Diss

Stefan Oppl

26. März 2009

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Teil I

Umsetzung

1 Input und Interpretation

In diesem Kapitel wird jener Teil des Werkzeuges beschrieben, in dem die Interaktion der Benutzer mit dem Werkzeug erfasst und interpretiert wird. Der erste Abschnitt behandelt grundlegende Möglichkeiten zur Erfassung der Benutzerinteraktion auf tisch-basierten Benutzungsschnittstellen und endet mit der Identifikation der im konkreten Anwendungsfalls geeignetsten Technologie. Diese wird durch die Beschreibung von dafür verfügbaren Frameworks konkretisiert, was letztendlich in der Entscheidung für ein konkretes Produkt mündet.

Basierend auf dieser Entscheidung wird in den folgenden beiden Abschnitten auf das Design der Hard- und Softwarekomponenten eingegangen, die unmittelbar der Eingabe von Information durch Benutzer dienen. Der Ausgabeaspekt wird hier bewusst ausgeklammert und im nächsten Kapitel beschrieben. Dieses Kapitel endet mit eine Beschreibung der Interpretationsroutinen, die aus den durch das Framework gelieferten Rohdaten höherwertige, anwendungsspezifische Information extrahieren und diese den nachgeordneten Software-Modulen zur Verfügung stellen.

1.1 Möglichkeiten zur Erfassung von Benutzerinteraktion

Im Gegensatz zu Systemen mit dezidierten Eingabegeräten (wie Tastatur oder Maus) ist die Informationseingabe bei Tangible Interfaces unmittelbar an physische Tokens gebunden, die unabhängig voneinander und gegebenenfalls auch simultan manipuliert werden können. Diese Manipulation wird von einer vorhandenen Infrastruktur erfasst und im Sinne von Eingabedaten interpretiert. Der wesentliche Unterschied zu dezidierten Eingabegeräten besteht darin, dass die Manipulation des physischen Artefakts selbst für den Benutzer bedeutungstragend ist und nicht nur dem Zweck einer Zustandsänderung des digitalen Informationsraums dient. Dies impliziert, dass der Zustand der verwendeten Tokens bzw. der aktuelle Wert deren relevanten Parameter (z.B. Position, Rotation, Form, ...) erfasst werden kann, ohne die Bedeutung der Tokens noch deren Manipulierbarkeit in der realen Welt zu beeinflussen. Je nach Anwendungsfall kommen dafür mehrere Unterschiedliche

technologische Ansätze in Frage. Die Beurteilungskriterien die dabei zu berücksichtigen sind, liegen nicht nur in den zu erhebenden Parametern begründet sondern umfassen auch die notwendige Erfassungsrate des Zustandes der Tokens sowie die Anzahl der simultan zu erfassenden Tokens bzw. Eigenschaften.

Im konkreten System muss - wie in Kapitel XY beschrieben - die planare Position von mehreren Tokens in Echtzeit (d.h. mehrmals pro Sekunde mit für den Benutzer nicht wahrnehmbaren Verzögerungen) erfasst werden. Neben der Position ist noch die Rotation eines Tokens als Raumparameter von Interesse. Bezüglich des Zustands eines Tokens muss erfasst werden können, ob es geöffnet oder geschlossen ist und ob es eingebettete Objekte enthält oder nicht. Mit diesen Anforderungen wird in den folgenden Unterabschnitten ein technologischer Ansatz zur Umsetzung der Interaktionserkennung ausgewählt.

1.1.1 In Frage kommende technologische Ansätze

Bei der Auswahl möglicher technologischer Ansätze zur Erfassung der Benutzerinteraktion müssen die zu erfassenden Parameter unterschiedlich behandelt werden. Konkret werden hier Ansätze zur Erfassung der Raumparameter (Position, Rotation) und Ansätze die Zustandsänderungen des Tokens erfassbar machen unterschieden.

Raumparameter

Zur Erfassung von Raumparametern von Tokens bieten sich mehrere technologische Ansätze an. In Frage kommen für das konkrete - tisch-basierte System - nur Technologien, die eine Erfassung dieser Parameter mit einer Genauigkeit im Zentimeter- bis Millimeter-Bereich ermöglichen, da eine niedrigere Raumauflösung zu zu großen Ungenauigkeiten in der Positionsbestimmung führen würden, die einen Einsatz für das hier vorgestellte Werkzeug nicht erlauben würden. Im Folgenden werden die in Frage kommenden Technologien in ihren Grundzügen beschrieben und hinsichtlich ihrer Eignung für das konkrete System bewertet.

Optisch Optische Positionsbestimmung erfolgt immer mit Hilfe von Kamera-Systemen und Methoden der digitalen Bildverarbeitung. Die Kamera erfasst dabei die zu identifizierenden Tokens, dass resultierende Bild wird mit in Software umgesetzten Algorithmen ausgewertet, wodurch zumindest Identität und Position, zumeist aber auch weitere Raumparameter (wie Rotation) aller im Kamerabild befindlichen Tokens ermittelt werden können. Die Erkennung ist dabei auf den Erfassungsbereich der Kamera beschränkt. Neben diesem Einflussfaktor bestim-

men zudem die Auflösung der Kamera sowie die Größe der Tokens die letztendlich erfassbare Fläche. Optische Systeme sind generell bei schlechten oder wechselnden Lichtverhältnissen eher fehleranfällig und nicht robust gegen Verdeckungen von Tokens (etwa durch Gliedmaßen oder andere Tokens).

Hinsichtlich des Identifikationsansatzes können zwei Arten von Systemen unterschieden werden. *Codebasierte* Systeme verwenden zur Identifikation eines Tokens einen von der Kamera erfassbaren Code (etwa einen "Barcode"), der eindeutig einem Token zugeordnet werden kann. *Featurebasierte* Systeme identifizieren ein Token aufgrund seiner äußeren Eigenschaften, zumeist über dessen Form (Schattenriss). Letztere bieten den Vorteil, dass ein Token nicht durch das Anbringen eines zusätzlichen Codes optisch verändert werden muss. Der größte Nachteil besteht in der Eigenschaft, dass nur Token mit unterschiedlichen Formen eindeutig identifiziert werden können. Die eindeutige Identifikation von mehreren Tokens einer Bauart ist bei featurebasierten Systemen nicht möglich. Codebasierte Systeme verwenden zumeist nicht herkömmliche Barcodes sondern robustere Systeme, bei denen eine Erkennung auch unter widrigen Beleuchtungsbedingungen oder niedrigen Bildauflösungen möglich ist und die zum Teil auch die Extraktion zusätzliche Information über weitere Raumparameter (wie Rotation, teilweise auch Parameter der dritten Dimension wie Neigung oder Entfernung) ermöglichen.

Codebasierte Systeme können hinsichtlich der Art der Codierung der Indentitätsinformation wiederum in zwei Klassen unterschieden werden. Eine Gruppe von Ansätzen integriert die eigentliche Nutzinformation, also im Wesentlichen die tokenspezifische Identifikationsnummer, direkt in den Code und ermöglicht so ein direktes Auslesen der Information (z.B. bei QRCode (REF)). Die zweite Gruppe verwendet eine indirekte Zuordnung zwischen Token-ID und Code. Bei derartigen Ansätzen muss die Identität eines Tokens in einem Zwischenschritt über eine Mapping-Tabelle abgebildet werden, im Gegenzug ist die Ausgestaltung des Codes flexibler, im Allgemeinen kann dabei eine höhere Robustheit bei der Erkennung erreicht werden (z.B. ARToolkit (REF)).

Kapazitiv Kapazitive Ansätze basieren auf der Änderung der Kapazität von Leiterbahnen, die durch deren Berührung mit leitfähigem Material verursacht wird. Ursprünglich wurde die Technologie zur Umsetzung von berührungssensitiven Oberflächen entwickelt, kann jedoch auch zum Tracking von Tokens verwendet werden. Im Gegensatz zu druckempfindlichen Oberflächen (klassischen "Touchscreens") ist keine Druckausübung zur Erkennung notwendig, es können außerdem auch mehrere Tokens (bzw. Finger) gleichzeitig erkannt werden.

Technologisch bedingt müssen bei kapazitiven Ansätzen alle zu identifizierenden Objekte die Oberfläche des Systems berühren. In dieser Oberfläche ist ein Me-

tallgitter eingebettet, zwischen dessen Adern eine elektrische Kapazität gemessen werden kann. Diese Kapazität verändert sich, sobald diese Adern berührt werden (wobei die Token in einen entsprechend geeigneten Material ausgeführt sein müssen). Durch die lokale Änderung der Kapazität kann die Position einer Berührung festgestellt werden. Die Genauigkeit ist dabei durch die Rasterweite des Metallgitters eingeschränkt. Der größte Nachteil eines kapazitiven Ansatzes ist in diesem Kontext aber, dass die Identität eine Tokens nicht direkt festgestellt werden kann (die Kapazitätsänderung ist für alle Token identisch). Zudem ist die Extraktion weiterer Raumparameter (wie Rotation) nicht bzw. nur mit zusätzlichen Aufwand möglich. Die Vorteile von kapazitiven Systemen liegen in der hohen Robustheit der Erkennung auch bei widrigen Umgebungseinflüssen (Lichtverhältnisse, Schmutz) sowie der prinzipiell beliebig großen und beliebig geformten Oberfläche, die zur Erkennung verwendet werden kann.

Kapazitive Systeme eignen sich also zur Positionsbestimmung, nicht aber zur Identifikation von Tokens. Dies macht sie für den konkreten Anwendungsfall nur in Kombination mit einer anderen Technologie geeignet.

Elektromagnetisch Die Ausstattung von Tokens mit elektromagnetisch erfassbaren Einheiten (z.B. RFID-Chips) ermöglicht ebenfalls die Erfassung von Raumparametern. Vorrangig eignet sich diese Technologie jedoch zur Identifikation von Tokens, die Positionsbestimmung kann nur mit erheblichem technischen Aufwand durchgeführt werden.

RFID-Chips (als Beispiel für einen elektromagnetischen Ansatz) sind passive Bauteile, die bei Energieversorgung durch ein elektrisches Feld aktiv werden und ihrerseits eine eindeutige Identifikationsnummer senden (im einfachsten Fall, komplexere Varianten sind möglich, werden hier aber nicht betrachtet). Historisch stammt die Technologie aus der Logistik und Warenwirtschaft und dient der Identifikation von Gütern und nicht der exakten Positionsbestimmung. Diese ist somit auch nur mittels erweiterter Infrastruktur möglich. Zum Auslesen eines RFID-Chips wird ein Lesegerät mit Antenne benötigt. Aus der Feldstärke, mit der diese Antenne die Antwort des Chips empfängt, kann auf die Entfernung des Chips von der Antenne geschlossen werden. Durch Kreuzpeilung mit mindestens zwei Antennen, deren Position bekannt ist, kann somit auf die ungefähre Position des Chips (und damit des Tokens, in das dieser eingebaut ist) geschlossen werden. Durch den Einsatz von "Antennenarrays" (matrixförmig angeordneten Antennen) mit geringer Reichweite ist so eine verhältnismäßig exakte (Größenordnung einige cm) Positionsbestimmung möglich. Die Feststellung der Ausrichtung eines Tokens (Rotation) ist auf diesem Wege allerdings nicht möglich. Die Identifikation eines Tokens ist jedoch unabhängig von Sichtkontakt und unmittelbarer Berührung und

somit äußerst robust gegen Umgebungseinflüsse.

Elektromagnetische Systeme eignen sich wegen des hohen technischen Aufwandes bei gleichzeitig beschränkter Genauigkeit nur bedingt zur Feststellung von Raumparametern. Durch die Ausrichtung auf Extraktion der Identitätsinformation ist der Ansatz jedoch gut zur Kombination mit anderen Technologien wie kapazitiven Ansätzen geeignet, die ihre Stärken in der Bestimmung der Raumparameter haben.

Akustisch Akustische Ansätze zur Positionsbestimmung basieren im Generellen auf der Laufzeitmessung von Ultraschallwellen im Raum. Mit entsprechender Infrastruktur ist damit in einem begrenzten Bereich ein hochexakte Feststellung der Raumparameter in drei Dimensionen (Genauigkeit im mm-Berich) sowie die Identifikation von Tokens möglich.

Ultraschallbasierte Techniken zur Positionsbestimmung basieren auf dem Einsatz von Bakensendern an bekannten Positionen. Diese Sender werden zumeist an der Zimmerdecke montiert und senden periodisch einen Ultraschallimpuls aus. Dieser Impuls wird von den Tokens (die in diesem Fall aktive Bauteile mit Stromversorgung sind) empfangen, die daraufhin einen sie identifizierenden Impuls zurücksenden. Aus der Laufzeit zwischen Absetzen des Sendeimpuls und Empfangen des Antwortimpulses bei verschiedenen Baken lässt sich so die Position des Tokens im Raum feststellen. Problematisch ist hierbei jedoch die durch den auf sequentieller Zeitmessung basierenden Ansatz beschränkte Anzahl von verfolgbaren Tokens, wenn Echtzeit-Ansprüche gestellt werden. Zudem ist der Ansatz nicht robust gegen (akustisch) verdeckte Tokens. Eine Anfälligkeit gegenüber anderen Störeinflüssen besteht nicht.

Für die Feststellung von Raumparametern sind ultraschall-basierte Systeme generell ausgezeichnet geeignet. Auch die Identifikation von Tokens ist prinzipiell möglich. Bei der Bewertung hinsichtlich des Einsatzes für tisch-basierte Systeme ist jedoch zu bedenken, dass eine drei-dimensionale Positionierung nicht zu den allgemeinen Anforderungen zählt und nur in speziellen Anwendungsfällen sinnvoll sein kann. Zudem kann die Notwendigkeit von stromversorgten Tokens einen Nachteil bzw. ein Hindernis beim Einsatz darstellen.

Bewertung Im konkreten Anwendungsfall ist die Feststellung der Identität sowie der planaren Position und Rotation von mehreren Tokens in hoher Genauigkeit sowie in Echtzeit gefordert. Aus oben genannten Gründen sind kapazitive und elektromagnetische Systeme im Einzeleinsatz nur bedingt geeignet. Akustische Systeme erscheinen für den Anwendungsfall als zu aufwändig und unflexibel und stoßen außerdem bei der Anzahl der simultan zu verfolgenden Tokens an ihre

Grenzen.

Die Kombination von kapazitiven und elektromagnetischen Systemen ist grundsätzlich eine Möglichkeit, die in Betracht gezogen werden könnte. Auch optische Systeme genügen den Anforderungen und kommen damit in Frage. Der kombinierte Ansatz ist im Vergleich mit optischen Systemen als robuster gegen Störeinflüsse aus der Umgebung zu betrachten. Für optische Systeme sprechen hingegen die weitaus geringeren Aufwände für Infrastruktur und Tokens sowohl bei Anschaffung als auch bei Wartung und Betrieb. Durch die geringere Komplexität des Systems sind auch weniger potentielle Fehlerquellen vorhanden, was bei der Erstellung des Werkzeug-Prototypen hilfreich ist. Aufgrund dieser Aspekte und einer vergleichbaren zur erwartenden Erkennungsleistung wurde für die hier vorgestellten Anwendungfalls die Entscheidung getroffen, ein optisches System zur Bestimmung der Positionsparameter sowie der Identität der Tokens einzusetzen.

Tokenzustand

Hinsichtlich des Tokenzustands sind im Kontext des hier vorgestellten Anwendungsfall Informationen zu erheben, die den Inhalt des Tokens betreffen. Wie in Kapitel XY beschrieben sind die Modellierungs-Tokens als Container ausgeführt, die geöffnet und geschlossen werden können und in die kleiner Tokens als Trägen von Zusatzinformation hineingelegt werden können. Die Auswahl eines Ansatzes, der die Identifikation des Öffnungs-Zustandes eines Tokens sowie dessen Inhalt erlaubt, ist Gegenstand dieses Abschnitts. Dazu wird grundlegend zwischen dem Einsatz von passiven Tokens und aktiven Tokens unterschieden. Passive Tokens besitzen keine zusätzliche Elektronik, die geforderten Informationen können lediglich durch die bereits vorhandene (optische) Infrastruktur festgestellt werden. Aktive Tokens werden hingegen mit zusätzlicher Elektronik zur Zustandsbestimmung ausgestattet, was allerdings eine Energieversorgung jedes Tokens bedingt.

Passive Token Bei passiven Tokens muss sichergestellt werden, dass die bereits vorhandene Infrastruktur die Zustandsänderungen eines Tokens erfassen kann. Das die vorhandene Infrastruktur auf optischen Technologien basiert, müssen sich alle Zustandsänderungen im äußeren - durch die Kamera erfassbaren - Erscheinungsbild eines Tokens wieder spiegeln.

Der Öffnungszustand eines Tokens kann durch Kameras einfach erfasst werden, wenn sich - je nach eigesetzter Technologie - durch das Öffnen der Umriss des Tokens verändert oder ein weiterer Code sichtbar wird bzw. der bestehende Code modifiziert wird. Diese Anforderung kann also durch passive Tokens erfüllt werden.

Zur Erfassung des Inhalts eines Container-Tokens sind zwei Ansätze denkbar.

Einerseits kann der Inhalt eines Tokens zu einem bestimmten Zeitpunkt erfasst werden, andererseits ist auch eine Erfassung der Änderung des Tokeninhalts möglich (Erfassung des Vorgangs von Hineinlegen und Herausnehmen). Diese beiden Möglichkeiten sind hinsichtlich der Umsetzbarkeit mit passiven Token unterschiedlich zu beurteilen. Eine Erfassung das aktuellen Tokeninhalts ist mit optischen Systemen nur schwer möglich. Die einzige sich bietende Möglichkeit ist die von transparenten Teilbereichen der Außenfläche eines Tokens. Damit ist es grundsätzlich möglich, den Inhalt eines Tokens mit einer externen Kamera zu erfassen, sowohl bei feature- als auch code-basierten Ansätzen sind jedoch Verdeckungen, Verzerrungen oder zu geringe Kameraauflösung potentiell problematisch und lassen diesen Ansatz für den praktischen Einsatz als ungeeignet erscheinen.

Die Erfassung der Änderung des Tokeninhalts lässt sich mit optischen Systemen einfach implementieren. So kann der Vorgang des Hineinlegens als auch des Herausnehmens von einer Kamera erfasst werden. Die größte Herausforderung hierbei ist die Identifikation des Tokens, das eingebettet wird. Hier kann es wiederum durch Verdeckungen zu Erkennungsschwierigkeiten führen, was in diesem Fall einen permanent fehlerhaften Modellzustand zur Folge hat, der sich im Falle wiederholter Fehlerkennungen sogar inkrementell verschlimmern kann. Diesem Umstand kann lediglich durch eine explizite Aktion des Benutzers Rechnung getragen werden, der das betreffende einzubettende Token ins Sichtfeld der Kamera halten muss, bis das System Feedback über eine erfolgreiche Erkennung gibt. Diese Lösung erscheint allerdings hinsichtlich der Anforderung, die Technologie für den Benutzer vollkommen in den Hintergrund treten zu lassen, als eher suboptimal.

Aktive Token Aktive Tokens beinhalten zusätzliche Sensorik, die die Erfassung des Tokenzustands ermöglicht. Derartige Tokens benötigen allerdings eine Energieversorgung und müssen über eine Möglichkeit zur Datenübertragung verfügen, um den Tokenzustand an das System zu übermitteln. Weiters ist im Allgemeinen eine Steuereinheit notwendig, um die Sensoren zu kontrollieren, die Daten zu aggregieren und letzendlich zu übertagen.

Im konkreten Fall einer optisch arbeitenden Infrastruktur bietet sich eine (ggf. aufladbare) Batterie als Energiequelle an, um im Kamerabild Verdeckungen durch ansonsten eventuell zu verwendende Kabel zu vermeiden. Eine Stromversorung über die Oberfläche (wie z.B. im Smart PINS (Gellersen REF) Ansatz vergestellt) scheidet hier aus, da die Blöcke dann mit Krafteinsatz auf die Oberfläche gesetzt werden müssten und nicht verschoben werden können.

Als Steuerungseinheit bietet sich neben selbst auf der Basis von Mikrocontrollern wie dem PIC oder 8051 (REFs) konzipierten Systemen auch Plattformen an, die explizit für den Anwendungszweck der Ansteuerung von Sensoren oder Aktuatoren

und der Kommunikation mit einem Basissystem gefertigt werden. Exemplarisch kann hier die Smart-ITs-Plattform (REF) angeführt werden, die neben der flexiblen Ansteuerbarkeiten von unterschiedlichen Sensoren bereits Module zur Vernetzung untereinander und mit zentralen Diensten in der Infrastruktur anbietet.

Aus den eben angeführten Gründen erscheint zur Datenübertragung eine drahtlos arbeitende Technologie am geeignetsten. Aufgrund der geringen benötigten Reichweite und der Anforderung, möglichst energieeffizient zu arbeiten, bieten sich die Technologien "Bluetooth" und "ZigBee" an. Bluetooth erreicht höhere Übertragungsraten, ist aber in der Anzahl der gleichzeitig verwendbaren Geräte (max. 7) für den hier vorgestellten Anwendungsfall zu beschränkt. Ein ZigBee-Netz kann mit bis zu 255 Geräten gleichzeitig arbeiten und ist außerdem im Einsatz energiesparender. Für den gegebenen Anwendungsfall erschiene also ZigBee als geeignete Technologie (und wurde auch bereits in (AON Cube, Simon Vogl REF) in einem ähnlichen Anwendungsfall erfolgreich eingesetzt).

Zur Feststellung des Öffnungsstatus eines Container-Tokens bieten sich bei aktiven Sensortechnologien mehrere Möglichkeiten an. Der Einsatz eines Schaltelements, das beim Öffnen den Kontakt herstellt oder unterbricht, erscheint als eine nahe liegende Lösung. Auch der Einsatz eines Drehelements am Angelpunkt des Öffnungsschaniers, dessen elektrische Eigenschaften (z.B. Widerstand oder Kapazität) mit dem Öffnungswinkel ändern, kann angedacht werden. Damit ist nicht nur eine Unterscheidung zwischen "offen" oder "geschlossen" sondern auch die Identifikation von Zwischenzuständen möglich.

Der Inhalt eines Container-Tokens kann ebenfalls mit unterschiedlichen Technologien erfasst werden. Die Zielsetzung ist hier nicht der Positionsbestimmung der eingebetteten Tokens sondern lediglich die Feststellung derer Identität. Naheliegend ist hierzu der Einsatz von elektromagnetischen Ansätzen wie oben beschrieben. Durch das Anbringen von z.B. RFID-Chips an den einzubettenden Tokens sowie eines Lesegeräts im Container-Token kann die Identifikation robust durchgeführt werden. Alternativ bieten sich Systeme an, die auf Gewichtsmessung basieren. Über einen in das Containertoken eingebauten Sensor wird dabei das Gesamtgewicht der eingebetteten Tokens bestimmt. Bei entsprechender Konzeption der einzubettenden Tokens (unterschiedliche Gewichte) kann aus dem Gesamtgewicht auf die tatsächlich enthaltenen Tokens geschlossen werden. Ein Nachteil dieses Ansatzes ist die beschränkte Anzahl von Tokens und die notwendige exakte Fertigung jedes einzelnen Tokens, da es bei Gewichtsabweichungen zu Fehlerkennungen kommt.

Bewertung Hinsichtlich der erreichbaren Flexibilität und zu erwartenden Servicequalität wäre in diesem Abschnitt eine Entscheidung zugunsten aktiver Tokens

zu treffen. Im Gesamtkontext betrachtet und unter Berücksichtigung der Entscheidung für optische Systeme zur Bestimmung der Positionsparameter ist diese Wahl jedoch zu relativieren. Wie oben beschrieben, erlaubt eine auf optischen Systemen beruhende Infrastruktur grundlegend die Umsetzung der geforderten Funktionalität. Gleichzeitig wird die Komplexität des Systems massiv reduziert und die Erstellung zusätzlicher Tokens vereinfacht (da keine zusätzliche Elektronik notwendig ist). Der Wegfall von Energieversorgung und Sensorlogik in den Token reduziert deren Gewicht und ermöglicht gleichzeitig mehr Platz für einzubettende Tokens.

Für den hier beschriebenen Anwendungsfalls bzw. die prototypische Umsetzung des Werkzeugs wird deshalb auf aktive Tokens verzichtet und der Einsatz von passiven Tokens bevorzugt. Der Mehraufwand in Erstellung und Wartung des Systems beim Einsatz aktiver Tokens wiegt in der Gesamtheit betrachtet die zu erwartende höhere Erkennungsqualität nicht auf.

1.1.2 In Frage kommende Frameworks

Unter Anbetracht der im vorherigen Abschnitt getroffenen grundlegender Technologieentscheidung zugunsten optischer Erkennungstechnologie mit passiven Tokens werden nun unterschiedliche Frameworks betrachtet, die die Umsetzung dieses Ansatzes erlauben. Es sind dabei zwei Klassen von Frameworks zu unterscheiden. *Generische Frameworks für Tangible Interfaces* beschäftigen sich generell mit dem zur Verfügung stellen von Services, die Kopplung von Sensoren und Aktuatoren mit Interpretations-Logik und letzendlich konkreten Applikationen erlauben. Sie gehen dabei nicht auf konkrete Sensortechnologie (wie die hier verwendeten optischen Ansätze) ein sondern versuchen eine Abstraktionsebene einzuführen, die die Applikationen von der konkreten Technologie entkoppelt und damit flexibler macht. *Frameworks für video-basierten Input für Tangible Interfaces* sind hochspezialisierte Produkte, die konkret für die Umsetzung von optischen Ansätzen zur Eingabe von Daten bei Tangible Interfaces entwickelt werden. Ihr Vorteil liegt im durch die Spezialisierung im Allgemeinen geringeren Aufwand zur Einrichtung und auch während des Betriebs. Echtzeit-Anforderungen sind oft nur mit spezialisierten Frameworks zu erreichen. Eine Kombinationsmöglichkeit zwischen Produkten der beiden Kategorien ergibt sich beim Einsatz eines spezialisierten Frameworks als Eingabe-Modul für eine generisches Framework. Durch diesen Ansatz kann die einfache Inbetriebnahme spezialisierter Frameworks mit der Flexibiltiät generischer Frameworks zusammengeführt werden.

Generische Frameworks

Generische Frameworks zur Behandlung von Input und Output bei Tangible Interfaces sind historisch nicht exakt von anderen Frameworks abzugrenzen, die im Umfeld des Ubiquitous bzw. Pervasive Computing entwickelt wurden. Derartige Ansätze wurden erstmals im Zusammenhang mit "Context Computing" erwähnt, um Applikationen eine generische Möglichkeit zu bieten, Information aus der Umgebung über beliebige Sensoren zu erfassen und diese zu aggregieren und zu interpretieren. Aufbauend auf dieser Interpretation sollen Aussagen über den aktuellen Zustand der Umgebung (den "Kontext") getroffen werden können, die diese Applikationen zur Adaption benutzen können. Der Rückkanal, also die Ansteuerung von Aktuatoren, wurde erst in späteren Entwicklungen berücksichtigt. Die meisten Systeme dienen explizit nicht der Erstellung von marktreifen Applikationen sonderen widmen sich eher der Umsetzung von "Rapid Prototyping"-Ansätzen im Bereich der Tangible Interfaces. Begründet wird dies mit der oft suboptimalen Ressourcen-Ausnutzung, die mit der Generalisierung und Flexiblisierung des Frameworks einhergeht.

Die Aufzählung der hier beschriebenen Frameworks erhebt keinen Anspruch auf Vollständigkeit. Es wurde eher darauf geachtet, historische bzw. für den konkreten Anwendungsfall geeignete Ansätze aufzunehmen und in ihren wesentlichen Eigenschaften zu beschreiben.

Context Toolkit Das Context Toolkit (Dey et al., 2001) ist das historisch erste Framework, das versucht, die starre Verbindung zwischen Sensoren und Applikationslogik aufzubrechen und eine konfigurierbare Schicht einzuziehen, die eine schnellere, generischere Applikationsentwicklung ermöglicht und die Wiederverwendbarkeit einmal entwickelter Komponenten erhöht.

Konzeptuell existieren im Framework drei Arten von Komponenten: Context Widgets, Context Interpreters und Context Aggregators. Context Widgets implementieren die Ansteuerung beliebiger Software- und Hardwaresensoren und sind für das Sammeln von Information über die Umgebung zuständig. Sie vermitteln zwischen der physischen Umgebung und den konzeptuell höher liegenden Komponenten indem sie die unverarbeiteten Kontextdaten mittels einer geeigneten Schnittstelle kapseln und bestimmte Funktionen zur ersten Auswertung der Rohdaten ausführen. Eine Anwendung kann diese Daten verwenden, ohne dass sie Detailkenntnisse über die zugrunde liegenden Sensortechnologien haben muss. Context Interpreter aggregieren die Sensordaten zu komplexeren Kontextinformationen d.h. sie konvertieren und interpretieren die Daten mehrerer Context Widgets und versuchen diese zu einheitlichen Clustern zusammenzufassen. Context Aggregators dient der Zusammenführung verschiedener Kontextinformationen die

für bestimmte Anwendungen relevant sind. Context Aggregators bilden damit die Schnittstelle zu den eigentlichen Applikationen.

Das Context Toolkit bietet nicht nur eine Softwareschnittstelle zu physischen Sensoren, es trennt auch die Akquisition und Repräsentation von der Auslieferung der Daten an kontextsensitive Applikationen.

SiLiCon Context Framework Das SiLiCon Context Framework (Beer et al., 2003) ist ein Vertreter jener Klasse von Frameworks, deren Verhalten zur Laufzeit dynamisch konfigurierbar sind. Dies bedeutet im konkreten Fall, dass Applikationen die auf Basis des SiLiCon Context Framework erstellt wurden ihr Verhalten und ihren Aufbau aufgrund eintretender Ereignisse verändern können. Dies betrifft sowohl das Aktivieren und Deaktivieren von Input- und Output-Kanälen also auch die Interpretation der eingehenden Information und die Reaktion darauf. Das Framework wurde entworfen, um Szenarien zu beschreiben, in denen interaktive Systeme kontextsensitiv - d.h. abhängig vom aktuellen Zustand ihrer Umwelt - reagieren müssen.

Die grundlegenden Bausteine des SiLiCon Context Framework sind "Entiäten", die Objekte der realen Welt konzeptuell abbilden. Diese "Entitäten" besitzen "Attribute", also Eigenschaften, mit Hilfe derer die Entität näher beschrieben wird. Über "Attribute" kann die Wahrnehmung einer Entität von deren Umwelt sowie deren Interaktionsmöglichkeiten mit derselben beschrieben werden. Mit Hilfe von "ECA"-Regeln (*E*vent-*C*ondition-*A*ction) wird beschrieben, auf welche Wahrnehmung der Umwelt (Event) eine Entität unter welchen Bedingungnen (Condition, formuliert auf Basis des internen Zustands der Entität) mit welchen Aktivitäten (Action) reagiert. Diese Regeln können zur Laufzeit dynamisch verändert und nachgeladen werden. Außerdem ist es möglich, in "Actions" das Nachladen von Entitäten oder das Hinzufügen oder Entfernen einzelner Attribute durchzuführen (Oppl, 2004, S. 90).

Das SiLiCon Context Framework abstrahiert durch seine konzeptionelle Struktur mit dem Einsatz von "Entitäten" und "Attributen" nicht so stark von der realen Welt wie der Context Toolkit Ansatz - die Abbildung ist im ersten Schritt "direkter" und muss erst im zweiten Schritt technisch konkretisiert werden, ein klassischer Softwareengineeringprozess (im Sinne von "Analyse - Design - Implementierung") wird damit vollständiger (auch in den ersten Phasen) durch das Framework abgebildet und unterstützt.

Papiermaché Eines der ersten Rapid-Prototyping Frameworks

TUIpist Das TUIpist-Framework (Furtmüller, 2007) wurde im Zusammenhang mit der hier vorgestellten Arbeit entwickelt(Furtmüller and Oppl, 2007). TUIpist verfolgt einen ähnlich modularen Ansatz wie die anderen hier vorgestellten Frameworks, setzt jedoch zur Koordination der Module untereinander einen datenzentrierten Ansatz - auf dem LINDA-Konzept (REF) beruhende Tuplespaces - ein. Die konzeptionelle Modulstruktur ist ähnlich der Aufteilung, die bereits von Dey et al. (2001) im Context Toolkit vorgeschlagen wurde.

Die grundlegenden Module, die im Framework verwendet werden, sind "Sensoren", "Aggragtoren" und "Anwendungen / Aktuatoren" (siehe Grafik XY 1). "Sensor"-Module binden externe Datenquellen an das Framework an. Sie enthalten dazu eine sensor-spezifische Komponente, die die Schnittstelle zur jeweiligen Hard- bzw. Software bildet. Über diese Schnittstelle gelieferte Daten werden in einer zweiten Komponente vorverarbeitet und soweit abstrahiert, dass die Datenrepräsentation unabhängig von der die Daten liefernden Sensortechnologie ist (z.B. Abbildung von GPS-Koordinaten auf logische Positionsinformation, die auch aus anderen Quellen stammen könnte). "Aggregatoren" fassen die Information mehrerer "Sensor"-Module zusammen und interpretieren ggf. einander ergänzende oder auch widersprechende Information. Sie werden immer aktiv, wenn neue Sensordaten zur Verfügung stehen und aktualisieren dabei die Information über den Gesamtzustand der den Framework bekannten Umwelt. "Anwendungen / Aktuatoren" bilden letztendlich die Schnittstelle zu konkreten Applikationen, die ihr Verhalten an den aktuellen Umweltzustand anpassen bzw. diesen darstellen oder Aktuatoren, die basierend auf dem aktuellen Umweltzustand Aktionen in dieser setzen. "Anwendungen / Aktuatoren" filtern dabei wieder den von "Aggregatoren" gelieferten Gesamtzustand der bekannten Umwelt und liefern nur die für die Applikation relevanten Daten aus. Dabei kann erneut eine Nachverarbeitung der Daten, z.B. im Sinne einer Anpassung an eine externe Schnittstelle, erfolgen.

Die Verbindung der Komponenten erfolgt wie erwähnt mittels einem datenzentrierten Ansatz. Eine wesentliche Eigenschaft dieses Ansatzes ist, dass keine explizite Verknüpfungen einzelner Module definiert werden. Die Zuordnung erfolgt vielmehr indirekt durch die Daten selbst. Jedes Modul kann Datensätze (Tupel) in einem definierten Format generieren und in einen gemeinsam genutzten Datenraum - den Tuplespace - stellen. Andere Module können nun Anfragen an den Tuplespace stellen, ob Daten, deren Struktur oder Inahlt gewissen Kriterien entspricht, vorhanden sind. Ist dies der Fall, können diese Daten aus dem Tuplespace entnommen werden und nach erfolgter Verarbeitung in modifizierter Form wieder eingestellt werden (siehe Abbildung XY). Das Tupelspace-Konzept erlaubt auch eine dynamische Erweiterung bzw. Veränderung sowohl der Ein- und Ausgabekanäle als auch der internen Datenverabeitung zur Laufzeit, indem zusätzlich Module am Tupelspace registriert werden. Durch die lose Koppelung der Module muss keine

zusätzliche Konfiguration an anderen Modulen oder am Tuplespace selbst vorgenommen werden.

Die Implementierung von TUIpist auf Basis des Java Jini-Frameworks (REF) ermöglicht eine Verteilung der einzelnen Module einer Applikation auf unterschiedliche Rechner (im Sinne eines "verteilten Systems" (REF)) ohne zusätzlich vom Implementierer zu leistenden Koordinationsaufwand. Es können damit auch (räumlich) entfernte Sensoren oder Webapplikationen angebunden werden und der ggf. auftretende Rechenaufwand zur Aggregation oder Interpretation von Daten auf mehrere Rechner verteilt werden.

- Grafiken aus TICE Paper? -

Frameworks für video-basierten Input

Im Gegensatz zu den eben beschriebenen generischen Frameworks wurden die hier vorgestellten Frameworks explizit für die Behandlung von video-basiertem Input entwickelt. Wie oben bereits erwähnt stehen die beschriebenen Frameworks mit obigen insofern in Zusammenhang, also dass sie zumeist als Sensor-Komponente in generischen Ansätzen eingesetzt werden können. In ihrer grundlegenden Ausrichtung sind Sie jedoch für den unmittelbaren Einsatz in einer Endanwendung konzipiert.

Die hier vorgestellten Systeme implementieren den Ansatz des code-basierten optischen Trackings. Feature-basierte Ansätze kommen im konkreten Anwendungsfall nicht in Frage, da zur Modellierung eine Vielzahl von gleichartigen Objekten eingesetzt wird, die sich in ihrem äußeren Erscheinungsbild nicht unterscheiden und damit in feature-basierten Ansätzen nicht eindeutig identifiziert werden können.

Wie bereits im letzten Abschnitt erhebt auch die hier angeführte Aufzählung keinen Anspruch auf Vollständigkeit. Neben der Darstellung der historischen Entwicklung des Feldes und der Beschreibung von in Forschung und Praxis relevanten Ansätzen wurde bei der Auswahl vor allem auf freie Verwendbarkeit und Zugriff auf den Source-Code der Erkennungsroutinen geachtet, da die Notwendigkeit applikationsspezifischer Anpassungen nicht auszuschließen war (etwa um benötigte aber nicht direkt unterstüzte Parameter zu extrahieren).

ARToolkit Historisch erstes Framework, Mapping, beschränkter Code Raum

Visual Codes Das Visual Codes System ist ein Vertreter der direkt codierenden Ansätze, d.h. dass die Nutzinformation ohne Zwischenschritt direkt aus dem Code extrahiert werden kann. Im Gegensatz zu den standardisierten und kom-

merziell genutzten Code-Formate QR-Code (REF) oder Datamatrix (REF), die eine Kapazität von bis zu einigen hundert Byte haben, bietet das Visual Code System lediglich 80 Bits an Nutzinformation an, was dazu führt, das in vielen Anwendungsfällen ein Mapping durch die Anwendung durchgeführt wird, um die zu verwaltende Information abbilden zu können.

-- Abb. Visual Codes --

Der Vorteil des Visual Code Systems liegt in seiner mächtigen Auswertbarkeit bei gleichzeitig geringem Bedarf an Rechenkapazität. In der Standardimplementierung werden neben der Position und der Rotation in der Ebene auch die Neigungsparameter im Raum extrahiert. Der Erkennungsalgorithmus läuft dabei in Echtzeit und kann unverändert sogar auf Java-fähigen Mobiltelefonen ausgeführt werden. Durch die relativ geringe Datendicht der Codes reichen dementsprechend auch verhältnismäßig niedrig auflösende Bilder (z.B. 320 x 240 Bildpunkte) für eine Erkennung aus. Wie alle anderen optischen Ansätzen leidet das Visual Code System unter Erkennungsproblemen bei wechselnden Lichtverhältnissen und insbesondere bei schlechtem Kontrast. Diese Verhalten kann in der vorliegenden Implementierung auch nicht durch Rekonfiguration korrigiert werden.

Das Visual Code System muss von auf ihm aufbauenden Applikationen direkt auf Source-Code-Ebene eingebunden werden, eine vorgegebene externe Schnittstelle existiert nicht. Anbindungroutinen für Kameras existieren nur für Mobiltelefonen und müssen für andere Plattformen ggf. neu erstellt werden.

ReacTIVision ReacTIVision (REF) ist ein frei verfügbares Framework zu optischen Erkennung von Codes in Echtzeit. ReacTIVision arbeitet mit proprietären Codes (siehe Abbildung 1.1), in denen die Information nicht an bestimmte Positionen sondern im Wesentlichen in die Anzahl und Schachtelung der Schwarz-Weiß-Übergänge codiert ist. Die Mächtigkeit der Informationscodierung direkt in den Code ist damit beschränkt, es wird deswegen ein Mapping-Ansatz verwendet, um die eigentliche Nutzinformation auf Codes abzubilden. Grundsätzlich wäre jedoch die direkte Codierung eine beschränkten Anzahl von Bits möglich, ist aber nicht unmittelbar vorgesehen.

Durch die Arte der Informationscodierung ist die Erkennungsleistung von ReacTIVision auch unter schlechten Lichtbedingungen oder bei verzerrtem Eingangsbildern akzeptabel bis sehr gut. Die Form der Codes ist außerdem nicht vorgegeben, sie kann frei gewählt werden. Sogar händisch gezeichnete Codes können erkannt werden, da ausschließlich eine geschlossene Außenlinie und entsprechende Schwarz-Weiß-Übergänge innerhalb dieser Linie erfassbar sein müssen.

Die ReacTIVision-Software ist plattformübergreifend für Windows, Linux und

Abbildung 1.1: ReacTIVision Code

Mac OS X verfügbar. Sie greift über plattformspezifische Schnittstellen auf angeschlossene Kamera zu und wertet das empfangene Bild in Echtzeit aus. Dabei sind bei einer Kameraauflösung von 1024 x 68 Bildpunkten Bildraten von 15-20 Bildern pro Sekunde erreichbar. Diese Leistung wird auch durch eine höhere Anzahl von gleichzeitig im Bild vorhandenen Codes nicht merklich geringer.

Neben der Position der Codes wird auch deren aktuelle Rotation sowie die erste Ableitung dieser drei Parameter (also ein Maß für die Bewegung) extrahiert. Zudem können Finger, die die Oberfläche berühren erkannt und deren Position extrahiert werden. Dies ist auch für mehrere Finger möglich, wobei eine eindeutige Zuordnung über die Zeit erhalten bleibt und so rudimentäre Multitouch-Funktionen umgesetzt werden können.

Als problematisch stellt sich wie auch bei allen anderen betrachteten optischen Ansätzen die Abhängigkeit der Erkennungsqualität von der Umgebungsbeleuchtung bzw. deren Änderung dar. ReacTIVision arbeitet mit adaptiven Filteralgorithmen zur Aufbereitung des Bildes, was jedoch nur leichte Beleuchtungsschwankungen ausgleichen kann. Die Software kann händisch an die Beleuchtungsverhältnisse angepasst werden (Einstellung der Blendenöffnung und der Bildverstärkung), bei sich ändernden Lichtverhältnissen muss jedoch regelmäßig eine manuelle Nachführung der Parameter vorgenommen werden.

Die aus dem Bilderstrom gewonnenen Daten werden im Falle einer Änderung zumindest eines Wertes über ein Netzwerkschnittstelle (UDP-basiert) in einem propritären Protokoll zur Verfügung gestellt. Zu diesem Protokoll werden Schnittstellen und Referenzimplementierungen in unterschiedlichen Programmiersprachen, unter anderem C(++) und Java, zur Verfügung gestellt. Applikationen können diese Schnittstelle implementieren und werden sie mittels insgesamt sechs zu implementierenden Methoden an die Erkennungsroutinen angebunden (3 für Code- und 3 für Fingertracking).

1.1.3 Technologieentscheidung

Auf Basis der Anforderungen, die im Anwendungsfall an das Werkzeug gestellt werden, ist nun nach der grundsätzlichen Entscheidung für ein auf optischen Ansätzen basierenden System die Entscheidung für ein konkretes Framework zu treffen.

Vergleich der Frameworks für videobasieren Input

Für die Anbindung von videobasiertem Input wurde drei Frameworks vorgestellt. Diese Frameworks sind nun hinsichtlich mehrere Aspekte zu vergleichen, die sich aus den Anforderungen der zu erstellenden Applikation sowie der Forderung nach möglichst einfacher (im Sinne von unaufwändiger) Einbindung des Frameworks ergeben. Im Einzelnen sind dies

- die Unterstützung bei der Einbindung von Kameras, eine Schnittstelle zur Programmiersprache Java,
- eine stabile und in Echtzeit ablaufende Bilderkennung,
- eine ausreichende Anzahl von Codes (Größenordnung 100),
- die Extraktion von Position und Rotationsinformation für Codes im Erfassungsbereich der Kamera sowie
- die technische und lizenzrechtliche Möglichkeit, die Frameworks an die eigenen Anforderungen anzupassen.

ReacTIVision bietet die umfassendste Unterstützung zur Einbindung unterschiedlicher Videoquellen und ist zur Anwendungsseite hin durch die Netzwerkschnittstelle am flexibelsten einsetzbar. Alle drei Ansätze bieten Schnittstellen bzw. Konnektoren für die Programmiersprache Java an. Die Frameworks selbst sind in C(++) oder Java erstellt und können auf den gängigen Plattformen (Windows, Linux, Mac OS x) ausgeführt werden. Eine Verteilung der Applikation auf mehrere Rechner wird nur von ReacTIVision explizit unterstützt.

Hinsichtlich der eigentlichen Bilderkennungs-Routinen sind die Ansätze in ihrer Leistung und Geschwindigkeit vergleichbar, leiden aber alle unter der Abhängigkeit von der Umgebungsbeleuchtung. ReacTIVision bietet hier als einziger Ansatz die Möglichkeit, Kameraparameter zur Laufzeit nachzuführen und so Schwankungen der Umgebungshelligkeit auszugleichen.

Durch den eingesetzten Mapping-Ansatz sind ARToolkit und ReacTIVision hinsichtlich Form und Inhalt der Codes flexibler als direkt codierende Ansätze wie Visual Codes. Dies ist im konkreten Anwendungsfall relevant, da aufgrund der Token-Form und deren beschränkter Größe eine ideale Platzausnutzung erfolgen muss, um ausreichende Erkennungsleistung zu gewährleisten.

Die Extraktion der Raumparameter (Neigungsinformation, ...), die von ARToolkit und Visual Codes ermöglicht wird, ist in bei tisch-basierten Werkzeugen wie dem hier vorgestellten nicht notwendig - die Erhebung planarer Parameter (Position und Rotation) ist ausreichend.

Die Anzahl der verfügbaren und zuverlässig unterscheidbaren Codes muss für die hier vorgeschlagene Anwendung in der Größenordnung 100 liegen. Visual Codes und ReacTIVision erfüllen diese Anforderung, ARToolkit bietet im Lieferumfang weniger Codes an, diese können jedoch erweitert werden und bieten auch in der gerforderten Anzahl noch ausreichend Unterscheidungsmerkmale für eine zuverlässige Unterscheidung (REF Schmalstieg TU System).

Von allen drei Systemen steht der Source-Code zur Verfüng. Lizenzrechtlich erlauben ebenfalls alle Systeme eine Veränderung des Sourcecode (LIZENZEN AUFZÄHLEN)

	ARToolkit	Visual Codes	ReacTIVision
Kamera-unterstützung	?	nativ nur für Mobiltelefone	nativ auf allen unterstützten Plattformen via USB und Firewire
Plattformen	x	Java-basiert auf allen Plattformen	Windows, Linux, Mac OS X
Schnittstelle zu Java	via Dritt-Software	nativ	ja
Stabilität der Bilderkennung	eher hoch	eher hoch	hoch
Geschwindigkeit der Bilderkennung	$>$ 10 fps	$>$ 10 fps	$>$ 10 fps
Anzahl von Codes	eher hoch	hoch	hoch
Erkennung von Position	ja	ja	ja
Erkennung von Rotation	ja	ja	ja
Source-Code verfügbar	ja	ja	ja
Lizenz	?	?	?

Auf Basis dieses Vergleichs ist erkennbar, dass das ReacTIVision-Framework für den hier verfolgten Ansatz am besten geeignet erscheint. Es wird deshalb zur Umsetzung des Werkzeugs verwendet. Der zusätzliche Einsatz eines generischen Frameworks zur Flexibilisierung der Applikationsstruktur ist damit nach wie vor möglich und wird im nächsten Abschnitt diskutiert.

Vergleich der generischen Frameworks

Der Einsatz eines generischen Frameworks ist im konkreten Anwendungsfall für die Modularisierung der Interpretations- und Output-Module denkbar. Eine weitere Anwendung von generischen Frameworks ist die Anforderung nach der Skalierbarkeit der Anzahl der Ausgabemodule ggf. auch während des Betriebs der Applikation (um z.B. weitere Viewer-Module zur Beobachung des Modellierungsvorganges während der Laufzeit zuschalten zu können). Die Aspekte die beim Vergleich der vorgestellten Ansätze berücksichtigt werden müssen, sind deshalb

- die Unterstützung von modularen funktionalen Einheiten sowie
- die Möglichkeit diese zur Laufzeit zu laden und entfernen und
- die Konfigurierbarkeit der Verknüpfungen dieser Module.
- Daneben sind nicht-funktionale Anforderungen wie Skalierbarkeit und
- Effizienz bei der Weiterleitung und Verteilung der Anwendungsdaten

zu berücksichtigen.

Die Modularisierung der Applikation wird von allen vier vorgestellten Frameworks unterstützt. Das Context Toolkit, Papiermaché und TUIpist unterscheiden dabei konzeptuell zwischen unterschiedlichen Arten von von Modulen (im Wesentlichen "Input", "Verarbeitung" und "Output"), das SiLiCon Context Framework unterscheidet nicht zwischen unterschiedlichen Modultypen, dort wird die Rolle eines Moduls ausschließlich über seine Attribute (also die nach außen sichtbaren funktionalen Einheiten) definiert.

Eine Erweiterbarkeit der Applikation zur Laufzeit ist nur beim SiLiCon Context Framework und bei TUIpist möglich. DAs SiLiCon Context Framework arbeitet hier mit eigens erstellten Java-Classloadern, die das Nachladen von Klassen (Modulen) und deren Integration in die Infrasturktur erlauben. TUIpist verwendet die inhärent dynamische Erweiterbarkeit des Java Jini (REF) Frameworks, auf Basis dessen es implementiert wurde. Das Context Toolkit und Papiermaché erlauben kein Nachladen oder Entfernen funktionaler Einheiten während der Laufzeit.

Die Konfiguration der Verbindungen zwischen den Modulen ist bei allen Frameworks möglich, konzeptuell jedoch unterschiedlich ausgeführt. Im Context Toolkit und in Papiermaché muss die Verbindung direkt im Programmcode codiert

werden und wird im wesentlichen auf Methodenaufrufe abgebildet. Das SiLiCon Context Framework verwendet ereignisbasierte Regeln, um Module zu verknüpfen. Ein von einem Modul ausgelöstes Ereignis kann hier (ggf. durch Bedingungen eingeschränkt) Aktionen in anderen Modulen auslösen. Eine Besonderheit dieses Ansatzes ist, das Teile der Interpretationslogik (also der Erkennung von Benutzerinteraktion aus den Rohdaten) in die Regeln ausgelagert werden und damit jederzeit dynamisch verändert werden können. So kann zum Beispiel die Interpretation der Nähe zwischen zwei Token in einer Regel codiert werden und so in die Reihe der zulässigen (bzw. erkennbaren) Interaktionen aufgenommen werden. TUIpist benötigt hingegen keinerlei explizite Verbindung der Module, da sämtliche Koordination über den geteilten Datenraum ausgeführt wird. Die Verknüpfungslogik wird hier in die Module verschoben, die selbst wissen müssen, für welche Daten für sie ggf. relevant sind und welche sie deshalb dem Tuplespace entnehmen.

Hinsichtlich der Skalierbarkeit der Frameworks können aufgrund mangelnder Vergleichsdaten keine fundierten Aussagen getroffen werden. Hinzuweisen ist jedoch auf die Unterstützung von verteiltem Betrieb einer Applikation durch das SiLiCon Context Framework (via SOAP-Aufrufe (REF) und das TUIpist Framework (via Jini und Java RMI (REF)). Durch diese Verteilbarkeit ist kann die Problematik mangelnder Rechenressource umgangen werden, die vorallem bei rechenintensiven Anwendungen wie der eingesetzten Bildanalyse im optischen Tracking auftritt.

Die Beurteilung der Effizienz der Frameworks ist hier in Hinblick auf die geforderte Echtzeit-Interaktion mit dem System als relevant zu erachten. Das Haupt-Kriterium für Effizienz ist dementsprechend die Verzögerung zwischen Eingabe-Ereignissen und dem Feedback auf Applikationsebene, die beim Einsatz der Frameworks auftritt. Eine Beurteilung in absoluten Zahlen kann hier mangels entsprechenden Datenquellen ebenfalls nicht erfolgen, konzeptuell sind aber bei den Frameworks, in denen Module direkt durch Programm-Code verknüpft werden (Context Toolkit und Papiermaché), eher geringe Verzögerungen zu erwarte. Die beiden Frameworks, die mit expliziter und konfigurierbarer Middleware zur Datenvermittlung arbeiten (SiLiCon Context Framework und TUIpist) lassen eher höhere Verzögerungen erwarten, wobei der ereignisbasierte Ansatz des SiLiCon Context Framework dem daten-basieren Ansatz von TUIpist insofern überlegen ist, als das bei der datenbasierten Interaktion die Module in regelmäßigen Intervallen nach neuen Daten suchen müssen (Polling, Information Pull-Ansatz), während bei ereignisbasierten Ansätzen die Benachrichtung der betroffenen Module automatisch und zum Zeitpunkt des Auftretens des Ereignisses erfolgt (Information Push-Ansatz). Der Information Pull-Ansatz sorgt dabei einerseits für erhöhten Kommunikationsaufwand und potentiell für Verzögerungen bei Ereignissen, die innerhalb eines Polling-Intervalls auftreten.

-- VERGLEICHSTABELLE --

Auf Basis dieser Gegenüberstellung erscheinen das SiLiCon Context Framework und TUIpist als die beiden geeignetsten Kandidaten für den Einsatz als generisches Framework im hier vorgestellten Anwendungsfall. Das Context Toolkit und Papiermaché scheiden aus verschiedenen Gründen, vorallem aber der mangelnden Flexiblität aus.

Stellt man die beiden verbleibenden Frameworks gegenüber, so sind hinsichtlich der funktionalen Anforderungen keine Vorteile für einen der beiden Kandidaten zu identifizieren. Hinsichtlich der nichtfunktionalen Anforderungen scheint TUIpist hinsichtlich der Skalierbarkeit leichte Vorteile zu haben, da das Nachladen von neuen Instanzen weniger Aufwand versacht als im Fall des SiLiCon Context Framework (lediglich Laden eines Moduls im ersten Fall gegenüber Laden und Einbinden über eine Änderung der Regelbasis im zweiten Fall). Außerdem ist die technologische Basis der Verteilungsarchitektur in TUIpist konzeptionell auf höherer Ebene angesiedelt und mächtiger in der Verwaltung der Applikationsstruktur (unter anderem werden neue Module bei TUIpist automatisch in die Infrastruktur eingebunden, sobald sie angemeldet werden, im SiLiCon Context Framework muss für jeden Kommunikationskanal die IP-Adresse des Empfängers bekannt sein und in die Regelbasis aufgenommen werden).

Hinsichtlich der Effizienz der Kommunikation bietet das SiLiCon Context Framework gegenüber TUIpist aus den oben beschriebenen Gründen (Benachrichtung über Änderungen gegenüber Polling) Vorteile. Für die hier beschriebene Applikation fällt die Entscheidung trotzdem zugunsten TUIpist. Neben der generell unaufwändigeren Konfigurierbarkeit kommt die Struktur von TUIpist einem iterativen Software-Entwicklungsprozess insofern eher entgegen, als dass die Modularisierung von initialen, monolithischen Prototypen (z.B. basierend auf der Struktur des zuvor ausgewählten ReacTIVision-Frameworks) einfacher möglich ist. Dies liegt darin begründet, dass in der modularen und dynamischen Struktur von TUIpist die gesamte Applikationslogik in den Modulen enthalten ist und so Teile aus einer monolithischen Applikation herausgelöst und direkt übernommen werden können, wobei lediglich die Logik der Datenübergabe von direkten Methoden-Aufrufen auf die TUIpist-Routinen zum Zugriff auf den Tuplespace umgearbeitet bzw. erweitert werden muss. Beim korrekten Einsatz des SiLiCon Context Framework wandert wie oben beschrieben ein Teil der Applikationslogik in die Regelbasis, was erhöhten Aufwand bei der iterativen Entwicklung verursacht und außerdem das Zusammenspiel der Applikations-Module unübersichtlicher und schwerer fassbar macht.

1.2 Konzeption und Umsetzung der Hardwarekomponenten

Basierend auf der oben getroffenen Techhnologieentscheidung zugunsten eines optischen, video-basierten Input-Systems für das hier zu erstellende Tabletop Interface wird in diesem Abschnitt das konkrete Hardware-Design beschrieben. Dies umfasst die Beschreibung des Tabletop Interfaces im Überblick und detaillierte Betrachtungen des Token-Designs sowie der Tisch-Oberfläche, die als Input-Kanal dient. Im weiteren werden spezifische Herausforderungen und der jeweilige hier verfolgte Lösungsansatz beschrieben. Nicht Gegenstand dieses Abschnitts sind jene Hardware-Komponenten, die für die Ausgabe von Information eingesetzt werden. Obwohl hier im Überblick angeführt, werden sie detailliert erst in Kapitel XY über die Visualisierung der Modellierungsinformation beschrieben.

1.2.1 Überblick

Das Tabletop Interface ist als Tisch mit einer Oberfläche von 100 cm x 80 cm ausgeführt. Die Höhe beträgt 110 cm. Die wesentlichen Hardwarekomponenten sind die Tischoberfläche, die in semi-transparenten Acrylglas ausgeführt ist und die Bodenplatte, in die die Kamera zum optischen Tracking der Tokens auf der Oberfläche sowie der Videoprojektor zur Ausgabe von Information auf der Oberfläche (Projektion von unten) eingebaut sind (siehe Abbildung 1.2). Der Tisch ist auf allen Seitenflächen mit Platten aus Styropor (MARKENNAME -> KUNSTSTOFF) verkleidet, um im Inneren kontrollierte Umgebungslichtbedingungen für die Bilderfassung mit der Kamera zu schaffen. Die kontrollierten Bedingungen werden durch den Einsatz von vier Beleutungsmodulen gewährleistet, die ebenfalls in der Bodenplatte integriert sind und über den Seitenflächen eingebaute Streuscheiben für eine einheitliche, diffuse Beleuchtung der Tischinnenraums sowie der Oberfläche sorgen.

Am Tisch befindet sich zusätzlich ein Bildschirm, auf dem entsprechend der Anforderungen zur Unterstützung der Modellierung Zusatzinformation ausgegeben wird. Eine zweite Kamera, die am Bildschirm sitzt und der unterstützenden Erfassung von Information dient (zum Beispiel der Registrierung von geschachtelten Token wie oben bereits beschrieben).

Das gesamte System ist so zerlegbar, dass es im Kofferraum eines Mittelklassewagens transportiert werden kann. Es werden dazu die Verkleidungsplatten und Tischbeine entfernt, die Tischplatte kann dann direkt auf die Bodenplatte gesetzt und verbunden werden. Das Volumen reduziert sich dabei auf 100 cm x 80 cm

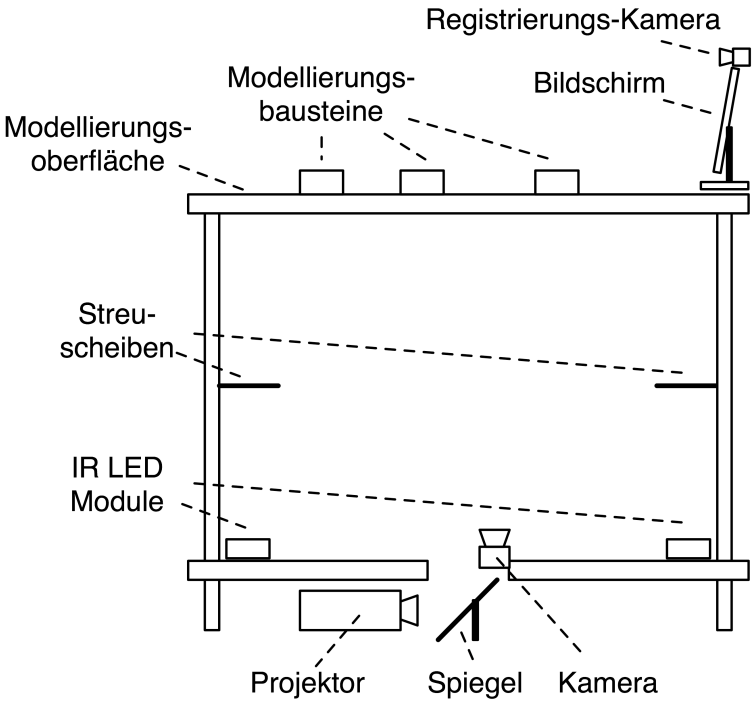

Abbildung 1.2: Überblick über den Aufbau des Tabletop Interfaces

x 20 cm, wobei die Tischbeine, die Verkleidungsplatten und der Videoprojektor separat transportiert werden müssen. Ein Zusammenbau des Systems inklusive Kalibrierung der Eingabe- und Ausgabe-Kanäle ist in etwa 30 Minuten möglich.

1.2.2 Tokens und Input-Werkzeuge

In diesem Abschnitt werden die einzelnen durch die Benutzer manipulierbaren Token-Arten beschrieben. Neben den eigentlichen Modellierungstokens sind dies die Tokens zur Einbettung in Container sowie die Werkzeugtokens, von denen es Ausführungen zur Manipulation des Modells und Tokens zur Auslösung bzw. Kontrolle spezifischer Funktionalitäten gibt.

Modellierungs-Tokens

Die eingesetzten Modellierungs-Tokens sind aus nicht transparentem Acrylglas gefertigt. Ihre Außenmaße betragen in etwa (je nach Form) 10 cm x 6 cm in der Grundfläche und 4 cm in der Höhe. Damit ist einerseits eine ausreichende Größe zur Anbringung der ReacTIVision-Codes gewährleistet, andererseits sind noch klein genug, um in einer Hand gehalten und manipuliert werden zu können. Die Codes werden auf der Unterseite der Tokens angebracht und auch von unten erfasst (siehe nächster Abschnitt), um eine Verdeckung der Codes während der Manipulation zu verhindern (siehe Abbildung 1.3).

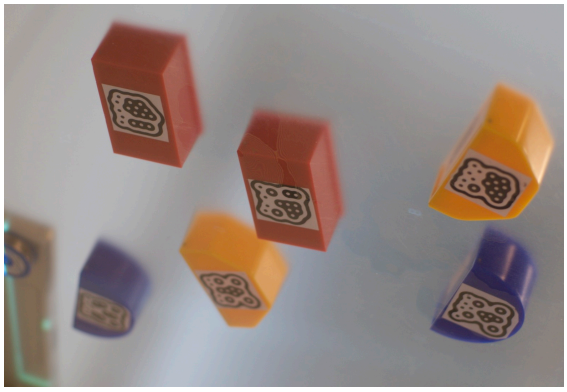

Abbildung 1.3: An Tokens angebrachte ReacTIVision-Codes zur Identifikation

Die Modellierungs-Tokens wurden in drei Ausführungen gefertigt (siehe Abbildung 1.4). Diese unterscheiden sich in Form und Farbe und können während der

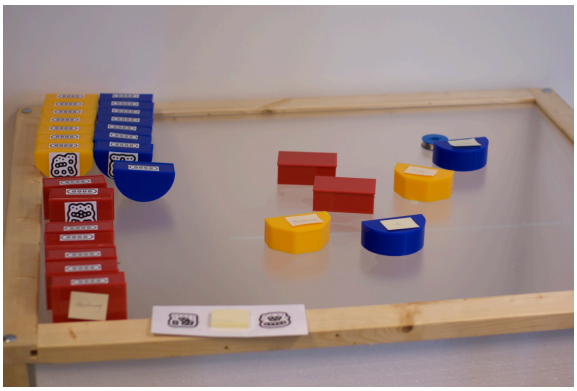

Abbildung 1.4: Arten von Modellierungstokens

Modellierung von den Benutzern frei mit Bedeutung belegt werden. Die Auswahl der Formen und Farben erfolgte inspiriert von den Symbolen gängiger Modellierungsnotationen in Abstimmung mit fertigungstechnischen Einschränkungen. Eine wie in XY (REF von TEI) vorgeschlagene Analyse geeigneter Token-Formen und eine auf diesen Ergebnissen basierende Umsetzung wurde nicht vorgenommen. Die liegt vor allem in der Tatsache begründet, dass sich der Fokus der hier vorgestellten Arbeit im Entwicklungsprozess von einem Werkzeug zur Geschäftsprozessmodellierung (mit vorgegebener Notation) hin zu einem allgemeiner einsetzbaren Werkzeug zur generischen Modellierung konzeptueller Modelle (ohne vorgegebene Notation) entwickelte. Die Auswahl der Token-Formen fiel in die erste Phase, wodurch die Tokens im äußeren Erscheinungsbild an gängige Notationen zur Ablauf-Modellierung angelehnt sind. Die Auswirkung der Token-Form auf die Modellierung scheint aber bei Benutzern ohne Modellierungs-Vorbildung geringen bis keinen Einfluss auf den Modellierungsprozess zu haben (siehe Kapitel XY - Evaluierung).

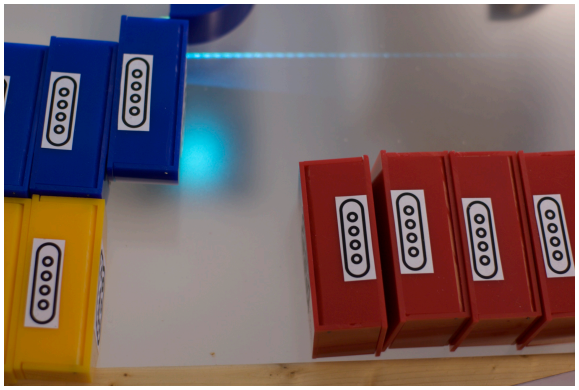

Abbildung 1.5: Rückwand von Container Tokens

Wie in der Beschreibung der Anforderungen gefordert und oben bereits beschrieben sind die Modellierungs-Tokens als Container ausgeführt. Im Abschnitt über die Erkennung des Token-Zustands (REF) wurde beschrieben, das beim Einsatz von optischem Tracking eine Möglichkeit geschaffen werden muss, im Kamerabild zu erkennen, ob ein Token geöffnet ist oder nicht. Um den konsequenten Einsatz eines Erkennungsframeworks (ReacTIVision) zu gewährleisten, wird zu diesem Zweck ein zweiter Code eingesetzt (siehe Abbildung 1.5), der nur dann für die Kamera sichtbar wird, wenn das Token geöffnet ist. Hardwareseitig ist dies so umgesetzt, das die Modellierungstokens einen Öffnungsmechanismus besitzen, dessen Schanier nicht am Deckel sitzt (und damit nur diesen beweglich machen würde), sondern an der Bodenplatte, wodurch sich beim Öffnen eines Containers nicht nur der Deckel sondern auch die Hinterwand des Tokens bewegt (siehe Abbildung 1.6). Die Hinterwand kommt im geöffneten Zustand auf der Oberfläche des Tisches zu liegen, wodurch der auf ihr angebrachte zweite Code für die Kamera sichtbar wird. So kann durch das ansich zur Positionsbestimmung eingesetzte optische Trackingsystem zuverlässig auch den Öffnungs-Zustand der Tokens auf der Oberfläche erkennen.

Einbettbare Tokens

Die einbettbaren Token erlauben wie in Kapitel XY beschrieben die Verschachtelung von Modellen und das Hinzufügen von Zusatzinformation. Sie werden in einem definierten Interaktionsvorgang an Information gebunden und dann in einen Container gelegt. Hinsichtlich des Hardware-Designs sind folgende Anforderungen zu beachten:

- Die Token müssen klein genug sein, um auch mehrere Einheiten ein einem Container unterzubringen.

Abbildung 1.6: Geöffnetes Container Token

- Sie müssen gleichzeitig groß genug sein, um einen ReacTIVision-Code zur eindeutigen Information anzubringen.
- Sie müssen so ausgeführt sein, dass haptisch oder akustisch erkennbar ist, ob in einem geschlossenen Container Tokens eingebettet sind oder nicht (z.B. durch Gewicht oder Geräusche beim Schütteln eines Containers).

Die einbettbaren Tokens wurden aus flexiblen Kunststoffmatten gefertigt und mit einem Metallstück versehen (siehe Abbildung 1.6), das einerseits als Griff dient und andererseits sowohl das Gewicht erhöht und akustisches Feedback beim Schütteln eines Container-Tokens verursacht. Die einbettbaren Tokens wurden exemplarisch in drei Ausführungen gefertigt, je nach Art und Anzahl der einzubettenden Information kann eine Definition weiterer Tokens sinnvoll sein. Folgende Tokens wurden für den aktuellen Entwicklungsstand des Werkzeugs angefertigt:

- Einbettung von Teilmodellen (inhärente Kernfunktionalität des Werkzeugs)
- Einbettung von digitalen Ressourcen bzw. Dateien am lokalen Rechner (Erweiterung zur Verknüpfung des Modells mit den realen digitalen Ressourcen aus dem Arbeitskontext)
- Einbettung von Fotos (Erweiterung zur Verknüpfung des Modells mit dem realen Arbeitskontext, z.B. mittels Fotos von Arbeitsmitteln oder Personen)

Alle Tokens sind auf der Unterseite mit einem ReacTIVision-Code versehen, der sie eindeutig identifiziert. Aufgrund der beschränkten Größe der Tokens ist dieser Code von der Kamera, die die Tischoberfläche erfasst, nur in der Mitte der Oberfläche erfassbar (da dort die geringsten Linsen-Verzerrungen auftreten und der Code somit gerade noch erkannt werden kann). Um etwaige Bedienungsprobleme zu vermeiden, wurde die deshalb die Interaktion mit einbettbaren Tokens (Registrierung, Binden von Information) auf die in Abbildung 1.2 dargestellte zweite

Kamera ("Registrierungs-Kamera") verlegt, die durch die physische Nähe zu den Modellierenden die Codes der einbettbaren Tokens problemlos erfassen kann.

Werkzeug Tokens

Neben den Tokens, die Modellierungsinhalt repräsentieren, wurden auch Tokens angefertigt, der Interaktion mit dem System ansich und der Steuerung der Modellierungsablaufs dienen. Hier sind zwei Arten zu unterscheiden: einerseits gibt es Werkzeuge, die der Manipulation des Modells dienen (z.B. Herstellung von Verbindungen zwischen Tokens, Benennung von Tokens) und andererseits solche, die der Steuerung von modellierungsunterstützender Zusatzfunktionalität dienen (etwa Tokens, die die Rückverfolgung der Modellierungshistorie kontrollieren). Außerdem sind orthogonal zu dieser Klassifizierung noch Tokens zu unterscheiden, die beim Einsatz ein Ereignis auslösen und solche, die die für einen Zustandswechsel des Systems sorgen, solange sie im Einsatz sind (gleich einem Schalter). Im Folgenden werden die einzelnen Werkzeuge beschrieben und den eben definierten Kategorien zugeordnet.

Manipulation des Modells Die bei der Verwendung des Systems wichtigsten Werkzeug-Tokens sind jene, die zur Benennung und Verbindung von Modellierungs-Tokens verwendet werden. Sie sind als auf die Oberfläche aufsetzbare Stäbe konzipiert, die unten eine Standfläche aufweisen, welche auch den Code aufnimmt. Am oberen Ende ist eine Verbreiterung angebracht, die eine stabile Handhabung gewährleistet (siehe Abbildung XY). Bedingt durch die kleine Standfläche kann nur der einfachste ReacTIVision-Code verwendet werden, der lediglich aus einem schwarzen Ring besteht. Das ReacTIVision-System erkennt diesen Ring nicht als regulären Code sondern als "Cursor", also als Zeiger, der ausschließlich eine Position in der Ebene besitzt, aber keine Rotation aufweist. Für die Verwendung zur Markierung von Modellierungs-Tokens zum Zwecke der Benennung oder Verbindung ist dies auch nicht notwendig.

-- Bild der Auswahl-Werkzeuge --

Das Modellierungswerkzeug muss zwischen unterschiedlichen Arten von Verbindungen unterscheiden können, da diese bei der vorgestellten Art der Externalisierung zum Einsatz kommen können. Im Wesentlichen ist zwischen ungerichteten und gerichteten Verbindern zu unterscheiden. Im Gegensatz zu ungerichteten Verbindern weisen gerichtete Verbinder Pfeilspitzen an einem Ende oder an beiden Enden auf, die von den Modellierenden zusätzlich mit Bedeutung belegt werden können. Für die Werkzeug-Tokens bedeutet dies, dass neben den bereits beschriebenen, einfachen Markierungs-Tokens auch solche zum Einsatz kommen, die an-

statt einer runden Grundfläche eine größere, dreieckige Grundfläche aufweisen, die eine Pfeilspitze symbolisiert (siehe Abbildung XY rechts). Technisch werden die beiden Arten von Markierungs-Tokens durch einen zweiten Cursor-Code unterschieden, der an Tokens mit dreieckiger Grundfläche zusätzlich angebracht ist.

Die Markierungs-Tokens sind der Kategorie der ein Ereignis auslösenden Tokens zuzuordnen. Dies bedeutet, dass das System zu jenem Zeitpunkt eine Aktion setzt, zudem das Token auf der Oberfläche aufgesetzt und erkannt wird. Wird ein Markierungs-Token auf der Oberfläche belassen, verursacht es keine weiteren Aktionen bis es entfernt und erneut aufgesetzt wird.

Dem hingegen ist das zweite hier behandelte Token der Kategorie der den Systemzustand beeinflussenden Tokens zuzuordnen. Es hat also solange Auswirkungen auf den Modellierungsablauf, solange es auf der Oberfläche belassen wird. Die konkrete Funktionalität dieses zweiten Tokens ist der Wechsel ein jenen Werkzeugmodus, der ein explizites Entfernen bzw. Löschen von Bestandteilen des Modells, insbesondere Verbindungen, erlaubt. Das Token ist als Radiergummi ausgeführt und codiert so die Metapher des expliziten Herausnehmens von Information aus dem Modell (siehe Abbildung XY). An der Unterseite des Tokens ist eine ReacTIVision-Code angebracht um es gegenüber dem System eindeutig zu identifizieren.

-- Bild des Radierer-Tokens --

Der Einsatz der Radierer-Tokens verändert im System im Wesentlichen die Interpretation des Markierungs-Tokens, die nun anstatt der Herstellung einer Verbindung das Löschen derselben auslösen. Der detaillierte zugehörige Interaktionsablauf ist in Abschnitt 1.3.4 dargelegt.

Ein weiteres Werkzeug im Kontext der Manipulation des Modells ist das Registrierungstoken für die Benennung von Blöcken mittels handgeschriebenen Haftnotizen. Wie in Abschnitt 1.3.2 im Detail ausgeführt, gibt es zwei Modalitäten um ein Modellierungs-Token zu benennen. Einerseits kann der herkömmlich Eingabeweg über die Tastatur gewählt werden, andererseits ist es möglich, die Modellierungs-Tokens mittels aufgeklebten Haftnotizen zu benennen. Um die handgeschriebene Information in die digitale Information übernehmen zu können, wird diese mittels Bildanalyse-Verfahren aus einem Bild extrahiert, das von der oben bereits erwähnten Registrierungskamera (siehe Abbildung 1.2) aufgenommen wird. Die Aufnahme wird durch das erwähnte Registrierungstoken ausgelöst. Dieses fungiert gleichzeitig als Halter für die noch unbeschrifteten Haftnotizen und weist am linken und rechten Rand jeweils einen ReacTIVision-Code auf (siehe Abbildung XY). Diese beiden Codes erlauben es, im Bild die Position der Haftnotiz und damit der zu extrahierenden Information zu identifizieren.

-- Bild des Registrierungstokens --

Das Registrierungstoken ist in die Kategorie der ein Ereignis auslösende Tokens einzuordnen. Seine Verwendung erwies sich im praktischen Einsatz als wenig intuitiv und technisch instabil bzw. zu langsam, so dass wie erwähnt als alternativer Eingabeweg zusätzlich eine Tastatur ins System aufgenommen wurde. Die Interaktionsabläufe sind in beiden Fällen ähnlich und werden im Detail in Abschnitt 1.3.2 dargestellt.

Steuerung von Unterstützungsfunktionen Im Werkzeug wurden entsprechend den Anforderungen auf Kapitel XY eine den Modellierungsablauf unterstützende Funktionen implementiert. Diese werden über dezidierte Werkzeug-Tokens kontrolliert. Der Kernbereich dieser Art von Tokens ist die Steuerung aller Funktionen, die im Kontext der Verfolgbarkeit der Modellierungs-Historie umgesetzt wurden.

Das Snapshot-Token dient der durch die Benutzer ausgelösten Speicherung des aktuellen Systemzustandes. Wie in Kapitel XY beschrieben, verfolgt das System den Modellierungsablauf und speichert selbsttätig stabile Systemzustände (zur Identifikation dieser Zustände siehe Abschnitt 1.3). Zusätzlich können Benutzer mit eben dem hier beschriebenen Token die explizite Aufnahme des aktuellen Modellzustandes veranlassen, wenn er ihnen wesentlich erscheint. Das Token ist als ereignis-auslösendes Element konzipiert und wird durch kurzes Aufsetzen des Tokens auf die Modellierungsoberfläche aktiviert. Physisch ist es als prototypisch als großes rotes Quadrat ausgeführt, an dessen Oberseite eine Griffstück angebracht ist. Auf der Unterseite sitzt wiederum ein ReacTIVision-Code zur Identifikation des Tokens.

Neben der Speicherung von Modellzuständen können diese auch wieder abgerufen und in ihrer zeitlichen Sequenz dargestellt werden. Zur Aktivierung des Historien-Modus und zur Navigation in der Zeitlinie wurde ein weiteres Token eingeführt, das als einziges sowohl der zustandsverändernden als auch der ereignisauslösenden Token-Kategorie zuzuordnen ist. Zustandsverändernd wirkt das Token insofern, als das das Aufsetzen auf die Oberfläche das System in den Historienmodus schaltet und ein Entfernen wiederum die aktuelle Modell-Ansicht wieder herstellt. Durch Rotation des Token im bzw. gegen den Uhrzeigersinn kann nun durch die in ihrer zeitlichen Abfolge angeordneten gespeicherten Modellzustände navigiert werden. Die Rotation löst dabei jeweils nach einem bestimmten zurückgelegten Drehwinkel ein Ereignis aus, das den entsprechend vorhergehenden bzw. nachfolgenden Systemzustand abruft. Um die Rotation möglichst natürlich handhabbar zu machen, ist das Token mit runder Grundfläche etwa mit Faustdurchmesser ausgeführt (siehe Abbildung XY). Auf der Unterseite der Grundfläche ist wiederum der ReacTIVision-Code angebracht.

-- Bild des Zeitglases --

Das letze hier behandelte Token zur Steuerung einer Unterstützungsfunktion ist jenes, dass die in Kapitel XY als Anforderung definierte und in Kapitel XY im Detail beschriebene Wiederherstellungsunterstützung auslöst. Die Verwendung dieses Tokens ist identisch mit jener des oben beschriebenen Snapshot-Tokens. Wie dieses ist es der ereignisauslösenden Kategorie zuzuordnen und aktiviert die Wiederherstellungsunterstützung wenn es bei aktivem Historien-Modus auf der Modellierungsoberfläche aufgesetzt wird. Die eigentliche Wiederherstellungsunterstützung ist dem Output-Kanal zuzuordnen und in ihrem Ablauf deshalb im Detail in Kapitel XY (Abschnitt XY) beschrieben.

1.2.3 Input auf der Tischoberfläche

Nachdem nun die Tokens beschrieben wurden, die den Benutzern unmittelbar zur Interkation mit dem System dienen, liegt der Fokus dieses Abschnitts nun auf der technischen Umsetzung der Erfassung der aktuell auf der Oberfläche sichtbaren ReacTIVision-Codes und damit Tokens.

Wie bereits oben beschrieben, wurde bei der Umsetzung des optischen Tracking-Ansatzes eine Identifikation der Codes von der Unterseite gewählt, um etwaigen Erkennungsproblemen mit Verdeckungen vorzubeugen. Dies bedingt, dass die Modellierungsoberfläche für Kameras möglichst unsichtbar sein muss und den Identifikationsvorgang nicht negativ beeinflussen soll. Grundsätzlich bietet sich dazu der Einsatz einer (Acryl-)Glasplatte an, die keinen Einfluss auf die Erfassung der auf ihr platzierten Codes hat. Der Einsatz einer (Acryl-)Glasplatte bringt jedoch zwei Probleme mit sich. Einerseits kann die Verwendung einer transparenten Oberfläche dazu führen, dass Modellierende durch den Einblick auf den inneren Aufbau des Systems von der Aufgabe abgelenkt bzw. irritiert werden. Andererseits verhindert eine transparente Oberfläche den Einsatz dieser als Rückkanal zur Darstellung von zusätzlicher Information durch einen Videoprojektor. Dies ist im konkreten Fall besonders schwerwiegend, da die generischen Modellierungs-Tokens erst durch die Projektion der zusätzlichen Information explizit bedeutungstragend werden. Deswegen wurde eine semi-transparente Acrylglas-Oberfläche eingesetzt, die einerseits für die Erfassung der Codes ausreichend durchlässig ist, andererseits aber auch das projizierte Licht soweit bricht, dass die darzustellende Information auf der Oberfläche sichtbar ist.

Die Kamera sitzt wie in Abbildung 1.2 schematisch dargestellt in der Mitte der Bodenplatte. Die Kamera ist mit einem Weitwinkel-Objektiv ausgestattet, das es erlaubt, die gesamte Oberfläche zu erfassen, ohne dass der Tisch höher als 100 cm sein muss (Öffnungswinkel etwa 45°). Die Kamera hat eine native Auflösung von 1024 x 768 Bildpunkten und ist über eine Firewire-Schnittstelle

(IEEE 1394 REF!) mit dem Rechner verbunden. Die Kamera kann so bis zu 30 Bilder pro Sekunde an den Rechner senden. Das Weitwinkelobjektiv verursacht Verzerrungen in den Randbereichen, was trotz der Möglichkeit von ReacTIVision, das Bild softwareseitig zu entzerren, dazu führt, dass links und rechts etwa 10 cm der Tischoberfläche nicht zur Idenifikation von Codes verwendet werden können. Dieser Problematik kann aufgrund der beschränkten Kameraauflösung nur durch größere Codes begegnet werden, was durch die beschränkte Größe der Tokens nicht möglich ist. Im Prototypen wurden deshalb die Randbereiche des Systems ausgespart und können zur Ablage von aktuell nicht eingesetzten Tokens verwendet werden.

Illumination und Umgebungslichtabhängigkeit

Optische Tracking-Systeme leiden generell unter ihrer Abhängigkeit von den Umgebungslichtbedingungen. Probleme, die in diesem Zusammenhang auftreten, hängen nicht nur mit der absoluten Stärke der Einstrahlung von Umgebungslicht zusammen (die zu schwach, aber auch zu stark sein kann), sondern auch mit der Änderung der Lichtbedingungen über die Zeit. Gegenstand dieses Abschnitts ist die genauere Erörterung dieser Problematik und die Darlegung von Möglichkeiten ihr zu begegnen.

Zu geringe Umgebungshelligkeit führt zu Erkennungsproblemen, da der Kamera nicht ausreichen Licht und damit Kontrast zur Verfügung steht, um ein Bild zu liefern, in dem die ReacTIVision-Codes zuverlässig identifiziert werden können (dies gilt im Übrigen auch für alle zusätzlichen optischen Identifikations-Ansätze). Das Problem wird zusätzlich verschärft, da zur Aufnahme des Bildes nicht das Spektrum des sichtbaren Lichts verwendet wird, sondern in den Infrarot-Bereich ("near infrared", Wellenlänge etwa 550 nm) ausgewichen wird. Dies ist notwendig, da andernfalls Reflexionen des auf die Oberfläche projizierten Output-Kanals (siehe Abschnitt XY) erfasst werden, was eine Identifikation der Codes unmöglich macht. Die meisten Kunstlicht-Quellen strahlen jedoch nicht ausreichend Licht im verwendeten Infrarot-Bereich ab, um eine ausreichende und gleichmäßige Ausleuchtung der Oberfläche zu gewährleisten. Dieser Problematik wurde dadurch begegnet, dass dezidierte im betreffenden Infrarot-Bereich strahlende Beleuchtungsquellen in den Tisch (konkret in den Ecken der Bodenplatte) eingebaut wurden (siehe Abbildung 1.2). Um eine gleichmäßige, diffuse Ausleuchtung zu erreichen, wurde der Tisch rundum mit weißen Kunststoffplatten mit strukturierter Oberfläche verkleidet, die das vorhandene Licht streuen. Um Reflexionen der Lichtquellen (die nach oben abstrahlen) auf der Oberfläche zu vermeiden, wurden etwa 50 cm über den Lichtquellen Streuscheiben angebracht. Es konnte so eine ausreichend diffuse Beleuchtungssituation geschaffen werden, deren Stärke ausreicht, um die Codes auch

bei vollkommener Dunkelheit in der Umgebung zuverlässig zu erkennen.

Das zweite Problemfeld betrifft zu starke Lichteinstrahung aus der Umgebung. Zu hohe Umgebungshelligkeit führt zu einem Überstrahlen des Kamerabildes in manchen Bereichen, womit in Teilen der Oberfläche keine Codes erkannt werden können. Dies liegt in dem von ReacTIVision eingesetzten adaptiven Code-Extraktions-Algorithmus begründet, der bei schwachen oder guten Beleuchtungsverhältnissen leichte Unterschiede in der Ausleuchtung korrigieren kann, bei starken Unterschieden oder großer Helligkeit jedoch nicht mehr zuverlässig funktioniert. Wie bereits erwähnt, arbeitet die Kamera im Infrarot-Bereich. Dies ist dann problematisch, wenn starke Sonneneinstrahlung vorhanden ist, da diese einen hohen Anteil an Lichts der relevanten Wellenlänge enthält. Auch bestimmte Kunstlichtquellen wie Leuchtstoffröhren, die unmittelbar über dem Tisch angebracht sind, verursachen Probleme. Während punktuelle Helligkeitszonen ("Spotlights") nicht durch die Software korrigiert werden können und vermieden werden müssen, kann eine generell zu hohe Umgebungshelligkeit durch die Kalibrierung der ReacTIVision-Software bzw. der ihr zugrunde liegenden Kamera-Treiber korrigiert werden. Im Konkreten ermöglicht ReacTIVision bei Firewire-Kameras eine Veränderung der Kamerablende (größere Blende - weniger Licht am Sensor) sowie der Signalverstärkung (weniger Verstärkung - dunkleres Bild). Durch Veränderung dieser beiden Parameter kann nahezu jede Beleuchtungssituation korrigiert werden, sofern sie über die Zeit stabil bleibt.

Bei Umgebungslicht-Bedingungen, die sich über die Zeit verändern (z.B. bei Tageslicht, das sich durch Wolkenbewegungen oder tageszeitabhängig verändert), kann die Konfiguration des ReacTIVision-Frameworks nicht sinnvoll eingesetzt werden, da sich nicht automatisiert über eine Software-Schnittstelle vorgenommen werden kann, sondern manuell durchgeführt werden muss. Leichte Veränderung der Helligkeit kann das ReacTIVision-Framework durch den adaptiven Erkennungs-Algorithmus noch selbst korrigieren, bei größeren Veränderungen funktioniert jedoch die Erkennung nicht mehr stabil. Dies äußert sich darin, dass Codes für einige Sekundenbruchteile von der Kamera nicht mehr erfasst werden, was wiederum zu unerwünschten von Framework gemeldeten Ereignissen führt (z.B. dass ein Container-Token verschwunden wäre, obwohl es sich physisch noch auf der Oberfläche befindet). Da diese Erkennungsausfälle im Grenzbereich grade noch möglicher Erkennung zwar regelmäßig auftreten, in ihrer zeitlichen Ausdehung aber eher kurz sind, wurde auf Ebene der Interpretation der vom ReacTIVision-Framework gemeldeten Ereignisse eine Stabilisierungsebene eingezogen, die versucht, Ausfälle und Fehlerkennungen zu identifizieren und dementsprechend nicht an die übergeordneten Software-Ebenen weiterzugeben. Die in dieser Ebene umgesetzten Maßnahmen sind in Abschnitt 1.4.2 beschrieben.

Durch die Kombination aller beschriebenen Maßnahmen ist es möglich, eine sehr

weitgehende Einsetzbarkeit der Werkzeugs unabhängig von den herrschenden Umgebungslichtbedingungen zu gewährleisten. Lediglich bei extremen Verhältnissen wie direkter Einstrahlung von in der Stärke stark variierenden Lichtquellen verhindert einen zuverlässigen Einsatz des Systems.

1.3 Benutzerinteraktion mit dem Werkzeug

Bevor nun die Interpretation der Eingabedaten durch die Software beschrieben wird, muss vorab geklärt sein, auf welche Art und Weise Benutzer mit dem System interagieren können. Bisher wurde geklärt, wie die Eingabe technische funktioniert und welche Elemente den Benutzern zur Interaktion mit dem System zur Verfügung stehen. Im Folgenden die einzelnen Aktionen, die Benutzer grundsätzlich im Kontext des Modellierungsablaufs setzen können, identifiziert und deren konkrete Umsetzung durch das vorgestellte System beschrieben.

Ein grundsätzliches Designziel war es, die Interaktion mit dem System so "natürlich" wie technisch möglich zu gestalten. Unter "natürlich" ist hier zu verstehen, dass

- die Eingabe von Information möglichst ausschließlich über direkte Aktionen der Hände der Benutzer (d.h. ohne Übersetzung in die digitale Welt durch Maus oder Tastatur) gestaltet werden kann, und
- dass diese Aktionen bzw. deren Ablauf nur durch inhaltliche Vorgaben der Modellierungsproblematik, nicht aber durch technische Einschränkungen, vorgegeben sind.

Wie bereits im letzen Abschnitt beschrieben, konnten diese Forderungen aufgrund der Eigenschaften des eingesetzten Tracking-Systems nicht vollständig umgesetzt werden. Wo noch Einschränkungen hinsichtlich der oben genannten Forderungen bestehen, wird in den folgenden Abschnitt darauf hingewiesen und ein möglicher Lösungsansatz skizziert.

1.3.1 Hinzufügen und Verändern von Modellelementen

Um einem Modell Modellelemente hinzuzufügen müssen entsprechende Tokens auf der Modellierungsoberfläche platziert werden. Sofern sich diese im Erfassungsbereich der Kamera befinden, werden sie identifiziert und dem Modell hinzugefügt. Beim erstmaligen Hinzufügen wird das Token dem Benutzern gegenüber mittels seiner Nummer (die durch den ReacTIVision-Code vorgegeben ist) identifiziert. Ist das Token bereits benannt und wird erneut hinzugefügt, wird die zuvor vergebene Benennung geladen und angezeigt.

Beim Erstmaligen Hinzufügen eines Elementtyps (also eines roten, blauen oder gelben Tokens) fragt das System nach der Bedeutung dieses Elementtyps. Diese kann angegebenen werden, kann aber auch unspezifiziert bleiben und zu einem späteren Zeitpunkt hinzugefügt oder verändert werden. Diese Interaktion zu einem späteren Zeitpunkt muss explizit durch ausgelöst werden, in dem ein Token des betreffenden Typs vor die Registrierungskamera gehalten wird.

Um ein Modellelement zu verändern (wobei hier lediglich die Veränderung der Raumparameter wie Position und Rotation gemeint sind), muss es entsprechend an einen anderen Ort verschoben oder gedreht werden. Dabei ist es nicht notwenig, dass das Token ständig von der Kamera erfasst wird. Kurze Aussetzer in der Erfassung, die durch schnelles Verschieben oder Abheben des Tokens und Aufsetzen an einem anderen Ort auf der Oberfläche ausgelöst werden, werden von der Software ignoriert, das Element wird so behandelt, als wäre es nicht verschwunden gewesen (hinsichtlich der Auswirkungen des Verschwindens eines Tokens siehe Abschnitt 1.3.4).

Eine simulaten Erweiterung oder Veränderung des Modells durch mehrere Benutzer ist möglich, die Software schränkt die Anzahl der möglichen zeitgleich ablaufenden Interaktionen nicht ein. Da alle in diesem Abschnitt beschriebenen Interaktionen eindeutig einem Token zugeordnet werden können und dieses zu jedem Zeitpunkt eindeutig identifiziert werden kann, kann das System mit dieser Art von Eingaben umgehen. Die Erfassungsrate liegt in der derzeitigen Implementierung bei etwa 15 Bildern in der Sekunde, woraus sich ein Auswerteintervall von etwa 67 Millisekunden ergibt. In diesem Abstand wird jeweils die gesamte Oberfläche ausgewertet und etwaige Änderungen an die Applikationsschicht weitergemeldet.

1.3.2 Benennen von Modellelementen

Die Benennung von Modellelementen ist eine der wichtigsten Interaktionsformen im Modellierungsprozess. Modellierungselementen wird damit ein Name zugewiesen, der zur Identifikation des Elements gegenüber den Benutzern verwendet wird. Um der Anforderung nach möglichst direkter, unmittelbare Interaktion mit dem System in der realen Welt (d.h. auf der Modellierungsoberfläche) nachzukommen, wurde ein auf der Verwendung von Haftnotizen basierender Interaktionsmodus eingeführt, der zur Benennung von Modellelementen eingesetzt wird. Alternativ kann die Benennung durch Eingabe der Bezeichnung über eine Tastatur erfolgen.

Im Falle der Benennung mittels Haftnotizen wird ein Werkzeugtoken eingesetzt, auf dem die Haftnotizen angebracht sind. Unmittelbar nach der Platzierung eines Elements wird eine neue Haftnotiz beschriftet und mittels der Registrierungskamera erfasst. Während die Haftnotiz nun auf dem Modellelement angebracht werden

kann, wird im System das erfasst Bild ausgewertet, der geschriebene Text wird (als Grafik) extrahiert und dem zuletzt auf der Oberfläche platzierten Token zugeordnet. Soll ein Element nachträglich (umbenannt) werden - wenn also zwischenzeitlich ein weiteres Element platziert wurde - muss das zu benennende Element mit dem Markierungstoken ausgewählt werden. Das Markierungstoken wird dazu in der Nähe des Elements auf die Oberfläche gesetzt. Visuelles Feedback auf der Oberfläche signalisiert eine erfolgreiche Auswahl. In der Folge kann die neue Benennung mittels des Registrierungstokens erfasst und am Element angebracht werden. Die Benennung mittels Haftnotizen ist insofern problematisch, als dass die Extraktion der handschriftlichen Benennung nur unzuverlässig funktioniert. Dies liegt vor allem an zu geringem Kontrast zwischen Haftnotiz und Schrift (bei schlechten Lichtverhältnissen) und einer zu geringen Auflösung der Registrierungskamera. Außerdem werden keine Schrifterkennungs-Algorithmen eingesetzt, wodurch der Schriftzug für das System immer lediglich eine Menge von Bildpunken ohne weitere Bedeutung bleibt.

Um diese Nachteile der Benennung mit Haftnotizen zu kompensieren, wurde ein alternativer Interaktionsmodus zur Benennung von Modellelementen eingeführt. Dieser läuft grundsätzlich gleich ab, setzt jedoch anstatt der Haftnotizen eine Tastatur ein, über die die Benutzer die Benennung dem System bekannt geben. Die eingebene Information wird dann auf der Oberfläche unmittelbar unterhalb des Elements angezeigt (siehe Kapitel XY). Die Auswahl des zu benennenden Elements erfolgt wiederum durch das Markierungstoken. Die Vorteile dieses Systems liegen in der geringen Fehlerrate und der hohen Geschwindigkeit der Eingabe (etwa Faktor 10 schneller als die Eingabe mittels Haftnotiz). Der Nachteil dieses Ansatzes ist der exklusive Zugang zum Eingabemedium, der Tastatur. Während es beim Einsatz von Haftnotizen grundsätzlich möglich ist, diese simultan zu erstellen (auch wenn die Registrierung sequentiell erfolgen muss), kann bei der Verwendung der Tastatur zu einem Zeitpunkt immer nur ein Element bearbeitet werden.

Grundsätzlich sind die beiden Interaktionsmodi zur Bennenung von Elementen als Ergänzung zueinander zu sehen. Sie können grundsätzlich gleichzeitig eingesetzt werden und bieten den Benutzern die Wahlmöglichkeit zwischen einer langsameren, dafür aber physisch vorhandenen oder einer schnelleren, dafür nur über die Ausgabekanäle des Systems sichtbaren Benennung. Die Erkennungsprobleme der Haftnotizen-Variante (die die Einführung des zweiten Interaktionsmodus ursprünglich ausgelöst hatte) können durch eine Verbesserung der technischen Rahmenbedingung ausgemerzt werden, wozu aber bei der Erstellung des hier vorgestellten Prototypen keine Ressourcen vorhanden waren. Im Falle einer Verbesserung könnten den Benutzern zwei gleichwertige Eingabekanäle zur Verfügung gestellt werden. Vorerst bietet der Einsatz der Tastatur jedoch nicht vernachlässigbare Vorteile hinsichtlich der Zuverlässigkeit und der Genauigkeit der Erfassung von Benennungen.

1.3.3 Verbinden von Modellelementen

Das Verbinden von Modellelementen ist neben dem Hinzufügen und Bennenen der dritte wesentliche Interaktionsmodus der zum Aufbau eines Modells benötigt wird. Verbindungen werden nicht physisch gelegt sondern mit Werkzeugtokens erstellt und dann über die Outputkanäle visuell dargestellt. Die Entscheidung gegen eine physische Repräsentation der Verbindungen fiel zugunsten einer leichteren Veränderbarkeit des gesamten Modells - das Verschieben eines Elements verursacht so keinen zusätzlichen Aufwand, wohingegen physische Verbinder nachgeführt werden müssten.

Die Erstellung von Verbindern kann wir die Benennung mit zwei unterschiedlichen Interaktionsmodi erfolgen, die sich aber ihrer Ausdrucksmächtigkeit bzw. Flexibliät unterscheiden. Der flexiblere Interaktionsmodus ist jener, der unter Einsatz von zwei Markierungstokens durchgeführt wird. Dabei wird jeweils ein Markierungstoken neben einem der beiden zu verbindenden Modellelemente platziert, wodurch diese ausgewählt und anschließend verbunden werden. Das Verbindungskriterium ist, dass die beiden Modellelemente innerhalb von 3 Sekunden nacheinander ausgewählt werden. Dies erlaubt sowohl eine gleichzeitige Platzierung von zwei Markierungstokens als auch die Arbeit mit nur einem Token, das rasch hintereinander neben die zu verbindenden Elemente gesetzt wird. Die Arbeit mit den herkömmlichen Markierungstokens erzeugt ungerichtete Verbindugen. Werden gerichtete Verbindungen benötigt (deren Richtung mit einer oder im Falle einer bidirektionalen Verbindung mit zwei Pfeilspitzen angezeigt wird), müssen spezielle Markierungstokens verwendet werden, deren Grundfläche die Form einer Pfeilspitze hat und die neben dem entsprechend zu markierenden Modellelemente platziert werden müssen.

Der alternative Interaktionsmodus zur Herstellung einer Verbindung kommt ohne Werkzeugtokens aus und basiert ausschließlich auf der Manipulation der Modellelemente. Werden zwei Modellelement an ihren Längsseiten zusammengefügt, so erstellt das System zwischen diesen beiden Elementen eine Verbindung. Im Interaktionsablauf bedeutet dies, dass ein Benutzer die beiden zu verbindenen Elemente ergreift und diese auf der Oberfläche von ihrer ursprünglichen Position aus kurz zusammenführt um sie danach sofort wieder an ihre Originalposition zurück zu stellen. Die so erstellten Verbindungen sind immer ungerichtet und werden nach der Erstellung gleich behandelt wie mit den Markierungstokens erstellte Verbindungen. Durch die Vermeidung der Verwendung von Werkzeugtokens ist dieser Weg zur Herstellung von Verbindungen schneller auszuführen als der zuvor beschriebene (etwa um Faktor 5). Problematisch wird diese Art der Interaktion dann, wenn die Modellierungsoberfläche bereits sehr voll ist, so dass nur noch wenig Platz zur Zusammenführung von zwei Elementen vorhanden ist. Auch bei Bedarf nach

gerichteten Verbindern muss auf Markierungstokens zurückgegriffen werden.

Markierungen können unabhängig von ihrem Erstellungsweg auch benannt werden. Da sie keine physische Repräsentation besitzen, kann der Bennungsmodus mit Haftnotizen nicht verwendet werden. Die Eingabe von Bezeichnungen erfolgt demnach ausschließlich über die Tastatur. Zur Benennung einer Verbindung muss unmittelbar nach deren Erstellung die Bezeichnung eingegeben werden, diese wird dann der zuletzt erstellten Verbindung zugewiesen. Um eine Bezeichnung zu ändern muss in der momentanen Implementierung die Verbindung gelöscht, neu hergestellt und wiederum benannt werden.

1.3.4 Löschen von Elementen und Verbindungen

Beim Löschen muss zwischen dem Entfernen von Elementen und Verbindungen unterschieden werden. Beide Modellaskpete sind nicht nur hinsichtlich der Interaktion mit dem System unterschiedlich zu behnadeln, es besteht vorallem eine Existenzbeziehung zwischen Elementen und Verbindungen, d.h. dass Verbindungen nicht ohne ihre als Endpunkte dienenden Elemente existieren können und beim Entfernen eines Elements ebenfalls gelöscht werden müssen.

Das Löschen eines Modellelements aus dem aktuellen Modell erfolgt durch Entfernen der zugehörigen Tokens auf der Oberfläche. Hier muss zwischen Entfernungsvorgängen zu unterscheiden, die vom Benutzer intendiert sind und solchen, die durch die Manipulation des Modells unabsichtlich und kurzzeitig auftreten. Nur in ersterem Fall dürfen die Verbindungen, die an einem Element hängen, ebenfalls gelöscht werden. Die Unterschiedung zwischen den beiden Fällen wurde durch die Einführung eines Zeitintervall von 5 Sekunden gelöst, nachdem die betroffenen Verbinder ebenfalls entfernt werden. Wird also ein Modellelement nur an eine andere Stelle verschoben (und verschwindet so kurzzeitig aus dem Blickfeld der Kamera), bleiben die Verbindungen erhalten. Wird das Element zur Seite gestellt, wird es und seine abhängigen Verbindungen zwar auf den Ausgabekanälen nicht mehr dargestellt, eine wirkliches Löschen erfolgt aber erst nach 5 Sekunden. Wird das Element innerhalb dieser Zeitspanne wieder auf der Oberfläche platziert, wird es und seine abhändigen Verbindungen so behandelt, als ob es nicht verschwunden gewesen wäre.

Das Löschen von Verbindern ist duch den Einsatz der Radierer-Tokens möglich, dass das System in den Lösch-Zustand schaltet sobald es auf der Oberfläche platziert wird (und diesen wieder deaktiviert, wenn es entfernt wird). Im Lösch-Modus wird der Einsatz von Markierungstokens alternativ interpretiert. Anstatt der Herstellung einer Verbindung wird bei Markierung von zwei Modellelementen eine eventuell zwischen diesen vorhandene Verbindung endgültig entfernt. Dabei ist

keine Unterscheidung zwischen gerichteten und ungerichteten Verbindungen notwendig, es können durchgängig die einfachen Markierungstokens eingesetzt werden.

1.3.5 Einbettung von Zusatzinformation

1.3.6 Kontrolle der Modellierungshistorie

1.4 Erfassung der Benutzerinteraktion durch Software

Das Software-System, das die Benutzereingaben von der Hardwareschnittstelle bis hin zur abstrahierten Modellierungsinformation aufbereitet, ist Gegenstand dieses Abschnitts. Die darüber hinausgehenden Komponenten sind Gegenstand der Kapitel X und Y. Die hier behandelten Software-Komponenten sind

- jene Komponente, die die Rohdaten von ReacTIVision empfängt und sie hinsichtlich der Modellierungsinformation auswertet und interpretiert,
- die Komponente, die für die Stabilisierung der Erkennungsleistung sorgt, wenn Fehlerkennung oder Erkennungsausfälle auftreten, sowie
- jene Komponente, die für das Tracking des Modellzustandes und damit für die Nachvollziehbarkeit des Modellierungsablaufs zuständig ist.

Wie in Abschnitt 1.1.3 ausgeführt, soll zur Verknüpfung dieser Komponenten das TUIpist-Framework zum Einsatz kommen. Im vorliegenden Prototypen wurde das System jedoch zwar modular aber ohne Einsatz des TUIpist-Frameworks implementiert. Eine erste Umsetzung der Architektur auf die TUIpist-Strukturen liegt vor, ist jedoch noch nicht im produktiven Einsatz. An dieser Stelle wird deshalb der Aufbau der Komponenten in ihrer derzeitigen Implementierung beschrieben, da diese auch bei der Evaluierung des Werkzeugs eingesetzt wurde. Erste Umsetzungserfahrungen deuten darauf hin, dass der Aufwand zur Einbindung von TUIpist sich wie erwartet in Grenzen hält.

1.4.1 Interpretation der Rohdaten

Das ReacTIVision-Framework liefert wie in Abschnitt 1.1.2 beschrieben die aus dem Kamerabild extrahierten Informationen bezüglich der erkennbaren Codes in einem propritären Nachrichtenformat über eine UDP-Schnittstelle an. Für jeden erkennbaren Code wird eine separate Nachricht gesandt, sobald sich zumindest ein Parameter (also Position oder Rotation) verändert. Diese Nachrichten tref-

fen über die UDP-Schnittstelle bei einem von ReacTIVision bereitgestellten Java-Objekt ein, das die Auswertung der Nachricht übernimmt und über einen Callback-Mechanismus (REF auf Entwurfsmuster, das hier eingesetzt wurde) vom Anwendungsentwickler implementierte Methoden aufruft, die auf das jeweilige Ereignis reagieren. Die Ereignisse, die auftreten können, beschränken sich auf das Erscheinen eines Tokens, die Änderung der Parameter eines Tokens und das Verschwinden eines Tokens sowie die gleichen drei Ereignisse für die Behandlung eines Cursors (wie das Markierungs-Werkzeugtoken). Dazu werden jeweils die relevanten Parameter zur Verfügung gestellt (im Wesentlichen die ID des Tokens, seine Position, seine Rotation und einige weitere abgeleitete Parameter, die in der konkreten Anwendung nicht eingesetzt werden).

In den betreffenden Methoden wird im ersten Schritt ausgewertet, ob das identifizierte Token ein Modellierungs-Element oder ein Werkzeug ist. Im Falle eines Modellierungs-Elements wird im nächsten Schritt festgestellt, um welche Art von Token es sich handelt und ob es im aktuellen Modellierungsvorgang bereits im Einsatz war (ob z.B. bereits Information über die Benennung des Tokens oder dessen Inhalt vorhanden ist). Diese Information wird ggf. geladen und den weiterverarbeitenden Komponenten zu Verfügung gestellt. Im Falle eines Werkzeug-Tokens wird diese identifiziert und an die für die Behandlung zuständige Methode weitergeleitet. Diese Methoden schalten das System je nach Art des eingesetzten Tokens in einen alternativen Zustand oder lösen eine Aktion auf den übergeordneten Ebenen aus.

1.4.2 Stabilisierung der Erkennungsleistung

1.4.3 Tracking des Modellzustandes

Literaturverzeichnis

Beer, W., Christian, V., Ferscha, A., and Mehrmann, L. (2003). Modeling Context-aware Behavior by Interpreted ECA Rules. In *Proceedings of the International Conference on Parallel and Distributed Computing (EUROPAR '03)*, volume 2790 of *LNCS*, pages 1064--1073. Springer.

Dey, A. K., Salber, D., and Abowd, G. D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction (HCI) Journal*, 16(2-4):97--166.

Fitzmaurice, G. (1996). *Graspable User Interfaces*. Phd-thesis, University of Toronto.

Fitzmaurice, G., Ishii, H., and Buxton, W. (1995). Bricks: laying the foundations for graspable user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, pages 442--449. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA.

Fjeld, M. (2001). *Designing for tangible interaction*. PhD thesis, Swiss Federal Institute of Technology.

Fjuk, A., Nurminen, M., and Smørdal, O. (1997). Taking Articulation Work Seriously: An Activity Theoretical Approach. Technical Report TUCS TR 120, Turku Centre for Computer Science.

Fujimura, J. (1987). Constructing 'Do-Able' Problems in Cancer Research: Articulating Alignment. *Social Studies of Science*, 17(2):257--293.

Furtmüller, F. (2007). Implementierung eines Frameworks für berührbare Benutzungsschnittstellen. Master's thesis, University of Linz.

Furtmüller, F. and Oppl, S. (2007). A Tuple-Space based Middleware for Collaborative Tangible User Interfaces. In *Proceedings of WETICE '07*. IEEE Press.

Gerson, E. and Star, S. (1986). Analyzing due process in the workplace. *ACM Transactions on Information Systems (TOIS)*, 4(3):257--270.

Hampson, I. and Junor, A. (2005). Invisible work, invisible skills: interactive customer service as articulation work. *New Technology, Work & Employment*, 20(2):166 -- 181.

Hanke, U. (2006). *Externale Modellbildung als Hilfe bei der Informationsverarbeitung und beim Lernen*. PhD thesis, University of Freiburg.

Hughes, F. (1971). *The Sociological Eye*. Aldine de Gruyter.

Ishii, H. and Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, pages 234--241. ACM Press New York, NY, USA.

Nonaka, I. and Takeuchi, H. (1995). *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press.

Oppl, S. (2004). Context-aware Group-Interaction. Master's thesis, University of Linz, Department for Pervasive Computing.

Patten, J., Ishii, H., Hines, J., and Pangaro, G. (2001). Sensetable: a wireless object tracking platform for tangible interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 2001 (CHI '01)*.

Rumelhart, D. and Norman, D. (1978). Accretion, tuning, and restructuring: Three modes of learning. In Cotton, J. and Klatzky, R., editors, *Semantic factors in cognition*, pages 37--53. Erlbaum, Hillsdale, N.J.

Schmidt, K. (1990). Analysis of Cooperative Work. A Conceptual Framework. Technical Report Risø-M-2890, Risø National Laboratory.

Seel, N. M. (1991). *Weltwissen und mentale Modelle*. Hogrefe, Göttingen u.a.

Semmer, N. and Udris, I. (2004). Bedeutung und Wirkung von Arbeit. In Schuler, H., editor, *Lehrbuch Organisationspsychologie*, pages 157--195. Huber, Bern, 3rd edition.

Strauss, A. (1985). Work and the Division of Labor. *The Sociological Quarterly*, 26(1):1--19.

Strauss, A. (1988). The Articulation of Project Work: An Organizational Process. *The Sociological Quarterly*, 29(2):163--178.

Strauss, A. (1993). *Continual Permutations of Action*. Aldine de Gruyter, New York.

Index