Project Seahaven -Script Overview

Viviane Kakerbeck

June 6, 2018

Abstract

1 Script Overview

Table 1: My caption

Name	Author	Purpose	Dependencies
ValidationAnalysis	Viviane Kakerbeck	Get some overall statistics over the validation values of the subjects.	Validation_VP#_Val#.txt files
PositionAnalysis	Viviane Kakerbeck	Extract data from position.txt files and save it in matlab matrices.	position_VP#.txt files
AnalysisMap	Viviane Kakerbeck	Analyze position data created with PositionAnalysis script. Create overlaid map, visualize north, heatmaps and repeated measurement comparisons.	North_VP_#.mat and Path_VP_#.mat
Heatmap3D	Viviane Kakerbeck	Create three dimensional heatmap from eye tracking data.	3DHeatmap(RandomX)_VP#.txt
AnalyzeAllViews	Viviane Kakerbeck	Extract data from ViewedHouses.txt and save in a matlab matrix. Possibility to visualize a time line.	ViewedHouses_VP# (created in seperate Unity scene called DrawViewingPath.)
Analysis_ViewedHouses	Viviane Kakerbeck	Create overall statistics about houses looked at. Also saves matrix with total house viewing times, distance and variance for each subject.	NumViewsD.mat
GazeStandVSWalk	Viviane Kakerbeck	Compare gaze points while standing to gazes while walking.	EyesOnScreen_VP#.txt and Positions_VP#.txt
LeftRightTurns	Viviane Kakerbeck	Look at gazes that were made while turning left and while turning right.	EyesOnScreen_VP#.txt and Positions_VP#.txt
Entropy	Viviane Kakerbeck	Look at how much people switched between looking at different houses and correlate this with the task performance.	ViewedHouses_VP#
Performance Analysis	Viviane Kakerbeck	Get overall performance for all tasks and performances for each house. Then correlate this with different variables like how long houses were seen, from which distance they were seen and the variance in the distance.	AlignmentVR_SubjNo_#.mat and NumViewsD_VP_suj_num.mat

2 Script Explanations

2.1 ValidationAnalysis

This script reads in all validation files from all subjects and calculates some overall statistics. These statistics are printed to the command line and can look like this:

```
Average Error (degree) for the last callibration before session starts:     1.4211 Median: 1.3796
Average X-Error (pixel) for the last callibration before session starts:    1.0971 Median: 0.85
Average Y-Error (pixel) for the last callibration before session starts:    3.3431 Median: 3.2333
Average Error (degree) directly after a callibration:                       2.9694 Median: 1.8657
Average Error (degree) directly after a callibration during session:        2.6182 Median: 1.5726
Average Error (degree) in one point callibration:                           3.5324 Median: 2.2894
Average X-Error (pixel) in one point callibration:                          2.2991 Median: 1.1003
Average Y-Error (pixel) in one point callibration:                          8.5778 Median: 4.6818
Average accepted error (degree) during Game:                                1.5403 Median: 1.2231
Average error (degree) at end of game:                                      3.0676 Median: 2.2007
```

Figure 1: Example output of the validation analysis script. The first column of numbers represents the mean, the second the median.

Errors in x and y direction are measured in pixels and are just meant the be as a reference to compare errors in the two directions. All other measures are in visual degrees.

Additionally you get a struct called *validations* containing structs for each subject you analyzed. These structs give you information about each single validation you performed on this subject. With this you can now perform your own additional analysis if you are interested in more than the overall statistics given to you. In figure 2 you can see an example for such a struct.

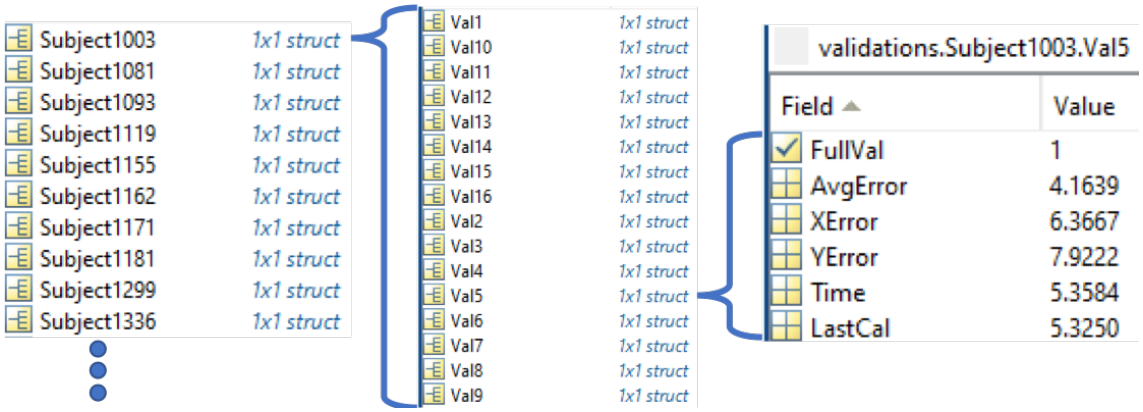

Figure 2: Structure of the data returned from the ValidationAnalysis script which can then be used for further analysis. In this example you see statistics for a full validation performed five minutes after training started and right after a calibration was performed.

Each subject struct contains several sub structs, one for each validation that has been performed on this subject. These structs contain several values:

Table 2: Validations - Variable Names

Variable	Meaning
FullVal	Is zero if this validation was a one point validation. Otherwise it is one (9 point validation)
AvgError	The average error of this validation in visual degrees
XError	The average error in X direction of this validation in pixels
YError	The average error in Y direction of this validation in pixels
Time	At what time in the experiment was the validation performed (in minutes since start of training)
LastCal	At what time in the experiment was the last calibration performed (in minutes since start of training)

2.2 PositionAnalysis

This script goes through all *positions_ VP#.txt* files and reads out their content. This content is then saved in three different Matlab matrices within the position folder. The first matrix is called *Map_ VP_ #.mat* and contains pixel values for a map with the subjects path drawn on it. Currently it also contains information about the amount of rotation made at each point which can later be visualized (figure 5). This can also be removed if not desired. The next matrix is called *Path_ VP_ #.mat* and contains all x and y positions a subject has visited. The last matrix which gets saved is called *North_ VP_ #.mat* and contains the rotation of the player at the end of a session which is when the player is asked to face towards the direction he or she perceives as north. Additionally the x and y coordinates of start and end point of a line which would draw this rotation are saved in this matrix.

These matrices are being further processed in the *AnalysisMap* script.

2.3 AnalysisMap

AnalysisMap contains several overall analysis of the data extracted from the *positions_ VP#.txt* files with the *PositionAnalysis* script. These analysis are separated into different sections which can be ran individually by pressing *Run Section* while having one of these sections selected.

The first section draws all subjects walking paths onto Seahaven's map in different colors. Figure 3 shows how this could look like for 177 subjects.

Figure 3: Walking paths of 177 subjects overlaid over the map of Seahaven.

The next two sections draw heat maps over all subjects. The first one visualizes the time that was overall spent in each area of the city. The values used for this map are the overall amount of frames spent in each part of the city of all subjects combined. These values are then normalized and visualized in a heat map (figure 4 - left). The second one looks very similar and shows the amount of subjects that have been in each area of the city.(figure 4 - right)

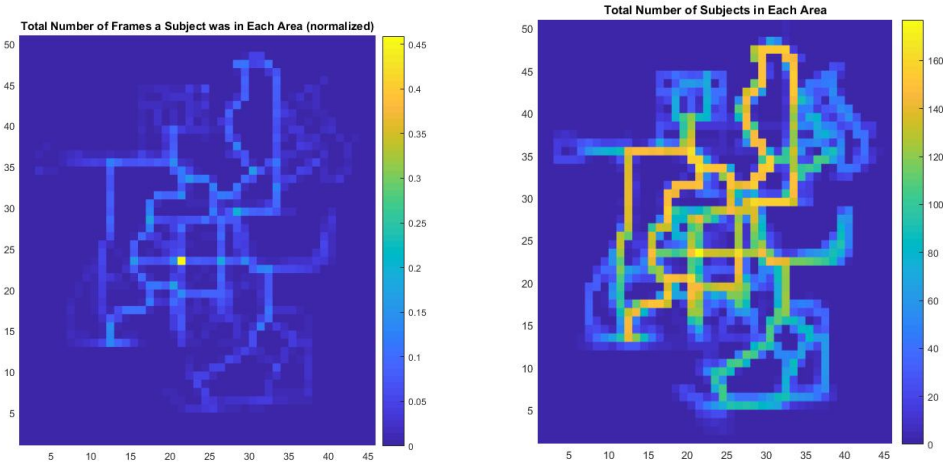

Figure 4: (Left) Total time spent in each area of the city (normalized). (Right) Total amount of subjects that visited each area of the city.

Additionally you can also visualize walking paths of single subjects. To do this just comment in the following section and change the range of the loop to the subject you want to take a closer look at. This path is colored by the amount of rotation the subject made at the respective position. This means that light blue regions correspond to turns of the head or body while dark blue regions correspond to walking and looking straight.

Figure 5: Walking path of one subject colored by rotation. Lighter blue corresponds to more change in rotation.

The next section can be used to draw the participants subjective perception of north. It plots all rotations during the last frame of a sessions as shown in figure 6 . True north would be a straight line from the middle to the top of the image.

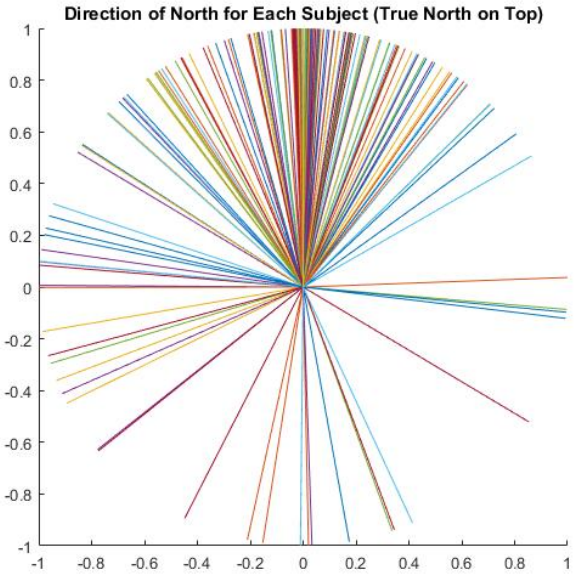

Figure 6: Individual perception of north. True north is on the top.

At last we can also compare walking paths between repeated measurements. To do this use the last section of the script [1]. To let the script know which subjects are repeated measurements and which VP numbers correspond to the same subject a table is being read in in which this information is specified. The path to this list needs to be specified at the beginning of the section. The table should adhere to the following structure: Each row is one Person, each column is one measurement where the first column corresponds to the first measurement, the second column to the second measurement and so on.

The script then draws walking paths of all subjects. Each subject is shown in a separate figure. Each figure contains three subplots, each showing the walking path for one measurement. The color corresponds to the time in the session when the respective spot was visited (blue = early, yellow = late). The red dot marks where the subject started.

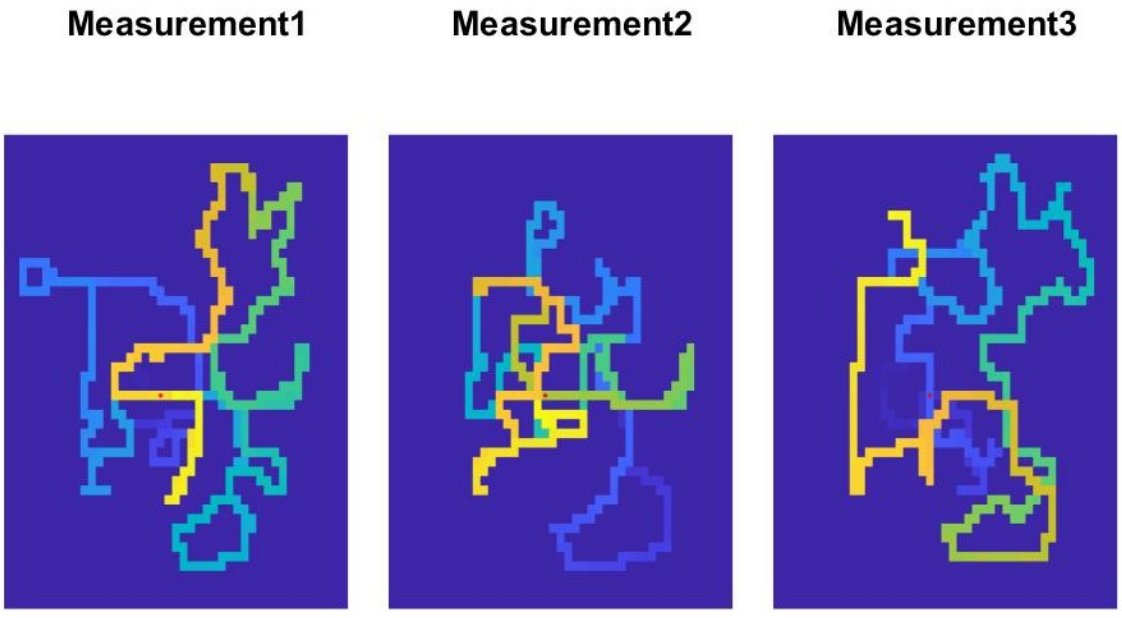

Figure 7: Walking path of one subject during repeated measurements. Each plot corresponds to one 30 minute session. Colors correspond to the time during the session (blue = early, yellow = late). The red dot marks the starting point.

Additionally the map coverage for each session is calculated. This is calculated by dividing the overall area of map coverage by all subjects by the map coverage of one subject and multiplying

[1] For all sections it is a prerequisite to run the very first section of the script

it with 100. By doing this we only get the percentage of area covered that is actually reachable by subjects. This is under the assumption that when letting 200 subjects walk through a VR environment almost all areas that are walkable will be visited at least by one subject. Figure 8 shows how the percentage of map coverage changes over sessions.

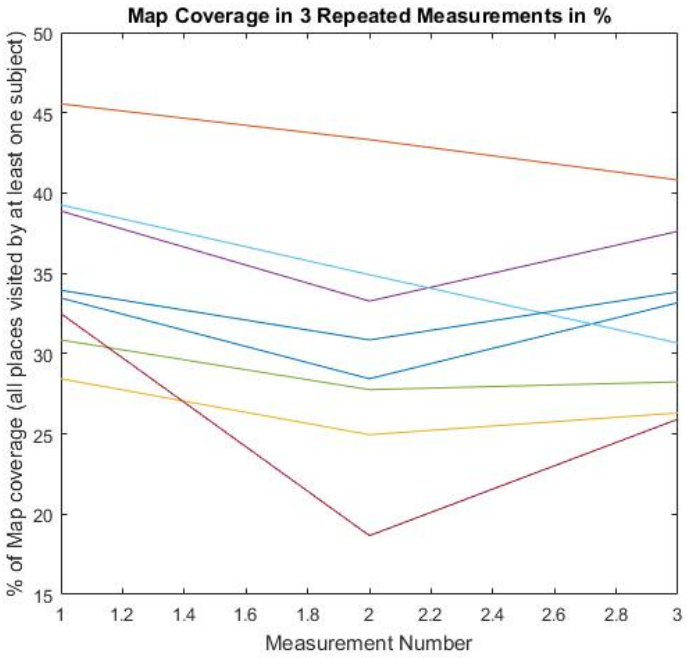

Figure 8: Percentage of map coverage over repeated sessions. Each line represents one subject.

2.4 Heatmap3D

This script created 3D heat maps which resemble usual eye tracking heat maps in the x and y direction but include a third dimension z which represents the distance of objects that were looked at. To run this script you need to create *3DHeatmap(RandomX)VP#.txt* files with the help of the *DrawViewingPath* Unity project. Here you have the option to either create a normal heat map file or pick one of two randomizations.

One of the randomizations is called *random position* which means that the list of the subjects positions is being shuffled and then matched with the subjects eye positions. This means that we use the original gaze vectors of the subject but shoot them from different positions into the virtual world to detect what was looked at. Therefor the x and y dimension of the 3D heat map will look exactly the same as in the normal heat map. The only thing that changes is the distance dimension since the gazes will now hit different objects at different distances.

The other randomization is called *random gaze* Here we use the original positions of the subject during the session and pair them with random gaze vectors. Therefor the x and y dimension of the heat map will just be randomly distributed points while the distance dimension reflects where random gaze vectors would hit objects in the virtual world.

The distribution of data points along the three dimensions (x,y and distance) is visualized as a 3D scatter plot where the color of each point represents how many close neighbors it has (the density at this point).

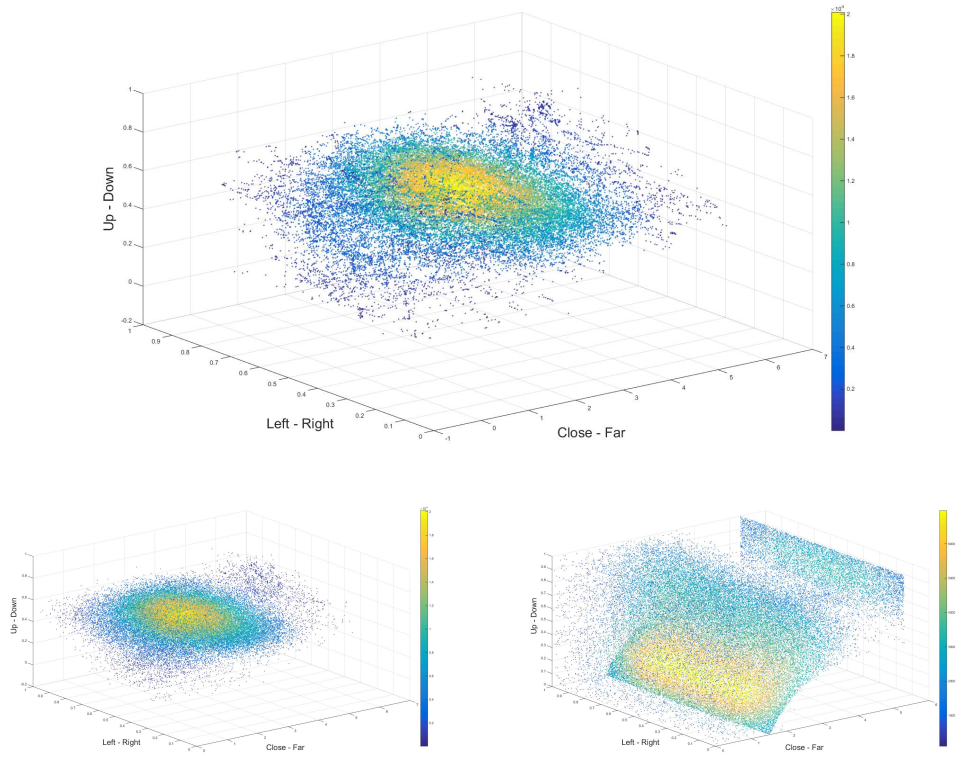

Figure 9: (Top) Heat map for original data of one subject. (Bottom Left) Heat map for original gaze data paired with shuffled position data for one subject. (Bottom Right) Heat map for original position data paired with random gaze vectors for one subject. Yellow represents areas with a high density of data points, blue has a low density.

To asses the differenced between the point clouds you can plot them in a 2D heat map and substract different conditions from each other. This is shown in figure 10. Additionally Matlab returns the distance of the points with the highest density of each condition.

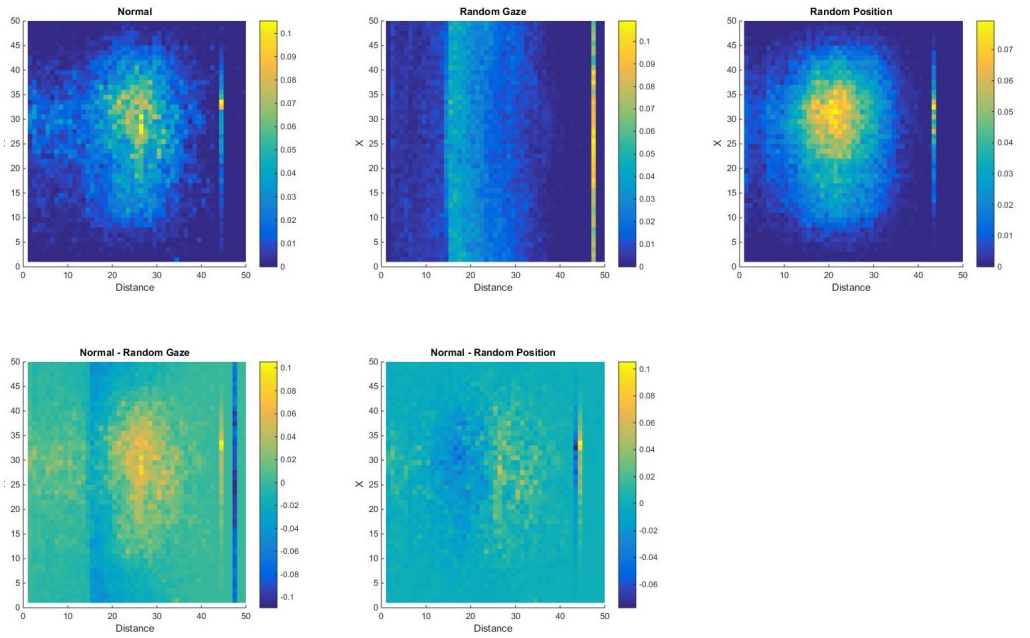

Figure 10: (Top) Heat map for original data of one subject. (Bottom Left) Heat map for original gaze data paired with shuffled position data for one subject. (Bottom Right) Heat map for original position data paired with random gaze vectors for one subject. Yellow represents areas with a high density of data points, blue has a low density.

2.5 AnalyzeAllViews

This script is just as *PositionAnalysis* a preprocessing script which extracts all information out of the recorded *ViewedHouses_ VP#.txt* files and saves it in Matlab matrices for faster further processing. The prerequisit for running this script is that you have created *ViewedHouses_ VP#.txt* files with the *DrawViewingPath* Unity project. The results are bein saved as *NumViewsD_ VP_ #.mat* files in the same folder as the original recordings. This table contains a row for each house the respective subject has seen. In the next three columns you can read the time this house was looked at (in seconds), the average distance from which this house was seen as well as the variance in distance respectively.

1 House	2 occ	3 DistanceMean	4 DistanceVariance
'001_0'	0.1333	29.3580	144.2646
'002_0'	4.2667	27.7014	209.6784
'003_0'	1.3333	28.9104	42.8721
'004_0'	9.8667	20.9670	853.2050
'005_0'	0.5000	50.5007	71.0219
'006_0'	2.9333	28.3217	534.3115
'007_0'	0.5333	36.2192	7.8287
'008_0'	7.3333	41.6814	592.8906
'009 0'	10	23.7917	429.7152

Figure 11: Example excerpt of the NumViews table for one subject. The *occ* column contains the overall time each house was looked at in seconds. The *DistanceMean* column contains the average distance from which each house was looked at by this subject and *DistanceVariance* shows the variance in the distances for each house.

Additionally you can create a time line for each subject which visualized when which house was looked at. This part is currently commented out since it is just for exploratory analysis and would be hard to look at for almost 200 subjects. However, if you want to just look at one subject specifically just comment this part in again and change the range of the overall loop.

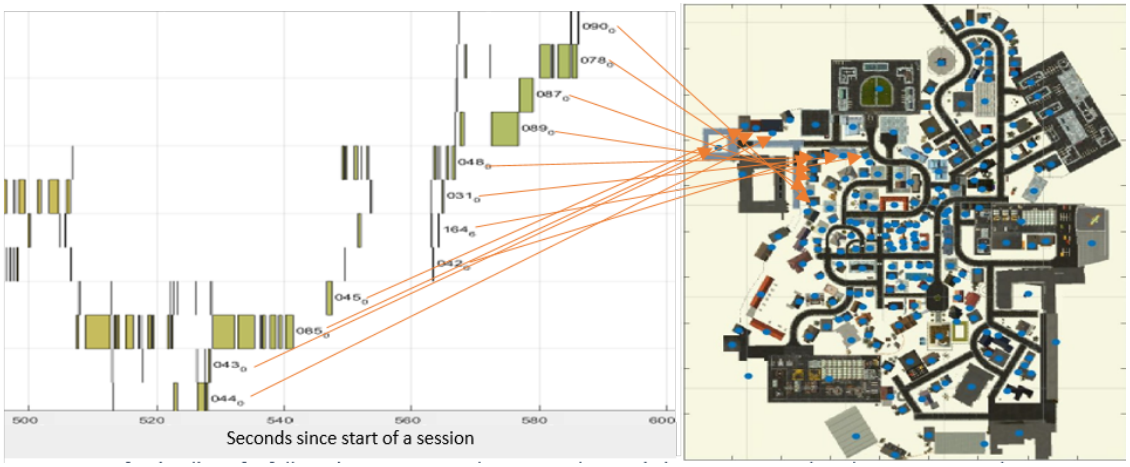

Figure 12: Excerpt out of a time line for one subject. (Left) This is what you get as an output from Matlab. On the x-Axis you have time in seconds. On the y-Axis you have the different houses. A box represents a certain time window during which a specific house was looked at. (Right) Corresponding houses in the city (just for visualization, this was created by hand).

2.6 Analysis_ViewedHouses

This scripts does overall analysis of the *ViewedHouses_ VP#.txt* files created with the *AnalyzeAllViews* script. It calculates some overall statistics and saves two matrices in a *Results* folder inside of the *ViewedHouses* folder.

One matrix is called *totalNum#.mat* and contains a list of all houses and corresponding overall statistics. There are six columns in the table:

Table 3: totalNum - Variable Names

Variable	Meaning
House	The name/number of the house.
occ	The overall time this house was looked at (in seconds).
DistanceMean	The mean distance from which this house was looked at.
DistanceVariance	The average variance in distances from which this house was seen.
numVP	A count of how many subjects have seen this house
avgocc	The average time this house was looked at (in seconds).

1 House	2 occ	3 DistanceMean	4 DistanceVariance	5 numVP	6 avgocc
'NH'	1.6791e+05	46.6520	3.8032e+03	194	865.4997
'033_0'	3.9751e+03	35.8667	442.2577	177	22.4584
'004_0'	3.6981e+03	20.9670	853.2050	181	20.4317
'022_0'	3.3751e+03	43.4619	671.3705	170	19.8537
'015_0'	2.8036e+03	32.1002	638.2327	188	14.9128
'009_0'	2.4240e+03	23.7917	429.7152	171	14.1752

Figure 13: Excerpt out of the totalNum file for 195 subjects. Variable explanations can be found in the table above.

The other matrix is called *ViewStats.mat* and contains statistics for each subject.

	1 NumHouses1	2 NumHouses2	3 NumHouses3	4 NumHouses4	5 NumHouses5
NumHousesSeen	197	178	127	203	211
PercentHousesSeen	0.9206	0.8318	0.5935	0.9486	0.9860
AverageTimeLookedAtOneHouse (s)	9.3934	11.1667	14.3207	9.2616	8.6608
TimeLookedAtHouses (min)	30.8417	33.1278	30.3122	31.3350	30.4572

Figure 14: Excerpt out of the ViewStats file for 195 subjects.

Additionally some extra statistics are calculated and can be found in the Matlab workspace under the following names:

Table 4: Analyze_ViewedHouses Workspace - Variable Names

Variable	Meaning
avg	How many houses were looked at on average.
avgper	How much percent of the houses were seen on average.
avgTime	How long on average were houses looked at (in seconds).
avgTotalTime	Average overall time looked at houses (in minutes).

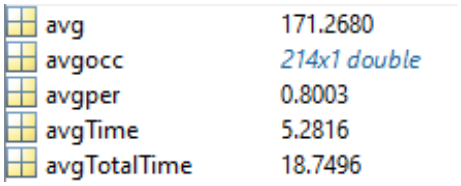

Figure 15: Example view of Matlab workspace after processing 195 subjects.

2.7 GazeStandVSWalk

This script compares gazes of subjects while they walk to gazes made by them while they stand. For this it uses the *EyesOnScreen_ VP#.txt* and the *positions_ VP#.txt* files. The results can be visualized in two different ways. The first way can be seen in figure 16 and simply plots the gazes for the two conditions in two different colors in a scatter plot. On top of this it then draws contour lines for both conditions.The orange dots and the dotted lines correspond to gazes while walking while the blue dots and the straight lines correspond to gazes while standing.

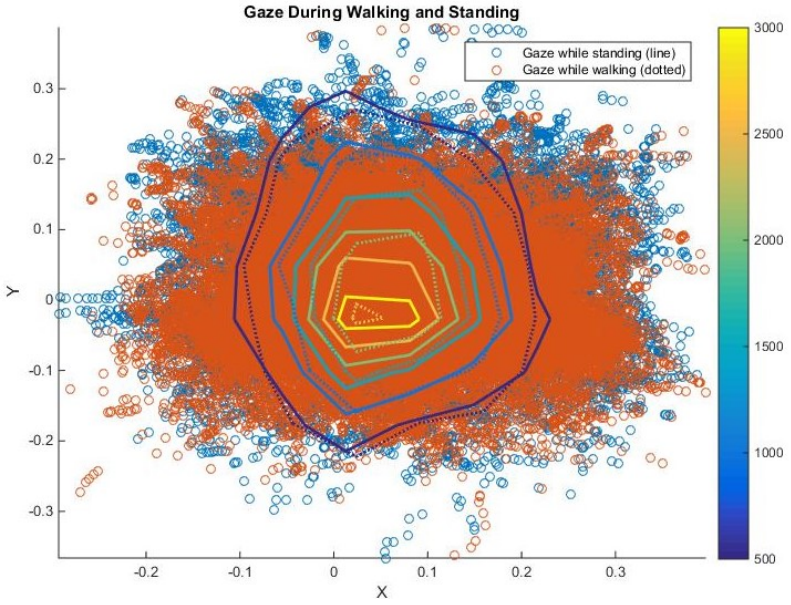

Figure 16: Example of gaze comparison plot for walking and standing in form of a scatter plot.The orange dots and the dotted lines correspond to gazes while walking while the blue dots and the straight lines correspond to gazes while standing.

The second way to visualize this information is to create heat maps and calculate the difference between the two conditions (figure 17).

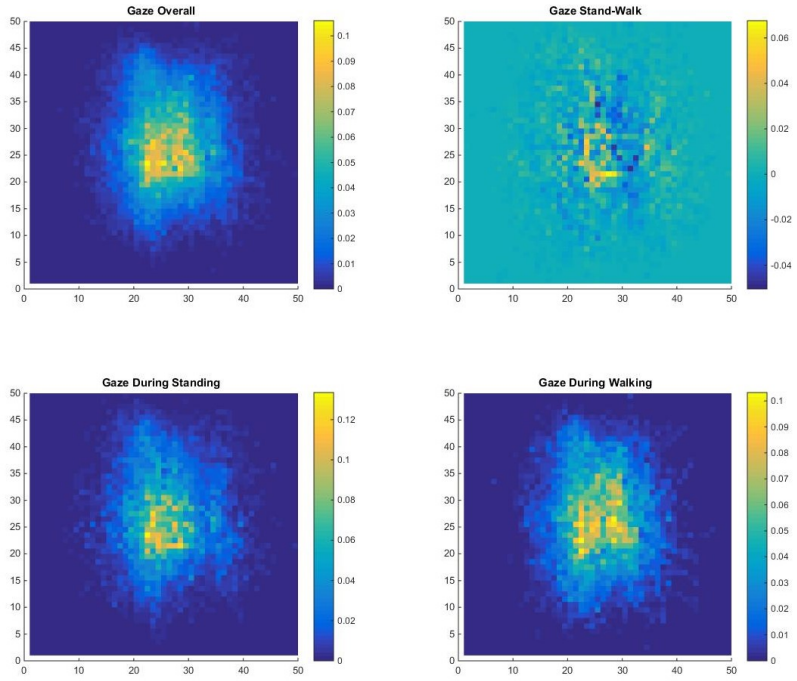

Figure 17: Example of gaze comparison plot for walking and standing in form of heat maps.(Top Left) All gazes. (Bottom Left) Gazes while standing. (Bottom Right) Gazes made while walking. (Top Right) Gazes while standing - Gazes while walking. All data is normalized.

2.8 LeftRightTurns

This script compares the gazes while turning left or right against gazes made while walking/looking straight. Just as the *GazeStandVSWalk* script it uses the *EyesOnScreen_ VP#.txt* and the *positions_ VP#.txt* files and the results can be visualized in a scatter plot or as heat maps (see figure 18).

There are two hyper parameters which can be fixed at the beginning of the script. The first is the *IntervalLength* which defines how big the interval is in which we look for a significant turn and also how many gazes we will then count as a turn. This means that if we for example have an interval length of ten we look if the rotation between time step t-10 to time step t what significant and then count the gazes of these ten frames as gazes during a turn. The second hyper parameter is *TurnSignificance* which sets by how many degrees the rotation at t-*IntervalLength* has to differ from the rotation at t to count as a turn.

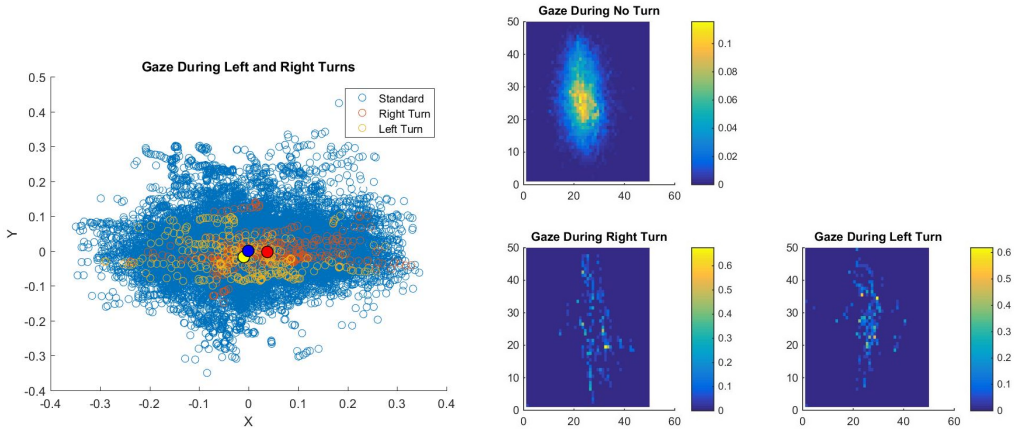

Figure 18: Example of gaze comparison plot for turning left and right. In this case we set the interval length to ten frames (1/3 second) and turn significance to 20 degree. (Left) Scatter plot where red dots correspond to gazes during right turns and yellow dots to gazes during left turns. Blue dots correspond to all gazes without a significant turn. The filled dots represent the means of the three classes respectively. (Right) Same three classes visualized as heat maps.

2.9 Entropy

This script looks at the entropy in viewed houses over a certain interval. The interval length can be defined as a hyper parameter at the beginning of the script. The entropy for an interval is calculated by creating a histogram over the houses seen in this interval and then calculating the entropy of this histogram. This means that if we for example have an interval length of ten seconds and in this time the subject has looked at a lot of different houses it will have a high entropy. If it has only looked at one or two houses in this time it will have a low entropy in this interval. We calculate the entropies with a sliding window over the time of the experiments and in the end save an average entropy for each subject. The table with all average entropies is saved in *Tracking\TaskPerformance\Entropy_ IntLen_ #SJs.mat*. It consists of two lines and as many columns as subjects you analyze. The first row gives the subject number, the second row gives the average entropy for this subject.

1	2	3	4	5	6
'1003'	'1081'	'1093'	'1119'	'1155'	'1162'
0.8842	0.9916	0.5977	1.0392	1.1959	0.6593

Figure 19: Excerpt out if the *Entropy_ IntLen_ #SJs* matrix. First row is the subject number, second row the average entropy for this subject.

These entropies can then be correlated with the performance in the different tasks. For this you need to run the second and third section of the script and it will produce a plot as shown in figure 20. The figure is also automatically saved in the results folder inside of the *TaskPerformance* folder.

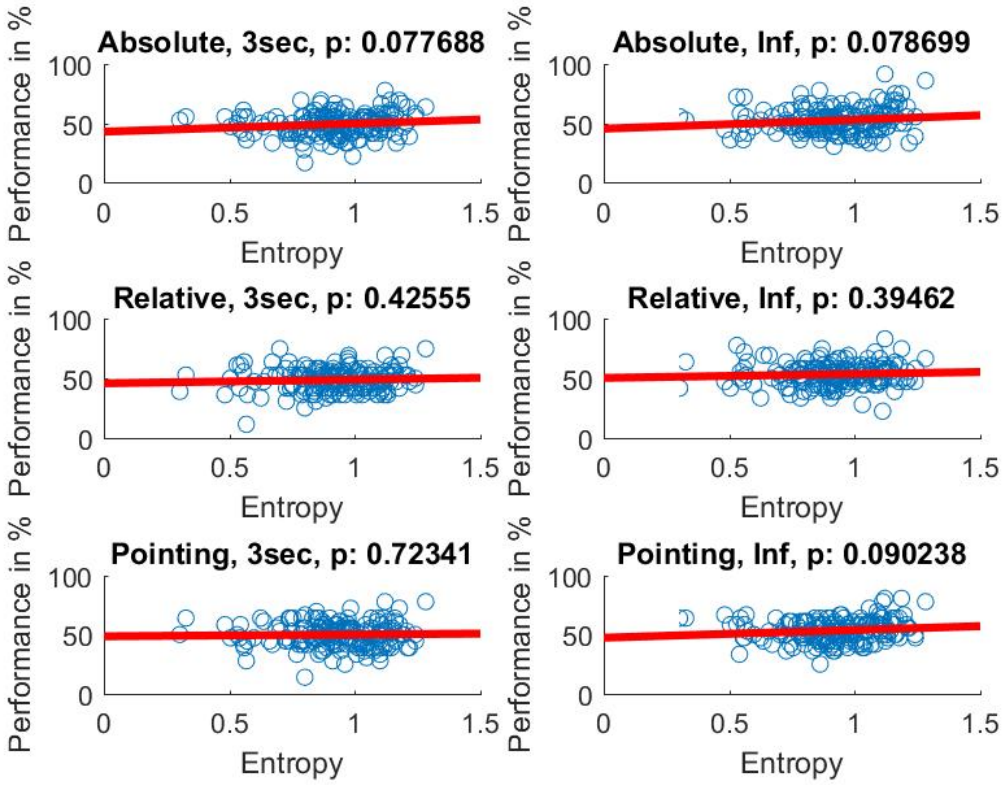

Figure 20: Average entropy correlated with the performance in the six tasks. These are the results of 194 subjects while looking at a two second interval. On the x-axis we can see the average entropy and on the y-axis the performance in the respective task in percent. Each blue circle represents one subject. The red line is a regression line fitted through the distribution of points. In the title you can see the respective p values.

2.10 PerformanceAnalysis

This script analyzes the performance of subjects in the alignement tasks which are following the 30 minute training session. Since this experiment is usually executed on a different computer the result files (*AlignementVR_SubjNo_#.mat*) have to be transfered manually into the *Tracking* folder. They can however also be analyzed in any other folder if you copy this script into the same folder and change the path referring to the *ViewedHouses* files (variable *dname* can be set at beginning of script).

The first section of the script performs an overall analysis of the task performance and returns average performance for all six tasks in the command line (figure 23).

```
Im Absolute Task 3sec korrekt: 49.6687%
Im Absolute Task Inf korrekt: 53.0199%
Im Relative Task 3sec korrekt: 49.21%
Im Relative Task Inf korrekt: 53.657%
Im Pointing Task 3sec korrekt: 50.446%
Im Pointing Task Inf korrekt: 53.0454%
```

Figure 21: Example command line output after running first section of the *PerformanceAnalysis* script.

The next sections then analyze the task performance in respect to different variables in *NumViewsD_VP_#.mat*. This includes the time a house was looked at as well as the distance and variance. Currently this is only implemented for the absolute orientation task since here we always only see one house in each trial. For the other task it needs to be decided how to combine the variables of multiple houses that are displayed in the task.

The results are displayed in the form of box plots. For each condition there is one box for correct responses and one for wrong answers. The boxes then show the spread of reaction times

(or distance or variance) corresponding to the houses which were seen in the task and answered correct or wrong respectively.

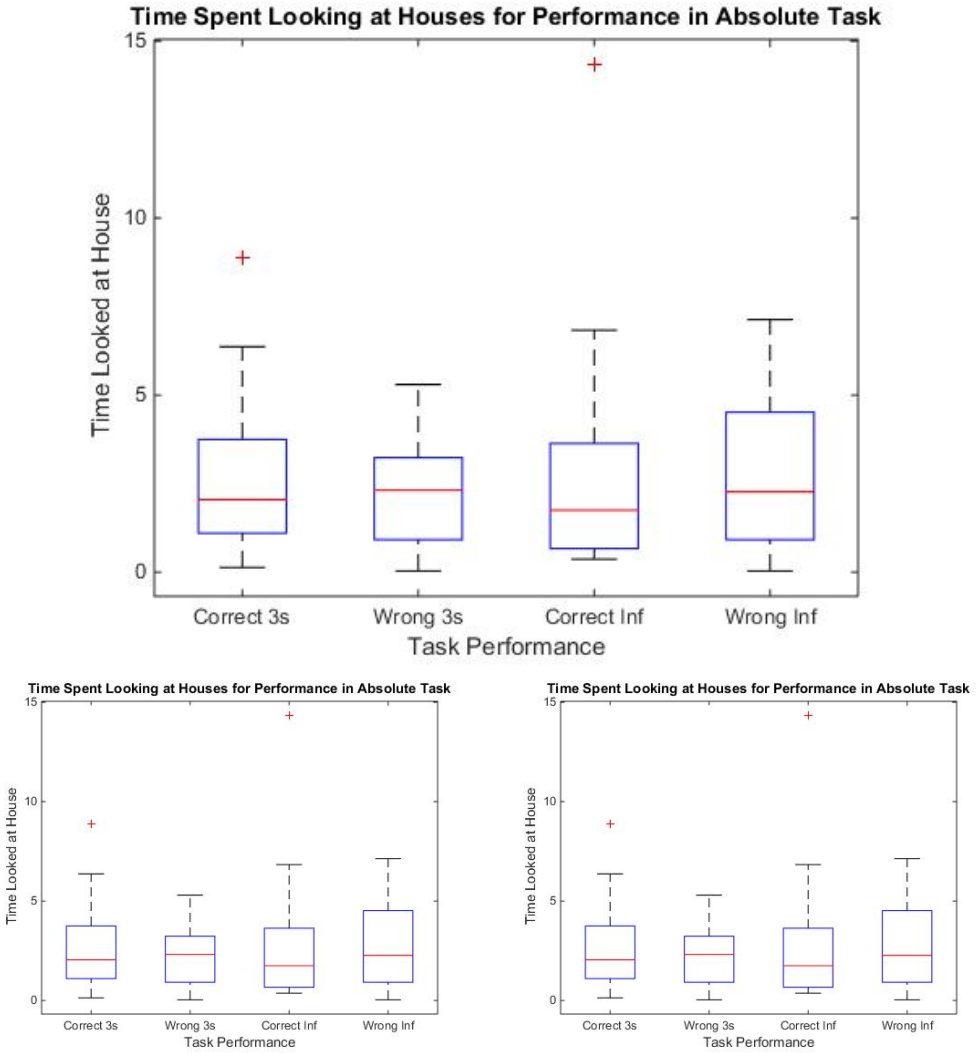

Figure 22: Example absolute task performance box plots for two subjects. (Top) Correct and wrong answers with respect to the time each house was seen. (Bottom Left) Correct and wrong answers with respect to the distance from which each house was seen. (Bottom Right) Correct and wrong answers with respect to the variance in the distances for each house in the task.

A Appendix

A.1 Raw Data Format

Table 5: My caption

File Name	Data	Script
EyesOnScreen_VP#	2D coordinates of gaze (normalized) = **(CenterX,CenterY)** or **(0.000000, 0.000000)**	PupilGazeTracker
EyeBoxPos_VP#	3D coordinates of box position: **(x,y,z)**	PupilGazeTracker
Positions_VP#	**(x,y,z,rx,ry,rz,timestamp (in sec),PupilTimeStamp)**	Recorder
Validation2D_#_NumVal	**Degree of error for each point, avg, time, last cal, error in x and y dir (+avg)**	PupilGazeTracker
ViewedHouses_VP#	**HouseViewed, distance, timestamp (sec. since start)**. If HouseViewed=NH: if distance=0: eye detected with confidence <0.5 if distance=200: no object hit (eg. Sky) else: other object than house hit	DrawViewingPath
Heatmap3D(RandomGaze/ RandomPos)_VP#	List of **(x,y,distance)** of fixations during the session	DrawViewingPath

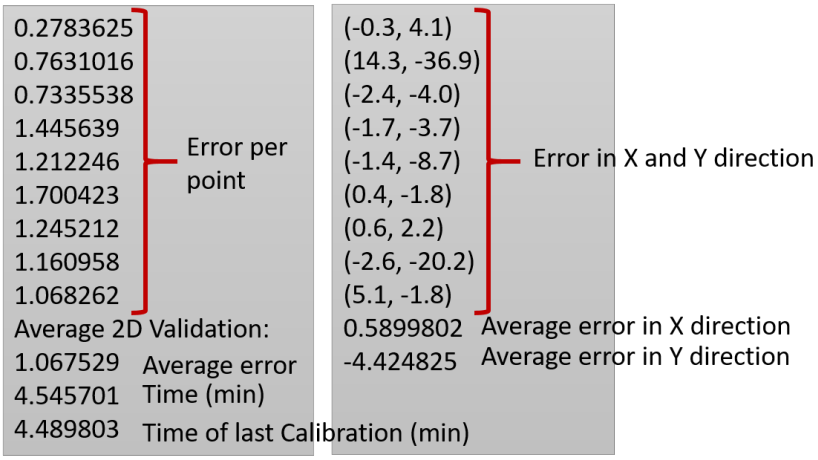

Figure 23: Detailed explanation of values in the validation files.

A.2 How to run *DrawViewingPath*

To create the *ViewedHouses_ VP#.txt* and *3DHeatmap_ VP#.txt* files you need to open the Unity project called *DrawViewingPath* which calculates the hit points of the participants. This is done offline since calculation during the session would take too many computational resources.

Once you open the *DrawViewingPath* scene (shortcut in common folder) you will see a window looking similar to the one shown in figure 24. To see the relevant script you need to click on *FPS Player* in the object hierarchy (top left). Once you click on it the inspector on the right should show properties of this object as well as all scripts attached to it. In our case we are interested in the two scripts on the bottom (right red box in figure 24).

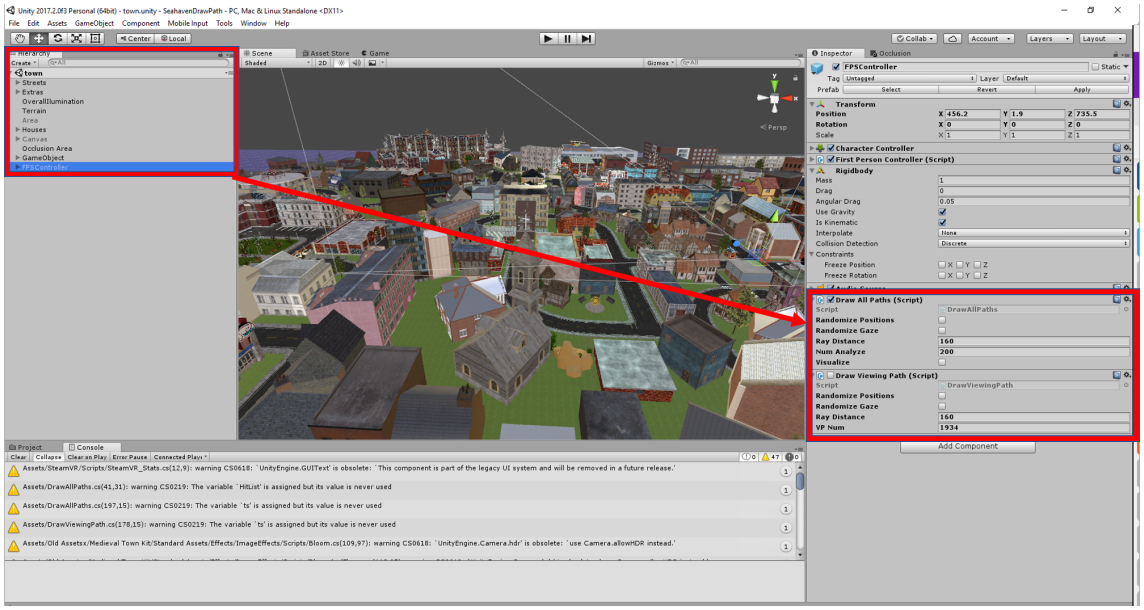

Figure 24: Unity view after opening a project. Click on FPS Player in the hierarchy to see the scripts in the inspector.

In the inspector you can now specify settings for these scripts. As you can see in figure 25 there are two scripts attached. You can activate them by checking the check box in front of the script name (in this case the script on top is activated and the other one deactivated). You should always only have one of the two scripts activated.

The *Draw All Paths* script (top) is very good when you want to analyze all your subjects at once. the *Draw Viewing Path* script (bottom) is good if you just want to analyze or visualize one specific subject.

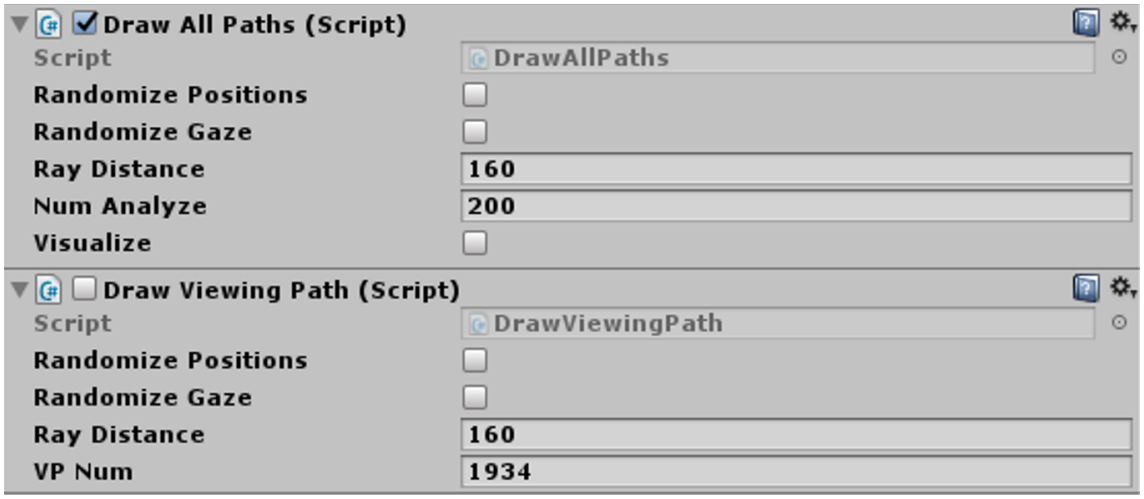

Figure 25: Unity view of script settings.

Both scripts have two randomization parameters. If you want to create randomized heat map files click one of the two randomization check boxes. An explanation what these randomizations mean can be found in the section about the Heatmap3D script.

The ray distance is usually set at 160 which is the distance up to which objects are displayed to the subjects. In these scripts it specifies how far the object detection ray is shot into the 3D world to detect intersections with houses and other objects.

In *Draw Viewing Path* you can specialize a single subject number in the *VP Num* field to fix which subject you want to analyze. In the *Draw All Paths* script you can specify the parameter *Num Analyze* instead which tells the script how many files it should analyze. It then just goes through the first n files in the *positions* folder which don't posses a *ViewedHouses.txt* file yet and analyzes them. If you want to analyze all you files simply set this to a high number [2].

The results of both scripts are always automatically saved. In the *Draw All Paths* script you have the additional option to decide if the viewing points should be visualized. In the *Draw Viewing*

[2]Remember that these calculations can take a while

Path the result is alway visualized. However, remember that if visualizing the results of multiple subjects with the *Draw All Paths* script the calculations take significantly longer than without visualization.The visualization can then look as in figure 26.

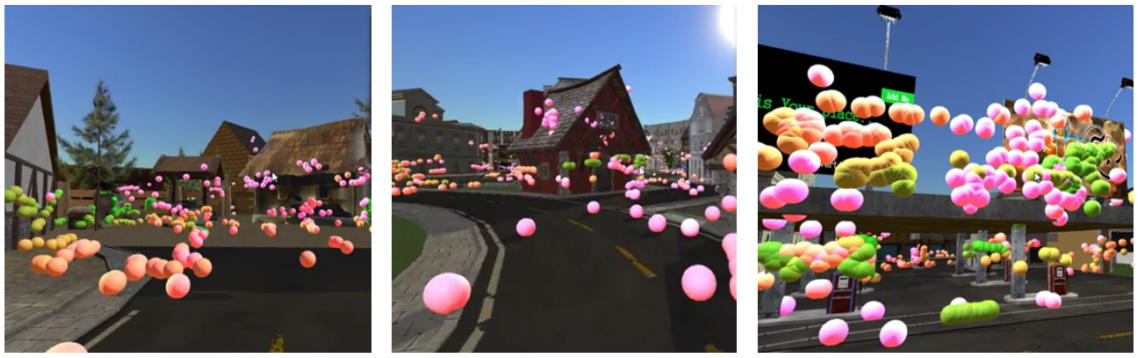

Figure 26: Gaze visualization inside of Unity. Colors correspond to the distance from which objects were looked at.