IC: Project 3

Universidade de Aveiro
DETI

Daniel Silva, (102537) / João Andrade, (103361)
Tiago Santos, (89356)

Contents

1 Introduction

The goal of this project is to process audio and image files, beginning with basic image manipulations using OpenCV. We then focus on compressing audio files using a Golomb code, which will be used in a subsequent lossless audio codec.

2 Part I

2.1 YUV reader

We started by creating a header file called yuv_reader.hpp, designed to read Y4M video files.

First we get the properties of the file such as resolution, frame rate, color space, interlace, and aspect ratio. To do so the constructor reads the file's header, which contains information about the video, including the properties. It looks for specific tokens like 'W' (width), 'H' (height), 'F' (frame rate), 'C' (color space), 'I' (interlace), and 'A' (aspect ratio), and if a token is found, the corresponding information is extracted and stored.

```
// get resolution and frame rate
resolution[0] = stoi(line.substr(line.find('W') + 1));
resolution[1] = stoi(line.substr(line.find('H') + 1));

String frame_rate_str = line.substr(line.find('F'));
frame_rate[0] = stoi(frame_rate_str.substr(1, frame_rate_str.find(':') - 1));
frame_rate[1] = stoi(frame_rate_str.substr(frame_rate_str.find(':') + 1, frame_rate_str.find(' ') - 1));
```

For properties, like color space, that are not specified in the header, we gave them default values. For instance, if there is no 'C' token, indicating no specified color space, the color space is set to the default value of C420. We also count the total number of frames in the video by iterating through the file and counting occurrences of the "FRAME" keyword, updating the frame_count member variable.

After extracting the data from the header, the file stream is cleared and reset to the beginning of the file to facilitate subsequent frame retrieval.

We then have the method get_frame, which makes use of the OpenCV 'Mat' class to create a matrix of pixel values for a specific frame. This method is responsible for extracting the pixel values for a specified frame. The process involves iterating through the file, line by line, searching for lines containing the "FRAME" keyword. When the desired frame is encountered, the method extracts the pixel values. For each pixel, a character is read from the file, and its value is assigned to the corresponding position in the matrix.

2.2 YUV writer

The file yuv_writer.h encapsulates a class that simplifies the process of creating Y4M video files. It achieves it by writing Y4M file headers and storing individual video frames in YUV format.

The class starts by getting the header properties from the arguments...

Similar to yuv_reader, this method writes individual video frames to the file. To do so It appends the "FRAME" header followed by the YUV frame data, obtained from the OpenCV matrix representing the frame. The pixel values are extracted and written to the file in a format suitable for YUV encoding.

3 Part II

3.1 Encoder

To encode a Y4M video file, we created the video_encoder.cpp file. This file employs both intra-frame and inter-frame coding techniques, to compress individual frames within the video. Intra-frame coding focuses on exploiting spatial redundancies within each frame independently, reducing the amount of data needed to represent the frame. While, aims to exploit temporal redundancies between consecutive frames, enhancing the compression efficiency of the video encoding process.

We start by extracting the header data from the file and writing it to the output bitstream using the BitStream object. The program then iterates through each frame in the video and applies either intra or inter-prediction, based on the command-line arguments.

The encoder follows the concept of Intra-frame and Inter-frame predictors, the former uses predictors based on pixel blocks within the same frame (spatial), the latter, however, uses predictors based on the pixel blocks of the frame before them (temporal), i.e.: frames of type intra are used as anchors allowing us to achieve more accurate predictors on a temporal dimension.

If intra-frame prediction is enabled the program employs the inter_prediction function, predicting pixel values by comparing the current frame with the previous frame.

```
1 vector<int> intra_prediction(Mat frame) {
2     vector<int> pred(frame.cols*frame.rows);
3     for (int h = 0; h < frame.rows; h++) {
4         for (int w = 0; w < frame.cols; w++) {
5             if (h==0 && w==0) pred[h*frame.cols+w] = frame.at<uchar
      >(h,w);
6             else if (h==0) pred[h*frame.cols+w] = frame.at<uchar>(h
      ,w) - (frame.at<uchar>(h,w-1)/3);
7             else if (w==0) pred[h*frame.cols+w] = frame.at<uchar>(h
      ,w) - (frame.at<uchar>(h-1,w)/3);
8             else pred[h*frame.cols+w] = frame.at<uchar>(h,w) - ((
      frame.at<uchar>(h-1,w-1)+frame.at<uchar>(h-1,w)+frame.at<uchar
      >(h,w-1))/3);
9         }
```

```
    }
    return pred;
}
```

For inter-frame prediction, the program employs the inter_prediction function, predicting pixel values by comparing the current frame with the previous frame.

```
vector<int> inter_prediction(Mat frame, Mat framePrevious) {
    vector<int> pred(frame.cols*frame.rows);
    for (int h = 0; h < frame.rows; h++) {
        for (int w = 0; w < frame.cols; w++) {
            pred[h*frame.cols+w] = frame.at<uchar>(h,w) - framePrevious.at<uchar>(h,w);
        }
    }
    return pred;
}
```

3.2 Decoder

The file video_decoder.cpp is responsible for decoding a compressed video file encoded with intra-frame and inter-frame coding. The decoding process involves reading Golomb-encoded data from the input bitstream, performing intra-frame or inter-frame prediction, and reconstructing the original video frames.

We first define two functions: one to perform intra-frame prediction and another for inter-frame prediction, both based on Golomb-decoded residuals as well as a reference block from the previous frame. Intra-prediction functions reconstruct a frame by predicting pixel values using spatial information from neighboring pixels and handling different cases based on the position of the pixel within the frame.

```
Mat intra_prediction(Golomb g, int* resolution) {
    Mat frame = Mat::zeros(resolution[1], resolution[0], CV_8UC1);
    for (int h = 0; h < resolution[1]; h++) {
        for (int w = 0; w < resolution[0]; w++) {
            int p = g.decode();
            if (h==0 && w==0) frame.at<uchar>(h,w) = p;
            else if (h==0) frame.at<uchar>(h,w) = p + (frame.at<uchar>(h,w-1)/3);
            else if (w==0) frame.at<uchar>(h,w) = p + (frame.at<uchar>(h-1,w)/3);
            else frame.at<uchar>(h,w) = p + ((frame.at<uchar>(h-1,w-1)+frame.at<uchar>(h-1,w)+frame.at<uchar>(h,w-1))/3);
        }
    }
    return frame;
}
```

The inter-prediction function also performs based on Golomb-decoded residuals and a reference block from the previous frame and reconstructs a frame by adding it to the corresponding pixels in the reference block.

```
Mat inter_prediction(Golomb g, Mat blockPrevious) {
```

```
    Mat frame = Mat::zeros(blockPrevious.rows, blockPrevious.cols,
    CV_8UC1);
    for (int h = 0; h < blockPrevious.rows; h++) {
        for (int w = 0; w < blockPrevious.cols; w++) {
            int p = g.decode();
            frame.at<uchar>(h,w) = p + blockPrevious.at<uchar>(h,w)
    ;
        }
    }
    return frame;
}
```

As for the main function, like the encoder, we start by reading the header information from the input bitstream. Then depending on the method used, intra-frame or inter-frame, we iterate through each frame in the encoded video. The type of prediction is determined by a boolean value read from the bitstream.

If using inter-frame, the program iterates through blocks in the current frame and performs inter-frame prediction by decoding Golomb-encoded residuals. Then the reconstructed block is set in the grid structure for the current frame.

The reconstructed frame is written to the output YUV file using the yuv_writer object, this entire process is repeated for all frames in the video.

3.3 Usage

Encoder:

```
../video_encoder inputFile outputFile
```

Encoder inter-frame:

```
../video_encoder -h <block heigh> -w <block width> -p <periodicity>
    inputFile outputFile
```

Decode:

```
../video_decoder inputFile outputFile
```

3.4 Results

File	Original Size (MB)	Block Size	Compressed Size (MB)
Example	8.5MB	384x288	3.4MB
		192x144	3.4MB
		96x72	3.4MB
		48x36	3.4MB
Coastguard	45.6MB	352x288	22.0MB
		176x144	21.8MB
		88x72	21.7MB
		44x36	21.7MB

Table 1: Achieved Compression

4 Part III

4.1 video_cmp.cpp

We implemented a method to calculate the Peak Signal to Noise Ratio, according to the following expressions:

$$\mathrm{PSNR} = 10\log_{10}\frac{\mathrm{A}^2}{\mathrm{e}^2} \tag{1}$$

$$\mathrm{e}^2 = \frac{1}{\mathrm{NM}}\sum_{\mathrm{r}=1}^{\mathrm{N}}\sum_{\mathrm{c}=1}^{\mathrm{M}}[f(r,c) - \tilde{f}(r,c)]^2 \tag{2}$$

```
double psnr(Mat original_frame, Mat new_frame, int max_pixel_value
    = 255)
{
    double mean_squared_error = mse(original_frame, new_frame);
    double max_pixel_value_squared = pow(max_pixel_value, 2);
    double peak_signal_to_noise_ratio = 10 * log10((
    max_pixel_value_squared) / mean_squared_error);

    return peak_signal_to_noise_ratio;
}

double mse(Mat original_frame, Mat new_frame)
{
    double mean_squared_error = 0.0;
    int height = original_frame.rows;
    int width = original_frame.cols;

    for(int i = 0; i < height; i++)
        for(int j = 0; j < width; j++)
            mean_squared_error += pow((original_frame.at<uchar>(i,
    j) - new_frame.at<uchar>(i, j)), 2);

    mean_squared_error /= (height * width);

    return mean_squared_error;
}
```

5 Conclusion

Thanks to this project, we were able to foster a better understanding of video data compression, which is a challenging matter as we are required to handle prediction on both the spatial and temporal dimensions.

We were unfortunately unable to complete all the problems for this assignment, as we couldn't implement the lossy codec in due time.

6 Group Contribution

We consider this project's development to have been shared equally amongst all members of the group.