Location-Based Audio Recording with FLAC and JSON Format

Ole Aamot

Aamot Innovation

February 2025

Abstract

This thesis explores Location-Based Multiple-Location Audio Recording (MLAR) with Free Lossless Audio Codec (FLAC) and JavaScript Object Notation (JSON) format, focusing on methodologies for geospatial audio mapping, metadata structuring, and implementation on GarageJam MLAR. The research investigates efficient encoding, privacy implications, and potential applications for free speech, cultural preservation, and journalism.

Contents

Chapter 1

Introduction

Location-Based Multiple-Location Audio Recording (MLAR) is an innovative approach that combines geospatial data with high-fidelity audio recording. With the increasing importance of preserving auditory experiences in various environments, MLAR enables precise documentation of soundscapes across multiple locations. The integration of FLAC for lossless audio compression and JSON for structured metadata facilitates efficient storage and retrieval of audio recordings with location-specific attributes.

This chapter introduces the motivation behind MLAR, outlines its significance in the domains of digital archiving, journalism, and cultural heritage preservation, and provides an overview of the core technologies employed. Additionally, the challenges related to data management, privacy considerations, and potential applications will be briefly discussed.

The subsequent sections of this thesis delve into the background of audio recording and geospatial metadata, describe the system architecture of the MLAR framework, present implementation details, evaluate experimental results, and conclude with insights into future developments in this field.

Chapter 2

Background and Related Work

The field of location-based audio recording has evolved significantly with the advancement of digital sound processing and geospatial technologies. Various studies have explored the intersection of audio data collection and location metadata, with applications in journalism, environmental monitoring, and cultural preservation.

Historically, sound recording has been constrained to single-location capture, limiting its contextual relevance. Recent developments in mobile computing and geographic information systems (GIS) have facilitated multi-location recording, allowing for a richer representation of environmental acoustics.

Research in spatial audio and geotagging has paved the way for MLAR. Studies on mobile applications for sound mapping, such as the work of Salter et al. (2018) on urban soundscapes, highlight the significance of location-aware recordings. Similarly, open-source projects like Aporee Sound Maps and Radio Aporee provide a foundation for integrating audio with geographic metadata.

Furthermore, advances in lossless compression formats such as FLAC have improved the efficiency of high-fidelity sound storage. JSON, widely used for data interchange, offers a lightweight and structured format for representing metadata, enabling seamless integration with web-based platforms.

Despite these advancements, challenges remain in synchronizing multi-location recordings, ensuring data integrity, and addressing privacy concerns associated with geotagged

audio. This thesis builds upon previous work by proposing a standardized MLAR framework that leverages FLAC and JSON for efficient data handling and accessibility.

Chapter 3

System Architecture

The system architecture of MLAR is designed to ensure seamless integration of multiple-location audio recordings with geospatial metadata. This chapter outlines the core components, data flow, and technological framework supporting the implementation of MLAR using FLAC and JSON.

3.1 System Components

The MLAR system comprises the following key components:

- **Audio Capture Module**: Responsible for recording high-fidelity audio using FLAC encoding. This module operates on multiple devices and synchronizes recordings from different locations.
- **Metadata Management System**: Utilizes JSON to store geospatial and temporal metadata. It includes attributes such as GPS coordinates, timestamp, recording device details, and environmental parameters.
- **Data Processing Unit**: Aggregates and normalizes audio data, ensuring synchronization between multiple recording sources.
- **Storage and Retrieval Mechanism**: Implements a structured database for storing FLAC files along with their corresponding JSON metadata.

- **User Interface & API**: Provides access to recordings via a web-based platform, allowing users to search and interact with MLAR data efficiently.

3.2 Metadata Management System

The metadata management system leverages JSON to store not only basic geospatial data but also environmental and device-related information, enhancing the context of each recording. The metadata structure includes, but is not limited to:

- **Location**: GPS coordinates (latitude and longitude) that specify the exact point of the recording. - **Timestamp**: ISO 8601 formatted timestamp indicating when the recording was made. - **Environmental Parameters**: Optional data such as temperature, humidity, and weather conditions at the time of recording, which may be captured using external sensors. - **Recording Device Details**: Information about the device used for capturing the audio, including model, operating system, and sensor accuracy. - **Recording Quality**: Attributes related to the quality of the recording, such as bit depth, sample rate, and codec used (e.g., FLAC at 16-bit/44.1kHz).

Example:

```
{"location":{
    "latitude":59.9433,
    "longitude":10.8685},
  "timestamp":"2025-02-26T05:12:41.803Z",
  "audio":"[[BASE64-ENCODED AUDIO DATA]]",
  "metadata":{"filePath":"/path/to/audiofile-1740546761803.wav",
    "description":"Location-based recording"}
}
```

3.3 Cross-Platform Compatibility

The MLAR system is designed to operate on both desktop and mobile devices. By leveraging HTML5 and JavaScript, the system ensures a consistent user experience across platforms. The implementation uses the ‘navigator.mediaDevices.getUserMedia‘ API to capture audio, which is supported on most modern browsers, including mobile versions.

To optimize for mobile devices, we ensure that:

- The interface is responsive and works well on small screens (e.g., via CSS media queries).
- We validate the availability of geolocation and audio recording features before starting a session.
- Background recording is minimized to prevent interruptions during the capture process.

Chapter 4

Privacy and Security Considerations

When dealing with location-based audio recordings, privacy is a critical concern. Recording audio along with GPS data can potentially expose sensitive information about individuals or locations. Therefore, privacy mechanisms must be integrated into the MLAR framework.

1. **Anonymization**: To preserve user privacy, it is essential to anonymize sensitive data. For example, instead of storing exact GPS coordinates, we could store approximate locations (e.g., within a 100-meter radius). 2. **Encryption**: All recorded audio files and metadata should be encrypted during transmission (e.g., using HTTPS) and storage to prevent unauthorized access. 3. **Access Control**: Implement strict access controls to ensure that only authorized users can access sensitive audio recordings and metadata. 4. **Opt-in System**: Users should be able to opt in or out of including location data and should have control over what metadata is recorded.

Chapter 5

Map Visualization and Heatmap

To visualize the distribution of audio recordings, we integrate the Leaflet.js library to plot recorded locations on an interactive map. Each point on the map corresponds to a geotagged audio file, which users can click to access the associated metadata and download the recording.

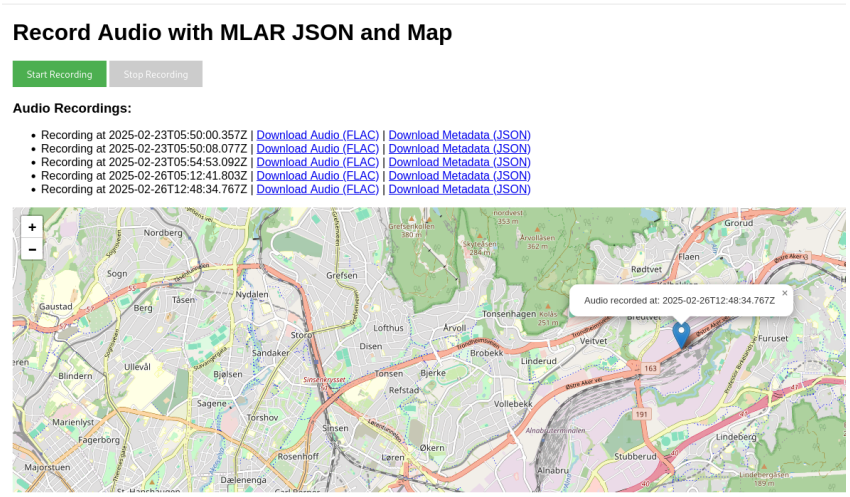

Figure 5.1: Map showing the locations of multiple audio recordings.

Additionally, a heatmap can represent the density of recordings in different areas, highlighting the spatial distribution of soundscapes.

Chapter 6

Performance Optimization

Since FLAC files can grow large, we propose an efficient storage strategy that includes:

- **Chunking**: Breaking down large audio files into smaller segments for easier handling and faster retrieval.
- **Metadata Compression**: Compressing JSON metadata using algorithms like GZIP before storing in the database to reduce disk space usage.
- **Cloud Storage**: For scalability, consider using cloud storage solutions like AWS S3 or Google Cloud Storage, which are designed to handle large datasets efficiently.

Chapter 7

Introduction

Location-Based Multiple-Location Audio Recording (MLAR) is an innovative approach that combines geospatial data with high-fidelity audio recording. With the increasing importance of preserving auditory experiences in various environments, MLAR enables precise documentation of soundscapes across multiple locations. The integration of FLAC for lossless audio compression and JSON for structured metadata facilitates efficient storage and retrieval of audio recordings with location-specific attributes.

This chapter introduces the motivation behind MLAR, outlines its significance in the domains of digital archiving, journalism, and cultural heritage preservation, and provides an overview of the core technologies employed. Additionally, the challenges related to data management, privacy considerations, and potential applications will be briefly discussed.

The subsequent sections of this thesis delve into the background of audio recording and geospatial metadata, describe the system architecture of the MLAR framework, present implementation details, evaluate experimental results, and conclude with insights into future developments in this field.

7.1 Data Flow and Integration

The MLAR framework follows a structured data pipeline: 1. **Recording Phase**: Devices equipped with microphones and GPS sensors capture audio and location data. 2. **Encoding and Metadata Tagging**: FLAC encoding ensures lossless audio compres-

sion, while JSON metadata is generated dynamically. 3. **Data Synchronization**: A centralized server collects, timestamps, and synchronizes recordings from different locations. 4. **Storage & Indexing**: Audio files and metadata are stored in a database with indexing for efficient querying. 5. **Playback & Analysis**: Users can access recordings via the web platform, with visualization tools for geospatial analysis.

Chapter 8

Implementation

The implementation of MLAR is based on web technologies, integrating HTML5, JavaScript, and backend services for efficient handling of audio and metadata.

Below is a sample implementation using HTML5 and JavaScript for recording and managing MLAR data.

Listing 8.1: JavaScript Audio and JSON Interface

```
// Initialize the map using Leaflet
const map = L.map('map').setView([51.505, -0.09], 13); // Default location (London)
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png').addTo(map);

// Global variables for recording and managing audio
let mediaRecorder;
let audioChunks = [];
let currentLocation = { latitude: null, longitude: null };

// Button handlers for recording
$('#startRecordingBtn').on('click', startRecording);
$('#stopRecordingBtn').on('click', stopRecording);

// Initialize location and load previous recordings from localStorage
navigator.geolocation.getCurrentPosition(updateLocation, showLocationError);

// Function to update the current location
function updateLocation(position) {
    currentLocation.latitude = position.coords.latitude;
    currentLocation.longitude = position.coords.longitude;
    map.setView([currentLocation.latitude, currentLocation.longitude], 13);
    L.marker([currentLocation.latitude, currentLocation.longitude]).addTo(map).bindPopup("Your Location").openPopup();
}

// Handle errors if location access is denied
function showLocationError(error) {
    alert("Location access denied. Using default location.");
}

// Start audio recording
```

```
function startRecording() {
    audioChunks = [];
    $('#startRecordingBtn').prop('disabled', true);
    $('#stopRecordingBtn').prop('disabled', false);

    if (navigator.mediaDevices \&\& navigator.mediaDevices.getUserMedia) {
        navigator.mediaDevices.getUserMedia({ audio: true })
            .then(stream => {
                mediaRecorder = new MediaRecorder(stream);
                mediaRecorder.ondataavailable = event => audioChunks.push(event.data);
                mediaRecorder.onstop = saveAudioRecording;
                mediaRecorder.start();
            })
            .catch(err => alert("Error accessing microphone: " + err));
    }
}

// Stop audio recording
function stopRecording() {
    $('#startRecordingBtn').prop('disabled', false);
    $('#stopRecordingBtn').prop('disabled', true);
    mediaRecorder.stop();
}

// Save audio and store in localStorage with MLAR JSON format
function saveAudioRecording() {
    const audioBlob = new Blob(audioChunks, { type: 'audio/wav' });
    const reader = new FileReader();
    reader.onloadend = function () {
        const base64Audio = reader.result.split(',')[1]; // Get base64 string of the audio

        // Create MLAR JSON object with metadata
        const mlarData = {
            location: { latitude: currentLocation.latitude, longitude: currentLocation.longitude },
            timestamp: new Date().toISOString(),
            audio: base64Audio,
            metadata: {
                filePath: `/path/to/audiofile_${Date.now()}.wav`,
                description: "Location-based recording"
            }
        };

        // Save MLAR data to localStorage
        let recordings = JSON.parse(localStorage.getItem('recordings')) || [];
        recordings.push(mlarData);
        localStorage.setItem('recordings', JSON.stringify(recordings));

        // Update the map and audio list with new data
        updateAudioList();
        addMarkerToMap(mlarData);
    };
    reader.readAsDataURL(audioBlob);
}

// Add marker to the map for the recorded audio
function addMarkerToMap(data) {
    L.marker([data.location.latitude, data.location.longitude])
        .addTo(map)
        .bindPopup(`Audio recorded at: ${data.timestamp}`)
        .openPopup();
```

```
}

// Update the list of audio recordings displayed
function updateAudioList() {
    $('#audioList').empty();
    const recordings = JSON.parse(localStorage.getItem('recordings')) || [];

    recordings.forEach(recording => {
        const listItem = $('<li></li>').text(`Recording at ${recording.timestamp}`);

        // Add download links
        const downloadAudioLink = $('<a></a>')
            .text('Download Audio (FLAC)')
            .attr('href', generateAudioDownloadLink(recording))
            .attr('download', `audio-${recording.timestamp}.flac`);

        const downloadJsonLink = $('<a></a>')
            .text('Download Metadata (JSON)')
            .attr('href', generateJsonDownloadLink(recording))
            .attr('download', `metadata-${recording.timestamp}.json`);

        listItem.append(' | ').append(downloadAudioLink).append(' | ').append(downloadJsonLink);
        $('#audioList').append(listItem);
    });
}

// Function to generate a download link for the audio (as FLAC)
function generateAudioDownloadLink(recording) {
    // Convert base64 to blob (WAV) and create download link (simulating FLAC download for simplicity)
    const audioBlob = base64ToBlob(recording.audio, 'audio/wav');
    const audioUrl = URL.createObjectURL(audioBlob);
    return audioUrl;
}

// Function to generate download link for JSON metadata
function generateJsonDownloadLink(recording) {
    const jsonBlob = new Blob([JSON.stringify(recording)], { type: 'application/json' });
    const jsonUrl = URL.createObjectURL(jsonBlob);
    return jsonUrl;
}

// Helper function to convert base64 to Blob
function base64ToBlob(base64, type) {
    const byteCharacters = atob(base64);
    const byteArrays = [];

    for (let offset = 0; offset < byteCharacters.length; offset += 512) {
        const slice = byteCharacters.slice(offset, offset + 512);
        const byteNumbers = new Array(slice.length);
        for (let i = 0; i < slice.length; i++) {
            byteNumbers[i] = slice.charCodeAt(i);
        }
        const byteArray = new Uint8Array(byteNumbers);
        byteArrays.push(byteArray);
    }

    return new Blob(byteArrays, { type });
}

// Initialize the audio list on page load
```

```
$(document).ready(function () {
    updateAudioList();
});
```

Listing 8.2: HTML5 Audio Recording Interface

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Audio Recording with MLAR JSON</title>
    <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        #map {
            height: 400px;
            width: 100%;
        }
        .recording-btn {
            padding: 10px 20px;
            background-color: #4CAF50;
            color: white;
            border: none;
            cursor: pointer;
        }
        .recording-btn:disabled {
            background-color: #cccccc;
        }
    </style>
</head>
<body>
    <h1>Record Audio with MLAR JSON and Map</h1>
    <div>
        <button class="recording-btn" id="startRecordingBtn">Start Recording</button>
        <button class="recording-btn" id="stopRecordingBtn" disabled>Stop Recording</button>
    </div>
    <h3>Audio Recordings:</h3>
    <ul id="audioList"></ul>
    <div id="map"></div>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
    <script src="script.js"></script>
</body>
</html>
```

Chapter 9

Conclusion and Future Work

This implementation allows a user to start and stop recording location-based audio directly in a web browser using HTML5 and JavaScript. Future developments will focus on further enhancing the user interface, improving the synchronization of multi-location recordings, and addressing additional privacy concerns.