Image Processing Methods For Mobile Horizontal Gaze Nystagmus Sobriety Check

Judson Wilson
Department of Electrical Engineering,
Stanford University
Email: judsonw@stanford.edu

Alexei Avakov
Department of Electrical Engineering,
Stanford University
Email: linrav@stanford.edu

***Abstract*—A method for eye gaze measurement is presented for the intended purpose of alcohol induced Horizontal Gaze Nystagmus detection. The method uses two pupil locating techniques on two still images, followed by geometric estimation to perform initial facial feature estimation. Laplacian-of-Gaussian filtering in scale-space is used to locate the pupil center on an image where the eye is centered, and gradient techniques are used to locate the pupil when the eye is not centered. After the initial geometric estimation, template matching techniques are used for video frame registration, and integral projection techniques combined with one dimensional template matching are used for fast, real-time gaze measurement. The method is designed to require minimal computation on video frames, and is tailored for use on a mobile device platform. A test implementation on a PC, using prerecorded video data, successfully tracks a noisy sinusoidal eye gaze with a mean error of approximately 11 percent.**

I. Introduction

The Horizontal Gaze Nystagmus (HGN) test is a procedure that is routinely performed by law enforcement personnel to determine whether a person is driving under the influence of alcohol or drugs. The test is performed by having the subject visually track an object while the motion of their eye is observed. Alcohol produces physiological nervous effects that impair one's oculomotor response. Eye tracking becomes erratic with visible 'jerks', especially at the horizontal extremes of eye motion [1].

This paper demonstrates methods to perform an HGN test on a mobile device platform. From a user interface perspective, we sweep moving graphics across the display of the device, and then track the users' eye movements from the video output of the front-facing camera as shown in Fig. 1. Image processing techniques are then used to efficiently perform gaze tracking.

Similar work exists, such as the proprietary Android app "BreathalEyes," and a system for an automotive setting by Thien et al. [2]. BreathalEyes relies on a second user's help to administer the test, to ensure camera alignment and to provide the moving visual cue by means of a finger. This is a different approach to user-experience from our intended method. The user-experience proposed by Thien et al. is closer to what we propose, although they envision a system mounted to the dash of a car rather than integrated into a mobile device. Their image analysis methodology is similar to ours in basic concept, but primitive and also potentially requiring more computational power.

The basic sequence of our method begins with the acquisition of two still images. In one image the eye is to be centered, and in the other the eye is to be pointed towards the outside corner, away from the nose. Visual cues aid the user in proper alignment, and screen dimming is used to reduce glare visible on the eye in the still images. These two images are used to compute a geometry estimation to locate the inner eye corner (tear-duct), along with the dimensions and path of the pupil. The second phase requires the user to track an object moving across the screen, following a linear path at sinusoidal or constant absolute velocity. Screen areas are kept as dark as possible, again to minimize reflections. The tear-duct location is used for frame registration to counteract movements of the camera or face. The pupil path estimation is used to create an image region mask for a projection into one dimension. Finally, the pupil is located in the horizontal dimension, and a position signal is produced.

II. System Model

We will begin with a statement of our assumptions. Then an overview of the system will be given, followed by implementation details for each component. The system will be simultaneously described from an image processing standpoint and a user-interface standpoint in order to highlight their complementary design.

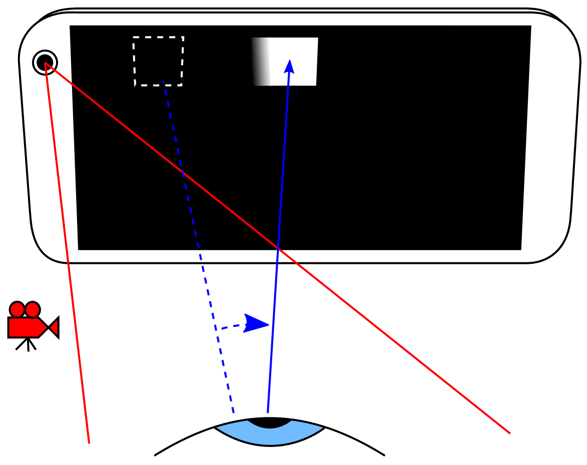


Fig. 1. Mobile device eye tracking and visual stimulation.

A. Assumptions

The proposed system was designed under several assumptions, including:

- Movements of the camera and head are large enough that frame registration is needed, but small enough that their effects on eye gaze are small in comparison to HGN effects.
- Distance and rotational effects are small enough that geometric distortion such as rotation and scaling will not be large or rapid enough to distort HGN measurements.
- Lighting conditions will remain stable throughout the duration of the test procedure.
- The user's facial characteristics will remain stable, with the exception of eye position. Under this assumption we impose the condition that the user will not blink (a case for future work).

B. Model Overview

The horizontal gaze measurement problem is twofold: frame registration and eye feature location. These two goals, ideally, should be accomplished in real-time for a pleasant user-experience, so we aim for computationally simple methods.

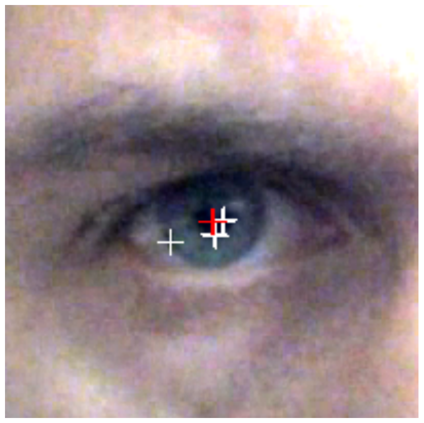
For dependable frame registration (in this case, a combination of camera and facial stabilization) we chose to stabilize the location of the inner eye corner: the tear-duct. This feature is defined by a unique structure with a sharp corner which is useful for tracking, and it is relatively stable under changes in facial condition, including eye movement. This observation was also exploited by [4], [5]. Our method of choice is template matching, for simplicity, which will be discussed later.

For eye feature location in the horizontal gaze test, only the horizontal coordinate is needed. For this task it is sufficient to observe a very small subset of the available pixels. We propose a mask based method, hiding all pixels except a small horizontal band which should always contain the pupil under proper use, along with some added horizontal span to ensure that the portions of the iris on the left and right hand side of the pupil are included, as well as a small portion of sclera (the white portion of the eye). With this horizontal band of pixels we can transform the problem into a single dimension by means of an integral projection [9]. Locating the pixel can now be accomplished quickly with a one-dimensional pupil/iris template.

The following sections detail the construction of the templates and masks needed to perform these steps, followed by the details of their use.

C. Centered Eye Acquisition and Calculations

The goal of these steps is to determine the pupil geometry when the eye is centered, and to determine the location of the tear-duct from which a template is made. Because of the unique nature of mobile devices, we can have the user aid in locating the eye. The user is instructed to center their eye on the screen of the device by repositioning the camera, and to also position it such that the eye is as large as possible, to increase resolution. If additional precision is needed, such as for the tear-duct location, a wireframe graphic may be overlaid on the video display so the user may manually align the feature.



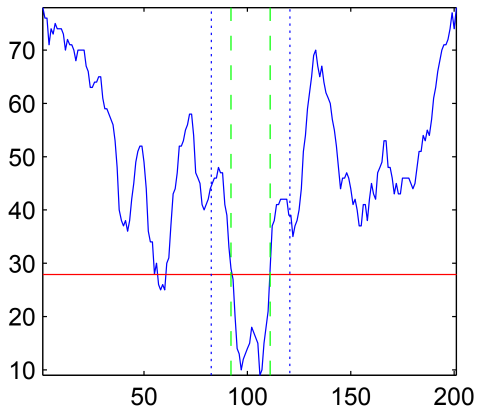


Fig. 2. Top: LoG centers, strongest 1/3 shown in red. Bottom: Integral projection, pupil edge thresholds (red, solid) and detection positions (green, large dashed) are indistinguishably close for std. dev. and Otsu methods. Otsu method boundary marked in blue, small dashes. *(Color adjusted for print.)*

Once the user has aligned the camera, proper location can be verified using an eye detector algorithm. Our implementation uses a Viola-Jones Haar Cascade trained specifically for the right eye [6], implemented in OpenCV. Setting the appropriate scale will usually yield detection results that are stable in position and scale.

Once the position is deemed suitable and stable, the screen of the device is darkened to reduce reflection, and an image is recorded.

1) Pupil Center: To find the pupil center, the Laplacian-of-Gaussian (LoG) filter was used to generate a scale-space detector. This is accomplished by passing LoG filters of increasing variance $t = \sigma^2$, and multiplying the outputs by the factor t. It was hoped that a strong local maximum would coincide with the pupil (a very strong "dark blob") in both space and scale, such that both a center location and size estimation could be made. Unfortunately this did not work in testing. We hypothesize this is due to the fact that an image of the eye, when centered, is effectively a combination of nearly concentric features at different scales.

On the other hand, this property also gives the detector good locational stability at different scales, even though the eye region has many features. Therefore, our method is to use LoG filters of varying scale (relative to the scale detected by Viola-Jones) in the region of interest, and detect the maximum values for each scale. The centers corresponding to the peaks in N scales (we use N = 1/3 of the computed scales) are averaged. This approach was chosen for simplicity, but the use of clustering may aid outlier elimination. See Fig. 2.

2) Pupil Width: Pupil width can be determined in one dimension. Our method is to sample a thin horizontal band of pixels, smaller than the minimum expected height of the pupil, and sum them into a vector, which is effectively an integral projection [9]. This strip region is chosen to be incident to the pupil center calculated previously. The pupil is a particularly dark region, surrounded by a lighter region, with a sharp transition between them. To detect this transition edge, variance analysis and thresholding may be used. Our implementation samples the 10 pixels nearest the pupil center in the integral projection, and uses their standard deviation σ as the basis of a threshold. As shown in Fig. 2, an averaging window is passed from the center outward in each direction, and an edge is marked when the moving average is larger than some multiple of σ. For our test case we used a value of approximately 5, although more tuning is needed. As a refinement step, a sample of pixels between the located edges, and extending a fraction of this pupil width beyond the edge into the iris, are taken and Otsu's method [3] is applied to produce a new edge threshold that maximizes intensity variance between the iris and pupil, from which the width is recomputed.

Using this method, it is worth noting that the centers from the LoG based detector may be more suitable for this step than a center from a purely circular edge based approach. The LoG detectors will be drawn to the center of dark blobs, and thus will 'dodge away' from reflections that are likely to exist on the upper edge of the pupil from overhead lighting. Circular edge based detectors have tendencies of choosing a center without regard for whatever is located at this center. Using this pupil width measuring scheme in the presence of strong bright reflections will yield poor results without the use of heavy gaussian filtering.

3) Tear-Duct Location: The tear-duct feature is located using a geometric approximation for simplicity. The tear-duct is estimated to start a fixed factor of the pupil width away from the pupil center, and to have a height and width that are also relative to the pupil width, as shown in Fig. 3. One drawback of this method is that it will, in general, produce templates of varying size and location even for the same user, simply by changes in pupil size. The optimal template will be as small as possible yet be rich in features for template matching, therefore geometric methods like the one proposed need to oversize the template to ensure usefulness. Performance will vary depending on setting.

Many advanced methods for finding the tear-duct, or inner eye corner, exist. They include the use of Gabor wavelets

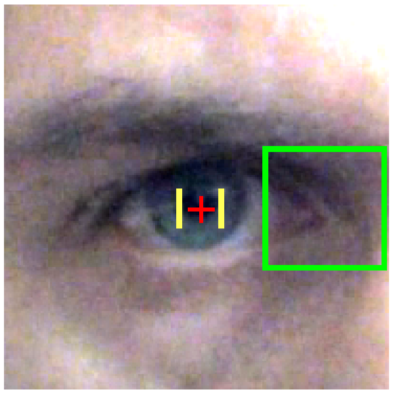


Fig. 3. Tear-Duct Location. *(Color adjusted for print.)*

[4] and specialized filters [5]. Both methods require specific models that were unavailable to us, and such custom development was outside the scope of this project. These methods, particularly [5], may prove attractive for further development, although it is unclear how tolerant they are to changes in scale. A simpler method based on deformable templates [7] was also tested, but was not successful in the prerequisite steps to determine the shape of the iris, likely due to a combination of iris lightness and shadowing.

As mentioned before, the user-interface can include visual cues that allow the user to manually align the tear-duct. This would likely still benefit from using one of the above methods as a validation test.

D. Rotated Eye Pupil Center Acquisition and Calculations

Detecting the pupil in the rotated position presents a different challenge than the centered position. Testing of the LoG method yielded less consistent results across different scales. We hypothesize that the previously concentric and symmetric features are now partially occluded and shifted, and therefore have lost much of the symmetry and concentricity.

The gradient based methods of [8] proved to be useful in this situation. To roughly summarize, this method finds a location c whose locational direction d_i to every pixel x_i aligns with the pixel x_i's gradient vector g_i in a sum-of-inner-products sense. See Fig. 4. The cost function is scaled by the inverse of intensity (of a blurred image), increasing the weighting of low-intensity pixels.

We found the method, when applied to blurred data, contained local maxima within the pupil, but sometimes contained local maxima in the dark eyelash as well. To mitigate these undesired peaks, a penalty function was applied. We describe this penalty function as a "deadzone-gaussian" function, plotted in Fig. 6 of the form:

$$z(x, w, \sigma) = \begin{cases} 1, & |x| \leq \frac{w}{2} \\ exp((|x| - |\frac{w}{2}|)^2/(2\sigma^2)), & |x| > \frac{w}{2} \end{cases}, \quad (1)$$

This penalty function is applied in the vertical dimension, to penalize local maxima that are distant from the previously

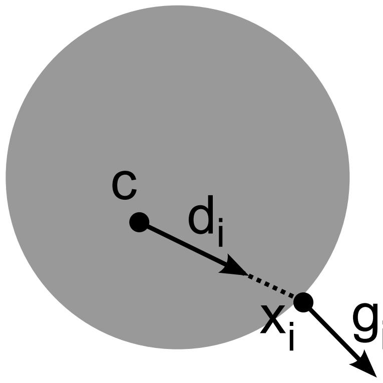


Fig. 4. Gradient method vectors of [8].

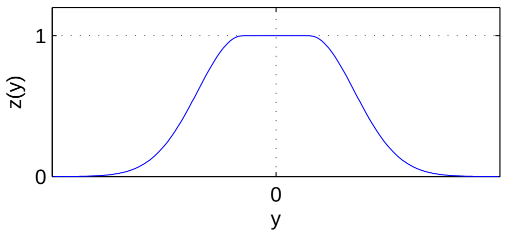


Fig. 5. Deadzone-Gaussian Function

calculated centered-eye vertical coordinate beyond some fraction of a pupil width. This enforces the assumption that the difference in eye position is almost entirely in the horizontal direction.

E. Mask Region Generation

The path of pupil motion is estimated to lie on a line connecting the two pupil centers calculated previously. A region mask is created to include a finite width strip following this path, with a height equal to the pupil width calculated previously. The path is extended linearly by a multiple of the pupil width to ensure that the portions of the iris to the left and right side of the pupil remain visible at the extremes of motion, which will prove useful for feature detection, as shown in Fig. 6. When this region mask is applied to the image, we model the image as having little variance in the vertical dimension in comparison to the horizontal dimension, yet the horizontal dimension contains sufficient information to estimate the horizontal angle of the eye.

F. Frame Stabilization and Face Centering

As discussed previously, frame registration, in this case a combination of frame stabilization and face centering, is performed using grayscale tear-duct template matching. Our best results were achieved with normalized cross-correlation, a cross-correlation variant that removes the mean of the template and the image (specifically the mean of the region of the image 'under' the template), and then performs cross-correlation normalized to both the template and the image. This process can either be used to shift the frame, or equivalently the result may be used to shift region mask coordinates in the next step.

For added precision, a sub-pixel estimation method was used. Following a 'Biquadratic Lagrange surface fitting algorithm" [10] on the 3x3 pixel region surrounding the peak of the cross-correlation, a more precise approximate maximum is found by applying a Newton Method approach on the surface. This fractional distance is stored as a pupil distance adjustment to be used in the final calculation. For adjustment factors greater than half of a pixel, the integer pixel shift is adjusted and the fractional amount is also adjusted accordingly to produce the correct numerical rounding behavior and yield a better mask registration.

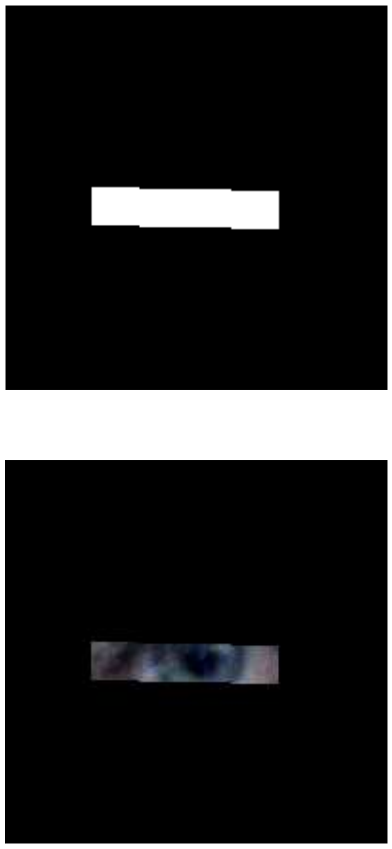


Fig. 6. Top: Region Mask, white areas represent region. Bottom: Corresponding region of a video frame. *(Color adjusted for print.)*

G. Eye Motion Tracking

Once the tear-duct is located, the region mask can be properly registered with the face image. This region mask facilitates the transformation of the horizontal eye position into a single dimension by means of the horizontal integral projection [9]. This projection sums the pixels in the mask region of the image in the vertical direction, reducing the problem to a single row of average intensity values. For this method, because the amount of data is now quite small, the operations are performed in RGB color space to retain a small amount of additional information. This processes is illustrated in Fig. 7.

Eye orientation is determined by one-dimensional normalized cross-correlation with a pupil/iris template. The normalized cross-correlation is calculated separately in the R, G, and B domains, and the results are summed before peak detection. The pupil/iris template is created by applying the mask and integral projection to the initial centered eye image, and using the known pupil center and width, a portion of the projected eye is saved as a template, whereas the width is extended a fractional ratio beyond the pupil to include a portion of the iris. (Generating this template occurs earlier, after the mask generation step, but is described here because it is more easily understood in the context of this step).

The normalized cross-correlation often has multiple prominent local maxima, due to the intensity changes in the pupil, eye corner, and any eye lashes or other features that intersect the region mask. For this reason we assume that the pupil locational difference between frames will be small, less than the width of the pupil itself. To help enforce this policy, a deadzone-gaussian penalty, as described previously, is applied to penalize local maxima that are distant from the pupil location in the previous frame. As with the frame registration procedure, a sub-pixel approximation is used. A simple quadratic equation is fitted to the 3 pixel peak region of the cross-correlation function, and fractional factors are stored and adjustments made.

This method does suffer from several theoretical drawbacks, but nonetheless appears to function as desired in testing. One factor that deserves additional attention is the geometry change the pupil and iris undergo in the image as the eye rotates, whereby the image of the pupil will shrink horizontally as the eye deviates from center. This phenomenon may possibly be accounted for using deformable template techniques. For a different approach, edges or peaks may be located with aid from a sub-pixel interpolation method such as the Cubic Convolution Interpolation [11], which fits several pixels at low computational cost.

From this stage, a pupil position is generated from the calculated position of the pupil relative to the mask, inclusive of any sub-pixel adjustments accumulated from frame registration or pupil location. The set of these measurements in time form a signal, which can be used for HGN classification.

III. Experimental Results and Analysis

The algorithm was developed and tested primarily in the MATLAB programming environment. A single video sequence was used, with minimal bias in selection or creation. Lighting was controlled only to ensure that picture intensity was reasonable, and that there were no strong reflections on the middle of the pupil. We argue this is a reasonable real-world operating condition, because the mobile device will block bright light sources directly in front of the eye, whereas there are likely to be bright light sources at high vertical angles above the user.

The video was produced using the factory Lenovo W520 Notebook PC webcam. The user was positioned in front of the PC display, in which the screen regions were kept dark as discussed previously, to reduce reflections. Two still images were displayed to position the eye for the two geometry/feature estimation steps, and then a square graphic was swept across the screen as discussed above. To simulate HGN conditions, a second video was played where the moving object randomly "froze" for short intervals before resuming the initial path. The sequence is carefully designed such that the path can be modeled as a constant sinusoidal frequency with added

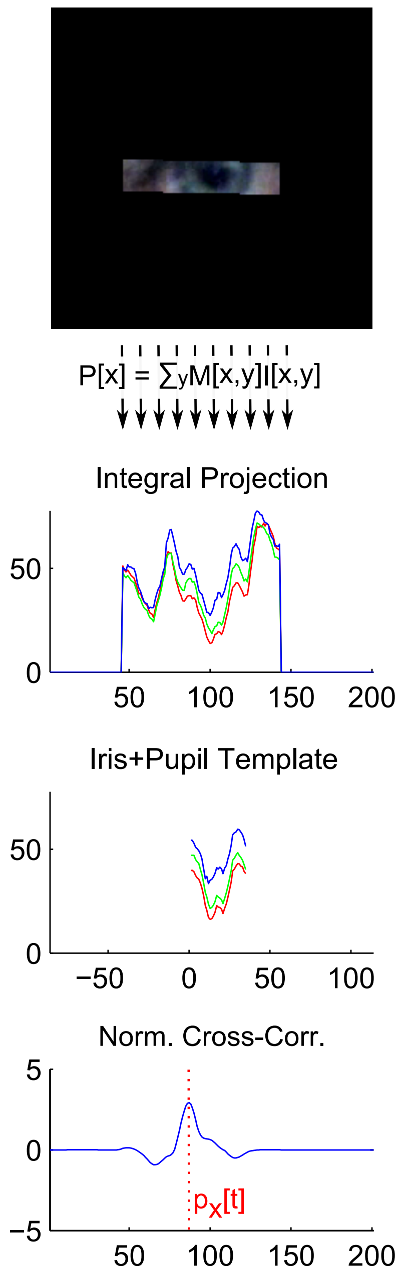


Fig. 7. Mask region of image is transformed by integral projection, then pupil position located using normalized cross-correlation. *(Color adjusted for print.)*

noise; phase shift does not accumulate in successive intervals of noise.

To test the ability of the algorithm to measure eye movement, a sequence of frames were hand-measured to determine the displacement between a point of the tear-duct, and an edge of the pupil. The mean was removed from the measurements performed by hand and our system, and the results were compared. See Fig. 8 and Table I. The results suggest that, on average, the algorithm detects position to within approximately 11 percent of the correct value, relative to the peak displacement. In producing the hand measurements it was determined

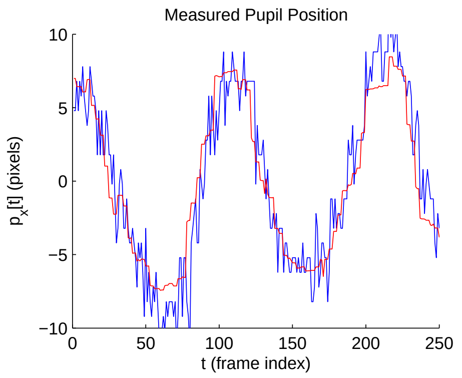


Fig. 8. Algorithmic measurements (red) and measurements made by a human (blue).

TABLE I. Pupil Position Output Analysis

Mean Normalized Absolute Difference	0.11
Maximum Normalized Absolute Difference	0.58

Normalized to the maximum absolute hand measurement, after subtracting the mean of measurements.

TABLE II. Pupil Trajectory THD+N Analysis

	Noiseless Stimulation	**Noisy Stimulation**
With Sub-Pixel Estimation	0.0253	0.0482
Without Sub-Pixel Estimation	0.0274	0.0498

that such measurements are very subjective, and suffer from low repeatability. We suggest further testing with an improved method, such as hand measurements at higher resolution for added precision, perhaps then downsampling the data prior to performing the proposed algorithm.

The eye position measurement results for both noiseless and noisy path stimulation videos are shown in Fig. 9. By visual inspection the “jerky” behavior is visible, but the signal is very noisy. To quantify the ability of the algorithm to detect the “jerky” behavior, a modified THD+N calculation was performed. This calculation measures the ratio of the power in frequencies higher than the fundamental, normalized by the fundamental sinusoidal signal power, assuming that an unimpaired individual would follow an ideal sinusoidal movement. See Table II. In the ideal case the ratio of the THD+N for a smooth signal and a HGN signal would be on the order of 10 or greater, which would lead to a very simple detector implementation. On the contrary, the measured ratio is on the order 2, and we suspect our simulated HGN signal is actually exaggerated and simpler to detect than an actual subject exhibiting HGN. On a positive note, it appears that the sub-pixel measurements gave a modest improvement in this regard, decreasing the THD+N measurement for the noiseless target test, with a relatively smaller change to the noisy target result. The calculation is relatively cheap, so we deem this a successful and worthwhile improvement, although more data is needed to statistically confirm this result.

Also visible in the data are discrete steps of relatively constant amplitude. Upon inspection of the video data it appears that the video compression method has removed much of the data between frames in small successive subsets. The difference in pixel values between successive frames follow a pattern of repeated sparseness except in a small percentage of pixel blocks, followed by an update frame where most blocks update. We believe we are losing significant motion fidelity due to this effect.

IV. Mobile Implementation and Challenges

While the bulk of experimentation and development of our algorithms was performed in MATLAB, exploring the feasibility of a mobile implementation was still one of our ultimate goals. We chose the Samsung Galaxy S3 as our target device as it is one of the most powerful and widely used smartphones on the market, and has a 2MP front facing camera. We employed heavy use of the OpenCV framework for the bulk of the image processing tasks. However we encountered many unforeseen difficulties on the mobile platform.

Initially we had hoped to perform this processing in real-time. This was desirable as it eliminates the need for either saving or buffering video information, eliminates compression artifacts, and it eliminates processing delay. Considerable effort was placed into making this possible, but eventually we decided to separate processing from data acquisition for initial testing. This in itself created new challenges; as it is, OpenCV on Android does not allow video encoding or support any kind of frame buffering (though support is planned in the future). Even more troublesome, the native camera API on Android makes it very difficult to record video in the background (in our case while the test sequence plays), and OpenCV/Android have no mechanism to decode videos. Some attempts were made to interface FFMPEG with OpenCV in JNI, but due to time constraints, lack of confirmed success stories, and lack of documentation, we eventually abandoned this.

Ultimately we found a way to overcome the challenges by simply storing camera frames in an array as they arrive, but this is obviously not optimal. To the best of our knowledge OpenCV/Android do not give access to camera frames before decompression, thus this buffering algorithm requires about 500Mb of RAM. Furthermore, the OS will not allow us to preallocate this buffer due to memory constraints, and copying frames to the buffer considerably reduces the captured frame rate. After recording we leave the buffer in memory for processing, and also save each frame as a JPEG so we can interface with our MATLAB code for verification. Despite all of this, one benefit to this processing is that it is untouched by motion estimation and can be used to test our previous hypothesis.

Due to time constraints the processing on the phone is still incomplete but it is coming along nicely. There are some variations from our MATLAB implementation detailed above. Most notably, instead of using a heuristic for tear-duct location we ask the user to verify it by touching the corner of the duct on the display. Apart from this major difference, the Android application follows the methods detailed in the previous sections.

V. Conclusions and Future Work

The algorithm presented successfully meets the intended goals of measuring horizontal gaze, using methods that are

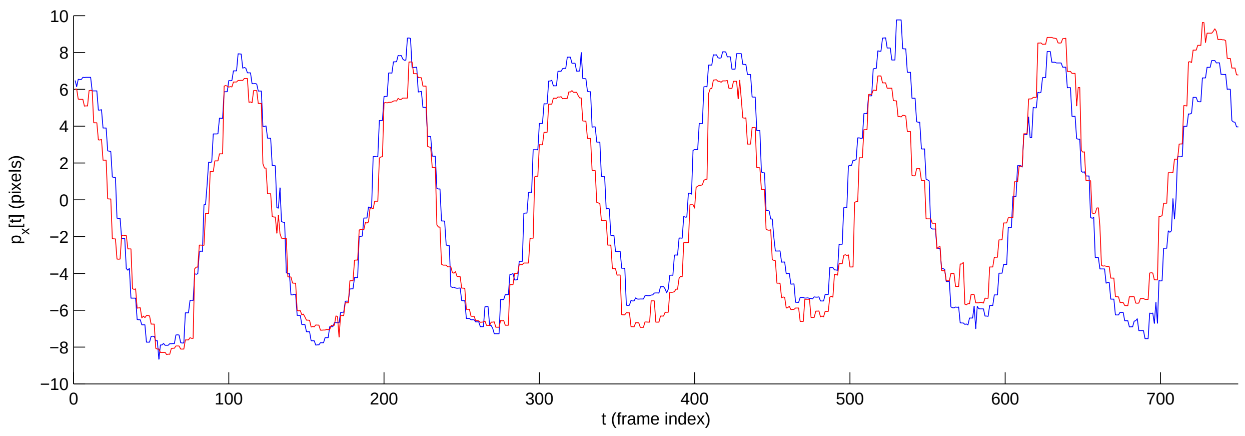


Fig. 9. Algorithmic measurements of pupil location for an eye tracking a sinusoidal signal (blue) and a noisy, or "jerky" sinusoidal signal (red).

computationally efficient for work on video frames. In the subjective opinion of the authors, the precision measured in our experimental analysis potentially falls short of the level required for an accurate and reliable assessment of alcoholic inebriation. It is unclear whether the low level of gaze measurement fidelity is due to issues with the test video, such as compression, or shortcomings in the proposed algorithm. Now that a basic proof of concept has been developed, a more rigorous test procedure is needed for further development.

While our current Android implementation is largely proof of concept, we can already use it to record more data samples for future testing. It may even be possible to crowd-source a test suite incorporating many different lighting conditions, faces, and levels of real-world HGN. From this test set proper analysis can be made, guiding improvements to the algorithm that may take form similar to the many proposed throughout this paper.

As a final note, despite a possible lack of fidelity, the algorithm itself may be useful in other contexts. Eye gaze tracking is a popular topic today, and the authors believe that the method of masking and using the Integral Projection to transform the data to a one-dimensional problem may be useful in other situations. While most other related problems involve much more change in facial pose, perhaps applying the masking and projection methods can be used on short subsequences of frames for a decrease in overall computation.

Appendix
Individual Author Contributions

Both members participated in brainstorming, testing and analysis, and final report to some degree.

Author J. Wilson was largely responsible for the PC implementation and development of the algorithm which was described throughout the paper. In addition, he performed considerable experimentation with other approaches to the problem. Lastly, he did the bulk of the poster design, as well as the non-Android associated sections of this report (by choice).

Author A. Avakov was responsible for the entirety the Android implementation. Initially this included experimentation with other various pupil detection/frame alignment methods, but these were not discussed here. Considerable time was required to learn Android and OpenCV. He also performed the frame annotation for manual algorithm verification, and wrote the Android related sections of this report.

Acknowledgment

The authors would like to thank Professor Bernd Girod and the teaching assistants David Chen and Matt Yu for teaching the course and providing the opportunity to do this project, to Keith Siilats for providing the project idea, an initial conceptual methodology (which was nearly completely revamped by the authors), and aid in setting up OpenCV Viola-Jones face detection in MATLAB, and to Andre Araujo for his recommendations regarding experimental validation.

References

[1] *Horizontal Gaze Nystagmus: The Science & The Law* [online]. Available: http://www.nhtsa.gov/people/injury/enforce/nystagmus/

[2] Thien et al. "Horizontal Gaze Nystagmus Detection in Automotive Vehicles" [Online]. Available: http://users.ece.cmu.edu/~nht/NystagmusDetectionReport.pdf

[3] N. Ostu, "A Threshold Selection Method From Gray-level Histogram," in *IEEE Transactions on Systems, Man and Cybernetics*, 1979, 9(1), pp. 62-66.

[4] Tian et al., "Eye-State Action Unit Detection by Gabor Wavelets," in *Advances in Multimodal Interfaces*, Beijing, China, 2000, pp. 143-150.

[5] J. Zhu and J. Yang, "Subpixel Eye Gaze Tracking," in *Fifth IEEE Int. Conf. Automatic Face and Gesture Recognition*, Washington, DC, 2002, pp. 124-129.

[6] M. Castrillón-Santana et al., "Face and Facial Feature Detection Evaluation," in *Third Int. Conf. Computer Vision Theory and Applications*, 2008.

[7] P. Kuo and J. Hannah, "An Improved Eye Feature Extraction Algorithm Based on Deformable Templates," in *IEEE Int. Conf. Image Processing*, 2005, Vol. 2, pp. II-1206.

[8] F. Timm and E. Barth, "Accurate Eye Centre Localisation by Means of Gradients," in *Proc. Int. Conf. Computer Theory and Applications*, Algarve, Portugal, 2011, pp. 125-130.

[9] G. G. Mateos, "Refining Face Tracking with Integral Projections," in *E-Commerce and Web Technologies: 4th Int. Conf.*, Prague, Czech Republic, 2003, vol. 3690, pp. 360-368.

[10] Y. Lv et al., "Sub-pixel Surface Fitting Algorithm in Digital Speckle Correlation Method," in Int. Conf. Electronic Measurement & Instruments, 2009 © IEEE. doi: 10.1109/ICEMI.2009.5274751

[11] R. Keys, "Cubic Convolution Interpolation for Digital Image Processing," in *IEEE Transactions on Acoustics, Speech and Signal Processing, 29, no. 6*, 1981, pp. 1153-1160.