J. Ziegler & A. Schmidt (Hrsg.): Mensch & Computer 2010
München: Oldenbourg Verlag, 2010, S. 321-330

Eine Architektur zum flexiblen Einsatz von gestischer Interaktion

Jens Heydekorn, Mathias Frisch, Raimund Dachselt

Otto-von-Guericke-Universität Magdeburg

User Interfaces & Software Engineering

Zusammenfassung

Interaktive Systeme werden sich zukünftig immer nahtloser in die Umgebung einbinden. Dabei werden sich auch die Interaktionstechniken wandeln. Gesten spielen dabei eine sehr vielversprechende Rolle, da sie sich durch eine hohe Ausdrucksstärke und Flexibilität auszeichnen. In dieser Arbeit wird eine Architektur zur Realisierung von gestenbasierter Interaktion beschrieben. Sie erlaubt die Definition von Gesten mit einem geeigneten Eingabegerät, das Auswählen von Gesten zu einem Set und ihre Verwendung in einer Anwendung. Darüber hinaus erlaubt die Architektur die Wiederverwendung und plattformunabhängige Integration von bestehenden Komponenten zur Gestenerkennung. Die Anwendbarkeit wird anhand von zwei prototypischen Implementierungen dargestellt, die Multi-Touch- und Stift-Gesten unterstützen.

1 Einleitung

Bisherige Anwendungen basieren in der Regel auf der WIMP-Interaktionstechnik (WIMP: Windows, Icons, Menu, Pointer). Sie hat sich im Alltag etabliert und bildet zusammen mit der Maus als Eingabegerät und einem grafischen Ausgabemedium die Gruppe der „Point&Click“-Schnittstellen. In den letzten Jahren wurde jedoch eine Reihe von alternativen Interaktionsansätzen entwickelt. Beispiele sind Schnittstellen, die digitale Stifte, Berührungen mit Oberflächen oder zusätzliche Modalitäten wie Lage und Beschleunigung von realen Objekten zur Interaktion zulassen. Bei den genannten Beispielen sind Gesten als Kommunikationsform geeignet und werden vielfach eingesetzt. Eine konkrete Verwendungsform von Gesten wird als gestische Interaktionstechnik bezeichnet.

Da Gesten in der natürlichen Kommunikation weit verbreitet sind, orientieren sich gestische Interaktionstechniken oftmals an realen Vorbildern und Vorlagen. Besonders prominent sind Hand- und Stiftgesten, die ebenfalls in Kombination mit anderen Modalitäten verwendet werden. Freie Handgesten im Umfeld eines Tabletop-Systems zeigt beispielsweise die Arbeit (Wu & Balakrishnan 2003). Hier werden Gesten sowohl mit punktuellen Fingerberührungen

als auch mit flächigen Berührungen durch die ganze Hand realisiert. Dies demonstriert die große Ausdruckskraft von Gesten. Ein aktuelles Beispiel für gestische Interaktion mit Mobilgeräten stellt die Arbeit "Throw & Tilt" (Dachselt, Buchholz 2009) dar. Hier wird eine Wurf-Geste als eine Metapher verwendet, um Bilder auf einer großen Anzeige dazustellen.

Neben der Integration von einzelnen Gesten in ein interaktives System werden verwendete Gesten als Zusammenstellung (kurz: „set“) betrachtet. Eine Aufstellung von verwendeten Gesten mit der Funktion und Bedeutung für eine Anwendung ist ein Gestenset, das verwendet werden kann, um Interaktionstechniken auf weitere ähnlich geartete Anwendungen zu übertragen. Dies ist von Interesse, wenn zum Beispiel eine bestimmte Zusammenstelllung eine hohe Zufriedenheit bei bestimmten Benutzern bewirkt (Wobbrock et al. 2009) (Frisch et al. 2009). Für die Übertragung einmal realisierte Gestensets zwischen Anwendungen gibt es derzeit keine technische Unterstützung.

Gestische Interaktionstechniken werden durch Komponenten realisiert, die Signale von Eingabegeräten analysieren und die Analyseergebnisse zur Verwendung für eine Anwendung nutzbar gestalten. Diese Komponenten sind bisher kaum wiederverwendbar oder können durch Kommunikationsschnittstellen systemübergreifend genutzt werden. Die hier vorgestellte Systemarchitektur kann die genannten Probleme lösen. Sie erlaubt es Gesten zu erkennen und diese verschiedenen Anwendungen zu Verfügung zu stellen. Darüber hinaus enthält sie Komponenten zur Beschreibung von Gesten und deren flexible Zusammenstellung zu Gestensets. Prinzipiell erlaubt die Architektur die Verarbeitung von Daten aus beliebigen Eingabegräten, wobei bisher die Anbindung von Multi-Touch und digitalen Stiften realisiert und in zwei Beispielanwendungen umgesetzt worden ist.

Bevor die Architektur im dritten Abschnitt vorgestellt und beschrieben wird, werden im Folgenden zur Thematik relevante Arbeiten genannt und diskutiert. Im vierten Abschnitt werden zwei Anwendungen vorgestellt, bei denen die Architektur zur Anwendung kommt. Abschließend wird die Arbeit zusammengefasst und ein Ausblick auf die Weiterentwicklung dieser Architektur gegeben.

2 Verwandte Arbeiten

Die Erkennung von Gesten zusammen mit der Integration bzw. Nutzung in interaktiven Systemen ist seit langem Forschungsgegenstand. Dieser Abschnitt gibt einen Überblick über bestehende Arbeiten und nennt Techniken, die in der hier vorgestellten Architektur Verwendung finden. In Hinblick auf die Anwendungsbeispiele (s. Abschnitt 4) beschränkt sich die Betrachtung auf Tabletop-, Stift- und Multi-Touch-Systeme.

2.1 Sketch Recognizer

Grundlegendes Problem bei der Realisierung von gestischen Interaktionstechniken ist die Erkennung von Mustern aus dem Strom von Eingabesignalen bzw. der Vergleich von einem Satz von Eingabesignalen mit einem Referenzmuster. Da in dieser Arbeit die Eingaben in

aller Regel aus angedeuteten bzw. gezeichneten Formen bestehen, werden diese Komponenten als *Sketch Recognizer* bezeichnet. Zur Erkennung und Identifizierung solcher Eingaben sind zahlreiche Verfahren und Algorithmen bekannt, die sich unter anderem im Typ der Eingaben, der algorithmischen Verarbeitung und dem Laufzeitverhalten unterscheiden. Weit verbreitet zur Erkennung von solchen Eingaben ist der $1-Algorithmus (Wobbrock et al. 2007), der "Dynamic Time Warp"-Algorithmus (Salvador & Chan 2004) sowie der Algorithmus von (Rubin 1991). Darüber hinaus sind Systeme bekannt, die für bestimmte Anforderungen spezialisiert wurden, zum Beispiel für die Erkennung handgezeichneter Diagramme (Plimmer & Freeman 2007) oder allgemeiner zur Verwendung von digitalen Stiften (Signer et al. 2007).

Das hier vorgestellte System nutzt mehrere *Sketch Recognizer*, die für die zu unterstützenden Gesten am besten geeignet sind. Hierdurch sind Kombinationen von Multi-Touch-Gesten, Stift-Gesten und Gesten mit weiteren Geräten als Signalquelle möglich.

2.2 Erkennung von Multi-Touch Eingabesignalen

Die Signale und Daten für eine Gestenerkennung hängen von der hardwaretechnischen Realisierung ab, die sich stark unterscheiden kann. Unabhängig von der konkreten technischen Realisierung ist die Grundlage weiterer Verarbeitungsschritte die Behandlung von erkannten Merkmalen der Benutzereingabe, wie zum Beispiel Berührungspunkte, Hand- bzw. Fingerpositionen oder die räumliche Lage von Eingabegeräten.

Zur Beschreibung und Übermittlung von Multi-Touch-Daten kommt in aktuellen Entwicklungen häufig das TUIO-Protokoll (Kaltenbrunner et al. 2005) zum Einsatz. Verwendung findet dieses beispielsweise in Verbindung mit einer optischen Markererkennung beim System „reacTIVision“ (Kaltenbrunner et al. 2007) und bei CCV[49] zur optischen Multi-Touch-Erkennung. Es enthält eine Kommunikationsschnittstelle, zu deren Beschreibungsfähigkeit verschiedene benötigte Eigenschaften zählen, wie z.B. Positionen, Geschwindigkeiten oder Zeitangaben zu Berührungen oder Markern. Daneben sind weitere kommerzielle Systeme zu nennen, wie z.B. NUITEQ Snowflake[50], Microsoft Surface[51] oder SMART Table[52].

Das hier vorgestellte System kann ebenfalls entsprechende Erkennungs- und Beschreibungskomponenten (insbesondere CCV) zur grundlegenden Anbindung der Multi-Touch Hardware nutzen. Die Informationen, wie z.B. die erkannten Berührungspunkte, werden in weiteren Schritten verarbeitet und durch *Recognizer* analysiert.

2.3 Gestenerkennung aus Multi-Touch Signalen

Es sind verschiedene Systeme zur Nutzung von Multi-Touch Signalen bekannt. Beispielsweise ist DiamondTouch (Dietz & Leigh 2001) in der Lage, Multi-Touch Signale zu erken-

[49] Community Core Vision. http://ccv.nuigroup.com/
[50] NUITEQ Snowflake, http://natural-ui.com/products/snowflakesuite.php
[51] Microsoft Surface, http://www.microsoft.com/surface/
[52] SMART Table, http://smarttech.com/smarttable

nen und einem Benutzer zuzuordnen. Um Gesten für eine Benutzerschnittstelle zu ermöglichen, wird zusätzliche Software (Shen et al. 2004) benötigt. Diesem Prinzip folgen auch Techniken auf optischer Basis, wie das schon genannte CCV. Die Berührungen werden durch identifizierbare Bereiche in einem Kamerabild sichtbar. Hierauf können verschiedene Frameworks aufbauen, wie zum Beispiel mit Flash-Technologie gezeigt in (Barth & Leidecker 2009).

Ein Beispiel für die zentrale Haltung und Definition von Gesten kann bei (Li 2009) gefunden werden. Hier können Funktionen zu einfachen Buchstaben bzw. Strichgesten innerhalb eines Mobilgerätes für mehrere Anwendungen festgelegt werden. Die Arbeit von (Echtler & Klinker 2008) stellt eine allgemeine Softwarearchitektur vor, die Gestenerkennung (Interpretation Layer) sowie verschiedene Eingabegeräte (*Hardware Abstraction Layer*) berücksichtigt. Die Architektur bei (Ramanahally et al. 2009) entspricht in vielen Belangen dieser Beschreibung und enthält eine Komponente zur Gestenerkennung, die fest definierte Signale an die Anwendung bzw. die Benutzerschnittstelle weitergibt.

Alle genannten Systeme erlauben in aller Regel ausschließlich eine Signalquelle simultan. Ein System zur simultanen Anbindung verschiedener Eingabegeräte ist zum Beispiel OpenInterface (Serrano et al. 2008), das allerdings keine flexible Gestenverwaltung bereit hält. Die hier vorgestellt Architektur ist in der Lage, von mehreren Geräte parallel gelieferte Daten integrativ zur Gestenerkennung zu verwenden.

3 Architektur

Die hier vorgestellte Architektur trennt sich grob in zwei Ebenen: *Low Level* und *High Level*. Dies entspricht einem ähnlichen Ansatz, wie er in (Zhao 1993) vorgeschlagen wird. Der *Low Level* ist für die Signalerfassung zuständig und transformiert die eingehenden Abfolgen von Punktkoordinaten in Symbole. Ein Symbol besteht dabei in der Regel aus einer oder mehreren Signalfolgen. Die Ergebnisse dieser grundlegenden Erkennung werden dann an die *High Level* Komponente weitergeleitet. Diese realisiert die Abbildung der erkannten Symbole auf konkrete Gesten eines Gestensets und damit auf die Funktionen der Anwendung. Um bereits bestehende Erkennungskomponenten einzubinden, bzw. die Anbindung verschiedenster Applikationen zu ermöglichen, sollte Programmiersprachenunabhängigkeit gewährleistet sein. Diese kann durch die Realisierung von *Low* und *High Level* als Server-Komponenten erreicht werden. Einen Überblick über die Architektur gibt Abbildung 1. In den folgenden Abschnitten werden die einzelnen Bestandteile näher beschrieben.

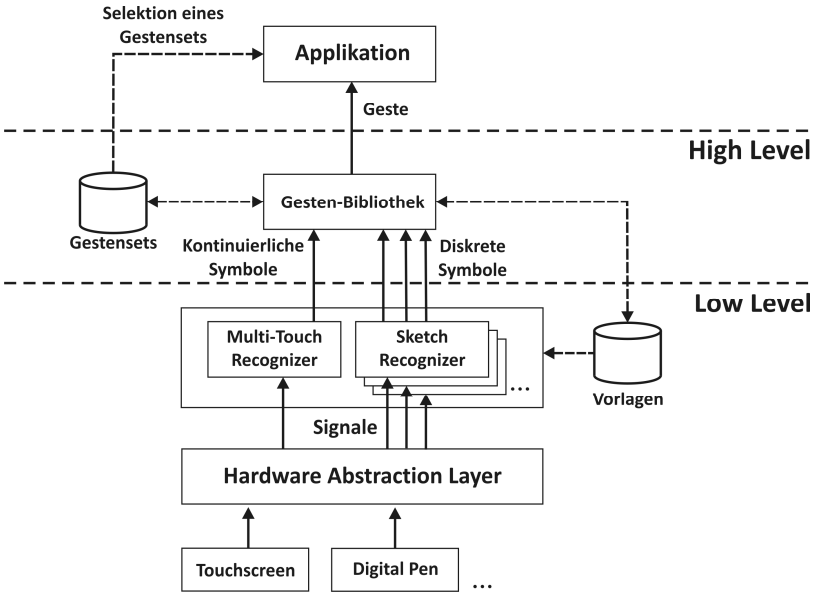

Abbildung 1: Überblicksansicht der Architektur

3.1 Low-Level

Der Bereich des *Low Level* beinhaltet Komponenten, die den Prozess vom Eingabesignal bis zu einem erkannten Symbol abdecken. Zu Beginn wird die Abstraktion zur Hardware bzw. zu den Eingabegeräten realisiert (*Hardware Abstraction Layer*, s. Abbildung1), wie zum Beispiel Touchscreens, digitale Stifte oder Sensoren für räumliche Gesten (z.B. Wii-Controller). Zurzeit ist die Anbindung von Multi-Touch-Displays über das TUIO Protokoll sowie von digitalen Stiften auf der Anoto-Technologie[53] realisiert. Die eingehenden Signale werden in ein einheitliches geräteunabhängiges Datenformat überführt. Dabei werden neben den Koordinaten der Eingabesignale auch eine eindeutige ID, der Zeitstempel und geräteabhängige Parameter wie z.B. die Gerätebezeichnung und Beschleunigung erfasst. Falls ein Wert nicht vom Eingabegerät zu Verfügung gestellt wird, muss er, soweit möglich, vom *Hardware Abstraction Layer* berechnet werden. Hierauf basierend erfolgt die Analyse der Signale in verschiedenen Erkennungskomponenten (*Recognizer*). Dabei existieren, in Hinblick auf die Verarbeitung der Signale, zwei verschiedene Ansätze: kontinuierliche Erkennung sowie diskrete Erkennung.

Kontinuierliche Recognizer nehmen Signale an, sobald diese auftreten, und sind dazu in der Lage, zum Beispiel zwischen sich bewegenden und sich nicht bewegenden Signalen zu unterscheiden. Sie leiten Veränderungen unverzüglich als kontinuierliche Symbole an die darü-

[53] Anoto Group AB, http://www.anoto.com/

ber liegende Schicht weiter. Die erkannten Symbole können dabei aus mehreren Eingabesignalen bestehen, etwa im Falle von Multi-Touch-Interaktion bzw. bei kombinierter Touch- und Stift-Interaktion. Der zurzeit verwendete *Multi-Point-Recognizer* verarbeitet 2D-Signale und erkennt drei verschiede Basis-Symbole: Tippen (kurze Berührung), Halten (lange Berührung ohne Bewegung) und Bewegen. Jedes dieser Symbole kann aus mehreren Signalfolgen bestehen. Insbesondere wird bei einer Bewegung, die aus mehreren Signalen besteht, die Bewegungen der Signale untereinander berücksichtigt. Dadurch können zum Beispiel *Pinch*-Gesten (Signale bewegen sich aufeinander zu) und *Rake*-Gesten (Signale bewegen sich parallel zueinander, siehe Abbildung 3 c)) unterschieden werden. Des weiteren werden Basis-Symbole auch kombiniert, etwa um eine asymmetrische Pinch-Geste zu erhalten (bestehend aus Halten und Bewegen, siehe Abbildung 3 d)-f)). Darüber hinaus werden weitere Werte eines Symbols berechnet, wie zum Beispiel der Schwerpunkt und die Rotation der Signale um den Schwerpunkt.

Diskrete Recognizer werten eingehende Signale hingegen nur zu einem definierten Zeitpunkt aus, zum Beispiel, wenn eine Signalfolge beendet ist oder nach dem Ablauf eines definierten Zeitfensters. Bei einer diskreten Erkennungskomponente kann es sich zum Beispiel um einen *Sketch-Recognizer* handeln, der digitales Zeichnen und visuelle Syntax unterstützen (siehe Abschnitt 2.1). Entsprechende Komponenten besitzen Zugriff auf eine Datenbasis, in der Referenzmuster für diskrete Symbole gehalten werden. Diese dienen zum Abgleich mit den eingehenden Signalfolgen, um daraus das wahrscheinlichste Symbol zu ermitteln. Zurzeit ist ein *Sketch-Recognizer*, der den $1-Algorithmus (Wobbrock et al. 2007) verwendet, eingebunden Die Architektur sieht darüber hinaus vor, dass mehrere diskrete *Recognizer* mit Hilfe eines Adapters angebunden werden und parallel angestoßen werden können. Alle erkannten Symbole werden an die Gesten-Bibliothek gesendet. Sie führt eine Häufigkeitsanalyse durch, um das Symbol mit der größten Wahrscheinlichkeit zu ermitteln, wobei die Charakteristiken der einzelnen Erkennungskomponenten berücksichtigt werden.

3.2 High Level

Der *High Level* Bereich besitzt Bezug zum Anwendungskontext und kann letztendlich Funktionen einer Anwendung mit Hilfe von Gesten auslösen. Zentrale Komponente ist die Gesten-Bibliothek, welche zur Organisation von bekannten Gesten dient. Sie nimmt sowohl kontinuierliche als auch diskrete Symbole aus dem *Low Level* an und bildet diese auf Funktionen der Anwendung, bzw. auf eine Geste innerhalb eines bestimmten Gestensets ab.

Die Festlegung der Abbildungsvorschrift und damit die Erzeugung eines entsprechenden Sets kann mit Hilfe eines Gesten-Browsers durchgeführt werden. Er ist Bestandteil der Gesten-Bibliothek und fungiert als deren Nutzerschnittstelle, um Zugang zu den beschreibenden Daten bekannter Gesten zu erlangen (siehe Abbildung 2). Der Gesten-Browser dient darüber hinaus dazu, die Gesten selbst festzulegen und zu ändern, bzw. Vorlagen für diskrete Symbole zu erzeugen. Diese werden dann jeweils in der entsprechenden Datenbasis abgelegt, auf die die Gesten-Bibliothek lesenden und schreibenden Zugriff hat. Die Speicherung erfolgt mit Hilfe eines XML-Formats. Eine auf dem System aufsitzende Applikation besitzt die Möglichkeit, ein konkretes Gestenset aus der Datenbasis auszuwählen und damit die für sie geltenden Abbildungsvorschriften festzulegen.

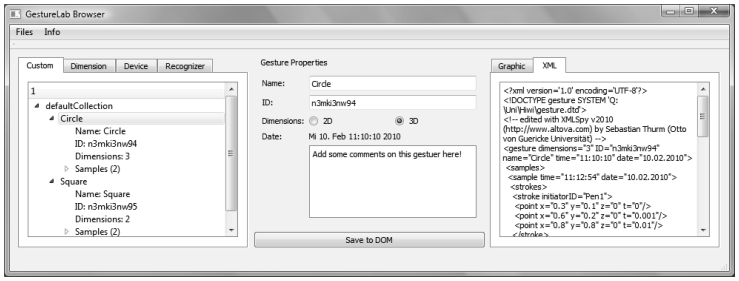

Abbildung 2 : Mit dem Browser können alle Gestensets, Gesten und einzelne Samples betrachtet werden: Auswahl der Gesten (links), Beschreibung in XML (rechts)

4 Anwendungsbeispiele

Die vorgestellte Architektur wird in unseren aktuellen Arbeiten mit Erfolg eingesetzt. Anhand der folgenden Beispiele wird die Praxistauglichkeit dieser Technik demonstriert. Die Beispiele decken verschiedene Anwendungsdomänen ab und wurden auf unterschiedlichen technischen Grundlagen umgesetzt.

4.1 Diagramm-Editor

Der in Java realisierte Diagramm-Editor ermöglicht die Erstellung und Bearbeitung von Graphen und Diagrammen unter der Verwendung von Multi-Touch-Gesten und digitalen Stiften. Mit Hilfe der vorgestellten Architektur können beide Modalitäten sowohl separat als auch in Kombination eingesetzt werden. Das verwendete Gesten-Set ist Nutzer-definiert und wurde durch eine Benutzerstudie erarbeitet (Frisch et al. 2009).

In Abbildung3 sind einige Beispiel-Gesten des Diagramm-Editors zu sehen. Kanten können zum Beispiel durch Zeichnen erstellt werden und Elemente können mittels Wisch-Geste gelöscht werden (Abbildung3 a) und b)). Diese Gesten verfolgen den Ansatz des Skizzierens, da sie die Interaktion mit Whiteboards nachahmen. Durch die Erkennung der Bewegung mit Hilfe des *Multi-Point-Recognizers* ist kontinuierliches Feedback möglich. Das jeweils gezeichnete Element (Kante bzw. Zick-Zack-Linie) wird von einem diskreten *Sketch-Recognizer* erkannt. Abbildung 3 c) zeigt die *Rake*-Geste, mit deren Hilfe eine durchgezogene Kante in eine gestrichelte Kante geändert werden kann. In Abbildung 3 d)-f) ist die asymmetrische *Pinch*-Geste illustriert. Sie ermöglicht das Kopieren eines Knotens und besteht aus zwei Basis-Gesten: Halten und Bewegen. Dabei kann Stift- und Touch-Interaktion kombiniert werden.

4.2 PicTable

Mit der in C# erstellten Anwendung „PicTable“ können Nutzer mit Hilfe von Stiftgesten durch Bildersammlungen browsen und Annotationen vornehmen (vgl. Abbildung 4). Es ist für mehrere Personen an einem Tabletop-System konzipiert, das ausschließlich mit digitalen

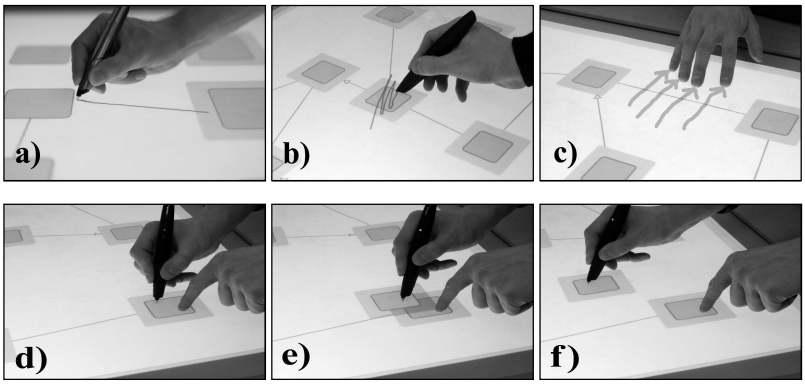

Abbildung 3: Erstellen einer Kante durch Zeichnen (a), Löschen eines Knotens durch Wischen (b), Erstellen einer gestrichelten Kante mit Hilfe der Rake-Geste (c), Kopieren eines Knotens durch Halten und Dragging (d-f)

Stiften bedient werden kann. Als Interaktionstechniken kommen Papierpaletten und Gesten zum Einsatz. Papierpaletten sind mit dem Stift selektierbare Menüs, die auf realem Papier gedruckt sind. Mit Gesten können zum Beispiel Sprungmarken gesetzt, sowie Bilder bewegt, gedreht und vergrößert werden. Sprungmarken bedienen sich einer Wurmloch-Metapher und dienen zum Bewegen von Bildern über Distanzen, die größer als die Armlänge eines Benutzers sind. Hierbei können an beliebigen Orten Sprungmarken durch eine Kreisgeste erzeugt werden, auf die Bilder gezogen und dadurch verschickt werden können. Das verwendete Gestenset kann im Framework frei variiert werden.

Abbildung 4 : Mit PicTable können Bildersammlungen frei über Papier-Paletten (linkes Bild) und Gesten annotiert werden. Sogenannte „Wurmlöcher“ helfen Distanzen zu überbrücken (rechtes Bild).

5 Zusammenfassung und Ausblick

Mit der vorgestellten Architektur können gestenbasierte Interaktionstechniken für Anwendungen flexibel eingesetzt werden. Die Architektur ist die Basis für ein Framework, das die freie Erstellung und Verwaltung einer Gestenbibliothek ermöglicht. In der Gestenbibliothek sind Gesten und Gestensets zentral gespeichert. Innerhalb des Frameworks ist es das Ziel, größtmögliche Flexibilität zum freien Einsatz von gestischer Interaktion zu gewähren und gleichzeitig technische Zusammenhänge (bspw. zu *Recognizern*) bei Bedarf aufrecht zu

erhalten. Die Realisierbarkeit dieser Architektur wurde durch die Implementierung des Frameworks und anhand von zwei prototypischen Entwicklungen demonstriert. Damit ist die Architektur eine wesentliche Grundlage für die universelle Einbindung von gestischen Interaktionstechniken und der Etablierung von einheitlichen Gestensets für bestimmte Domänen und Aufgaben.

Im Ausblick ist die Erweiterung zu unterschiedlichen Geräten zu realisieren, um hierdurch beispielsweise freie, räumliche Gesten zu unterstützen. Weiterhin ist die Architektur hin zu einem Modell generisch zu gestalten. In diesem Zusammenhang ist der Bezug zu semantischen Inhalten von Gesten sowie Kontexte zu Elementen der Benutzungsschnittstelle stärker zu berücksichtigen. Hierzu werden theoretische und konkrete Anforderungen an eine entsprechende Anwendung, als auch an das Framework direkt identifizier- und formulierbar.

6 Literaturverzeichnis

Barth, P. & Leidecker, T. (2009) Gestenerkennung für Multitouch-Anwendungen in UI-Frameworks. Konferenzband Mensch & Computer 2009, Oldenbourg, München, pp. 243-252.

Dachselt, R. & Buchholz, R. (2009) Natural Throw and Tilt Interaction between Mobile Phones and Distant Displays, In Extended Abstracts on Human Factors in Computing Systems, ACM.

Dietz, P. & Leigh, D. (2001) DiamondTouch: a multi-user touch technology. In Proceedings of Symposium on User interface Software and Technology, ACM, New York, pp. 219-226.

Echtler, F. & Klinker, G. (2008). A multitouch software architecture. In Proceedings of the Nordic Conference on Human-Computer interaction, ACM, pp. 463-466.

Frisch, M., Heydekorn, J., Dachselt, R. (2009) Investigating Multi-Touch and Pen Gestures for Diagram Editing on Interactive Surfaces. In Proceedings of Interactive Tabletops and Surfaces, ACM, New York, pp. 167-174.

Johnson, G., Gross, M. D., Hong, J., & Yi-Luen Do, E. (2009). Computational Support for Sketching in Design: A Review. Found. Trends Hum.-Comput. Interact., 2 (1), pp. 1-93.

Kaltenbrunner, M., Bovermann, T., Bencina, R. & Costanza, E. (2005) TUIO - A Protocol for Table Based Tangible User Interfaces. In Proceedings of Workshop on Gesture in Human-Computer Interaction and Simulation, Springer, Heidelberg.

Kaltenbrunner, M. & Bencina, R. (2007) reacTIVision: a computer-vision framework for table-based tangible interaction. In Proceedings of the 1st international Conference on Tangible and Embedded interaction, ACM, New York, NY, pp. 69-74.

Li, Y. (2009) Beyond Pinch and Flick: Enriching Mobile Gesture Interaction. Computer, 42(12), pp. 87-89.

Long, A. C., Landay, J. A., & Rowe, L. A. (1999) Implications for a gesture design tool. In Proceedings of the Conference on Human Factors in Computing Systems, ACM, New York, pp. 40-47.

Lyons, K., Brashear, H., Westeyn, T., Kim, J. & Starner, T.(2007) GART: The Gesture and Activity Recognition Toolkit, In Proceedings Human-Computer Interaction, Springer, Heidelberg, pp. 718-727.

Nielsen, M., Störring, M., Moeslund, T.B. & Granum, E. (2004) A procedure for developing intuitive and ergonomic gesture interfaces for HCI. In: Int'l Gesture Workshop, LNCS vol. 2915, Springer, Heidelberg, pp. 409-420.

Plimmer, B. & Freeman, I. (2007) A Toolkit Approach to Sketched Diagram Recognition. In Proceedings of Conference on HCI, British Computer Society, Swinton, pp. 205-213.

Rubin, D. (1991) Specifying Gestures by Example. Computer Graphics, 25(4), pp. 329-337.

Salvador, S. & Chan, P. (2004) FastDTW: Toward accurate dynamic time warping in linear time and space. Workshop on Mining Temporal and Sequential Data, ACM, New York.

Signer, B., Kurmann, U., & Norrie, M.C. iGesture (2007) A General Gesture Recognition Framework. In Proceedings of the International Conference on Document Analysis and Recognition, IEEE, pp. 954 - 958.

Ramanahally, P., Gilbert, S., Niedzielski, T., Velázquez, D. & Anagnost, C. (2009) Sparsh UI: A Multi-Touch Framework for Collaboration and Modular Gesture Recognition. Proceedings of the World Conference on Innovative VR 2009.

Serrano, M., Nigay, L., Lawson, J. L., Ramsay, A., Murray-Smith, R., & Denef, S. (2008) The openinterface framework: a tool for multimodal interaction. In Extended Abstracts on Human Factors in Computing Systems, ACM, New York, pp. 3501-3506.

Shen, C., Vernier, F. D., Forlines, C., & Ringel, M. (2004) DiamondSpin: an extensible toolkit for around-the-table interaction. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, New York, pp. 167-174.

Wobbrock, J. O., Wilson, A.D. & Li, Y. (2007) Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In Proceedings of User interface Software and Technology, ACM, New York, pp. 159-168.

Wobbrock J. O., Morris, M. R., & Wilson, A. D. (2009) User-defined gestures for surface computing. In Proceedings of the Conference on Human Factors in Computing Systems, ACM, New York, pp. 1083-1092.

Wu, M. & Balakrishnan, R. (2003) Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In Proceedings of User interface Software and Technology 2003, ACM, New York, pp. 193-202.

Zhao, R. (1993) Incremental recognition in gesture-based and syntax-directed diagram editors. In *Proceedings of the INTERACT '93 and Conference on Human Factors in Computing Systems* ACM, New York, pp. 95-100.

Kontaktinformationen

Jens Heydekorn (Otto-von-Guericke-Universität Magdeburg)

Telefon: +49 (391) 6712189

Telefax: +49 (391) 6711164

E-Mail: jens.heydekorn@ovgu.de