Virtueller Regisseur in Audiovisueller Gruppen-zu-Gruppen Kommunikation

Wolfgang Weiss[1], Rene Kaiser[1], Manolis Falelakis[2], Harald Mayer[1]

1 JOANNEUM RESEARCH Forschungsges. mbH, DIGITAL - Institut für Informations- und Kommunikationstechnologien, Graz, Austria, {Vorname.Nachname@joanneum.at}

2 Goldsmiths, University of London, Department of Computing, London, UK, {m.falelakis@gold.ac.uk}

Zusammenfassung

Die „*Orchestration Engine*" ist ein regelbasierter virtueller Regisseur zur Unterstützung von audiovisueller Gruppen-zu-Gruppen Kommunikation. Diese selektiert selbstständig zwischen verfügbaren Videostreams aus mehreren Orten und arbeitet mithilfe eines Satzes von pragmatischen und cinematographischen Regeln, die zusammen mit Experten aus Film und audiovisueller Kommunikation in einem Wissenserhebungsprozess erarbeitet wurden. Elementare Ereignisse, von einer audiovisuellen Analysesoftware detektiert, werden in einem mehrstufigen Prozess mit Hilfe von temporalem Reasoning verarbeitet. In einer Evaluierung konnte gezeigt werden, dass unser Ansatz schnell genug Entscheidungen trifft um den Echtzeitanforderungen zu genügen.

1 Einleitung

Die Aufgabe eines virtuellen Regisseurs (engl. *virtual director*) in einer audiovisuellen Gruppen-zu-Gruppen-Kommunikation (Williams et al., 2011) ist es, aus einer Reihe von verschiedenen Kameras zu jedem Zeitpunkt eine auszuwählen und dabei *cinematographische* Aspekte der Ästhetik sowie pragmatische Prinzipien zur Unterstützung der Kommunikation zu berücksichtigen. Mit der Entscheidung eine bestimmte Kamera bzw. einen Audiostream auszuwählen wird nicht nur entschieden welche der parallelen Geschehnisse die Teilnehmer mitverfolgen können – es wird auch die Kommunikation innerhalb der Gruppe aktiv beeinflusst. Unsere Softwarekomponente arbeitet nach ähnlichen Prinzipien wie ein Regisseur und sein Broadcast-Team und hat das Ziel diese weitgehend zu ersetzen. Dabei steht das Erleben (engl. *immersive experience*) der Teilnehmer im Vordergrund, die Technik soll möglichst unbemerkt bleiben.

Im von der Europäischen Kommission unterstützen Forschungsprojekt TA2[1] wird ein Gruppen-Videokonferenz-System entwickelt und technologische Ansätze zur Implementierung eines virtuellen Regisseurs untersucht. Im Anwendungsszenario von TA2 gibt es mehrere örtlich voneinander getrennte Standorte mit jeweils mehreren Personen und Kameras, siehe Abbildung 1. Die *Orchestration Engine*, der virtuelle Regisseur im TA2 System, wählt für jeden Teilnehmer individuell die optimalen Kameraeinstellungen aus den jeweils anderen Standorten aus.

Um überhaupt automatisiert Entscheidungen treffen zu können, braucht es Wissen über das, was vor den Kameras passiert. Eine audiovisuelle Analysesoftware erkennt dazu elementare Ereignisse (*cues*) (Korchagin et al., 2011) im Audio- und Videostrom wie z.B. „*Person X beginnt zu sprechen*" oder „*Position des Kopfes der Person Y ist …*". Diese Basisinformationen werden in mehreren Verarbeitungsschritten auf eine semantisch höhere/abstraktere Ebene gebracht um schlussendlich die zu einem bestimmten Zeitpunkt interessanteste Kamera auszuwählen. Dabei wird kontinuierlich ein Regelwerk evaluiert, welches das gewünschte Verhalten des virtuellen Regisseurs definiert.

Wir untersuchen die Möglichkeiten und Grenzen eines regelbasierten Ansatzes und suchen nach vordefinierten Mustern und Veränderungen im Ereignisstrom und bilden mit Hilfe von logischen Schlussfolgerungen den aktuellen Status der gesamten Kommunikation ab, z.B. „*Wer ist zum aktuellen Zeitpunkt die*

[1] http://www.ta2-project.eu/

aktive Person in einem Gespräch?“. Die Herausforderung ist hier, die Ereignisse mit möglichst wenig Zeitverzögerung zu verarbeiten.

Ein Ansatz um das Problem des Auswählens von Kameras zu einem bestimmten Zeitpunkt formell zu beschreiben ist es, das gewünschte Verhalten durch Regeln auszudrücken. Auch für Menschen ohne fundiertes Wissen in der Wissensrepräsentation ist es möglich, dieses Problem in konkreten Regeln auszudrücken z.B. „*Schneide zu einem Close-Up einer Person X wenn sie gerade spricht und vorher ein Wide-Shot gezeigt wurde.*“

Im Folgenden präsentieren wir die Orchestration Engine und diskutieren die Herausforderungen und Erkenntnisse, die bei der Umsetzung einer solchen Softwarekomponente entstehen. Ferner wird die Implementierung des Regelwerks und die Untersuchung zeitlicher Abhängigkeiten, die Softwarearchitektur sowie die Ergebnisse aus der Evaluierung einzelner Komponenten beschrieben.

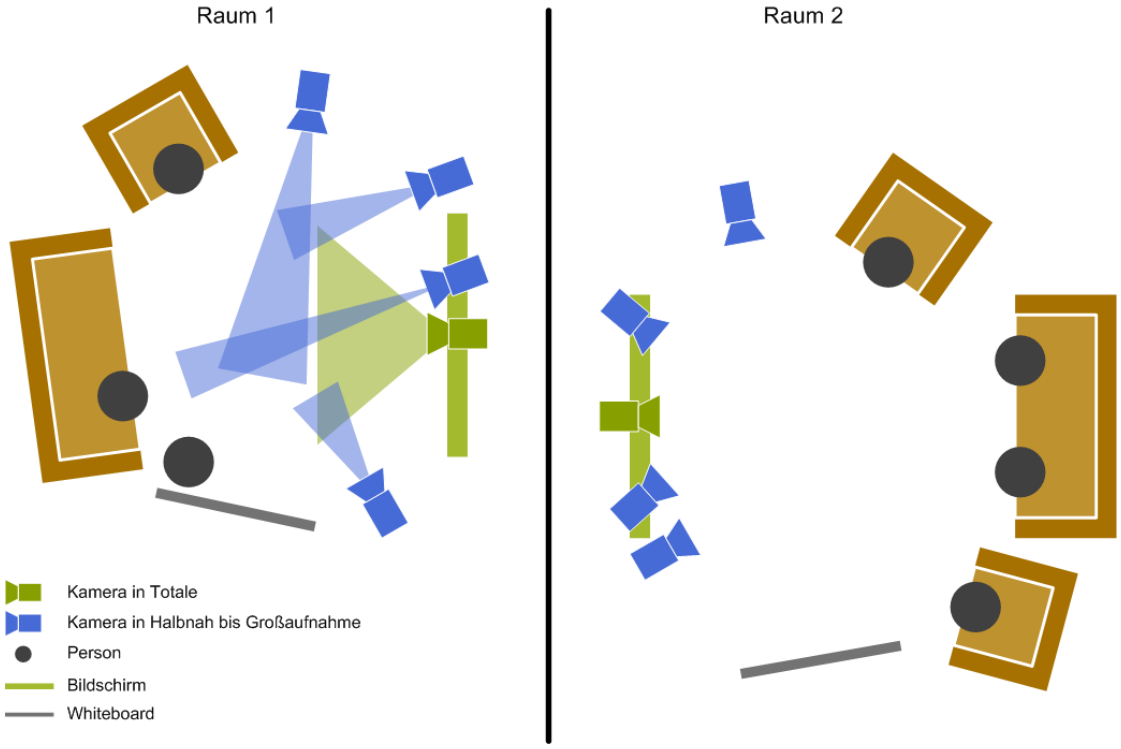

Abbildung 1: Beispiel für einen Zwei-Raum Aufbau

2 Die Orchestration Engine

Die Orchestration Engine (siehe Abbildung 2) empfängt fortlaufend Ereignisse (*Cues*), die von einem Analysemodul (siehe (Korchagin et al., 2011)) aus dem Audio- und Videostrom der Kameras und Mikrofone extrahiert werden. Das Ergebnis der Orchestration Engine, die Auswahl einer Kamera für jeweils einen Standort, wird an die Video-Communication-Engine (Jansen et al., 2011) weitergeleitet, die dann den gewünschten Videostrom der Kamera rendert. In der aktuellen Implementierung der Orchestration Engine setzen wir stark auf Regelbasierte- und Event Processing Technologie, da wir in einem früheren Ansatz basierend auf Beschreibungslogik (engl. *description logic*) Einschränkungen in der Skalierbarkeit festgestellt haben, siehe (Kaiser et al., 2010).

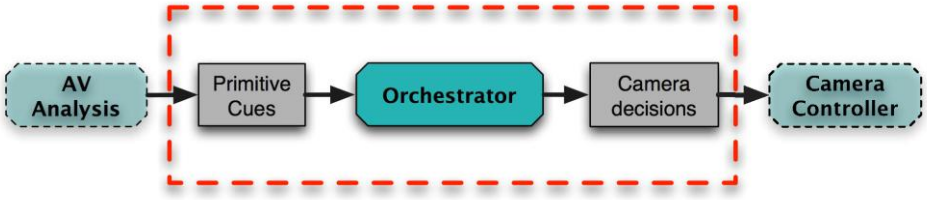

Abbildung 2: Systemübersicht der Orchestration Engine

Nachfolgend beschreiben wir, wie das gewünschte Verhalten der Orchestration Engine modelliert wurde und welche Anforderungen daraus abgeleitet wurden. Im Anschluss wird die Architektur kurz diskutiert und jedes einzelne Modul beschrieben.

2.1 Modellierung des Verhaltens

Gemeinsam mit Experten aus Filmproduktion und audiovisueller Kommunikation wurde das Konzept der „*Orchestrierung*“ entwickelt. Um das Verhalten der Orchestration Engine zu beschreiben wurden in einer Befragung der oben genannten Experten Verhaltensregeln dafür aufgestellt. Diese kann man grob in pragmatische und cinematographische Regeln einteilen.

Pragmatische Regeln stellen Aspekte der Kommunikation und Interaktion der beteiligten Akteure in den Vordergrund. Ein Beispiel dafür ist die Regel 1 in Beispiel 1. Die cinematographischen Regeln hingegen stellen sicher, dass u.a. ästhetischen Prinzipien aus der Filmproduktion eingehalten werden. Ein passendes Beispiel dazu ist die Regel 2 in Beispiel 1.

1. Wenn eine Person zu sprechen beginnt, dann zeige die Person in einer Großaufnahme von vorne.

2. Wenn eine Großaufnahme einer Person länger als X Sekunden gezeigt wurde, dann wähle eine weiter entfernte Kameraeinstellung.

Beispiel 1: Regeln in natürlicher Sprache

Wenn alle Regeln gesammelt wurden, kann man im nächsten Schritt diese zusammenfassen, gruppieren und in einer formalen Sprache präsentieren, so wie in Beispiel 2 dargestellt als pseudo Prädikatenlogik. Regel 1 aus Beispiel 2 ist die formale Darstellung der Regel 1 aus Beispiel 1. Diese Regel besagt wenn eine Person am Zug ist „*turnShift*“ und sich diese Person in einem anderen Raum befindet als der Zuseher „*isInOtherLocation*“, dann soll diese Person in einer Großaufnahme gezeigt werden „*cut2CUFront*“. Auch die Regel 2 aus Beispiel 1 wird hier in einer formalen Syntax beschrieben, siehe dazu Regel 2 in Beispiel 2.

```
1. turnShift(P) ∧ isInOtherLocation(L,P) → cut2CUFront(P)

2. (timeSinceLast(cut2Wide) > threshold) ∧ activePerson(P) ∧ isI-
nOtherLocation(P) → cut2Wide(P)

3. patternOfShortTurnTaking(P1,P2) ∧ isInOtherLocation(L,P1) ∧ is-
InSameLocation(P1,P2) → cut2Wide(P1)

4. relevancyMarker(P) ∧ isInOtherLocation(P) → cut2CUFront(P)
```

Beispiel 2: Regeln in pseudo Prädikatenlogik erster Stufe

2.2 Anforderungen

Das Ergebnis des oben beschriebenen Wissenserhebungsprozess ist ein Satz von Regeln. Damit ein Reasoningmechanismus in der Lage ist, diesen Satz von Regeln auszuführen, haben sich nachfolgende Anforderungen ergeben:

1. Die Bedingungen der Regeln zum Auswählen der Kameras basieren auf Ereignissen (z.B. „*turnShift*“ oder „*crossTalk*“) die als solche nicht vom AV Analysemodul erkannt werden. Elementare Ereignisse vom AV Analysemodul müssen deshalb auf einer höheren semantischen Ebene gebracht werden um damit Entscheidungen treffen zu können.
2. Das Verhalten zum Auswählen einer Kamera muss in einer Art und Weise formalisiert werden damit es von einer Softwarekomponente ausgeführt werden kann, aber immer noch so übersichtlich bleibt damit man es möglichst leicht feinabstimmen kann. Zusätzlich wird eine Konfliktlösungskomponente benötigt, die die Ergebnisse von gleichzeitigen – aber gegensätzlichen Entscheidungen auflöst.
3. Die Regeln von den Experten aus Film und Kommunikation orientieren sich an Kameraeinstellungen z.B. „*zeige von Person P eine Großaufnahme von vorne*“. Dabei ist zu berücksichtigen, dass sich die Position von Kameras und Personen verändert und erst die optimalste Kamera für eine Kameraeinstellung zu einer Person gefunden werden muss.
4. Die Interaktion der beteiligten Akteure hängt stark vom aktuellen Status bzw. Kontext der Konversation ab. Das heißt, dass gleiche Ereignisse unterschiedliche Ergebnisse liefern können.
5. Die verwendeten Regeln benutzen temporale Logik, deshalb wird ein Reasoningmechanismus benötigt, der alle oder zumindest einen Teil der temporalen Operatoren von Allen (Allen, 1983) implementiert.
6. Ereignisse von der AV Analyse können verzögert und außer der Reihe ankommen (engl. *out-of-order*), d.h. ein Ereignis, das früher aufgetreten ist kommt in die Orchestration Engine später an als andere Ereignisse.
7. Es gibt viele Kameras und Mikrofone, deshalb auch entsprechend viele AV Analyse Module die Ereignisse erzeugen. Diese Ereignisse von den verschiedenen Modulen müssen zusammengeführt werden.

2.3 Architektur

Die Orchestration Engine bekommt Basisereignisse als Input von den AV Analysemodulen, verarbeitet diese und sendet Anweisungen zum schalten zwischen Kameras (engl. *camera directives*) an die Video-Communication-Engine (VCE), siehe Abbildung 3. Die VCE ist eine intelligente Rendering Engine für HD-Video, siehe (Jansen et al., 2011).

Die Logik der Orchestration Engine wurde in verschiedene Module aufgeteilt um das Verhalten leichter implementieren, debuggen und verfeinern zu können. Das Cue Lifter Modul sowie der Geometric Reasoner verarbeiten Basisereignisse der AV Analyse. Der Cue Lifter bring die Basisereignisse auf eine höhere semantische Ebene um damit später Entscheidungen treffen zu können. Der Geometric Reasoner berechnet welche Kameras welche Kameraeinstellungen für welche Personen unterstützen. Die Entscheidung welche Kameraeinstellung zu verwenden ist trifft das Director Modul. Dabei werden pragmatische und ästhetische Prinzipen berücksichtigt. Am Ende der Kette sitzen das Camera Compiler und das Camera Dispatcher Modul. Diese wählen schlussendlich die Kamera für eine Person aus. Eine detaillierte Beschreibung der Architektur kann in (Falelakis et al., 2011) nachgelesen werden.

Aufgrund des Ereignisgesteuerten Verhaltens (*Event-driven Architecture*) und der Anforderung für temporales Reasoning (Anforderung Nr. 5) entschieden wir uns einen Event Processing basierten Ansatz (vgl. (Etzion & Niblett, 2010)) für den Cue Lifter und den Director zu verwenden. Dazu wurde JBoss

Drools[2] in einer internen Evaluierung ausgewählt. JBoss Drools ist ein Open Source regelbasiertes System mit Event Processing Funktionalität und erfüllt einerseits unsere Anforderungen für temporales Reasoning und andererseits ist die verwendete Regelsprache ausdruckstark genug für unsere Anforderungen.

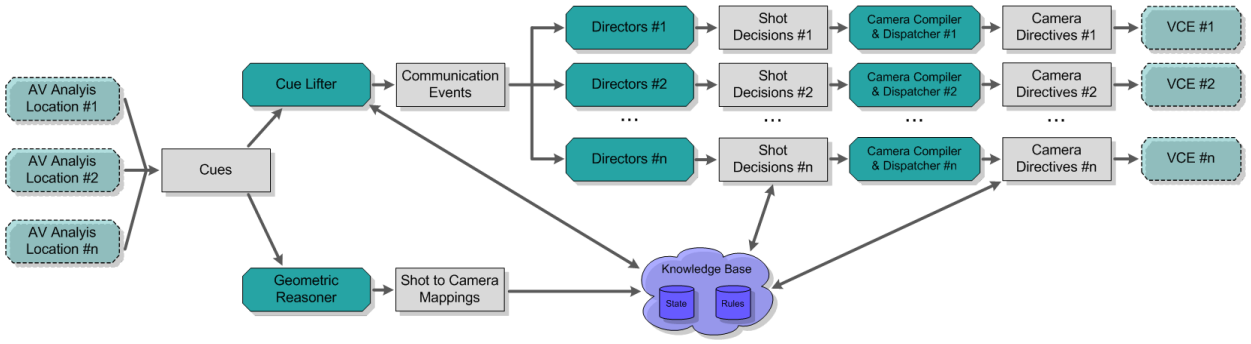

Abbildung 3: Architekturdiagramm der Orchestration Engine

2.4 Cue Lifter

Das Cue Lifter Modul vereint die Ereignisströme aus unterschiedlichen Quellen, die von verschiedenen AV Analysemodulen extrahiert werden. Hierbei kann es vorkommen, dass die Ereignisse in der falschen Reihenfolge (engl. *out-of-order*) im Cue Lifter ankommen als sie erkannt wurden (Anforderung 6 und 7). Z.B. kann es sein, dass das Ereignis der visuellen Analyse später ankommt als das Ereignis der erkannten Sprache einer Person. Um diese Aufgaben kümmert sich die Event Processing Komponente, die Ereignisse immer in ihrem zeitlichen Kontext analysiert.

Dieses Modul arbeitet mithilfe eines regelbasierten Ansatzes und sucht nach vordefinierten Muster und Veränderungen im Ereignisstrom und bildet mit Hilfe von logischen Schlussfolgerungen den aktuellen Status der gesamten Kommunikation ab d.h. „*Wer ist gerade die aktive Person*“ oder „*Sprechen gerade mehr als eine Person gleichzeitig (engl. cross-talk-situation)*“. Einige der Regeln, die zur Verarbeitung der Ereignisse angewendet werden sind in Beispiel 3 abgebildet. Regel 1 beschreibt wie eine aktive Person erkannt wird: Wird Sprache von einer Person erkannt und ist diese Person gerade nicht an der Reihe und ist diese Person auch nicht in einer „*cross-talking-situation*“ involviert, dann wird diese Person als die aktive in der Interaktion angesehen. Regel 2 beschreibt wie eine „*short-turn-taking*“ Situation erkannt wird: Wenn die Anzahl der „*turn-takings*“ (Wechsel der aktiven Person) innerhalb einer Zeitspanne einen Schwellwert übersteigt, dann wird ein short-turn-taking Ereignis gesendet. Mit diesem Verhalten des Cue Lifters wird die semantische Lücke zwischen den Basisereignissen von der AV Analyse und den höherwertigen Ereignissen, die notwendig für eine Entscheidungsfindung sind, geschlossen. Dadurch wird die Anforderung Nr. 1 erfüllt.

```
1. voiceActivity(P) ∧ ¬turn(P) ∧ ¬crosstalk(P) → turnShift(P)
2. number(turnShift()) during (t, t-X) > Y → shortTurnTaking()
```

Beispiel 3: Cue Lifter Regeln in pseudo Prädikatenlogik erster Stufe

2.5 Geometric Reasoner

Der Geometric Reasoner berechnet fortlaufend den Grad inwieweit eine Kamera eine gewünschte Kameraeinstellung für eine bestimmte Person erfüllt. Die Berechnungen, siehe auch Abbildung 4, basieren auf den Aufbau des Raumes (Position und Ausrichtung der Kameras) kombiniert mit der Position des Kopfes

[2] http://www.jboss.org/drools

einer Person. Der Geometric Reasoner erstellt fortlaufend ein Mapping von Personen zu Kameraeinstellungen inklusive dazugehöriger Wahrscheinlichkeitswerte und erfüllt die Anforderung Nr. 3.

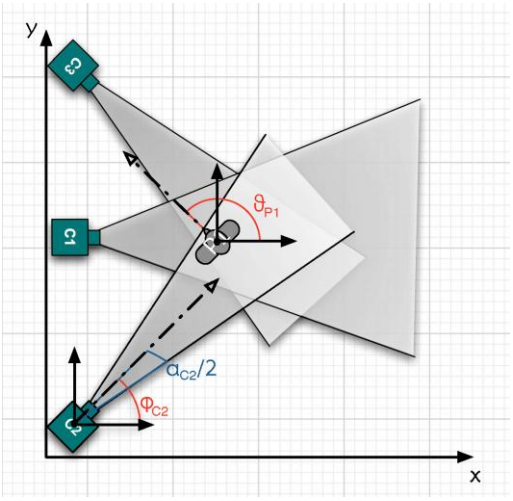

Abbildung 4:Räumlicher Aufbau und Berechnungsgrundlage für den Geometric Reasoner

2.6 Director

Die Logik zum Auswählen der Kameras ist in der Director Komponente abgebildet. Hier werden die Regeln wie in Kapitel 2.1 beschrieben ausgeführt und die Entscheidungen für die Kameraeinstellung getroffen. Die Position des Directors innerhalb der Orchestration Engine ist in Abbildung 3 dargestellt. Es gibt pro Raum eine Director Instanz die Entscheidungen über die Kameraeinstellungen für den jeweiligen Raum trifft. Die Architektur wurde an ein echtes Szenario angelehnt bei dem auch *ein* Regisseur pro Raum basierend auf die Interaktionen der Akteure die Entscheidungen trifft.

2.7 Camera Compiler und Camera Dispatcher

Die Entscheidung für eine Kameraeinstellung aus dem Director wird im Camera Compiler und Camera Dispatcher in eine konkrete Auswahl einer Kamera umgelegt. Dabei wird die gewünschte Kameraeinstellung vom Director verglichen mit den verfügbaren Kameras, die eine passende Kameraeinstellung liefern können, welche vom Geometric Reasoner erstellt wird.

2.8 Knowledgebase

Das Knowledgebase Modul erfüllt die Anforderung Nr. 4. Es bietet für alle Module Zugriff auf den Status von globalen und lokalen Eigenschaften und besitzt ein Interface mit dem es einfach möglich ist Daten zu manipulieren und abzufragen.

3 Evaluierung

Die Evaluierung des gesamten TA2 Systems hat das Ziel herauszufinden, inwiefern durch eine automatische Orchestrierung die Kommunikation entfernter Gruppen unterstützt wird. (Groen et al., 2011) konnten beweisen, dass das Konzept der „Orchestrierung“, wie es bei uns angewendet wird, Vorteile für die beteiligten Personen beim Ausführen von Aufgaben (z.B. Lösen von Aufgaben in Spielsituationen) hat. In der nachfolgenden Evaluierung beschränken wir uns auf die technischen Komponenten der Orchestration Engine die Cues verarbeiten. Es soll festgestellt werden, ob ein regelbasierter Event Processing Ansatz in der Lage ist, die zeitkritischen Anforderungen bei gewünschter Ausdrucksstärke der Logik erfüllen kann.

Hersteller von Event Processing Komponenten geben an, dass sie eine sehr hohe Anzahl von Ereignissen pro Zeiteinheit bei niedriger Verzögerung verarbeiten können[3]. Aber, eine vergleichbare Evaluierung verschiedener Implementierungen konnten wir, nach bestem Wissen und Gewissen, nicht finden. Darüber hinaus wird in einem regelbasierten System die Performance durch folgende Punkte beeinflusst: die Anzahl der Ereignisse die pro Zeiteinheit verarbeitet werden müssen, die Anzahl der Attribute eines Ereignisses, die Anzahl der Regeln, die Komplexität der Regeln, die aktuelle Anzahl der Fakten im Arbeitsspeicher, usw.

Wichtige Performanceindikatoren sind in unserem Fall die durchschnittliche Verzögerung, die maximale Verzögerung und die Standardabweichung in der Verzögerung zum Verarbeiten eines Ereignisses – also die Zeit in der ein neues Ereignis den Cue Lifter verlässt minus der Zeit an dem ein korrespondierendes Ereignis den Cue Lifter passiert. In diesem Test wurde ausschließlich die Cue Lifting Komponente getestet. Ein Test der alle Komponenten inkludiert ist nicht möglich, da temporale Operatoren in den Regeln verwendet werden, die „künstlich" Zeit für die Entscheidungsfindung hinzugeben – also abwarten.

Der verwendete Testdatensatz besteht aus 4098 Cues, die mit ca. 89 Cues pro Minute den Cue Lifter passieren. Die Logik des Cue Lifters besteht aus 12 Regeln. Im durchgeführten Test haben wurde eine durchschnittliche Verzögerung von 0,62 ms beim Erkennen eines „turn-shifts" gemessen. Die maximale Verzögerung ist 63 ms, der Median wie auch das obere Quartil < 1 ms. Es konnte auch gezeigt werden, dass 98% der Cues in weniger als 1 ms verarbeitet wurden. Der Test wurde auf einem Intel Core 2 Duo 6400 mit 2,13 GHz und 3,25 GB Ram mit Windows XP (SP 3) und Sun's Java Virtual Machine in der Version 1.6.0-26 ausgeführt.

4 Zusammenfassung und Diskussion

In dieser Arbeit präsentierten wir die „*Orchestration Engine*" ein regelbasierter virtueller Regisseur der Gruppen-zu-Gruppen Kommunikation über Audio- und Videokanäle ermöglicht. Die Orchestration Engine wählt selbstständig zwischen mehreren verfügbaren Kameras aus mehreren Räumen aus und arbeitet mithilfe eines Satzes von pragmatischen und cinematographische Regeln. Die Architektur der Implementierung wurde durch eine Reihe von Anforderungen bestimmt, welche aus dem gewünschten Verhalten abgeleitet wurden. Deshalb werden die elementaren Ereignisse aus der audiovisuellen Analyse in verschiedene Module verarbeitet. Der Cue Lifter bringt die Basisereignisse auf einer höheren semantischen Ebene um damit Entscheidungen treffen zu können. Ein Geometric Reasoner Modul berechnet fortlaufend alle möglichen Kameraeinstellungen zu Personen. Das Director Modul kann mithilfe der Ergebnisse aus dem Cue Lifter Entscheidungen für eine passende Kameraeinstellung treffen.

Eine Herausforderung der Orchestration Engine ist es Entscheidungen in Echtzeit zu treffen. Dabei muss ein Strom von Ereignissen möglichst ohne Verzögerung verarbeitet werden. Aus diesem Grund wird stark auf die Event Processing Technologie gesetzt, die es auch ermöglicht temporales Reasoning über Ereignisse durchzuführen. In einer Evaluierung konnten wir zeigen, dass das Treffen von Entscheidungen im Cue Lifter im Durchschnitt weniger als 1 ms dauert.

Ein derzeit noch ungelöstes Problem ist die Evaluierung der Orchestration Engine im Gesamtsystem. Dabei muss evaluiert werden ob ein automatisch orchestriertes System eine Verbesserung der Gruppen-zu-Gruppen Kommunikation darstellt. Das Problem dabei: jede Entscheidung der Orchestration Engine kann die Gruppe in der Kommunikation selbst beeinflussen.

Ein weiterer offener Punkt und mögliche zukünftige Erweiterung ist das Auflösen von konträren Entscheidungen, so wie in Anforderung Nr. 2 beschrieben. Dies ist in unserem Aufbau derzeit noch nicht notwendig, da wir nicht parallel Entscheidungen treffen, die man dann auflösen müsste. Auch werden derzeit die Wahrscheinlichkeiten der Ereignisse aus der AV Analyse nicht berücksichtigt. Eine Möglich-

[3] z.B. gibt Esper an 500.000 Ereignisse pro Sekunde mit einer durchschnittlichen Verzögerung von weniger als 3 Mikrosekunden verarbeiten zu können. http://docs.codehaus.org/display/ESPER/Esper+performance (Abgerufen am 3.10.2011)

keit die Orchestration Engine dahingehend zu erweitern ist es, einen Reasoningmechanismus zu verwenden der mit Wahrscheinlichkeitswerten umgehen kann.

5 Danksagung

Diese Arbeit wurde im Rahmen des Projekts TA2, *Together Anywhere, Together Anytime* erstellt. Das TA2 Projekt erhält Förderungen der Europäischen Kommission aus dem 7. Forschungsrahmenprogramm mit der Nummer 214793.

6 Literaturverzeichnis

Allen, J., (1983). *Maintaining knowledge about temporal intervals*. New York: Communications of the ACM.

Etzion, O. & Niblett, P., (2010). *Event Processing in Action*. Manning Publications.

Falelakis, M., Kaiser, R., Weiss, W. & Ursu, M., (2011). *Reasoning for Video-mediated Group Communication*. Barcelona: Proceedings of the IEEE International Conference on Multimedia Expo.

Groen, M., Falelakis, M., Michalakopoulos, S., Gasparis, E. & Ursu, M., (2011). *Improving video mediated communication with orchestration*. Journal of Computer-Mediated Communication.

Jansen, J., Cesar, P., Bulterman, D., Stevens, T., Kegel, I. & Issing, J., (2011). *Enabling Composition-Based Video-Conferencing for the Home*. IEEE Transactions on Multimedia.

Kaiser, R., Torres, P. & Höffernig, M., (2010). *The Interaction Ontology : Low-Level Cue Processing in Real-Time Group Conversations*. Florenz: Proceedings of the 2nd ACM international workshop on Events in multimedia.

Korchagin, D., Motlicek, P. & Duffner, S., (2011). *Just-in-Time Multimodal Association and Fusion from Home Entertainment*. Proceedings IEEE International Conference on Multimedia & Expo.

Williams, D., Ursu, M., Meenowa, J., Cesar, P., Kegel, I. & Bergström, K., (2011). *Video mediated social interaction between groups: System requirements and technology challenges*. Elsevier Telematics and Informatics.