3D Gaze Tracking Guide

Purpose

The purpose of this software is to determine a user's 3D gaze in virtual space by utilizing a geometric model. It offers a variety of calibration setups and fixation determining algorithms which can be tweaked and chosen to fit a particular user and his/her intended application. For instance, shorter duration calibrations may be chosen if speed is important or setting all calibration points to appear inside the screen (behind the convergence plane) if their application displays all objects behind as well.

The software involves multiple stages which can be used (or skipped) depending on the user's application. It begins with a calibration with third party eye tracking software provided by Gazepoint followed by a 2D and 3D calibration. After each calibration the user is given a chance to review the captured data and resulting fixation estimations. The final stage is a sandbox which includes 3 environments meant to help the user evaluate the precision of various calibration methods. Following calibration, the user can output the 3D gaze point to be used in their application.

Materials

PC

Processor: Intel® Core™ i5-4460 @ 3.20 GHz

Installed memory (RAM): 4.00 GB

System type: Windows 10 64-bit Operating System, x64-based processor

GPU: GeForce GTX 750 Ti

Monitor

ASUS VG248 24" LCD 144 Hz 3D

Eye Tracker

Gazepoint GP3 60 Hz

3D Glasses

NVIDIA 3D Vision 2

Chin Rest

Physical Setup

Figure 1 shows the 3D gaze tracking setup. The users head is held steady using a chin rest. The height of the rest should be adjusted to align the x and y coordinates of the point between the users eyes to be equal to the monitors center. The distance between the monitor and between the user's pupils should be 92cm. The Gazepoint GP3 eye tracker should be setup directly in front the monitor approximately 70 cm from the user's eyes. Since the user will be wearing NVidia Vision 2 Glasses special attention should be paid to ensure that no part of the screen or the eye trackers view of both eyes is obstructed by the

glasses. This may require adjusting the position of the eye tracker as well as lighting conditions in the room.

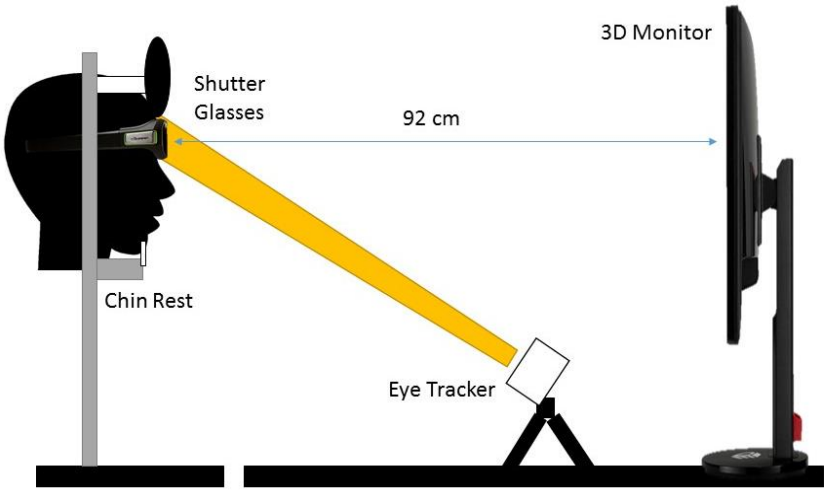

Figure 1: Physical setup

3D Gaze Calculation and Camera Setup

While generating believable stereo is important for this type of application it is even more important that that the frustum we create for both the left and right eye match the frustum we are creating in our physical setup. This is important since we calculate our virtual 3D gaze position by finding the intersection of rays defined by the location of each pupil to their corresponding fixation point on the monitor. If the field of view in the virtual frustums and the FOV created for each eye by the edges of the monitor are the same then we can simply use the 3D gaze position calculated in physical space as our virtual 3D gaze position without any conversions.

You may notice the similarity between the physical viewing frustums in figure 2 and the virtual ones in figure 3. The convergence point in figure 3 is treated as if it is the monitor. More information regarding the actual implementation of the stereo frustums can be found here: http://www.animesh.me/2011/05/rendering-3d-anaglyph-in-opengl.html

Important: To increase accuracy it is recommended that you edit the eye separation value found in Camera.h to the exact interocular distance (m) of the user. This will adjust the location of the frustums as well as the calibration points and gaze depth calculations. If distance between the eyes and monitor is changed the value of the variable Convergence should also be modified to reflect this.

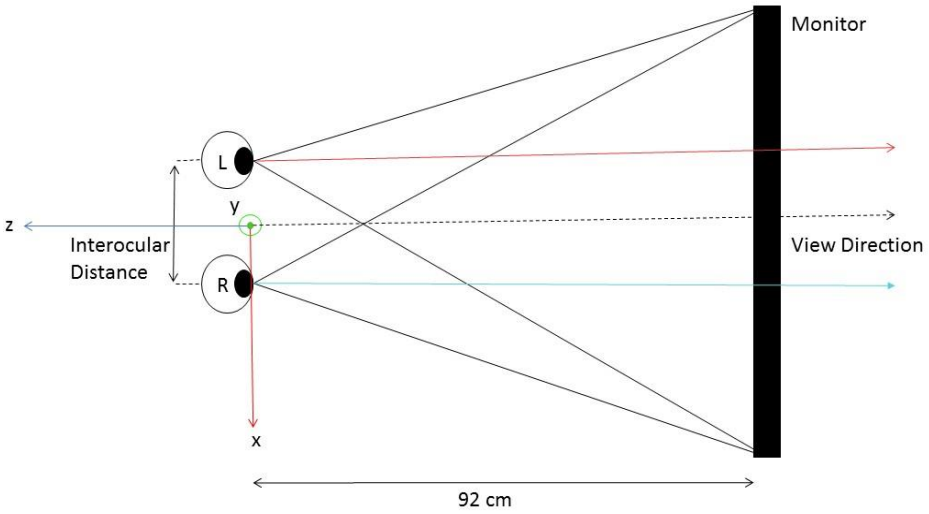

Figure 2 Physical viewing frustums

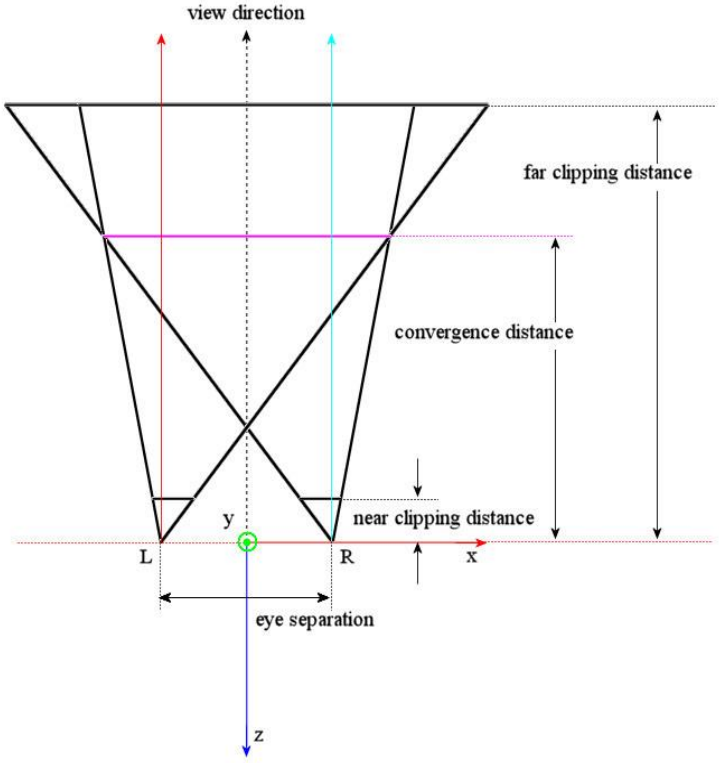

Figure 3 Virtual view frustums

Calibration Guide

The following is a simple guide to the software. Please note the key map for each stage after its description.

Eye Track Calibration

First step is to run the calibration program provided by your eye tracker. It is recommended that you try and keep the setup consistent between this third party calibration and while running the program. This means using the chin rest and wearing the 3D glasses (although the shutter feature will not be active).

If you are using Gazepoint GP3 you can run the calibration as follows

1. Open Gazepoint Control application
2. Adjust eye tracker to make sure both eyes are focused on (in green squares)
3. Click calibrate (top left corner)
4. Follow 9 point calibration
5. Leave software running

2D Calibration

Once the third party calibration is complete you can launch the software which begins with another 2D calibration. The 2D Calibration involves 9 calibration points. Simply follow the target with your eyes as it moves to, and holds at, each point. The positions, transition time and hold time of all calibration points can be adjusted in “Plane_Calibration.h”. the purpose of this calibration is to generate the first calibration method, a linear fit between each eyes fixation point and the actual location of the target.

Key Map

q->skip to 3d calibration

2D Calibration result

If the user completes the 2D calibration a summary of the captured data is shown on the screen. Each calibration point is drawn in red with the eye track points drawn for each eye. Blue represents the data for the right eye while green the left. The more recent the eye track data at each calibration point the brighter its color.

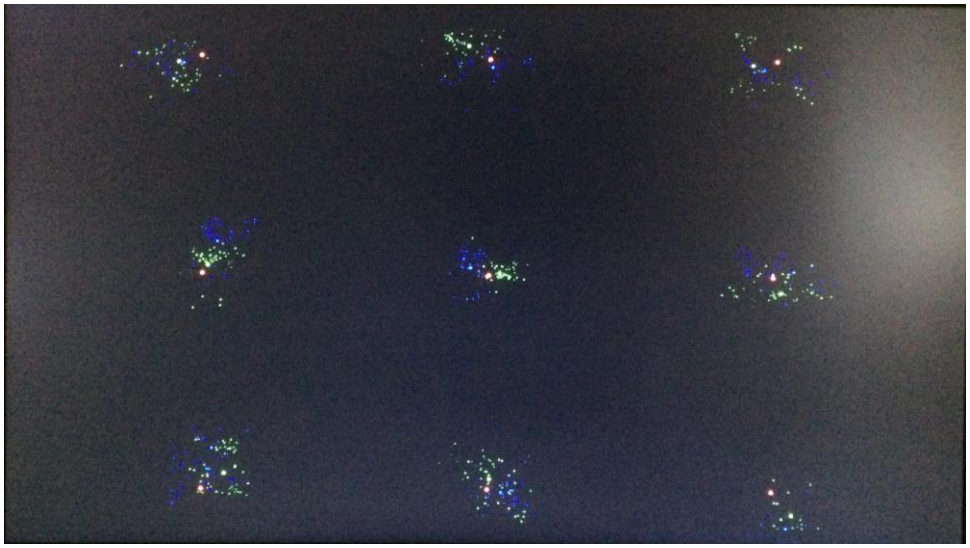

Figure 4 2D Calibration result screen

Key Map

q->next

r->toggle right eye raw

Shows data collected at each calibration point for right eye

l->toggle left eye raw

Shows data collected at each calibration point for left eye

w->toggle left average

Draws large green circle where the average observed left eye gaze point was for each calibration point

e->toggle right average

Draws large blue circle where the average observed left eye gaze point was for each calibration point

t->toggle linear fit

Linear fit adjusts the x and y values of each eye separately to try and move the average gaze point closer to the calibration point

3D Calibration

The 3D calibration begins immediately after the 2D calibration whether it was skipped or not. The 3D calibration functions similarly to the 2D calibration except it moves through 9 points at 4 different depths each (36 total points). Once again the location for the points, duration and transition times as well as plane depths can all be adjusted. This stage must be completed in order to make use of the second and third calibration method. This includes a linear regression between left and right eye fixation separation, average x coordinate, average y coordinate to the 3D gaze position depth. The second method maps the basic 3D gaze position (using average values) to the actual 3d position of each target using 3 separate linear regressions. The calibration can also be run consecutive times which will allow you to observe the variation in data sets. Adjustment can be made in “Depth_Calibration.h”

Key Map

q->skip to Sandbox

3D Calibration result

If the user completes the 3D calibration a summary of the captured data is shown on the screen. Calibration points for the left eye is drawn in white while the right is drawn in red with the eye track points drawn for each eye. Blue represents the data for the right eye while green the left. Once again the more recent gaze points are brighter and the fixation point for each calibration is shown as a larger green or blue dot.

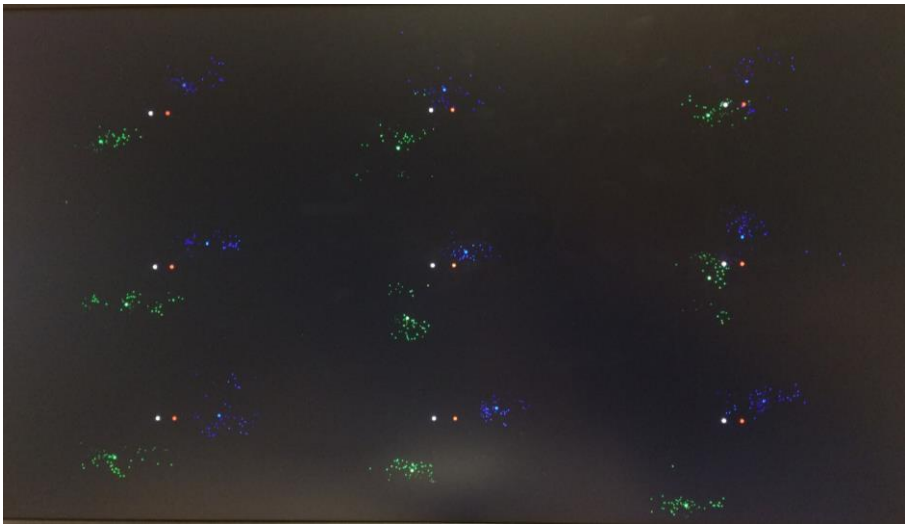

Figure 5 3D Calibration result screen

Key Map

q->next
r->toggle right eye raw
l->toggle left eye raw
w->toggle left fixation
e->toggle right fixation
s->plane 1
    Show the 9 points and corresponding data for the first plane
d->plane 2
f->plane 3
g->plane 4
h->set 1
    Displays the first set of data (used if multiple calibrations are run consecutively)
j->set 2
k->set 3
l->set 4
y->display average
    Fixation is calculated using an average

u->display clean average

Fixation is calculated using an average with outlier's removes

i->display weighted average

Fixation is calculate using average with more weight on more recent observed gaze points

o->display median

p->display dti

Fixation is calculated using a Dispersion-threshold algorithm which determined fixations based on grouping of consecutive gaze point over a minimum duration which do not vary passed a certain distance threshold

t->toggle linear fit

Adjusts all displayed data based on linear fit found during 2D calibration

Sandbox

The sandbox includes three different scenarios to test the accuracy of the 3D gaze position. The first involves targets drawn randomly within a predefined range. The user trues to eliminate a target by moving his/her gaze point (drawn in yellow) with a certain distance of the target. If this happens the target is redrawn elsewhere. The second scenario involves a full screen image which can be moved to various depths. A meter to the right shown the depth of the gaze point on a scale of 0 to 100 cm. 0 being directly at the convergence point with 100 being 100 cm behind the screen. The third is a grid of 9 targets each at varying depths. The differs from the image viewer in that the 3D "effect" is only visible at the targets instead of the entire screen. Adjustment can be made in "Sandbox.h"

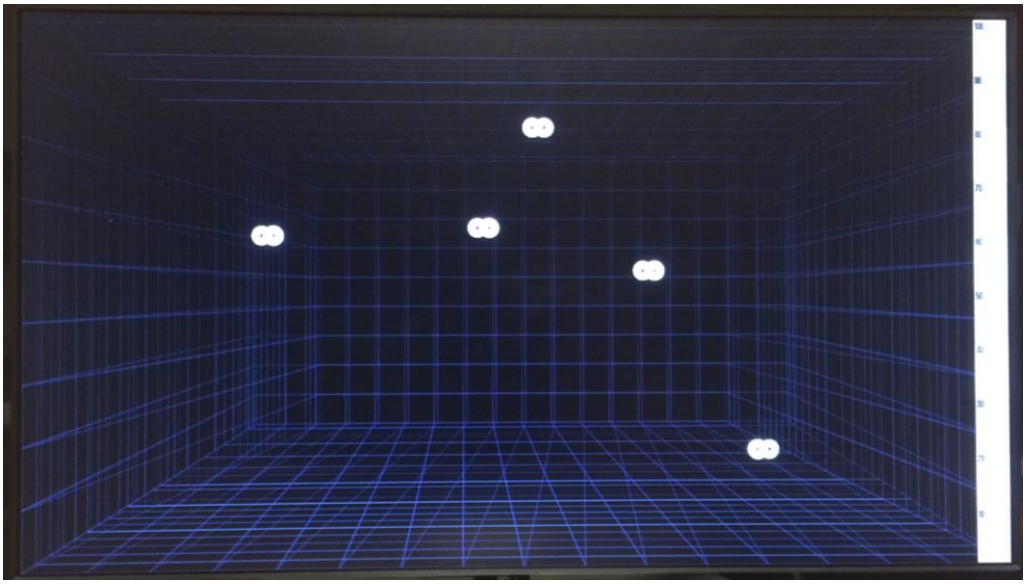

Figure 6 Target practice mini game

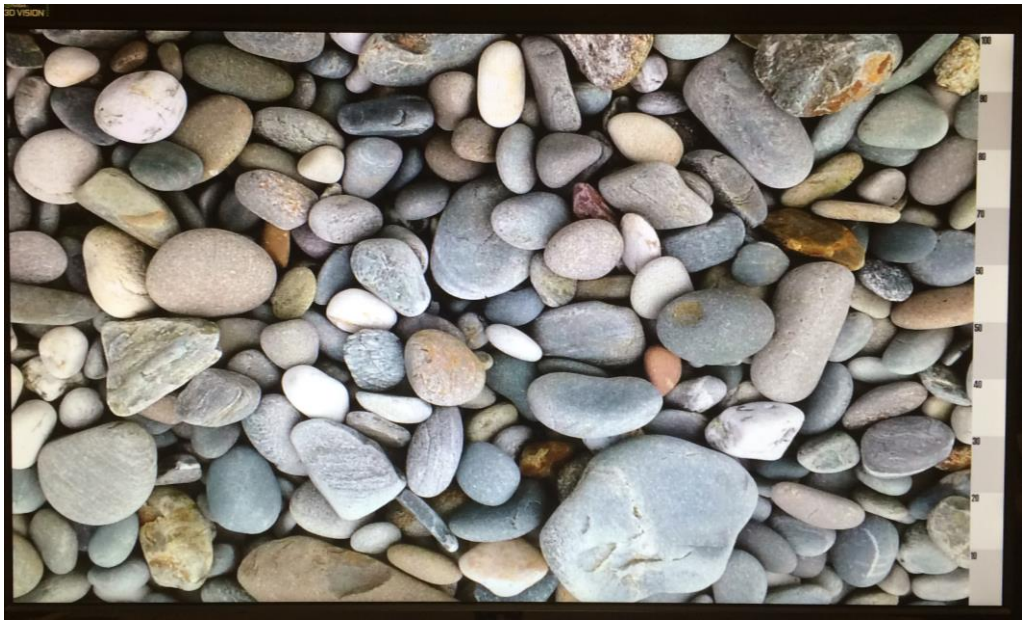

Figure 7 Depth calculation precision test

Figure 8 Static targets test

Key Map

i->select target practice mini game
o->select image viewer
p->select static target environment
s->increase image depth

d->decrease image depth

e->no fit

Calculate 3D gaze point by using the average of the last 1.5s of collected data

r->linear fit

Calculate 3D gaze position using averages of the last 1.5s of collected data adjusted by the linear fit calculated during the 2D calibration

t->l regression

Calculate 3D gaze position using averages of the last 1.5s of collected data adjusted by using a linear regression calculated during the 3D calibration

y->linear regression v2

Calculate 3D gaze position using averages of the last 1.5s of collected data adjusted by using a linear regression calculated during the 3D calibration

f->0

Move the image to different depth behind the screen (0 – 100 cm)

g->10

h->20

j->30

k->40

l->50

z->60

x->70

c->80

v->90

b->100