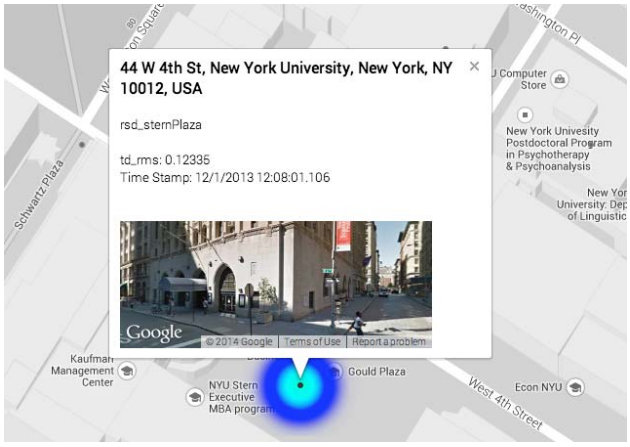

Figure 3: RSD popup, showing relative address, RSD UID, feature value, most recent timestamp, and Google street view image.

The node representing each RSD also provides users with the ability to get more information about a particular RSD: when a user selects an RSD through one of the nodes, a number of additional details associated with the RSD are accessible. This includes; static information, physical address of the RSD, RSD unique ID (as will be discussed below, this allows users to get features from this RSD through the Citygram API), and a Google Street View image to lend perspective on the location as shown in Figure 3. Additionally, if the user has selected a feature type to visualize, these "popup windows" will display the current feature and timestamp values. Finally, in supported browsers, a playback bar to enable monitoring of live audio stream from the RSD is available (see Figure 3 and 4).

In addition to visualizing feature values through the heatmap, this web page allows Citygram API users to find necessary information about RSDs to pull features from them. Another feature included in the interface can be found in the "Map Location" section. Here users can enter a locations "known name", addresses, or latitude/longitude coordinates, to find RSDs in specific areas, as shown in Figure 4.

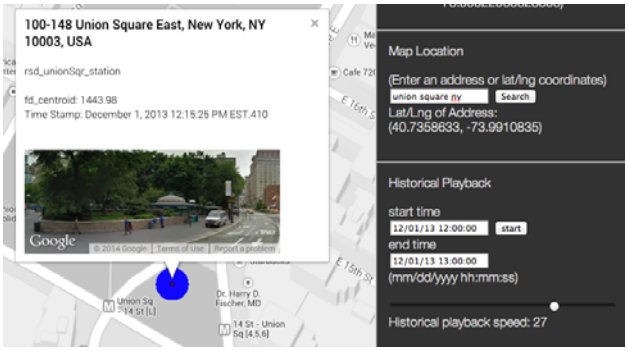

Figure 4: Visualization Page showing Map Location Box, and Historical Playback Control section.

2.2. Pushing/Pulling Data with SuperCollider

SuperCollider is an open source programming language for real-time sound synthesis and composition [8]. With a widespread community of users and developers it is used in a number of situations including sound synthesis, music education, and composition. In addition to these traditional uses, it is increasingly being leveraged as a tool for analysis and music information retrieval (MIR) [9]. The flexibility, power, open source nature, and user community that SuperCollider possesses makes it an apt tool to develop within and provide a library that ties into the Citygram API.

As of this writing, two classes have been developed for the SuperCollider Citygram library to ease communication with the Citygram infrastructure and include feature streaming and feature pulling functionalities.

2.2.1. CitygramStreamer

The `CitygramStreamer` class allows users to stream audio features from their computer to the Citygram server. In order to instantiate a `CitygramStreamer` object within SuperCollider, the user simply inputs the username and password they registered through the Citygram website. These are then sent to the Citygram server with the user's computer device ID. A number of verification steps occur on the server side: the server first checks that the user exists and then applies the device ID to determine if the user has RSDs registered with this particular device. If this is the first time a user has streamed features from this RSD, a new unique ID (UID) is created, representing this computer as an RSD. If it is determined that this computer has already been registered, the

existing computer UID is collected. The verification process ends when the UID is returned to the user – the UID is subsequently utilized to stream features to the server. Our UID management mechanism facilitates RSD management for Citygram users and also ensures that a single UID exists for one hardware device.

In addition to supplying the username and password during object instantiation, the user can optionally provide latitude and longitude coordinates during or after the registration process. In the former case, latitude/longitude values are updated for existing RSDs or newly created for new RSDs. In the latter case, the latitude and longitude are updated only when it is a new RSD.

Once the `CitygramStreamer` class is instantiated a call to the object's `streamFromMic()` method starts the streaming process. This method is a convenience method, which assumes the user will be streaming from one of the audio input busses on their selected hardware device. Alternatively, the user can call the `streamFromBus()` method if s/he would like to stream from an internal bus, thereby applying their own signal processing to the audio path prior to feature extraction and streaming. When calling either of these methods, the user can optionally supply the bus number that will be used as input, the feature poll delay amount, and the push delay amount. The feature poll delay refers to the polling rate, in seconds, for feature extraction, with default value of 0.02 seconds. The push delay amount is the amount of time between feature set pushes to the server, which defaults to 8.0 seconds.

The streaming class utilize a number of SuperCollider's built-in analysis unit generators (UGens) and a `Task()` to control the feature polling and pushing. The task polls all of the analysis UGens according to the feature poll delay value and collects these values, along with a Unix Epoch formatted timestamp, into a series of arrays. All data sent to the server is formatted to a JSON object and transmitted to the server via a time-stamped `cURL` call.

```
// Instantiation Example,
// without lat/lng supplied
~cg = CitygramStreamer(username:
    "YourUserName", password: "YourPassword");

//Start Streaming from Mic
~cg.streamFromMic( in: 0 );
```

Figure 5: Showing the ease of use to instantiate and start streaming from SuperCollider.

2.2.2. CitygramPuller

The `CitygramPuller` class is intended as a wrapper for handling the Citygram API calls to pull data from the server. When instantiating this class, the user must choose one of two methods of instantiation: `oneRSD()` or `multipleRSDs()`. Instantiating `CitgramPuller.oneRSD()` creates an object that pulls all available features from a single specific RSD. Instantiating with `CitgramPuller.multipleRSDs()`, creates an object that pulls one specified feature from a list of one or more RSDs. Each RSD is identified and selected via a 32-character UID, which is created for an RSD when a client registers new RSDs with Citygram. To pull features from deployed RSDs, Citygram provides a custom-mapping interface built on the Google Map API, mentioned in the previous sections.

The user has a number of options for pulling data from the Citygram server. This includes pulling real-time feature streaming from *active* RSDs and also pulling historic data from *past* RSD streams archived on the Citygram server. The date/time is supplied as a Unix timestamp to the object when pulling historic data. When using the `multipleRSDs()` method, the user also needs to supply the feature they would like to pull for the RSD list being queried.

After successful instantiation of a pulling object, the user can call `getData()` on the returned object or simply call `start()`. The `getData()` method returns a single set of data whereas the `start()` method places the `getData()` method in a repeating task that works through the returned data sets at a specified frame rate. The window size of the data pulls and the data update rate are user definable class variables. These classes use a sample-and-hold algorithm (similar to the visualization page on the website) to work through each RSDs returned feature timestamps and they assign the associated value to the classes' value variables and data busses.

By placing the returned features in both class variables and control buses, the user is able to access the data as it fits their particular sonification situation. The two custom methods described provide multiple ways of pulling data from the Citygram server, and also provide flexibility for how a user can incorporate the data into their SuperCollider workspace.

2.3. Pushing/Pulling Data with Max

Cycling 74's Max (formerly Max/MSP) is a graphical language for audio, data, and video processing widely used in the computer music community[1]. The language focuses on real-time interactivity, and is primarily programmed through the use of virtual "objects" connected by virtual "patch-chords." As with SuperCollider, Max enjoys a large user community across various multimedia disciplines, which include musicians, visual artists, and cross-disciplinary multimedia practitioners. The capability of the language may be extended via user-created objects written in C/C++, Java, Javascript, or Max's proprietary Gen language. As of this writing, objects for pushing and pulling data to the Citygram database have been written.

2.3.1. citygram~

`citygram~` is the main object used for streaming data to the Citygram server via Max. Registration of the object is handled in the same manner as with the SuperCollider classes: the user enters a username, password, and device ID before being allowed to stream. Internally, `citygram~` performs all of the same checks on the Citygram server as the SuperCollider

[1] www.cycling74.com

classes, allowing for ease of use and minimal chance of duplicate entries in the RSD database.

Changing or updating an RSD's latitude and longitude values are accomplished via an interactive mapping interface on the Citygram website as shown in Figure 6.

Figure 6: Showing a user's collection of RSDs and allowing for individual latitude/longitude updating through a "drag and drop" map interface.

2.3.2. citygram.puller

The counterpart to the `citygram~` streaming object is the `citygram.puller` object. Unlike the `citygram~` object, `citygram.puller` requires no registration or initialization. Users can access and pull data from any RSD if the UID is known. RSD UIDs can be determined through the Citygram visualizer as shown in Figure 6.

Feature vector pulling is accomplished via a `cURL` call to the Citygram server. Three types of requests can be made to the server: one feature vector from a single RSD, all feature vectors from a single RSD, and a single feature vector from a set of RSDs. In order to obtain the former, the user sends the `oneFeature` message to the object after initializing the RSD UID, feature, and time fields. Similarly, a message of `allFromOne` returns all features from an RSD in a given time window selected by the user. The `many` message can be sent along with multiple arguments – the UIDs of a number of RSDs – in order to retrieve a single feature vector for a set of different RSDs. All information is returned as a nested JSON string. Using Max's built-in "dictionary" family of objects, this JSON string is translated into a nested dictionary, which can be easily parsed by value or data-type. This information can also be streamed iteratively at a rate specified by the user, enabling many possibilities for sonification.

3. FUTURE WORK

There are currently a number of plans for the deployment of fixed RSDs throughout select locations in New York City via Android-based hardware and software as outlined in [10, 11, 12]. In addition to providing mechanisms for data access to the public, we are also investigating and developing specific applications for the Citygram cyberphysical system. As mentioned in the introduction, one of the potential applications of Citygram will be in areas such as citizen health and the well-being of city-dwellers. We are also investigating the use of our system to capture emergency situations, such as Hurricane Sandy, which brought devastation to NYC in 2012. Other examples include augmenting existing gunfire locator systems used by public safety organizations [13] or simply using soundmaps to find the quietest walking path in Manhattan. In addition to "research," we are also exploring the use of Citygram for artistic purposes, where the features streamed from performers around an urban environment could be used to augment electroacoustic performances, or to drive interactive installations.

However, in order for Citygram to be meaningful and perhaps even successful, the number of deployed RSDs will have to reach a critical mass. This criterion is one of the main reasons why substantial emphasis is being placed on cost-effective scalability and developing community engagement mechanisms. Finally, the fixed-RSD solution detailed in [12] allows for collaborative deployment with government, academic, public safety, and research institutions. The devices are non-intrusive, have low-power consumption specifications, and are capable of having their software updated remotely. These are the factors that have allowed for the current partnerships with various institutions to form and mature.

4. SUMMARY

This paper presented the Citygram project as an infrastructure that allows for multimodal participation for professional research organization as well as the general public. The Citygram infrastructure makes it possible to straightforwardly pull real-time or historic data, from any of the field-deployed RSDs. In addition to emphasizing ease-of-use, it has been demonstrated that due to its scalability, varied deployment methods, and data accessibility mechanisms, Citygram is a potential infrastructure for the acquisition, analysis, visualization, and auditory display of spatio-acoustic data. Exploratory tools, such as the Citygram web interaction portal, were demonstrated with a focus on future applications. A detailed explanation of the project's data acquisition methods, vis-a-vis the pushing and pulling architecture, focused primarily on the desktop environments. These environments provide the data in a way that naturally lends itself to sonification, visualization, and big-data analysis.

5. ACKNOWLEDGEMENT

We would like thank Google Research for their support.

6. REFERENCES

[1] J. Fecht, "New York Mayor in fight against noise pollution," *City Mayors Archive*, 2004. [Online]. Available:

http://www.citymayors.com/environment/nyc_noise.html. [Accessed: 12-Feb-2014].

[2] A. Nadakavukaren, *Our Global Environment: A Health Perspective*. Waveland Press, Inc., 2011, p. 534.

[3] "NYC Open Data." [Online]. Available: https://nycopendata.socrata.com/. [Accessed: 1-Jan-2014].

[4] A. Fink, B. Mechtley, G. Wichern, J. Liu, H. Thornburg, A. Spanias, and G. Coleman, "Re-Sonification of Geographic Sound Activity using Acoustic, Semantic and Social Information," in *The 16th International Conference on Auditory Display (ICAD)*, 2010.

[5] I. McGregor, A. Crerar, D. Benyon, and C. Macaulay, "Soundfields and soundscapes: Reifying auditory communities," in *Proceedings of the International Conference on Auditory Display (ICAD)*, 2002.

[6] A. Valle, M. Schirosa, and V. Lombardo, "A Framework For Soundscape Analysis and Re-Synthesis," in *Proceedings of the 6th Sound and Music Computing Conference (SMC)*, 2009, pp. 23–25..

[7] D. A. Sumikawa, "Guidelines for the integration of audio cues into computer user interfaces," *Lawrence Livermore National Laboratory Technical Report, UCRL 53656*, Jun. 1985.

[8] J. McCartney, "Rethinking the Computer Music Language: SuperCollider," *Computer Music Journal*, vol. 26, no. 4, pp. 61–68, Dec. 2002.

[9] N. Collins, "SCMIR: A SuperCollider music information retrieval library," in *Proceedings of the International Computer Music Conference*, 2011, August, pp. 499–502.

[10] T. H. Park, J. Turner, M. Musick, J. H. Lee, C. Jacoby, C. Mydlarz, and J. Salamon, "Sensing Urban Soundscapes," in *Mining Urban Data (MUD) Workshop Proceedings of the EDBT/ICDT 2014 Joint Conference*, 2014.

[11] T. H. Park, B. Miller, A. Shrestha, S. Lee, J. Turner, and A. Marse, "Citygram One : Visualizing Urban Acoustic Ecology," in *Proceedings of the Conference on Digital Humanities*, 2012.

[12] T. H. Park, J. Turner, C. Jacoby, A. Marse, M. Musick, A. Kapur, and J. He, "Locative Sonification: Playing The World Through Citygram," in *Proceedings of the 2013 International Computer Music Conference (ICMC)*, 2013.

[13] A. Klein, "District Adding Gunfire Sensors," *Washington Post*, Washington D.C., 05-Jul-2008.