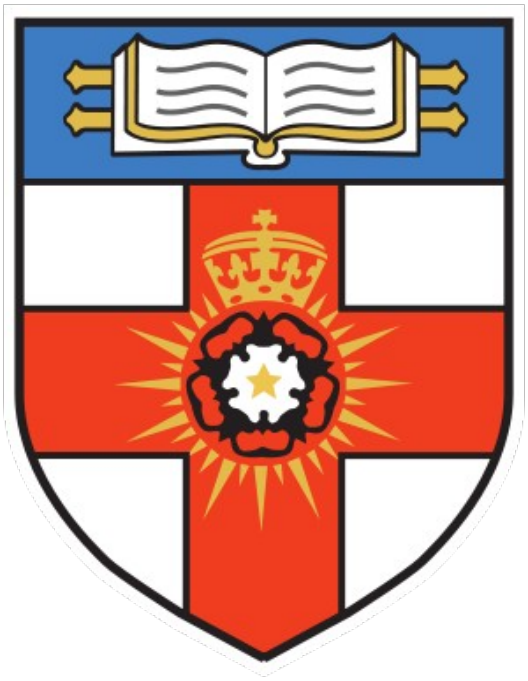
UNIVERSITY OF LONDON
INTERNATIONAL PROGRAMMES

BSc Computer Science (Machine Learning and Artificial Intelligence)



CM3065 PROJECT
ISP FINAL WRITTEN REPORT

Date of Submission: 11/03/2024

Table of Contents

1 Exercise 1

1.1 Frame Differencing Techniques

The frame differencing technique in this application involves comparing consecutive frames to detect changes, such as moving objects. This is achieved by subtracting the current frame from the previous frame to highlight the areas where motion occurs. The resulting foreground mask helps identify objects of interest, like cars, in the video stream.

1.2 Background Subtraction Techniques

The background subtraction technique in this application involves using a Gaussian blur on the frame, applying a background subtraction algorithm, thresholding the foreground mask to remove shadows, and applying morphological operations to clean up the mask. Contours of detected objects are found, filtered based on size and position, and the convex hull is used for clean contours without holes. Debug options allow for drawing contours on the original image and updating the tracker with contours.

1.3 Table of Task 2

	Total number of cars	Cars per minute
Traffic_Laramie_1.mp4	6	2.02338
Traffic_Laramie_2.mp4	4	2.27101

1.4 Analysis of the Application

The Exercise 1.1 application reads a video using cv2.VideoCapture(), then de-noises each frame with cv2.GaussianBlur(), a low-pass filter removing high-frequency parts like noise and edges. This is a common step in image processing to prep images for further work.

Each frame goes through a background subtractor created with cv2.createBackgroundSubtractorKNN(). I have chosen KNN for its noise reduction and object shape preservation. The apply() method generates a foreground mask, which is then thresholded with cv2.threshold() to create a binary image. After morphological operations with cv2.morphologyEx() to fill holes, contours of foreground objects are found with cv2.findContours(). Contours are filtered by a minimum area and removed if in the top image half. Remaining contours are processed with cv2.convexHull() for car-like shapes, enclosed in rectangles with cv2.boundingRect(), and drawn back on the original frame with cv2.

This method is efficient, straightforward, and customizable. However, it needs manual parameter adjustment for optimal outcomes. The key parameter, MIN_CONTOUR_AREA, sets the minimum

contour area for car identification. A smaller value may misclassify pedestrians and bicycles as cars, while a larger value may misclassify small cars as noise. Therefore, this parameter should be adjusted for each unique scenario.

The exercise 1.2 app uses the same background extraction as exercise 1.1. It tracks moving objects with a Tracker class, updating their coordinates based on frame contours. Object movements are predicted using a Kalman filter via cv2.KalmanFilter(). The Kalman filter is a math model using noisy measurements to estimate object coordinates over time. It predicts the next object position based on previous ones, ensuring tracking robustness by assuming objects do not vanish mid-scene. (412 WORDS)

2 Exercise 2

2.1 Jupyter Link in Coursera

https://hub.labs.coursera.org:443/connect/sharedsukdfnzv?forceRefresh=false&isLabVersioning=true

2.2 Requested Table

	Original size	Rice (K = 4 bits)	Rice (K = 2 bits)	% Compression (K = 4 bits)	% Compression (K = 2 bits)
Sound1.wav	1002088	1516265	4115718	-51.3106	-310.714
Sound2.wav	1008044	1575347	4348595	-56.2776	-331.389

2.3 Results Analysis

The encoded file is 1.5 times larger than the original with K=4 bits and 4 times larger with K=2 bits in Rice coding. The parameter K dictates how input data is divided between unary and binary parts of code words. A higher K value utilizes more bits in the binary part and fewer in the unary part. The best K value relies on the statistical characteristics of input values, like zero count and value distribution. More zeros suggest a smaller optimal K, while higher input value variance suggests a larger optimal K. This suggests that the compression efficiency varies based on the chosen K value, indicating the importance of finding the optimal K value through experimentation for the given input data.

2.4 Brief Analysis of the Application

The application uses Rice encoding and decoding with variable k. Functions rice_encode() and rice_decode() were created for message encoding and decoding. Testing confirmed the implementation's correctness, ensuring the original message is maintained after encoding and decoding with the same k. Additionally, the decoder was tested with leading zeros or ones in the message.

The application includes functions to convert a list of numbers to bits and vice versa, tested for correctness. The list of numbers was successfully converted to bits and back without alteration. The application contains a function is_files_equal() to compare two files bit by bit. It returns True if the files are the same and False if they are different. Testing confirmed it correctly identified equal and different files.

The application reads the input file as a sequence of bytes in the encode_file_rice() function, encoding the bytes using rice_encode() for each byte. The encoded messages are joined into a big string of byte strings, then converted to bytes with bits_to_bytes() and saved to the output ex2 file.

For decoding the file, the process involves reading the input file as bytes, converting them to bits with bytes_to_bits(), and then decoding the string of bits with rice_decode() for each byte. The decoded messages are then written to the output Env_Dec file. (353 WORDS)

3 Exercise 3

3.1 Jupyter Link in Coursera

https://hub.labs.coursera.org:443/connect/sharedsukdfnzv?forceRefresh=false&isLabVersioning=true

3.2 Description of ffmpeg Installation and Configuration

I am using Windows 10 to carry out this exercise. ffmpeg is downloaded from https://github.com/BtbN/Ffmpeg-Builds instead of the official website. This website consists of static builds of ffmpeg which contains all necessary code within the executable, eliminating the need for external files. It includes required shared libraries, such as libavfilter, directly in the executable, making it larger in size.

After downloading, it is extracted to the C drive for ease of navigation. Then, I add the ffmpeg build to the system path to run it from any directory on Windows 10 by simply typing "ffmpeg" instead of the full path to the binary.

3.3 Analysis of the Application

The application runs the ffprobe command on each supplied file, parsing the JSON results to ascertain the file's characteristics. Although an existing ffmpeg-python library is available for Python integration with FFmpeg, I opted for leveraging the ffprobe command by itself.

The application checks if the properties determined by ffprobe match the requirements. It prints properties that do not meet the requirements and marks the file as INVALID if any property does not match. The application uses the ffmpeg command to convert files marked as INVALID to the required format. The converted files are saved in the "results/" directory with the "_formatOK" suffix in the filename. The converted files are rechecked for validity by ffprobe using the same criteria as the original file. All said files marked as VALID.

3.4 Definitions

3.4.1 Video Format (Container)

A video format (container) is a file format that holds various types of data such as video, audio, and subtitles. It acts as a wrapper for these different elements to be stored and played back together. Examples include MP4, AVI, and MKV.

3.4.2 Video Codec

A video codec is a technology used to compress and decompress video files. The primary aim of a video codec is to decrease the file size without significantly reducing the quality of the video. Popular codecs include H.264 and HEVC (H.265) and AV1.

3.4.3 Audio Codec

An audio codec is a technology used to compress and decompress audio files. The primary aim of an audio codec is to decrease the file size without significantly reducing the quality of the audio. Lossless codecs maintain audio quality, whereas lossy codecs sacrifice quality for smaller file sizes. Popular codecs include MP3, AAC, and FLAC.

3.4.4 Frame Rate

Frame rate refers to the number of frames displayed or processed per second in a video. It is typically measured in frames per second (fps). Increasing the frame rate results in smoother motion in the video.

3.4.5 Aspect Ratio

Aspect ratio refers to the proportional relationship between the width and height of an image or screen. It is typically expressed as a ratio, such as 16:9 or 4:3. The aspect ratio of a video is typically determined by the display device's aspect ratio. For instance, modern TVs have a 16:9 aspect ratio, while old TVs have a 4:3 aspect ratio.

3.4.6 Resolution

Resolution refers to the number of pixels in each dimension that can be displayed on a screen or captured by a camera. It is typically expressed as width x height, such as 1920x1080 for Full HD or 3840x2160 for 4K. Increasing the bit rate enhances video quality but increases file size.

3.4.7 Video Bit Rate

Video bit rate refers to the amount of data transmitted per unit of time in a video file. It is typically measured in kilobits per second (kb/s) or megabits per second (Mb/s). Increasing the bit rate enhances video quality but increases file size.

3.4.8 Audio Bit Rate

The audio bit rate refers to the amount of data transmitted per unit of time in an audio file. It is typically measured in kilobits per second (kb/s). Increasing the bit rate improves audio quality but also increases file size.

3.4.9 Audio Channels

Audio channels refer to the number of separate audio signals that can be transmitted simultaneously. Common configurations include stereo (2 channels) and surround sound (5.1 channels). (680 WORDS) (TOTAL WORDS: 1445)