INTRODUCTION

The goal is to offer an alternative to the conventional contact-based lie detection system by using contactless video analysis. This approach offers several advantages:
I)No court order required.
II) Could be applied to real time as well as pre recorded videos.
III) Can be applied without the knowledge of the observed person
The setup incorporates the "Eulerian Video Magnification" approach published by MIT CSAIL[1] that can be used for motion and color amplification of video data. Here, we visualize the amplification of temporal color variations and low-amplitude motion with this technique without the need for image segmentation or optical flow computation.

REQUIRED DEPENDENCIES

1.Our source code has two modes :- a)Library Mode(as we foundly call it "Old School" mode) and b)Non-Library Mode(we call this "Gen X").

a) Library Mode : This is our default mode .
(i) This mode uses Open CV version 2.8.12(a native C library) to implement the basic EVM. This right now is significantly faster as compared to its counterpart as it uses native C code present in Open CV. To operate in this mode you have to install Open CV in your machine. Installation steps varies for different operating system(for details visit http://docs.opencv.org/).

(ii) Xuggle API for Video processing. The API can be found our on the lib folder of our source code and is platform independent. This API is used to convert a real time video to indivisual image frames.

b)Non Library Mode : Our source code has all the functionality to work without Open CV. This mode appears to be slow compared to the Library Mode but nevertheless provides the correct output. To switch to this mode, one has to call the Java methods instead of the Open CV module. Steps involved to invoke this mode is defined in later section.

2. Java needs to be installed in your machine for the code to work.

ALGORITHM

The algorithm is divided into three parts : a)Basic EVM ; b)Color Amplification(Pulse Detection) and c) Motion Amplification

A) Basic EVM

1.We first take the video signal as our input parameter.

2.Using a pre-defined frame per second value (in our case it is 30, defined in the property file), we segregate the video signal to multiple frames.

3.Each frame is a image file.

5.We keep all the frames in a buffer and process each frame individually.

6.These steps are repeated for each and every frame: -

i) We take the image, pass it though a smoothening filter (which causes a blur effect) and downsize it to a smaller image (by discarding the all the even/odd rows and columns). Smaller images are taken to make it less computationally intensive. The degree of downsizing is defined by the parameter BLUR_LEVEL present in our property file.

ii) Convert the images from space domain to time/frequency domain by applying Fourier transformation. While converting the images to frequency domain take all the images present in the buffer. The data on which FT has to be applied is: - each channel of each pixel at a particular location of all the frames. Once FT applied to all the frames we get the frequency component of each pixel/channel.

iii) We filter our frequency component that we desire. The value of the desired frequency band is amplified to our system from our property file.

iv) Perform an inverse FT on the image and convert it back from time/frequency domain to space domain.

v) Up size the image by the same BLUR_LEVEL that you used to downsize it.

vi) Add the image back with the original frame.

vii) All the frequency components remain the same except the amplified desired frequency.

B) COLOR AMPLIFICATION (Pulse Detection)

1.Color Amplification is achieved by applying Gaussian Filter kernel during the above mentioned “Basic EVM” algorithm.

2.To detect pulse from the color amplified video frames follow the steps below :

i) Take the color amplified video signal as input.
ii) Find out the peak of the intensities in the signal.
iii) Calculate the frequency of the peak intensity. This is the BPM.

C) MOTION AMPLIFICATION
---

1. Motion Amplification is achieved by applying Laplacian Filter kernel during the above mentioned “Basic EVM” algorithm.

IMPLEMENTATION

The source code is implemented in java. The design framework followed is a MVC (Model-View-Controller) architecture.

A) MODEL: Consist of the data structure used to achieve design goal. The model is neatly segregated into substructures such as : Base Model , Amplification, Matrix Operation, Spatial Decomposition and Temporal Filtering.

I) Base Model: It has the Color Amplification and Motion Amplification Files, which implement the above-mentioned algorithm.

II) Matrix Operation: It has the files to perform matrix related operations such as subtraction, correlation, addition, multiplication and convolution.

III)Spatial Decomposition :- It has Gaussian and Laplacian implementations as java class files. It is used to reduce and expand the image frames.

IV)Temporal Filtering :- It is by far the most important and complex module of this project. It contains files for FastFourierTransformation(to convert images to frequency domain), FrequencyFilter(to filter out unwanted frequency,which uses ideal bandpass filter) and ComplexNumber(to denote real and imaginary part of Fourier Transformation)

V)Amplification :- This module has files for Amplification(which amplifies the desired frequency after temporal filtering) and ScaleImage(which expands the reduced images).

B) VIEW : The view is a in build Jframe API for java.

C) CONTROLLER : The controller is used to delegate resposibilty to different modules. It acts as the middle man between the Model and View.

CODE ANALYSIS

The “Eulerian Video Magnification” algorithm consists of two sequential filtering steps: spatial filtering is applied to the individual frames of the video to suppress unwanted high-frequency components ; subsequently, the development of each pixel is analyzed over time and movements/color variations of interest can be amplified via different filtering techniques.

Gaussian Pyramid

The first step in augmenting a video is to compute a Gaussian pyramid for every single frame. We implement this method from scratch. This is accomplished by building a Gaussian pyramid with each layer size decreasing by a factor of 2.

Laplacian Pyramid

This is accomplished by reusing the Gaussian pyramid module. A Laplacian pyramid is constructed by subtracting a new pyramid (composing of the original Gaussian with each layer doubled in size) from original Gaussian pyramid. Before we build the pyramid we convert video frame from RGB to NTSC. In order to choose pyramid layer automatically, we choose layer size to be round(log(min(w, h))-3), where w, h are width and height of frame respectively.

Temporal Filtering

We consider the time series corresponding the value of a pixel in the same position of every frame in one layer. Then loop it every layer. We convert this time series to the frequency domain using fft (Fast Fourier Transform) function and then apply a band pass filter to this signal. The last step is to convert it back using ifft. We used the Ideal band pass filter for Color amplification and IIR filter for Motion Amplification. The frequencies taken for the above mentioned code is as per MIT Research paper and is defined in “Config.properties ” file

Pixel Change Magnification

We used the parameters mentioned in the paper. And used a multi-scale lambda to control alpha in each step;The deeper the layer, the current alpha is bigger, if it is smaller than alpha, then use current alpha instead of alpha. Main formula for motion amplification is:(1+alpha) * delta < lambda/8.

Image Reconstruction

After amplify the signals, all that left is to collapse the Laplacian/Gaussian pyramid into a single image per frame. Since we didn't change in the last Gaussian layer, we need to add Laplacian results back to the original video for motion amplification. Don't forget to convert it back into RGB space.

Pulse Detection

To detect pulse we take the color amplified video signal and count the frequency of the highest peak in the matrix. The peak frequency is then used to calculate the BPM(Beats per minute) of the signal.

Switch Modes

As mentioned above, there are two modes. Library mode is petty straight forward. To use non-library mode all the methods are defined in there respective Java file according to the algorithm present above. The code is neatly divided into substructures and it is quite obvious as to which function to call. This will

decrease the speed by a factor of 20. But before doing that comment out all the Open CV calls apart from face detection.

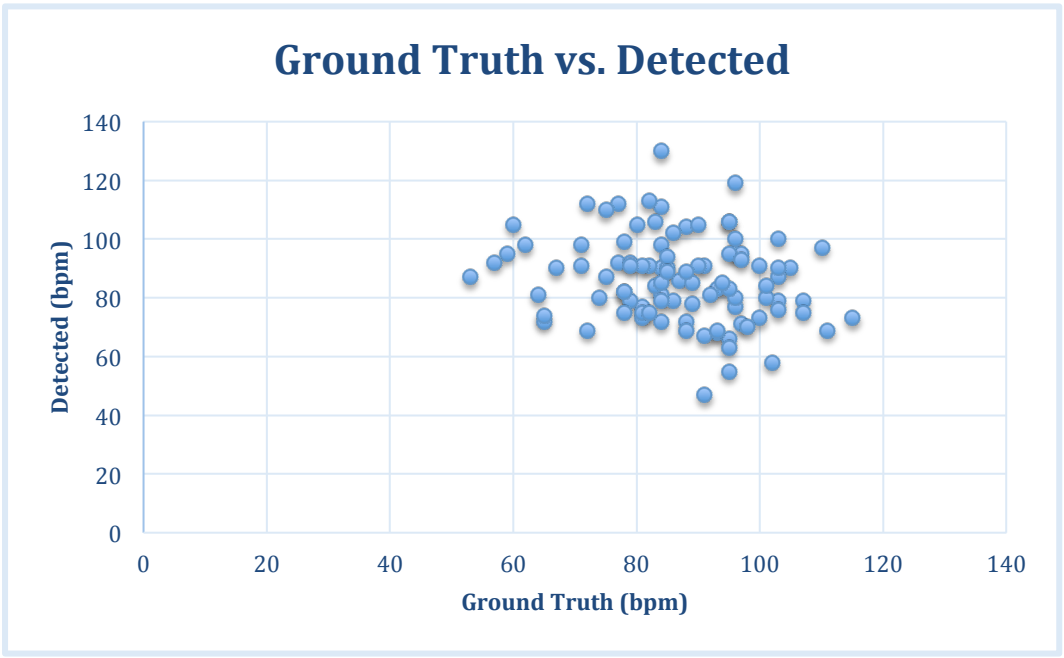
RESULT

The result that we achieve for pulse detection is mentioned in the table below

ID	Ground Truth (bpm)	Detected (bpm)
Subject1	77	92
Subject2	86	79
Subject3	84	90
Subject4	77	112
Subject5	75	110
Subject6	95	66
Subject7	93	83
Subject8	74	80
Subject9	100	91
Subject10	93	68
Subject11	95	106
Subject12	84	98
Subject13	84	81
Subject14	65	72
Subject15	103	79
Subject16	82	91
Subject17	91	91
Subject18	96	95
Subject19	81	77
Subject20	87	86
Subject21	78	82
Subject22	103	76
Subject23	71	98
Subject24	105	90
Subject25	101	80
Subject26	107	79
Subject27	89	85
Subject28	107	75
Subject29	84	72
Subject30	83	84
Subject31	79	79
Subject32	92	81
Subject33	79	92
Subject34	84	111
Subject35	81	73

Subject36	82	113
Subject37	53	87
Subject38	81	75
Subject39	59	95
Subject40	90	91
Subject41	96	77
Subject42	75	87
Subject43	95	63
Subject44	91	47
Subject45	85	90
Subject46	111	69
Subject47	97	95
Subject48	85	94
Subject49	96	100
Subject50	95	95
Subject51	110	97
Subject52	64	81
Subject53	96	119
Subject54	88	89
Subject55	71	91
Subject56	60	105
Subject57	72	69
Subject58	84	130
Subject59	91	67
Subject60	95	106
Subject61	80	105
Subject62	82	75
Subject63	95	55
Subject64	97	71
Subject65	103	87
Subject66	57	92
Subject67	67	90
Subject68	89	78
Subject69	101	84
Subject70	72	112
Subject71	81	91
Subject72	78	82
Subject73	84	85
Subject74	78	99
Subject75	96	80
Subject76	62	98
Subject77	115	73

Subject78	65	74
Subject79	100	73
Subject80	102	58
Subject81	83	106
Subject82	88	104
Subject83	78	75
Subject84	97	93
Subject85	90	105
Subject86	88	72
Subject87	98	70
Subject88	84	79
Subject89	103	100
Subject90	85	89
Subject91	95	83
Subject92	86	102
Subject93	93	69
Subject94	88	69
Subject95	94	85
Subject96	79	91
Subject97	103	90



CONCLUSION

The result appears to be near about correct with proper lighting conditions and proper motion less video of the person but fails miserably in poor lighting condition and when there is motion in the input video signal. So to test this application: Take properly illuminated video of the person with least motion.

HOW TO RUN THE CODE

To run the code call the Controller.java file from your terminal/command prompt. Pass the absolute path of your video file along with the operation you want to perform. An example is mentioned below :-

>java Controller sameedha.mov pulse
>java Controller sameedha.mov magnify

The first argument is your video file path and the second argument is the operation type. The first and second arguments are separated by space.

FUTURE WORK

I) Currently we are implementing neural network with our own face detection mechanism to create an emotion detection system which could be used to assist in lie detection.

II) Once we create those two modules we can remove Open CV and switch to a Non-Library mode framework.

REFERENCES

1.http://abcnews.go.com/US/story?id=92847&page=1
2.http://www.businessinsider.com/how-to-pass-a-polygraph-test-2015-5
3.http://cs231n.stanford.edu/reports2016/022_Report.pdf
4.http://www.paulekman.com/micro-expression-training/
5.http://people.csail.mit.edu/mrub/vidmag/#publications
6.http://users.wpi.edu/~mborowski/summer/42.world.movement.pdf
7.http://arxiv.org/pdf/1511.00423v1.pdf
8.http://bth.diva-portal.org/smash/get/diva2:830774/FULLTEXT01.pdf

ACKNOWLEDGEMENT

1.Prof David Crandall5 and course staff [CSCI-B657] of Spring 2016
2.Venu Gopal Puripanda6 et. al. from BTH Sweden for his thesis report [8]