Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Anwendungen 2 - SoSe 2009

Hauke Wittern

Sketch Recognition

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Kurzzusammenfassung

Diese Arbeit fast den aktuellen Forschungsstand über Sketch Recognition Systeme zusammen. Anschließend werden die Forschungsergebnisse von den Zielen bei der Entwicklung eines eigenen Sketch Recognition Systems abgegrenzt. Das eigene System soll bei der Softwareentwicklung mit Multitouch-Tischen eingesetzt werden.

Abstract

This paper summarizes the results of the research on sketch recognition systems. Finally the results are differentiated from the own development of a sketch recognition system. The own will support software development through sketch-enabled multitouch desks.

Inhaltsverzeichnis

1 Einleitung

Das Zeichnen von Skizzen ist eine allgegenwärtige Tätigkeit, die nicht aus dem Alltag wegzudenken ist. Nahezu jeder lernt das Zeichnen mit Stift und Papier oder auf Tafeln bereits im Kleinkindalter. Deshalb wird das Zeichnen häufig als eine natürliche Tätigkeit bezeichnet [Cherubini u. a. (2007), Davis (2007)]. Skizzen ermöglichen es, Gedanken auszudrücken und zu kommunizieren. Mit Skizzen kann man Probleme analysieren und Ideen für Lösungen entwickeln.

Auch im heutigen Computerzeitalter werden Skizzen auf Papier oder Tafeln häufig verwendet, obwohl es Computerprogramme gibt, mit denen man Diagramme erstellen kann. Häufig werden Papier oder Tafeln bewusst Computerprogrammen vorgezogen [Cherubini u. a. (2007)]. Der Grund dafür ist, dass heutige Computerprogramme es nicht erlauben, mit Maus und Tastatur Skizzen auf so natürliche Weise anzufertigen wie mit Stift und Papier [Cherubini u. a. (2007), Davis (2007), Brieler und Minas (2008)]. Herkömmliche Freihandzeichnungen sind einfach anzufertigen und schränken den Inhalt der Zeichnungen nicht ein. Sie können beliebig annotiert werden und es können unterschiedliche Notationen gemischt werden. Dies erlaubt es, komplexe Zusammenhänge verständlich auszudrücken. Dennoch haben elektronische Zeichnungen einige Vorteile gegenüber herkömmlichen Zeichnungen. Am Computer kann man Zeichnungen auf einfache Weise editieren, kopieren, speichern, öffnen, versionieren und durchsuchen [Brieler und Minas (2008)]. In der Software Entwicklung werden Diagramme darüber hinaus zum Generieren von Programmcode verwendet.

Um die Vorteile des Zeichnens mit Stift und Papier mit den Vorteilen der elektronischen Datenverarbeitung zu vereinen, wurde in den letzten Jahren untersucht, wie Freihandzeichnungen am Computer (PC, elektronisches Whiteboard, Tablet PC o. Ä.) erstellt und deren Bedeutung automatisiert erkannt werden können. Das automatisierte Erkennen von Freihandzeichnungen durch Computer bezeichnet man als Sketch Recognition (engl. Skizzen Erkennung) [Hammond u. a. (2008)]. Ein zentrales Problem beim Sketch Recognition ist der Zielkonflikt zwischen dem Freiheitsgrad beim Zeichnen und der zuverlässigen Erkennung.

1.1 Eigene Arbeit und Vision

An der Hochschule für Angewandte Wissenschaften Hamburg (HAW) wird aktuell untersucht, wie das Software-Engineering von dem Einsatz von Multitouch-Tischen profitieren kann. Wie Freihandzeichnungen bzw. Skizzen auf Multitouch-Tischen dazu beitragen können, ist ein Teilgebiet der Untersuchungen. Als Hardware wird bei den Untersuchungen exemplarisch ein Microsoft Surface [Microsoft (2009)] verwendet.

Multitouch-Hardware erlaubt mehrere gleichzeitige Touch-Eingaben und ermöglicht damit kollaboratives Arbeiten mehrerer Benutzer. Zusätzlich zur Eingabe mit Fingern können Multitouch-Tische wie das Microsoft Surface physische Gegenstände erkennen, die auf den Tisch gestellt werden. Dies bietet zusätzliche Interaktionsmöglichkeiten, die nur bei Hardware mit horizontaler Display-Fläche sind. Über das Arbeiten mit solchen Multitouch-Tischen gibt es bisher nur wenig Forschungsergebnisse. Insbesondere gab es noch keine Erkenntnisse über das Entwickeln von

Software mit Multitouch-Tischen. Die Fragestellungen der Untersuchungen an der HAW sind deshalb zurzeit, wann es bei der Entwicklung von Software sinnvoll ist, Multitouch-Tische einzusetzen und wie die Benutzungsmodelle dabei aussehen. Eine Einsatzmöglichkeit von Multitouch-Tischen könnte das kollaborative Skizzieren von Modellen eines Softwaresystems sein.

Der Einsatz von Sketch Recognition verspricht, die Vorteile von per Hand skizzierte Modellen und am Computer erstellten Modellen zu vereinen. Es wird deshalb an der HAW ein Sketch Recognition System für die Modellierung von Software mit Multitouch-Tischen entwickelt.

Diese Arbeit stellt die bereits vorhandenen verwandten Forschungsergebnisse aus dem Bereich des Sketch Recognition vor und bewertet anschließend, inwieweit diese für die Entwicklung des eigenen Systems relevant sind.

2 Vorstellung verwandter Forschungsergebnisse

Ein erster Schritt in die Richtung der heutigen Sketch Recognition Systeme war das 1963 von Ivan Sutherland vorgestellte System namens SketchPad [Sutherland (1963)]. SketchPad war das erste System, mit dem elektronische Freihandzeichnungen erstellt werden können. SketchPad ist allerdings kein Sketch Recognition System bezeichnen, da es die Zeichnungen nicht selbst erkennen kann, sondern auf die Eingabe des Benutzers angewiesen ist. Erst seit den 1990er Jahren wird versucht elektronische Freihandzeichnungen mit Computern automatisch zu erkennen. Grundlage dafür war ein Algorithmus von Rubine (1991), der es erlaubt einzelne Striche zu klassifizieren. Seit Mitte der 1990er dem Jahre wurden schließlich zahlreiche Systeme entwickelt, die Freihandzeichnungen automatisiert erkennen können. In dieser Zeit wurden die Grundlagen heutiger Sketch Recognition Systeme erforscht. Insbesondere die Forschungen von Christine Alvarado, Randall Davis und Tracy Hammond haben grundlegende Bedeutung. Alvarado, Davis und Hammond bildeten zunächst eine Forschungsgruppe am Massachusetts Institute of Technology (MIT) und sind nun an unterschiedlichen Hochschulen in den USA weiterhin in der Forschung aktiv. Neuere aktive Forschungsgruppen sind unter anderen an der University of Auckland (Beryl Plimmer u. a.), an der Bundeswehr Universität in München (Florian Brieler und Mark Minas) und an der University of California (Nicolas Mangano u. a.) zu finden. Dieses Kapitel fast im Folgenden den aktuellen Stand der Forschung zusammen.

2.1 Klassifikation von Sketch Recognition Engines

Die Untersuchungen von Sketch Recognition Systemen haben einige charakteristische Merkmale aufgezeigt, mit denen man diese Systeme klassifizieren kann. Man kann Sketch Recognition Systeme darin unterschieden, was sie erkennen können, wie die Syntax von Zeichnungen beschrieben wird, wie die Erkennung stattfindet und wann die Erkennung stattfindet. Im Folgenden werden diese Unterschiede kurz erläutert.

2.1.1 Vorgegebene vs. konfigurierbare Domänen

Die Domänen der Zeichnungen, die Sketch Recognition Systeme erkennen können, sind entweder fest vorgegeben oder konfigurierbar.

Die ersten Sketch Recognition Systeme konnten Freihandzeichnungen nur aus vorgegebenen Domänen erkennen. Dem Erkennungs-Algorithmus ist bei solchen Systemen die konkrete Syntax der Domäne bekannt. Aufgrund dieser engen Kopplung kann die zu erkennende Domäne nicht geändert werden. Tahuti [Hammond und Davis (2002)] und SUMLOW [Chen u. a. (2003)] sind solche Systeme, die auf die Erkennung von bestimmten UML-Diagrammen beschränkt sind.

Bei neueren Sketch Recognition Systemen wie SketchRead [Alvarado und Davis (2004)], InkKit [Plimmer und Freeman (2007)] oder DSketch [Brieler und Minas (2008)] ist die Syntax der Zeichnungen hingegen nicht fest vorgegeben. Der Systemadministrator kann die konkrete Syntax der Domänen beschreiben und das System konfigurieren diese Domänen zu erkennen. Der Vorteil solcher Systeme ist ihre Wiederverwendbarkeit.

2.1.2 Inkrementelle vs. vollständige Erkennung

Die Erkennung einer Zeichnung kann entweder schrittweise während des Zeichnens oder vollständig nach Abschluss des Zeichnens erfolgen.

Die Erkennung während des Zeichnens bezeichnet man als inkrementell [Liao u. a. (2005)] bzw. kontinuierlich. Jedesmal wenn der Benutzer einen Strich gezeichnet hat, führt das System den Erkennungs-Algorithmus aus. Diese Art von Erkennung wurde u. a. bei Tahuti [Hammond und Davis (2002)], SketchRead [Alvarado und Davis (2004)] und von Brieler und Minas (2008) verwendet.

Ein konträrer Ansatz ist die vollständige Erkennung einer Zeichnung. Der Zeitpunkt, an dem die Erkennung durchgeführt wird, ist hierbei immer nach dem Abschluss des Zeichnens. Wann genau dieser Zeitpunkt erreicht ist, kann entweder das System oder der Benutzer entscheiden. Bei SUMLOW [Chen u. a. (2003)] nimmt das System an, dass nach einer bestimmten Zeitdauer ohne Benutzereingaben die Zeichenaktivität abgeschlossen ist. InkKit erlaubt es dem Benutzer, durch einen Klick auf einen Button dem System mitzuteilen, dass die Zeichnung erkannt werden soll [Plimmer und Freeman (2007)].

Die bisherigen Untersuchungen haben die beiden vorigen Ansätze kontrovers diskutiert und lassen keinen allgemeingültigen Schluss zu, welcher Ansatz der bessere ist. Plimmer und Freeman (2007) ziehen die vollständige Erkennung vor, weil sie einfacher zu Implementieren ist. Die inkrementelle Erkennung ist komplexer als die vollständige, weil zum Zeitpunkt der Erkennung eventuell noch nicht alle Informationen vorhanden sind [Brieler und Minas (2008)]. Bei der inkrementellen Erkennung ist außerdem die Zeitdauer der Erkennung ein Problem. Die Erkennung von Zeichnungen ist sehr rechenintensiv und damit sehr zeitaufwendig. Deshalb besteht die Gefahr, dass die wiederholt durchgeführte Erkennung den Zeichenvorgang der Benutzer unterbricht [Brieler und Minas (2008), Li u. a. (2005)]. Andererseits erlaubt die inkrementelle Erkennung auf natürlichere Weise zu Zeichnen als bei vollständiger Erkennung. Zeichnungen, die auf herkömmliche Weise mit Stift und Papier gezeichnet werden, kann ein Betrachter unmittelbar erkennen. Systeme, die den

Ansatz der vollständigen Erkennung umsetzen, erfordern vom Benutzer eine zusätzliche Aktivität, damit der Erkennungsvorgang ausgelöst wird (s. o.). Brieler und Minas (2008) betrachten dies als eine Behinderung des Benutzers, welche die Vorteile elektronischer Freihandzeichnungen reduziert. Plimmer und Freeman (2007) nehmen diese Einschränkung wiederum in kauf. Sie gehen davon aus, dass der Erkennungsvorgang nur selten ausgelöst werden muss. Dies sei in der Regel der Fall, wenn eine Zeichnung fertiggestellt ist.

2.1.3 Konfigurationssprache vs. Training mit Beispielen

Ein Sketch Recognition System muss wissen, wie die konkrete Syntax der Domäne einer Zeichnung aussieht. Dies kann man dem System entweder mit Beispielen oder mit einer speziellen Beschreibungssprache für die Syntax mitteilen.

InkKit [Plimmer und Freeman (2007)] ist ein Vertreter von mit Beispielen trainierbaren Systemen. Bei InkKit zeichnet ein Entwickler des Systems mit der Hand mindestens ein separates Beispiel für jedes Element der Domäne und ordnet die Beispiele den Elementen der abstrakten Syntax zu.

Der andere Ansatz ist eine Beschreibungssprache, mit der ein Entwickler die Eigenschaften der Syntax einer Domäne definieren kann. Die bisherigen Implementierungen dieses Ansatzes verwenden Sprachen, die es erlauben die Syntax hierarchisch zu definieren [Hammond und Davis (2007), Costagliola u. a. (2005), Brieler und Minas (2008)]. Mit diesen Sprachen werden primitive Elemente, wie z. B. Striche und Bögen, zu komplexeren Elementen (z. B. Rechte) zusammengesetzt. Die zusammengesetzten Elemente können ebenfalls wiederverwendet werden, um auf einfache Weise noch komplexere Elemente zu beschreiben (z. B. UML-Klassen). Diese Sprachen sind üblicherweise textuelle Sprachen [Hammond und Davis (2007), Costagliola u. a. (2005), Brieler und Minas (2008)]. Mit der Sprache LADDER [Hammond und Davis (2007)] können die Domänen Elemente zusätzlich mit einem grafischen Editor gezeichnet werden, was die Komplexität der Domänenbeschreibung reduziert.

Das Training mit Beispielen ist zwar sehr einfach [Plimmer und Freeman (2007)], hat aber einen gegenüber einer Beschreibungssprache den Nachteil, die selben Informationen mehrfach angeben zu müssen. Beim Training mit Beispielen werden in der Regel mehrere Beispiele für jedes zu erkennende Domänenelement benötigt. Falls das System die exakte Darstellung benötigt, muss diese separat definiert werden. Beides verletzt das DRY-Prinzip (Don´t Repeat Yourself) und schränkt die Evolvierbarkeit des Systems ein.

2.1.4 Gestenbasierte vs. sichtbasierte vs. geometriebasierte Erkennung

Bis heute sind drei verschiedene Ansätze für Erkennungsalgorithmen entwickelt worden. Der älteste Ansatz ist die gestenbasierte Erkennung. Gestenbasierte Erkennungsalgorithmen erkennen die Inhalte einer Zeichnung daran, wie sie gezeichnet wurden. Häufig wird dazu der Algorithmus von Rubine (1991) verwendet, der die Pfade einzelner Striche klassifiziert, indem er sie mit Vorlagen vergleicht. Eine solche gestenbasierte Erkennung wurde z. B. von Chen u. a. (2003) verwendet.

Sicht basierte (engl. vision based) Algorithmen vergleichen die Zeichnungen ebenfalls mit Vorlagen. Im Gegensatz zum gestenbasierten Ansatz erkennt der Algorithmus die Inhalte einer Zeichnung daran, wie sie aussehen und nicht daran, wie sie der Benutzer gezeichnet hat. Dieser Ansatz wurde InkKit System von Plimmer und Freeman (2007) verwendet. Plimmer und Freeman (2007) haben dabei den Algorithmus von Rubine (1991) so erweitert, dass er auch eine Kombination von Strichen klassifizieren kann.

Der dritte Ansatz ist die geometriebasierte Erkennung. Ein geometriebasierter Erkennungsalgorithmus zerteilt eine Zeichnung anhand der geometrischen Eigenschaften in Primitive (wie z. B. Ellipse, Rechteck o. Ä.). Dieser Ansatz wurde zuerst mit Tahuti [Hammond und Davis (2002)] präsentiert. Alvarado und Davis (2004) haben die geometriebasierte Erkennung mit SketchRead weiterverfolgt und mit dynamisch erzeugten Bayes'schen Netzwerken umgesetzt. DSketch [Brieler und Minas (2008)] ist ein weiteres geometriebasiertes System, welches derzeit noch weiterentwickelte wird.

Für eine geometriebasierte Erkennung mit Bayes'schen Netzwerken sprechen verschiedene Argumente. Erstens ist ein solcher Ansatz am geeignetsten um Mehrdeutigkeiten effizient und robust zu lösen [Liao u. a. (2005)]. Zweitens harmoniert die geometriebasierte Erkennung besonders gut mit einer Beschreibungssprache für die zu erkennenden Domänen. Eine Beschreibungssprache kann die geometrischen Eigenschaften der Domänenelemente explizit ausdrücken. Der gestenbasierte Ansatz ist kann nur einzelne Striche klassifizieren und hat deshalb Defizite bei der Benutzbarkeit.

2.1.5 Zusammenfassung

Die vorigen Abschnitte haben vier charakteristische Merkmale der heutigen Sketch Recognition Engines vorgestellt. Alle diese Merkmale sind weitestgehend voneinander unabhängig und können in beliebigen Kombinationen verwendet werden. Am Anfang der Entwicklung einer neuen Sketch Recognition Engine muss man sich dafür entscheiden, zu welchen der hier vorgestellten Klassen das System gehören soll.

2.2 Lösen von Mehrdeutigkeiten

Mehrere Untersuchungen haben Mehrdeutigkeiten in Skizzen als ein zentrales Problem beim Sketch Recognition erkannt. Sie haben versucht Mehrdeutigkeiten zu lösen, ohne den Freiheitsgrad beim Zeichnen zu reduzieren [Alvarado und Davis (2001), Alvarado und Davis (2004), Costagliola u. a. (2006), Brieler und Minas (2008)].

Skizzen enthalten häufig Mehrdeutigkeiten, welche die automatisierte Erkennung erschweren. Ein Sketch Recognition System könnte beispielsweise ein nicht sauber gezeichnetes Rechteck mit runden Ecken auch als Ellipse interpretiert. Je mehr Freiheiten ein Sketch Recognition System beim Zeichnen erlaubt, desto mehr Mehrdeutigkeiten kann eine Zeichnung enthalten. Den Inhalt von Skizzen zu beschränken ist eine Möglichkeit die Mehrdeutigkeiten zu minimieren. Allerdings verhindern solche Einschränkungen ein natürliches Skizzieren wie mit Stift und Papier [Brieler und Minas (2008)]. Es besteht also ein Zielkonflikt zwischen dem natürlichen Zeichnen von Skizzen und der einfachen Erkennung von Skizzen.

Das Lösen von Mehrdeutigkeiten basiert bei allen bekannten Untersuchungen darauf, Wahrscheinlichkeiten für die möglichen Interpretationen zu berechnen. Die Anzahl der theoretisch möglichen Interpretationen ist eine exponentielle Funktion. Es ist deshalb nicht handhabbar die Wahrscheinlichkeit für jede mögliche Interpretation zu berechnen [Alvarado und Davis (2004), Alvarado (2004)]. Es gibt verschiedene Verfahren, mit denen die Zahl der möglichen Interpretationen gering gehalten werden kann.

Ein Ansatz ist mithilfe der Syntax der Domäne einer Zeichnung die Erkennung kontextabhängig durchzuführen [Alvarado und Davis (2004), Brieler und Minas (2008), Costagliola u. a. (2006)]. Z.B. können in einem unsauber gezeichneten Petri Netz Stellen und Transitionen ähnlich aussehen (Kreise und Rechtecke). Eine Stelle darf aber nicht auf eine Stelle folgen bzw. eine Transition nicht auf eine Transition. Mit diesem Wissen können die Interpretationen ausgeschlossen werden, welche die Syntax verletzen.

Alvarado und Davis (2001) sowie Plimmer und Freeman (2007) haben heuristische Regeln verwendet, die dabei helfen, die korrekten Interpretationen finden. Eine solche Heuristik ist beispielsweise, dass man ein Objekt normalerweise zu Ende zeichnet, bevor man mit dem nächsten beginnt. Damit kann man Striche finden, die bei der weiteren Berechnung nicht berücksichtigt werden müssen, da sie mit hoher Wahrscheinlichkeit nicht Teil des gezeichneten Objekts sind, für das die Interpretation gesucht wird [Alvarado und Davis (2001)].

Die zuvor vorgestellten Verfahren können miteinander kombiniert werden. Ohne sie wäre eine zuverlässige Erkennung von Skizzen kaum möglich. Dennoch muss mit Fehlern bei der Erkennung gerechnet werden. Alvarado und Davis (2001) schlagen deshalb vor, den Benutzer fehlerhafte Interpretationen korrigieren zu lassen.

2.3 Domänenübergreifende Skizzen Erkennung

Die meisten bisherigen Untersuchungen haben sich auf die Erkennung einzelner Skizzen konzentriert. Mögliche Beziehungen zwischen den Skizzen spielten dabei keine Rolle. Wenn jedoch mehrere Skizzen verschiedene Sichtweisen auf den selben Sachverhalt beschreiben, können zwischen den Skizzen Beziehungen bestehen. Wenn das System die Skizzen verarbeiten soll, sind die Beziehungen von Bedeutung. Zum Beispiel kann die dynamische und statische Sicht eines Software Systems mit Diagrammen unterschiedlicher Domänen beschrieben werden (z. B. Klassendiagramme für die statische Sicht und Aktivitätsdiagramme für die dynamische Sicht). Die Beziehungen zwischen diesen Sichten sind von Bedeutung, wenn Code generiert werden soll.

Schmieder u. a. (2008) haben untersucht, wie ein Sketch Recognition System Skizzen aus verschiedenen Domänen unterstützen kann. Bei ihrem System InkKit muss der Benutzer dem System die Beziehungen zwischen den Skizzen mitteilen. Dafür muss der Benutzer die Diagramm-Elemente mit einem so genannten Rubberband verbinden [Schmieder u. a. (2008)]. InkKit kann für solche Skizzen Code generieren.

Schmieder u. a. (2008) schlagen vor, entweder ein Kommunikationsprotokoll oder ein geteiltes Objekt als Information Carrier zu verwenden, um bei der Verarbeitung von Skizzen die Abhängigkeiten zwischen den Skizzen aufzulösen. Wenn das System eines geteilten Objekts verwendet, muss es

vor der Verarbeitung alle verfügbaren Informationen aus den Skizzen sammeln und in dem geteilten Objekt speichern. Anschließend kann das System die Skizzen verarbeiten, wobei der Information Carrier die zur Verarbeitung benötigten Informationen über die Diagramm-Elemente liefert. Bei der Verwendung eines Kommunikationsprotokolls fragt das System während der Verarbeitung der Skizzen die benötigten Informationen ab. Wenn die Informationen über ein Element noch nicht verfügbar sind, stellt das System die Verarbeitung des Elements solange zurück, bis sie nach der Verarbeitung der anderen Elemente verfügbar sind. Laut Schmieder u. a. (2008) sei ein geteiltes Objekt zwar einfacher zu implementieren, könne aber nicht immer alle benötigten Informationen bereitstellen. Deshalb raten Schmieder u. a. (2008) dazu, ein Kommunikationsprotokoll zu verwenden.

2.4 Benutzbarkeit von Sketch Recognition Systemen

Die Benutzbarkeit von Sketch Recognition Systemen haben zahlreiche Untersuchungen als wichtig bezeichnet. Aber nur wenige Untersuchungen haben sich auf die Benutzbarkeit konzentriert. Plimmer u. a. (2006) sowie Wais u. a. (2007) haben untersucht, wie die Benutzbarkeit eines Sketch Recognition Systems gewährleistet werden kann.

Ein Faktor der negative Auswirkungen auf die Benutzbarkeit von Freihand-Zeichen-Werkzeugen haben kann, ist die begrenzte Displayfläche [Plimmer u. a. (2006)]. Eine Möglichkeit mit der begrenzten Displayfläche umzugehen ist, die Zeichenfläche zoombar und scrollbar zu machen. Plimmer u. a. (2006) haben dies in ihrer Untersuchung nicht in Betracht gezogen, weil die heutigen Eingabemethoden von Whiteboards dafür nicht geeignet seien. Eine Alternative ist, mehrere Zeichnungen zu verwenden und die Größe der einzelnen Zeichnungen zu begrenzen [Plimmer u. a. (2006)]. Um die Benutzbarkeit dieses Ansatzes zu gewährleisten, haben Plimmer u. a. (2006) ein zweites Display verwendet, welches das Portfolio der vorhandenen Zeichnungen angezeigt. Die zu bearbeitende Zeichnung kann der Benutzer aus dem Portfolio auswählen und wird auf dem anderen Display angezeigt. Laut dem Usability-Test von Plimmer u. a. (2006) beeinträchtigen die zwei getrennten Displays den Benutzer nicht bei seiner Arbeit.

Wais u. a. (2007) haben sich mit ihren Usability-Tests auf die Interaktion des Benutzers mit dem System konzentriert. Eines ihrer Ergebnisse ist, dass die Erkennung durch den Benutzer explizit ausgelöst werden sollte, wenn das System keine kontinuierliche Erkennung (siehe Kapitel 2.1.2) durchführt. Ein System, das den Zeitpunkt an dem die Erkennung stattfinden soll selbst ermittelt, kann den Benutzer verwirren [Wais u. a. (2007)]. Weiterhin haben Wais u. a. (2007) beobachtet, dass die Benutzer es akzeptieren, Erkennungsfehler zu korrigieren. Die Korrektur der Fehler sollte erst nach dem Abschluss der Zeichenaktivitäten erfolgen, um die Arbeit der Benutzer nicht zu unterbrechen Wais u. a. (2007)]. Außerdem sollte eine Skizze beim Feedback des Systems nur so wenig wie möglich verändert werden [Wais u. a. (2007)].

2.5 Texterkennung

Nahezu alle bisherigen Sketch Recognition Systeme können keinen mit der Hand geschriebenen Text erkennen. Bei einigen Systemen können die Elemente nach der Erkennung der Zeichnung

mithilfe einer Tastatur beschriftet werden. InkKit ist ein System, das erstmals die Erkennung von mit Hand geschriebenem Text mit der Erkennung von Skizzen kombiniert hat [Plimmer und Freeman (2007)].

Die herkömmliche Texterkennung unterscheidet sich grundlegend von der Erkennung von Formen [Davis (2007)]. Eine Erkennung von Text und Formen mit dem gleichen Algorithmus verringert deshalb die Erkennungsraten gegenüber der separaten Erkennung erheblich [Plimmer und Freeman (2007)]. Plimmer und Freeman (2007) haben deshalb mit InkKit das Konzept eines Dividers eingeführt. Der Divider analysiert jeden gezeichneten Strich und vergibt eine Wahrscheinlichkeit, nach der ein Strich Text ist oder nicht. Die weitere Erkennung erfolgt je nachdem, ob ein Strich als Text klassifiziert wurde mit einem herkömmlichen Verfahren zur Texterkennung oder einem Sketch Recognition Verfahren. Text und Formen können so mit gewohnter Zuverlässigkeit erkannt werden [Plimmer und Freeman (2007)].

2.6 Benutzererkennung beim kollaborativen Zeichnen

Beim gemeinsamen Zeichnen kann die Erkennung der Benutzer einen Mehrwert darstellen. Mit undo/redo Funktionen könnten Benutzer ihre eigenen Aktivitäten rückgängig machen, ohne die Arbeit anderer Benutzer zu zerstören. Ebenso könnte mit einer Replay-Funktion der Zeichenvorgang wiederholt werden, um den Gedankengang nachzuvollziehen. Außerdem könnte festgehalten werden, wer welchen Teil zu einem Diagramm beigetragen hat, sodass der Autor später gezielt gefragt werden kann, was er beabsichtigt hat [Eoff und Hammond (2009)].

Eoff und Hammond (2009) haben in ihrer Untersuchung festgestellt, dass es möglich ist, jeden einzelnen Strich mit einer hohen Wahrscheinlichkeit einem bestimmten Benutzer zuzuordnen. Zur Klassifikation können die Geschwindigkeit beim Zeichnen und die Neigung sowie der Druck des Stifts verwendet werden.

2.7 Erkennung von Skizzen bei der Softwareentwicklung

Einige Sketch Recognition Systeme wie Tahuti [Hammond und Davis (2002)], SUMLOW [Chen u. a. (2003)] oder Calico [Mangano u. a. (2008a)] sind speziell dafür entwickelt worden, um das Skizzieren von Diagrammtypen zu unterstützen, die bei der Entwicklung von Software verwendet werden (insbesondere UML-Diagramme). Viele andere Systeme, wie InkKit [Plimmer und Freeman (2007)] oder DSketch [Brieler und Minas (2008)] können konfiguriert werden, solche Diagrammtypen zu erkennen und wurden mit solchen Diagrammen getestet. Das Hauptziel bei der Entwicklung dieser Systeme war meistens, die Skizzen zuverlässig zu erkennen (z. B. Tahuti, InkKit, DSketch). Einige Untersuchungen haben sich darüber hinausgehend mit dem Einsatz von Sketch Recognition Systemen im Kontext der Softwareentwicklung auseinandergesetzt. Die Ergebnisse dieser Untersuchungen werden im Folgenden zusammengefasst.

Alle Untersuchungen sind sich einig, dass die Softwareentwicklung von der automatisierten Erkennung von Skizzen profitieren kann. Dafür spricht, dass Softwareentwickler häufig Modelle eines

Systems auf Papier oder Tafeln skizzieren [Cherubini u. a. (2007), Hammond u. a. (2002), Mangano u. a. (2008a)]. Die Digitalisierung und Persistierung solcher Skizzen ist sehr aufwendig. Deshalb werden die Skizzen häufig nicht digitalisiert bzw. persistiert und sind später nicht mehr verfügbar [Cherubini u. a. (2007), Hammond u. a. (2002)]. Jedoch besteht bei der Implementierung eines Systems bedarf an den Modellen. Die Modelle dienen bei der Implementierung als Bauplan oder werden bei modellgetriebener Softwareentwicklung zu ausführbarem Programmcode transformiert.

Es gibt zwar Software mit der Softwareentwickler grafische Modelle eines Systems erstellen können (z. B. UML Editoren). Trotzdem ziehen es laut einer Studie von Cherubini u. a. (2007) viele Softwareentwickler vor, Modelle zunächst per Hand auf Papier oder Tafeln zu skizzieren. Die Gründe dafür sind im Wesentlichen die Leichtigkeit Skizzen zu zeichnen und deren Freiheit von Einschränkungen [Cherubini u. a. (2007), Hammond u. a. (2002)]. Aus diesen Gründen verspricht der Einsatz von Sketch Recognition Systemen, die Produktivität bei der Entwicklung von Software zu erhöhen. Diesbezüglich gibt es aber noch keine Erkenntnisse über die tatsächlichen Auswirkungen auf die Softwareentwicklung.

Mangano u. a. (2008a) haben darüber hinaus untersucht, wie von Softwareentwicklern erstellte Skizzen aussehen. Charakteristisch ist die Unvollständigkeit der Skizzen. Die Skizzen beinhalten nur die Details, die relevant sind, um einen bestimmten Aspekt zu verstehen. Außerdem ist die Bedeutung einiger Teile der Skizzen häufig nicht eindeutig [Mangano u. a. (2008a)]. Charakteristisch ist weiterhin, dass Softwareentwickler beim Skizzieren oft keine formal korrekte Notation verwenden und verschiedene Notation mischen [Mangano u. a. (2008a)].

3 Abgrenzung von eigenen Arbeitszielen

Die in Kapitel 2 vorgestellten Forschungsergebnisse dienen als Grundlage für die Entwicklung eines Sketch Recognition Systems für die Modellierung von Software an Multitouch-Tischen.

Die Forschungen an Sketch Recognition Systemen wie SketchRead [Alvarado und Davis (2004)] InkKit [Plimmer und Freeman (2007)] oder DSketch [Brieler und Minas (2008)] haben untersucht, wie man effizient und zuverlässig Skizzen automatisiert erkennen kann. Dies ist kein Ziel bei der Entwicklung des eigenen Sketch Recognition Systems an der Hochschule für Angewandte Wissenschaften Hamburg. Statt dessen sollen die Ergebnisse der abgeschlossenen Forschungen so weit wie möglich wiederverwendet werden.

Das Hauptziel bei der Entwicklung eines eigenen Sketch Recognition Systems ist, zu untersuchen, wie solche Systeme bei der Entwicklung von Software eingesetzt werden können. Das eigene Sketch Recognition System wird Teil einer Entwicklungsumgebung werden, die das Microsoft Surface als kollaborative Arbeitsfläche verwendet. Ein Werkzeug das Reviews von Modellen unterstützt, ist ein bestehender Teil dieser Entwicklungsumgebung und wurde im Sommersemester 2009 entwickelt [Löffelholz u. a. (2009)]. Das Sketch Recognition System soll die Entwicklungsumgebung ergänzen, sodass neue Modelle skizziert werden können. Die in Kapitel 2.7 erwähnten Systeme erlauben das Skizzieren von Software Modellen. Das eigene System soll allerdings darüber hinausgehende Funktionalität bieten. Modelle, die das dynamische Verhalten von Software beschreiben (z. B. Petri Netze), sollen ähnlich wie bei ActiveCharts [Sarstedt u. a. (2007)] simuliert

werden können, was keines der vorhandenen Systeme kann. Außerdem sollen die Skizzen durch den Einsatz modellgetriebener Softwareentwicklung zum Prototyping eines Systems verwendet werden können. InkKit [Plimmer und Freeman (2007)] bietet Funktionalität zur Codegenerierung und kann, wie in Kapitel 2.3 erläutert, Beziehungen zwischen verschiedenen Zeichnungen herstellen. Der Ansatz von InkKit ist allerdings nicht mit der modellgetriebenen Softwareentwicklung konform. Insbesondere unterstützt InkKit keine Metamodelle.

Mangano u. a. (2008a) haben sich mit dem Einsatz von Sketch Recognition Systemen bei der Softwareentwicklung beschäftigt und verfolgen ähnliche Ziele. Sie haben sich allerdings hauptsächlich darauf konzentriert, wie und was Softwareentwickler skizzieren. Allerdings waren weder das Prototyping von Software noch die Simulation von Modellen Teil ihrer Arbeit.

Das eigene System wird auf einem Microsoft Surface laufen. Das Microsoft Surface ist ein Multitouch-Tisch und hat signifikante Unterschiede zu der Hardware, die bei den im vorigen Kapitel vorgestellten Systemen verwendeten wurde. Bisher haben Sketch Recognition Systeme elektronischen Whiteboards, Tablet PCs oder klassischen PCs verwendet. Im Gegensatz zum Microsoft Surface unterstützte diese Hardware weder die gleichzeitige Eingabe durch mehrere Benutzer noch die Erkennung von physikalischen Gegenständen. Die Erkennung von auf die Oberfläche gestellten Gegenständen verspricht neue Interaktionsmöglichkeiten [Microsoft (2009), Löffelholz u. a. (2009)], die bei der bisher verwendeten Hardware nicht möglich waren. Die Eingabemethode zum Zeichnen ist außerdem eine andere. Bei der bisher verwendeten Hardware wurde mit einem Stift oder einer Maus gezeichnet. Die Eingabe am Microsoft Surface erfolgt allerdings durch Kontakte mit den Fingern. Ein Teilziel der eigenen Forschungen ist deshalb, zu untersuchen, welche Besonderheiten Sketch Recognition Systeme auf Multitouch-Tischen haben und wie die Interaktion der Benutzer mit dem System aussehen kann.

Aufgrund der unterschiedlichen Eingabemethoden und Interaktionsmöglichkeiten ist unklar, welche der in Kapitel 2.4 erwähnten Usability-Richtlinien anwendbar sind. Zumindest die Verwendung von zwei verschiedenen Displays kommt nicht in Frage, da dies dem kollaborativen Arbeiten an einem Tisch widerspricht. Die übrigen Erkenntnisse über die Benutzbarkeit von Sketch Recognition Systemen können als Orientierung dienen. Die Entwicklung von eigenen Usability-Richtlinien ist aber kein Ziel und ist allenfalls ein Nebenprodukt.

Da das Microsoft Surface kollaborative Arbeit ermöglicht, könnte die Erkennung der Benutzer wie in Kapitel 2.6 beschrieben einen Mehrwert für das eigene System bieten. Eine weitergehende Forschung in dieser Richtung ist jedoch nicht beabsichtigt.

4 Zusammenfassung

Diese Ausarbeitung hat den aktuellen Forschungsstand bei der Sketch Recognition zusammengefasst. Die Grundlagen für die Entwicklung eines Sketch Recognition Systems sind bekannt. Die Forschung konzentriert sich nun darauf die Erkennung von Skizzen zuverlässiger zu machen, die Benutzbarkeit der Benutzungsschnittstellen zu verbessern und die Systeme um zusätzliche Funktionen zu ergänzen, was den Mehrwert von Sketch Recognition Systemen erhöhen soll. Es wird darüber hinaus der Einsatz von Sketch Recognition Systemen in bestimmten Domänen erforscht.

Besondere Relevanz hat die Forschung im Bereich der Softwareentwicklung, da an der Hochschule für Angewandte Wissenschaften Hamburg ein System zur Modellierung von Software mit Multitouch-Tischen entwickelt wird. Die vorherigen Untersuchungen haben allerdings andere Zielsetzungen verfolgt und werden als Basis für die Entwicklung eines eigenen Sketch Recognition Systems dienen.

Literatur

[Alvarado 2004] ALVARADO, Christine: *Multi-Domain Sketch Understanding*, Massachusetts Institute of Technology, Dissertation, August 2004

[Alvarado und Davis 2001] ALVARADO, Christine ; DAVIS, Randall: Resolving Ambiguities to Create a Natural Sketch Based Interface. In: *Proceedings. of IJCAI-2001*, URL http://rationale.csail.mit.edu/publications/Alvarado2001Resolving.pdf, August 2001

[Alvarado und Davis 2004] ALVARADO, Christine ; DAVIS, Randall: SketchREAD: a multi-domain sketch recognition engine. In: *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*. New York, NY, USA : ACM, 2004, S. 23–32. – ISBN 1-58113-957-8

[Brieler und Minas 2008] BRIELER, Florian ; MINAS, Mark: Recognition and processing of hand-drawn diagrams using syntactic and semantic analysis. In: *AVI '08: Proceedings of the working conference on Advanced visual interfaces*. New York, NY, USA : ACM, 2008, S. 181–188. – ISBN 0-978-60558-141-5

[Chen u. a. 2003] CHEN, Qi ; GRUNDY, John ; HOSKING, John: An e-whiteboard application to support early design-stage sketching of UML diagrams. In: *HCC '03: Proceedings of the 2003 IEEE Symposium on Human Centric Computing Languages and Environments*. Washington, DC, USA : IEEE Computer Society, 2003, S. 219–226. – ISBN 0-7803-8225-0

[Cherubini u. a. 2007] CHERUBINI, Mauro ; VENOLIA, Gina ; DELINE, Rob ; KO, Andrew J.: Let's go to the whiteboard: how and why software developers use drawings. In: *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 2007, S. 557–566. – ISBN 978-1-59593-593-9

[Costagliola u. a. 2005] COSTAGLIOLA, Gennaro ; DEUFEMIA, Vincenzo ; RISI, Michele: A Trainable System for Recognizing Diagrammatic Sketch Languages. In: *VLHCC '05: Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing*. Washington, DC, USA : IEEE Computer Society, 2005, S. 281–283. – ISBN 0-7695-2443-5

[Costagliola u. a. 2006] COSTAGLIOLA, Gennaro ; V, Vincenzo ; RISI, Michele: A Multi-layer Parsing Strategy for On-line Recognition of Hand-drawn Diagrams. In: *VLHCC '06: Proceedings of the Visual Languages and Human-Centric Computing*. Washington, DC, USA : IEEE Computer Society, 2006, S. 103–110. – ISBN 0-7695-2586-5

[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: *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 2000, S. 518–525. – ISBN 1-58113-216-6

[Davis 2007] DAVIS, Randall: Magic Paper: Sketch-Understanding Research. In: *Computer* 40 (2007), Nr. 9, S. 34–41. – ISSN 0018-9162

[Eoff und Hammond 2009] EOFF, Brian ; HAMMOND, Tracy: *Who Dotted That 'i'?: Context Free User Differentiation through Pressure and Tilt Pen Data*. 2009. – URL http://srlweb.cs.tamu.edu/srlng/research/paper/83?download=1&from=/srlng/research/

[Hammond 2009] HAMMOND, Tracy: IUI'09 workshop summary: sketch recognition. In: *IUI '09: Proceedings of the 13th international conference on Intelligent user interfaces*. New York, NY, USA : ACM, 2009, S. 501–502. – ISBN 978-1-60558-168-2

[Hammond und Davis 2002] HAMMOND, Tracy ; DAVIS, Randall: Tahuti: A Geometrical Sketch Recognition System for UML Class Diagrams. In: *AAAI Spring Symposium on Sketch Understanding*. Stanford, California : AAAI Press, March 25-27 2002, S. 59–68

[Hammond und Davis 2007] HAMMOND, Tracy ; DAVIS, Randall: LADDER, a sketching language for user interface developers. In: *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*. New York, NY, USA : ACM, 2007, S. 35

[Hammond u. a. 2008] HAMMOND, Tracy ; EOFF, Brian ; PAULSON, Brandon ; WOLIN, Aaron ; DAHMEN, Katie ; JOHNSTON, Joshua ; RAJAN, Pankaj: Free-sketch recognition: putting the chi in sketching. In: *CHI '08: CHI '08 extended abstracts on Human factors in computing systems*. New York, NY, USA : ACM, 2008, S. 3027–3032. – ISBN 978-1-60558-012-X

[Hammond u. a. 2002] HAMMOND, Tracy ; GAJOS, Krzysztof ; DAVIS, Randall ; SHROBE, Howard: *Sketch Recognition in Software Design*. 2002. – URL http://rationale.csail.mit.edu/pubs/hammond/2002HammondMITSoftware.pdf

[Hong u. a. 2002] HONG, Jason ; LANDAY, James ; L, James ; LONG, A. C. ; MANKOFF, Jennifer: Sketch Recognizers from the End-User's, the Designer's, and the Programmer's Perspective. In: *Sketch Understanding, Papers from the 2002 AAAI Spring Symposium*, 2002, S. 73–77

[Löffelholz u. a. 2009] LÖFFELHOLZ, Daniel ; PERGANDE, Thorben ; WITTERN, Hauke ; ZUKUNFT, Olaf: *Supporting early phases in software development through multitouch desks*. May 2009. – Unpublished

[Li u. a. 2005] LI, Junfeng ; ZHANG, Xiwen ; AO, Xiang ; DAI, Guozhong: Sketch recognition with continuous feedback based on incremental intention extraction. In: *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*. New York, NY, USA : ACM, 2005, S. 145–150. – ISBN 1-58113-894-6

[Liang u. a. 2005] LIANG, Shuang ; SUN, Zheng-Xing ; LI, Bin ; FENG, Gui-Huan: Effective sketch retrieval based on its contents, Aug. 2005, S. 5266–5272 Vol. 9

[Liao u. a. 2005] LIAO, Shi-Zhong ; WANG, Xiao-Jun ; LU, Jin-Liang: An incremental Bayesian approch to sketch recognition [approach read approach], Aug. 2005, S. 4549–4553 Vol. 7

[Mangano u. a. 2008a] MANGANO, Nicolas ; BAKER, Alex ; HOEK, André van der: Calico: a prototype sketching tool for modeling in early design. In: *MiSE '08: Proceedings of the 2008 international workshop on Models in software engineering*. New York, NY, USA : ACM, 2008, S. 63–68. – ISBN 978-1-60558-025-8

[Mangano u. a. 2008b] MANGANO, Nicolas ; BAKER, Alex ; HOEK, André van der: *Calico: A Tool for Early Software Design Sketching*. 2008. – URL https://www.cs.auckland.ac.nz/research/conferences/skekchws/proceedings/vlhcc_stws_p51.pdf

[Microsoft 2009] MICROSOFT CORPORATION: *Microsoft Surface*. 2009. – URL www.microsoft.com/surface

[Plimmer und Freeman 2007] PLIMMER, Beryl ; FREEMAN, Isaac: A toolkit approach to sketched diagram recognition. In: *BCS-HCI '07: Proceedings of the 21st British CHI Group Annual Conference on HCI 2007*. Swinton, UK, UK : British Computer Society, 2007, S. 205–213. – ISBN 978-1-902505-94-7

[Plimmer u. a. 2006] PLIMMER, Beryl ; TANG, Gene ; YOUNG, Mark: Sketch tool usability: Allowing the user to disengage. In: *People and Computers XX (Proceedings of HCI 2006)*. London : Springer, 2006. – URL http://www.cs.auckland.ac.nz/research/publications/index.php?pid=767

[Rubine 1991] RUBINE, Dean: Specifying gestures by example. In: *SIGGRAPH Comput. Graph.* 25 (1991), Nr. 4, S. 329–337. – ISSN 0097-8930

[Sarstedt u. a. 2007] SARSTEDT, Stefan ; KOHLMEYER, Jens ; RASCHKE, Alexander ; GESSENHARTER, Dominik: Eine pragmatische Umsetzung der MDA mit UML2-Aktivitätsdiagrammen. In: *OBJEKTspektrum* Nov./Dez. (2007), 06, S. 57–63. – URL http://www.de.capgemini-sdm.com/web4archiv/objects/download/pdf/1/sdm_pub_sarstedt.pdf

[Schmieder u. a. 2008] SCHMIEDER, Paul ; PLIMMER, Beryl ; VANDERDONCKT, Jean: *Cross-Domain Diagram Sketch Recognition*. 2008. – URL http://www.cs.auckland.ac.nz/research/publications/index.php?pid=897

[Sutherland 1963] SUTHERLAND, Ivan E.: Sketchpad: a man-machine graphical communication system. In: *AFIPS '63 (Spring): Proceedings of the May 21-23, 1963, spring joint computer conference*. New York, NY, USA : ACM, 1963, S. 329–346

[Wais u. a. 2007] WAIS, Paul ; WOLIN, Aaron ; ALVARADO, Christine: Designing a sketch recognition front-end: user perception of interface elements. In: *SBIM '07: Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*. New York, NY, USA : ACM, 2007, S. 99–106. – ISBN 978-1-59593-915-3