Praktikumsbericht

Oculus Rift Retina Display

von Zhejun Zhang
(03642795)

Dauer der IP: 9 Wochen
28.07.2014 – 26.09.2014

Betreuer: Prof. Dr. Jörg Conradt, Nicolai Waniek

Inhaltsverzeichnis

1 Überblick

1.1 Über das Projekt

Im Sommer 2014 habe ich mein Ingenieurpraktikum am Lehrstuhl für Neuroscientific System Theory (NST) absolviert. Seit neun Woche, vom 28.07.2014 bis 26.09.2014, habe ich mich mit dem Projekt *Oculus Rift Retina Display* beschäftigt. Das Projekt basiert auf einer Frage: Wie viele Informationen braucht man, um eine richtige Navigation in unbekannten Umgebungen zu ermöglichen? Um diese Frage zu beantworten und Experimente durchzuführen, soll eine Software entwickelt und ein Setup gebastelt werden. Und schließlich soll das ganze System getestet und eine Schlussfolgerung gezogen werden.

1.2 Aufgabenstellung

Während der meisten Zeit des Praktikums arbeitete ich mit meinem Kollege, Herrn Huaijiang Zhu, zusammen. Während Herr Zhu den Hauptteil des Programms schrieb, kümmerte ich mich hauptsächlich um die Umgebungseinstellungen für das Betriebssystem (Ubuntu) und für die IDE (Eclipse) sowie die spezifische Teile von unserem Projekt. Aber die meisten Aufgaben haben wir zusammen erledigt.

2. Technischer Hintergrund und Ausgangslage

2.1 eDVS

Ein Stichwort der Ausgangsfrage ist „Information“. Die embedded dynamic vision sensors (eDVS) bieten uns die Möglichkeit, die untere Grenze der notwendigen Informationen zu untersuchen. Der eDVS funktioniert wie unsere Netzhaut und unterschiedlich als normale Kamera werden nur die lokale Änderungen auf Pixel-Ebene gesendet, die durch Bewegung verursachtet werden. Das Ergebnis ist eine Folge von Ereignissen (a stream of events) in Mikrosekundenauflösung. Die Rechneranforderungen werden damit deutlich reduziert. [1]

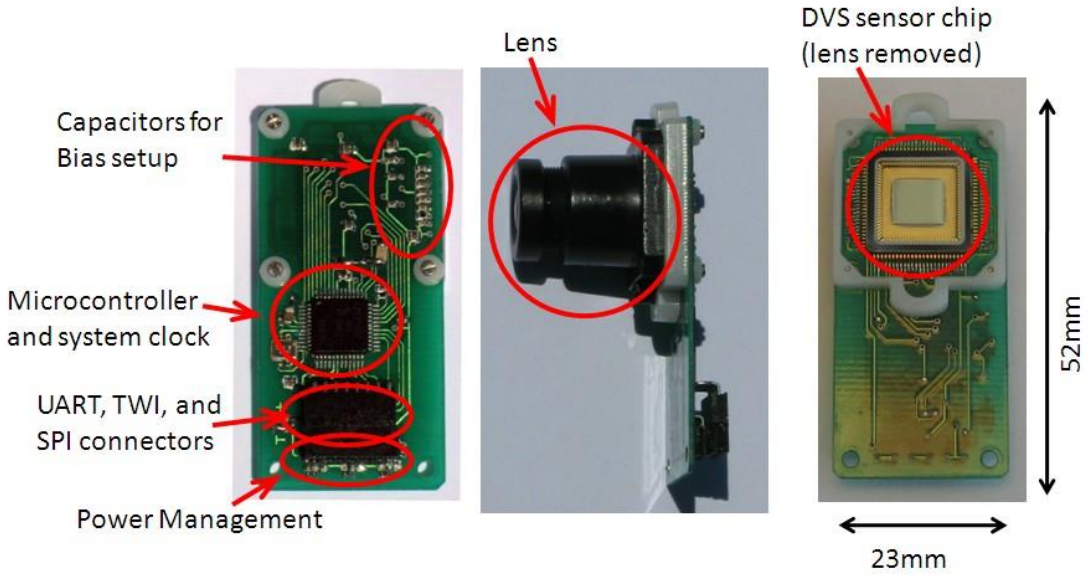

Abb. 1: Details von edvs128[1]

2.2 Oculus Rift

Die von Oculus VR[2] entwickelte VR-Brille für 3D-Spiel ist heutzutage wegen des großem Sichtfeld und der schnellen Bewegungssensoren die beliebteste HMD[3]. Im März 2013 wird die VR-Brille (Virtual Reality Headset) Oculus Rift DK1 (erster Development Kit) veröffentlicht. Das Display vom DK1 besitzt eine Bildauflösung von 1280*800 Pixeln, was einem Seitenverhältnis von 16:10 entspricht. Das Display wird in zwei separate Bilder für beide Augen geteilt, deswegen hat das einzelne Bild eine effektive Auflösung von 640*800 und ein Seitenverhältnis von 4:5. [2]

[1] https://wiki.lsr.ei.tum.de/nst/programming/edvsgettingstarted
[2] http://www.oculusvr.com/
[3] Head-Mounted Display, wörtlich „am Kopf befestigte Anzeige“

Abb. 2: Oculus Rift Development Kit 1[4]

2.3 API

In unserem Programm haben wir hauptsächlich drei Programmierschnittstelle (API) angewendet, OpenGL 3.3[5], Oculus SDK v0.3.2[6] und edvstools von NST Lehrstuhl. Das OpenGL ist eine Spezifikation für eine API, die zum Rendern von 2D und 3D-Grafiken verwendet wird.[7] Das Oculus SDK und die vom Lehrstuhl NST entwickelten edvstools bieten die API zwischen Oculus Rift und Computer sowie eDVS. Mithilfe dieser beiden Schnittstellen wird die Kommunikation zwischen alle Hardware leicht realisiert.

[4] http://de.wikipedia.org/wiki/Datei:Oculus_Rift_Dev_Kit.jpg

[5] http://www.opengl.org/sdk/docs/man3/

[6] https://developer.oculusvr.com/?action=dl&v=16

[7] http://www.opengl.org/wiki/FAQ

3. VR-System: „RetinaRift“

3.1 Konzeption

Das System „RetinaRift“ besteht aus Hardware- und Softwareteil. Im Hardwareteil haben wir zwei eDVS Sensoren als Kamera und ein Oculus Rift DK1 als Bildschirm benutzt. Dazu noch ein Laptop und insgesamt drei USB-Kabel[8], um die Kommunikation und Datenverarbeitung zwischen Oculus Rift und eDVS zu ermöglichen. Als Software wird ein Programm auf C++ mit Eclipse (v3.8) und im Ubuntu (14.04 LTS) erstellt. Das auf OculusSDK, OpenGL und edvstools basierte Programm funktioniert nicht anders als eine Schnittstelle zwischen Oculus Rift und zwei eDVS.

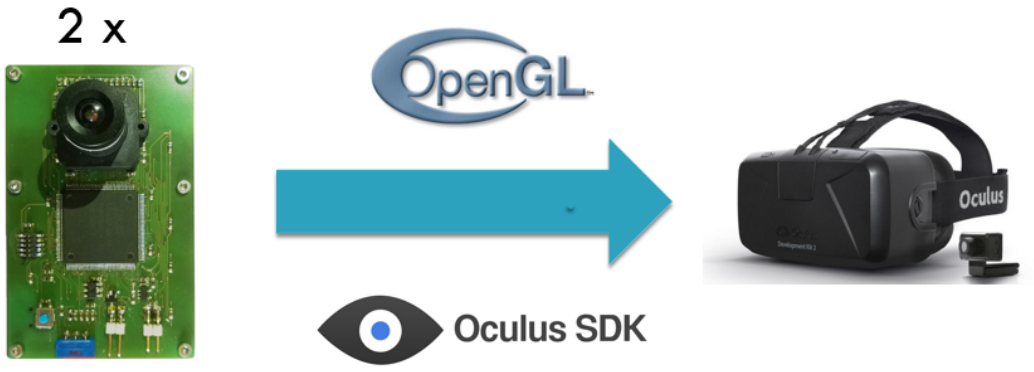

Abb. 3: Zugrundeliegende Konzeption des Systems „RetinaRift“ [9]

3.2 Methodik

Der Hauptteil des Programms ist eine do-while Schleife. In jeder Schleife werden zuerst zwei Events Stream mithilfe edvstools aus beiden eDVS abgeholt und in zwei malloc-Folgen gespeichert. Die Größe der Folge kann durch eine Konstante, *Max-event-num*, leicht gesetzt werden. Normalerweise ist diese Konstante kleiner als die Anzahl der Events, die ein eDVS in einer einzelnen Schleife zurückgibt, deswegen dient diese Konstante auch als die erste Filterung. Und jetzt ist jedes Element in diesen beiden malloc-Folge ein Event, das aus vier Komponenten besteht, nämlich Abszisse x (0-127), Ordinate y (0-127), Polarität p (0/1) und Zeit t (Systemzeit). Bevor wir mit dem Rendern-Prozess anfangen, wird die beiden Events Stream nochmal durch eine Variable (*updateinterval*) gefiltert werden. Diese Variable definiert das kleinste Zeitintervall zwischen zwei Rendern-Prozesse eines Punktes mit

[8] USB 2.0 Mini Kabel, USB-A Stecker-Mini USB-B Stecker
[9] OpenGL logo: https://www.khronos.org/news/logos/

bestimmter x und y. Für jeden Punkt wird die Systemzeit nach seinem letzten Rendern gespeichert und diese Zeit wird immer mit der aktuellen Zeit verglichen. Ein Punkt wird erst auf dem Oculus Rift gerendert, wenn die Zeitdifferenz größer als die *updateinterval* ist. Das heißt, während des Zeitintervalls wird ein bestimmter Punkt höchstens einmal gerendert. Anschließend benutzen wir OpenGL 3.3 und malen für jeden Event in den gefilterten Folgen einen Punkt, der je nach seiner Polarität schwarz oder weiß ist. Das originelle Bild muss durch Oculus SDK zum ersten Mal verzerrt (Barrel Distortion, tonnenförmige Verzeichung) und dann auf dem Oculus Rift ausgegeben werden. Das durch Software verzerrte Bild wird im Headset nochmal durch Linse physikalisch verzerrt (Pincushion Distortion, kissenförmige Verzeichnung). Und schließlich sieht man das unverzerrte Bild im Oculus Rift genauso wie das originelle direkt nach der Verarbeitung von OpenGL.

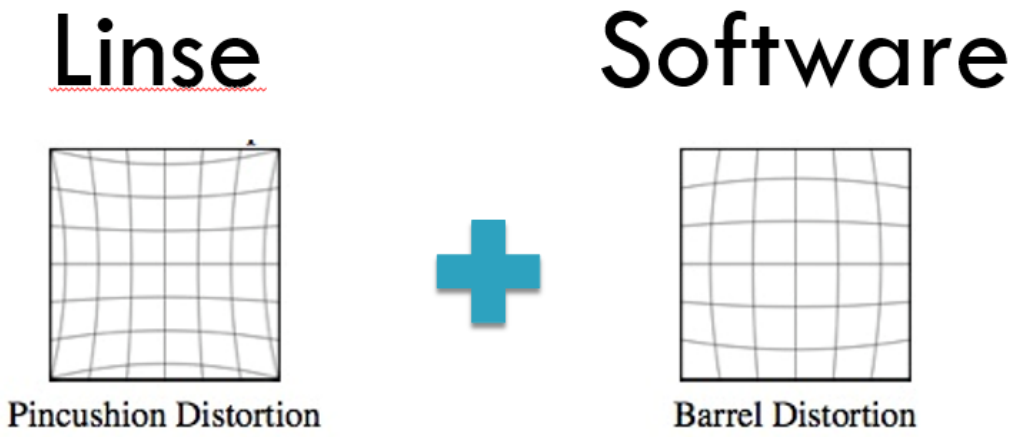

Abb. 4: Skizze von den beiden Verzeichnungen[10]

3.3 Stereo Vision und Zubehör

In diesem Projekt haben wir die Bilddaten aus beiden eDVS separat auf den beiden Bildschirmen vom Oculus Rift gerendert, um eine 3D-Vision zu realisieren. Der Abstand zwischen zwei Sensoren ist quasi gleich wie unserem Augenabstand, damit versuchen wir, die menschliche Vision zu simulieren. Dazu haben wir ein einfaches Zubehör gebastelt, damit die beiden Sensoren an dem Oculus Rift festgestellt werden kann. Das Zubehör besteht hauptsächlich aus Polyethylene-Platten, die vom Laserschneider geformt werden. Mithilfe zwei kleinen Stück Schaum kann man das Oculus Rift leicht in diesen Zubehör stecken und das ganze System stabilisieren. Eine Bewertung des 3D Effektes vom RetinaRift befindet sich im Kapitel 4.

[10] Quell: Oculus_SDK_Overview.pdf aus OculusSDK v0.3.2

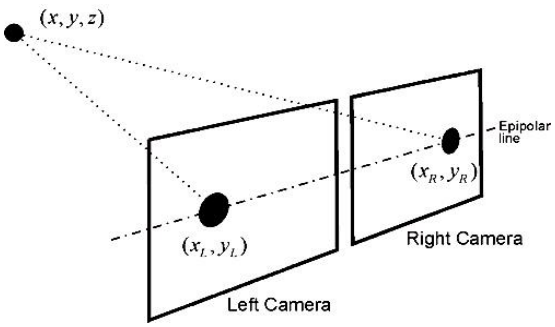

Abb. 5: Schema: Stereo Vision[11]

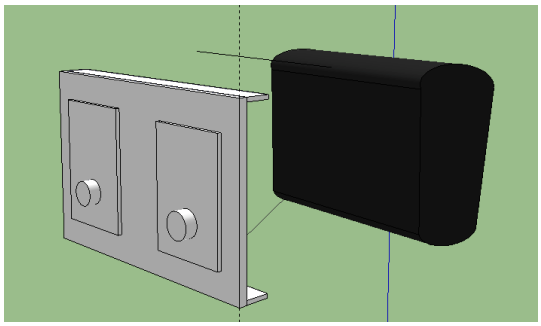

Abb. 6: Skizze des Zubehöres[12]

3.4 Parametereinstellungen

Der Benutzer des Systems „RetinaRift“ kann verschiedene Parameter per Tastatur ändern, auch wenn das RetinaRift getragen wird. Einige Parameter spielen eine wichtige Rolle für 3D-Vision, während andere uns dabei helfen, die untere Grenze des Bedarfs von Information zu untersuchen. Die Parameter sind individuell unterschiedlich. Eine vollständige Liste über Parametereintellungen ist unten zu sehen.

	Liste 1: Parametereinstellunen
Up arrow:	to enlarge the field of view.[13]
Down arrow:	to shrink the field of view.
Left arrow:	to narrow the interpupilar distance.
Right arrow:	to widen the interpupilar distance.
W:	to increase the displaying time of every single event.
S:	to decrease the displaying time of every single event.
A:	to decrease the number of events to be displayed per unit time.
D:	to increase the number of events to be displayed per unit time.
F1:	to save the current settings in /WORKSPACE_LOC/retinarift/src/settings.txt.
F2:	to load the existing settings in /WORKSPACE_LOC/retinarift/src/settings.txt.

Die Größe des Sichtfeldes und der Abstand zwischen den Pupillen dienen dazu, Bildüberlappung zu vermeiden und 3D Effekt zu verbessern. Die Anzeigedauer und die Anzahl der Events pro Zeiteinheit teilen uns die Untergrenze des Bedarfs der Information mit. Wie oben genannt kann die Anzahl des Events durch die Variable *updateinterval* und die Konstante *Max-event-num* verändert werden. Experiment zeigt, dass eine *updateinterval* von 1000 Microsekunden und eine *Max-event-num* von 1024 den besten 3D Effekt liefern. Weitere Forschung muss durchgeführt werden, um die

[11] Source: http://123nonstop.com/biography/Stereo_Vision
[12] Erstellt von Huaijiang Zhu mithilfe Coreldraw
[13] Abkürzung: FOV, auf Deutsch Sichtfeld

Bedeutung der Folgerung zu erklären.

3.5 Projektprozess

Das gesamte Projekt ist laut des Arbeitsplans in sieben Teile gegliedert: Output Side, Input Side, Kommunikation, Software, Datenfilter, Experiment und Erstellen des Berichtes. Detaillierte Angabe ist im Arbeitsplan des Projektes zu sehen. Der Hauptteil von dem Projekt ist Software. Es kostete mehr als einen Monat, um das Programm zu entwickeln und die Programmierumgebung kennenzulernen. Hier ist ein Überblick über den ganzen Prozess. Als erstes haben wir ein einfaches OpenGL Beispielprogramm auf dem Oculus Rift Developer[14] Forum gefunden. Es dauerte einige Tage, bis das Beispiel auf unserem Computer funktionierte, und erst dann sind wir breit, OculusSDK zu benutzen. Wir verwendeten einige Tagen, OpenGL 3.3 zu lernen, denn das Beispiel aus Forum benutzt veraltete OpenGL. Mithilfe OpenGL Tutorials[15] haben wir geschafft, einen farbigen Würfel zu malen. Danach versuchten wir, diesen Würfel auf dem Oculus Rift zu rendern. Nachdem uns Herr Nicolai Warniek zwei eDVS gegeben hatte, warfen wir schnell einen Blick auf das edvstools. Das edvstools wurde auf unser Programm angewendet, um Events aus den beiden eDVS zu holen. Es ist nicht schwer, das OpenGL-Teil des Programms zu ändern, damit anstatt eines Würfels Daten von den eDVS auf den Rift auszugeben. Zum Schluss haben wir verschiedene Funktionen hinzufügt, ein README erstellt und die Struktur des Projekt optimiert, sodass das Programm leicht auf anderen Computer transplantiert werden kann. Abb. 7 zeigt das endgültige Ergebnis der Softwares.

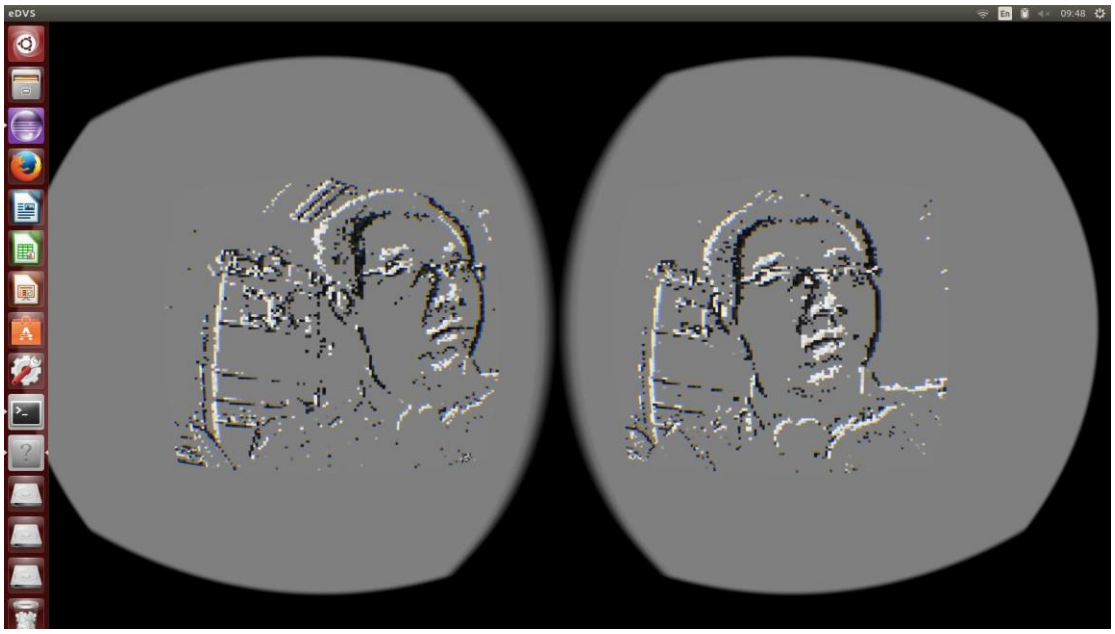

Abb. 7: Ein Selfie vom System „RetinaRift“

[14] https://developer.oculusvr.com/forums/
[15] www.opengl-tutorial.org/

4. Bewertung des Systems „RetinaRift“

Ein informelles Experiment wurde durchgeführt, um das 3D Effekt des Systems „RetinaRift“ zu bewerten. In diesem Experiment werden den Probanden zwei Aufnahmen gezeigt, in denen sich nichts außer einer Streife in unterschiedlichen Abständen vom Aufnahmestandort befindet. Damit versuchen wir, die Probanden nicht auf die Größe der Streife zu konzentrieren, sondern die Entscheidung allein mithilfe 3D Vision zu treffen. Danach werden die Probanden gefragt, in welcher Aufnahme die Streife näher ist. 3 von den insgesamt 20 Erprobungen werden falsch beurteilt. Unten wird die genauere Statistik beigefügt.

Abstand (Aufnahme 1) in cm	Abstand (Aufnahme 2) in cm	Beurteilung
3	4	richtig
4.5	3.5	richtig
3.5	4	richtig
5.5	5	richtig
3	3.5	richtig
5	4	richtig
2.5	3	falsch
5.5	4.5	richtig
4	4	falsch
5	4.5	richtig

Liste 2: Proband 1

Abstand (Aufnahme 1) in cm	Abstand (Aufnahme 2) in cm	Beurteilung
2.5	3	richtig
4	4.5	richtig
3.5	3.5	falsch
5	4	richtig
3	5	richtig
3.5	4.5	richtig
4	4.5	richtig
5	2.5	richtig
3	3.5	richtig
5	5.5	richtig

Liste 3: Proband 2

Die Ergebnisse sind zwar perfekt, aber laut der Probanden ist die Beeinflussung durch die Größe der Streife nicht zu vermeiden, deswegen muss das Experiment noch optimiert werden. Leider ist diese Optimierung wegen Zeitmangel bei uns während des Projektes nicht mehr möglich. Hier können nur einige subjektive Schlussfolgerungen ohne Unterstützung vom Experiment gezogen werden. Das

System „RetinaRift“ bietet tatsächlich eine 3D-Vision, denn die Probanden können mithilfe RetinaRift alltägliche Tätigkeit schaffen und der meisten Teilnehmer erzählen, dass es einen sehr starken 3D Effekt innerhalb 20cm bis 50cm gibt. Wir vermuten, dass der Grund hauptsächlich in dem unveränderlichen Kameraabstand und Fokuspunkt liegt. Meiner Meinung nach, dass mit zwei drehbaren eDVS und einem auf Mathematik und Bildverarbeitung basierten Bildfusionsprogramm das System „RetinaRift“ eine richtige 3D-Vision realisieren könnte.

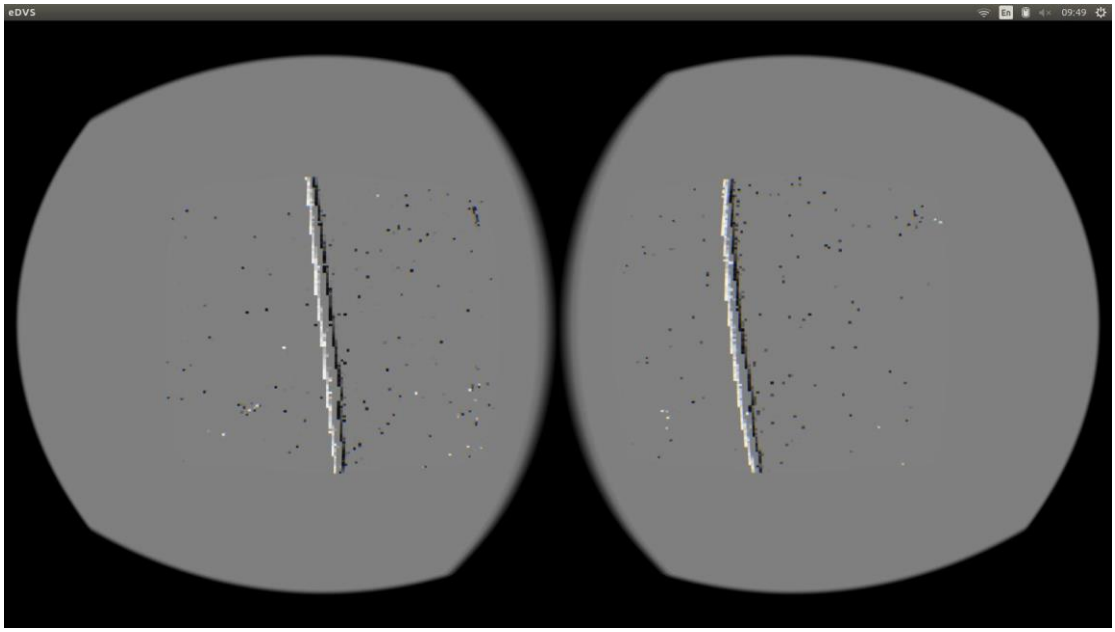

Abb. 7: Eine der Aufnahmen für das Experiment

5. Zusammenfassung

Vorher haben wir nicht erwartet, dass wir während dieser neunwöchigen Ingenieurpraktikums alle im Arbeitsplan stehenden Ziele erledigt können. In diesem Ingenieurpraxis habe ich viel gelernt, hauptsächlich Projektmanagement und die Erstellung von Projektbericht auf Deutsch. Die Zusammenarbeit mit Herrn Huaijiang Zhu war sehr angenehm. Während Herr Zhu in seinem Bericht mit OpenGL und OculusSDK beschäftigt, lege ich den Schwerpunkt auf den ganzen Projektprozess. In diesem Bericht versuche ich, eine ausführliche Angabe von dem Projekt zu liefern. Zum Schluss muss ich meinen Betreuer, Prof. Dr. Jörg Conradt und Herrn Nicolai Waniek, dafür danken, dass sie mir so viel beigebracht und mir so viel Freiheit gegeben haben.

6. Literaturverzeichnis

[1] Lehrstuhl Neuroscientific System Theory (NST): Projekt eDVS
http://www.nst.ei.tum.de/projekte/edvs/ [Stand 16.09.2014]

[2] Wikipedia: „Oculus Rift“
http://de.wikipedia.org/wiki/Oculus_Rift [Stand 20.09.2014]