Spectrum Analyzer

Design Document

Ashly Altman
Brooke Ebetino
Joey Thompson
Masood Afzali
Tyler Haley

Date: (09/24/2023)
Department: Computer Science
Course: SWE 4724
Professor: Dr. Huang

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

The purpose of this SDD is to provide a comprehensive understanding of the architecture and system design of the SpecVidAnalyzer application. It is intended for stakeholders to get an overview of the project's goals and scope.

1.2 Scope

The SpecVidAnalyzer is designed to analyze spectrum videos. So it will read the video files, extract the plot points for the graph and will calculate data values based on this reading. It will then log this data into a separate CSV. The goal for the project is to make an automated wave spectrum analysis. The benefits of this are that it will allow the process of wave spectrum analysis to be simplified and easier to do.

1.3 Overview

This document is made to provide an overview of the SpecVidAnalyzer software. It is organized into sections including scope, system overview, system architecture, detailed design, data design and human interface design. Each section will go into detail about the software and the components of the SpecVidAnalyzer.

1.4 Definitions and Acronyms

SpecVidAnalyzer: Software developed for real-time video spectrum analysis.

NumPy: Python library for large, multi-dimensional arrays and matrices.

OpenCV: Library for real-time computer vision applications.

2. SYSTEM OVERVIEW

The SpecVidAnalyzer is a tool for analyzing spectrum wave videos. It will process video files and output data values that can be assessed. The software will be used for data analysis which would otherwise have to be found manually. This software will help to speed up this process and allow for it to be automated.

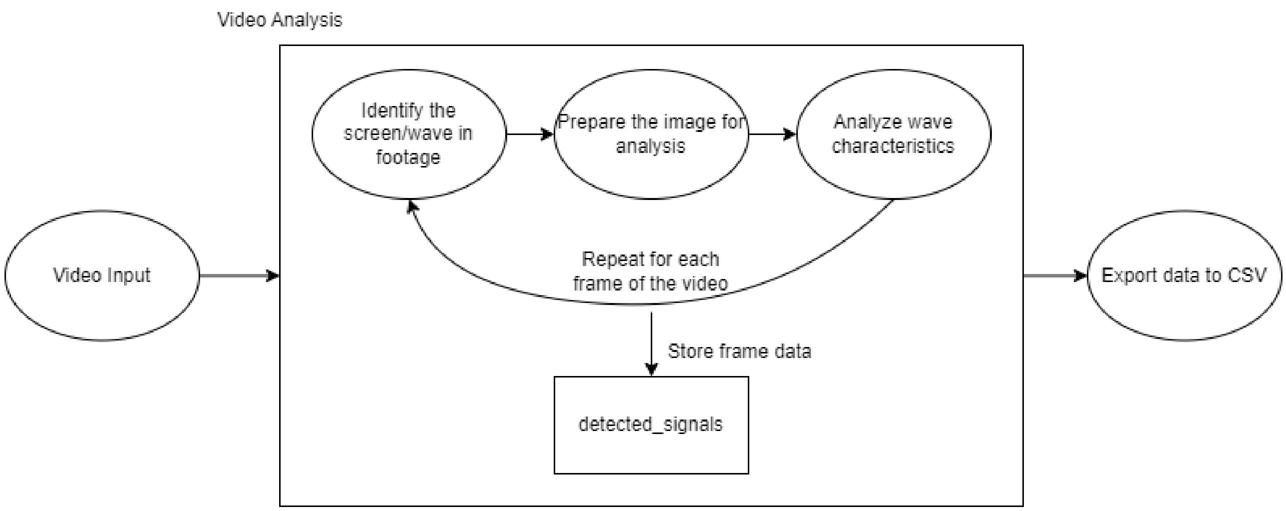
3. SYSTEM ARCHITECTURE

3.1 Architectural Design

The root goal of the application was broken down into the following tasks in order to achieve the end goal of analyzing and breaking down wave spectrum videos into numerical information:

- Input
    - Videos will be placed into a local folder where they can be accessed by the program for analysis.
- Determine video attributes
    - The video file to be analyzed will be examined by the program in order to identify the screen to be read and to collect information on characteristics of the footage before frame-by-frame analysis can begin. Characteristic information gathered on videos includes attributes such as screen color, line color, frames per second, and others.
- Prepare the frame for analysis
    - Using the video attributes previously determined by the program, a color filter is applied to the image to create a consistent image so that the wave form can be analyzed.
- Wave analysis
    - The standardized image is analyzed to give relevant statistics: center frequency, minimum amplitude, maximum amplitude, and center amplitude
- Output
    - When the program has been completed, the frame-by-frame data will be exported in a CSV format to a local folder.

Diagram 1. Depiction of program systems



3.2 Technical design

Python was selected as the language for this project because of its ease of use, developer experience with the language, and the availability of pre-existing packages that are well suited for tasks such as video processing and mathematical functions.

- OpenCV – https://opencv.org
  - A open source machine learning software library well versed in image recognition. This library consists of more than 2500 optimized algorithms for uses such as facial recognition, movement tracking, image searches, image editing, augmented reality, etc. Extensively used by researchers, companies, and governmental bodies.
- NumPy – https://numpy.org
  - An open-source project built for mathematical functions and computing using the Python language. Used in a large variety of STEM fields for processing and visualization.
- SciPy - https://scipy.org/
  - Open-source software for mathematics, engineering, and other scientific fields. A NumPy extension containing algorithms for analyzing differential equations, optimization, algebra, statistics and others.

3.3 Design Rationale

OpenCV was selected to locate the spectrum screen and prepare it for analysis due to its robust library of machine learning functions that are well trained in image recognition. NumPy and SciPy were chosen for their ease of use and quick use for mathematical work due to their excellent optimization and efficiency. All of these libraries are well respected and are often used, meaning that they can be depended on to be reliable options for the project.

A java based front end was considered, but was ultimately decided against in favor of placing videos into a local folder that the program will access upon execution to simplify the program design.

4. DETAILED DESIGN

The SpecVidAnalyzer application consists of the following key components:

- **VideoProcessor**: Responsible for reading video files and extracting frames for analysis. Key methods include:
  - readVideo(filepath): Reads a video file and returns a frame generator
  - getFrame(generator): Gets the next frame from the generator
  - release(generator): Releases the generator when done

- **ColorFilter**: Applies color thresholds to isolate areas of interest. Methods:
  - filterFrame(frame, lower, upper): Filters a frame between lower and upper bounds
  - getMask(filtered): Converts a filtered frame to a binary mask

- **ContourFinder**: Identifies contours in a binary mask. Methods:
  - findContours(mask): Finds all contours in a mask
  - getLargest(contours): Returns the largest contour by area

- **WaveAnalyzer**: Extracts characteristics from a wave contour. Methods:
  - fitCurve(contour): Fits a parabolic curve to the contour
  - getCharacteristics(curve): Calculates frequency, amplitude etc. from the curve

- **CSVExporter**: Logs analysis data to a CSV file. Methods:
  - exportRow(data): Exports a row of data
  - close(): Closes the file when done

5. DATA (DATABASE) DESIGN

The SpecVidAnalyzer application does not use a database, it exports analysis data to a CSV (comma separated values) file.

The CSV file contains the following columns:

- **Timestamp**: The video timestamp of the analyzed frame in seconds
- **Center Frequency**: The extracted center frequency of the wave
- **Min Amplitude**: The minimum amplitude of the wave
- **Max Amplitude**: The maximum amplitude of the wave
- **Center Amplitude**: The center amplitude between min and max

The CSVExporter component handles writing rows of data to the CSV file as waves are analyzed from the video. This provides a simple way to log the analysis results for external processing and visualization.

The CSV file is saved locally by default with an automatically generated filename. The user can optionally specify a custom path and filename.

There is no persistent database, each analysis starts with a new CSV file. This design is simple and sufficient for the application requirements.

6. HUMAN INTERFACE DESIGN

6.1 UI design

To use this application, the user will simply run an executable file with the desired video included in the same folder and a .csv spreadsheet file will be generated. There will be no UI for this application.

7. Requirements Matrix

Req # Unique ID	Feature	User Story	Acceptance Criteria	Current Status
RQ_01	Video Processor		-User can select desired video file for processing -The application can read the video and generate frame data	Testing
RQ_02	Color Filter		The application can use pixel color information to isolate areas of interest	Testing
RQ_03	Contour Finder		The application is able to use the binary mask created by the color filter to identify contours	Testing
RQ_04	Wave Analyzer		The application can use the characteristics of the contours identified by the contour finder to determine the measurements of the wave to then be converted into radio frequency relevant metrics	Testing
RQ_05	CSV Exporter		The user can have the application generate a csv spread sheet file with properly formatted data and measurements in terms of radio frequency metrics such as Hz so the data can be analyzed for anomalous readings	Testing