

**Philosophische** Fakultät III

Sprach- , Literatur- und Kulturwissenschaften

Institut für Information und Medien, Sprache und Kultur (I:IMSK)  
Lehrstuhl für Medieninformatik

Praxisseminar

Modul: MEI-M26.1

Sommersemester 2019

Leitung: Dr. Valentin Schwind

**Predicting Avatar Movement in Virtual Reality using Neural Networks**

Jakob Fehle, David Halbhuber, Jonathan Sasse

E-Mail: [jakob.fehle@stud.uni-r.de](mailto:jakob.fehle@stud.uni-r.de), [david.halbhuber@stud.uni-r.de](mailto:david.halbhuber@stud.uni-r.de), [jonathan.sasse@stud.uni-r.de](mailto:jonathan.sasse@stud.uni-r.de)

Abgegeben am XX.09.2019

Inhalt

[1 Einleitung 3](#_Toc20700631)

[1.1 Problemstellung und Ziel dieser Arbeit 3](#_Toc20700632)

[2 Projektmanagement 4](#_Toc20700633)

[3 Movement Prediction System 5](#_Toc20700634)

[3.1 Motion Capturing 6](#_Toc20700635)

[3.2 Intercepter Client 6](#_Toc20700636)

[3.3 Unity-Anwendung für die Benutzerstudie 8](#_Toc20700637)

[3.3.1 Aufbau der Anwendung 8](#_Toc20700638)

[3.3.2 Konfiguration und Verwendung 10](#_Toc20700639)

[3.3.3 Ergebnisse 11](#_Toc20700640)

[4 Latency Test Framework 12](#_Toc20700641)

[4.1 Zielsetzung Latency Test Framework 12](#_Toc20700642)

[4.2 Versuchsaufbau Latency Test Framework 12](#_Toc20700643)

# Einleitung

Dieses Dokument wurde parallel zur Projektdurchführung des Projekts „Predicting Avatar Movement in Virtual Reality using Neural Networks“ erstellt und dient dabei der Beschreibung des entwickelten Systems, sowie zur Anleitung zum Einsatz eben jenes Systems. Neben der Erläuterung des Einsatzes und der Dokumentation des entstanden Systems, dient dieses Dokument auch zur Projektdokumentation. Dabei werden kurz der vorgestellt entwickelte Projektplan, sowie die anschließende tatsächliche Umsetzung erläutert. Zusätzlich zu diesem Dokument, dem vorgestellten System wurde im Rahmen des Projekts auch ein wissenschaftliches Paper verfasst, welches im Repository gefunden werden kann.

## Problemstellung und Ziel dieser Arbeit

Zur Heranführung an die Thematik, soll an dieser Stelle noch einmal kurz das bearbeite Problem vorgestellt und ausgeführt werden.

Der Lehrstuhl der Medieninformatik betreibt in der TechBase ein Virtual Reality Labor. In diesem Labor ist es möglich via Motion Capturing (MoCap) menschliche Bewegungen direkt in entsprechende Anwendungen zu integrieren. Durch den Einsatz von Head Mounted Devices (HMD) ist es möglich so eine maximal immersive Virtual Reality (VR) für Anwender zu erstellen. Durch den Einsatz der vielen verschiedenen Systeme ist für den Benutzer bei der Nutzung eine Latenz spürbar. Konkretisiert bedeutet dies zum Beispiel, wenn ein Benutzer, der via MoCap getrackt wird und ein HMD benutzt, seinen Arm hebt, die Armbewegung erst mit einiger Verzögerung auf dem HMD angezeigt bekommt. Diese Latenz entsteht, wie oben schon angesprochen, durch den Einsatz der verschiedenen technischen Systeme. Der erste Schritt beinhaltet dabei, das Tracken der Benutzerbewegungen durch die MoCap Kameras, dieser Feed wird dann an den MoCap Rechner gesendet. Angekommen auf dem Rechner werden die eintreffenden Informationen verarbeitet und via Netzwerkkommunikation an einen zweiten Rechner gesendet. Der zweite Rechner wiederum muss das ankommende Signal erneut verarbeiten und hat zusätzlich die Aufgabe die erhalten Information an das angeschlossene HMD zu senden. Abschließend muss das HMD die Informationen aufbereiten und dem Benutzer anzeigen. Alle diese Prozesse benötigten Zeit – Zeit die sich schlussendlich in einer verzögerten Darstellung für den Benutzer auf dem HMD manifestiert.

Ziel dieser Arbeit ist die Entwicklung eines Systems das mit Hilfe von künstlichen Neuronalen Netze (NN) die Bewegungen eines MoCap Benutzers vorhersagen und somit die entstanden System Latenz kompensieren kann. Des Weiteren wird vermutet, dass die oben beschriebene Latenz nicht nur das Präsenz Gefühl des Benutzers verschlechtert, sondern auch negative Auswirkungen auf Performance beim Erhalten und Ausführen von bestimmten Tasks hat.

# Projektmanagement

Zur Erhöhung der Übersichtlichkeit und zur besseren Möglichkeit den Projektfortschritt zu kontrollieren, wurde das Projekt in 5 kleinere Unterprojekte unterteilt. Die einzelnen Unterprojekte wurden erneut in SCRUM ähnlichen Projetmanagement geplant und abgearbeitet. Der Projektfortschritt wurde in wöchentlich angehaltenen Projektmeetings besprochen und diskutiert. In diesen wöchentlichen Meetings wurde auch das jeweilige Ziel für die nächste Woche festgelegt. Abbildung 1 zeigt den entworfenen Projektplan mit jeweiligen Meilensteinen für die angesprochenen Unterprojekte. Unterprojekt 1 (lila) befasste sich dabei hauptsächlich mit der Einarbeitung in die verschiedenen Themen des Projekts (Virtual Reality, Machine Learning, Deep Learning, OptiTrack MoCap, Python, Unity 3D, usw.). Unterprojekt 2 (rot) hat die Entwicklung des eigentlichen Vorhersagesystems zum Bestand. Im Rahmen von Unterprojekt 2 wurde auch die Datenerhebung geplant und realisiert. Unterprojekt 3 (blau) zeigt den veranschlagten Projektplan für die Entwicklung und Integration einer Unity Anwendung für das anschließende Unterprojekt 4 (gelb), in welchem das entworfene System durch eine Nutzerstudie evaluiert werden sollte. Das letzte Unterprojekt 5 (grün) lief parallel über die gesamte Projektdauert und befasst sich ausschließlich mit dokumentarischen Arbeiten. In dessen Rahmen wurde dieses Dokument, sowie das angesprochene Paper verfasst.

Bei der Durchführung des Projektes konnte im Großen und Ganzen der geplante Ablauf des Projektplans umgesetzt werden. Lediglich die Entwicklung des Latency Test Frameworks findet im Projektplan keine Beachtung, da dieses Unterprojekt nachträglich definiert und fixiert wurde. Für die Gesamtheit des Projektsplans bedeutete dieser Einschub eine Verzögerung von etwa zwei Wochen. Dieser Verzögerung konnte jedoch bei der Entwicklung des Neuronalen Netzwerks und bei der Entwicklung des Intercepter Clients wieder kompensiert werden.

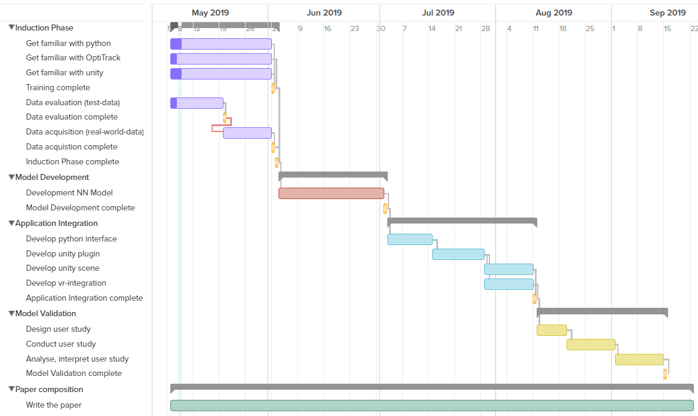


Abbildung : Projektplan, zeigt die verschiedenen Phasen (Unterprojekte) des Projekts, sowie die dafür veranschlagte Zeit.

# Movement Prediction System

Das entwickelte Movement Prediction System (MPS) besteht aus drei verschiedenen Komponenten:

1. Motion Capturing NaturalPoint
2. Intercepter Client
3. Zielanwendung, Unity-Anwendung

Im Folgenden werden die entsprechenden Komponenten vorgestellt und ausgeführt wie diese zu konfigurieren sind um das entwickelte MPS zu verwenden.

## Motion Capturing

Das entwickelte System wurde ausschließlich mit Natural Points MoCap System „OptiTrack“ entwickelt und getestet.

Für die Verwendung muss der Anwender entweder das 49 Marker Set, welches nur aus passiven Markern besteht, oder das 55 Marker Set, welches aktive Marker Handschuhe integriert, tragen. Da die Rotation und Position der Finger in dem vorgestellten System nicht vorhergesagt, sondern durch den Intercepter Client geschleust werden, hat die Wahl des Marker Sets keine Auswirkung auf das MPS. Die Darstellung der Finger wird jedoch durch Verwendung der aktiven Marker maßgeblich verbessert und ist deswegen zu empfehlen.

Neben der Wahl eines Markersets muss in der OptiTrack Software lediglich eine Änderung vorgenommen werden. Im „Streaming Pane“ der Anwendung muss der Target Data Port[[1]](#footnote-1), von 1510 auf 8205 geändert werden. Diese Änderung hat zur Folge, dass die MoCap Daten nicht mehr an die Default Anwendung gesendet werden, sondern von unserem Intercepter Client „abgefangen“ werden.

## Intercepter Client

Der vorgestellte Client wurde mit Hilfe von Python und der QT[[2]](#footnote-2) für Python entwickelt. In seiner aktuellen Version fängt der Client die OptiTrack Streaming Daten auf Port 8205 ab und sendet die verarbeiten Daten wieder auf dem OptiTrack Default Streaming Data Port 1510 ans Netzwerk. Zur Kommunikation benötigt der Client die Quelle-IP Adresse, den Quell-Port, die Ziel-Adresse sowie den gewünscht Ziel-Port. Falls die Default Konfiguration des OptiTrack Streaming Protokoll verwendet werden soll, müssen hier lediglich die Quell- und Ziel-IP Adresse angepasst werden. Neben der Konfiguration Netzwerkkommunikation kann über das Benutzer Interfaces des Clients auch der Prediction Modus gewählt werden. Dabei ermöglicht der Client in der aktuellen Version folgende Modi:

* 0 Frame Prediction (Keine Prediction)
* 12 Frame Prediction (50ms Prediction)
* 20 Frame Prediction (83ms Prediction)
* 25 Frame Prediction (104ms Prediction)
* 50 Frame Prediction (208ms Prediction)
* 100 Frame Prediction (417ms Prediction)

Getestet und evaluiert wurden lediglich die Modi; 0 Frame, 12 Frame und 25 Frame. Alle anderen Modi wurden zwar getestet, aber nicht in einer Benutzerstudie evaluiert. Neben der Wahl des Baseline- Prediction Modus kann im Client auch gewählt werden ob Finger Knochen/Gelenke ebenfalls vorhergesagt werden sollte. Diese Funktion ist experimentell und wurde zwar kurz getestet aber ebenfalls nicht in einer Benutzerstudie evaluiert. Nach entsprechender Konfiguration kann der Intercepter Client durch einen Klick auf „Intercept“ damit beauftragt werden den OptiTrack Stream zu unterbrechen und das gewählte Prediction Model einzuschleusen. Abbildung 2 zeigt den entwickelten Intercepter Client. Die aktuelle Version des Intercepter Clients kann im Repository gefunden werden. Des Weiteren kann der Intercepter Client bereits einsatzbereit und vorkonfiguriert auf dem OptiTrack Rechner im VR4 Labor in der Techbase gefunden werden.

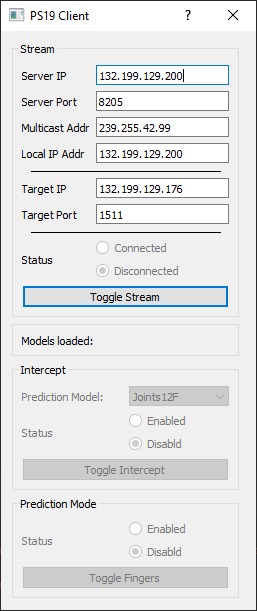


Abbildung : GUI des Intercepter Clients.

## Unity-Anwendung für die Benutzerstudie

Für die Durchführung der Studie wurde eine Unity-Anwendung entwickelt. Hierbei werden die Körperbewegungs-Daten des Motion-Capturing-Systems auf ein Modell in der Unity-Scene projiziert. Die Anwendung benötigt keinerlei erweiterte Anpassung bezüglich der Vorhersage der Körper-Bewegungen.

### Aufbau der Anwendung

Die Anwendung kann in drei Segmente unterteilt werden:

#### Körper-Darstellung

Beim Start der Anwendung befindet sich der Benutzer ohne erweiterte Konfiguration in einer Replik des Studios.

Hier können die (prognostizierten) Bewegungen des Körpers anhand von freiem Bewegen oder vordefinierten Tätigkeits-Aufgaben nachempfunden werden.

#### Bewertung der Selbst-Wahrnehmung

Wie der Körper beziehungsweise dessen Bewegungen in der vorhergehenden Phase wahrgenommen wurde, kann anhand eines Fragebogens evaluiert werden. Dieser kann bei Bedarf aktiviert werden. Wobei ein Start-Button erscheint, der mit einer der beiden Hände per „Berührung“ betätigt werden kann. Die im folgenden dargestellten Fragen besitzen jeweils fünf Antwort-Optionen. Mit der Selektion einer dieser Optionen erscheint ein „Weiter“-Button bis alle Fragen beantwortet wurden und der Fragebogen wieder automatisch deaktiviert wird.

Der Fragebogen selbst untergliedert sich wiederum in zwei inhaltlich unterschiedliche Teile. Während einer dieser Fragen aus dem IPQ-Fragebogen[[3]](#footnote-3) beinhaltet, enthält der andere Fragen zur „Limb Ownership“[[4]](#footnote-4).

Bei jeder Ausführung des Fragebogens werden die Fragen zufällig geordnet, wobei die beiden Teile in ihrer Reihenfolge statisch bleiben.

#### Performance Task

In diesem Segment wird die Performance Task ausgeführt. Hierbei werden die beiden Whiteboards in der Szene aktiviert und auch in der realen Welt entsprechend positioniert.

In der Benutzer-Studie wird der Proband dazu angehalten, die beiden Whiteboards alternierend zu berühren, während der Studienleiter die Berührungen zählt und notiert. Es werden keine digitalen Daten diesbezüglich von der Anwendung erfasst.

### Konfiguration und Verwendung

#### Konfiguration für OptiTrack

Für die Verwendung der gestreamten Körperdaten von OptiTrack muss ein entsprechendes Plugin in Unity installiert und konfiguriert werden. Das Unity Plugin kann auf der Homepage von Natural Point bezogen werden[[5]](#footnote-5). Es muss wie in **Abbildung 3** dargestellt konfiguriert werden. Wobei es sich bei **„Server Address“** um die Host-IP des Motion Capturing Systems handelt. Zusätzlich muss noch die Eigenschaft „Skeleton Asset Name“ des GameObjekts „Avatar“ auf die zu empfangende Skelett-ID, die im OptiTrack Client definiert wird, geändert werden.

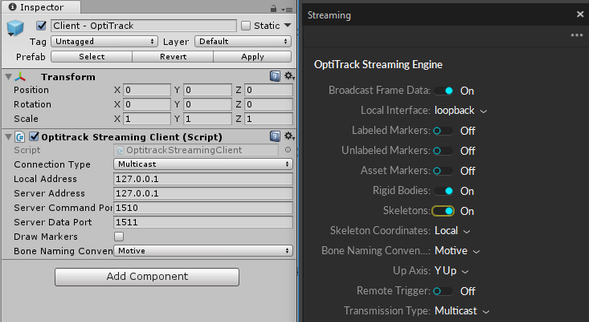


Abbildung 3: OptiTrack Stream Unity Konfiguration

#### Standard Szene

In der Start-Szene sollte das Fragebogen-Objekt („Questionnaire“), die beiden inneren Whiteboards („FlipchartRight“, „FlipchartLeft“) deaktiviert und die beiden äußeren Whiteboards („FlipchartRight\_Back“, „FlipchartLeft\_Back“) aktiviert sein.

#### Konfigurationen vor der Studie

Vor Durchführung der Studie muss die Probanden-ID („personId“) im Fragebogen-Objekt („Questionnaire“) festgelegt werden, um die Ergebnis-Daten einer Person zuordnen zu können.

#### Konfigurationen während der Studie

Während der Studie muss bei Bedarf das Start-Fragebogen-Objekt („StartQuestionnaire“) aktiviert werden und die Whiteboards für die Performance-Task entsprechend eingeblendet werden.

### Ergebnisse

Während die Ergebnisse der Performance-Task ausschließlich manuell vom Studienleiter erfasst werden, werden die abgegebenen Antworten der Fragebögen in einer CSV-Datei im Projekt-Verzeichnis gesichert.

Für jede Antwort wird eine neue Zeile in der CSV-Datei angehängt (zum Beispiel „*5,6,4,20190823144016*“). Das Format hierbei setzt sich wie folgt zusammen: Probanden-ID, Frage-ID, Antwort-ID und ein Zeitstempel.

# Latency Test Framework

Da das entwickelte und im Paper vorgestellte Latency Test Framework (LTF) bei der initialen Projektplanung nicht beachtet werden konnte (der Projekt-umfang und -inhalt war zu diesem Zeitpunkt noch nicht klar definierbar), wird es auch in diesem Dokument gesondert behandelt.

## Zielsetzung Latency Test Framework

Das Ziel des entwickelten LTFs ist es die Latency die im gesamten System (MoCap-System -> MoCap-Server -> Netzwerk -> Unity-Anwendung -> HMD) vorherschert, verlässlich zu ermitteln. Der ermittelte Wert wird in anschließenden Anwendungen das Null-Latenz Model definiere, also das Model bei dem davon auszugehen ist, dass der Benutzer keine Latenz mehr erfährt oder wahrnehmen kann.

## Versuchsaufbau Latency Test Framework

Der Versuchsaufbau des LTFs, wie auch beschrieben im zugehörigen Paper, besteht aus:

* Arduino Mikrocontroller
* HMD HTC Vive Go
* Motion Capturing NaturalPoint
* Unity-Anwendung
* Externer Rechner zur Latenzmessung

Im Versuch wurde der Arduino mit einem Vibrationssensor, sowie einem lichtempfindlichen Photosensor erweitert. Der gesamte Versuch wird gestartet in dem ein Benutzer ein Objekt auf eine Holzplatte fallen lässt. Die Objektbewegungen werden dabei vom MoCap System getrackt und an die Unity-Anwendung weitergegeben. Die Holzplatte ist über den Vibrationssensor mit dem Arduino verbunden. Sobald der Vibrationssensor einen Einschlag detektiert wird vom Arduino ein Zeitstempel geschrieben und an den externen Rechner weitergegeben. Durch das Tracken und die Weitergabe des fallenden Objekts wird auch in Unity einen Einschlag erkannt sobald das Objekt die Holzplatte trifft. Die Kollision in Unity löst einen Befehl aus, der das angeschlossene HMD erhellt. Diese Veränderung in der Helligkeit wird von dem angeschlossenen Photosensor registriert und reagiert erneut damit das der Arduino Mikrokontroller einen Zeitstempel schreibt und an den externen Rechner sendet. Dem externen Rechner liegen nun zwei Zeitstempel vor; Einmal der Zeitpunkt des tatsächlichen Aufpralls auf die Holzplatte und einmal der Zeitpunkt zu welchem der Einschlag in Unity registriert und an das HMD gesendet wurde. Durch den Vergleich der beiden Zeitstempel kann nun ermittelt werden, wieviel Zeit zwischen realem Vorgang und der Repräsentation in VR vergangen ist. Diese Zeitspann steht nun für die ermittelte systemweite Latenz. Abbildung 3 zeigt eine schematische Darstellung des Versuchsaufbau. Der so ermittelte Mittelwert der Latenz beläuft sich auf 50,3ms (n=120, SD=6,9ms). Abbildung 4 zeigt eine erste Iteration des Versuchsaufbaus. Das gezeigte HMD wurde in einer späteren Iteration durch das kabellose Pendant getauscht.

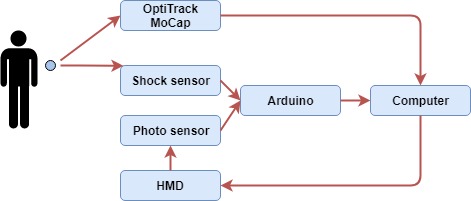


Abbildung 3: Schematische Darstellung des LTF Workflows zur Ermittlung der Systemlatenz



Abbildung 4 Erste Versuchsaufbau des Latency Test Framework. HMD wurde in finaler Version gewechselt.

1. Vgl. <https://v21.wiki.optitrack.com/index.php?title=Data_Streaming> [↑](#footnote-ref-1)
2. Vgl. <https://riverbankcomputing.com/software/pyqt/intro> [↑](#footnote-ref-2)
3. Vgl. <http://www.igroup.org/pq/ipq/index.php> [↑](#footnote-ref-3)
4. Vgl. <https://dl.acm.org/citation.cfm?id=3225158> [↑](#footnote-ref-4)
5. Vgl. <https://optitrack.com/unity-integration/> [↑](#footnote-ref-5)