Multi-Person Non-Contact Heart Rate Tracking Using Real-Time Eulerian Video Magnification

Michael Whitney and Andrew Carr
Brigham Young University

Abstract

We present a synthesis of deep learning facial detection and dynamic eulerian video magnification (EVM) to extract heart rate information from multiple people in a pseudo real-time manner. We present a desktop application that utilizes a simple web camera to acquire video, a deep residual network to detect and extract faces, and an EVM implementation that works together to extract and display heart rate information of every detected person in the scene.

1 Introduction

A human heart rate is a key indicator of health. Tracking it can help people to maintain a healthy lifestyle. However, most devices used to do this tracking require the user to make direct contact with them such as various forms of wearable technology—especially a wristband of some sort. We explore the possibility of a contact free heart rate tracker using a combination of techniques.

We use a technique called Eulerian video magnification (EVM) [1], to extract heart rates from videos of single people. Coupling this with facial detection, we are able to isolated faces in a video to extend our technique to multiple people at once. With facial recognition we associate each face we detect with their respective heart rates. We combine these techniques into a cross platform application. With some fine tuned optimization we are able to perform the entire process in pseudo real-time.

2 Eulerian Video Magnification

Many scenes in the world that appear to be static, actually contain movement that is undetectable by the human eye. One common example of this is a video of people that are not consciously moving. Although the people are not intending to move, their bodies are operating behind the scenes. For example, the color of people's face appear to become more red while blood flows through their head and less red when it moves out. This color change cannot be seen by basic human vision or even a simple camera. However, with EVM [1], these undetectable color changes are revealed.

Magnifying imperceptible color changes in video involves treating each pixel as a signal through time. These signals are then converted to the Fourier domain using the fast Fourier transform (FFT). In the Fourier domain, an ideal filter is applied to isolate a desired band of frequencies. The amplitude of these frequencies are increased and the signals are reverted back to the temporal domain using the inverse fast Fourier transform (IFFT). The resulting video will then reveal these unseen changes in color as seen in figure (1).

With the video's color magnified, we end up with a video where one can see the slight color change in skin tone due to the person's pulse. The revealed color changes can then be tracked to allow heart rate to be taken with high accuracy. Therefore, given an input video segment, using EVM, we can extract a person's heart rate.

650R Computer Vision (BYU 2019), Provo, UT.

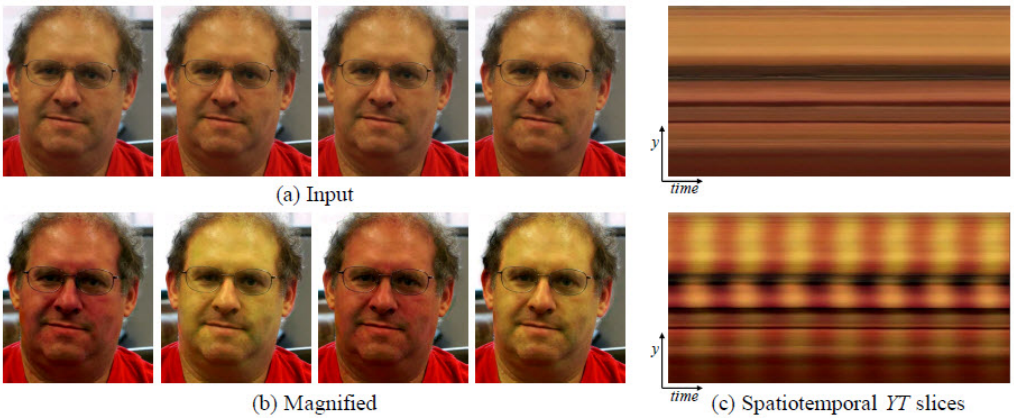

Figure 1: Example of Eulerian Video Magnification on a single male face. [1]

3 Facial Recognition with Deep Residual Networks

In traditional deep learning, and through the early 80's and 90's, most neural networks were six or fewer layers deep, with relatively few parameters. As these networks grew deeper, they were significantly harder to train. In [2] a new paradigm was developed that allowed networks to be significantly deeper, which over-parameterization increased capacity and learning potential.

The key paradigm shift that was introduced in Residual networks is that of the skip connection. The skip connection, formally defined in equation (1), intuitively introduced a gradient path for the flow of information through the network.

$$h_{i+1} = f(h_i) + h_i \tag{1}$$

Here, h is the activation of a hidden layer in the network and f is a parameterized non-linearity. Graphically, a layer in a deep resnet is depicted in figure (2).

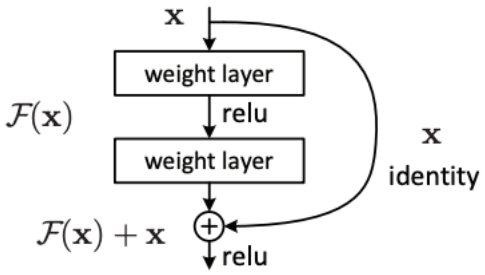

Figure 2: Deep Residual Layer with Skip Connection

These layers are stacked together to drastically increase the performance and trainability of a deep neural network.

4 Implementation

Implementation for this application involved using a combination of several components in the Python3 language. These components, which will be explained below, include: an EVM package, an open-source cross-platform graphical user interface framework called Kivy, the open-source computer vision library OpenCV, and a facial recognition package.

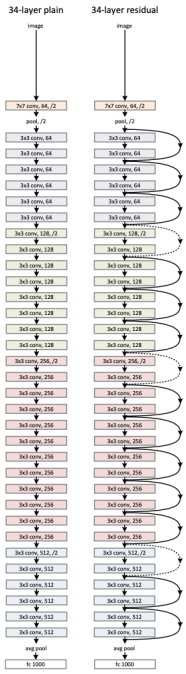

Figure 3: Full Residual Network, composed of single deep residual layers

4.1 EVM

To implement EVM, we took inspiration from the author's provided Matlab code and an open-source Python implementation from github. Ultimately though, we ended up implementing our own Python package called evm. The code allows for both magnification of color and motion, but for our purposes, we just focused on the former. We analyze the magnified queue of frames from the webcam in the frequency domain according to [3] in order to extract the heart rate see figure (4).

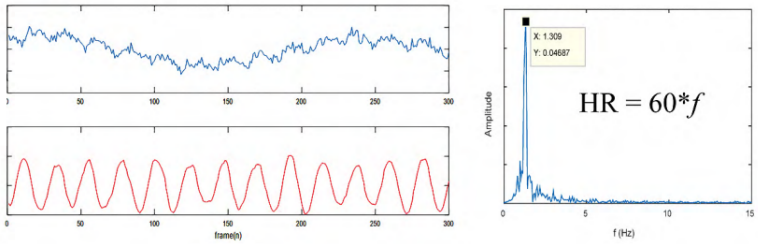

Figure 4: MVSGC smoothing for improved signal quality [3]

4.2 Kivy

Kivy is an open source, cross platform, framework that provides an API for building GUI applications in Python. There are several advantages to such a framework. In our case, we wanted a simple and extensible application that could be ported to a number of devices. Kivy even supports mobile deployment which would be an interesting avenue for future research. Kivy operates by allowing you to create a single application class. This is valuable because of its modular nature. We can integrate this with our facial recognition pipeline, computer vision capturing, frame processing, and display of heart rate. All in a single, modular, code base.

```
from kivy.app import App
from kivy.uix.widget import Widget
```

```
class HeartRateApplication(Widget):
    pass


class HeartRateFactory(App):
    def build(self):
        return HeartRateApplication()


if __name__ == '__main__':
    HeartRateFactory().run()
```

4.3 OpenCV

As described in [4] "OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products."

We make significant use of this library for common computer vision transformations in our MPHRT system. Primarily we use OpenCV to capture frames from the web camera, rotate, color transform, crop, and display text. The ability to draw directly on the frames is extremely useful and makes OpenCV an integral part of our project.

4.4 Facial Recognition

The main novelty of our method is the insight to pair facial recognition with EVM. Facial recognition allows us to extract a small subsection of the image for analysis. By using facial recognition (under the assumption it will only recognize faces), we can extract a small crop[1] and run EVM. In figure (5) we see an example of our facial recognition and EVM running on a single face.

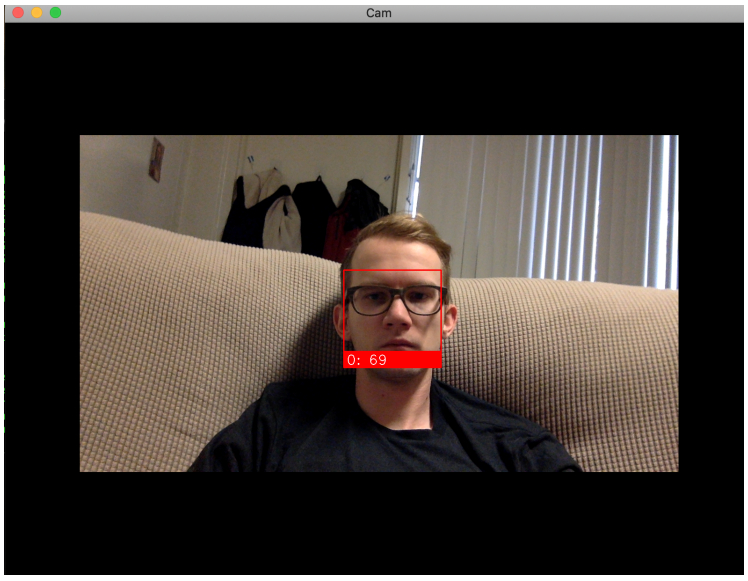

Figure 5: Example application with single person HR detection

Interestingly, and slightly surprisingly, we found that even under large motions (e.g., swaying back and forth) the heart rate remains constant and accurate. This is due, in part, to the ability of the

[1] We use an 80x80 crop of the recognized face

resnet's internal translation invariance to detect a face in multiple poses. By leveraging this ability, we are able to crop the image and run EVM in pseudo real time on multiple faces.

We leveraged an existing Python package that uses dlib's facial recognition technology written in c++. This allowed us to prototype rapidly without having to wait for extensive training time. Similarly, any improvements in the overall facial recognition (from academic research) could be leveraged soon after based on the open source contributions to the dlib project. There was some fine tuning that we had to do based on web camera focal information, but once that was in place the facial recognition worked smoothly. This was also a great opportunity to utilize more open source technology, and be exposed to pretrained deep learning models.

5 Results

Here is one of the main work horse functions of our algorithm. We take in a tensor of video frames, smooth the signal using the MVSGC method. We then extract the frequencies using the Fast Fourier Transform. After filtering the data to remove unwanted noise frequencies, we return the beats per minute (which is our heart rate information).

```
def find_heart_rate(vid, times, fps, low, high, levels=3, alpha=20):
    res = magnify_color(vid, fps, low, high, levels, alpha)
    num_frames = vid.shape[0]

    # partitions = np.split(res, 5)

    true_fps = num_frames / (times[-1] - times[0])

    avg = np.mean(res, axis=(1, 2, 3))
    even_times = np.linspace(times[0], times[-1], num_frames)

    processed = detrend(avg)#detrend the signal to avoid interference of light
        change
    interpolated = np.interp(even_times, times, processed) #interpolation by 1
    interpolated = np.hamming(num_frames) * interpolated#make the signal become
        more periodic (advoid spectral leakage)
    norm = interpolated/np.linalg.norm(interpolated)
    raw = np.fft.rfft(norm*30)

    freqs = float(true_fps) / num_frames * np.arange(num_frames / 2 + 1)
    freqs_ = 60. * freqs

    fft = np.abs(raw)**2#get amplitude spectrum

    idx = np.where((freqs_ > 50) & (freqs_ < 180))#the range of frequency that HR
        is supposed to be within
    pruned = fft[idx]
    pfreq = freqs_[idx]

    freqs = pfreq
    fft = pruned

    idx2 = np.argmax(pruned)#max in the range can be HR

    bpm = freqs[idx2]
    return bpm
```

There are several parameters that need to be tuned for our method to be effective. However, with minimal tuning, we found it to be surprisingly consistent as shown in figure (6). We notice that, while it is consistently reporting lower heart rate information, it is able to follow the trend when the subject raises or lowers their heart rate (through pushups or meditation).

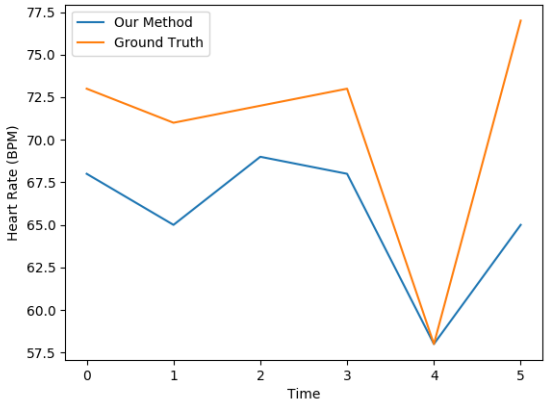

Figure 6: Accuracy of our method compare against ground truth heart rate

6 Discussion

6.1 Added Creativity

While we leverage many existing ideas and implementations. The use of facial recognition to segment the image is, to our knowledge, a novel idea for heart rate tracking. Additionally, by making our method cross platform we can potentially deploy this software to a myriad of devices.

6.2 Implementation Gotchas

There were several small decisions that needed to be made during the development of this tool. Firstly, we had to decide how many frames to processes and when to discard previous frames. This turned out to be harder than anticipated and required some data structure wizardry.

Additionally, we had an MVP working fairly consistently for a single person. We thought it would be relatively simple to extend to multiple people. However, it turned out that when we tried that naively, everyone shared heart rate information and it was a mess. It felt like the weirdest race condition. It was tricky to extend our MVP to multiple faces, but it worked out well in the end.

Also, we naively followed the work in the original EVM paper for the processing pipeline after magnifying the color. However, the signal was much too noisy. We spent a large amount of time tuning hyper parameters before realizing the source of our problems. After digging into the MVSGC paper, we were able to properly smooth the signal and see much better results.

6.3 Extensions and Improvements

There are many directions for future work in the case of multi person no contact heart rate detection. From a user experience side, it would be interesting to improve the facial recognition technology to give it more stability. Similarly, a visualization of heart rate in a wave form would be a lovely extension. Or the addition of interactive buttons to stop and start heart rate tracking.

Also, there is some hyper parameter tuning that could be done to improve the color magnification. However, more interestingly, we could potentially build a dynamic hyper parameter tuner that uses keys about lighting and position to infer better magnification vectors. This would be a challenging extension to our work, but potentially very valuable.

6.4 Relevant Papers

- [2] Was extremely useful for building the facial recognition system. Without the work done in this paper, it is unlikely that our method would have worked.

- [3] This paper was extremely beneficial. By using part of their MSVGC algorithm, we were able to significantly smooth the frequency signal extracted from the EVM magnified color. The paper was well written and so following their algorithm was delightful.
- [5] We looked into this paper since the color magnification is reliant on the same pixels being consistent across frames—hence the need for static scenes. However, we found that applying a center crop to the face detected bounding box actually provided the robustness to motion that we needed and never ended up implementing this paper. In addition, the methods in the paper seem to be more complicated than would be feasible for just a single component of the application.
- [1] We use the technique of magnifying color change in the human skin tones to measure the heart rate of a subject. In other words, this paper is some of the foundational work to our application.
- [6] This paper introduces a good way of approximating a filter for eulerian video magnification that gives better results than hand crafted filters. However, it only applies to motion magnification so after looking into it as a better way of extracting the heart rate, we decided to go with the original implementation in order to get the color magnification.
- [7] One of the potential problems we will face in building this system is the motion of people being recorded. In the original EVM work, they assume a stationary target for color magnification. This assumption is broken if we use a web camera to capture multiple people's heart rate information simultaneously. This work could be useful if we encounter problems with large scale motions. However, this was not the case case due to facial recognition and cropping which was again, surprisingly robust to motion

7 Time Report

Hourly report

7.1 Andrew Carr

- Idea Exploration - 8 Hrs
- Coding - 52 Hrs
- Reading - 14 Hrs
- Writing - 12 Hrs
- TOTAL: 86 Hrs

7.2 Michael Whitney

- Idea Exploration - 7 Hrs
- Coding - 51 Hrs
- Reading - 12 Hrs
- Writing - 11 Hrs
- TOTAL: 81 Hrs

References

[1] Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand, and William Freeman. Eulerian video magnification for revealing subtle changes in the world. 2012.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] Guochen Bai, Jifeng Huang, and Huawei Liu. Real-time robust noncontact heart rate monitoring with a camera. *IEEE Access*, 6:33682–33691, 2018.

[4] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[5] Miao Wang, Guo-Ye Yang, Jin-Kun Lin, Ariel Shamir, Song-Hai Zhang, Shao-Ping Lu, and Shi-Min Hu. Deep online video stabilization. *arXiv preprint arXiv:1802.08091*, 2018.

[6] Tae-Hyun Oh, Ronnachai Jaroensri, Changil Kim, Mohamed Elgharib, Fr'edo Durand, William T Freeman, and Wojciech Matusik. Learning-based video motion magnification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 633–648, 2018.

[7] Mohamed Elgharib, Mohamed Hefeeda, Fredo Durand, and William T Freeman. Video magnification in presence of large motions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4119–4127, 2015.