Eye Tracking for Target Acquisition in Sparse Visualizations

Feiyang Wang
feiyang.wang@ontariotechu.ca
Ontario Tech University
Oshawa, Ontario

Adam James Bradley
adam.bradley@ontariotechu.ca
Ontario Tech University
Oshawa, Ontario

Christopher Collins
christopher.collins@ontariotechu.ca
Ontario Tech University
Oshawa, Ontario

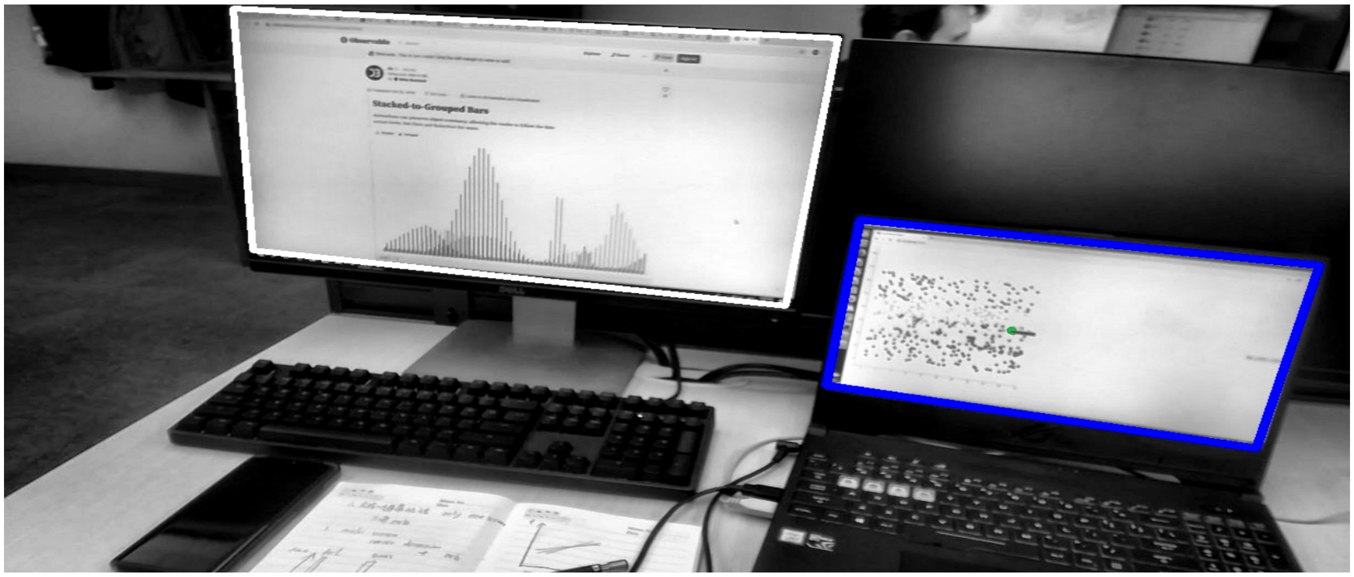

Figure 1: Our method is able to discern between multiple screens in space with low computational overhead and recognize complex visualizations with large amounts of white space (a challenge when using head-mounted eye tracking glasses with visualizations). The blue frame shows our screen detection, and the green dot shows the gaze point.

ABSTRACT

In this paper, we present a novel marker-free method for identifying screens of interest when using head-mounted eye tracking for visualization in cluttered and multi-screen environments. We offer a solution to discerning visualization entities from sparse backgrounds by incorporating edge-detection into the existing pipeline. Our system allows for both more efficient screen identification and improved accuracy over the state-of-the-art ORB algorithm.

CCS CONCEPTS

• **Human-centered computing → HCI design and evaluation methods**; **HCI design and evaluation methods**; • **Computing methodologies** → *Image processing*.

KEYWORDS

HCI, computer vision, gaze detection, Multi-user

ACM Reference Format:
Feiyang Wang, Adam James Bradley, and Christopher Collins. 2020. Eye Tracking for Target Acquisition in Sparse Visualizations. In *Symposium on Eye Tracking Research and Applications (ETRA '20 Short Papers), June 2–5, 2020, Stuttgart, Germany.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3379156.3391834

1 INTRODUCTION

In the field of eye-tracking research, gaze target acquisition when using head-mounted eye tracking glasses in a multi-screen environment is a known challenge [Mardanbegi and Hansen 2011]. In this workflow, to locate gaze on a screen first the world-view camera image is used to determine the location and identity of the screen in the field of view, and the eye camera is used to determine the gaze point within the detected screen. Determining which display screen is the one of interest in cluttered, multi-screen environments in which the user is moving is the error-prone and rate-limiting step. In response to this, state-of-the-art systems use one of two approaches. The first is calibrating the eye tracker to the corners of the screen using ArUco markers (a synthetic square marker composed by a wide black border and an inner binary matrix which determines its identifier) so that the eye tracker can easily identify the space of interest and know where the borders of the screen are [Garrido-Jurado et al. 2014]. The second is the ORB (Oriented FAST and Rotated BRIEF) feature detection algorithm which can

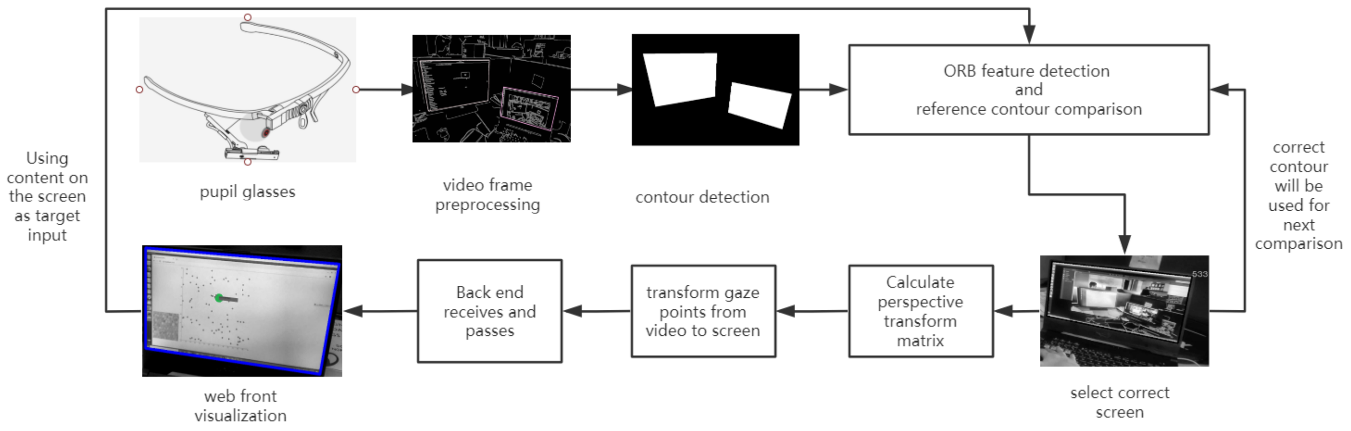
Using content on the screen as target input
pupil glasses
video frame preprocessing
contour detection
ORB feature detection and reference contour comparison
correct contour will be used for next comparison
select correct screen
Calculate perspective transform matrix
transform gaze points from video to screen
Back end receives and passes
web front visualization

Figure 2: The flowchart of our pipeline. The eye-tracking glasses provide video frame and gaze point data to our program. Various filters from OpenCV are used to preprocess the video frames and detect screen contours. Our core module ORB feature detection and reference contour comparison can detect the correct screen. The identified screen is used for both calculating the homography matrix and the next frame detection. With the matrix, gaze points are transformed from 2D video plane to 2D screen plane. Finally, our back end server passes the gaze points to our web front end.

extract unique key points from a target image and match them with key points from the interest area of user's gaze [Rublee et al. 2011]. When it comes to using the ORB with information visualizations, the sparse plotting and often white backgrounds cause problems for target acquisition. One solution is to place a textured pattern behind the visualization so that the ORB algorithm has sufficient visual features to discern the screen accurately. However, this approach can interfere with the visualization design.

In this paper, we contribute a method for multi-screen detection that allows users to forego ArUco markers and background textures in their visualizations (see Figure 1). This significantly improves interaction times by removing a major step at the beginning of an engagement (marker registration) and allows visualizations to appear as designers intend without the compromise of manipulated background textures. Collaborative environments where more than one person is working around multiple screens can also benefit from this method.

2 RELATED WORK

There is a large unexplored design space at the intersection of visualization and eye tracking because of technological constraints. Several projects have addressed this crossover space in previous work, but all have been hampered by these roadblocks. Dostal et al. developed a system for multi-user markerless real-time eye tracking on a big screen [Dostal et al. 2014] and Zhang et al. have studied spontaneous interaction between users and displays in public spaces by directing users into the tracking range [Zhang et al. 2014]. Both of these systems suffer from only being able to detect users when they stand separately facing the screen and react based on the user's proximity to the display. In this instance, the eye detection is used to make sure the user is looking at the screen but is not used to determine exactly where they are focused. To enable full marker-free yet precise gaze tracking for information visualization evaluation and interaction, we need reliable screen and gaze point detection, ideally for multiple people and screens. Our system can do just that.

Novel approaches at understanding different users and their gaze have been presented in the past but many need to calibrate using screen markers. Barz et al. [Barz et al. 2018], Jungwirth et al. [Jungwirth et al. 2018], Kasprowski et al. [Kasprowski and Harezlak 2018] and Zhang et al. [Zhang et al. 2016]. all use marker-based display detection in their work. This is a strong demonstration of the potential of this research area but is still limited by calibration issues. In our work, a visualization can be programmed to show different information to different users because the system knows which user is wearing which eye-tracking headset and where they are looking; this opens up new avenues for interaction based on individual user gaze.

3 METHODOLOGY

Our method uses a pipeline approach to process data from eye tracking glasses. The high-level overview is shown in Figure 2 and consists of preprocessing to determine candidate screen contours in the world-view, ORB detection to determine the correct screen, and transformations of the gaze point to visualization-relative coordinates.

3.1 Gaze position Detection and Calculation

For the eye-tracking software and hardware, Pupil Capture Software and Pupil Core were used [Kassner et al. 2014]. The Pupil Core has two cameras: the world-view camera records the scene in front of the user and the eye camera is pointed backwards to capture eye movement. We chose this type of input because our long term goal is a collaborative environment with multiple eye-tracked users to produce dynamic interactions. The eye position (pupil) is detected in each camera frame, using the Pupil Lab 2D algorithm, and the

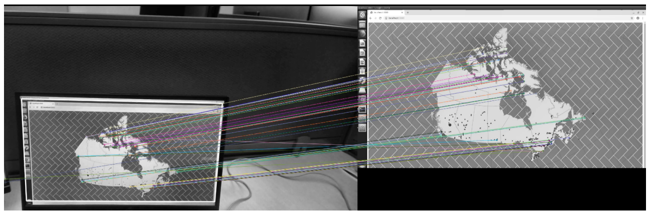

Figure 3: A successful example of ORB feature detection and matching with a textured background visualization.

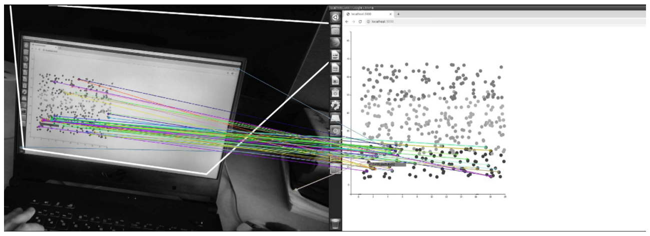

Figure 4: A failure example ORB feature detection and matching with a large white space visualization. Note the screen contour (white polygon) is incorrect.

gaze position in the world camera was estimated after a 5-points calibration. Both the world camera frame and the gaze position are then sent to an open port that our software is listening on.

3.2 Candidate Screen Detection

In order to transform the gaze point from coordinates in the world view image to coordinates relative to the target visualization, the contour of the correct screen needs to be detected. For this, we use image filters from OpenCV to pre-process each frame from the world camera with the following steps: 1. Reduce video frame noise and detail with Gaussian Blur (kernel size 3 × 3). 2. Detect a wide range of edges in the video frame with the Canny Edge filter. To counter various lighting environments, Canny Edge filter's threshold is adjustable when the system is working (in our case, with 3 as factor, and a threshold from 50 to 80 performs best). Here, the Canny Edge filter does not only ignore the boundary of small objects but also keeps edges of screens. 3. Dilate the edges with a morphological rectangle (4 × 4) to keep the integrity of contours since Canny Edge filter might fail to provide a complete edge of a object. 4. Find contours (a list of arrays that store coordinates of an edge). 5. Remove unqualified contours. Contours with less than 3 edges and over 5 edges are discarded in this step since our target is 4-edged screens. Contours with 3 or 5 edges are corrected to 4 edges with the `approxPolyDP` function. Detected contours are then passed to the next section. In addition, small contours with an area falling under a threshold percentage of the full world-camera video frame (currently 10%) are discarded as they are likely too small for the user to read.

3.3 ORB Feature Detection

It has been shown in previous implementations that the ORB algorithm works well when being used with photographs [Lander et al. 2015], but when we tested the algorithm using sparse visualizations such as scatterplots, the success rate dropped dramatically. ORB is basically a fusion of FAST (Features from Accelerated Segment Test) key point detection [Rosten and Drummond 2005, 2006] and BRIEF (Binary Robust Independent Elementary Features) descriptors [Calonder et al. 2010] with many modifications to enhance the performance. First, it uses FAST to find key points on both world camera video and an input stream of the target screen image, then applies the Harris corner measure to find the top N points among them (see Figure 3). The original algorithm suffers from an abundance of mismatches between the video frame and the actual screen when using information visualizations (see Figure 4). Those mismatches are not only detrimental to the accuracy of the gaze point transformation but also slow the speed of the software. In response to this, we designed a new method that uses ORB as a starting point. One of the main drawbacks of the ORB algorithm in this instance is that the system has a hard time deciphering images that have a lot of uniformity (e.g. blank backgrounds). This is compounded when visualizations are on the screen with many nodes that are not differentiated by size and color. To address this issue, our method is designed to restrict ORB detection to the contents of candidate contours only, reducing the possibility of spurious feature matches in the world camera video stream. This combination of computer vision and ORB has shown great potential for eye tracking and sparse visualizations.

3.4 Reference Contour Comparison

Computational cost is another problem with the original ORB algorithm, incurring latency in interactive scenarios. To reduce this processing overhead, our method compares the previous video frame's output contour to the current frame's candidate contours. If a candidate contour's central point is inside the last frame's output contour it is selected as a homography transformation and ORB detection is skipped. Otherwise, ORB feature detection is used to match candidate contours to detect the correct screen.

To prevent a compounding impact of an erroneous detection, ORB feature matching is run every 10 frames irrespective of the reference contour comparison. ORB matching overrides the reference contour matching when it runs. In testing, this was found to balance accuracy against computational demand. Our system forwards the contour boundaries of the detected screen in the frame to the next step in our pipeline.

3.5 Gaze Point Transformation

Since the user's gaze point is relative to the world camera video, we use a homography matrix [Szeliski 2010] to transform the user's gaze point from the 2D world camera video plane to the 2D screen plane given the corners of the screen contour in the frame.

Given the screen-relative gaze point, it is possible to identify which window the user is looking at on the correct screen in multi-screen environments. Thus the gaze information can be passed to the correct program, such as visualizations or other applications, for use populating interest models, gaze-interaction techniques, or other purposes. To process the real-time information we use a back end server to capture the time, the user's ID, and each individual user's gaze data.

Table 1: Accuracy of screen detection by method and setup.

	Dense Visualization		Sparse Visualization	
	Stable one screen	***Moving two screens***	***Stable one screen***	***Moving two screens***
ORB feature detection	87.1%	0%	13.6%	0%
Our method	99.7%	96.45%	96.9%	90.86%

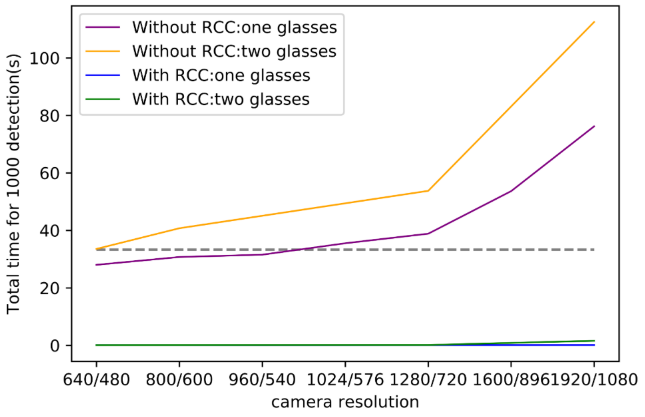

Figure 5: Timing results of our method with and without the reference contour check (RCC). The dotted gray line is the baseline for processing 30fps.

3.6 Front End Display

Our demo, shown in Figure 1, is built on an HTML front end with JavaScript. The demo listens and collects the gaze data from our back end server. Each object in the web front end is assigned an event handler to check if the user's gaze is on it or not. When users are looking at one object, the data model records this action, how long, when, which part of the visualization, and which user is looking at it. Also, each object is given an attribute of which user currently "owns" each object. This attribute gives us a lot of flexibility to design interactions between multiple users.

4 EVALUATION

To analyze the performance of our method we designed and ran an experiment to test the original ORB algorithm against our improved version using on-screen information visualizations.

The experiment varied three factors: 1. The content of the visualization on the screen (sparse visualization as in Figure 4 or dense visualization as in Figure 3). 2. The screen detection method (our method and ORB feature detection). 3. The viewing context (stable 1 screen or head and gaze moving slowly between 2 screens). In the 2 screen condition, the second screen showed a distractor bar chart visualization as shown in Figure 1. The measure for the experiment was the count of frames in which the target screen bounds were detected correctly (all four corners of the target screen aligned correctly). This was assessed manually by viewing each frame and counting the correct detections.

For each of the 4 conditions (one/two screen × sparse/dense visualization) we captured raw video and eye-tracking data of a user looking at the target visualization on WSL system. For each test, we captured 500 frames of data. This number of frames was found to be sufficient to consistently differentiate methods. The output from the Pupil cameras was post-processed twice: by our pipeline and the ORB algorithm alone.

As seen in Table 1 our method outperformed the original ORB algorithm while not requiring ArUco markers or the need to add a textured background to break up the abundance of white space. In particular, when the user's head was moving between the two screens, the ORB algorithm alone could not tell the difference between the screen with the active information and the distractor.

We then tested multiple resolutions with one and two users using our method to pinpoint the limits of the scalability of our approach. Starting without the reference contour check (RCC) (i.e., ORB on every frame), the results are shown in Figure 5. The performance decreased dramatically with resolutions above 1280/720 (time per frame increased). If we check the ORB output every frame the entire approach is simply too slow. Based on these results we added the reference contour comparison. Because users are often keeping their heads relatively still while looking at visualizations, we rely on this consistency to skip ORB detection and increase the overall speed of the system. The speed increase is visible in Figure 5 where the blue and green lines (with RCC) are significantly lower than the orange and purple ones (without RCC). Overall by using our method which combines edge detection, ORB, and the heuristic RCC, efficiency and accuracy are improved over the ORB alone.

Although using calibration markers is faster than our method, removing the extra calibration step is desirable for situations such as public visualizations where calibration is a barrier to engagement. Markers can also be visually distracting or aesthetically displeasing.

5 CONCLUSION

In conclusion, we have presented a method that outperforms the ORB algorithm in target acquisition on multiple screens without the need for ArUco markers. Our System can detect multiple user's gaze points in visualization-relative coordinates in real-time. It also can detect multiple screens and recognize the correct screen with our ORB feature detection and reference contour comparison module. Future work will be focused on the design space with this system. There are several things that can be done: 1. Custom information access for users. 2. Personal preference on style or representation on the object on the screen. 3. Awareness and conflict in collaboration with eye-tracking technology.

REFERENCES

Michael Barz, Florian Daiber, Daniel Sonntag, and Andreas Bulling. 2018. Error-aware gaze-based interfaces for robust mobile gaze interaction. In *Proc. ACM Symposium on Eye Tracking Research & Applications*. ACM, Article 24, 10 pages. https://doi.org/10.1145/3204493.3204536

Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. 2010. Brief: Binary robust independent elementary features. In *Proc. European Conference on Computer Vision*. Springer, 778–792.

Jakub Dostal, Uta Hinrichs, Per Ola Kristensson, and Aaron Quigley. 2014. SpiderEyes: Designing attention- and proximity-aware collaborative interfaces for wall-sized displays. In *Proc. ACM Conf. on Intelligent User Interfaces*. 143–152.

Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. 2014. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition* 47, 6 (2014), 2280–2292.

Florian Jungwirth, Michael Haslgrübler, and Alois Ferscha. 2018. Contour-guided gaze gestures: Using object contours as visual guidance for triggering interactions. In *Proc. ACM Symposium on Eye Tracking Research & Applications*. ACM, Article Article 28, 10 pages. https://doi.org/10.1145/3204493.3204530

Pawel Kasprowski and Katarzyna Harezlak. 2018. Comparison of mapping algorithms for implicit calibration using probable fixation targets. In *Proc. ACM Symposium on Eye Tracking Research & Applications*. ACM, Article Article 11, 8 pages. https://doi.org/10.1145/3204493.3204529

Moritz Kassner, William Patera, and Andreas Bulling. 2014. Pupil: An open source platform for pervasive eye tracking and mobile gaze-based interaction. In *Adjunct Proc. ACM Int. Joint Conf. on Pervasive and Ubiquitous Computing*. ACM, 1151–1160. https://doi.org/10.1145/2638728.2641695

Christian Lander, Sven Gehring, Antonio Krüger, Sebastian Boring, and Andreas Bulling. 2015. GazeProjector: Accurate gaze estimation and seamless gaze interaction across multiple displays. In *Proc. ACM Symposium on User Interface Software & Technology*. ACM, 395—-404. https://doi.org/10.1145/2807442.2807479

Diako Mardanbegi and Dan Witzner Hansen. 2011. Mobile gaze-based screen interaction in 3D environments. In *Proc. Conf. on Novel Gaze-Controlled Applications*. ACM, Article 2, 4 pages. https://doi.org/10.1145/1983302.1983304

Edward Rosten and Tom Drummond. 2005. Fusing points and lines for high performance tracking. In *IEEE Int. Conf. on Computer Vision*, Vol. 2. IEEE, 1508–1515.

Edward Rosten and Tom Drummond. 2006. Machine learning for high-speed corner detection. In *European Conf. on Computer Vision*. Springer, 430–443.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. ORB: An efficient alternative to SIFT or SURF. In *Proc. Int. Conf. on Computer Vision*. IEEE CS, 2564—-2571. https://doi.org/10.1109/ICCV.2011.6126544

Richard Szeliski. 2010. *Computer Vision: Algorithms and Applications*. Springer Science & Business Media.

Dan Zhang, Darius Coelho, and Klaus Mueller. 2016. Google Glass for personalized augmentations of data visualizations. In *IEEE Visualization (Posters)*.

Yanxia Zhang, Jorg Muller, Ming Ki Chong, Andreas Bulling, and Hans Gellersen. 2014. GazeHorizon: Enabling passers-by to interact with public displays by gaze. In *Proc. ACM Int. Joint Conf. on Pervasive and Ubiquitous Computing (Ubicomp)*. 559–563.