Safe City

Jason Li, Emma Qian, Hongsen Qin, Shu Fay Ung

3 March 2019

1 Introduction

According to Washington D.C.'s ShotSpotter program, nearly eight out of every ten shooting events in D.C. go unreported. As a response, the government uses ShotSpotter, a program that uses acoustic tracking technology to better monitor and control gun violence in cities. It was first implemented in Historic Anacostia starting in 2005, and the whole process took 2 million dollars and four years to reach completion throughout the district. Currently around 300 sensors are placed in D.C., and each time noises were suspected to be a gunshot, they are verified by humans before being passed to the Metropolitan Police Department. There is about a one minute delay for verification.

Our project proposes a crowd-sourcing solution that will not only use significantly less resources and time to implement, but also require less maintenance and likely provide a faster and more accurate response time. We spent a significant amount of time on the mathematical modeling aspect of our solution, as outlined below. Unlike ShotSpotter, our solution does not rely on new infrastructure and can be installed on users' phones easily, adding close to no overhead cost and a much faster distribution. Our app can automate the process of gunshot identification and localization, where we crowd-source the verification aspect to people near the area of interest. Under this structure, we expect to reduce the number of false positives and delay time since we will have input from many users instead of just one human and users can provide feedback at faster rates. This is because users are only asked about events that occur near them, as opposed to employees of ShotSpotter who have to respond to many events.

2 Locationing

2.1 Time-Delay Method

When a loud sound is detected at approximately the same time from multiple sensors, the sensors that detected the loud sound upload their location and the time they received the sound to the server, forming the N sensors.

We use the method described in Smith and Abel, 1987 [2]. For each sensor n, we know its location vector X_n from GPS, and the time the signal from the source was received t_n.

We take our first sensor (sensor 1) as the origin of our coordinate system. So we define $R_n = ||X_n - X_1||$, and define $d_{n,1} = v_s(t_n - t_1)$, where v_s is the velocity of sound in air.

We then define

$$\delta = \begin{bmatrix} R_2^2 - d_2^2 \\ R_3^2 - d_3^2 \\ \cdots \\ R_N^2 - d_N^2 \end{bmatrix}, d = \begin{bmatrix} d_{2,1} \\ d_{3,1} \\ \cdots \\ d_{N,1} \end{bmatrix}, \text{ and } \boldsymbol{S} = \begin{bmatrix} (x_2 - x_1) & (y_2 - y_1) & (z_2 - z_1) \\ (x_3 - x_1) & (y_3 - y_1) & (z_3 - z_1) \\ & \cdots & \\ (x_N - x_1) & (y_N - y_1) & (z_N - z_1) \end{bmatrix}$$

We define a diagonal, positive weight matrix $\boldsymbol{W}$; in this case all ones (the identity matrix) since we don't know how reliable each sensor's data is *a priori*. We also define $\boldsymbol{P}_{\frac{1}{d}} = \boldsymbol{I} - \frac{dd^T}{d^T d}$, where $\boldsymbol{I}$ is the identity matrix. $\boldsymbol{P}_{\frac{1}{d}}$ is idempotent.

Finally, we compute $\Delta x = \frac{1}{2}(\boldsymbol{S}^T \boldsymbol{P}_{\frac{1}{d}} \boldsymbol{W} \boldsymbol{P}_{\frac{1}{d}} \boldsymbol{S})^{-1} \boldsymbol{S}^T \boldsymbol{P}_{\frac{1}{d}} \boldsymbol{W} \boldsymbol{P}_{\frac{1}{d}} \delta$, which is the offset of the sound source position x_s from the location of the first receiver x_1. We compute $x_s = \Delta x + x_1$ to find the origin position of the sound of interest.

In order to make the computation robust to a possibly extraneous x_1 sensor match (as the fact that the sensor x_1 received a loud sound could be a coincidence), k other x_1s are selected randomly, and the results averaged. The l result locations that are farthest from the mean are discarded. k and l are user-configurable parameters.

The standard deviation of the predicted position $\sigma_{\hat{x_s}}$ is used to predict the accuracy of the final result. Temporal filtering is also used to reduce the occurrence of coincidences, by automatically clearing reported sensor data farther in the past than about 30 seconds ago.

The algorithm is implemented with `numpy` in Python.

2.2 "Hot-Spot" Method

In addition to the method described above, we also implement a method that considers the difference in sound amplitude (loudness) that each receiver measures. This is because the amplitude decreases as distance increases, hence receivers near the source would measure a larger amplitude compared to receivers farther from the source. By interpolating between these amplitude measurements, we could find the maximum point in the interpolation and estimate the sound source to be located at this point. This method is most adequate when there are ≥ 20 receivers to pick up the sound.

For testing, we modelled the sound amplitude coming from a source with the equation[1]:

$$\beta(R) \approx (10\text{dB}) \left[\log\left(\frac{P_0}{4\pi I_0} \right) - 2 \log(R) \right]$$

$$\begin{aligned} P_0 &= \text{ the power emitted by the source} \\ I_0 &= \text{ the reference sound intensity} \\ &= 10^{-12}\text{W/m}^2 \\ R &= \text{ the distance from the source to a point} \end{aligned}$$

Performing the interpolation with multiquadric radial basis functions (using `scipy.interpolate.rbf`), the error in predicting the source location is ≤ 0.06 for 20 receivers randomly scattered around the source. This approach was considered as secondary to the time-delay based method.

[1] See Appendix for derivation.

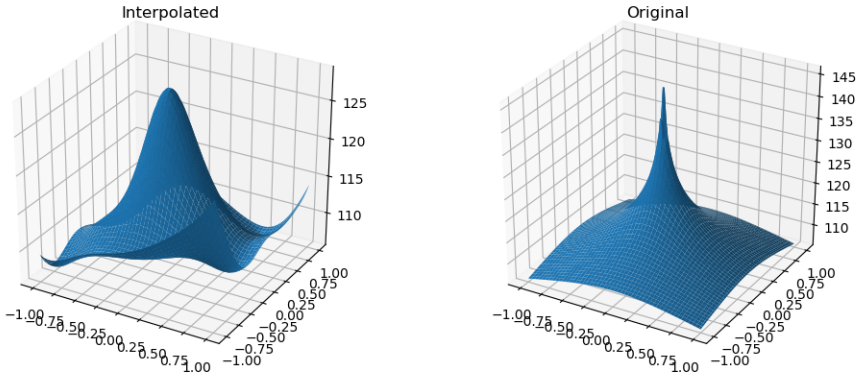

Figure 1: Plot of the interpolated $\beta(r)$ and the original $\beta(r)$ for $N = 20$. The interpolation is able to produce the location of maximum $\beta(r)$ with small error.

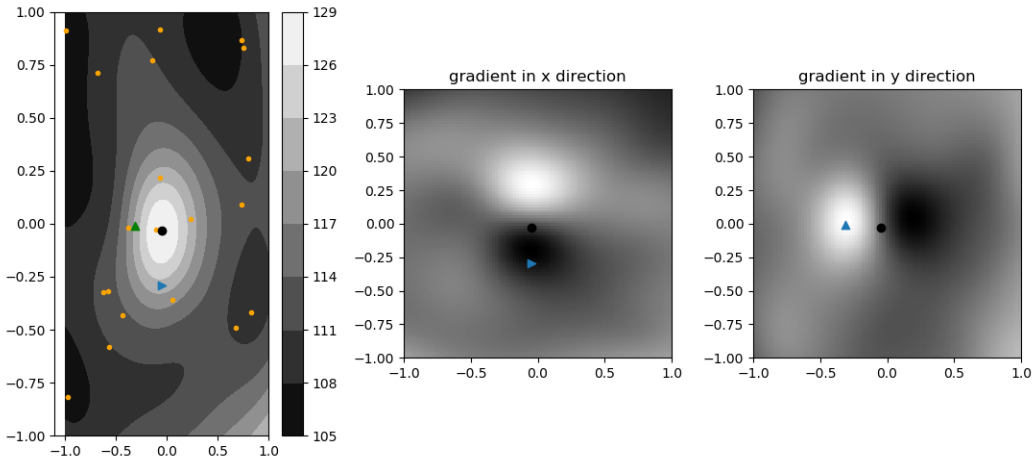

Figure 2: The left most plot shows the interpolation of random data points (in orange) taken to be the receiver locations. The black dot denotes the location of maximum amplitude; the green triangle denotes the location of maximum gradient in the x direction (g_x); and the blue triangle denotes the location of maximum gradient in the y direction (g_y). In the middle and right plots, the black dot denotes the location of maximum amplitude while the blue arrows denote the location of maximum gradient in the x and y direction respectively. We considered using g_x and g_y as well which might help improve predictions of the sound source location, but this was not implemented in the end.

3 Audio Data Analysis

3.1 Hardware Limitations

Gunshots produce sound with very large amplitudes, but phone microphones have an upper limit to the strength of the sound they can measure - above that limit, the sound will be clipped. We originally aimed to analyze features within the sound using a machine learning approach to help distinguish gunshots, but this

limit would affect the predictive power of the learning algorithm. Additionally, we were unable to obtain sufficient data sets for training (one required external approval). Thus, we decided not to continue with the ML strategy, though we describe what we attempted below.

3.2 Machine Learning

We implement a Keras and Tensorflow 5-layer CNN model described in Salamon and Bello's paper attempt to classify 10 categories of common city sounds from the publicly available Urban Sound dataset [1]. We were able to achieve an overall accuracy of 50%, which is close to the accuracy the researchers achieved. However, when we breakdown accuracy by category, we discover that the accuracy for gunshot noises is the lowest among all categories. This might be due to the fact that there are less samples of gunshots and they are easily confused with street performers, jackhammer noises, etc., which are also common city noises. On the other hand, it could also be due to the hardware limitations mentioned above.

4 Front End

In order to utilize hardware and software that we already have, and rapidly deploy our solution to affected areas, we choose to use hardware already available in most people's pockets - the smart phone. The front end was written in Java and uses android's audiosampling capability to produce a timeseries of magnitudes and geolocations. Whenever we detect large and sharp increase in audio intensity, we send the geolocation and intensity, along with the timestamp, to the server for analysis. We then get a callback if enough phones report the same sound and push this information to the mobile app so that users learn where the danger is.

5 Back End

The backend which runs the locationing code is running on a Raspberry Pi. `ngnix` faces the Web, and forwards all HTTP requests to a Tcl daemon through the FastCGI interface. The Tcl daemon in turn passes through all HTTP requests for this application to a locally-running Python server.

6 Appendix

6.1 Deriving the Model for a Sound Source

Since sound intensity is defined as the power carried by a sound wave through a unit area perpendicular to its direction of travel, we have

$$I = \frac{dP}{dA} \implies P = \int_S I \cdot dA,$$

$$I = \text{sound intensity}$$
$$P = \text{power}$$
$$A = \text{area}$$

where the integral is a surface integral over some surface S. Assuming that I is spherically symmetric and that the power emitted by the source is P_0, then at some distance r from the source, we have

$$P_0 = \int_S I \cdot dA = I \cdot 4\pi r^2 \implies I = \frac{P_0}{4\pi r^2}.$$

Now the amplitude (or loudness) of a sound is defined as $\beta = (10\text{dB})\ \log_{10}(I/I_0)$, so we have

$$\begin{aligned}\beta(r) &= (10\text{dB})\ \log_{10}\left[\frac{1}{I_0}\frac{P_0}{4\pi r^2}\right] \\ &= (10\text{dB})\ \left[\log_{10}\left(\frac{P_0}{4\pi I_0}\right) - 2\ \log_{10}(r)\right].\end{aligned}$$

References

[1] Justin Salamon and Juan P. Bello. "Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification". In: *IEEE Signal Processing Letters* (Nov. 2016). DOI: 10.1109/LSP.2017.2657381.

[2] Julius O. Smith and Jonathan S. Abel. "Closed-Form Least-Squares Source Location Estimation from Range-Difference Measurements". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 35.12 (1987). DOI: 10.1109/TASSP.1987.1165089.