Intelligent Signal Processing. End of term assignment

Introduction

The End of term assignment for Intelligent Signal Processing consists of a series of three individual exercises. These exercises cover the last five topics of the course:

- Capturing, representing and processing camera input
- Computer vision: movement detection
- Computer vision: face detection
- Audio file formats. Compressing audio signals
- Video file formats. Compressing video signals

The exercises are strongly based on the subjects covered during the course, but also invite the student for further investigation.

It is recommended that the students carefully read all the sections of this document, both to ensure a good understanding of the coursework exercises, in addition to knowing what to submit.

Exercise 1

Description

The city of Laramie (Wyoming, US) will launch a series of initiatives in order to improve the city planning, traffic flow, public transportation and parking efficiency.

One of these initiatives consists of developing an automated traffic monitoring system using computer vision, and the major of the city has contacted you to develop this system.

First, to evaluate your skills, you are asked to perform two tasks, which consist of developing two motion detection applications.

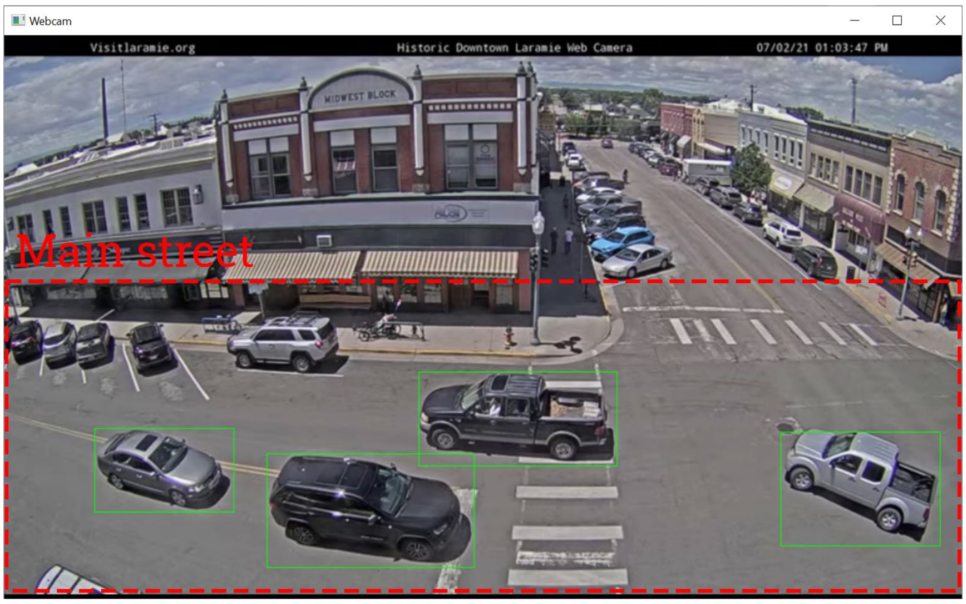
<u>Task 1</u>

The first task consists of developing an application (*exercise 1.1*) for detecting and tracking moving cars from a camera recording.

The application must run on a Jupyter notebook written in Python and use the OpenCV library.

Technically, the algorithm of the application must be based on the frame differencing and background subtraction techniques.

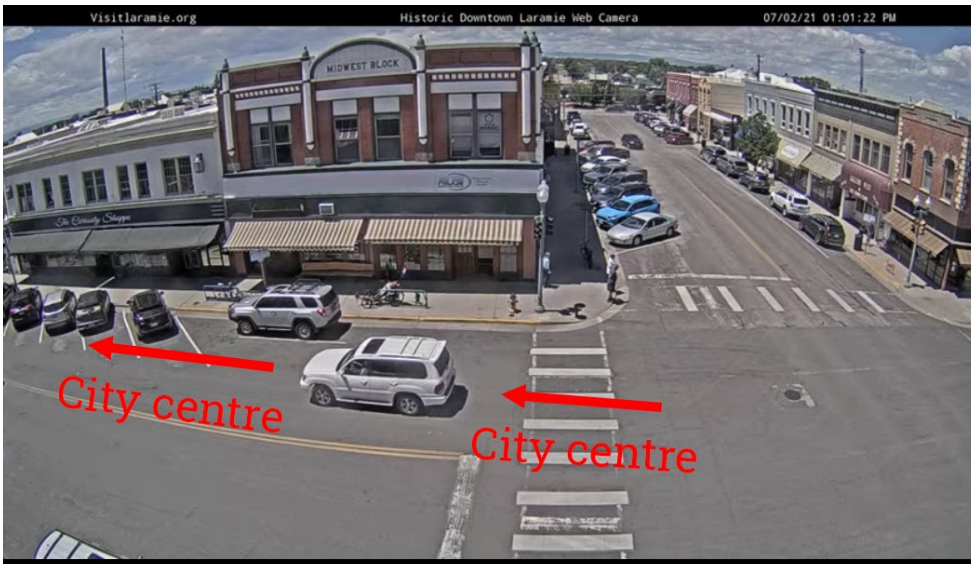
To verify that the application is operating as expected, your client has sent you a recording from one of the city traffic cameras (Traffic_Laramie_1.mp4), and the following image showing the expected results. (Note: your application can focus only on detecting cars that are in the 'main street').



Task 2

In order to prevent traffic jams, your client is also very interested in the number of cars that go from the city's downtown to the city centre in peak hours.

Therefore, Task 2 consists of developing a computer vision application (*exercise 1.2*) for counting the number of cars that go from the city's downtown to the city centre for a specific time interval.



As in Task 1, the application must run on a Jupyter notebook written in Python, use the OpenCV library, and be based on the frame differencing and background subtraction techniques.

To verify that the application works correctly, you should use the recordings Traffic_Laramie_1.mp4 and Traffic_Laramie_2.mp4 provided by the client and fill in the following table with the data generated by the application.

	Total number of cars	**Cars per minute**
Traffic_Laramie_1.mp4		
Traffic_Laramie_2.mp4		

List of deliverables

For Exercise 1, you should submit in a ZIP file:

- The Jupyter notebooks of the applications *exercise 1.1* and *exercise 1.2*.
- A written report in PDF format, approximately 500 words. This report must include:
    - The table of Task 2.
    - A brief description of the frame differencing and background subtraction techniques.
    - A brief analysis of the application.

Marking criteria

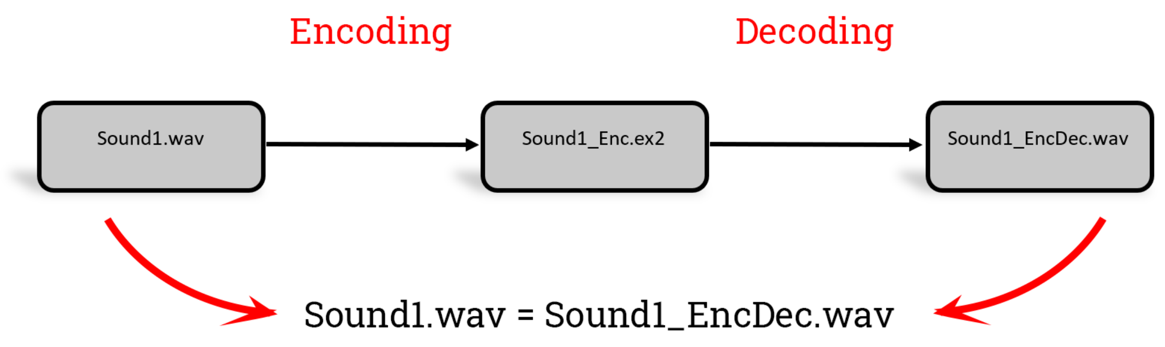
	Done?	Marks
The Jupyter notebook for *exercise 1.1* correctly detects and tracks the cars on the recording Traffic_Laramie_1.mp4.		2
The Jupyter notebook for *exercise 1.1* installs the necessary libraries for running the application (using the package-management *pip*) and includes markdown cells describing the code in detail.		1
The Jupyter notebook for *exercise 1.2* correctly counts the number of cars that go from the city's downtown to the city centre.		3
The Jupyter notebook for *exercise 1.2* installs the necessary libraries for running the application (using the package-management pip) and includes markdown cells describing the code in detail.		1
The written report includes a brief description of the frame differencing and background subtraction techniques.		2
The written report includes the table of Task 2 and a brief analysis of the application.		1
Total		10

Exercise 2

As we already know, data compression can be lossless or lossy. Lossless compression techniques involve no loss of information, while lossy compression techniques reduce file size by eliminating data in the file.

In this exercise, you are going to create your own application (*exercise 2*) for encoding and decoding a series of uncompressed audio files (WAV files). The application must run on a Jupyter notebook written in Python and be based on the lossless data compression method 'Rice coding'.

Therefore, if Sound1.wav is an uncompressed audio file, the application should be able to encode this file and save it in your drive with the name Sound1_Enc.ex2 (we can use, for example, the extension 'ex2' to identify encoded files). Then the application should be able to decode the file Sound1_Enc.ex2 and create a new WAV file, with the name Sound1_Enc_Dec.wav. As the application is based on a lossless data compression method, the files Sound1.wav and Sound1_Enc_Dec.wav should be exactly the same.



To analyse the effectiveness of the Rice coding algorithm with bit length K = 2 bits and K = 4 bits to compress the audio files Sound1.wav and Sound2.wav, fill in the following table.

	Original size	**Rice (K = 4 bits)**	**Rice (K = 2 bits)**	**% Compression (K = 4 bits)**	**% Compression (K = 2 bits)**
Sound1.wav					
Sound2.wav					

Ideas for further development:

1. Find ways to improve the compression rates obtained in the previous table. You can try to 'improve' the Rice coding algorithm you have used in your application or implement another lossless data compression method.

List of deliverables

For Exercise 2, you should submit in a ZIP file:

- The Jupyter notebook of the application *exercise 2*.
- A link to the application running in a Jupyter notebook link in Coursera.
- A written report in PDF format, approximately 500 words. This report must include:

- A brief analysis of the application.
- The requested table and an analysis of the results. In particular, justify why the compression rates of the files are so different.
- A brief description of the further development implemented.

Marking criteria

	Done?	Marks
The Jupyter notebook for *exercise 2* correctly encodes an uncompressed WAV audio file using the lossless data compression method 'Rice coding'.		2
The Jupyter notebook for exercise 2 correctly decodes a compressed 'ex2' audio file.		2
The submission includes a link to the application running in a Jupyter notebook link in Coursera. The notebook for *exercise 2* includes markdown cells describing the code in detail.		1
The written report includes the requested table and an analysis of the results (the effectiveness of the Rice coding algorithm).		2
The written report includes a brief analysis of the application.		1
The application includes further development.		2
Total		10

Exercise 3

The Narbonne Online Film Festival receives every year more than one hundred films in digital format.

The problem is that some of these films are not submitted in the format specified by the festival organisation, and they must be converted.

The organisation wants to automate this process by developing an application (*exercise 3*) for examining the format of the films and, if necessary, convert them.

You have been selected by the festival to develop such application. The application should meet the following requirements:

1. The application must run on a Jupyter notebook written in Python, use ffprobe to examine the files, and ffmpeg to convert the films.
2. From a series of video files, the application should generate a brief report in TXT indicating which films do not respect the digital format specified by the festival and what are the 'problematic' fields.
3. The application should automatically convert the submitted films with an incorrect format. The program will create a new copy of the films by adding '_formatOK' at the end of the name.
4. The format of the films specified by the festival organisation is:

- Video format (container): mp4
- Video codec: h.264
- Audio codec: aac
- Frame rate: 25 FPS
- Aspect ratio: 16:9
- Resolution: 640 x 360
- Video bit rate: 2 – 5 Mb/s
- Audio bit rate: up to 256 kb/s
- Audio channels: stereo

To verify that the application works correctly, the film festival organisation has sent you a series of film excerpts in a ZIP file (see attached file Exercise3_Films.zip).

List of deliverables

For Exercise 3, you should submit in a ZIP file:

- The Jupyter notebook of the application *exercise 3*.
- A link to the application running in a Jupyter notebook link in Coursera.
- The report in TXT that the application examining the films submitted by the organisation.
- A written report in PDF format, approximately 500 words. This report must include:
  - A brief description about how ffprobe and ffmpeg has been installed and configured in your machine.
  - A brief analysis of the application.
  - A brief description of the terms video format (container), video codec, audio codec, frame rate, aspect ratio, resolution, video bit rate, audio bit rate and audio channels.

Marking criteria

	Done?	Marks
The application *exercise 3* correctly examines the series of film excerpts provided and generates a brief report in TXT indicating which films do not respect the digital format specified by the festival and what are the ‘problematic’ fields.		3
The application *exercise 3* automatically converts to the correct format the submitted films with an incorrect format. The application creates a new copy of the films by adding ‘_formatOK’ at the end of the name.		3
The submission includes a link to the application running in a Jupyter notebook link in Coursera. The notebook for exercise 2 includes markdown cells describing the code in detail.		1
The written report includes a brief description about how ffprobe and ffmpeg has been installed and configured in your machine.		1
The written report includes a brief analysis of the application.		1
The written report includes a brief description of the requested terms.		1
Total		10