Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Anwendungen 1 - WiSe 2010/2011

Dennis Kilan

Ein gestenbasiertes Freihand Modellierungswerkzeug

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung in das Themengebiet

1.1 Motivation

Bei dem Entwurf von Softwaresystemen sind verschiedene Rollen wie der Projekteigentümer, der Kunde oder der Projektleiter beteiltigt. Um die Zusammenarbeit mit allen Rollen kollaborativ zu gestalten sollte der Entwurf auf Medien erfolgen, die für alle Teilnehmer gut einsehbar sind, so dass über den Entwurf diskutiert werden kann. Heutzutage werden dabei 2 verschiedene Ansätze genutzt.

Zum einen gibt es den „Pen & Paper“ Ansatz. Die Teilnehmer zeichnen ihre Entwürfe hierbei mit einem Stift oder Kreide auf Medien wie z.B. Flipcharts oder Tafeln. Das Zeichnen auf diesen Medien ist sehr intuitiv, da der Mensch das Zeichnen bereits im Kindesalter lernt. Außerdem können mit mehreren Stiften gleichzeitig auf diesen Medien gezeichnet werden. Der „Pen & Paper“ Ansatz ist also Mehrbenutzerfähig. Dieser Ansatz besitzt jedoch mehrere Nachteile:

- Das entworfene Modell wird ab einer bestimmten Größe unübersichtlich. Es besteht keine Möglichkeit mehrere Elemente (wie z.B. Klassen) zu einem Element einer höheren Abstraktionsebene (wie z.B. Komponenten) zusammenzufassen, da man auf diesen Medien nicht auf eine höhere Abstraktionsebene herauszoomen kann.
- Entwurfselemente die einmal gezeichnet wurden, lassen sich nur schwer oder gar nicht mehr ändern oder löschen. Eine Element lässt sich nicht verschieben, ohne es neu zeichnen zu müssen. Das Löschen von Elementen kann zum Beispiel nur über das Durchstreichen erfolgen. Solche Änderungen machen den Entwurf noch unübersichtlicher.
- Ist der Entwurf des Systemes auf diesen Medien beendet, so sollte das so entstandene Modell auf digitale Medien übertragen werden, damit zum Beispiel das Modell verbreitet werden kann. Dieses Digitalisieren ist zum einem sehr zeitaufwändig und es entstehen sehr schnell Fehler, zum Beispiel durch unleserliche Beschriftung.

Der zweite Ansatz nutzt CASE Tools. Hierbei erfolgt der Entwurf direkt auf einem digitalen Medium, wie einem Desktop PC. Somit verschwindet der Zusatzaufwand des Digitalisierens. Die Entwurfselemente lassen sich auch nach ihrer Zeichnung einfach ändern und entfernen. In den meisten CASE Tools kann der Anwender auch einfach dasselbe Modell gleichzeitig in mehreren Abstraktionsebenen entwerfen. Somit kann beliebig hinein- und herausgezoomt werden. Jedoch bietet dieser Ansatz einige Nachteile, weshalb dieser Ansatz in kollaborativen Sitzungen nicht häufig benutzt wird:

- CASE Tools sind in der Regel nicht für den kollaborativen Entwurf geeignet. Das liegt vor allem an der benutzten Hardware. Diese Tools werden über Maus und Keyboard bedient, somit ist ein paralles Zeichnen nicht möglich. Der Monitor bietet nur ein eingeschränktes Sichtfeld, so dass die Anzahl der Betrachter nur gering bleiben kann.
- Die Bedienung eines CASE Tools ist nicht intuitiv. Der Anwender muss sich beispielsweise durch mehrere Menüs klicken um ein einfaches Modellelement zu zeichnen. Dieses ist gerade beim kollaborativen Arbeiten hinderlich, da solche Verzögerungen den kreativen Fluß stören können.
- Der Anwender benötigt für CASE Tools zusätzliches Fachwissen über dessen Bedienung. Er kann sich nicht mehr rein auf das Modellieren konzentrieren.
- Skizzen Elemente, wie Anmerkungen, die nicht zur Syntax der Modellierungssprache gehören, lassen sich durch das CASE Tool nur eingeschränkt erstellen. Solche Skizzen Elemente werden jedoch häufig beim Entwurf von Systemen auf Papier genutzt (zum Beispiel: „Ich markiere das Element schnell mal so“)

Ein besserer Ansatz nutzt die Vorteile beider Verfahren. Die Entwurfselemente werden wie bei den CASE Tools digital gezeichnet, jedoch erfolgt das Zeichnen durch einfache Gesten wie beim Zeichnen. Dieser hybride Ansatz kann mittels digitaler Medien wie Tabletops umgesetzt werden. Diese erlauben die Bedienung über Touch Gesten. Da Tabletops von oben betrachtet werden, ist zusätzlich der Nachteil des eingeschränkten Sichtfeldes von Monitoren verringert.

Das Ziel dieser Arbeit soll der Entwurf eines solchen hybriden Ansatzes auf Microsofts Tabletop „Surface“ sein. Surface bietet die Möglichkeit der Gestensteuerung. Der Anwender soll dabei die Elemente mit den Fingern auf der Touch - Oberfläche zeichnen und das zu entwerfende Tool soll diese Gesten erkennen. Die so erkannten Gesten sollen nach Erkennung digitalisiert werden. Im Kontext eines UML Klassendiagrammes soll das Tool zum Beispiel ein vom Anwender gezeichnetes Rechteck in eine UML Klasse umwandeln.Durch diese Digitalisierung lassen sich die Element einfach modifizieren und leicht abspeichern.
Für welche konkrete Modellierungszenarien dieses Tool eingesetzt werden soll, ist noch nicht festgelegt. Möglichkeiten wären zum Beispiel die Nutzung für UML Modellierung oder das Zeichnen von GUI Prototypen, die durch die Erkennung durch das Tool direkt ausgeführt werden könnten (Rapid Prototyping).

1.2 Aufbau dieser Ausarbeitung

In dem Hauptteil dieser Arbeit werden zunächst Gesten zum Zeichnen von Formen klassifiziert [2]. Dabei werden diese auch nach ihrer Relevanz für das zu entwickelnde Tool bewertet. Im nächstem Kapitel [3] werden der Rubine- , der SiGrid- und der SiGer Algorithmus als bereits entwickelte Algorithmen für die Erkennung von Zeichengesten auf Oberflächen vorgestellt. Der Hauptteil schließt mit einer Übersicht der erwarteten Risiken [4] ab, die bei der Nutzung eines gestenbasierten Tools auftreten können und schlägt Lösungen vor, wie mit diesen Risiken umgegangen werden kann.

Der Schlußteil fasst diese Ausarbeitung zusammen [5] und gibt einen Ausblick [6] über das mögliche Vorgehen während des Masterstudiums.

2 Sinnvolle Gesten

Vor dem Entwurf eines gestenbasierten Tools muss analysiert werden, welche Klassen von Gesten es gibt. Dieses Kapitel befasst sich außerdem damit, inwieweit die Gestenklassen für das zu entwickelnde Tool sinnvoll sind.

Single-Stroke vs. Multi-Stroke
Eine Single-Stroke Geste ist eine Geste, die in einem Streich ausgeführt werden kann. Mit Single-Stroke Gesten können beispielsweise einfache Formen wie Rechtecke, Kreise oder Linien gezeichnet werden. Multi-Stroke Gesten sind Gesten, die in mehreren Strichen ausgeführt werden. Im Kontext des Zeichnens also Gesten wo der Stift mehrfach von der Zeichenoberfläche abgehoben werden muss.
Es lassen sich sehr viele Elemente in Single-Stroke Gesten zeichnen. Jedoch ist das Zeichnen bei komplizierten Elementen, wie einem UML Actor nicht intuitiv. Weiterhin lassen sich einige Elemente nur mit Multi-Stroke Gesten zeichnen, wie zum Beispiel die gestrichelte Linie bei einem UML Implementierungskonnektor. Das zu entwerfende Tool sollte also Multi-Stroke Gesten unterstützen. Eine Möglichkeit der Umsetzung von Multi-Stroke Gesten können zeitliche Intervalle (z.B. 1 Sekunde Inaktivität) sein, nachdem die Erkennung der Geste als abgeschlossen gilt.

Singletouch vs. Multitouch
Singletouch Gesten werden durch ein Eingabegerät zur Zeit ausgeführt, zum Beispiel das Zeichnen mit einem Stift. Multitouch Gesten sind Gesten, die durch mehrere Quellen an verschiedenen Punkten gleichzeitig ausgeführt werden. Ein Beispiel hierbei ist das Vergrößern von Elementen durch das auseinander ziehen von 2 Fingern auf einer Oberfläche.

Der Haupteinsatz von Multitouch Gesten liegt in der Manipulation bereits gezeichneter Elemente. Viele dieser Gesten sind bereits für Elemente auf dem Microsoft Surface vorhanden und können somit für das Tool übernommen werden. Multitouch Gesten sind sinnvoll zur Manipulation von Modellen. Für das Zeichnen von Elementen sollen nur Singletouch Gesten genutzt werden, da das Zeichnen von Formen mit einem Stift intuitiver ist. Multitouch ist für die Zeichenphase nur sinnvoll, wenn mehrere Benutzer gleichzeitig verschiedene Elemente an verschiedenen Position zeichnen wollen.

2-Dimensionale Gesten vs. 3-Dimensionale Gesten
Gesten können noch in 2-Dimensionale Gesten und 3-Dimensionale Gesten unterschieden werden. Für eine 3-Dimensionale Geste wird zusätzlich noch die Tiefe bei der Ausführung der Geste berücksichtigt. Das Zeichnen von Formen findet auf Oberflächen statt, so dass 3-Dimensionale Gesten für das zu entwickelnde Tool nicht relevant sind.

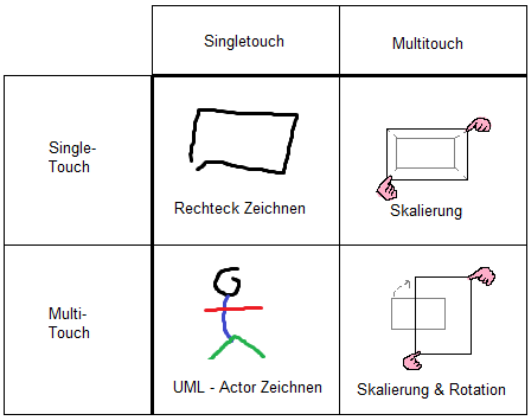

Abbildung 1: Vergleich der Gesten Klassen

3 Algorithmen zur Gestenerkennung

Es gibt bereits viele Arbeiten die sich mit der Erkennung von Formen durch Freihandgesten beschäftigt haben. Somit sind auch bereits viele Algorithmen für diese Problemstellung entwickelt worden. Dieses Kapitel betrachtet drei verschiedene Ansätze, die bereits zur Gestenerkennung auf Medien wie PDAs oder Tablet Pcs genutzt werden. Es gibt noch viele weitere Ansätze, die zum Beispiel neuronale Netze oder Hidden Markov Modelle nutzen. Diese Algorithmen sind jedoch für diese Arbeit zu komplex, somit werden diese Varianten in dieser Arbeit nicht weiter vorgestellt.

3.1 Der Rubine Algorithmus

Der Rubine Algorithmus [Rubine (1991)] ist bereits 1991 von Dean Rubine für sein GRANDMA[1] Toolkit entwickelt. Dieser Algorithmus dient als Basis vieler Algorithmen zur Gestenerkennung. Dieses Kapitel stellt den ursprüngliche Algorithmus von Rubine vor.

Der Rubine Algorithmus nutzt Feature Vektoren zur Erkennung von Gesten. Der Algorithmus merkt sich beim Zeichnen einer Geste in periodischen Intervallen die aktuelle Position des Stiftes. Aus diesen Positionen ermittelt der Algorithmus nun 13 verschiedene Features. Features sind zum Beispiel der Sinus und der Cosinus des Startwinkels der Geste, die absolute Entfernung des End- zum Startpunkt, die Länge der Diagonale der Bounding Box, die Dauer und Geschwindigkeit der Geste. Einige der Features sind in Abbildung 2 zu sehen. Die so berechneten Features werden zu einem Featurevektor zusammengefasst. Dieser Featurevektor wird vom Algorithmus mit allen bekannten Featurevektoren verglichen.

Bekannte Featurevektoren entstehen beim Rubine Algorithmus in einem Trainingsmodus. Dort wird jede Geste, die vom Tool erkannt werden soll, n-mal gezeichnet um daraus dessen gemittelte Features zu ermitteln. Diese Features können noch verschieden gewichtet werden. Der Algorithmus ist aufgrund seines Alters für Singlestroke und Single-Touch Gesten entwickelt worden. Der Algorithmus ist jedoch erweiterbar und so gibt es inzwischen Anpassungen die auch Multistroke Gesten unterstützen.

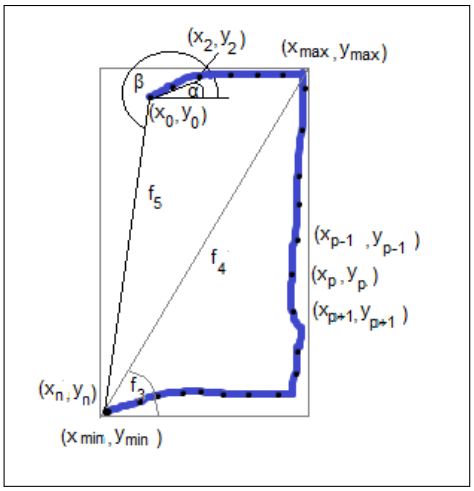

Abbildung 2: Beispielfeatures des Algorithmus von Rubine

[1] GRANDMA: Gesture Recognizers Automated in Novel Direct Manipulation Architecture

3.2 SiGrid Algorithmus

Der SiGrid Algorithmus wurde von Beat Signer, Moira Noirre und Ueli Kurrmann im Jahr 2007 für ihr iGesture Framework entwickelt. Dieser Algorithmus basiert auf Bit Signaturen in einem Grid. Das Grid besteht hierbei aus n gleich großen Quadraten, die jeweils durch einen eindeutigen Bit String identifiziert werden. Die Nachbarn unterscheiden sich dabei jeweils um genau 1 Bit, somit kann die Richtung der Geste bestimmt werden. Zum Beispiel unterscheiden sich alle Bit Strings der zweiten Reihe durch ein gesetzes zweites Bit von ihren oben liegenden Nachbarn.

Der Algorithmus merkt sich beim Zeichenvorgang die gezeichnete Form und transformiert sie auf eine einheitliche Größe. Die so gezeichnete Form wird auf das Grid gelegt und es werden alle Bit Strings des Grids ermittelt in denen die Form liegt. Alle diese Bit Strings werden zu einem Gesamt Bit String konkateniert. Dieser Bit String wird mit allen bekannten Bit Strings verglichen. Bekannte Bit Strings entstehen wie beim Rubine Algorithmus in einem Trainingsmodus. Eine Beispielsignatur ist in Abbildung 3 zu sehen. Der Vergleich mit den bekannten Bit Strings erfolgt optional über die Levensthein oder die Hamming Distanz. Die Levensthein Distanz zählt wieviele Bits invertiert werden müssen um beide Signaturen gleich zu machen. Die Hamming Distanz ist eine Generalisierung der Levensthein Distanz und kann zusätzlich Bits hinzufügen oder entfernen um den kürzesten Abstand zu berechnen.

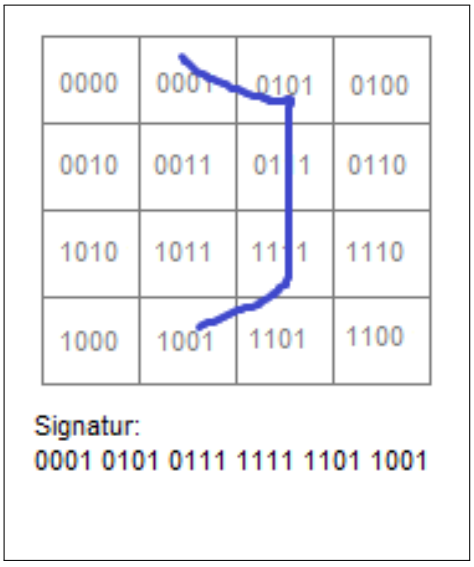

Abbildung 3: Signatur des SiGrid Algorithmus

3.3 SiGeR Algorithmus

Einen völlig anderen Ansatz geht der SiGeR (Simple Gesture Recognition) Algorithmus. Der SiGeR Algorithmus wurde 2005 von Scott Swigart für das Microsoft Tablet PC SDK entwickelt. SiGer nutzt reguläre Ausdrücke zur Gestenerkennung. Zeichnet der Anwender eine Geste, so merkt sich der Algorithmus in bestimmten Zeitintervallen die Bewegung der Geste in den 8 Richtungen: Oben, Links Oben, Links, Links Unten, Unten, Rechts Unten, Rechts und Rechts oben. Außerdem merkt sich der SiGeR Algorithmus verschiedene statistische Information wie die Geschwindigkeit der Geste oder die Entfernung des Start- und Endpunktes. Der so entstandene Vektor wird mit allen bekannten regulären Ausdrücken verglichen, um die Geste zu erkennen. Bekannte reguläre Ausdrücke werden hier jedoch nicht in einem Trainingsmodus ermittelt, sondern werden programmiert.

Eine schließende eckige Klammer kann zum Beispiel folgendermaßen beschrieben werden: R R U U U U L L (R=Rechts, U=Unten, L=Links),sie hat 4 Stop Points und End- und Startpunkt sind weit auseinander (siehe 4). Stop Points sind dabei Punkte, an welchen die Geschwindigkeit der Geste stark abnimmt, wie es zum Beispiel beim Zeichnen einer Ecke der Fall ist. Somit kann eine eckige Klammer von einer runden Klammer unterschieden werden. Zeichnet der Anwender nun diese Klammer, zeichnet SiGer die Bewegung auf und vergleicht sie zum Beispiel mit folgendem regulären Ausdruck (in C#-Sharp):

Listing 1: Beispiel eines SiGeR Recognizer

```
protected override bool Recognize()
{
    return (
        StrokeInfo.StrokeStatistics.StartEndProximity >
        CLOSED_PROXIMITY
        && StrokeInfo.StrokeStatistics.StopPoints == 4
        && StrokeInfo.StrokeStatistics.Square > 0.9
        && StrokeInfo.IsMatch(Vectors.StartTick + Vectors.Rights +
        Vectors.Downs + Vectors.Lefts + Vectors.EndTick,
        0, false, true));
}
```

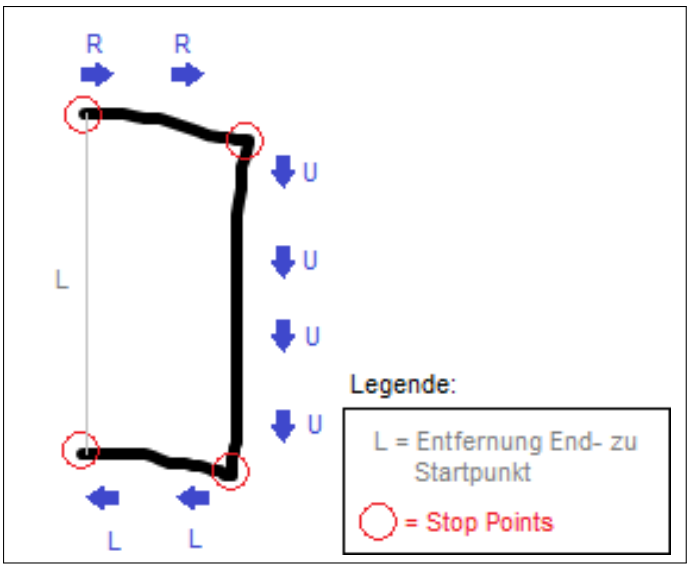

Abbildung 4: Gestenerkennung mit SiGer

4 Risiken

Eine Software die Gesten erkennt birgt einige Risiken. Diese Risiken sollten vor dem Entwurf der Software geklärt werden, um diesen nach Möglichkeit vorzubeugen oder um zu ermitteln wie mit diesen Risiken umzugehen ist. Dieses Kapitel gibt einen Überblick über die erwarteten Risiken und Schwierigkeiten, die auftreten können. Außerdem werden mögliche Lösungsvorschläge vorgestellt.

- **Missinterpretation einer Geste**
  Der Anwender kann eine Geste sehr ungenau zeichnen, was besonders bei sehr ähnlichen Formen zu Problemen führen kann. So kann eine eckige Klammer als eine runde Klammer erkannt werden, wenn der Anwender die Ecken zu rund zeichnet. Das Tool kann den Anwender jedoch nicht dazu zwingen sauber zu zeichnen, da darunter die Usability leiden würde. Somit ist die Eintrittswahrscheinlichkeit dieses Risiko hoch, wenn keine Gegenmaßnahmen ergriffen werden.
  Eine erste Maßnahme die Eintrittswahrscheinlichkeit dieses Risikos zu senken liegt darin, dass beim Entwurf der Gesten darauf geachtet wird, dass sich Gesten nicht zu ähnlich sind. Dieses ist jedoch in einigen Modellierungssprachen nicht immer möglich, wenn es dort sehr ähnlich aussehende Elemente gibt. In diesem Fall kann mit dem Risiko umgegangen werden, indem das Tool den Anwender explizit, z.B. über ein Pop-Up, fragt, eine Auswahl aller möglichen Elemente anzeigt, die der Geste mit einem bestimmten Schwellwert entsprechen könnten.

Hat zum Beispiel die vom Anwender gezeichnete Form 90% Übereinstimmung mit Element A und 91% Übereinstimmung mit Element B, so sollte das Tool den Anwender fragen, ob er Element A oder Element B meinte. Eine Interpretation des Tools zu Element B, da es eine höhere Übereinstimmung hat, ist nicht sinnvoll, da diese 1% Abweichung durch unsauberes Zeichnen verursacht sein kann.

- **Nichterkennen einer Geste**
  Das zweite Risiko ist dem ersten Risiko ähnlich. Hierbei weicht die vom Anwender gezeichnete Geste von allen bekannten Gesten zu stark ab. Hat zum Beispiel die vom Anwender ausgeführte Geste nur maximal 20% Übereinstimmung mit der bekannten Geste C, so sollte das Tool die Geste nicht als Geste C interpretieren und in Element C umwandeln. Das würde der Anwender als merkwürdiges Verhalten des Tools interpretieren. Außerdem neigen Anwender beim Entwerfen eines Modells dazu, auch Elemente zu zeichnen die nicht zu der Syntax der Modellierungssprache gehören, wie zum Beispiel kurze Anmerkungen oder Notizen (im folgenden Skizzen Elemente genannt). Dadurch ist die Eintrittswahrscheinlichkeit dieses Risikos als sehr hoch einzuschätzen.
  Das Tool sollte also gezeichnete Gesten nur dann als eine bekannte Geste interpretieren, wenn sie eine Mindestübereinstimmung mit mindestens einer bekannten Geste besitzt. Liegt die Übereinstimmung unter diesem Schwellwert, so kann das Tool diese ohne Umwandlung darstellen. Alternativ kann das Tool sehr restriktiv sein, und diese Gesten ignorieren. Dadurch kann die Einhaltung der genutzen Modellierunssprache eingehalten werden, zum Preis der Anwenderfreundlichkeit.

- **Unintuitive Gesten**
  Es gibt sehr viele verschiedene Arten eine bestimmte Form zu zeichnen, so kann zum Beispiel ein Rechteck in einem Strich gezeichnet werden oder jede Seite kann als einzelner Strich gezeichnet werden. Während Person A die „Eine Strich“ Variante als intuitiv betrachtet, kann Person B diese Art das Rechteck zu zeichnen als unintuitiv empfinden. Jedoch werden viele Formen von den meisten Menschen auf diesselbe Weise gezeichnet. Die Eintrittswahrscheinlichkeite dieses Risikos ist somit nicht sehr hoch, wenn beim Entwurf der Gesten darauf geachtet wird, das dieses für die meisten Menschen intuitiv ist.
  Optimalerweise müsste also vor dem Entwurf der Gesten eine Studie durchgeführt werden, in der ermittelt wird auf welche Art und Weise eine Form am häufigsten genutzt wird. Dieses ist jedoch im Rahmen des Studiums zu aufwändig. Also werden die Gesten zunächst nach eigenem Empfinden der Intuitivität entworfen. Eine Evaluiering der Intuitivität kann in späteren Semestern durch Usability Tests erfolgen.

- **Handschrifterkennung**
  Das Erkennen von Schriften ist weitaus komplexer als das Erkennen von einfachen Modellelementen. Jeder Mensch hat eine einzigartige Handschrift und viele Buchstaben sind sehr ähnlich. Gerade die Ähnlichkeit der Buchstaben ist dabei problematisch, da die Handschrift eines Menschens sehr unsauber sein kann. Beispielsweise sieht inn vielen Handschriften das kleine „u“ aus wie ein kleines „n“, welches sogar der lesende Mensch manchmal nicht unterscheiden kann. Das Eintrittsrisiko dieser Geste ist sehr hoch.
  Es gibt bereits viele Algorithmen die eine Handschrifterkennung ermöglichen. Diese Algorithmen sind sehr komplex und können in ihrer Implementierung viel Zeit in Anspruch nehmen. Somit ist Handschriftenerkennung vorläufig noch nicht vorgesehen. Eine Möglichkeit zur Texteingabe kann durch eine Bildschirmtastatur erfolgen, die über eine bestimmte Geste geöffnet werden kann.

5 Zusammenfassung

In Entwurfsitzungen entwerfen viele verschiedene Rollen kollaborativ Modelle auf kollaborativen Medien. Jedoch haben sowohl analoge Medien wie Papier oder Tafeln und digitale Medien wie CASE Tools einige Nachteile. Deshalb ist das Ziel dieser Arbeit eine hybride Lösung, die die Vorteile beider Modellierungsansätze vereint. Es soll ein Tool entwickelt werden, welches das Zeichnen per Gesten auf der Touch Oberfläche von Microsofts Surface Tabletop ermöglicht.

Dafür müssen zunächst die Gesten zum Zeichnen klassifiziert werden und ihre Relevanz für das Tool bewertet werden. Dabei wurde festgestellt, dass das Tool Multi-Stroke Gesten unterstützen sollte, da einige Formen in mehreren Strichen gezeichnet werden müssen. Außerdem sind Multi Touch Gesten zur Manipulation bereits gezeichneter Elemente sinnvoll, es sollen aber nach Möglichkeit die bereits vorhandenen Gesten vom Surface genutzt werden.

Es gibt bereits viele Algorithmen die das Erkennen von Gesten auf Oberflächen ermöglichen. Einige basieren auf bestimmten Features der Geste, wie der Rubine Algorithmus. Andere nutzen gridbasierte Signaturen, wie der SiGrid Algorithmus. Und es gibt auch noch Algorithmen die Gesten durch reguläre Ausdrücke beschreiben, wie den SiGer Algorithmus. Feature- und Gridbasierte Algorithmen müssen vor der Nutzen in einem Trainingsmodus erlernt werden, während die anderen hart einprogrammiert werden.

Die Gestenerkennung birgt einige Risiken, die hauptsächlich auf unsauberes Zeichnen zurückzuführen sind. Somit muss das Tool mit diesen Risiken umgehen können. Dazu sollten sich verschiedene Gesten nicht zu stark ähneln und falls es unvermeidlich ist, der Anwender durch ein Menü eine Auswahl der Gesten angezeigt bekommen. Gesten die keiner bekannten Geste ähnlich genug sind, sollten als Skizzenelemente erkannt werden und nicht in eine bekannte Form umgewandelt werden. Außerdem soll das Tool keine Handschriftenerkennung ermöglichen, da diese sehr komplex und fehleranfällig ist. Texteingaben sollen durch eine Bildschirmtastatur erfolgen.

6 Ausblick

- **2. Semester**
  Ziel des zweiten Semesters ist die Entwicklung eines Prototypen des Tools für die Gestenerkennung. Dieser Prototyp soll nur einfache geometrische Formen erfolgreich erkennen können. Dadurch soll festgestellt werden, welcher Algorithmus sich am besten für die Problemstellung einsetzen lässt, ohne dabei auf spezielle Problematiken einer bestimmten Modellierungssprache achten zu müssen. Somit ist das Tool außerdem zunächts unabhängig von der konkreten Modellierungssprache.

- **3. Semester**
  Im 3. Semester soll der Prototyp für eine spezielles Anwendungsgebiet weiterentwickelt werden. Außerdem gehört hierzu auch der Test des Tools. Ziel des dritten Semesters ist somit ein geteteste und funktionsfähige Version des Tools für ein spezielles Gebiet.

- **4. Semester**
  Im vierten Semester soll die Masterarbeit verfasst werden. Große Änderungen des Tools sind dabei nicht mehr vorgesehen.

Literatur

[Chen u. a. 2008] CHEN, Qi ; GRUNDY, John ; HOSKING, John: SUMLOW: early design-stage sketching of UML diagrams on an E-whiteboard. In: *Softw. Pract. Exper.* 38 (2008), July, S. 961–994. – URL http://portal.acm.org/citation.cfm?id=1386348.1386351. – ISSN 0038-0644

[Damm u. a. 2000] DAMM, Christian H. ; HANSEN, Klaus M. ; THOMSEN, Michael: Tool support for cooperative object-oriented design: gesture based modelling on an electronic whiteboard. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 2000 (CHI '00), S. 518–525. – URL http://doi.acm.org/10.1145/332040.332488. – ISBN 1-58113-216-6

[Frisch u. a. 2009] FRISCH, Mathias ; HEYDEKORN, Jens ; DACHSELT, Raimund: Investigating multi-touch and pen gestures for diagram editing on interactive surfaces. In: *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. New York, NY, USA : ACM, 2009 (ITS '09), S. 149–156. – URL http://doi.acm.org/10.1145/1731903.1731933. – ISBN 978-1-60558-733-2

[Morris u. a. 2006] MORRIS, Meredith R. ; HUANG, Anqi ; PAEPCKE, Andreas ; WINOGRAD, Terry: Cooperative gestures: multi-user gestural interactions for co-located groupware. In: *Proceedings of the SIGCHI conference on Human Factors in computing systems*. New York, NY, USA : ACM, 2006 (CHI '06), S. 1201–1210. – URL http://doi.acm.org/10.1145/1124772.1124952. – ISBN 1-59593-372-7

[Rubine 1991] RUBINE, Dean: Specifying Gestures by Example. In: *Computer Graphics* 25 (1991), Nr. 4, S. 329–337. – ISSN 0097-8930

[Signer u. a. 2007] SIGNER, Beat ; NOIRRE, Moira C. ; KURMANN, Ueli: iGesture: A Java Framework for the Development and Deployment of Stroke-Based Online Gesture Recognition Algorithms / ETH Zuerich. URL http://www.inf.ethz.ch/personal/signer/publications/2007-snk-tr561.pdf, 2007. – Report

[Swigart 2005] SWIGART, Scott: *Easily Write Custom Gesture Recognizers for Your Tablet PC Applications*. Webseite. 2005. – URL http://msdn.microsoft.com/en-us/library/aa480673.aspx. – Letzter Aufruf am 11. Januar 2011

[Wu und Balakrishnan 2003] WU, Mike ; BALAKRISHNAN, Ravin: Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In: *Proceedings of the 16th annual ACM symposium on User interface software and technology*. New York, NY, USA : ACM, 2003 (UIST '03), S. 193–202. – URL http://doi.acm.org/10.1145/964696.964718. – ISBN 1-58113-636-6

[Wu u. a. 2006] WU, Mike ; SHEN, Chia ; RYALL, Kathy ; FORLINES, Clifton ; BALAKRISHNAN, Ravin: Gesture Registration, Relaxation, and Reuse for Multi-Point Direct-Touch Surfaces. In: *Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*. Washington, DC, USA : IEEE Computer Society, 2006, S. 185–192. – URL http://portal.acm.org/citation.cfm?id=1109723.1110635. – ISBN 0-7695-2494-X

Glossar

Bounding Box Eine Bounding Box ist ein Rechteck, welches eine gesamte Zeichnung wie ein Rahmen umschließt

CASE Tool CASE (Computer Aided Software Engineering) Tools unterstützen Software Ingenieure bei der Entwicklung von Software Systemen

Feature Vektor Eine Zusammenfassung mehrerer Merkmale (=Features) in einem Vektor

Kollaboratives Arbeiten Gemeinsames Arbeiten um eines oder mehrere gemeinsame Ziele zu erreichen

UML UML ist eine Modellierungssprache um Software Systeme als Modelle darstellen zu können