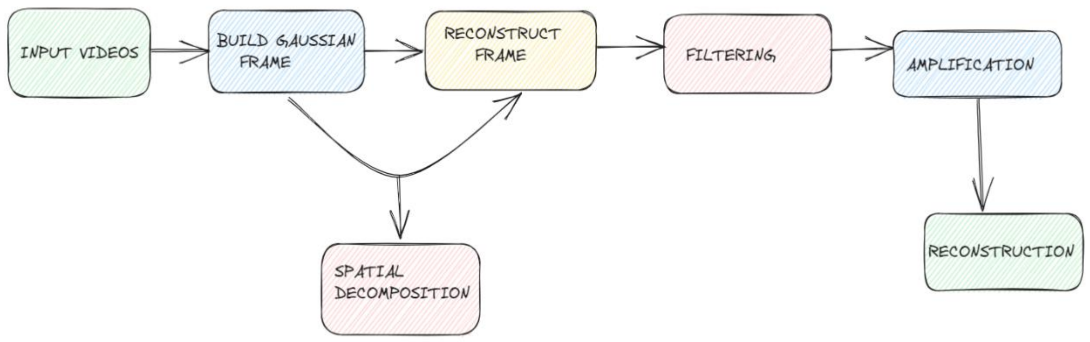
System Design Document

Heart Rate Estimation

By - Rupsa Mitra

THE EVM PIPELINE

The Eulerian Video Magnification (EVM) pipeline is a technique used to amplify



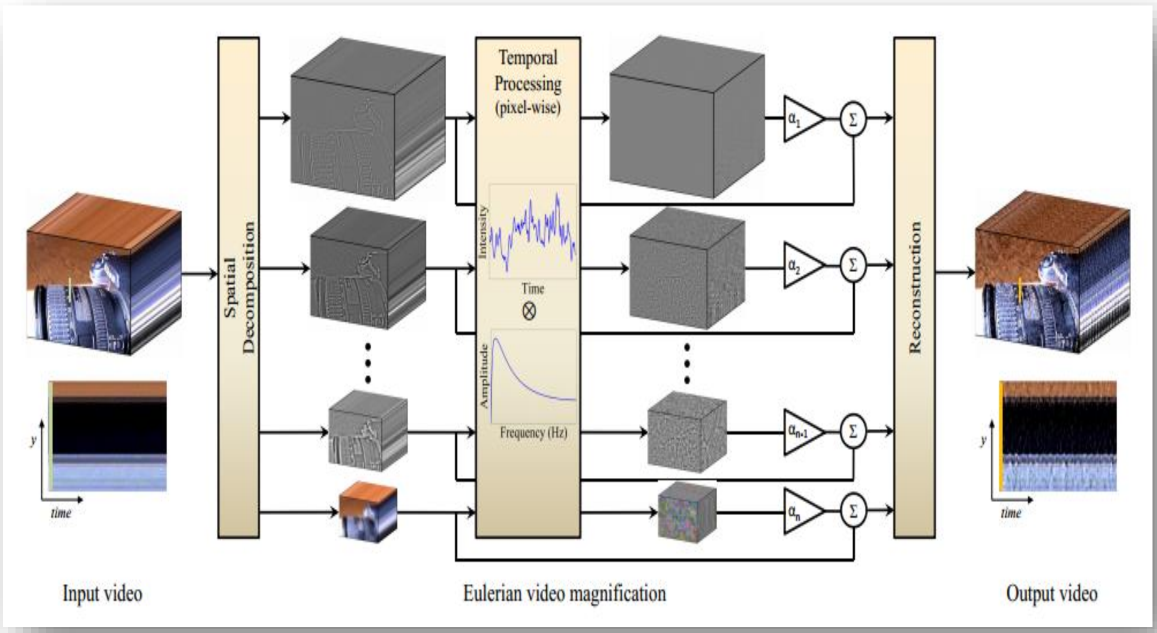
subtle temporal variations in videos, making them more visible to the human eye. It was introduced in the paper titled "Eulerian Video Magnification for Revealing Subtle Changes in the World" by MIT researchers Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand, and William Freeman.
The pipeline consists of several steps to achieve Eulerian Video Magnification:

1. Video Input
   The pipeline begins with a standard video input
2. Spatial Decomposition
   In this step, the video frames are decomposed into different spatial frequency bands. This is typically done using the Gaussian pyramid or other spatial decomposition techniques. The Gaussian pyramid breaks down the image into multiple layers, each capturing different scales of details.
3. Temporal Filtering
   Temporal filtering is applied to each layer of the spatial decomposition. The goal is to amplify or attenuate the temporal variations in the video. The filtering process accentuates the changes in the video over time, making them more pronounced.
4. Amplification
   After temporal filtering, the temporal variations are amplified. This is the key step in the Eulerian Video Magnification technique. By increasing the magnitude of the temporal variations, the subtle changes in the video become more apparent.
5. Reconstruction
   After the amplification step, the processed layers are combined using inverse spatial decomposition to reconstruct the magnified video.

An overview of how a video data is extracted, transformed, and loaded

Firstly, the video frames from the video are captured using the OpenCV library. The video frames are spatially downsampled multiple times to create a Gaussian pyramid. This pyramid represents the video frames at different spatial scales, where each level of the pyramid corresponds to a different level of detail. The Fourier Transform is applied to each level of the Gaussian pyramid in the frequency domain. Then a bandpass filter is applied to emphasize specific frequencies associated with heart rate variations (between minFrequency and maxFrequency). After filtering, the Fourier Transform is inverted to obtain the filtered frames. These frames are then amplified by the alpha value to enhance the subtle color variations.

The amplified frames are combined and reconstructed to create the magnified video. The reconstructed frames are then overlaid on the original video frames to visualize the magnification effect. This code requires the OpenCV library to work correctly, and it uses a combination of spatial and temporal filtering techniques to perform Eulerian Video Magnification and heart rate estimation. The heart rate estimation algorithm relies on the assumption that the subtle color variations captured in the video are related to blood flow changes associated with the heartbeat. However, the accuracy and robustness of this approach may vary depending on the video quality, lighting conditions, and subject's skin tone.



Potential Output

The output using the Eulerian Video Magnification pipeline is an output video that shows the real-time or recorded video from the webcam with subtle color changes amplified to reveal potential heart rate variations. The output video is created by overlaying the amplified frames on the original video frames.

Expectations from the output :

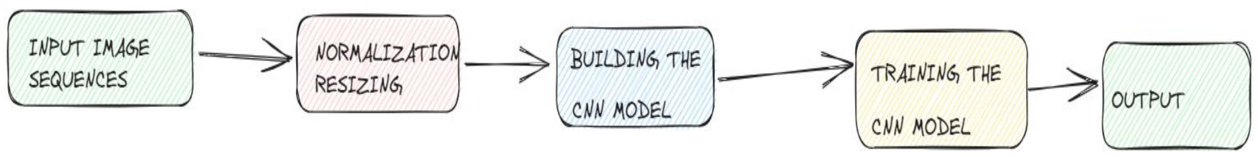
1. **Video Display**: If the script is run without a command-line argument (i.e., with no input video file specified), the output video frames will be displayed in real-time in a window titled "Detector." You will see the live webcam feed with the heart rate information overlaid on top of the video.
2. **Heart Rate Estimation**: The heart rate estimation is displayed on the video frame as "HR: [heart rate value]." The script calculates the heart rate by analyzing the frequency content of the amplified video frames. The heart rate value is updated every few seconds (controlled by bpmCalculationFrequency), and it is based on the frequency with the highest amplitude in the Fourier Transform of the amplified frames.
3. **Output Video File**: The processed video frames, with amplified color changes, are written to the output video file "output.mov." This file contains the video with the amplified effect applied.

The output video and the heart rate estimation are meant to visualize and estimate subtle variations in the color of the subject's skin, which are associated with changes in blood flow and indicative of heart rate fluctuations. The Eulerian Video Magnification technique amplifies these subtle changes, making them more visible and easier to detect.

Please note that the accuracy of the heart rate estimation heavily depends on the quality of the video, lighting conditions, and subject's skin tone. In some cases, the heart rate estimation may not be highly accurate due to various factors affecting the video quality.

To use this output for heart rate monitoring or medical purposes, it is essential to validate the results with other reliable heart rate measurement methods. This script provides a basic demonstration of Eulerian Video Magnification for heart rate estimation, but it may not be suitable for precise medical applications without further refinement and validation.

THE CNN PIPELINE



1. Preprocessing the image data
    - Load the image sequence file and read each image.
    - Resize the images to a consistent size suitable for CNN model.
    - Normalize the pixel values to a range between 0 and 1.
2. Loading the heart rate data
    - Reading the file containing the heart rate values for each image.
    - Converting the heart rate values into a format suitable for training.
3. Splitting the data into training and testing sets
    - Dividing the image data and corresponding heart rate values into training and testing sets.
4. Building the CNN model
    - Defining a CNN architecture suitable for heart rate estimation.
    - Using convolutional layers, pooling layers, and fully connected layers.
    - Choosing an appropriate activation function for each layer.
    - Considering adding dropout or regularization techniques to prevent overfitting.
5. Training the model:
    - Compiling the model by specifying an optimizer and a loss function.
    - Training the model using the training data.
6. Evaluate the model:
    - Once the model is trained, evaluate its performance on the testing data.
    - Calculate metrics such as root mean squared error (RMSE), mean absolute error (MAE) and R2 squared for heart rate estimation.

An overview of how image sequence is extracted, transformed, and loaded

1. **Extract Image Sequences**: An image sequence consists of a series of consecutive images, usually captured at a regular interval. In a video, each frame can be considered as an individual image in the sequence. To train a CNN on image sequences, we need to gather a set of such sequences, each representing a specific activity or event.
2. **Preprocess the Images**: Before feeding the image sequences to the CNN model, it's essential to preprocess the images to ensure they are in a suitable format for training. Common preprocessing steps include resizing the images to a fixed size, normalizing pixel values to a range of 0 to 1, and potentially applying other transformations like data augmentation to increase the diversity of the training data.
3. **Data Loading**: The pre-processed image sequences need to be loaded into memory in a format that the CNN model can use. For this purpose, you can create a NumPy array or TensorFlow Dataset object that holds the image data and corresponding labels .The image data will be the input to the CNN model, while the heart rate values will serve as the target variable for training the model.

Overall, to train a CNN on image sequences, we need to ensure that the data preparation steps handle the sequences correctly and that the model architecture is suitable for processing sequences of images. Additionally, adjusting hyperparameters, such as learning rate and number of epochs, may be necessary to optimize the model's performance.

Potential Output

1. The model's output, i.e., the predicted heart rates, can be used for various purposes, such as health monitoring, fitness tracking, or medical diagnostics.
2. In a real-world scenario, we would have new, unseen image sequences (not included in the training or testing sets) for which we can use the trained model to estimate heart rates.
3. By comparing the predicted heart rates with the ground truth (actual heart rates) for new data, we can assess how well the model generalizes to unseen samples and make adjustments if necessary.