Self-contained, passive, non-contact, photoplethysmography: real-time extraction of heart rates from live view within a Canon PowerShot

Henry Dietz, Chadwick Parrish, and Kevin D. Donohue;
Department of Electrical and Computer Engineering, University of Kentucky; Lexington, Kentucky

Abstract

Photoplethysmography (PPG) is the detection of blood flow or pressure by optical means. The most common method involves direct skin-contact measurement of light from an LED. However, the small color changes in skin under normal lighting conditions, as recorded by conventional video, potentially allow passive, non-contact, PPG. A variety of methods have been applied to extract heartrate from such a video.

In another paper submitted to this conference by the same authors, a new processing algorithm based on autocorrelation is shown to be effective without needing extensive video preprocessing to enhance the signal. That method was implemented using floating-point arithmetic in MatLab to analyze complete videos. However, the algorithm's structure suggested that it might be possible to create a simplified, integer-only, approximation that is entirely incremental: updating a heart rate estimate as each frame is captured. This new, simplified, incremental algorithm allows a reprogrammed Canon PowerShot camera to function as a stand-alone, passive, non-contact, PPG device. The incremental integer algorithm and implementation are explained and evaluated in this paper.

Introduction

Photoplethysmography (PPG) refers to the detection of changes in blood volume, flow, or pressure by optical means. Although the precise cause of the appearance change being measured might vary depending on details of how the measurement is made, the cyclic pattern of appearance changes is known to correlate well with beats of the subject's heart. PPG is widely used for non-invasive measurement of heartrate.

PPG monitoring is usually implemented using devices that combine active light sources and optical sensing. A typical *pulse oximeter* works by measuring the difference in absorption between two different wavelengths of light passing through a finger[1]. The transmissive properties of hemoglobin depend on the amount of oxygen being transported. Higher oxygen levels absorb more red light (660nm), while lower levels absorb more infrared light (960nm), so the ratio between the two absorptions reveals the percentage oxygenation of the blood. However, the light absorption by hemoglobin varies with the volume of blood and is a tiny fraction of what is absorbed passing through the finger. With each pulse, the volume of hemoglobin in the arteries changes; by subtracting-out the base absorption, a low-amplitude cyclic waveform, the *plethysmographic trace* (pleth), is produced that reveals the heartbeat and allows scaling absorption ratios to compute percentage oxygenation.

Figure 1. *KOBRE live BPM display on ELPH180*

These pulse oximeters commonly use LED (light emitting diode) light sources, allowing them to be very small, efficient, and inexpensive. The device is placed, and firmly held in position, in direct contact with the subject's skin – typically, enclosing the last inch or two of a finger. In medical applications, the optical readings are converted to an electrical signal that is sent through wires to a heartrate monitor that performs analysis of the waveform, and computes and displays the trace, percentage oxygenation, and the heartrate in beats per minute (BPM). However, a similar type of contact PPG sensor has been built-into some cell phones, such as the Samsung Galaxy S5[2]. The cell-phone sensor measures absorption of active lighting by reflectance and the processors in the cell phone are used to compute, display, and track the BPM rate via a health monitoring application.

While contact PPG technology using active lighting is cheap, effective, and non-invasive, it is not ideal for many potential applications. Even monitoring a patient in a hospital bed, the wires leading from the patient to the heartrate monitor easily could become tangled or the connection interrupted; movements also could alter how the sensor is held to the skin, distorting the readings. There are also many circumstances in which it is awkward or infeasible to place a device on the subject, ranging from

Figure 2. *Canon PowerShot ELPH180*

environments that are hostile to electronics (e.g., near magnetic resonance imaging equipment) to remote diagnostics where the patient does not have access to specialized equipment.

There also are many potential non-medical applications of PPG for which active lighting and contact sensing would be unacceptable. A camera might implement unobtrusive monitoring of at-risk individuals (e.g., seniors) as they perform normal activities in their home, or it could provide heartrate data to serve as an additional parameter to help detect anomalous, potentially dangerous, behaviors in public places.

As the survey by Sun and Thakor[3] documents, the transition from point contact to imaging non-contact PPG is technically challenging and actively researched. While a single-point sensor is sufficient for contact measurements, it is well known that even slight changes in the positioning of the point sensor during measurement can be significant: motion artifact corruption of data is a major issue. For non-contact PPG, imaging allows simultaneous sampling of many points to produce images of the changes in blood flow, but it is also possible to use the image data to more accurately position regions to sample, perhaps even tracking to minimize motion artifacts. Poh et al.[4] was the first to demonstrate that it is possible to use a motion-tolerant webcam to monitor BPM of multiple subjects simultaneously.

The goal of the current work is to produce a fully self-contained, non-contact, imaging PPG device that operates using ambient lighting. The device should be able to produce a readout of BPM in real time, performing all necessary image and signal processing using the computing resources embedded in the device. Ideally, it should be possible to monitor multiple subjects simultaneously. Perhaps most importantly, the device should be cheap, even as a prototype; our prototype cost just $70. The software is called KOBRE, and a live display of BPM of this paper's lead author is shown in Figure 1.

The Target Platform: Canon PowerShots

Although a cell phone might seem the obvious target platform, compact cameras offer various advantages – and cost less. Canon's PowerShot line of digital compact cameras provides a wide array of models with varying features and performance. Although the KOBRE software can run on most models, the primary target is a low-end model called the ELPH180. Despite the low cost, it contains a Bayer-filtered 20MP sensor that delivers 12 bits per pixel with an 8X optical zoom (28-224mm in 35mm equivalent focal lengths). Figure 2 shows the ELPH180 with its lens extended; the cameras measures only about 4"x2.5"x1" with the lens retracted. Unlike most cell phones, it accepts a standard SD card for storage and comes with a removable battery and separate charger. 3rd-party batteries cost less than $10.

CHDK: The Canon Hack Development Kit

Of course, the low cost and nice features of the camera would mean nothing if the camera couldn't be reprogrammed to function as a non-contact PPG. Although Canon does not support such reprogramming, there is free 3rd-party software that does.

The community developed and maintained CHDK, Canon Hack Development Kit[5], provides for very flexible and fairly portable programming of most PowerShot models. Rather than replacing the Canon firmware, it can be automatically loaded from an SD card when the camera is turned on; it also is possible to have the camera immediately begin executing a script under CHDK. Since it does not disturb the firmware, not only can CHDK invoke any operations the firmware supports, but the camera can be reset from most programming errors by simply turning power off and removing the SD card.

Programming of a camera under CHDK can be done using a combination of interpreted scripts and compiled C code. User C code can be implemented as modules compiled into CHDK, changing or extending its functionality. Although recompiling is awkward, the resulting code runs as fast native ARM instructions. Alternatively, users can write scripts in either BASIC or Lua. Scripts run slower than native ARM code, but are easier to develop. It is even possible to interactively load and run Lua scripts from a computer being used as a development system via CHDKPTP over a USB connection.

ELPH180 Capabilities Under CHDK

There are a variety of different mechanisms used by CHDK to allow programmable control of a PowerShot camera.

The most basic interface uses the firmware's event-handling data structures. Each user-interface action, such as pressing a button, enqueues an event that is eventually processed by the firmware. Thus, any action that can be performed through a sequence of user interface actions can be invoked under program control by simply enqueuing the appropriate sequence of events. For example, capture of an image is a very complex process, but it can be triggered by merely queuing events that simulate press and release of the shutter button.

However, for most PowerShot models, the entry points for a large number of lower-level internal firmware functions, and the location and layout of many key data structures, have been identified. Some of these functions and data structures are essentially the operating system interface, but many allow direct control of various firmware parameters and hardware devices. For example, the user interface allows setting the imager gain by specifying an ISO film speed. However, the ISO numbers seen in the user interface are rounded "market" values. Lower-level functions allow

directly examining and setting the actual ISO value, and not just to the values accessible via the user interface. Even deeper, exposure settings are computed using scaled APEX (Additive System of Photographic Exposure) values, and the sensor gain can be manipulated very precisely at that level. Other functions provide such diverse control as stepping the zoom lens to a specific setting, reading the temperature of the image sensor, and displaying text and graphics on the rear LCD.

For PPG inside a Canon PowerShot under CHDK, the key functionality is the ability to rapidly capture and access a stream of images. There are three ways image sequences can be captured: as images, video, or live view.

High-quality full-resolution images can be captured using the mechanical shutter. Normally, each image automatically would be written to the SD card as a JPEG, but CHDK allows directly accessing the ELPH180's 20MP raw 12-bit-per-pixel sensor data from a frame buffer in main memory. Unfortunately, most PowerShots cannot sustain a rate of even 2 frames per second (FPS) in this way; reading a complete image from the CCD sensors used in most models takes too long.

Another alternative is to capture a video. The ELPH180's video modes are typical of CCD-based PowerShots, allowing capture of either 1280x720 pixel HD video at 25 FPS or 640x480 pixel at the NTSC standard rate of 29.97 FPS. The higher FPS for video is enabled by not sampling the image sensor at full resolution, and a series of smaller frame buffers are used during H.264 video compression. Unfortunately, this video processing consumes most of the camera's computing resources, leaving insufficient time for PPG processing.

The last option is to intercept the live view image stream. Live view images are very similar to video frames, but the processing uses fewer computing resources. Interception of live view frames does not ensure a constant FPS rate, but it is possible to achieve up to about 25 FPS in good lighting conditions without consuming all the available computing resources. This is the method used to implement KOBRE.

The computing resources available to CHDK are significantly poorer than in most cell phones. The 30MB frame buffer takes up most of the main memory and the 83 MIPS delivered by the ELPH180's processor is not fast by current standards. However, under most circumstances, one of the two 32-bit ARM cores is usually available for execution of user code. By avoiding floating-point arithmetic and optimizing the algorithms and data structures, complex computations can be done in real time.

The New PPG Algorithm

Imaging PPG using ambient lighting is a very challenging task. There are three major complications:

1. The SNR (signal to noise ratio) is extremely poor. The reported strength of active-lighting contact PPG signals is approximately 2% of the measured values, and the signal observed using ambient lighting can be significantly weaker. The noise margin is potentially further compromised by encoding; JPEG stills and most video encoding schemes reduce luminance information to just 8 bits per pixel and use even fewer bits to represent color.
2. The shape of the heartbeat waveform is not a simple sine wave.
3. Ambient lighting and autoexposure drift over time, often imposing potentially complex trends in the image data. Some types of ambient lighting even flicker at rates which could interfere with the PPG signal.

Many different algorithms have been explored for PPG, and approaches using FFTs (fast Fourier transforms) are perhaps most common. However, frequency domain analysis is impaired by all three of the above issues. Thus, most of the algorithms rely on extensive filtering that restricts the frequencies to a narrow band around the expected heartrate.

In theory, autocorrelation should be less sensitive to the above signal quality issues without needing heavy filtering. Parrish et al.[6] presents an autocorrelation-based approach to ambient light imaging PPG that seems more effective than most methods. Although the algorithm used in that paper is also computationally cheaper than most alternatives, it is floating-point intensive, far too complex to run inside a CHDK PowerShot, and is organized as batch processing rather being structured as an incremental algorithm for updating results in real time. The following four subsections describe how each of the four major components of the autocorrelation-based algorithm were (heavily) modified to be suitable for real-time use inside an ELPH180.

Reduction To One Value/Sample

KOBRE's analysis is based on examining images intercepted from the ELPH180 live view feed, but performing full analysis on each pixel is neither feasible nor desirable. Instead, a region of interest (ROI) is defined and pixel values within that ROI are summed. This summing not only reduces the amount of data which must be processed by the rest of the algorithm, but also effectively increases the SNR.

CHDK provides a Lua-callable module that examines the live view feed to detect motion and trigger captures. Thus, the KOBRE Lua script can use `md_detect_motion()` to summarize a live view frame and use multiple `md_get_cell()` to sum cells covering the desired ROI. The catch is that the live view feed framerate is not constant, so the Lua script uses the millisecond-accurate `get_tick_count()` and `sleep()` calls to control when the live stream is sampled. The Lua script delays sampling as needed to obtain the desired constant framerate, and aborts with an error message if any sample occurs too late.

At this writing, the ROI is a patch in the center of the frame. However, to guide autofocus, PowerShots can automatically recognize and track multiple human faces, and bounding-box coordinates are embedded in JPEG captures. This face detection and tracking also happens during live view, and that is how the white marks bracketing the face in Figures 1 and 5 were drawn on the live view. Unfortunately, the internal data structure in the PowerShot that is used to maintain face tracing data has not yet been identified. If/when it is, each face being tracked could define an ROI and KOBRE could easily be updated to simultaneously output BPM rates for each tracked face.

Detrending Of Values

Unfortunately, changes in the sample values are not only due to the heartbeat changing the volume and pressure of arterial blood. Assuming that the ROI remains well-positioned on the subject, there are two major phenomena that can impose trends on the sequences of ROI sample values:

- Changes in live-view autoexposure. Live-view exposure parameters are automatically set by the Canon firmware, and can change in response to scene content that is not part of the ROI. Changes can be dramatic, but are made with some hysteresis.
- Changes in subject lighting. Natural lighting can change as clouds or other objects filter or block the Sun and artificial lighting often flickers, but even light reflecting off the shirt of a person walking nearby can impose a color cast with a magnitude that is significant compared to the changes caused by the heartbeat.

Although the detrending processing used by Parrish et al.[6] is less expensive than the filtering used by many others, it was not feasible to use it in the CHDK versions of KOBRE. Thus, the initial versions did no detrending at all. Surprisingly often, trends did not prevent KOBRE from detecting the heartrate, but it was very common that trends would cause KOBRE to latch on the highest allowed BPM because most trends are increasingly disruptive over longer pulse periods.

The previous subsection talked about a sample value being computed from an ROI. In reality, these cameras capture color information as well as brightness, so it is possible to use both aspects in summarizing an ROI. This opens the possibility of cheaply approximating detrending by distinguishing spectral components. Conceptually, color-based detrending is similar to how difference between red and near infrared components is used to recover actively-lit contact PPG signals.

Measuring ambient lighting reflected by bare skin, especially the subject's face, Verkruysse et al.[7] found that the strongest pulse amplitude and lowest noise came from using the green channel of an RGB (red, green, blue) imaging sensor. Why is the prefered color different from pulse oximeters?

The absorption spectrum of both HbO2 and Hb (oxygenated and non-oxygenated Hemoglobin) in 2nm wavelength steps were reported by Prahl[8]. The 660nm red and 960nm near infrared wavelengths actually correspond to the points of largest *difference* between the spectra of hemoglobin based on oxygenation, but neither is absorbed very strongly compared to other wavelengths. Absorption of HbO2 actually peaks around 414nm and Hb peaks around 432nm, which would be in the near ultraviolet – wavelengths not easily processed using ambient lighting and commodity cameras. The peaks in the range best-detected by sensors used in commodity cameras actually occur at 576nm for HbO2 and 556nm for Hb, corresponding to yellow-green colors, but even blue absorption is significantly higher than red or infrared. Comparing the spectra to typical digital camera RGB sensitivity curves, the green channel is obviously the most responsive to volume of hemoglobin (oxygenated or not). In addition, consumer digital camera sensors generally use a Bayer color filter array to distinguish RGB colors, and the Bayer pattern repeats two green pixels for each red or blue pixel. Because the green channel is sensed by more pixels, the green channel image has lower noise.

Although the sensor in most Canon PowerShots uses a conventional RGB Bayer filter to capture 12-bit pixel values, the live-view image stream is interpolated, scaled to a lower resolution, and converted into a logarithmic 8-bit YUV colorspace (compatible with NTSC television and JPEG's Y'CbCr) encoding approximately 9 stops of scene dynamic range. Each Y value represents the luminance or brightness (which is dominated by the green channel), while the U and V components specify the color as signed chroma offsets. The live-view image stream actually represents each group of four consecutive pixels with a sequence of six 8-bit values, UYVYYY, in which color information is shared by each group of four pixels. Thus, averaging the four Y values gives a very low noise approximation to the part of the spectrum absorbed by hemoglobin. Computing a red channel value from each UYVYYY record provides a brightness measurement that is far less affected by the volume of blood, thus it can be used as an offset representing the approximate brightness in ambient lighting.

The above logic suggests that the Y or a re-computed G channel should yield the best quality readings from a PowerShot, and this is empirically the case. The re-computed R channel can be productively used to subtract out changes in brightness of ambient lighting: Y-R or G-R effectively implements detrending.

Autocorrelation

Autocorrelation is basically a form of differencing, and differencing can be done fully incrementally. Here, the heartrates reported are always based on sums of squared differences. CHDK integer-only Lua code to add a new value to the squared difference sums is shown in Figure 3.

Each BPM output represents an average over the last `depth` ROI samples at the selected framerate. The difference sums are incrementally updated using a circularly-indexed table, `vals[]`, which holds the `depth` most recent values. The value of the ROI in frame `fno` is therefore stored in `vals[fno%depth]` and the value from `k` frames earlier is in `vals[(fno-k)%depth]`.

The number of frames that corresponds to the period of the fastest pulse rate to be detected is `minfw`, and the period of the slowest pulse rate to be detected is `maxfw`, each of which must be less than `depth`. A table, `difs[]`, is used to maintain the sum of the last `depth` squared differences for each period from `minfw` to `maxfw`. Incremental update of `difs[i]` begins by subtracting the squared difference contribution from the oldest entry in `vals[]`, and then adds the contribution from the new entry. Note that the oldest difference contribution is between the oldest value, `ov`, and the sample `i` entries younger: `vals[(fno+i)%depth]`. The new addition is between `v` and the sample `i` entries older: `vals[(fno-i)%depth]`.

The incremental update computation therefore only requires `maxfw-minfw+1` difference sum updates per frame. In the same loop where these are being computed, the minimum and maximum squared differences are also recorded in `mind` and `maxd`, to be used in judging quality of the heartrate waveform.

```
function addvalue(v)
  local i, t, d
  -- get old value, save new
  local ov = vals[fno%depth]
  vals[fno%depth] = v
  -- for all possible wavelengths
  for i=minfw,maxfw do
    -- subtract oldest squared difference
    t = ov - vals[(fno+i)%depth]
    d = difs[i] - (t * t)
    -- add new squared difference
    t = v - vals[(fno-i)%depth]
    d = d + (t * t)
    -- record new sum of squared differences
    difs[i] = d
    -- update mind, maxd for scaling
    if d < mind then mind = d end
    if d > maxd then maxd = d end
  end
  -- next frame
  fno = fno + 1
end
```

Figure 3. *Lua incremental squared difference sum update code*

Selection Of The "Best" Correlation

After calling `addvalue()`, the heartrate corresponding to the period with the smallest value in `difs[]` would be the obvious choice to report. Originally, KOBRE did precisely that. If the sampling framerate was `fps` and `i` was the sample interval with the smallest `difs[i]` value, the reported BPM was simply `(60 * fps) / i`. Unfortunately, this method for selecting the heartrate to report easily can be driven by noise to report the maximum allowed BPM; a more robust selection method was needed.

The values computed for `mind` and `maxd` provide insight into the quality of the PPG waveform. If both values are similar, the waveform is not being recovered with good fidelity. Empirically, `maxd` is at least several times larger than `mind` when the PPG waveform is above the noise level. Further, `mind` should be small, generally within three orders of magnitude of the number of samples for which the squared differences are summed – typical values would be in the hundreds or thousands, not millions. KOBRE can thus detect and indicate at least some situations in which its BPM readings are highly suspect.

To improve the accuracy and stability, the latest version of KOBRE selects the BPM to report by computing a weighted average over all periods represented in `difs[]`. The weighting for each period length, `i`, is computed by subtracting the value of `difs[i]` from the maximum allowable difference sum. Although `maxd` can be used as the maximum, it is more reasonable to set a threshold which is somewhat lower, to a limiting value we'll call `limd`. Any match poorer than `limd` is omitted from the weighted average. Lua code implementing this is shown in Figure 4.

It is interesting to note that use of `(sumi / sumd)` in the last line forces the `bpm` reported to have a granularity corresponding to the number of frames per second. For example, sampling a

```
for i=minfw,maxfw do
  d = difs[i]
  if d < limd then
    d = limd - d
    sumd = sumd + d
    sumi = sumi + (d * i)
  end
end
bpm = (60 * fps) / (sumi / sumd)
```

Figure 4. *Lua weighted selection code*

heartrate around 60 BPM at 20 FPS, a difference in period of one frame (i.e., 1/20s timing error) would change the reading by 3 BPM. By instead computing bpm as `bpm = (60 * fps * sumd) / sumi` the weighted averaging allows production of `bpm` values with a much smaller step size, but at the risk of overflowing CHDK Lua's 32-bit integers if `limd` is set too large.

Parameters To The Lua Script

CHDK allows scripts to have parameters that can be set directly through CHDK's menu system. The parameters can have specified defaults, and user-set values are remembered across runs, so this provides a very simple way to allow users to adjust the behavior of a script without programming. KOBRE currently has five parameters:

Parameter	Default	Range
Lowest BPM	40	30–60
Highest BPM	180	100–200
FPS target	0	0 (automatic)–24
Buffer depth	0 (automatic)	0–120
UYVRGB color	2 (Y)	0–6 (6 is Y-R)

Adjusting the Lowest BPM and Highest BPM is normally not necessary, but setting a smaller range will increase processing speed and tends to improve reliability. The FPS target rate is normally set automatically to allow the requested range of BPM measurements, but can be manually set higher to decrease step size for reported BPM or can be set lower to ensure KOBRE will not abort due to late sampling of frames. The Buffer depth is also normally set automatically to the minimum needed so that the desired BPM range can be measured with a quick response time, but larger buffer depths increase hysteresis. For example, a Buffer depth of 60 with FPS target 20 would mean each BPM reading is averaging over 3 seconds – although the BPM reading will be updated every 1/20s. The UYVRGB color parameter allows selecting the color channel to be used for recognizing the PPG waveform. Channel 6 requires a modified version of CHDK, but is how the color-based detrending is implemented.

Testing

As should be apparent throughout this paper, consumer digital cameras shooting at video framerates barely have the dynamic range necessary to record non-contact PPG waveforms using typical ambient lighting. Even with the same subject and controlled lighting, people cannot directly control their heartbeat, so it is

Figure 5. *ELPH180 PPG of a video playback*

unlikely that an experiment based on directly monitoring a live subject could ever be precisely repeated. This makes testing and tuning real-time algorithms implemented inside a camera very difficult because even minor changes to the implementation of the algorithms can affect timing of samples, etc.

The solution was to create a simulated subject such that real-time experiments not only could be accurately repeated, but it also would be possible to directly vary the heartbeat to explore potential timing issues with the algorithm and its implementation. Initial testing used a simple javascript program to repaint the background of the web page it runs in with a repeating cycle of a solid color of varying brightness. By pointing the camera at a monitor, preferably an active LCD display with a high refresh rate, and running that javascript, highly repeatable experiments could be performed at any desired heartrate.

The problem is that the cycle of the javascript program is not a very realistic model of heartrate-induced changes in the appearance of human faces. Thus, we switched to playing videos of human faces with known heartrates. By varying the playback framerate, the effective heartrate could be changed to any desired value within a fairly large range. Figure 1 shows the rear LCD of an ELPH180 running KOBRE while pointed at a video of author Dietz being played on a monitor. In addition to videos we captured, we experimented with the sample videos from Wu et al.'s work in Eulerian Video Magnification (EVM)[9]; Figure 5 shows an ELPH180 running KOBRE on one of their test videos.

Tests primarily varied the video playback framerate to see if KOBRE would correctly track the heartrate +/- 1 step. However, both the original and magnified videos from Wu et al.[9] were used to confirm that the detected heartrates were the same for both the subtle original and enhanced videos. Before detrending and weighted-average selection were used, only about 1/3 of the tests settled on the correct BPM; most became stuck at the maximum BPM. Color-based detrending improved that to about 1/2. However, the weighted-average selection dramatically improves stability. Combining detrending with weighted-average selection can drop the achievable sampling FPS significantly, but appears to work comparably well to the smarter selection without detrending.

Conclusions And Future Work

This paper has presented a fully incremental computation, approximating a new autocorrelation-based algorithm, that implements non-contact PPG using ambient lighting. While the current CHDK implementation of KOBRE in a sub-$100 Canon PowerShot ELPH180 is not sufficiently stable to be useful as a production PPG device, it has clearly demonstrated that this type of implementation is a viable non-contact way to monitor heartrate.

The use of variable-rate playback of videos as live test scenes with known and adjustable heartrates proved critical in allowing the implementation of KOBRE to be tuned. Most test videos were captured without special lighting. However, the subjects were always relatively well-lit. It also is unknown how accurately the tonal properties of the displayed video mimic the live scene. There is potential for multi-person BPM tracking using the face-tracking of the PowerShot camera to select ROIs, but sensitivity to subject motion has not been tested: subjects in the videos were all seated facing the camera.

An undergraduate senior project team (of which Parrish is a member and Dietz the advisor) in the Electrical and Computer Engineering Department of the University of Kentucky is currently working on improving this approach and testing with live subjects.

References

[1] Prasanna Tilakaratna, How pulse oximeters work explained simply, https://www.howequipmentworks.com/pulse_oximeter/; accessed January 27, 2019

[2] Samsung: What does the heart rate sensor measure?, https://www.samsung.com/us/heartratesensor/; accessed January 27, 2019

[3] Y. Sun and N. Thakor: Photoplethysmography Revisited: From Contact to Noncontact, From Point to Imaging, IEEE Transactions on Biomedical Engineering, vol. 63, no. 3, pp. 463-477, March 2016. doi:10.1109/TBME.2015.2476337

[4] Poh, Ming-Zher, Daniel J. McDuff, and Rosalind W. Picard: Noncontact, automated cardiac pulse measurements using video imaging and blind source separation, Optics Express 18 (2010): 10762, doi:10.1364/OE.18.010762

[5] Canon Hack Development Kit (CHDK), http://chdk.wikia.com/wiki/CHDK accessed January 31, 2019

[6] Chadwick Parrish, Kevin D. Donohue, and Henry Dietz: Autocorrelation-Based, Passive, Non-Contact Photoplethysmography: Computationally-Efficient, Noise-Tolerant, Extraction of Heart Rates from Video, Electronic Imaging 2019, Computational Imaging XVII.

[7] W. Verkruysse, L. O. Svaasand, J. S. Nelson: Remote plethysmographic imaging using ambient light, Optics Express 16 (26): 21434–21445. 2018. doi:10.1364/OE.16.021434

[8] Scott Prahl: Optical Absorption of Hemoglobin, https://omlc.org/spectra/hemoglobin/, accessed January 27, 2019

[9] Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Fredo Durand, and William T. Freeman: Eulerian Video Magnification for Revealing Subtle Changes in the World, ACM Transactions on Graphics (Proc. SIGGRAPH 2012), vol. 31, no. 4, 2012. doi:10.1145/2185520.2185561