Achieving Vergence for a Binocular Vision System

Pranathi Byappanahalli Suresha
Capstone Project Final Report Spring 2024
Advisor: Prof. Shreyas Kousik
M S Robotics, Georgia Institute of Technology

***Abstract*—Binocular vision is the ability to see with both the eyes simultaneously and perceive the three dimensional real world by combining the two dimensional views obtained from the eyes. When the eyes focus on a target object and when we consider the pair of images from both the eyes at this point, combining the image pairs reveals the object overlapping on the resultant image. When this happens, we say that the visual system has achieved vergence. We can mimic this behaviour of the binocular vision system by using two cameras in the place of our eyes. In this article, we propose a method to achieve vergance and formulate a vergence metric to measure vergence. Through experiments in simulation and hardware we show how vergenace can be achieved and how the images overlap perfectly when vergence is achieved.**

***Index Terms*—Vergence, Fixation point, Binocular vision**

I. Introduction

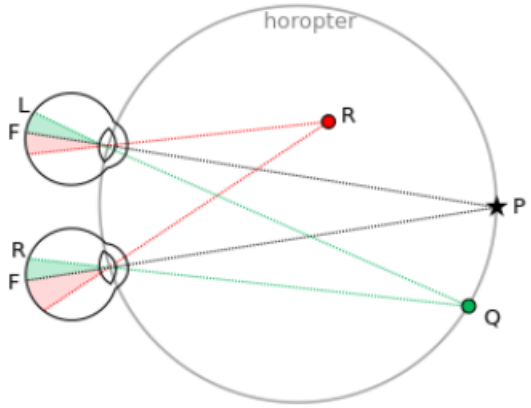
Binocular stereo vision is a branch of computer vision which involves using two cameras at the same time to obtain the three dimensional (3D) information of the scene [1]. It is based on the same principle that pertains to the human stereo vision. Human brain combines the two images captured by the left eye and the right eye to gauge and measure the depth of the environment. Binocular stereo vision intends to duplicate this effect by using two cameras and combining the two images captured by them.

Fixation point is the point where the rays from the eyes/cameras meet. When the two eyes focus/gaze on an object at the same time, the object lies on the fixation point of the visual system. This fixation point can be seen in Figure 1 as point P.

Vergence is the process of adjusting the angle between the eyes (or cameras) so that both eyes are directed at the same world point. When the object lies on the fixation point, the binocular system has achieved vergence. When the pair of images at vergence are combined, they overlap and hence the object is in focus.

Achieving vergence has applications in gaze control, gaze fixation and gaze shifting [2]. In robot embodiment applications, when robots are seeing a person or an object their gaze has to be fixed on the particular object. This involves maintaining fixation on a moving object [2]. In this way, they can track the object and also get more information on the object and ignore the details in the background. And if the target object is changed, by achieving vergence robots can quickly shift their gaze on to the new object. Gaze shifts are also called saccades and it involves dynamically transferring the fixation from one target to another.

Fig. 1. Human Eye Visual System showing the fixation point P



In this article, we show through experimentation how vergence can be achieved and how we can measure if vergence has been achieved by introducing a metric called vergence metric.

II. Related Work

There has been quite a few works which describe how to achieve vergence. Here we list the prominent ones which are related to our work.

Oslon et al [2] propose an error feedback solution to develop a real-time vergence control system for Rochester Robot. They propose a design which mounts the cameras together on a platform that can be tilted up and down, and allows each camera to pan from side to side independently. With this design, the angle formed at the fixation point - gaze angle and the difference of the angles formed by the cameras will no longer be independent. Control needs to be done in two methods. One for normal operating mode which will be optimized for smooth, continuous changes expected during fixation. Another one for saccadic movements which will have bang-bang control. A proportional-derivative (PD) controller is used in the feedback loop to do this. The error function used is based on disparity. They use cepstral filter to measure the disparity. It is very good accuracy and in best case can achieve an RMS error of a small fraction of a pixel. They are able to implement this system and generate a smooth vergence camera movements in respond to changes in the

desired vergence angle. The system also performs reasonably well to step changes during saccadic movement.

Coombs et al discuss [3] the tracking problem of a moving object when the robot along with the cameras are also moving. They donot use object detection to distinguish between different objects in the scene. With the help of real time controlling of camera movements and binocular fixation segmentation they locate the target moving object. By implementing a gaze holding system on the robot's binocular system, even when the robot is moving the cameras are able to perfectly track the moving object. Due to gaze holding, the target object can be separated from the surroundings. The cameras converge on the imaginary horopter where the fixation point lies. And at the horopter the stereo disparity is zero. The binocular features at the horopter with zero disparity can be obtained from simple features and hence we can get the location of the object in the image. Here also the cepstral filter is used to calculate the disparity for error calculations. When the target object is fixated, its retinotopic location can be found by disparity filtering. A vergence and pursuit control system work together to track the moving obejct while the robot is moving. The vergence system controls the vergence angle to minimize the stereo disparity. The pursuit control system controls the pan and tilt angles of the cameras to center them on the object. It uses a PID controller to do this. It achieves localizing visual attention in 3D space and demonstrates interaction between motors and sensors to achieve binocular fixation on the target object.

Reid et al [4] propose a method to calculate the fixation point on a moving object for a real time gaze control system. The main logic involves using a corner detector and tracker. The image position of the desired fixation point is then reconstructed from the clusters detected on the object. Corner features is detected in the small window at the center of the field of view and is computed in real time. For every matched corner, the corner's position is predicted in the new frame using Kalman filter prediction equations. Then the window is searched for corners in the current frame. THe closest corner satisfying cross-correlation threshold is declared as the best match. The position of the match is used to update the Kalman filer. To achieve fixation from corner features, segmentation is used to identify which corners belong to the target object and which donot. It is assumed that the background is stationary and target is moving and there is only one moving object in the field of view. They have successfully shown that attention to detail of a visual processing active tracking system increases the overall performance of the fixation system. This method is fast, reliable, view-point invariant and does not require a prior model of target appearance. The implementation is successfull in real time even though it's using limited computational resources.

Ding et al [5] propose a method to localize and track a moving target in an indoor setting using binocular stereo vision. Otsu Delaunay HS (O-DHS) method is used to extract the moving target from its background. Stereo matching algorithm based on deep matching and stereo vision is developed to extract the dense stereo matching point pairs called stereo deep matching(S-DM). Next, using the extracted dense stereo matching points the 3D coordinates of the points in the moving target area is reconstructed. Then using the centroid of the image, localization is performed of the target object. Here the moving object is tracked using stereo deep matching.

Liu et al [6] employed a method using binocular vision and improved Yolov3 model to identify and localize pineapple fruits. Two MV-CA060-10GC GigE industrial cameras were used for the binocular camera setup. The binocular camera took images of the left and right components. The image from the left camera was sent into the Yolov3 model to detect pineapples. And then stereo matching was performed using the left and right images. The parallax of feature points was calculated by matching the region of interest and combining the matching feature points of the left and right images. Finally the 3D co-ordinates of the pineapple fruit was obtained from the average parallax of all feature points.

Based on these papers, we propose our solution to the vergence problem in the next section.

III. Method

In this section we explain in detail the calculation of fixation point, simulation experiments and hardware experiments that was conducted. We also explain the hardware design and the vergence metric formulation.

A. Calculating the fixation point

Fixation point is calculated using the triangulation principle. Consider the two cameras C1 and C2 separated by a distance d as shown in Figure 2. We know the distance between the cameras, and the angle the cameras make with respect to the ground plane. Fixation point O can be calculated using trigonometry. Consider camera C1 as the origin, then C2 coordinates become (d,0). Taking the tangent of the angles formed at C1 and C2 we get,

$$tan(C1) = OM/C1M$$

$$tan(C2) = OM/C2M$$

$$C1M + C2M = d$$

Here we have three unknowns and three equations. OM, C1M and C2M are unknown. Whereas the tangent of angles and d is known. Using this system of equations we can calculate OM and C1M. And hence we get the coordinates of fixation point O as (C1M,OM).

B. Simulation Experiments

To verify if vergence can be achieved by steering the cameras, we performed a priliminary experiment using Isaac Sim simulator. We used a simple room environment and rubix cube was chosen as the target object. Two cameras were simulated and placed at a distance from the target object. The pair of images obtained from the two cameras is shown in Figure 3. We kept the left camera angle contsant and moved the right camera till vergence was achieved. The combined images when vergence was achieved is shown in Figure 4.

Fig. 2. Fixation Point calculation

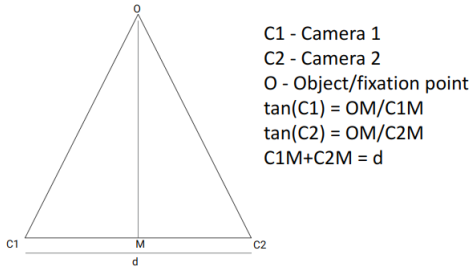
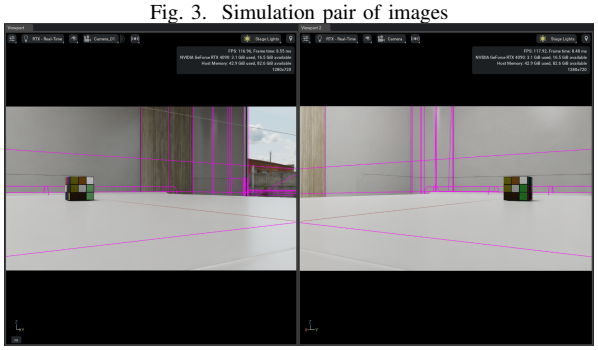


Fig. 3. Simulation pair of images



C. Hardware Design

We first started of with a cardboard design for the two camera system. We removed the shells of the web cameras and fitted them to the cardboard. The cardboard setup is shown in Figure 5. But this setup was not stable and precise enough to adjust the angles of the cameras.

We decided to 3D print the hardware since cardboard design was not sturdy. First, we followed the eye mechanism design of Inmoov 3D printed robot [7]. We removed the unnecessary parts like the mask for the eyes and nose and we printed it. This 3D printed design is shown in Figure 6. The idea was to control the two cameras using servo motors for pan and controlling the nose with yet another servo motor which connects both the eyes for tilt.

Even though this design was much better than the cardboard design, it was not perfect. The wires did not have adequate space and the nose design was not good for tilt. There was also not enough space for the servo motors.

Keeping all this points in mind, we came up with a new design without the nose structure. We provided adequate space for motors and wires. The pan motors were placed on top which controlled the two cameras rotation. The tilt motor was attached to a bar which hosted the two cameras. We 3D printed this second design and used it for our experiments. This design is shown in Figure 7.

We used Microsoft LifeCam HD-3000 web cameras and SG90, MG995 servo motors for the hardware setup. Hardware

Fig. 4. Simulation Vergence

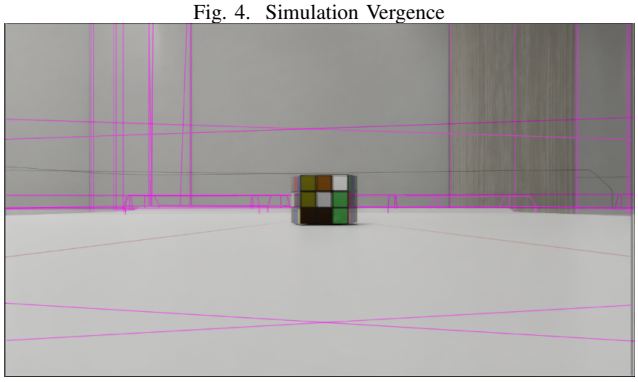


Fig. 5. Cardboard Design

Fig. 6. Hardware design 1.0

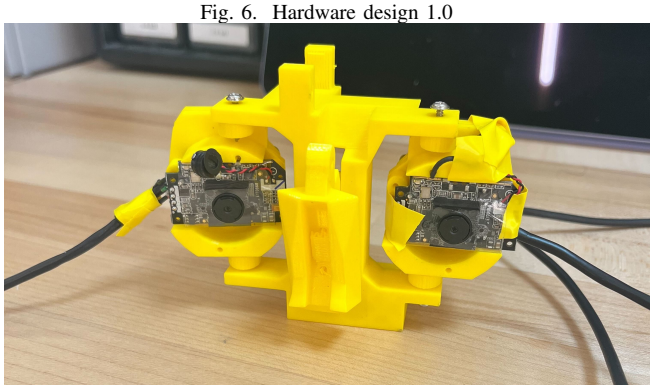


Fig. 7. Hardware design 2.0

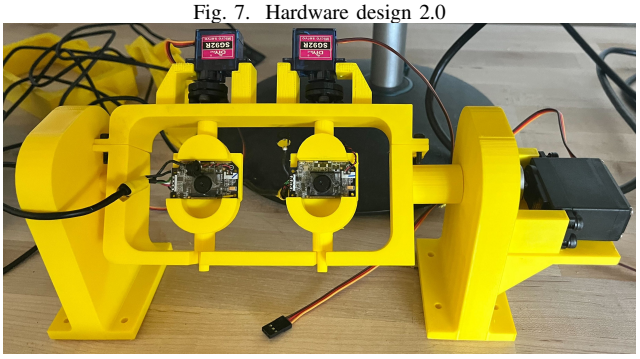


Fig. 8. Right Image

Fig. 9. Left Image

Fig. 10. Edge image combined

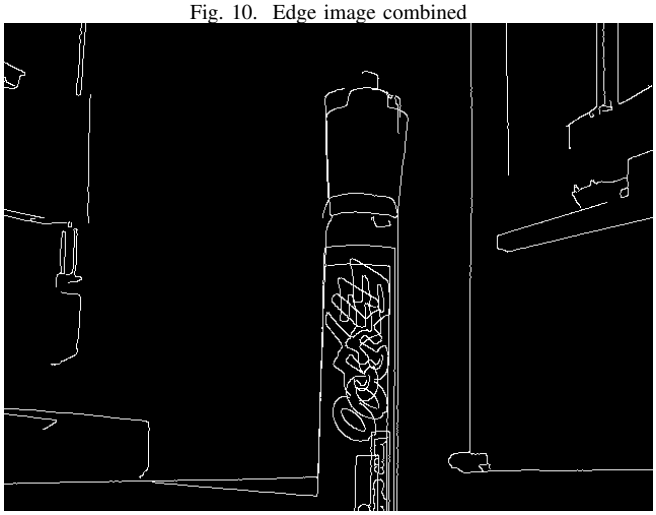
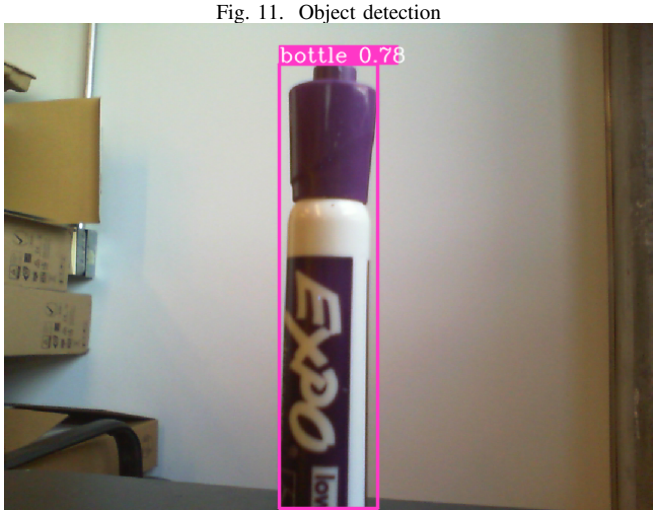


Fig. 11. Object detection



was connected to a Raspberry Pi 4 for motor control. The web cameras were connected to a Laptop using the USB cables.

D. Hardware Experiments

We choose marker as the target object. Considering the two angles as 80 degrees, we calculated the fixation point. Using the raspberry pi and python programming we steered the motors to 80 degrees. Next we placed the marker at the fixation point and got the pair of images. The image pairs can be seen in Figure 8 and 9. We converted these images to grayscale and added a gaussian blur. explain canny

We then performed Canny edge detection on the pair of images to get the features. The edge images were combined to see if the marker overlapped. The combined edge image is shown in Figure 10. We could visually see the marker from both the images overlap in the edge image.

We also performed YOLOV6 object detection [8] on the color images to get the bounding box of the target object. The object detection results is shown in Figure 11.

E. Vergence Metric Formulation

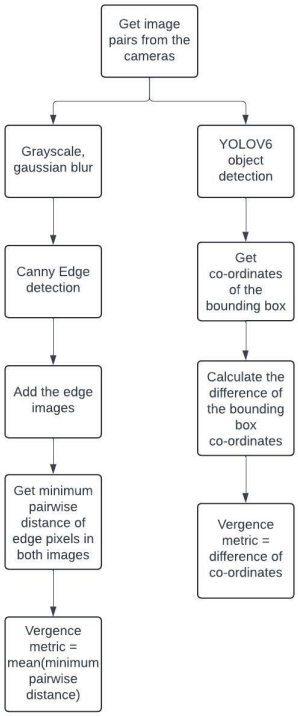
Visually confirming that vergence has been achieved is insufficient. We need to formulate it to prove that vergence has been achieved. We came up with a metric called vergence metric for this purpose. We have two approaches to calculate the vergence metric.

1) Approach - I: We consider the center pixels of the left and right edge image and calculate the pairwise difference of every edge pixel on the left image to every edge pixel on the right image. Then we calculate the minimum pairwise distance by taking the minimum of every row in the distance matrix. Next we calculate the mean of this metric to with respect to all points to get the final Vergence metric. We also calculate this from the right image to left image. The formula can be written as follows:
Vergence metric LR = mean(min(pairwise distance of every edge pixel from left image to right image))
Vergence metric RL = mean(min(pairwise distance of every edge pixel from right image to left image))

Fig. 12. Vergence metric results

```
Original image shape =  (480, 640, 3)
Right sliced image shape =  (50, 50)
Left sliced image shape =  (50, 50)
Vergence metric left to right =  5.0304574175733965
Vergence metric right to left =  4.757454980326118
Right sliced image shape =  (100, 100)
Left sliced image shape =  (100, 100)
Vergence metric left to right =  4.508043345461392
Vergence metric right to left =  4.659822196292295
Right sliced image shape =  (150, 150)
Left sliced image shape =  (150, 150)
Vergence metric left to right =  5.157809044600441
Vergence metric right to left =  5.750958751757263
Right sliced image shape =  (200, 200)
Left sliced image shape =  (200, 200)
Vergence metric left to right =  5.35491098137355
Vergence metric right to left =  5.440376030153205
Right sliced image shape =  (250, 250)
Left sliced image shape =  (250, 250)
Vergence metric left to right =  5.724345727033527
Vergence metric right to left =  5.1104108085968045
Right sliced image shape =  (300, 300)
Left sliced image shape =  (300, 300)
Vergence metric left to right =  5.095955725541411
Vergence metric right to left =  15.293900275326237
Vergence left to right for image sizes  [25, 50, 75, 100, 125, 150] is =  [5.0304574175733965, 4.508043345461392, 5.157809044600441, 5.35491098137355, 5.724345727033527, 5.095955725541411]
Vergence right to left for image sizes  [25, 50, 75, 100, 125, 150] is =  [4.757454980326118, 4.659822196292295, 5.750958751757263, 5.440376030153205, 5.1104108085968045, 15.293900275326237]
Images saved
```

Fig. 13. Flowchart



2) Approach - II: In the second approach we calculate the vergence metric using the coordinates of the bounding box obtained from YOLOv6 [8]. We take the coordinates of the top left corner and bottom right corner from one the left image and subtract it from the right image respective coordinates. We calculate both right to left and left to right metric in this approach too.

F. Algorithm

The algorithm used to calculate the vergence metric using both the approaches is shown in the form of a flowchart in Figure 13. Python programming language was used for coding and opencv library was used to capture and process images.

IV. Results

We considered different center image sizes and calculated the vergence metric using both the approaches. We iterated this for center images sizes from 50*50 to 300*300. We calculated both left to right and right to left vergence metric for each case. Except for the last 300*300 image size, all the other image sizes had a constant vergence metric. The average vergence metric was about 5 pixels. The vergence metric calculation results for the marker target object is shown in Figure 12.

V. Discussion and future work

In this project we successfully calculated the fixation point and performed experiments to achieve vergence. We also formulated vergence metric using two approaches. The future work would be to design a controller which will automatically calculate the fixation point and adjust the camera angles accordingly for any target object placed anywhere in the real world. The vergence metric can be used for this. As and when the motors adjust the angles of the cameras, the vergence metric changes and when the vergence metric approaches zero we can say that true vergence has been achieved.

VI. Acknowledgement

I would like to thank my advisor Prof. Shreyas Kousik for all his guidance throughout the two semesters without which this project would not be possible. I also want to thank my labmate Liliana Whitesell for all her work throughout this project in designing the hardware and 3D printing it. I want to thank all my lab members for all the support. I also like to thank my friends and family for their constant support. Lastly I would like to thank the Robotics department for giving me an opportunity to do this capstone project

References

[1] T. Wang, B. Liu, Y. Wang, and Y. Chen, "Research situation and development trend of the binocular stereo vision system," *AIP Conference Proceedings*, vol. 1839, 05 2017. 020220.

[2] T. J. Olson and D. J. Coombs, "Real-time vergence control for binocular robots," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 67–89, 1991.

[3] D. Coombs and C. Brown, "Real-time binocular smooth pursuit," *International Journal of Computer Vision*, vol. 11, pp. 147–164, 1993.
[4] I. D. Reid and D. W. Murray, "Active tracking of foveated feature clusters using affine structure," *International Journal of Computer Vision*, vol. 18, no. 1, pp. 41–60, 1996.
[5] J. Ding, Z. Yan, and X. We, "High-accuracy recognition and localization of moving targets in an indoor environment using binocular stereo vision," *ISPRS International Journal of Geo-Information*, vol. 10, no. 4, p. 234, 2021.
[6] T.-H. Liu, X.-N. Nie, J.-M. Wu, D. Zhang, W. Liu, Y.-F. Cheng, Y. Zheng, J. Qiu, and L. Qi, "Pineapple (ananas comosus) fruit detection and localization in natural environment based on binocular stereo vision and improved yolov3 model," *Precision agriculture*, vol. 24, no. 1, pp. 139–160, 2023.
[7] "Inmoov eye mechanism." https://inmoov.fr/eye-mechanism/.
[8] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, *et al.*, "Yolov6: A single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.