UC Davis
UC Davis Previously Published Works

Title
Real-time Pupil Localization Using 3D Camera

Permalink
https://escholarship.org/uc/item/5t72t73q

Author
Liu, Yuanzheng

Publication Date
2016-07-01

Data Availability
The data associated with this publication are not available for this reason: Not intended to be public unless it's mandatory by UC Davis.
Real-time Pupil Localization Using 3D Camera

Yuanzheng Liu
May 24, 2016

1 INTRODUCTION

The localization of human eye pupils is important on many occasions, such as human-computer interaction, and gaze estimation. Tobii group built a 2D camera based eye tracker that helps people who completely paralyzed from the neck down interacting with the real world simply with their sight movements [1].

In some Virtual Reality (VR) environments such as Oculus Rift and HTC Vive, the system requires its user to use a headset for head interaction. However, the headset usually works with a cable, which may restrict the active area and may cause user discomfort. Moreover, the VR systems usually provide unreal scenes by ignoring or forcing users’ focus because of not knowing the sight direction. In some immersive Virtual Reality environments such as UC Davis KeckCAVES, the pupil localization could improve users’ immersive experience, and the
pupil could be a third precise interaction tool besides two hands.

In this project, I built a real-time 3D eye pupil localization system using an Intel RealSense R200 3D camera. The system can detect multiple faces within +/-30° of yaw, +/-30° of pitch, and +/-30° of roll. Then the system can localize pupils of detected faces from 0.7m to 1.5m of the camera.

2 PRIOR WORK

2.1 PUPIL LOCALIZATION

During the last two decades, several algorithms have addressed image-based pupil detection. Perez et al. [8] threshold the image and compute the mass center of the resulting dark pixels. The Starburst algorithm [6] removes the corneal reflection and then locates pupil edge points using an iterative feature-based approach. Long et al. [7] downsample the image and search there for an approximate pupil location. Timm et al. [10] localize the pupil by calculating the location where the most gradient vectors intersect. Agustin et al. [9] threshold the input image and extract points in the contour between pupil and iris, which are then fitted to an ellipse to eliminate possible outliers. The methods above have shown acceptable results in laboratory conditions. In this project, I used a project based on the method of Timm et al. [10].

3 METHOD

The system shown as Figure 3.1 works in the following steps:

- 1. pre-processing: Get stream images from the depth camera and the color camera. Set image ROI by depth image, rotate image independently for the following steps.
• 3. pupil detection: Based on the detected eyes, use eyeLike [4] to detect pupils in color images.

• 4. pupil localization: Based on the detected pupils in color images, find the corresponding positions in depth images, and the 3D positions in the environment.

• 5. post-processing: Output the 3D position of pupils. The results can be received by sockets.

Figure 3.1: The main structure of this project
4 DETAILS AND EXPLANATIONS

4.1 PRE-PROCESSING

By using the libRealSense [3], the system retrieves depth frame, color frame, and depth aligned to color frame (a synthetic stream containing depth data but sharing intrinsics of color stream). All streams are with 640*480 resolution and 30 fps.

Because the depth aligned to color frame contains depth information while sharing the same coordinates as color intrinsics, I use this frame to restrict the ROI of color frame. The new ROI is set to the smallest rectangle which contains all depth pixels from 0.5m to 3.0m.

OpenCV provides a Haar Cascades method[5] to detect faces. It has a good result for faces within +/-30° of yaw and +/-30° of pitch, but it can only detect faces within +/-20° of roll. To detect most faces of reasonable degrees of rolls, at least +/-40° of roll is necessary. Therefore, I independently run faces and eyes detection for the same frame 3 times with different degrees of rotations, which are 0°, -20° and 20°.

4.2 FACE AND EYE DETECTION

For each independent detection process, the eye detection runs only if faces are detected. It is possible that only one eye is detected for a face. But in my project, this condition will not be considered as a successful detection, the results will not be stored for future processing. The results are saved as "raw objects".

For each independent detection process, after all the faces and eyes are detected, I run a "rotate back" function to transform locations of detected raw objects into the correct locations of the original image. Then for the detected faces and their corresponding eyes, I save them as
"result objects". It is possible that some faces are detected in multiple detection processes. If a newly detected face overlaps with an already existing face, the new face will be discarded.

4.3 PUPIL DETECTION

In my project, after the face and eye detection, I process each detected eye's image using the last step of eyeLike algorithm, which is the pupil detection step. The last step of eyeLike uses the means of gradients method. The method could always return a 2D point, which is considered as the detected pupil. I only use this step of eyeLike because the eyes have already been detected. And the accuracy will be discussed in the following section.

4.4 PUPIL LOCALIZATION

Pupil detection step provides pupils' 2D locations of color image. Unfortunately, R200 doesn't provide a direct way to convert 2D locations of color image to 3D locations of depth image. But it provides a way of converting 3D locations of depth image to 2D locations of color image. Therefore, for pixels in depth image, I retrieve their locations of color image and build a look-up table. Then the look-up table provides a direct way to retrieve 3D locations of depth image from 2D locations of color image.

However, some pixels in color image can't be retrieved because there is no depth pixel matching. And some pixels' values in depth image are 0, which indicates the depth of that point be detected by the camera (such as background, object edges and pupils). For each pupil, especially for the pupil which has no corresponding depth data, I use the following algorithm to search two "reference points" instead:

• For a detected pupil point \((c_x, c_y)\) in color image:

• If it has corresponding depth value, save the 3D point twice as reference points, stop the algorithm.
• If it doesn’t have depth value, retrieve depth values for two points \((c_x - dx, c_y - dy)\) and 
\((c_x + dx, c_y + dy)\) (\(d\) indicates distance).
  
  – Firstly, set \(dx = 1, dy = 0\) and retrieve, which means retrieve pupil’s left and right pixel with distance 1.
  
  – If both points have corresponding depth value, save the two corresponding 3D points as reference points, stop the algorithm.
  
  – If one or both of the points doesn’t have corresponding depth value, set \(dx = 0, dy = 1\), which means retrieve pupil’s top and bottom pixel with distance 1.
  
  – Repeat retrieving by setting \((dx = d, dy = 0)\) and \((dx = 0, dy = d)\) when \(d = 1, 2, ..., 10\), until both points have corresponding depth value or \(d\) reaches 10 (which is a pre-defined max search distance).
  
• If \(d\) reaches 10 but no reference pairs are found, tell the system that the pupil’s 3D location cannot be detected, stop the algorithm.

For each pupil, when its reference points are found, the pupil’s actual 3D location is the median of its reference points.

4.5 POST-PROCESSING

All detected pupils’ 3D locations are stored in a vector which contains 3D points with format `rs::float3`. Then the message of points’ information is sent by windows socket. The message is a char array named ”SendBuffer” with a maximum size of 260. The format is as follows:

• `SendBuffer[0]` : an integer \(n\) which indicates the number of 3D points are send.

• `SendBuffer[1 + i * 12]` to `SendBuffer[4 + i * 12]` : a float number which indicates the \(x\) coordinate of the \(i^{th}\) 3D point. \((i \in [0,...,n-1])\).

• `SendBuffer[5 + i * 12]` to `SendBuffer[8 + i * 12]` : a float number which indicates the \(y\) coordinate of the \(i^{th}\) 3D point. \((i \in [0,...,n-1])\).
• SendBuffer[9 + i * 12] to SendBuffer[12 + i * 12]: a float number which indicates the z coordinate of the \(i^{th}\) 3D point. \((i \in [0, ..., n - 1])\).

• SendBuffer[1 + n * 12]: ‘\0’, end of a char array.

5 RESULTS AND ANALYSIS

Because the system requires a particular device, I didn’t have plenty of volunteers to have their faces tested. I had 5 volunteers tested for several conditions. In each condition, I captured 100 frames per person, then calculated the average accuracy of face and eye detection, and the average accuracy of pupil localization. In other words, for each condition, the number of captured images is 500 (100 image per volunteer * 5 volunteers). The accuracy of face and eye detection comes from a proportion of images where faces and eyes are correctly detected with all images captured. The accuracy of pupil localization comes from a proportion of images where pupils are correctly localized with all images captured.

Some conditions with successful face detections are shown as Figure 5.1, including basic condition (the top-left image), nearly 30° pitch (the top-right image), nearly 40° roll with nose and mouth covered (the bottom-left image), and nearly 30° roll with nearly 30° yaw (the bottom-right image). Some conditions with unsuccessful face detections are shown as Figure 5.2, including nearly -50° roll (the left image) and nearly -40° pitch with mouth opened (the right image).

5.1 BASIC CONDITION

In this condition, all test faces are frontal with normal expression. The yaw, pitch and roll are all 0 degrees. The distance from the camera is 1.0m. The results are shown as Table 5.1.

Although the accuracy of eye detection is 100%, there is still an accuracy loss for pupils. The
Figure 5.1: Detected faces, eyes and pupils

Figure 5.2: Failure of detection
pupil detection method sometimes returns the eyebrow or the lower orbital instead, because the shade caused by eyelid makes it difficult to distinguish the pupil and the sclera.

<table>
<thead>
<tr>
<th></th>
<th>Face accuracy</th>
<th>Eye accuracy</th>
<th>Pupil accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100.0%</td>
<td>100.0%</td>
<td>93.4%</td>
</tr>
</tbody>
</table>

Table 5.1: Average accuracy of the basic condition

5.2 Expressions

In this condition, all test faces are frontal. The yaw, pitch and roll are all 0 degrees. The distance from the camera is 1.0m. The results are shown as Table 5.2.

The accuracy drops significantly for small eyes or the eyes are lightly closed. It is sometimes because the eyes are hard to detect. Even if the eyes are detected, the pupil detection method would be more likely to return the eyebrow or the lower orbital instead.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Face accuracy</th>
<th>Eye accuracy</th>
<th>Pupil accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal expression</td>
<td>100.0%</td>
<td>100.0%</td>
<td>93.4%</td>
</tr>
<tr>
<td>Eyes lightly closed</td>
<td>100.0%</td>
<td>80.8%</td>
<td>35.0%</td>
</tr>
<tr>
<td>Mouth opened</td>
<td>100.0%</td>
<td>100.0%</td>
<td>94.6%</td>
</tr>
</tbody>
</table>

Table 5.2: Average accuracy with different expressions

5.3 Distances

In this condition, all test faces are frontal with normal expression. The yaw, pitch and roll are all 0 degrees. The distances shorter than 0.7m are not tested, because the depth camera cannot retrieve actual depth values correctly. The distances longer than 2.0m are not tested, because the face resolution of color image is too small to be detected. The results are shown as Table 5.3.
The pupil accuracy is better when the eye is closer to the camera. When the face is farther from the camera, the loss of resolution causes the low accuracy of both face and eyes.

<table>
<thead>
<tr>
<th>Distance</th>
<th>Face accuracy</th>
<th>Eye accuracy</th>
<th>Pupil accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7m</td>
<td>100.0%</td>
<td>100.0%</td>
<td>97.6%</td>
</tr>
<tr>
<td>1.0m</td>
<td>100.0%</td>
<td>100.0%</td>
<td>93.4%</td>
</tr>
<tr>
<td>1.3m</td>
<td>100.0%</td>
<td>97.8%</td>
<td>47.6%</td>
</tr>
<tr>
<td>1.6m</td>
<td>100.0%</td>
<td>31.0%</td>
<td>12.4%</td>
</tr>
<tr>
<td>1.9m</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

Table 5.3: Average accuracy with different distances

5.4 Facial angles

In this condition, all test faces are with normal expression. The distance from the camera is 1.0m. The results are shown as Table 5.4, Table 5.5 and Table 5.6. Because of the difficulty in controlling specific yaws, rolls and pitches, I used the "Compass" app in iPhone for reference. The basic condition varies by each volunteer. For each volunteer, I adjust the normal of the iPhone's screen to be parallel with the normal of the volunteer's face. With the movement of the volunteer's head, I simulate the iPhone to the degree I need to test. Because of the measurement method, the angles are not precise in this report.

When the yaw reaches +/-30°, the eye which is farther from the camera is blocked by the nose, causing the significant drop of eye accuracy. When the yaw reaches +/-40°, only half of the face are captured by the camera. Then the face detection method is not working properly.

When the pitch reaches +/-30°, the detection of faces and eyes are good enough. However, because of the shade caused by eyelid, the localization of pupils is still poor.
<table>
<thead>
<tr>
<th>Yaw</th>
<th>Pitch</th>
<th>Roll</th>
<th>Face accuracy</th>
<th>Eye accuracy</th>
<th>Pupil accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0°</td>
<td>0°</td>
<td>0°</td>
<td>100.0%</td>
<td>100.0%</td>
<td>93.4%</td>
</tr>
<tr>
<td>+/-10°</td>
<td>0°</td>
<td>0°</td>
<td>100.0%</td>
<td>100.0%</td>
<td>94.4%</td>
</tr>
<tr>
<td>+/-20°</td>
<td>0°</td>
<td>0°</td>
<td>100.0%</td>
<td>100.0%</td>
<td>93.6%</td>
</tr>
<tr>
<td>+/-30°</td>
<td>0°</td>
<td>0°</td>
<td>88.8%</td>
<td>26.0%</td>
<td>8.6%</td>
</tr>
<tr>
<td>+/-40°</td>
<td>0°</td>
<td>0°</td>
<td>24.2%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

Table 5.4: Average accuracy with different yaws

<table>
<thead>
<tr>
<th>Yaw</th>
<th>Pitch</th>
<th>Roll</th>
<th>Face accuracy</th>
<th>Eye accuracy</th>
<th>Pupil accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0°</td>
<td>0°</td>
<td>0°</td>
<td>100.0%</td>
<td>100.0%</td>
<td>93.4%</td>
</tr>
<tr>
<td>0°</td>
<td>+/-10°</td>
<td>0°</td>
<td>100.0%</td>
<td>100.0%</td>
<td>92.6%</td>
</tr>
<tr>
<td>0°</td>
<td>+/-20°</td>
<td>0°</td>
<td>100.0%</td>
<td>100.0%</td>
<td>92.0%</td>
</tr>
<tr>
<td>0°</td>
<td>+/-30°</td>
<td>0°</td>
<td>98.4%</td>
<td>98.4%</td>
<td>81.2%</td>
</tr>
<tr>
<td>0°</td>
<td>+/-40°</td>
<td>0°</td>
<td>67.6%</td>
<td>63.4%</td>
<td>38.6%</td>
</tr>
</tbody>
</table>

Table 5.5: Average accuracy with different pitches

For the original Haar Cascade face detection method, faces only within +/-20° are detected. This range is not acceptable for all reasonable rolls of faces. My system has a valid range of +/-30°. The low accuracy on +/-40° is because the roll degree after rotation is still too large to detect faces.

5.5 Obstacles

In this condition, all test faces are frontal with normal expression. The yaw, pitch and roll are all 0 degrees. The distance from the camera is 1.0m. The results are shown as Table 5.7. For pupil accuracy of the "wearing glasses" item, the pupil location is represented as the pupil...
<table>
<thead>
<tr>
<th>Yaw</th>
<th>Pitch</th>
<th>Roll</th>
<th>Face accuracy</th>
<th>Eye accuracy</th>
<th>Pupil accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0°</td>
<td>0°</td>
<td>0°</td>
<td>100.0%</td>
<td>100.0%</td>
<td>93.4%</td>
</tr>
<tr>
<td>0°</td>
<td>0°</td>
<td>+/-10°</td>
<td>100.0%</td>
<td>100.0%</td>
<td>92.4%</td>
</tr>
<tr>
<td>0°</td>
<td>0°</td>
<td>+/-20°</td>
<td>100.0%</td>
<td>100.0%</td>
<td>94.0%</td>
</tr>
<tr>
<td>0°</td>
<td>0°</td>
<td>+/-30°</td>
<td>100.0%</td>
<td>100.0%</td>
<td>93.4%</td>
</tr>
<tr>
<td>0°</td>
<td>0°</td>
<td>+/-40°</td>
<td>66.0%</td>
<td>63.6%</td>
<td>58.4%</td>
</tr>
<tr>
<td>0°</td>
<td>0°</td>
<td>+/-50°</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

Table 5.6: Average accuracy with different rolls

image on the lens, but not the pupil's actual location.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Face accuracy</th>
<th>Eye accuracy</th>
<th>Pupil accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>No obstacles</td>
<td>100.0%</td>
<td>100.0%</td>
<td>93.4%</td>
</tr>
<tr>
<td>Mouth covered by a hand</td>
<td>100.0%</td>
<td>100.0%</td>
<td>94.4%</td>
</tr>
<tr>
<td>Mouth and nose covered by a hand</td>
<td>100.0%</td>
<td>100.0%</td>
<td>93.8%</td>
</tr>
<tr>
<td>One eye covered by a hand</td>
<td>75.8%</td>
<td>75.8%</td>
<td>63.8%</td>
</tr>
<tr>
<td>Wearing glasses</td>
<td>100.0%</td>
<td>96.7%</td>
<td>64.2%</td>
</tr>
<tr>
<td>Wearing sunglasses</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

Table 5.7: Average accuracy with different obstacles

The results of obstacles on the nose and the mouth imply that the eyes play an important role in face detection. And the result of item "one eye covered by a hand" confirms that the eye is necessary for Haar Cascade face detection. This system uses the Haar Cascade classifier which could detect faces with glasses. Both the face and eye accuracy are high when people are wearing glasses. But the glasses could drop the pupil accuracy because the depth camera could be noised by the lens.
5.6 SIGHT DIRECTIONS

In this condition, all test faces are frontal with normal expression. The yaw, pitch and roll are all 0 degrees. The distance from the camera is 1.0m. The results are shown as Table 5.8. It shows the eyeLike method in [4] works well for the pupil localization.

<table>
<thead>
<tr>
<th>Sight direction</th>
<th>Face accuracy</th>
<th>Eye accuracy</th>
<th>Pupil accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center</td>
<td>100.0%</td>
<td>100.0%</td>
<td>93.4%</td>
</tr>
<tr>
<td>Left</td>
<td>100.0%</td>
<td>100.0%</td>
<td>92.8%</td>
</tr>
<tr>
<td>Right</td>
<td>100.0%</td>
<td>100.0%</td>
<td>93.0%</td>
</tr>
<tr>
<td>Top</td>
<td>100.0%</td>
<td>100.0%</td>
<td>93.0%</td>
</tr>
<tr>
<td>Bottom</td>
<td>100.0%</td>
<td>100.0%</td>
<td>94.2%</td>
</tr>
</tbody>
</table>

Table 5.8: Average accuracy with different sight directions

5.7 SYSTEM ENVIRONMENT

The detection time per frame of this system is shows as Table 5.9. It is three times as the time of original Haar Cascade method as a result of running face detection for same frame three times. The detection time is longer when faces are detected because the Haar Cascade method runs its detection for a longer process to confirm the face. However, when the roll degree of face is large enough, the face can only be detected by one of the three rotated frames of the same frame. In this case, the detection time is about 60ms faster because the Haar Cascade method works individually each time.

This system is built and executed on a desktop computer with a 64-bit Windows 10 operating system, Intel i5-4690K @ 3.50GHz CPU, 8.00 GB memory and a NVIDIA GTX 770 graphics card. Because both the libRealSense and the OpenCV is cross-platform, this system is also usable for Linux system if using the Linux socket.
Table 5.9: Detection time per frame

<table>
<thead>
<tr>
<th>Method</th>
<th>Detection times</th>
<th>Face not detected</th>
<th>Face detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Haar Cascade</td>
<td>1</td>
<td>32.5ms</td>
<td>60.9ms</td>
</tr>
<tr>
<td>The system in this report</td>
<td>3</td>
<td>93.6ms</td>
<td>181.5ms</td>
</tr>
</tbody>
</table>

6 DISCUSSION

6.1 FACE AND EYE DETECTION

The face detection in this system depends on the OpenCV Haar Cascades method. According to [11], it has 94.1% accuracy on all tested images, which had frontal faces with different backgrounds and environments. However, my system is designed to be used indoor for faces with a wide range of yaw, pitch and roll degrees.

The eye detection method in OpenCV is Haar Cascades with the eyes training sets. It is only executed when a face has been detected. There is no official document of its experimental accuracy. In my system, the accuracy depends on facial angles, and the results turn out to be acceptable.

6.2 PUPIL DETECTION AND LOCALIZATION

The pupil detection in this system is based on means of gradients, which requires high resolution images of eyes. It is executed only after eyes have been detected. The overall pupil accuracy for detected eyes are around 93% because the resolution and image quality of the camera makes the system difficult to get high resolution images. But for some conditions, the accuracy is lower because the shade caused by eyelid could fool the pupil detection. However, the overall results are still acceptable.
7 Future Work

The performance of this system may be improved in the following ways:

- Multi-threading. Now this system is detecting the same frame three times in one thread. Because the Haar Cascade detection is individual for each image, multi-threading can make the system run much faster.

- Higher resolution. The system is using 640*480 on both color and depth camera. If the resolution of color image is 1920*1080, the eyes are more easily detected. However, this will cause the drop of speed.

- Using the depth information to minimize the face detection area, such as background extraction. In practice, a large amount of area in the camera is background. By only detecting the foreground, the detection time will be significantly improved.

- Detecting face landmarks using both color and depth information. The 3D shape of the face can be easily distinguished from other shapes. This may be a faster way to detect faces.

8 Terminology

8.1 The Haar Cascades Method in OpenCV

The Haar Cascades object detection method is a widely-used object detection framework to provide competitive object detection rates in real-time proposed by Paul Viola and Michael Jones [11]. OpenCV provides an interface of the Haar Cascades method to detect various kinds of objects including human faces.

8.2 Means of Gradients and EyeLike

Means of gradients is an approach for accurate and robust eye center localization by using image gradients [10]. It has a simple objective function, which only consists of dot products. The
maximum of this function corresponds to the location where most gradient vectors intersect and thus to the pupil.

EyeLike [4] is an OpenCV based webcam gaze tracker based on a means of gradients pupil localization method. However, the eyeLike does not track gaze by the time of this report finished. According to its author Tristan Hume, it is basically just a developer reference implementation of Fabian Timm's algorithm that localizes human pupils from color images. Originally it runs OpenCV Haar Cascade method to detect faces. Then it estimates the locations of eyes by experience area of detected faces, which has less accuracy than Haar Cascade eye detection. For each detected or estimated eyes, eyeLike uses means of gradients to find the pupil center, which is the last step of the eyeLike algorithm. EyeLike also has several post-processing steps which are trivial for my project.

8.3 REGION OF INTEREST

A region of interest (ROI) is a selected subset of samples within a dataset identified for a particular purpose [2]. In computer vision, the ROI defines the borders of an object under consideration. In this report, the ROI is a rectangle area of an image for further processing.

8.4 LIBREALSENSE

Intel RealSense Cross Platform API [3], also known as libRealSense, is a cross-platform library (Linux, OSX, Windows) for capturing data from the Intel RealSense F200, SR300 and R200 cameras. It was initiated to better support researchers, creative coders, and app developers in domains such as robotics, virtual reality, and the internet of things. Several often-requested features of RealSense devices are implemented in this project, including multi-camera capture.
REFERENCES

[1] Tobii AAC. Stephen murray and his tobii pceye go - focus on the game. https://www.youtube.com/watch?v=2zh2UMK8xf0.


[9] Javier San Agustin, Henrik Skovsgaard, Emilie Mollenbach, Maria Barret, Martin Tall, Dan Witzner Hansen, and John Paulin Hansen. Evaluation of a low-cost open-source gaze
