BERUFSAKADEMIE MANNHEIM
university of cooperative education | staatliche studienakademie

Fachrichtung Informationstechnik

-- Studienarbeit --

ENTWURF UND REALISIERUNG EINER KONTAKTLOSEN MAUS ALS EINGABEGERÄT

Betreuer:
Prof. Joachim Schmidt

Mannheim, den 03.06.2005

Autor:	Christian Hasselbach	Christian Siegl
Matrikelnummer:	141056	148858
Kurs:	TIT02BNS	TIT02BNS

Eidesstattliche Erklärung

Hiermit versichere ich an Eides Statt, dass ich meine Studienarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutz habe.

Mannheim, den 03.06.2005

Christian Hasselbach Christian Siegl

ENTWURF UND REALISIERUNG EINER KONTAKTLOSEN MAUS ALS EINGABEGERÄT

Autor:	Christian Hasselbach	Christian Siegl
Matrikelnummer:	141056	148858
Kurs:	TIT02BNS	TIT02BNS

Ziel dieser Studienarbeit ist es, eine aktuelle Übersicht über Forschungen und Entwicklungen auf dem Gebiet der kontaktlosen Maus vorzustellen und eines dieser Konzepte selbstständig zu realisieren. Zunächst werden hierfür die unterschiedlichen Forschungen, wie Auswertung von Sensoren, Kameradaten und Gehirnströme aufgezeigt und erläutert. Diese werden anschließend bewertet und ein System herausgefiltert, welches interessante und realisierbare Ansätze einer Implementierung bietet. Innerhalb dieser Studienarbeit wird auf die Auswertung von Kameradaten zurückgegriffen, die mit einer möglichst preiswerten Kamera umgesetzt wird. Hier ergeben sich folgende Teilaspekte für eine eigenständige Realisierung:

- Empfangen des Kamerabildes,
- Auswertung der Kameradaten und finden eines geeigneten Punktes,
- Übertragen dieses Punktes auf den Bildschirm und
- Tests und Testergebnisse.

Besonderer Wert wird bei der Ausarbeitung auf die verschiedenen Möglichkeiten der Auswertung von Kameradaten gelegt und anschließend der *Optical-Flow*-Algorithmus angewandt. Bei der Übertragung des Zielpunktes auf den Bildschirm werden weiterhin mehrere Positionierungsverfahren aufgezeigt, beispielsweise relative und absolute Positionierung, und ihre Implementierung dargestellt. Die abschließenden Tests und Testergebnisse zeigen die aufgetretenen Probleme auf und die entsprechend gefundenen Lösungen sowie weitere Punkte, die für zukünftige Projekte dieser Art relevant sind.

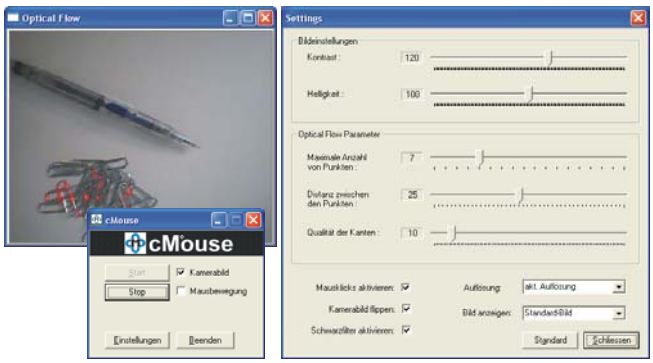

Abb. 1: Oberfläche des implementierten Systems

DESIGN AND IMPLEMENTATION OF A CONTACTLESS MOUSE AS INPUT DEVICE

Author:	Christian Hasselbach	Christian Siegl
Matriculation number:	141056	148858
Course:	TIT02BNS	TIT02BNS

This student research project aims to present an overview of developments in the field of contactless mice and to realize one of these concepts single-handedly. To begin with, different aspects of current research are presented such as interpretation of sensor and camera data as well as brain waves. Afterwards, these are evaluated and one of the methods is selected for implementation. The selection is based on interest and feasibility. Within this student research project the evaluation of camera data will be used. The implementation is divided into the following components:

- Capturing image data,
- Evaluating the image data and finding a suitable tee,
- Mapping the tee to the screen and
- Tests and test results

Particular interest during the project is put on the various methods of interpreting image data. Specifically, the optical-flow algorithm will be used later on. To map the tee to the screen, the advantages and disadvantages of relative and absolute positioning are compared to each other on the basis of their implementation. The final tests and there results will show problems encountered during the project. This paper will then provide solutions to these problems as well as further aspects with relevance for future projects.

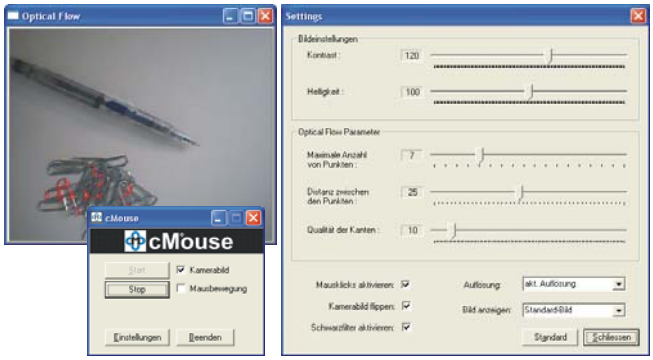

Figure 1: Interface of the realised system

Vorwort

Die Computermaus hat sich im Laufe der Zeit zu einem der wichtigsten Interaktionsmöglichkeiten zwischen Mensch und Maschine entwickelt. Zur Steuerung des Computers muss man heutzutage immer noch eine Maus in die Hand nehmen. Die Technik konnte zwar im Laufe der Jahre verbessert werden, von anfänglich mechanischen bis heute zu optischen Mäusen. Trotz der Vorteile einer Maus den Computer mit einfachen Schritten zu bedienen, gehen auch Nachteile, vor allem im gesundheitlichen Bereich, mit der intensiven Nutzung der Maus einher. Laut der Ursachen für das RSI-Syndrom [WE05d] oder auch Sekretärinnenkrankheit, wird durch das häufige und zu lange verwenden der Maus der Bewegungsapparat des Menschen im Bereich der Hand, der Arme und der Schulter wesentlich beeinträchtigt. Wenn es also eine Möglichkeit gibt, die Schnittstelle zwischen Mensch und Maschine zu verbessern, dann ist die Weiterentwicklung der Maus sicherlich ein Thema, womit man sich in der nächsten Zeit beschäftigen sollte. Mit der Forschung an dieser Problematik sind bereits einige Projekte im Gange, dennoch steht man mit diesen Entwicklungen meist noch am Anfang. Es ist auch noch nicht sicher, in wie weit ein kontaktloses System das bisherige ablösen kann. Hier muss darauf geachtet werden, dass die Ergonomie der kontaktlosen Maus sich im Verhältnis zum bestehenden System nicht verschlechtert. Andererseits braucht man für eine Maus immer noch Platz auf dem Schreibtisch und Kontakt zu einer Unterlage, auch wenn sich das durch die Entwicklung der optischen Mäuse verbessert hat. Sicherlich könnte dieser Platz anders verwendet werden. An diesem Punkt greift das Thema dieser Studienarbeit an. Anfangs werden die bestehenden Systeme einer „kontaktlosen Maus“, wie z.B. die Sensoransteuerung, das Auswerten von Kameradaten oder die Steuerung über das Gehirn, beschrieben und analysiert, die Vor- und Nachteile aufgezeigt und schließlich miteinander verglichen. Auf der Basis der Auswertung wird anschließend das unter Berücksichtigung der gegebenen Randbedingungen beste Verfahren herausgegriffen und als Grundidee verwendet, um basierend darauf ein eigene Realisierung und Implementierung für eine kontaktlose Maus vorzustellen. Der Anstoß für diese Arbeit war in den schon benannten Problematiken der Maus, wie Platz, Ergonomie etc. zu sehen. Dieses Thema enthält auch ein großes Potenzial und ist zukunftsträchtig, denn die Entwicklung der Maus in diese Richtung wird weitergehen. Ziel dieser Arbeit ist es ein System zu entwickeln, welches ein mögliches Konzept einer kontaktlosen Maus realisiert. Hierfür wird dabei auf die Entwicklung der Eingabegeräte, wie Maus und Tastatur eingegangen.

Inhaltsverzeichnis

Abkürzungsverzeichnis

API ..... Application Programming Interface
ASCII ..... American Standard Code for Information Interchange
BCI ..... Brain-Computer-Interface
CPU ..... Central Processing Unit
DLR ..... Deutsches Zentrum für Luft- und Raumfahrt e.V.
FIRST ..... Fraunhofer Institut für Rechnerarchitektur und Softwaretechnik
FZ ..... Forschungszentrum
GUI ..... Graphical User Interface
LED ..... Light Emitting Diode
MFC ..... Microsoft Foundation Classes
RSI ..... Repetitive Strain Injury
TEG ..... Technologie-Entwicklungsgruppe (Fraunhofer)
USB ..... Universal Serial Bus
VPL ..... Visual Programming Language

Abbildungsverzeichnis

Tabellenverzeichnis:

1 Vorbetrachtung

Die Schnittstelle zwischen Mensch und Computer hat sich in den letzten 50 Jahren erheblich verändert. Anfänglich, etwa bis Mitte der 1950er-Jahre, wurden Computer durch den Einsatz von Lochkarten noch mit Daten buchstäblich „gefüttert". Diese Art und Weise der Programmierung und Programmausführung war mehr als umständlich. Anschließend, etwa bis Mitte der 1960er-Jahre, verwendete man bereits Magnetbänder, -trommeln oder -platten, um mit der aus Transistoren aufgebauten Rechenmaschine zu kommunizieren bzw. Daten auszutauschen. Ab Mitte der 1970er-Jahre haben Computer sich deutlich verbessert, sie bestanden aus integrierten Schaltkreisen, so genannten Computerchips und man verwendeten erstmals ein Betriebssystem, welches den Speicher sowie die Eingabe und Ausgabe steuert. Bis heute hat man diesen Aufbau beibehalten, die einzigen Veränderungen zu der in der Mitte der 1970er-Jahre entwickelten Computergeneration sind in der Größe sowie der Ansteuerung der Hardware zu sehen. Dies bedeutet man benutzt heute Speichereinheiten anstelle von Relais, auf die sehr schnell und direkt zugegriffen werden kann, außerdem wird die Hardware über die Software angesteuert. [BIB02]

Das ab Mitte der 1970er-Jahre entwickelte Ein- und Ausgabesystem war eine sehr innovative Entwicklung, die den Umgang mit einem Computer revolutionierte. Dem Menschen war die Kommunikation mit dem Computer sehr vereinfacht worden, so konnte man nun Zeichen oder Befehle anhand einer Tastatur direkt in den Computer eingeben. Die Tastatur war dabei das erste Eingabegerät, was von den Ingenieuren entwickelt wurde. Bis heute verwendet man Tastaturen zur Eingabe bestimmter Befehle oder aber auch im Zusammenhang mit einem Schreibprogramm als Schreibmaschine. Ungefähr zur gleichen Zeit wurde an der Entwicklung einer Maus gearbeitet[1], mit der Maus konnten aber keine direkten Befehle eingeben werden, sondern man konnte einen Mauscursor zur Markierung von Text benutzen oder aber auch bestimmte Objekte auf dem Bildschirm auswählen. Durch die Verwendung von grafischen Oberflächen, besonders bei der Steuerung von modernen Betriebssystemen hat die Maus ihre Vorzüge gezeigt, man kann nun Ereignisse auslösen, indem man auf ein Symbol (Icon) klickt und so ein Programm startet. Diese Art der Programmsteuerung und Bedienung des Betriebssystems hat die Maus zu einem sehr beliebten Eingabegerät gemacht, ohne derer die Steuerung eines Computers für eine Vielzahl von Benutzern kaum noch möglich ist. [BIB02]

[1] Erster Entwurf einer Computermaus von D. Engelbart (1965) [IS05]

2 Theoretische Grundlagen der Eingabegeräte, Beschleunigungssensoren sowie der kameraunterstützten Objekterkennung

In diesem Kapitel werden alle notwenigen Grundlagen erklärt und zusammenfassend beschrieben, die für die Realisierung einer kontaktlosen Maus notwendig sind. Dabei wird zunächst auf den Aufbau von den Eingabesystemen Maus und Tastatur eingegangen, um deren grundlegende Arbeitsweise zu verstehen. Weiterhin werden der prinzipielle Aufbau und die physikalische Funktion von Beschleunigungssensoren erklärt, da diese zur Realisierung in Betracht gezogen wurden. Im letzten Bereich werden einige mögliche Algorithmen vorgestellt, die für die Auswertung von Kameradaten und dort speziell zum Finden von Objekten herangezogen werden können.

2.1 Die Tastatur

Die Tastatur (Keyboard) ist das am häufigsten verwendete Eingabegerät für einen Computer, mit dem die Daten und Steuerbefehle eingegeben werden. Eine Tastatur besteht dabei aus mehreren Gruppen von Tasten, Kontrollleuchten sowie einem Gehäuse mit Anschlusskabel (heutzutage können die Signale auch über Funk, Infrarot oder Bluetooth übertragen werden). Eine Tastatur funktioniert nach ganz einfachen Prinzipien, durch das Betätigen einer Taste wird ein Schalter und somit der Stromkreis geschlossen. Ein wenig anders funktionieren die Folientastaturen, deren elektrische Signale werden anschließend an einen Controller gesendet oder aber auf die Systemplatine weitergeleitet. Dieser Controller wandelt die empfangenen Signale um, so dass eine Kompatibilität zu anderen Tastaturmodellen bzw. Tastaturvarianten gewährleistet ist. Die Signale, die der Computer dann empfängt, sind in der Regel im ASCII-Code kodiert. Anhand dieses Codes können den Signalen genaue Zeichen zugeordnet werden. Somit ist es einfach möglich, unterschiedliche Tastaturenlayouts (z.B. für eine andere Sprache) zu verwenden, ohne den Aufbau einer Tastatur zu verändern. Es können außerdem nicht nur Befehle übertragen werden, sondern auch ganze Bedienungsschritte, mit denen dann Programme ausgeführt werden können. Damit dies möglich ist, werden so genannte Tastenkombinationen oder aber auch Hotkeys verwendet.

Eine Tastatur untergliedert sich in mehrere Blöcke, die nach unterschiedlichen Funktionen angeordnet sind. Es gibt folgende Funktionsblöcke:

Alphanumerischer Block oder Schreibmaschinenfeld ähneln einer Schreibmaschinentastatur. In Deutschland verwendet man im Normalfall die QWERTZ-Tastatur (entsprechend der DIN-Norm 2137, Teil 1), in angelsächsischen Ländern die QWERTY-Tastatur.

Numerischer Block oder Zehner- bzw. Ziffernblock ist für die Eingabe von Zahlen, Rechenoperatoren gedacht. Die Tasten sind gewöhnlich doppelt belegt. Man kann sie nach Umschaltung (mit der Num-Taste) auch als Cursor-Tasten benutzen.
Funktionstasten. Hierzu zählen meist zwölf frei belegbare Tasten, die F-Tasten (F1-F12), die über dem alphanumerischen Block angeordnet sind, andererseits fest belegte Tasten (»Esc«, »Entf« usw.). Hinzu kommen Umschalttasten wie »Alt«, »Strg«, »Alt Gr« usw. (gilt vorwiegend nur für Tastaturen für Büro- und Heimarbeiten, in der Industrie werden auch andere Funktionen und Tasten benutzt).
Cursor- oder Richtungstasten zum Bewegen der Einfügemarke. Sie stehen als vier nebeneinander liegende, einzelne Tasten zur Verfügung (die Tasten Bild nach oben, Bild nach unten, Pos 1 und Ende dienen auch zum Navigieren in Dokumenten, zählen aber nicht zu den Cursor-, sondern zu den Funktionstasten).

Daneben unterscheidet man folgende Tastaturtypen:

Die *PC-XT-Tastatur* mit 83 bis 86 Tasten, meist ohne Kontrollleuchten. Bei dieser veralteten Tastatur befanden sich die Funktionstasten links neben dem alphanumerischen Block.
Die *Multifunktions-Tastatur (MF)* mit 101 oder 102 Tasten. Die Funktionstasten dieser früher sehr weit verbreiteten und heute noch z. T. verwendeten Tastatur befinden sich über dem alphanumerischen Block.
Die *Windows(95)-Tastatur*. Sie stellt mit drei Zusatztasten für Startmenü (doppelt vorhanden) und Kontextmenü eine erweiterte Version der MF-Tastatur dar und löst zunehmend die MF-Tastatur ab.

Der Aufbau von Tastaturen kann weiterhin in zwei Kategorien eingestuft werden, entweder als separat zu erhaltenes Gerät oder als Einbaugerät. Die separat zu erhaltenden Tastaturen findet man vor allem bei den heutigen Desktoprechnern und sind meist über ein Kabel mit dem Computer verbunden. Allerdings findet man auch Tastaturen, die die Signale über Funk an den Rechner übertragen. Die Tastatur als Einbaugerät findet man vorwiegend bei Mobilcomputern. Der maßgebliche Unterschied zwischen diesen zwei Varianten liegt in der Anzahl der verwendeten Tasten. Die Zeit und die intensive Nutzung einer Tastatur brachten es mit sich, dass darüber hinaus noch ergonomische und auf Schreibgeschwindigkeit optimierte Tastaturen auf dem Markt zu erhalten sind. Ergonomische Tastaturen verfügen über eine Handballenauflage, die den Anwender bei längerem Schreiben unterstützen und somit medizinische Spätfolgen (RSI) reduzieren soll. In der Mitte ge-

knickte Tastaturen sollen der natürlichen Stellung der menschlichen Unterarme gerecht werden, welches dem Benutzer eine bessere Bedienung ermöglichen soll. [BIB02, CT04]

2.2 Die Maus

Die Maus ist ebenfalls ein Eingabegerät, mit der sich Bildschirminhalte auswählen oder markieren lassen. Das Einsatzgebiet einer Maus beschränkt sich meist auf Computer, die mit einer grafischen Benutzeroberfläche (GUI) ausgestattet sind. Mäuse lassen sich danach unterscheiden, wie sie die Bewegungen der Hand in Signale umsetzten, die der Computer anschließend verarbeiten kann. Im Wesentlichen lassen sich Mäuse in mechanische oder aber auch in optische Mäuse unterscheiden, wobei es bei der mechanischen noch eine optomechanische Variante gibt. [BIB02, ST03]

2.2.1 Die mechanische Maus

Am Boden mechanischer Mäuse ist eine mit Gummi überzogene Stahlkugel eingebaut, welche sich durch die Bewegung der Maus auf der Unterlage bewegt. Die Rollbewegung wird im Innern der Maus durch zwei senkrecht zueinander angeordnete Walzen abgenommen. Eine zusätzliche Stützrolle verleiht der Konstruktion die nötige Stabilität. Jede Walze ist für eine Bewegungsrichtung zuständig (senkrecht oder waagerecht). Diagonale Bewegungen kommen also durch die Bewegung beider Walzen zustande. Die Walzen setzen die mechanische Bewegung in Computersignale um.

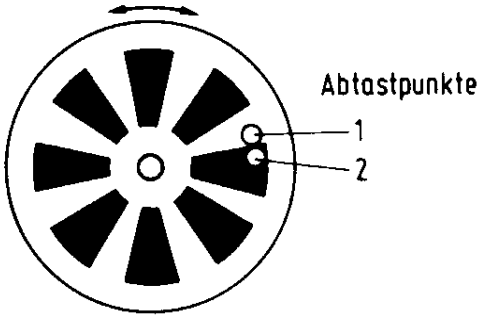

Abb. 2.1: Schema der Lichtschranke einer Maus [ST03]

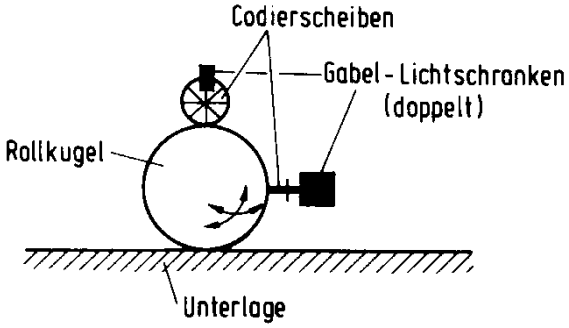

Abb. 2.2: innerer Aufbau einer mechanischen Maus [ST03]

Am Ende der Walzen sind zwei Lochscheiben (siehe Abb. 2.1 und Abb. 2.2) angebracht, die über Lichtschranken abgetastet werden. Abhängig vom zurückgelegten Weg der Kugel werden so für den Computer zählbare Impulse erzeugt und der

Mauszeiger auf dem Bildschirm bewegt. Das Abfragen und Auswerten der Daten erledigt der so genannte Maustreiber. So werden für beide Bewegungsrichtungen (senkrecht und waagerecht) die Signale berechnet.

2.2.2 Bewegungen der Maus

In der Abbildung 2.3 werden die Signale der Bewegung einer Maus in senkrechter oder waagerechter Richtung dargestellt. Die Linie ① kennzeichnet das Signal des Abtastpunktes 1 (siehe Abbildung 2.1). Linie ② kennzeichnet das Signal des Abtastpunktes 2. Anfangs stehen beide Signale auf "Aus", das heißt es fällt kein Lichtstrahl auf beide Abtastpunkte. Bekommt nun der Abtastpunkt 1 zuerst das Signal "An", etwas später dann der Punkt 2, dann dreht sich das Rad im Uhrzeigersinn. Je nach dem wie viel Verzögerung zwischen beiden Signalen liegt, kann der Computer die Geschwindigkeit des Rades berechnen und auf dem Bildschirm darstellen. [ST03]

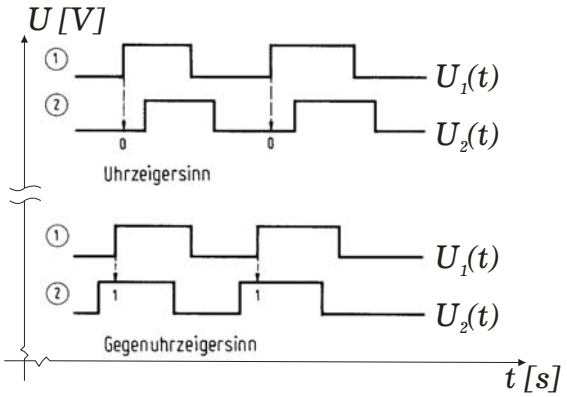

Abb. 2.3: Signale einer mechanischen Maus [ST03]

2.2.3 Die optische Maus

Bei einer reinen optischen Maus entfällt die Rollkugel-Mechanik und die Unterlage wird mittels fotoelektrischer Sensoren abgetastet. Je nach Bewegungsrichtung liefern die Sensoren unterschiedliche Impulsformen. Für die beiden Richtungen (senkrecht und waagerecht) werden meist Sensoren verschiedener Farbempfindlichkeit benutzt. Die Auswertung und Übertragung der Signale erfolgt wie bei der mechanischen Maus über einen eingebauten Mikrocontroller. Bei den optischen Mäusen haben sich zwei unterschiedliche Verfahren entwickelt, je nach dem wie die Signale erkannt und ausgewertet werden. Bei dem ersten Verfahren benötigt man eine Unterlage, auf der ein farbiges Gitter vorhanden ist [ST03]. Die Maus beleuchtet die Linien mit zwei verschiedenfarbigen Leuchtdioden und registriert durch zwei Sensoren die unterschiedlichen Farbreflexionen. Dieses Verfahren ist

sehr alt und findet heute fast keine Anwendung mehr. Die neuere Art von Mäusen benötigt keine Unterlage mehr, hier wird durch ein Fotoverfahren die unterschiedliche Struktur der Unterlage erfasst. Bewegt man die Maus, registriert sie die Bewegung anhand des Unterschieds zum vorherigen Bild. Dieser Unterschied kann dann in Bewegungen auf dem Bildschirm umgesetzt werden. [BIB02, BW04]

2.3 Beschleunigungssensoren

Beschleunigungssensoren oder auch Beschleunigungsaufnehmer werden sehr oft für hochgenaue messtechnische Verfahren verwendet. Sie finden ihren Einsatz in vielen Anwendungsgebieten, wie zum Beispiel in der Luft- und Raumfahrt, Medizintechnik und Verkehrstechnik. Dabei gibt es auch verschiede Varianten von Beschleunigungsaufnehmern, die speziell auf ihren Einsatz ausgelegt sind. In [FB02] wird die Unterteilung von Beschleunigungssensoren in kapazitive, piezoelektronische bzw. piezoresistiv und thermodynamische Beschleunigungsaufnehmer vorgenommen. Jeder dieser Sensoren arbeitet nach dem gleichen physikalischen Prinzip (außer die thermodynamische Variante), die Bewegungen werden aber mit unterschiedlichen Messverfahren ermittelt.

Die Messung von Beschleunigungen (z.B. durch auftretende Schwingungen, hervorgerufen durch eine Krafteinwirkung) basiert auf physikalischen Prinzipien. Dieses sind die Newtonsche Axiome. Besonders das zweite Newtonsche Axiom spielt hier eine entscheidende Rolle, dieses besagt: Wirkt eine Kraft auf einen Körper so wird er in Richtung der wirkenden Kraft beschleunigt. Die Änderung der Bewegung ist der Einwirkung der bewegenden Kraft proportional und geschieht nach der Richtung derjenigen geraden Linie, nach welcher jene Kraft wirkt[2]. [FB02]

Daraus ergeben sich folgende Formeln:

$$\vec{a} \sim \vec{F} \qquad \vec{a} \sim \frac{1}{m} \qquad \vec{F} = m \cdot \vec{a}$$

Die Gleichung $\vec{F} = m \cdot \vec{a}$ berücksichtigt keine Massenänderungen, d.h. in diesem Fall wird die Masse als konstant betrachtet. Ändert sich die Masse, gilt folgender Zusammenhang:

[2] Newton: “Mutationem motus proportionalem esse vi motrici impressae et fieri secundum lineam rectam, qua vis illa imprimatur.” [HU05]

$$\vec{F} = \frac{d}{dt}(m\vec{\upsilon}) = \frac{d}{dt}\vec{p}$$

Diese Gleichung sagt aus, dass eine zeitliche Änderung des Impulses auch eine Änderung der Kraft bewirkt.

Bei den Beschleunigungsaufnehmern werden diese Kräfte indirekt gemessen. Um die Kräfte elektronisch zu erfassen, werden je nach Art des Beschleunigungsaugnehmers Bauteile verwendet, mit denen es möglich ist, die Positionsänderung der Masse messen zu können. Das Prinzip der Positionsänderung wird in der Abbildung 2.4 dargestellt.

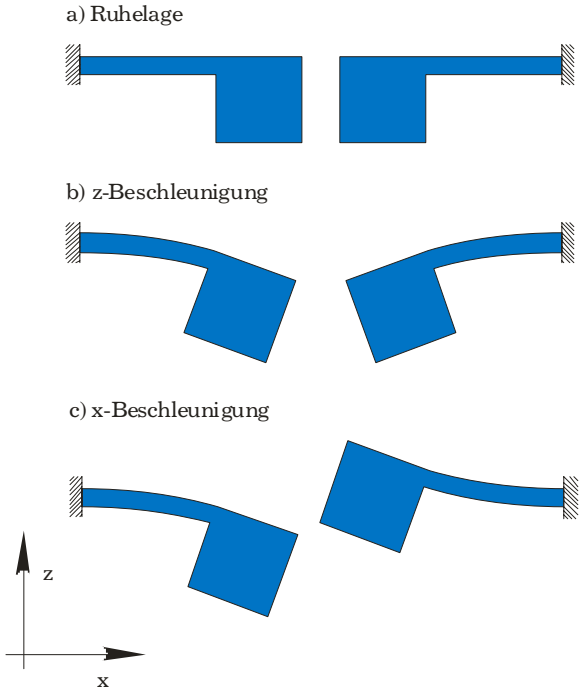

Abb. 2.4:
Darstellung der Auslenkung der Massen bei einem 2D-Beschleunigungssenor.
Masse aufgehängt an einem Biegebalken mit Piezowiderständen. Bild a) Zeigt die Massen bei Ruhelage des Sensors.
Bei einer z-Beschleunigung werden die Massen in gleiche Richtung (b) und bei einer x-Beschleunigung in entgegengesetzte Richtungen (c) ausgelenkt. [FB02]

Auf dem Biegebalken werden vier Widerstände angeordnete und zu einer Wheatstonschen Brücke (siehe Abbildung 2.5) verschaltet. Dadurch ist es möglich auftretende Widerstandsänderungen, durch Dehnung oder Stauchung der Piezowiderstände, messen zu können. Für die Wheatstonsche Brücke gilt, dass die Signalspannung der Brücke aus dem Produkt der Speisespannung und dem Quotienten der Widerstandsänderung zum Gesamtwiderstand gebildet wird.

$$U_S = U_0 \cdot \frac{\Delta R}{R}$$

Die abgegebene Signalspannung verläuft proportional zur Beschleunigung, d.h. je größer die Beschleunigung, desto größer auch die Signalspannung. Die gemessenen Spannungen können anschließend mit einem Analog-Digital-Wandler und mit Hilfe einer Auswertesoftware verarbeitet werden.

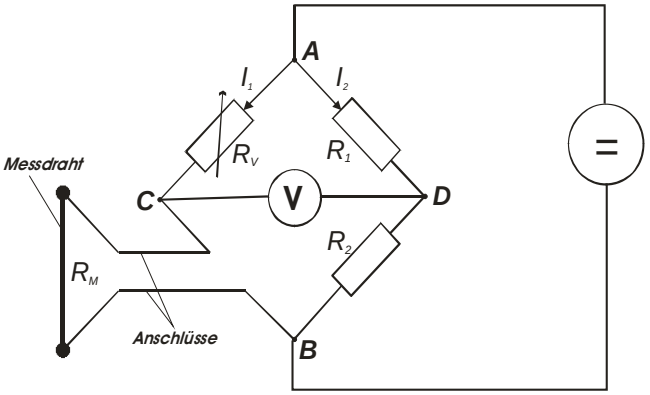

Abb. 2.5: Darstellung einer Wheatstone'schen Brückenschaltung [SC04]

Die Wheatstone'sche Brückenschaltung besteht aus zwei parallel geschalteten Spannungsteilern: ADB und ACB. Beide Spannungsteiler werden mit einer Spannung U gespeist. Im abgeglichenen Zustand fließt kein Strom durch die Punkte C und D (Messdiagonale). Für die Ströme I_1 und I_2 in den beiden Zweigen der Schaltung ergeben sich folgende Werte:

$$I_1 = \frac{U}{R_V + R_M} \qquad I_2 = \frac{U}{R_1 + R_2}$$

Für die Spannung zwischen den Punkten C und B sowie den Punkten D und B ergeben sich:

$$U_{CB} = I_1 \cdot R_M = \frac{U \cdot R_M}{R_V + R_M} \qquad U_{DB} = I_2 \cdot R_3 = \frac{U \cdot R_2}{R_1 + R_2}$$

Die Messspannung U_M wird an den Messpunkten C und D abgenommen, sie setzt sich aus der Differenz der beiden Spannungen U_{CB} und U_{DB} zusammen.

$$U_M = U_{CB} - U_{DB} = U \cdot \left(\frac{R_M}{R_V + R_M} - \frac{R_2}{R_1 + R_2} \right).$$

Im abgeglichenen Zustand, d.h. wenn gilt:

$$\frac{R_M}{R_V + R_M} = \frac{R_1}{R_2},$$

ist die Spannung $U_M = 0$. Im Allgemeinen ist die Brücke wegen streuender Widerstandswerte nicht exakt abgeglichen. Aus diesem Grund wird der Widerstand R_V einstellbar ausgelegt. Für den Abgleich wird R_V so eingestellt, dass gilt:

$$\frac{R_M}{R_V + R_M} = \frac{R_1}{R_2}.$$

Dann ist:

$$U_M = U \cdot \left(\frac{1}{2} - \frac{1}{2}\right) = 0.$$

Wirkt nun eine Kraft auf den Messdraht (siehe Abbildung 2.5) so ändert sich sein Widerstand von R um ΔR. Wählt man $R_2 = R_3 = R$ und $R_V = R$ sowie $R_M = R + \Delta R$, so vereinfacht sich die Gleichung zu:

$$U_M = U \cdot \left(\frac{R + \Delta R}{R + R + \Delta R} - \frac{1}{2}\right) = U \cdot \left(\frac{R + \Delta R}{2R + \Delta R} - \frac{1}{2}\right)$$

$$= U \cdot \left(\frac{\Delta R}{4R + 2\Delta R}\right)$$

Ist $\Delta R << R$, so folgt:

$$U_M = U \cdot \left(\frac{\Delta R}{4R}\right) = \frac{U}{4} \cdot \frac{\Delta R}{R}$$

$$U_M \sim \Delta R$$

Somit ist die gemessene Spannung U_M innerhalb des linearen Bereiches direkt proportional zur spezifischen Widerstandsänderung ΔR der Brücke. Das heißt, dass eine Änderung der Ausgangsspannung eine direkte Änderung der gemessenen Spannung bewirkt. [SC04]

2.4 Algorithmen der Objekterkennung

Die Bildverarbeitung ist ein eigenes, sehr intensiv gepflegtes und komplexes Forschungsgebiet in der Informatik. Ziel ist es, über ein Bild oder eine Sequenz von Bildern, Objekte zu erkennen, diese zu beeinflussen oder anderweitig Informationen daraus zu generieren. Die Überlegung aus einem Bild und später aus einem Datenstrom der Kamera einen Punkt zu ermitteln, der danach auf die Auflösung des Monitors abgebildet werden kann, um hierüber eine Mausbewegung zu realisieren, soll in diesem Abschnitt genauer betrachtet werden.

Da zum Erreichen des oben genannten Ziels mehrere Verfahren zum Einsatz kommen können, sollen hier zunächst einige Algorithmen und Ansätze diskutiert werden. Diese Verfahren sind der Bildabgleich über Korrelation, die verformbare und aktive Konturenerkennung sowie der Ansatz über den *Optical-Flow*-Algorithmus [SD05]. Bei diesem Vergleich soll ein möglichst effektives Verfahren herausgefiltert werden, welches die bestmöglichen Ergebnisse in einer ansprechenden Zeit zur Verfügung stellen. Weiterhin soll der Algorithmus auch mit einer preiswerten (Preisniveau von ca. 20€) und „langsamen“ Kamera arbeiten können, die nur die Standardauflösung einer Webcam (320 x 240 Pixel) liefern kann. Als letzte Anforderung soll das genutzte Bildauswertungsverfahren so flexibel wie möglich sein und nur wenige Einschränkungen dem Nutzer auferlegen.

2.4.1 Bildabgleich über Korrelation

Das Ziel des Bildabgleichs über Korrelation [KW03] ist, jede Instanz eines speziellen Objektes in einer Szene über die Verwendung einer bestimmten Maske oder Schablone zu finden. Die Schablone ist hierbei ebenfalls ein Bild oder genauer eine bestimmte Ausprägung von Pixelwerten, welches das gesuchte Objekt beinhaltet, beziehungsweise diesem entspricht. Am Beispiel eines kugelförmigen Gastanks in Abbildung 2.6 soll dieses Prinzip genauer erläutert werden. Dieser Gastank und alle ähnlichen Ausprägungen sollen in einem Bild (Abbildung 2.7) wieder gefunden werden.

Abb. 2.6: Schablonen-Bild [KW03]

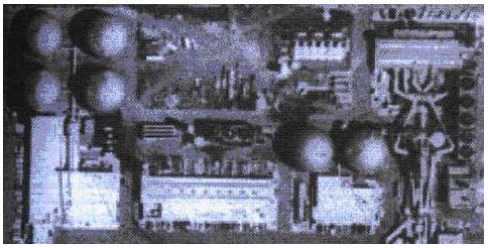

Abb. 2.7: Fabrikbild mit Objekten [KW03]

Zur Erfüllung dieser Lokalisierungsaufgabe wird das Schablonen-Bild so auf das eigentliche Bild abgebildet, dass Gruppen von Pixel, die mit der Schablone korrelieren nahe weiß und jene die nicht übereinstimmen schwarz werden. Die folgende Abbildung verdeutlicht diesen Vorgang:

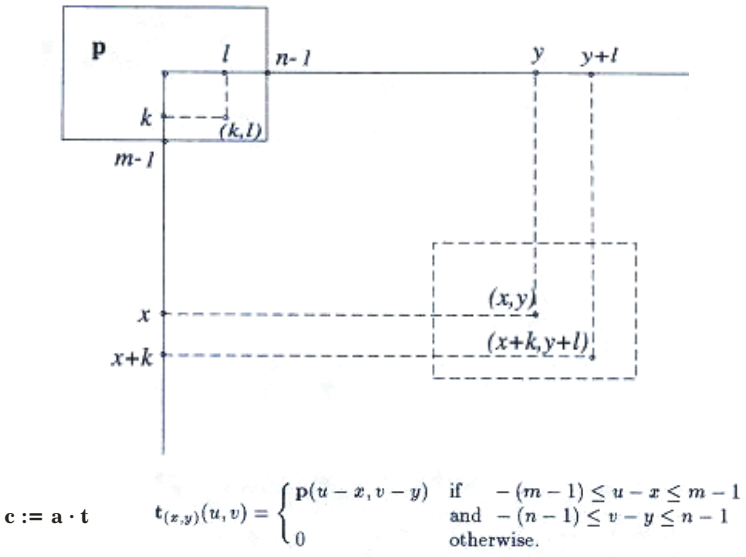

Abb. 2.8: Schematische Darstellung der Korrelation [KW03]

In der obigen Gleichung in der Abbildung 2.8 entspricht dabei c dem Ausgabebild, a dem Quellbild und t ist die Schablone, die durch Pixelwerte (p) repräsentiert wird. Dabei ist t so definiert, dass t schwarz wird, wenn die Pixel nicht übereinstimmen und andernfalls je nach Ähnlichkeit helle Pixel gesetzt werden. Da das Schablonenbild überlappend aufgelegt wird, werden vor allem bei sehr hohen Übereinstimmungen viele Pixel gesetzt, so dass an diesen Stellen besondere Konzentrationen im Ausgabebild entstehen.

Das Ergebnis dieser Methode angewandt auf das Fabrikbild sieht dann wie folgt aus:

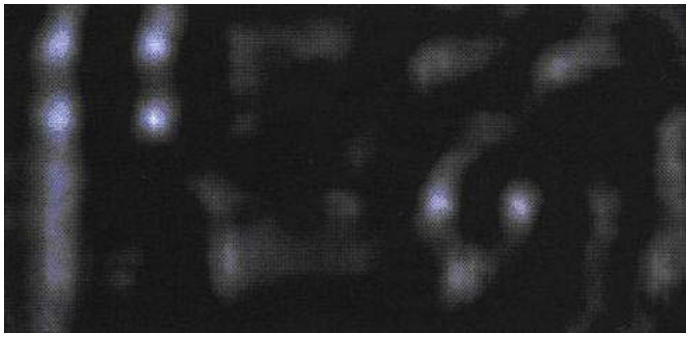

Abb. 2.9: Bild nach der Korrelation [KW03]

An der Abbildung 2.9 ist zu erkennen, dass diese Methode die Gastanks erfolgreich gefunden und weiß hervorgehoben hat. Es fällt allerdings auch auf, dass es weitere Objekte gibt, die fast als Gastank identifiziert worden sind, obwohl diese keine starke Ähnlichkeit (erkennbar an der Graufärbung) besitzen. Die Gastank-Lokalisierung kann deutlicher gemacht werden, indem Grenzwerte berücksichtigt werden (siehe Abbildung 2.10). Alle Pixel, die dabei nicht nahe genug an den Wert weiß sind, werden schwarz eingefärbt. Dadurch kann folgendes Bild gewonnen werden:

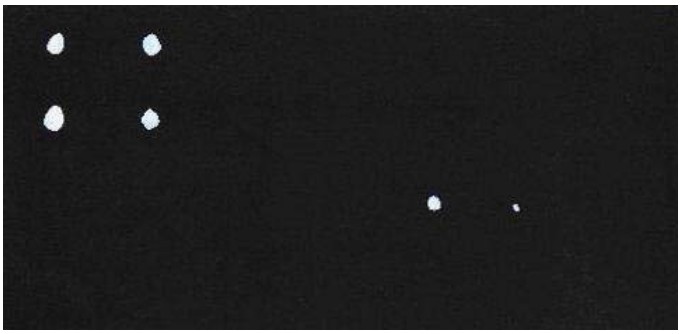

Abb. 2.10: Nachbearbeitet Version des Bildes [KW03]

Der Ort der sechs Tanks kann nun sehr genau bestimmt werden. Diese Methode kann dabei auch in der Frequenzbereichsanalyse benutzt werden. Das Finden von Objekten über den Bild-Abgleich mittels Korrelation unterliegt allerdings einigen Beschränkungen. Es erfordert ein genaues Bild des gesuchten Objekts und zwar so, wie es auch im Suchbild vorhanden ist. Die Abhängigkeit der Methode von den Lichtverhältnissen ist ebenfalls ein Problem, denn diese müssen permanent gleich bleiben, da sonst die Pixelwerte weniger „gleich“ zu den originalen Werten sind. Weiterhin führen Änderungen in der Ausrichtung und der Größe des Objektes zu erheblichen Beeinflussungen der Performance, so dass Echtzeit-Anwendungen, wie es auch innerhalb dieser Studienarbeit angestrebt ist, über diese Methode nicht verwirklicht werden können.

2.4.2 Konturerkennung

Bei der Konturerkennung kann man zwei wesentliche Methoden unterscheiden. Zum einen sind das die verformbare Konturerkennung (*deformable contour model)* [KW03] und zum anderen die aktive Konturerkennung (*active contour model*).

Die verformbare Konturerkennung ist in [AY02] beschrieben. Die grundlegende Idee ist, dass es ein ideelles Umrissmodell oder eine Umrissschablone des zu findenden Objektes existiert. Dieses Modell ist eine einfache geometrische Struktur, die nur aus wenigen Parametern besteht. Abhängig von der Parametrisierung kann sich diese geometrische Figur verändern, zum Beispiel skalieren oder ziehen,

so dass auch undeutliche oder größere Objekte der gleichen Art ebenfalls erkannt werden. In dem unteren Bild der Abbildung 2.11 ist es die Kontur eines Schach-Turms. Die angenäherte Umriss-Schablone ist durch eine Serie von Punkten definiert. Innerhalb des Bildes wird dieser Satz von Punkten mit anderen nahen Konturen verglichen. Wenn sich die Konturen in akzeptabler Verformung zum Schablonen-Umriss befinden, wird dieses als Übereinstimmung gewertet. Die unteren Bilder enthalten einige Beispiele für den Algorithmus und die darin gefundenen Konturen. Wie in dem letzten Bild deutlich wird, hat diese Methode Probleme, wenn der Hintergrund eine ähnliche Helligkeit wie das Objekt selbst hat.

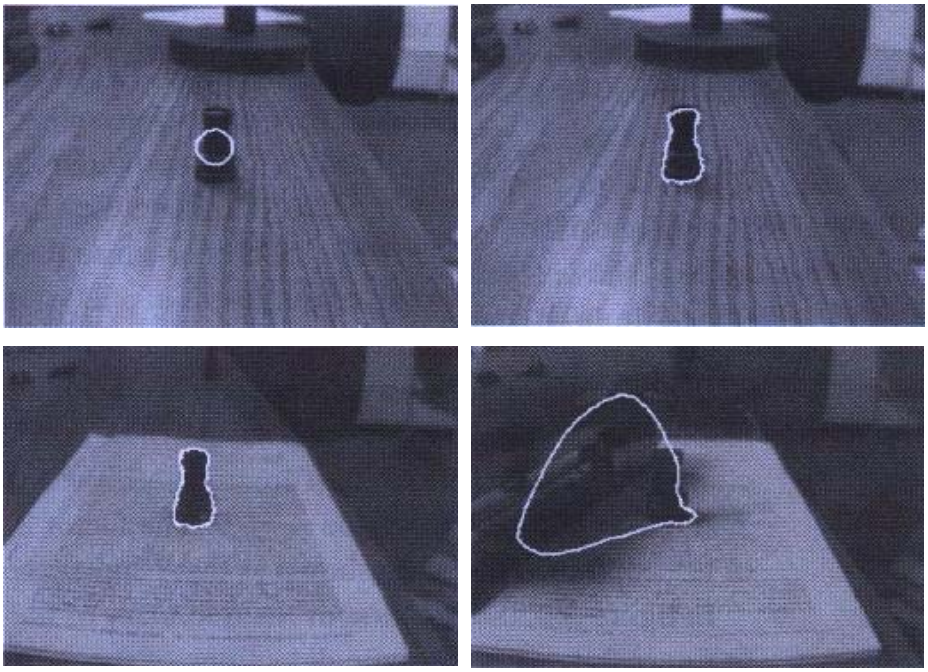

Abb. 2.11: Beispiele für die verformbare Konturerkennung [KW03]

In [KWT87] wird das Konzept der aktiven Konturerkennung, auch *Snakes* genannt, erläutert. Bei diesem Verfahren können beliebige Objekte nach dem Kontrast, der zwischen ihren Konturen und dem Hintergrund besteht, extrahiert werden. Innerhalb dieser Methode werden die so genannten *Snakes* eingesetzt. *Snakes* bezeichnen eine geschlossene, einfach zusammenhängende Kurve in der Ebene. Diese Kurve wird so lange verkleinert bis sie ein einfaches Objekt an Hand seiner Begrenzung zum Hintergrund, also seiner Kontur, umrahmt. Deshalb wird zunächst auf das Original-Bild eine Kantendetektion meist mit dem Sobel-Algorithmus [HT05] durchgeführt, um diese Kontur leicht annähern zu können und unwichtige Bildmerkmale zu eliminieren. Anschließende wird die optimale *Snake* gesucht, wofür allerdings viele Iterationsschritte von Nöten sind. Die Arbeitsweise dieses Verfahrens ist in Abbildung 2.12 dargestellt. Eine genauere Erläuterung zur Optimierung diese Problematik auf Basis von Energieniveauschemata, kann [MM00] entnommen werden.

Abb. 2.12: Arbeitsweise der aktiven Korrelation, annähern der *Snake* (blau) ans Objekt [YK05]

Neben dem hohen Rechenaufwand kommt die Problematik der Kantenerkennung hinzu. Es kann durch seitliche Beleuchtung, Schatten oder fließenden Helligkeitsverläufen zwischen Hintergrund und Objekt dazu kommen, dass die Konturen an einigen Stellen unterbrochen und nur sehr schwach oder überhaupt nicht erkennbar sind. Vor allem an solchen Punkten muss darauf geachtet werden, dass die *Snake* nicht in das Objekt hineinläuft. Ein weiteres Problem kann durch mögliche Kantenfragmente hervorgerufen werden, die durch Rauschen oder Schatten entstehen. Diese Stellen müssen von der *Snake* während der Annäherung übersprungen werden, da sie sonst den Weg zu den Kanten aufhalten.

Bei den beiden vorgestellten Konturerkennungsmethoden wird deutlich, dass diese sich auch nur bedingt bis überhaupt nicht für das Verfolgen genau eines Punktes eignen. Bei der verformbaren Konturerkennung müsste entweder ein besonderer Punkt oder ähnliches auf einem Finger verfolgt werden, da es sonst Schwierigkeiten gibt, die einzelnen Finger auseinander zu halten. Dies bedeutet auch, dass man auf eine vordefinierte Struktur begrenzt ist. Bei der aktiven Konturerkennung ist dies ebenso, da zwangsläufig der Arm aus dem Kamerabild herauslaufen wird und so kein geschlossenes Objekt verfügbar ist und die Hand als Verfolgungsobjekt zu groß wäre. Der Rechenaufwand dieses Systems ist ebenfalls eine große Problematik für das Thema „kontaktlose Maus“.

2.4.3 Optical-Flow-Algorithmus

Der *Optical-Flow*-Algorithmus [SD05] ist kein Konturerkennungsverfahren, wie die zuvor beschriebenen Verfahren sondern eine Methode der Bildpunkt-Verfolgung. Genauer gesagt wird versucht besondere Merkmale eines Bildes in einem Folgebild wieder zu finden. Mathematisch ausgedrückt heißt das, die gegebenen Punkte $[u_x, u_y]^T$ eines Bildes I_1 werden als Punkt $[u_x + \delta_x, u_y + \delta_y]^T$ im zweiten Bild I_2 so gesucht, dass ε minimal wird:

$$\varepsilon(\delta_x, \delta_y) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} \left(I_1(x,y) - I_2(x+\delta_x, y+\delta_y) \right)$$

Diese Methode benutzt weiterhin zur Berechnung neben den Bildkoordinaten die Intensität und den Aufnahmezeitpunkt. Mit geeigneten Annahmen, zum Beispiel dass sich die Intensität eines Pixels (bewegend oder ruhend) innerhalb eines Objektes im Verlauf der Zeit nicht ändert, kann man einen Punkt verfolgen.

Dieses Verfahren wird in der Bildverarbeitung häufig verwendet um:

- Objekte eines Bild in einem anderen wieder zu finden.
- Herauszufinden, wie sich ein Objekt bewegt hat.
- Die Tiefe in einem Bild mit nur einer Kamera zu ermitteln.

Der erste Punkt ist vor allem für die Problematik „kontaktlose Maus“ interessant. Zunächst ist es jedoch nötig geeignete Punkte zu ermitteln, die verfolgt werden können. In [ST94] werden Merkmale die sich besonders gut zur Verfolgung eignen wie folgt definiert:

Sie haben eine scharfe Struktur oder eine Zweideutigkeit bei der Verfolgung oder sind besonders kontrastreiche Kanten.

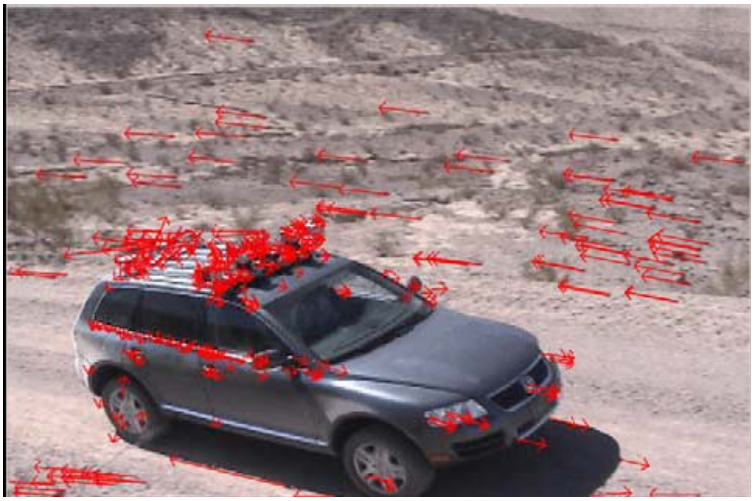

Abb. 2.13: *Optical-Flow*-Algorithmus bei der Verfolgung eines sich bewegenden Objekts [SD05]

Die Filterung nach solchen Punkten geschieht also relativ einfach über vergleiche mit Nachbarpunkten. Hat man diese Punkte aus dem Bild erfolgreich gefiltert und anschließend in dem nächsten Bild wieder gefunden, ist es nun möglich Aussagen über die Bewegung innerhalb eines Bildes zu treffen. Dieses kann man anschließend wie in Abbildung 2.13 beispielsweise durch Vektoren visualisieren.

Durch das Verwenden dieser Methode kann man festere Aussagen über die einzelnen Pixel tätigen. Verwendet man die Punkte, die direkt bei der Suche nach besonderen Merkmalen gefunden werden, kann man sich nicht sicher sein, ob diese nicht vielleicht nur sehr kurze temporäre Streuungen oder Rauschen sind. Es ist zwar auch bei Verwendung des *Optical-Flow* nicht ausgeschlossen, da aber die Verifizierung über zwei Bilder geschieht und nur die gleichen Punkte wieder gefunden werden, kann man über eine genauere Weiterverarbeitung, zu einem hinreichend sicheren System gelangen.

Das Problem des *Optical-Flow*-Algorithmus ist, dass der Algorithmus für das Finden der besonderen Merkmale nur optimal funktioniert, wenn sich in dem zu untersuchenden Bild besonders scharfe Kanten oder eine starke Strukturen befinden. Diese Tatsache ist bei einer normalen Hand und guten Beleuchtung leider nicht überall gegeben, wie sich in verschiedenen Tests zeigte (vgl. Kapitel 7). An diesem Punkt kann man sich mit einer einfachen Methode behelfen, und zwar kann man beispielsweise einen zusätzlichen Punkt oder einen Buchstaben auf dem Finger anbringen, damit dieser dann ein besonders gutes Merkmal im Bild darstellt und somit einfacher detektiert und extrahiert werden kann. Im Kapitel 7 ist beschrieben, mit welchen einfachen Methoden Strukturen gefunden werden können, die man dann als markantes Merkmal benutzen kann, damit eine Verfolgung vereinfacht wird.

3 Bestehende Konzepte für „kontaktlose Mäuse“

In diesem Kapitel werden bestehende Technologien vorgestellt, die die Steuerung einer Hard- oder Softwarekomponente ermöglichen bzw. die Verfahren bereithält, mit denen man eine Maussteuerung zu realisieren ist. Die Technologien können dabei in drei Kategorien eingestuft werden: Der Ansteuerung und die Erfassung von Sensordaten, das Auswerten von Kamerabildern und die Steuerung über das Gehirn. Hier werden die bestehenden Komponenten und Technologien eingegliedert und am Beispiel exemplarisch vorgestellt.

3.1 Sensoransteuerung

Die Verwendung von Sensoren zur Auswertung der Bewegung - beziehungsweise der Position der Maus - ist im Prinzip nicht neu. Sie wird beispielsweise auch bei optischen Mäusen eingesetzt. Durch den Einsatz neuer beziehungsweise anderer Sensoren ist es allerdings möglich, eine Maus zu realisieren, die gewissermaßen kontaktlos arbeitet. Es ist offensichtlich, dass die Sensoren selbst irgendwo befestigt werden müssen. Dieses muss entweder an der Hand, einem anderen Körperteil oder einem Gegenstand geschehen.

3.1.1 Beschleunigungsmaus – TEGMouse

Abb. 3.1: TEGMouse mit Beschleunigungssensoren [TEG04]

Die Entwicklungsgruppe TEG des Fraunhofer Institut in Stuttgart hat eine Maus, wie sie in Abbildung 3.1 dargestellt ist, entwickelt und patentieren lassen, die über die Auswertung von Beschleunigungssensoren (vgl. Kapitel 2.3) reagiert. Durch die Verwendung von Beschleunigungssensoren, ist kein mechanischer oder optischer Kontakt zu einer Unterfläche mehr nötig, was ein verschleißfreies Arbeiten zur

Folge hat. Weiterhin ist es dadurch möglich diese in feuchten oder stark verschmutzten Bereichen einzusetzen, da keine Öffnungen für Sensoren vorhanden sind. Die Maus ist dabei mit zwei Beschleunigungssensoren ausgestattet. Die Auswertung findet über den aktuellen Neigungswinkel der Maus statt, so dass die Maus ebenfalls als Pointer im Raum verwendet werden kann. Ein weiterer Vorteil gegenüber konventionellen Mäusen ist ein sehr geringer Stromverbrauch des Systems. Dies ermöglicht ein langes Arbeiten, ohne ständig auf die Energiereserven achten zu müssen. Derzeit existiert nur ein kabelgebundener Prototyp. Es wird allerdings an einer Variante gearbeitet, die die Signale per Funk zum Computer transferiert. Die Zukunft dieser Technologie sieht Fraunhofer in der Miniaturisierung des Systems, so dass eines Tages über extrem kleine Sensoren eine sehr kleine Fingermaus ermöglicht wird. [TEG04]

3.1.2 Gyration Ultra Cordless Optical Mouse

Abb. 3.2: Gyration Maus mit Gyrosensoren [GI04]

Die Firma Gyration hat sich auf kabellose Eingabegeräte spezialisiert. Darunter auch die Ultra Cordless Optical Mouse (Abbildung 3.2). Sie ist so konzipiert, dass sie wie eine normale Maus auf dem Tisch mit optischen Sensoren arbeitet. Darüber hinaus ist es mit ihr möglich, über einen integrierten Kreiselsensor – Gyroskop – die Maus in der Luft zu benutzen. Die Bewegung der Maus in der Luft entspricht der natürlichen Bewegung der Arme und vor allem der Handgelenke. Durch ihre ergonomische Form liegt sie in der Hand, wie eine Fernbedienung, nur dass man mit ihr die Maus steuern kann und damit das kontaktlose Arbeiten ermöglicht. Die Daten werden über einen internen Sender mittels Funk (80Hz) zum Computer übertragen, wodurch eine Bedienung im Vergleich zu anderen Mäusen auch aus größerer Entfernung möglich ist. [GI04]

3.1.3 Datenhandschuhe

Datenhandschuhe (Abbildung 3.3) sind in den verschiedensten Varianten verfügbar. Ihre Preisspanne beginnt bei etwa 900 € und geht bis zu 15.000 €, je nach Funktionsumfang und -prinzip. Das grundlegende Prinzip ist, dass über Bewegungen von Hand und Finger eine Orientierung im Raum berechnet wird. Es gibt mehr als ein Dutzend verschiedene Ausprägungen, die sich vor allem im Funktionsumfang und Preis unterscheiden. Einige Datenhandschuhe bieten heute, neben der normalen Orientierung, Funktionen zum Ertasten eines Gegenstandes (Taktiles Feedback), Kraftrückkoppelung (Force Feedback) und die Gestenerkennung an.

Abb. 3.3: Datenhandschuh von VPL Research [HKHV04]

Die fortgeschrittensten Produkte sind heutzutage eher ein Exoskelett[3] als nur ein einfacher Datenhandschuh. Das Exoskellet wird an jeden Finger der Hand angebracht. Man kann somit für jeden einzelnen Finger die Beugung der drei Fingergelenke messen und umsetzen. Es existieren bereits auch kabellose Datenhandschuhe, die über Funk oder Bluetooth mit dem Rechner kommunizieren können. Eine ausführliche Zusammenfassung der unterschiedlichen Methoden, wie die Beugung gemessen werden kann, und weitere Beispiele kann [HKHV04] entnommen werden.

[3] Ein Exoskelett ist eine Stützstruktur, welche eine stabile äußere Hülle um etwas bildet. Meist bei Gliederfüßern oder Insekten vorhandene äußere (Exo) Hülle (Skelett). [WDE05]

3.2 *Auswerten von Kamerabildern*

Die Verwendung von Kameras zur Auswertung einzelner Bilder und deren entsprechende Interpretierung ist eine weit entwickelte Technik. Für diese Umsetzung wird ein Kamerabild in Echtzeit ausgewertet. Hierbei wird untersucht, wo sich ein gewisser Punkt befindet und dieser als Mauszeiger interpretiert. Dabei gibt es verschiedenste Ansätze: Zum einen kann man das reine Kamerabild oder dieses durch weitere prägnante Merkmale unterstützt auswerten. Die Findungsweise der zu verfolgenden Punkte kann sehr unterschiedlich angegangen werden. Einige Beispiele sind bereits in Kapitel 2.4 beschrieben.

3.2.1 *Das System „Siemens Virtual Touchscreen" im FZ Karlsruhe*

Abb. 3.4: Wetterinformationssystem (Virtual Touchscreen) im FZ Karlsruhe

Der Virtual Touchscreen der Firma Siemens besitzt weder Maus noch Tastatur als Eingabegerät. Er beruht auf dem System der Infrarotabtastung, wobei die Hand als Maus fungiert. Hierzu ist über einem horizontal befestigten Bildschirm eine Box angebracht, die eine Kamera, eine Lichtquelle sowie einen Infrarotsensor besitzt. Die Position der ganzen Hand wird durch dieses System gemessen und der nördlichste Punkt auf den Bildschirm übertragen. Die Bewegung des Cursors ist deutlich langsamer als bei einer normalen Maus. Mausklicks[4] werden über die Zeit bestimmt. Wenn man länger den Cursor nicht bewegt, d.h. die Hand verharrt im Zustand der Ruhe, so wird ein linker Mausklick an der aktuellen Position der Hand und damit auch des Cursors ausgeführt. Ein Klick mit der „rechten Maustaste" ist bei diesem System nicht möglich, wird von dem darauf angepassten Menu allerdings auch nicht benötigt.

Im Eingangsbereich der Kantine des Forschungszentrums Karlsruhe (FZK) ist eine solche Anlage installiert (siehe Abbildung 3.4). Hier können allgemeine Wetterinformation für Europa abgerufen werden.

[4] Ein Mausklick ist ein Ereignis, welches durch das Betätigen einer Maustaste ausgelöst wird.

3.2.2 *Verfolgung der Nasenspitze mittels „Nouse“*

Nouse steht für „Nose and Mouse“, hierbei handelt es sich um ein Projekt der Computational Video Group des Institute for Information Technologie unter Schirmherrschaft des National Research Council Canada. [PVT04]

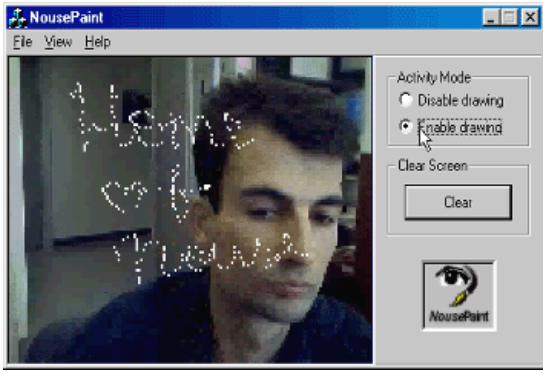

Abb. 3.5: *Nouse*Paint [PVT04]

Das Projekt wurde im September 2001 gestartet und hat die Steuerung des Computers über die Verfolgung der Nasenspitze zum Ziel. Es ist vor allem für Personen gedacht, die wegen körperlicher Behinderungen nicht mit Maus und Tastatur arbeiten können.

Das System basiert auf der Verfolgung der konvexen Form der Nase, dessen Extremum der Konvexenhülle die Nasenspitze ist. Die Nasenspitze wird deshalb verwendet, da sie unabhängig von Skalierungen und Rotationen ist und bei jeder Gesichtsrichtung gesehen werden kann. Da die Nase mit einer Subpixel-Genauigkeit verfolgt werden kann, ist es möglich mit diesem Punkt wie mit einer Maus zu arbeiten. Die Fakten, dass die Nasenspitze einzigartig in ihrer Form ist, genau in der Mitte des Gesicht und zwischen den Augen liegt sowie den weitentferntesten Punkt von den Gesichtsachsen bei einer Bewegung darstellt, sind weitere Gründe weshalb die Nase als zu verfolgender Punkt gewählt wurde.

Das System kann mit einem doppeltem Blinzeln - Doubleblink™ - aktiviert und deaktiviert werden. Mausklicks werden ebenfalls auf diese Art und Weise übermittelt. Dies geschieht mit einer binären Folge von Blinzeln und der Erkennung, welches Auge diese durchgeführt hat, so dass linker und rechter Mausklick unterscheidbar sind.

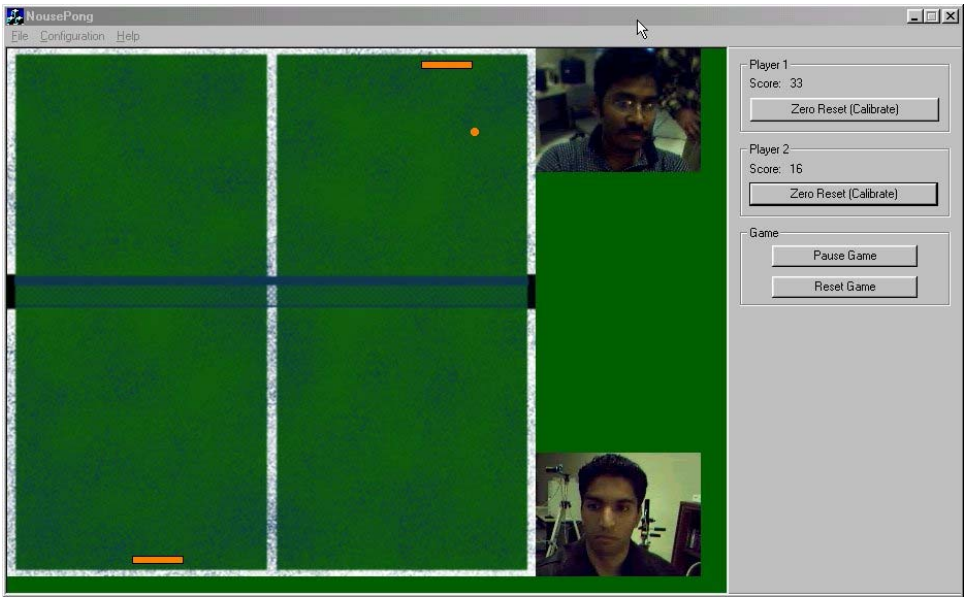

Abb. 3.6: *Nouse*Pong [PVT04]

Mit der *Nouse* sind bereits einige weitere Projekte umgesetzt, so ein rudimentäres Zeichenprogramm (siehe Abbildung 3.5) oder auch Spiele wie Pong (siehe Abbildung 3.6), die über die Verfolgung der Nasenspitze gespielt werden können.

3.2.3 Playstation2-Erweiterung „EyeToy“

Abb. 3.7: EyeToy Kamera [SI04]

EyeToy ist die bekannteste Art und Weise der Steuerung über die Auswertung von Kameradaten. Es ist ein von Sony in Kooperation mit Logitech entwickeltes System, mit dem Spiele für die Playstation2 über eine Kamera gesteuert werden. Die Hardware besteht aus einer hochauflösenden Webcam von Logitech (siehe Abbildung 3.7), die via USB mit der Spielkonsole verbunden wird. Das System ist nach der Treiberinstallation in der Lage, die Bewegungen des Spielers durch die Kamera aufzuzeichnen und in das Spiel zu integrieren. Am Anfang war das System in der Lage, nur simple Bewegungen wahr zu nehmen. Deshalb waren die Spiele auch eher einfach gehalten und besaßen einen statischen Hintergrund, in dem der

Spieler eingeblendet wurde und er Gegenstände wegschlagen oder verschieben musste.

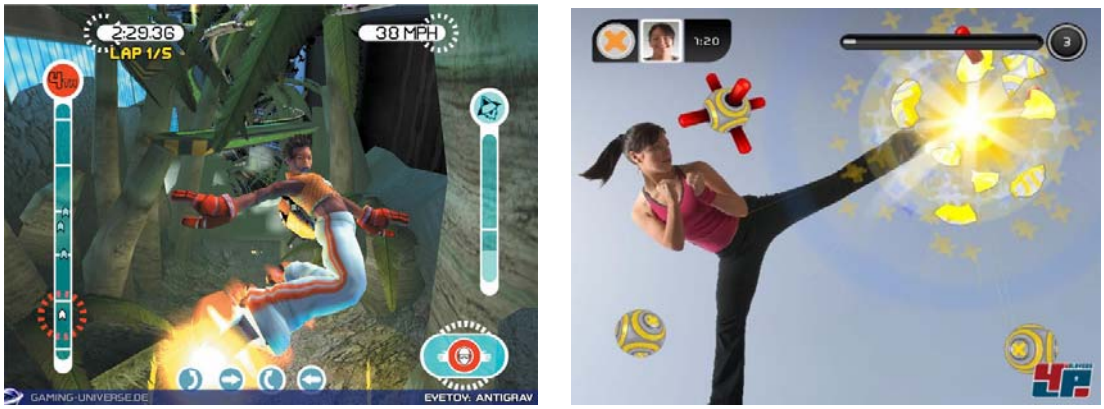

Abb. 3.8: EyeToy Spiele AntiGrav (links; [PGU05]) und Kinetic (rechts; [S4P05])

In der neuen Generation, wie das Spiel *AntiGrav* (siehe Abbildung 3.8), ist EyeToy in der Lage einen räumlichen Bezug herzustellen, um so ein mehr oder weniger realistisches 3D-Gefühl zu ermöglichen. Dabei wird die Kamera zunächst auf den Kopf des Spielers kalibriert. Anschließend verfolgt das Spiel dessen Bewegung im Raum und überträgt diese auf die virtuelle Spielfigur. Geht der Spieler beispielsweise in die Knie, duckt sich auch sein „Alter Ego“ im Spiel, mit seitlichen Bewegungen kann man Hindernisse ausweichen etc. Zusätzlich zum Kopf überwacht das Spiel auch die Position der Hände – da die Leistungsfähigkeit des Prozessors bis hier allerdings schon ausgelastet ist, finden hier nur einfache Algorithmen Anwendung. Über die Hände kann man im Spiel dann beispielsweise Gegenstände einsammeln, so dass eine hohe Anforderung an Körper und Koordination gestellt werden. [FW05, WE05, SI04]

3.3 Steuerung über das Gehirn

Die komplexeste und herausfordernste Art der Mausbewegung ist die direkte Steuerung über das Gehirn. Hier sind bereits einige Forschungen im Gange, wobei viele Grundlagen erst ergründet werden müssen. Das Projektes BBCI – Berlin Brain-Computer Interface des Fraunhofer Institut für Rechnerarchitektur und Softwaretechnik forscht bereits seit einigen Jahren in diese Richtung.

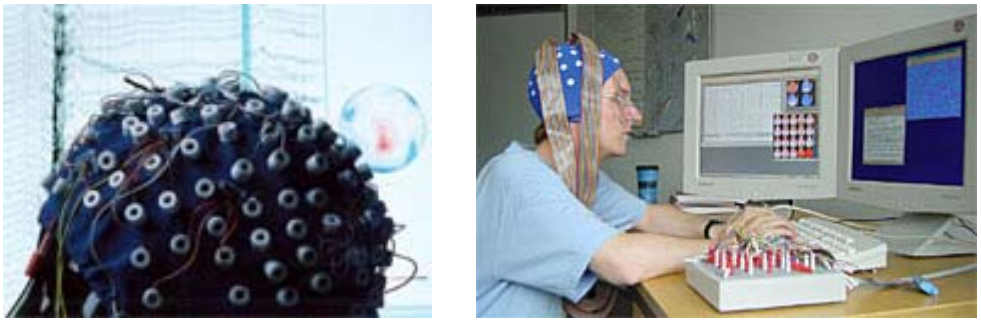

Abb. 3.9: EEG-Kappe [BR04]

Über die Auswertung elektrische Impulse in Form des Elektroenzephalogramms (EEG), will man eine Schnittstelle zwischen Gehirn und Computer entwickeln, das so genannte Brain-Computer-Interface (BCI). Darüber soll es beispielsweise möglich sein, den Cursor zu versetzen oder andere Kommandos direkt vom Gehirn in den Computer zu übermitteln. Dazu werden an dem Kopf Elektronen angebracht, welche die hirnelektrischen Signale messen, die beim Denken bis in die Kopfhaut übertragen werden (siehe Abbildung 3.9). Diese werden anschließend an einen Computer übermittelt, der diese Signale in technische Steuersignale wandelt. Das Funktionsprinzip beruht darauf, dass bereits bei rein gedanklicher Vorstellung eine Hirnaktivität zu verzeichnen ist.

Dieses Prinzip funktioniert bereits seit einiger Zeit, allerdings sind lange Trainingsphasen nötig gewesen, bei denen die Probanten beispielsweise Bälle von einer Seite des Bildschirms bewegen müssen. Neuartige Methoden haben dieses Problem bereits behoben, wobei das System nicht bei allen Menschen funktionsfähig ist.

Die Einsatzorte sind vielfältig, wobei die Verwendung des Systems in erster Linie für körperlich benachteiligte Menschen entwickelt und demnach auch von solchen Menschen benutzt wird. Der Einsatz als Presave-Technologie ist jedoch ebenfalls denkbar. Wenn es zum Beispiel darum geht in einer Notsituation schnell zu handeln, ist das Gehirn deutlich schneller als die darauf folgende Aktion des Menschen. Greift man diese Informationen ab, kann ein Computer schneller reagieren und gegebenenfalls größeren Schaden verhindern. [FF05, BR04]

3.4 Vor- und Nachteile der vorgestellten Konzepte

Einige der vorgestellten Möglichkeiten, eine kontaktlose Maus beziehungsweise eine neuartige Maussteuerung zu entwickeln, befinden sich noch in einer frühen Entwicklungsphase. Am fortgeschrittensten sind das EyeToy-System sowie die Ultra Cordless Optical Mouse von Gyration, die es bereits bis zur Marktreife ge-

bracht haben. An diesen wird trotzdem noch weiterentwickelt, da das Potenzial welches in diesen Technologien liegt, noch nicht vollständig erschlossen ist.

Die meisten Forschungen konzentrieren sich derzeit auf die Auswertung von Kameradaten und dem Brain-Computer-Interface, wobei die Steuerung einer Maus nur ein Nebenziel der Entwicklung darstellt. Nachfolgend sollen die wichtigsten Vor- und Nachteile der Systeme betrachtet werden, um einen groben Überblick bezüglich einer eigenen Umsetzung zu erhalten.

Die Steuerung einer Maus über Sensorauswertung ist schon in vielen Bereichen recht fortgeschritten. Nicht zuletzt durch die Entwicklungen der letzen Jahre, die eine Umsetzung über optische Sensoren mit sich brachte. Dieses System ist sehr vorteilhaft aufgrund der Schnelligkeit, mit der Sensoren Daten liefern und angesprochen werden können. Hochauflösende Sensoren ermöglichen weiterhin eine genaue Positionierung auf dem Bildschirm. Dadurch, dass die Sensoren irgendwo eingebettet werden müssen, was meistens in einem Design der üblichen Mäuse resultiert, ist die Eingewöhnungsphase recht kurz und ein flexibles System kann gewährleistet werden. Hat sich eine Technologie durchgesetzt und wird vom Menschen ohne großen Widerstand angenommen, so können die Kosten durch eine Mehrproduktion gesenkt und somit auch zu einem niedrigeren Preis angeboten werden. Die Ultra Cordless Optical Mouse von Gyration ist beispielsweise schon ab 120 € verfügbar [GI04]. Durch die Verwendung von modernen Computerchips, ist auch eine Umsetzung der verschiedenen Signale auf bewährte Standards möglich, so dass sie mit den meisten Systemen relativ schnell und einfach kompatibel sind. Nachteilig, vor allem im Hinblick auf eine kontaktlose Maus ist allerdings, dass man immer noch ein Gerät in der Hand halten muss, um eine Reaktion zu erzeugen. Weiterhin benötigt man einiges an elektrotechnisches Fachwissen um die verschiedenen Sensoren richtig anzusprechen, die Daten umzuwandeln und einen entsprechenden Schaltplan zu erstellen, der auch innerhalb eines kleinen Geräts integriert werden kann.

Die Entwicklung eines Eingabegeräts über die Auswertung von Kameradaten, wird vor allem durch Entwicklungen und Bedürfnisse der Roboter-Forschung und der Entertainment-/Spiele-Branche getragen. Die meisten Systeme in diesem Bereich sind ebenfalls relativ Preiswert, was die Hardware angeht. Der größte Kostenfaktor macht die Software-Entwicklung und Forschung neuerer, schnellerer und zuverlässigerer Algorithmen aus. Bei komplexeren Systemen, die Spezialkameras verlangen, können weitere erhebliche Kosten entstehen. Die Vorteile dieses Systems sind primär, dass durch eine Visualisierung des Kamerabildes ein schnelles und intuitives Arbeiten möglich ist. Dies wird vor allem bei dem EyeToy-System deutlich, bei dem der Nutzer direkt ins Spielgeschehen eingebettet ist. Die Möglichkeit mit einer Kamera einen ganzen Bereich abzudecken kann genutzt

werden, nicht nur um ein Merkmal zu extrahieren sondern gleich mehrere Punkte zu verfolgen und darauf zu reagieren. Problematisch an dieser Methode ist allerdings derzeit der Hohe Rechenaufwand, der für die Umsetzung erforderlich ist. Es gibt zwar viele verschiedene Algorithmen, die zum Teil beachtliche Ergebnisse liefern (vgl. Kapitel 2.4), da diese allerdings mit sehr großen Datenmengen agieren müssen, benötigt man bei einigen Systemen genug Rechenleistung und Rechenzeit. Dies wird bei den Tests mit dem System *Nouse* schnell deutlich, verursacht es doch teilweise eine Rechnerauslastung von 90%[5]. Bei der Erklärung zum EyeToy-System ist ebenfalls bereits deutlich geworden, dass es derzeit schon an seine Grenzen trifft und so keine komplexeren Algorithmen mehr eingesetzt werden könnten, die ein besseres Ergebnis liefern. Die Anfälligkeit der einzelnen Systeme gegenüber Einflüssen aus der Umgebung ist ebenfalls ein Problem, vor allem was den Umgang mit Licht, Rauschen und Schatten betrifft. Diese Störeinflüsse der Umwelt können zwar durch unterschiedliche Filtermethoden reduziert werden, die Anwendung von Filtern bedeutet aber meist auch erheblich mehr Rechenaufwand.

Das Brain-Computer-Interface (BCI) ist wohl die anspruchvollste und faszinierendste Technik, der vorgestellten System. Sowohl der Einsatz für körperlich benachteiligte Menschen, die sich so wieder mit ihrer Umwelt in Verbindung setzen könnten, die Möglichkeiten der genannten Presave-Technik und des Entertainmentbereichs sind besondere Gründe, warum versucht wird ein solches System zu entwickeln. Vor allem die schnelle Reaktionszeit des Gehirns gegenüber den Muskeln ist ein beachtlicher Vorteil. Die Forschungen und Entwicklungen sind allerdings noch verbesserungswürdig, so dass eine Vermarktung des Systems auch einige Zeit in Anspruche nehmen wird. Da die Gehirnströme teilweise von Mensch zu Mensch verschieden sind, wird es wohl in absehbarer Zeit kein System geben, das für alle sofort anwendbar ist. Es sind jeweils lange Trainingssitzungen nötig, um die richtigen Signale „zu denken“. Teilweise sind 300 Trainingsstunden erforderlich, um ein BCI halbwegs vernünftig benutzen zu können. Bei anderen Trainingsmethoden sind dies Bestenfalls nur 20 Minuten, wobei diese Systeme sehr individuumsabhängig sind und nicht bei allen Menschen gleichermaßen funktionieren. Die EEG-Kappe, die eingesetzt werden muss um die Gehirnströme aufzunehmen, ist ebenfalls in mehrer Hinsicht zu verbessern. Zum einen ist diese sehr unbequem und mit sehr vielen Kabeln versehen. Zum anderen wurde von vielen Testpersonen berichtet, dass sie eher ein Gefühl des kontrolliert werden verspürten, als selbst etwas zu kontrollieren [BR04]. Vor allem das Misstrauen welches der Technik gegenübersteht, dass man komplett überwacht oder gar beeinflussbar wird, wird ein großes Problem bei einer zukünftigen Markteinführung mit sich bringen. Diese wird allerdings noch einige Jahre auf sich warten lassen.

[5] Dieser Wert ist Systemabhängig und beruht auf dem Testsystem 1 (vgl. Kapitel 7).

4 Machbarkeitsstudie

In der Machbarkeitsstudie wird zunächst überprüft, welche der aufgeführten Verfahren sich am besten für eine preiswerte und umsetzbare Realisierung anbieten. Auf Grund der extrem hohen Komplexität wird das Auswerten von Gehirnströmen hier vernachlässigt.

4.1 Beschleunigungssensoren

Wie im Kapitel 2.3 beschrieben können Beschleunigungssensoren durch ihre physikalischen Eigenschaften zum Positionieren eines Mauszeigers benutzt werden. In einem Projekt der Fraunhofer Technologie-Entwicklungsgruppe, wurde diese Art der Positionierung bereits verwirklicht und patentiert [TEG04]. Zur Verwendung solcher Sensoren innerhalb dieses Projektes wurden die Eigenschaften sowie die Vor- und Nachteile analysiert und bewertet.

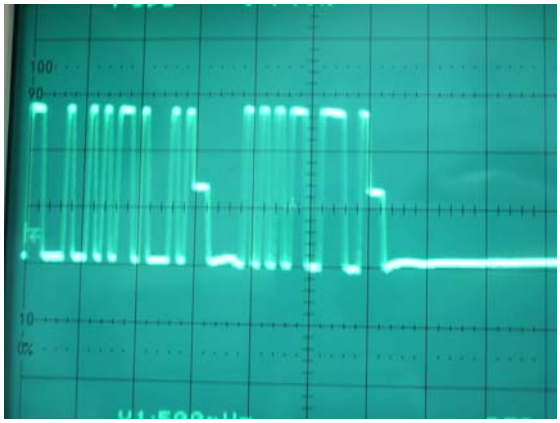

Abb. 4.1: Signale einer USB-Maus auf dem Oszilloskop

Dazu wurde anfangs versucht die Signale einer USB-Maus zu analysieren. Hierzu wurden die Signale der Maus mittels eines Oszilloskops dargestellt (siehe Abbildung 4.1). Die Betrachtung der Oszilloskop-Ergebnisse ergab, dass das so gewonnene Signal eine sehr komplexe Struktur aufweist, welches sich für eine Nachahmung nur sehr schwierig eignet. Ziel war es mit den so gewonnenen Werten, das Signal der Beschleunigungssensoren auf das eigentliche Signal der Maus (vgl. Abbildung 4.1 und Kapitel 2.3) abzubilden, um so eine Bewegung des Mauszeigers zu verwirklichen. Je nach Art des Beschleunigungssensors kann man so die Bewegung im zwei- bzw. im dreidimensionalen Raum verwirklichen. Dagegen spricht allerdings, dass es zwar einerseits ein kontaktloses System darstellt, bezüglich des Raumes, der Benutzer aber dennoch einen Gegenstand in die Hand nehmen muss, um den Mauszeiger zu bewegen.

Letztendlich entscheidend, dass Beschleunigungssensoren zur Positionierung des Mauszeigers innerhalb dieses Projekts nicht zu verwenden sind ist, dass man mit Beschleunigungssensoren ein hohes Rauschen in jeder Bewegungsrichtung erhält. Dieses Rauschen wird allein schon durch geringste Bewegungen gegenüber der Erdgravitation hervorgerufen, das heißt die Reduzierung des Rauschens würde beträchtlichen Rechenaufwand bedeuten oder den Einsatz einer zusätzlichen Filter-Schaltung, die das System noch weiter vergrößern würde. Mit Sensoren, die eine eingebaute Filter-Schaltung besitzen und damit nicht so rauschanfällig sind, wäre das Projekt finanziell nicht mehr tragbar (Sensoren ab 1.000 €) und ein Ziel dieser Arbeit verletzt, und zwar ein möglichst Kosten sparendes System zu implementieren.

4.2 Auswertung von Kameradaten

Für die Auswertung von Kameradaten spricht in erster Linie die Möglichkeit, mit einem geringen Budget erste Schritte und Implementierungen zu prüfen. Dies wurde mit einer handelsüblichen Webcam durchgeführt, deren Preis um die 20 € liegt. So können ungefähre Geschwindigkeiten der Verarbeitungsgeschwindigkeit der einzelnen Bilder abgeschätzt und allgemeine Ansätze ermittelt werden.

Für die Realisierung einer Maus auf Basis von Kameradaten spricht weiterhin, dass keine komplexen elektrotechnischen Kenntnisse erforderlich sind. Die Daten der Kamera – die einzelnen Bilder – werden komplett von einer zu implementierenden Software empfangen, verarbeitet und eine Bewegung darüber erzeugt. Hier zeigte sich in ersten Tests, dass viele Möglichkeiten einer Implementierung bestehen. Vor allem die unterschiedlichsten Verfahren, die die Verfolgung eines Punktes in einem Bild ermöglichen (vgl. Kapitel 2.4) fallen hier auf. Somit ist eine große Bandbreite an Untersuchungsmaterial gegeben, was den Reiz erhöht, eine Implementierung auf diese Art und Weise durchzuführen.

Nachteilig bei einer Implementierung ist, dass die Software eine hohe Anforderungen an die Systemleistung stellen wird, dies ist vor allem auf den Umgang mit den einzelnen Bildern zurück zu führen, die jeweils zur Findung eines Punktes übertragen werden müssen. Weiterhin reagiert die Kamera sehr unterschiedlich auf verschiedene Beleuchtungsintensitäten und –einfallswinkel, so dass das System von hier aus einige Schwachstellen mitbringt, die über eine robuste Software gelöst werden müssen.

4.3 Fazit der Machbarkeitsstudie

Die Nutzung von Beschleunigungssensoren zeigt, wie beschrieben, dass viele Problematiken mit diesem Bereich einhergehen. Zum einen die Kosten als auch die vielen physikalischen Eigenschaften, die sich schnell negativ auf das Gesamtsystem auswirken können, sind hier zu nennen. Weiterhin ist eine Überführung der Beschleunigungssensor-Daten in Mausdaten, wie sie normalerweise übertragen werden, relativ umständlich. Dahingegen ist die Implementierung einer Software, die einzelne Bilder auswertet und einen gewissen Punkt auf den Bildschirm überträgt, deutlich einfacher. Das Beispiel *Nouse* (vgl. Kapitel 3.2.2) zeigt weiterhin die interessanten Möglichkeiten, die diese Methode mit sich bringt. Sie ist sehr weit ausbaubar und bietet vielerlei Ansätze für eine Implementierung und spätere Erweiterungen, so dass ein kamerabezogenes System forschungsträchtiger ist und sehr flexibel erstellt werden kann. Die zukünftigen Möglichkeiten, ein solches Projekt weiter zu bearbeiten, sind deutlich umfangreicher als eine abgeschlossene Erstellung einer Maus mittels Sensoren.

Aus den aufgeführten Gründen geht hervor, dass eine Umsetzung innerhalb dieser Studienarbeit, mit der Auswertung von Kameradaten am erfolgversprechensten ist. Dieses Verfahren kommt auch der Thematik „kontaktlose Maus“ am nächsten. Deshalb soll folgend die Implementierung dieser Methodik erläutert und hierbei zunächst ein Systementwurf durchgeführt werden. Dabei wird darauf geachtet, dass sehr umfangreiche Daten verarbeitet werden müssen, was eine hohe Hardwareauslastung nach sich ziehen könnte, so dass effiziente und datensensitive Verfahren[6] zu bevorzugen sind.

[6] Verfahren, die besonders Wert auf die Qualität der Daten im Verhältnis zu einer geringen Quantität legen.

5 Systementwurf

Bei dem Systementwurf, soll ein möglichst flexibler und robuster Entwurf eines Programms entstehen, welches auf Basis der Auswertung von Kameradaten eine Bewegung einer Maus nachempfindet. Dabei soll das Programm so implementiert werden, dass es möglich ist mit einer sehr preiswerten Kamera und einer damit verbunden geringen Auflösung trotzdem ein System zu entwickeln, mit dem ein alltägliches Arbeiten möglich ist. Das Programm wird mit Microsoft Visual Studio 6.0 erstellt. Visual Studio wird deshalb verwendet, da es somit möglich ist in C++, ein im Vergleich zu Java schnelleres Programm zu erstellen [WE05a], was vor allem dem Auswerten von Bildern und damit großer Datenmengen zugute kommt.

Bei der Implementierung wird auf eine intuitive und einfach zu bedienende grafische Benutzeroberfläche geachtet. Die grafische Benutzeroberfläche (GUI) ist die Schnittstelle zum Nutzer, sie wird in C++ unter Verwendung der Version 3.0 der Microsoft Foundation Classes [WE05b] implementiert[7]. Hierüber ist es möglich dem Benutzer eine Oberfläche bereitzustellen, die wie eine normale Windows-Anwendung aussieht und sich auch so verhält, so dass eine kurze Einarbeitungszeit gewährleistet ist.

Abb. 5.1: Schematische Darstellung des Programmablaufs

Zunächst wird hierfür eine kurze Einleitung zu der verwendetet Programm-Bibliothek *OpenCV* [IC04] geliefert. Die drei primären Elemente der Implementierung (siehe Abbildung 5.1), und zwar der Empfang des Kamerabildes, das Auswerten der Daten sowie die Implementierung der Mausbewegungen (optional Mausklicks) sollen folgend genauer erläutert werden. Anschließend wird erklärt, wie diese Teile zusammengeführt werden, um ein komplettes System zu erhalten.

5.1 Vorstellung der Programm-Bibliothek OpenCV

Die Bibliothek *OpenCV* ist von der Firma Intel [IC04]. *OpenCV* wurde entwickelt um Forschern, kommerziellen Software-Entwicklern und Kamera-Entwicklern eine Plattform zu bieten, um computerunterstütztes Sehen in Mensch-Computer-Schnittstellen, der Robotik, des Monitoring, der Biometrie sowie der Sicherheit zu

[7] Zur Einrichtung von Microsoft Visual Studio 6 siehe Anhang

bieten. Sie stellt zur Erfüllung dieser Ziele eine freie und offene Infrastruktur mit mehr als 200 Funktionen zur Verfügung. Da das vorgestellte System *Nouse* (vgl. Kapitel 3.2.2) ebenfalls auf *OpenCV* zurückgreift, scheint genügend Potenzial in dieser Bibliothek vorhanden zu sein, um diese gewinnbringend und zur Erfüllung des Projektes einzusetzen. [IC04]

Die Bibliothek *OpenCV* ist in der Programmiersprache C implementiert. Dies ist darauf zurückzuführen, dass C eine sehr maschinennahe Sprache ist und so eine schnelle Abarbeitung der einzelnen Programmteile ermöglicht, im Vergleich zu Java [WB05a]. Das ist besonders wichtig da beim Arbeiten mit Bildern große Datenmengen anfallen können und der Zeitverzug so gering wie möglich sein muss, besonders wenn der Aspekt der Echtzeit hinzukommt.

5.2 *Empfangen des Kamerabildes*

Für die Bildauswertung ist es zunächst erforderlich ein Bild einer Kamera zu empfangen, um aus diesem besondere Merkmale und Punkte zu extrahieren. Die bereits vorgestellte Bibliothek *OpenCV* (vgl. Kapitel 5.1) stellt hierfür *CvCam* zur Verfügung. *CvCam* ist ein universelles und plattformunabhängiges Modul, für die Verarbeitung von Videodatenströme[8] digitaler Kameras. Es ist als dynamische gelinkte Bibliothek (dynamic link library) für Windows und als Objekt-Bibliothek (shared object library) für Linux implementiert. Es enthält eine einfache und praktische API für das Lesen und Überwachen von Videodatenströmen, die Verarbeitung einzelner Frames[9] und für das Darstellen der Ergebnisse. Weiterhin ist es mittels *CvCam* möglich, direkt verschiedene Kameras anzusprechen, sich alle angeschlossenen Kameras anzeigen zu lassen oder die Höhe und Breite eines Bildes zu ermitteln. Auf die einzelnen Frames kann über Zeiger zugegriffen werden, die auf ein in *OpenCV* mitgeliefertes Bild-Format verweißen. Am Anfang wird überprüft, ob eine Kamera angeschlossenen ist. Dies wird automatisch bei der Initialisierung ausgeführt. Sollte kein Gerät gefunden werden, so wird das Programm automatisch beendet[10]. Mittels der Funktion `cvQueryFrame` kann anschließend auf die einzelnen Frames zugegriffen werden. Damit besteht quasi kein kontinuierli-

[8] Ein Videodatenstrom gibt ein kontinuierliches Datensignal an den Computer ab. Vor allem im Internet wird diese Technik verwendet, um es dem Nutzer zu ermöglichen, ein Video oder ein Ton zu hören, ohne dass die ganze Datei bereits herunter geladen werden muss.

[9] Frame (engl.) – Bezeichnung für ein einziges Bild.

[10] Sind mehrere aufnahmefähige Geräte an den Computer angeschlossen, wählt das Programm das nächste Gerät aus und stellt das Bild dieses Gerätes dar. Wenn also eine Fernsehkarte in den Computer integriert ist, mit der es möglich ist das laufende Programm aufzunehmen, dann empfängt das Programm ein Bild des Fernseh-Tuners und stellt dieses dar. Das Bild ist allerdings nicht zum Bewegen der Maus geeignet. Das Programm sollte an dieser Stelle manuell beendet werden.

cher Datenstrom der verarbeitet werden muss, sondern es wird immer nur dann ein Bild geholt, wenn es benötigt wird

5.3 *Auswerten der Daten mittels Optical-Flow*

Da jetzt ein Kamerabild bereitgestellt ist, kann dieses so ausgewertet werden, dass ein bestimmter Punkt im Bild verfolgt wird, der als Referenzpunkt für den neuen Mauszeiger dient. Dieser Punkt muss anschließend auf die Bildschirmauflösung skaliert und abgebildet sowie der Mauszeiger zu diesem Ort repositioniert werden, da dieser Punkt den aktuellen Mauszeiger angibt.

Um einen speziellen Punkt innerhalb des Bildes zu verfolgen, wird der *Optical-Flow*-Algorithmus benutzt, der im Kapitel 2.4.3 vorgestellt ist. Die Entscheidung für dieses Verfahren ist darauf zurück zu führen, da hier kein spezifisches Objekt gefunden wird, sondern zunächst alle besonderen Merkmale aus einem Bild extrahiert werden. Die Merkmale werden dabei bei besonders scharfe Strukturen oder kontrastreiche Kanten im Bild gefunden. Dies erlaubt eine hohe Flexibilität. Da der Algorithmus für sehr markante Stellen im Bild, auch mehrere Merkmale findet und die einzelnen Ergebnispunkte gewichtet, ist es über eine geschickte Weiterverarbeitung der Ergebnisse möglich nur das hervorstechendste Merkmal zu extrahieren. Diese Weiterverarbeitung kann so aussehen, dass eine Punktkonzentration für ein besonderes Merkmal als Referenzpunkt für die spätere Verfolgung dient. Dazu muss gewährleistet sein, dass diese Konzentration im Bild wirklich einzigartig ist und einen gewissen Schwellwert nicht unterschreitet.

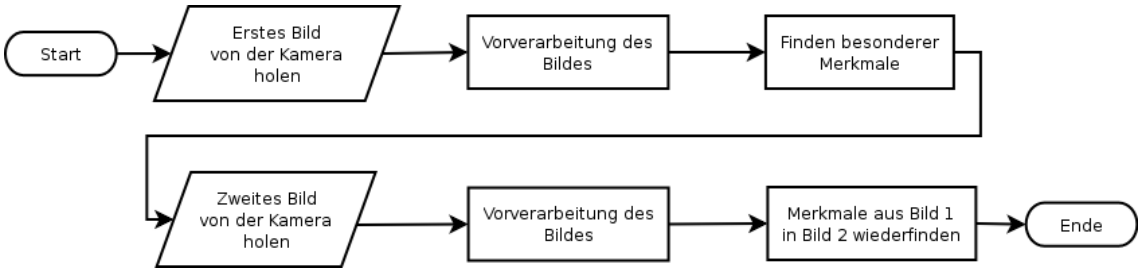

Abb. 5.2: Schematische Darstellung des *Optical-Flow*-Algorithmus

Im Abbildung 5.2 ist eine schematische Darstellung des Algorithmus dargestellt. Zunächst wird ein Bild von der Kamera empfangen. Dieses wird im Sinne der Vorverarbeitung in ein Grauwert-Bild[11] umgewandelt, da die weiteren Schritte nur auf Bilder solcher Farbtiefe anwendbar sind. Des Weiteren wird in der Bildverarbeitung allgemein, vor allem aus Geschwindigkeitsgründen, auf Grauwertbilder zurückgegriffen. Weiterhin werden hier zusätzliche Schritte zum Unterdrücken von

[11] Ein Bild, bei dem es keine Farbwerte gibt. Es existieren ausschließlich Informationen zu den Grauwerten in einem Bereich von 0 - 255, wobei 0 schwarz und 255 weiß entspricht.

Rauscheffekten vorgenommen, die beispielsweise durch Schatten oder zu dunkler Umgebung entstehen und störenden Einfluss auf das Finden von hervorstechenden Kanten haben. Im Besonderen wird hier eine Anpassung des Kontrastes und der Helligkeit unternommen. Dieser Teil wird im Abschnitt Aspekte der Realisierung (Kapitel 6) besondert betrachtet. Desgleichen wird mit dem zweiten Frame verfahren, so dass quasi das erste Bild das Referenz-Bild darstellt und das zweite Bild darauf aufsetzt und in dem die Merkmale des ersten Bildes wieder gefunden werden sollen.

5.3.1 *Finden besonderer Bildmerkmale*

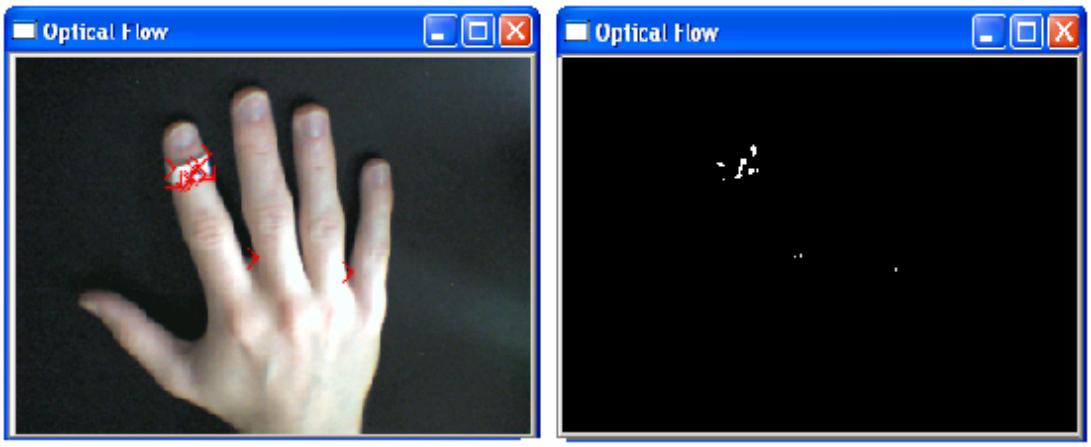

Abb. 5.3: Eigenwert-Bild[12] im vergleich zum Kamerabild

Auf das erste Bild wird nach dem Prinzip von Shi und Tomasi (vgl. [ST94]) der Algorithmus über die *OpenCV* Funktion `cvGoodFeaturesToTrack` zur Detektion besonders starker Kanten angewandt. Dieser Algorithmus findet Eckpunkte mit besonders großem Eigenwert (siehe Abbildung 5.3) in einem Bild. Hierzu wird für jeden Pixel im Bild, der minimale Eigenwert berechnet. Anschließend wird das Bild so gefiltert, dass nur noch Maxima in einer Nachbarschaft von 3 x 3 übrig bleiben. Im nächsten Schritt, werden die gefundenen Kanten nach ihrem Eigenwerten gewichtet. Dazu werden alle Kanten die weniger als *(mindest Qualität)* x *max(minimale Eigenwert(x,y))* aus der Ergebnismenge der gefunden Punkte entfernt. Der prozentuale Wert – mit dem Wertebereich 0,01 bis 1 – für die mindest Qualität, wird als Parameter der Funktion übergeben. Durch Anpassung dieses Wertes ist es also möglich nur Kanten zu finden, die in ihrer Ausprägung zu einem gewissen Grad genauso stark sind, wie der Maximalwert. Als letztes wird sicherge-

[12] Eigenwert gehört zur linearen Algebra und treten auf, wenn ein Vektor mit einer Matrix multipliziert wird und hierbei das gleiche Ergebnis entsteht, als wenn der Vektor mit einem Skalar multipliziert wird. Das Skalar wird dann Eigenwert genannt. (vgl. [WE05c]) Hierüber ist es möglich Punkte in einem Bild zu finden, die besonders starke Strukturen oder Kontraste aufweisen.

stellt, dass die gefunden Merkmale alle weit genug von einander entfernt sind. Dabei werden die starken Kanten zuerst betrachtet. Es wird der euklidische Abstand zwischen diesem und den anderen Merkmalen betrachtet und alle Merkmale die kleiner als eine gewisse mindest Distanz aufweisen, ebenfalls eliminiert. Diese mindest Distanz wird der Funktion ebenfalls als Parameter übergeben. Da jedoch eine Punktkonzentration innerhalb eines Bildes erwünscht ist, wird dieser minimale Abstand innerhalb der Implementierung, somit gleichzeitig der Parameter, auf null gesetzt. Eine genaue Erläuterung der verschiedenen Parameter, kann der OpenCV-Dokumentation [SF05] oder der Implementierung entnommen werden.

5.3.2 *Wieder finden der besonderen Bildmerkmale*

Mit Hilfe dieser gewonnen Punkte wird auf das zweite Bild der eigentlich *Optical-Flow*-Algorithmus angewendet, der diese Punkte wieder finden soll. Dazu wird eine iterative Version, das heißt über Schleifen und nicht über mehrmaligen Funktionsaufruf, des *Optical-Flow* von Lukas und Kanade [LK81] nach Bouguet [BJY99] über Pyramiden verwendet. Der Algorithmus *Optical-Flow* auf Pyramiden heißt deshalb so, da der Bereich in dem die eigentlichen Vergleiche und Berechnungen durchgeführt werden, verschiedene Größen der Originalbilder sind, die wie Pyramiden aussehen (siehe Abbildung 5.4). Der Algorithmus berechnet hierüber für die gegebenen Koordinaten, die Koordinaten im Folgebild.

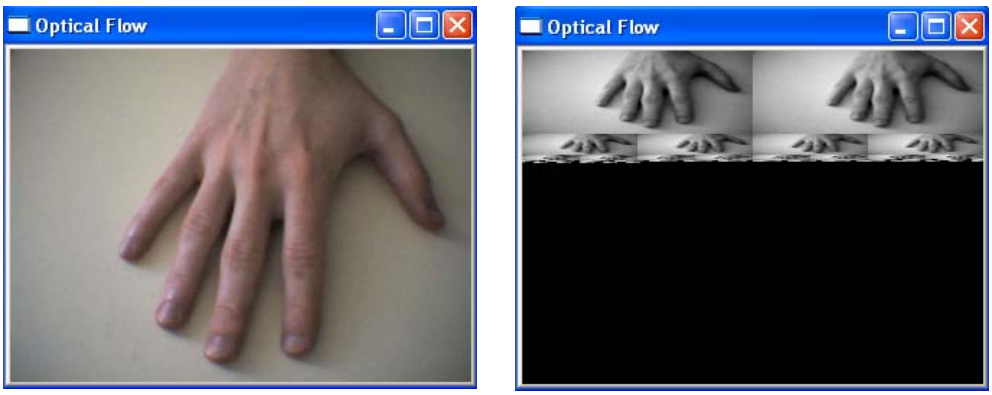

Abb. 5.4: Bilder des *Optical-Flow* nach Lukas und Kanada Pyramiden

Es wird beginnend mit dem kleinsten Bild versucht, die Punkte wieder zu finden. Anschließend werden die Ergebnisse dieser Berechnung an das nächst größere Pyramidenbild als initialisierte Vermutung weitergegeben und die Berechnung über diese Werte erneut durchgeführt. Dieses wird so oft wiederholt, bis man beim größten und damit dem Originalbild angekommen ist. Der Vorteil dieser Methode liegt darin, dass eine schnellere Berechnung des Lucas und Kanada Algorithmus möglich ist. Dies ist durch die übergebenen Vermutungen bedingt, die heuristisch gesehen sehr oft zutreffen und bei einmaliger Verifizierung jeweils fertig abgear-

beitet sind. Eine komplette mathematische Beschreibung dieses Algorithmus kann [BJY99] entnommen werden.

Nachdem Ausführen dieses Teils, liegen nun drei Arrays vor. Eines enthält die Orte der wieder gefundenen Merkmale, jeweils für beide Bilder, und das dritte Fehler für die nicht gefundenen Merkmale. Diese können in den nächsten Schritten, bei Berechnung und Durchführung der Mausbewegung, weiter ausgewertet werden.

5.4 Implementierung der Mausbewegungen

Die Realisierung einer Mausbewegung ist auf verschiedene Arten und Weisen möglich. Dabei wird immer auf die Funktion `SetCursorPos` zurückgegriffen, die Teil der Windows-Standard Bibliothek user32.dll ist. Diese Funktion ermöglicht es den Mauszeiger zu positionieren. Um ein möglichst gleichmäßige Bewegung ohne auffällige Sprünge zu erhalten, muss zwischen dem Zielpunkt und der aktuellen Cursorposition auf dem Bildschirm interpoliert und die Funktion mehrmals für diese Punkte aufgerufen werden. Diese Interpolation ist relativ einfach über das Aufstellen einer linearen Gleichung (siehe Kapitel 5.4.3) zwischen den beiden Punkten möglich. Als erstes ist es allerdings notwendig den Zielpunkt aus den Kamerabildern, beziehungsweise aus den bereits vorliegenden Arrays aller besonderen Merkmale zu bestimmen. Anschließend kann dieser Punkt auf den Monitor übertragen werden. Für diesen Schritt sind wiederum verschiedene Szenarien denkbar, zum Beispiel kann er über Absolutwerte oder relativ zum aktuellen Cursor gesetzt werden. Als nächstes kann man den Cursor entsprechend der Interpolation versetzen, wobei das Aufstellen der linearen Gleichung und Schrittweite, für die Zwischenwerte die ebenfalls gesetzt werden müssen um eine gewohnte Art der Mausbewegung zu erhalten, gleichermaßen wichtig ist. Der vierte und letzte Schritt ist die Integration von Mausklicks. Diese vier Bereiche sollen folgend genauer betrachtet werden.

5.4.1 Bestimmung des Zielpunktes

Für die Bestimmung des Zielpunkts aus den Bildern heraus können die Arrays des *Optical-Flow*-Algorithmus verwendet werden. Hierbei handelt es sich um zwei Arrays. Das erste Array enthält die besonderen Merkmale des ersten Bildes. Das zweite Array enthält die wieder gefundenen Merkmale im zweiten Bild. Da für besonders hervorstechende Merkmale mehrere Ausprägungen in den Arrays vorhanden sind, kann über folgende Auswertung eine einzige Merkmalskonzentration extrahiert werden. In den einzelnen Array wird für jeden einzelnen Punkt überprüft, wie weit die anderen Punkte von diesem entfernt sind (siehe Abbildung 5.5).

Über alle Merkmale, die einen kleineren Abstand haben, als der vom Nutzer in den Einstellungen (vgl. Kapitel 6.4) definierte, wird das arithmetische Mittel berechnet und dieses gespeichert. Desgleichen wird mit dem zweiten Array dieser Art verfahren. Die Berechnungsvorschrift für diese Problematik sowie ein Zahlenbeispiel sind in Anhang B Berechnung der Punktkonzentration zu finden.

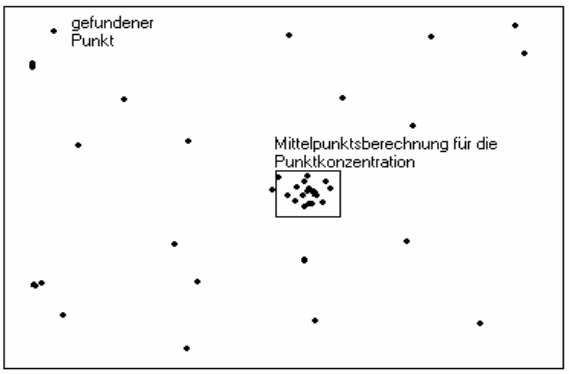

Abb. 5.5: Schematische Darstellung der Nachbarschaftsberechnung

Für die einzelnen Punktkonzentrationen wird zusätzlich ein Zähler mitgeführt, der angibt, wie viele Merkmale in der entsprechenden Nachbarschaft existieren. Dieser kann später verwendet werden, um zu ermitteln für welche Konzentrationen die meisten Nachbarn vorhanden sind. Wenn diese Konzentration einen bestimmten Schwellwert überschreitet, werden die beiden extrahierten Punktkonzentrationen vom ersten Bild und jene der wieder gefundenen Merkmale vom zweiten Bild arithmetisch zusammengefasst und als Referenzpunkt im Kamerabild interpretiert.

5.4.2 Übertragung des Zielpunkts auf den Monitor

Der Zielpunkt liegt durch die obigen Berechnungen in Kamerakoordinaten vor. Bei einer normalen Webcam entspricht das einem Bereich von 320 x 240 Pixeln. Dieser Punkt muss für das effektive Versetzen des Mauszeigers auf den Monitor abgebildet werden. Hier sind zwei Möglichkeiten denkbar. Zum einen kann über die Annahme einer absoluten und zum anderen einer relativen Positionierung ausgegangen werden.

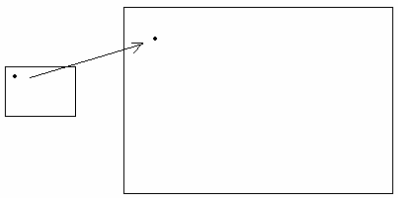

Abb. 5.6: absolute Positionierung

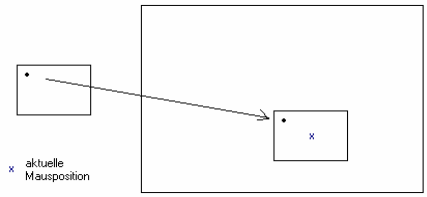

Abb. 5.7: relative Positionierung

Bei der absoluten Positionierung ergibt die Position des Zielpunktes aus den Kamerabildern genau einen neuen Punkt in einer anderen Ebene (Abbildung 5.6). Die Ebene, in der projiziert wird, kann andere Ausmaße in Höhe und Breite besitzen. In diesem Fall muss eine Berechnungsvorschrift eingehalten werden, die eine Breite in die andere überführt und ebenso mit der Höhe verfährt, so dass alle Punkte, die im Original möglich sind, eine eindeutig Entsprechung in der neuen Ebene finden. Dies kann über den einfachen Dreisatz durchgeführt werden, wobei Ausdehnungen in x- und y-Richtung differenziert betrachtet und die Größe der Bilder mit eingerechnet wird. Befindet sich die Punktkonzentration beispielsweise bei P_1 (80; 20) und liegt in einem von der Kamera aufgenommenem Bild (320 x 240), so liegt der x-Wert des neuen Punkts an der Stelle:

$$x_{neu} = 80 * (\text{maximale x-Richtung der neuen Ebene}) / 320$$

Die relative Positionierung hingegen geht von der aktuellen Mausposition als Mittelpunkt aus. Das Bild, in dem sich die Punktkonzentration befindet, wird in die andere Ebene hinein projiziert (Abbildung 5.7) und nicht nur der Punkt selbst, wie dies bei der absoluten Positionierung geschieht. Diese Projektion verläuft so, dass der Mittelpunkt des Originalbildes mit der aktuellen Position des Mauszeigers in der neuen Ebene gleich gesetzt wird. Hierüber ist ebenfalls eine Berechnung eines neuen Mauspunktes möglich, wobei der neue Punkt näher am alten Mauszeiger liegt, als es bei einer absoluten Positionierung der Fall ist. Beide Systeme werden innerhalb dieser Studienarbeit getestet und es bleibt dem Nutzer überlassen, welches der beiden Systeme er einsetzt.

Die absolute Positionierung wird in drei unterschiedlichen Ausprägungen implementiert: Zum einen wird das Kamerabild im Verhältnis 1:1 umgesetzt, das heißt dass die Maus nur in einem Rahmen von 320 x 240 Pixeln gesetzt werden kann. Damit wird nur die obere, linke Ecke des Bildschirms angesteuert. Die zweite Möglichkeit ist an die erste angelehnt, nur dass hier eine Bildbreite von 640 x 480 Pixeln angenommen wird. Dies wird vornämlich integriert, um zu testen, wie genau sich diese Abbildung verhält und um Aussagen treffen zu können, wie stark die Ungenauigkeit beim Berechnen der neuen Punkte bei den unterschiedlichen Auflösungen zunimmt. Die letzte Art ist eine Umsetzung auf die aktuelle Auflösung des Monitors. Hierzu wird zunächst die aktuelle Auflösung betrachtet und zwischengespeichert. Damit ist es möglich, die kontaktlose Maus über den ganzen Bildschirm hinweg zu benutzen. Hier ergeben sich jedoch Probleme bei der Positionierung im Randbereich, da hier der Zielpunkt ebenfalls nahe am Rand des aufgenommenen Kamerabildes sein muss.

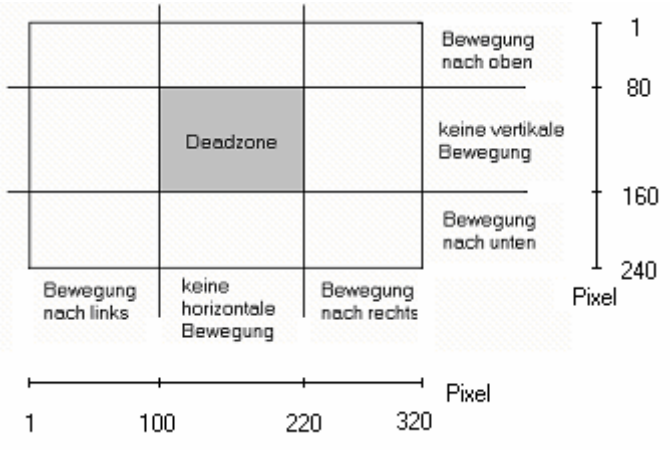

Abb. 5.8: Bereichseinteilung der relativen Positionierung

Bei der relativen Positionierung wird das Kamerabild in neun verschiedene Bereiche unterteilt (siehe Abbildung 5.8). In der Mitte befindet sich eine Art Deadzone, dass heißt befindet sich der Zielpunkt im Kamerasystem in diesem Bereich, findet keine Bewegung statt. Bewegt sich der Zielpunkt in eines der angrenzenden Felder, wird im Verhältnis zur aktuellen Cursorposition die Maus bewegt. Erst wenn der Zielpunkt wieder in den Mittelpunktsbereich geführt wird, wird die Bewegung gestoppt. Je nachdem, ob die absolute oder relative Positionierung angewandt wird, wird ein anderer Mauszielpunkt errechnet. Dieser Punkt enthält dann die Bildschirmkoordinaten, die von der aktuellen Cursorposition erreicht werden sollen.

5.4.3 Bewegen der Maus

Da die beiden Bildschirmpunkte bekannt sind, zwischen denen die Mausbewegung vollzogen werden soll, ist es möglich eine lineare Gleichung aufzustellen (siehe Anhang C Berechnung der Bewegungsgeraden), die alle Punkte zwischen diesen interpoliert. Um unterschiedliche Geschwindigkeiten zu realisieren und ein Springen des Mauszeigers im Ruhezustand, welches durch finden verschiedener Zielpunkte der Maus für sich leicht bewegende Merkmalskonzentration die bei dem *Optical-Flow* entstehen können, zu verhindern sind verschiedene Schwellbereiche integriert. Bei Bewegung kleiner als 1/25 der Bildschirmauflösung, wird der Cursor nicht umgesetzt, was das obige Problem des Springens verhindert. Bei einer Positionsänderung größer als 1/25 und kleiner als 1/10 der Auflösung des Monitors wird der Cursor versetzt. Dazu wird zunächst überprüft, in welche Richtung (x oder y) die größere Bewegung stattfindet, um einen sauberen Bewegungsablauf zu gewährleisten. Danach wird die entsprechende lineare Berechnungsformel für die einzelnen Punkte erstellt und der Cursor in einer Schritten versetz. Ist die Bewegung noch größer, werden fünfer Schritte, für eine schnellere Bewegung, durchgeführt.

5.4.4 Mausklicks

Die Mausklicks werden über die Funktion `mouse_event` realisiert, die Bestandteil der user32.dll ist. Je nach Art der übergebenen Parameter kann ein linker oder rechter Mausklick durchgeführt werden. Ein Mausklick wird über die Findung einer Punktkonzentration (vgl. Kapitel 5.4.1) ermittelt. Es wird ein Flag gesetzt, wenn keine besondere Merkmalskonzentration gefunden wurde. Weiterhin wird gespeichert, zu welchem Zeitpunkt das Flag gesetzt wurde und an welcher Position das Merkmal das letzte Mal aufgetreten ist. Die Position wird deshalb gespeichert, da nicht genau abschätzbar ist, wo die neue Punktkonzentration wieder auftritt. Da man nicht an dieser möglicherweise willkürlichen Stelle klicken will, wird später an der gespeicherten Position der Mausklick durchgeführt.

Über die Zeit, die zwischen dem Fehlen einer Punktkonzentration und dem wieder auftreten einer solcher vergeht, kann mittels eines Time-Shifting-Verfahrens ermittelt werden, welche Art – linke oder rechte – Mausklick durchgeführt werden soll. Befindet sich diese Zeitdauer innerhalb von 500 und 1.000 Millisekunden, wird an den zwischengespeicherten Koordinaten ein einfacher Linksklick durchgeführt. Bei einer Zeitspanne von 1.000 bis 2.500 Millisekunden wird ein Rechtsklick, ebenfalls an der gespeicherten Position, durchgeführt. Alle anderen Zeiten werden ignoriert.

Das Flag wird jeweils beim Finden einer neuen Punktkonzentration, sofern es besteht, zurückgesetzt. Es steht somit für das nächste Mal wieder zur Verfügung, wenn keine Punktkonzentration gefunden wird. Der komplette Funktionsablauf ist in Abbildung 5.9 dargestellt.

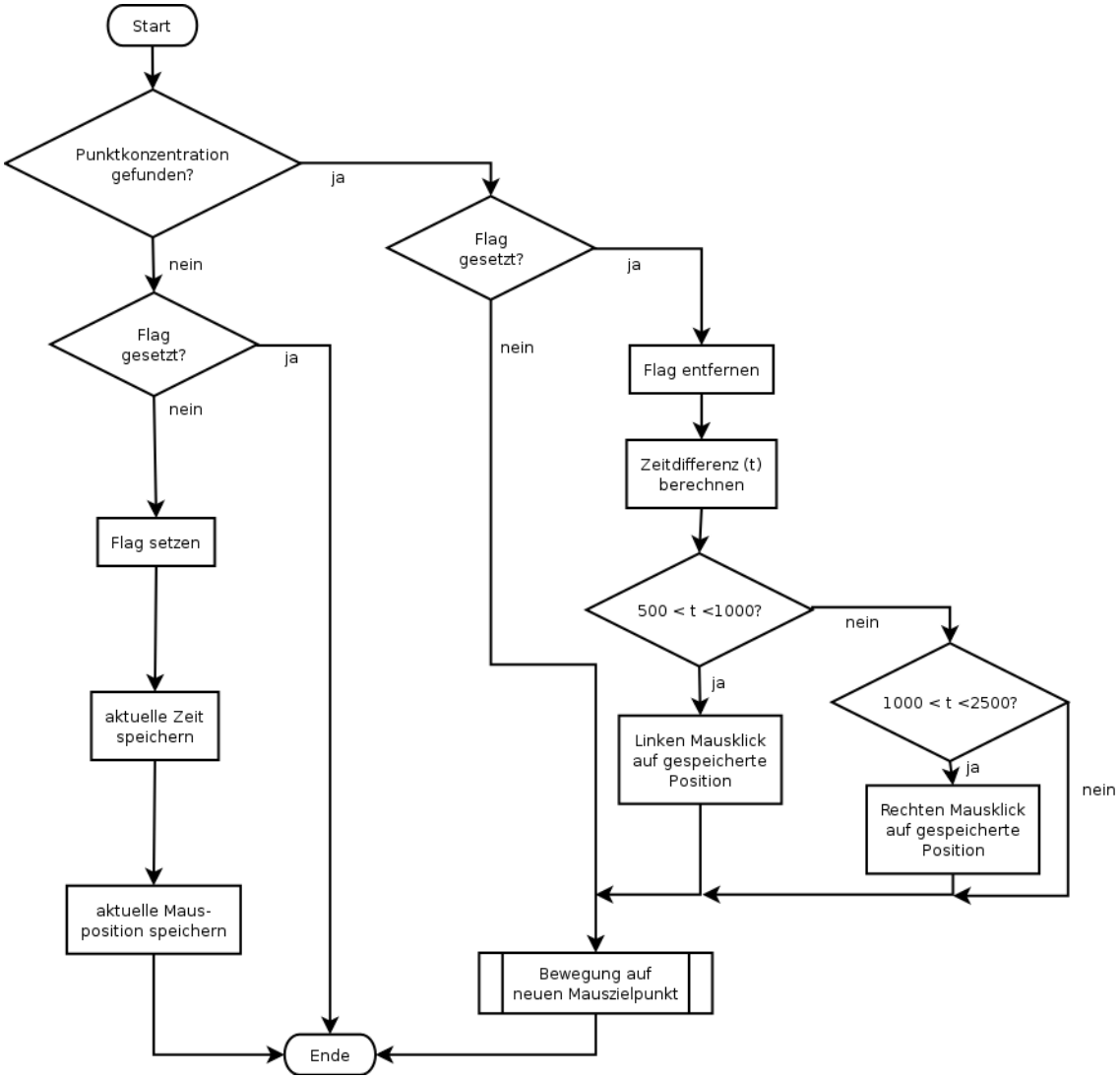

Abb. 5.9: Flussdiagramm der Mausklickproblematik

5.5 *Zusammenführen der Komponenten*

Die einzelnen Komponenten sind ausführlich in den vorhergehenden Kapiteln erläutert worden. Diese müssen nun zusammengeführt werden, um ein komplettes System zu erhalten. Eine schematische Darstellung dieser Zusammenführung ist in Abbildung 5.10 wiedergegeben. Dabei ist das System in drei Threads[13] eingeteilt. Der erste Thread, ist der *Optical-Flow*-Thread, der für die Bildholung und Verarbeitung sowie das Anzeigen des Kamerabildes verantwortlich ist (vgl. Kapitel 5.2 und 5.3). Dieser *Optical-Flow*-Thread kann darüber hinaus, sofern die Option für Mausbewegung aktiviert ist, den Maus-Thread aufrufen. Damit dieser Maus-Thread arbeiten kann, wird diesem beim Aufruf eine Kopie der beiden Arrays, mit den gefundenen Merkmalen aus dem ersten Bild und den wieder gefundenen im zweiten, übergeben. Dieser Maus-Thread ist anschließend für alle Aufgaben, wie sie im Kapitel 5.4 beschrieben sind, zuständig. Dabei ist der Maus-Thread so implementiert, dass er jeder Zeit abgebrochen werden kann, um schnell auf Änderungen während der Mausbewegung reagieren zu können. Der dritte und letzte Thread ist der Haupt-Thread, der für die grafische Benutzeroberfläche und die Koordination der anderen beiden Threads verantwortlich ist.

Um dem Benutzer die Möglichkeit zu geben Einfluss auf das System während der Laufzeit zu nehmen, ist eine Benutzeroberfläche unter Verwendung der Microsoft Foundation Classes implementiert (siehe Abbildung 5.11). Hier ist es möglich, das Programm zu starten und zunächst nur das Kamerabild zur Kalibrierung des Systems anzeigen zu lassen. Diese Kalibrierung ist über einen umfangreichen Einstellungsdialog (siehe unten) möglich. Dieser Dialog ist nicht-modal implementiert, um auch während der Laufzeit auf alle Komponenten gleichermaßen Einfluss zu nehmen. Sämtliche Einstellungen, die vom Benutzer getroffen werden, müssen nach Beendigung und erneuten starten des Programms wieder eingestellt werden. Von einer Speicherung der Einstellungen wurde abgesehen, da man davon ausgehen kann, dass sich die Lichtverhältnisse kontinuierlich ändern, zum Beispiel durch die unterschiedlichen Beleuchtungsverhältnisse, die die unterschiedlichen Tageszeiten mit sich bringen. Es wird empfohlen, nach jedem Programmstart das Kamerabild optimal auf die Beleuchtungsverhältnisse abzustimmen, um so die Findung von Merkmalen im der betrachteten Szene zu unterstützen.

[13] Threads sind in der Software-Architektur Ausführungsstränge, die innerhalb eines Prozesses nebenläufig zu anderen Ausführungssträngen ausgeführt werden. Somit ist eine Quasi-Parallelität von Teilprozessen möglich.

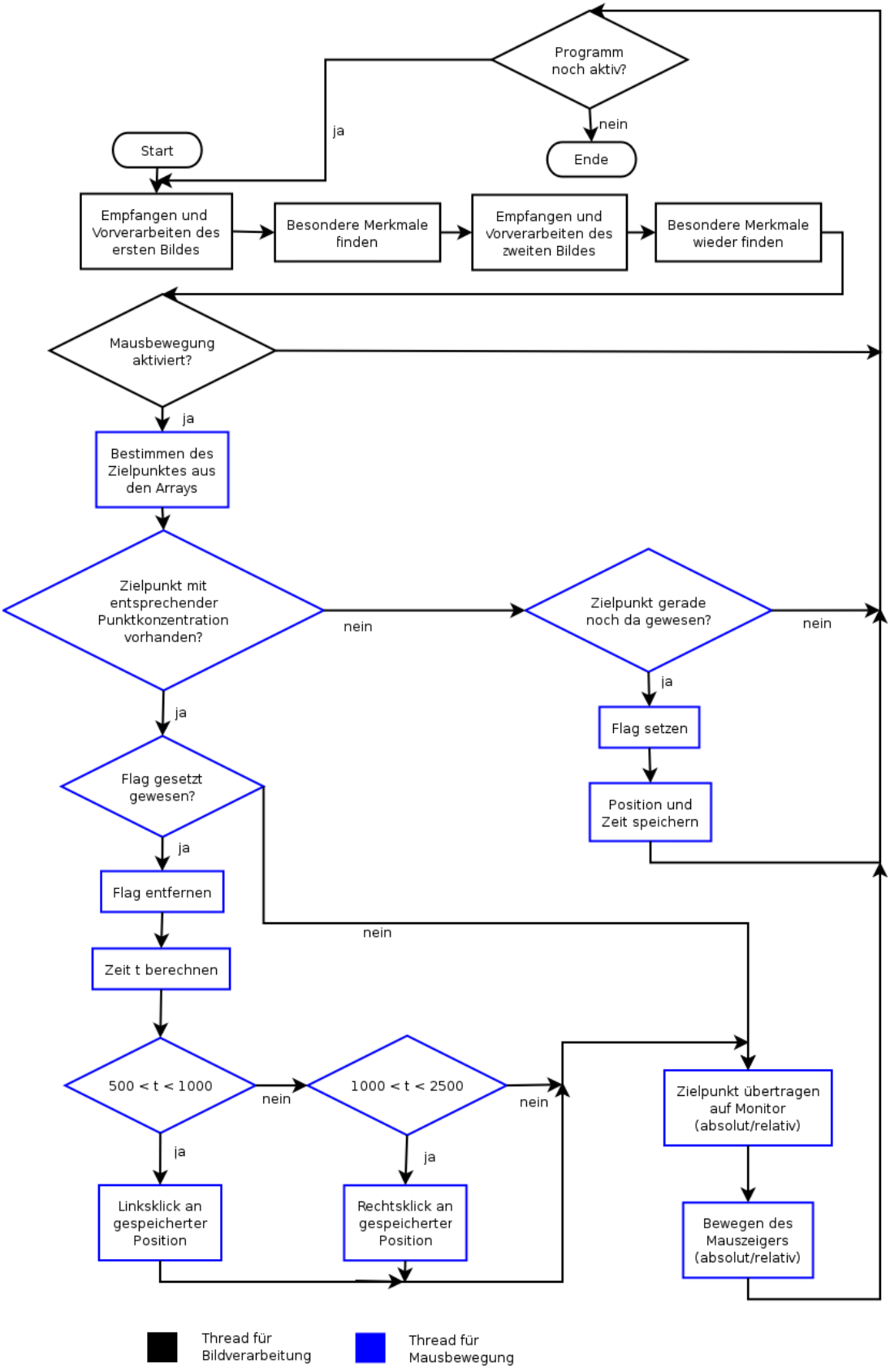

Abb. 5.10: Flussdiagramm des Systementwurfs

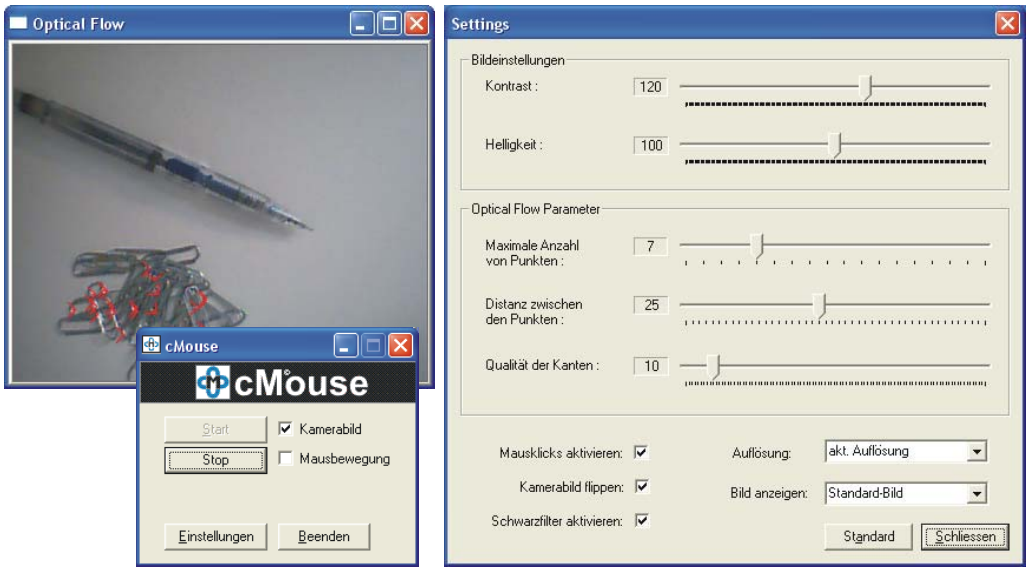

Abb. 5.11: Benutzeroberfläche

Im Fenster „Settings“ finden sich Einstellungen für den Kontrast und die Helligkeit. Die oben erwähnten Grenzwerte für die maximale Anzahl der Punkte, ihre Distanz sowie die prozentuale Qualität der Kanten zur stärksten Ausprägung können hier ebenfalls angepasst werden. Weiterhin ist es hierüber möglich, die Reaktion auf Mausklicks zu aktivieren oder deaktivieren und die vorgestellten absoluten und relative Positionierungsarten zu bestimmen. Der Benutzer kann ebenfalls auswählen, welches Bild er sich anzeigen lassen will. Es stehen das Eigenwertbild, das erste und zweite Graubild, ein Pyramidenbild sowie das normale Standardbild zur Auswahl (vgl. Kapitel 5.3). Weiterhin sind verschiedene Einstellungen der Filterung des Bildes integriert. Die einzelnen Einstellungsmöglichkeiten, werden in Kapitel 6.4 – Aspekte der Realisierung – genauer betrachtet. Die Schaltfläche Standard, stellt alle Einstellungen auf getestete Werte zurück, mit denen es in den meisten Fällen möglich ist zu arbeiten. Für eine bessere Funktion der kontaktlosen Maus, sollten die Einstellungen allerdings je nach Lichteinfall und Umgebung angepasst werden. Hier empfiehlt es sich durchaus ein wenig zu experimentieren, um die optimalen Einstellungen zu wählen.

Über den Hauptdialog ist es anschließend möglich die eigentliche Bewegung der Maus zu aktivieren. Um ein ungestörtes Arbeiten zu ermöglichen, ist das Programm so angelegt, dass es beim Minimieren nicht störend in der Taskleiste ist, sondern sich dezent innerhalb des Infobereichs der Taskleiste (Trayleiste) aufhält. Durch ein Doppelklick auf das Icon ist es möglich, den Dialog zu reaktivieren. Die Funktionsmöglichkeit der kontaktlosen Maus ist während der Minimierung nicht eingeschränkt, so dass man weiter mit dem System arbeiten kann.

6 Aspekte der Realisierung

Bei dem Systementwurf von *cMouse* wurde deutlich, dass eine „kontaktlose Maus“ über das Verfahren der Auswertung von Kameradaten sehr komplexe Anforderungen zu erfüllen hat. Bei den Bildern handelt es sich um große Datenmengen[14], die zunächst empfangen und anschließend entsprechend bearbeitet werden müssen. Dieser Abschnitt soll sich hauptsächlich mit den generellen Rahmenbedingungen der Kamera und den daraus resultierenden Ergebnissen, der Vorverarbeitung der einzelnen Bilder und den möglichen Einstellungen, die vom Nutzer gemacht werden können, beschäftigen.

6.1 Rahmenbedingungen

Um die Auswertung der Kamera-Daten zu unterstützen, sind einige Rahmenbedingengen eingeführt worden. Dabei handelt es sich um eine unifarbene Unterlage sowie einem besonderen Merkmal, welches die Kamera in Verbindung mit dem Algorithmus finden soll.

Da der *Optical-Flow* die gesamten besonderen Merkmale eines Kamerabildes berücksichtigt und somit auch Gegenstände die sich ebenfalls im Bildbereich befinden aber keinen Beitrag zur Findung eines aktuellen Mauspunkts haben, wurde eine Unterlage eingeführt. Hierbei handelt es sich um eine unifarbene schwarze Platte. Die Kamera muss so ausgerichtet werden, dass sie nur diese Platte erfasst, also ein schwarzer Hintergrund im dargestellten Kamerabild erscheint. Das hat zum Vorteil, dass andere Objekte nicht im Bild sind und die Kontur der Hand besser zum tragen kommt.

Für die Unterstützung des *Optical-Flow* ist weiterhin das Anbringen einer ausgeprägten Struktur erforderlich. Die Hand allein birgt nicht genügend scharfe Kanten, so dass der Algorithmus nur wenige Merkmale findet. Die Findung von Punktkonzentrationen ist ohne die Einführung eines Punktes ebenfalls nicht gewährleistet. Deshalb ist die Anbringung eines Punktes (siehe Kapitel 7) unerlässlich. An diesem markanten Punkt ist der *Optical-Flow* anschließend in der Lage, mehrere Kanten zu finden und eine Konzentration von Merkmalen zu verzeichnen, so dass eine Berechnung nach dem oben geschilderten Entwurf möglich ist.

[14] Bei einer Auflösung von 320 x 240 Pixel, einer Bildwiederholrate von 30 Frames pro Sekunde und einer Farbtiefe von 3 Byte (RGB) entstehen Datenmengen von 320 Pixel x 240 Pixel x 30 Bilder pro Sekunden x 24 Bit pro Pixel = 54.000 KBit pro Sekunde. Dies entspricht einer Datenmenge von rund 6,6 Megabyte pro Sekunde, die über den Bus transferiert werden muss.

6.2 Kamera

Bei der Kamera, die bei dem entwickelten System *cMouse* verwendet wurde, handelt es sich um eine Webcam, mit einer Auflösung von 320 x 240 Pixel und einer Framerate von maximal 30 Frames pro Sekunde. Da die meisten Computerbenutzer allgemein eine Monitor-Auflösung von 1024 x 768 Pixel besitzen, ist ein gewisses Problem zu berücksichtigen. Eine neue Merkmalskonzentration im Kamerabild, die nur ein paar Pixel von der alten entfernt ist, resultiert schnell in einen deutlich merkbaren Sprung auf dem Monitor. Beispielsweise würde eine Sprung von fünf Pixeln, was noch relativ klein ist, bei der Umrechnung in einen Sprung von immerhin 16 Pixeln resultieren. Dies ist unzumutbar für den Nutzer. Deshalb wird innerhalb des Algorithmus ein gewisser Schwellwert gesetzt, der abhängig von der aktuellen Auflösung ist. Erst wenn dieser überschritten wird, darf die Bewegung ausgeführt werden. Dies hat zur Folge, dass ein Springen des Cursors bei Stillstand des Merkmals unterdrückt wird. Gleichzeitig wird dafür in Kauf genommen, dass ein genaues „Zielen“ mit dem Cursor erschwert wird. Hier musste ein möglichst effektiver Wert zwischen Zuverlässigkeit, Genauigkeit und Robustheit gefunden werden. Es stellte sich bei praktischen Tests heraus, dass ein Schwellwert der 1/25 der Monitorauflösung beträgt, gerade akzeptabel und für den Nutzer zumutbar ist.

Die geringe Datenrate von 30 Frames pro Sekunde ist ebenfalls problematisch. Das System *cMouse* ist so entworfen, dass für die Findung eines neuen Mauspunktes zwei Bilder verwendet werden, um eine ausreichende Qualität und Robustheit dieses Punktes zu gewährleisten. Hierüber kann auch sichergestellt werden, dass es sich nicht um eine temporäre Mauskonzentration handelt, da die erste Konzentration im zweiten Bild wieder gefunden wurde. Das bedeutet, dass unter optimalen Bedingungen, mit der vorliegenden Kamera, maximal eine Aktualisierung des Mauszeigers alle 66 Millisekunden – 15-mal pro Sekunde – geschieht, so dass keine Echtzeit (normalerweise 24 Bilder pro Sekunde) gewährleistet werden kann. Weiterhin resultiert das in einer nicht so schnellen Reaktion des Systems auf Bewegungsänderungen, jedenfalls im Vergleich zu einer normalen Maus. Es können hier kurze Latenzzeiten – zwischen der getätigten Bewegung und dem sichtbar werden auf dem Monitor – auftreten, die je nach zu Grunde liegendem System mehr oder weniger deutlich werden. Diese Latenzzeit ist bei modernen Rechnern allerdings so gering, dass trotzdem ein ungestörtes Arbeiten möglich ist.

6.3 Bildvorverarbeitung

Der Algorithmus des *Optical-Flow* sowie die Merkmalsextraktion aus dem Kamerabild sind so implementiert, dass sie auf Grauwertbilder angewandt werden. Dieses ist in der Bildverarbeitung so, da die Farbwerte nur einen verschwindend geringen Anteil an Informationen über Kanten, Kontrast und Helligkeit beherbergen. Diese Werte sind viel einfacher aus einem Graubild extrahierbar, außerdem entfällt mit den Farbwerten auch gleich ein erheblicher Teil an Daten, die sonst ebenfalls bearbeitet werden müssten. Aus diesen Gründen, wird das Bild von *cMouse* direkt nach dem Empfangen in ein Grauwertbild umgewandelt, da die verwendete Kamera dafür keine direkte Funktion beinhaltet.

Anschließend wird eine Anpassung des Kontrast und der Helligkeit vorgenommen. Dieses wird über eine Lookup-Table (LUT) realisiert. Dies ist eine homogene Bildpunktoperation, bei der eine Grauwertskala in eine neue Grauwertskala umgeformt wird [HT05]. Der Vorteil, die Anpassung über eine solche Lookup-Table zu realisieren, liegt darin, dass es sich hierbei um eine sehr schnelle Art und Weise der Transformation handelt, da die Transformierung über eine direkte Funktion bestimmt ist. Deshalb können auch einzelne Pixel, ohne Bezug zu ihren Nachbarn, bearbeitet werden. Nachteilig ist allerdings, dass es nicht für alle Transformationen eine Rücktransformation gibt, so dass Bildinformationen verloren gehen können. Die Anpassung des Kontrasts und der Helligkeit haben bei guter Wahl der Parameter den Vorteil, dass starke Konturen hervorgehoben werden und schwache noch mehr zusammen schmelzen, so dass eine Extraktion eines besonderen Punktes, oder das Finden von besonderen Merkmalen an sich, erleichtert wird.

Als weitere Option besteht die möglich eine Anpassung der Grauwerte vorzunehmen. Dieses ist in Form eines Filters für sehr dunkle Bereiche realisiert. Dabei wird direkt auf den Bildpuffer zugegriffen und für jeden einzelnen Pixel der Grauwert bestimmt. Grauwerte befinden sich in einem Bereich von 0 – 255, wobei null für Schwarz und 255 für weiß steht. Ist dieser Grauwert unter einer definierten Schwelle, wird eine Anpassung vorgenommen. Innerhalb des Systems liegt diese Schwelle, bei dem Wert 15. Jeder Pixel, der einen geringeren Grauwert besitzt, wird mit dem Grauwert null – für schwarz – überschrieben. Dieses hat zur Folge, dass Merkmale die sich in diesem Bereich befinden, nicht berücksichtigt werden, und somit bei einer schwarzen Unterlage eine Rauschminimierung vollzogen wird.

6.4 *Einstellungsmöglichkeiten von cMouse*

Die Möglichkeiten des Nutzers auch während der Ausführung des Programms Einfluss zu nehmen, sind vielfältig. Hierzu wurde extra ein Dialog implementiert (siehe Abbildung 6.1), der die neuen Einstellungen sofort weiter propagiert und somit ein schnelles Umstellen und Ändern der Einstellungen ermöglicht.

Im oberen Bereich befinden sich zwei Schieberegler für den Kontrast und die Helligkeit. Diese sind so bereitgestellt, dass sie direkten Einfluss auf die in Kapitel 6.3 vorgestellte Lookup-Table nehmen. Ihr Wertebereich geht dabei von 1 – 200. Ein Wert von 100 stellt das normale Bild bereit, so wie es von der Kamera aufgenommen wird. Je nach Verschiebung kann man hierüber den Kontrast und die Helligkeit verstärken oder abschwächen. Ersichtlich werden diese Änderungen in den Graubildern, die ebenfalls angezeigt werden können und als Grundlage für spätere Berechnungen dienen. Als Standardwerte sind für den Kontrast 120 und die Helligkeit 100 voreingestellt. Das bedeutet, dass die Helligkeit nicht verändert und der Kontrast erhöht wird.

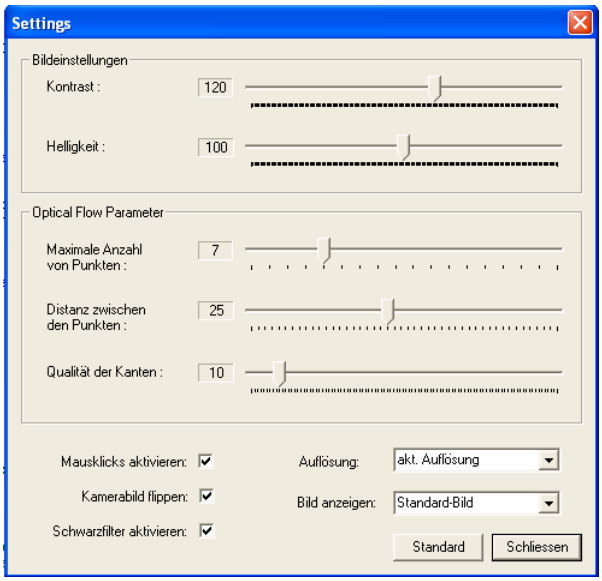

Abb. 6.1: Einstellungsdialog der kontaktlosen Maus

Im zweiten Bereich wird dem Benutzer direkter Zugriff auf einige Einstellungen des *Optical-Flow*-Algorithmus gewährt. Über drei Schieberegler kann man die Anzahl der nötigen Punkte, deren Distanz zueinander sowie die Qualität der Kanten verändern. Diese Einstellungen werden bei der Zielpunktsuche berücksichtigt. Standardmäßig werden sieben Punkte, die sich in einer maximalen Distanz von 25 Pixeln befinden, als Zielpunkt interpretiert. Sollte es mehrerer solcher Punkte geben, die dieses Kriterium erfüllen, wird derjenige mit den höheren Punkten gewählt und der Rest automatisch verworfen. Der Schieberegler für die Qualität der

Kanten ermöglicht es dem Nutzer schon bei der Findung aller Kanten – `cvGoodFeaturesToTrack` vgl. Kapitel 5.3.1 –, deren Qualität zu beeinflussen. Hierbei handelt es sich um einen prozentualen Wert, der mit der Qualität aller stärksten Kanten verglichen wird.

Im unteren Bereich des Dialogs stehen einige generelle Einstellungen zur Verfügung. Hier kann über eine Checkbox bestimmt werden, ob man während der Mausbewegung, die im Hauptdialog ein- und ausgeschaltet werden kann, Mausklicks umgesetzt werden sollen. Diese Einstellung ist standardmäßig aktiviert. Bei Tests, bezüglich der Funktionsfähigkeit der Mausbewegung, empfiehlt es sich dieses Feld zu deaktivieren, da so sichergestellt werden kann, dass keine versehentlichen Mausklicks das Kamerabild in den Hintergrund befördern. Der in Kapitel 6.3 vorgestellte Filter für die Anpassung der Grauwerte, die sich nahe an schwarz befinden, kann hier ebenfalls über eine Checkbox beeinflusst werden. Dies ist für spätere Tests bedeutsam, da so der Nutzen dieses Filters, der möglicherweise nur partiell lohnenswert ist, bestimmt werden kann. Der Filter ist standardmäßig ebenfalls aktiviert. Die Combobox „Auflösung“ beherbergt Einstellungen, die die Positionierungsart des Mauszeigers betreffen (vgl. Kapitel 5.3.2). Hier kann man zwischen aktuelle Auflösung, 320 x 240 Pixel, 320 x 240 Pixel um Maus und 640 x 480 Pixel wählen, wobei ersteres der Standardeintrag ist und eine absolute Positionierung umgerechnet auf die aktuelle Monitoreinstellung darstellt. Bei 320 x 240 Pixel sowie 640 x 480 Pixel handelt es sich ebenfalls um absolute Positionierungen. Im Gegensatz zur aktuellen Auflösung, ist das Arbeitsfenster allerdings nur die obere linke Ecke, mit der genannten Pixelbreite und Pixelhöhe. Bei 320 x 240 Pixel um Maus handelt es sich um eine experimentelle Positionierung, die relativ zum aktuellen Mauszeiger berechnet wird. Als letztes kann man im unteren Bereich Einstellungen bezüglich des anzeigbaren Kamerabildes tätigen. Mit der Checkbox „Kamerabild flippen“, kann man das anzuzeigende Bild horizontal und vertikal gespiegelt anzeigen lassen. Dieses ist vor allem von Bedeutung, da davon ausgegangen wird, dass die Kamera das Bild von oben vorn aufnimmt. Deshalb ist die Einstellung im Normalfall aktiviert, so dass das Bild gedreht wird und der Nutzer es aus seiner Perspektive wahrnehmen kann. In der Combobox „Bild anzeigen“, steht wie in Kapitel 5.5 beschrieben eine Auswahl zwischen verschiedenen Bildern zu Verfügung, von denen einige teils zur Berechnung eingesetzt werden. Hierüber kann einfach geprüft werden, wie sich die einzelnen Schritte des *Optical-Flow* verhalten. Des Weiteren kann man daran die verschiedenen Filter visualisieren und somit ihren Nutzen abschätzen. Das Bild, welches hier standardmäßig angezeigt wird, ist eine Kopie des Referenzbildes des *Optical-Flow*. Zusätzlich werden hierin die Vektoren, die durch die gefundenen Merkmale und ihre wieder gefundenen Äquivalente beschrieben werden, als unterstützende optische Merkmale eingezeichnet.

7 Test und Testergebnisse

Das System *cMouse* wurde mit den oben genannten Eigenschaften und Fähigkeiten komplett implementiert. In diesem Abschnitt werden Tests und deren Ergebnisse beschrieben. Die einzelnen Tests werden benutzt, um abschätzen zu können, wie gut sich das implementierte System für eine kontaktlose Maus auf Basis der Auswertung von Kameradaten über den *Optical-Flow*-Algorithmus verhält. Vor allem werden hier die einzelnen Schwierigkeiten, die bei dem System auftreten, untersucht. Hierzu wird konkreter auf die Übertragungsgeschwindigkeit der Kamera, die einzelnen Filter, die Zielpunkterkennung, die Positionierungsarten und die Mausklicks eingegangen. Aus den so gewonnen Informationen, kann das entwickelte System bewertet und optimiert werden.

Für die Tests werden zwei Systeme verwendet, die folgende Vorraussetzungen haben:

1. Das erste Testsystem basiert auf einem Intel Centrino mit 1,6 GHz Taktfrequenz und einem Frontsidebus von 533 MHz, 512 MB Arbeitsspeicher sowie einer Monitorauflösung von 1.400 x 1.050 Pixeln. Als Betriebssystem kommt Windows XP Home Service Pack 2 zum Einsatz.
2. Das zweite Testsystem besteht aus einem Intel Pentium IV-Prozessor mit HT-Technologie, einer Taktung von 3,0 GHz, einem Frontsidebus mit 800 MHz sowie 512 MB Arbeitsspeicher. Die Monitorauflösung beträgt 1.600 x 1.200 Pixel und als Betriebssystem wird Windows XP Professional Service Pack 2 benutzt.

Zum Einsatz kommt weiterhin die Webcam Sitecom Easycam VP-001, mit einer Auflösung von 320 x 240 Pixel und einer Bildwiederholrate von 30 Bildern pro Sekunde.

7.1 Test der Kamerageschwindigkeit

Die Geschwindigkeit der Kamera ist vom Hersteller mit 30 Bildern pro Sekunde angegeben. Das implementierte System ist so ausgelegt, dass die einzelnen Bilder nicht in einem permanenten Datenstrom abgearbeitet werden. Das Programm holt sich vielmehr in einem eigenen Thread die so genannten Frames, führt die einzelnen Berechnungen darauf durch und gibt die Ergebnisse weiter. Anschließend wird dieser Zyklus neu begonnen und das nächste Referenzbild geholt (vgl. Kapitel 5.3).

Der *Optical-Flow*-Algorithmus, benötigt für seine Abarbeitung allerdings zwei Bilder, so dass nicht jedes einzelne Bild einen Zielpunkt für die neue Mausposition bereitstellt. Wenn man diesen Fakt auf die theoretisch 30 Bilder anwendet heißt das, dass nur 15 Bilder pro Sekunde und somit 15 Punkte extrahiert werden. Echtzeit kann mit diesem System schon auf Grund der Kamera also nicht erreicht werden, da erst ein Rate von 24 Bildern pro Sekunde als bewegt gelten. Die Datenrate der Kamera ist also nur zum Teil ausreichend. Das Programm zeigt bei der visuellen Überprüfung über das Kamerabild eine kurze Verzögerung, die auch beim abschalten aller Mauspositionierungen, Filter und des *Optical-Flow* – also mit einer Framerate von 30 Hz – bestehen bleibt. Diese Verzögerung beträgt teilweise bis zu einer Sekunde, die sich aus dem System also nicht entfernen lässt, da sie auch bei der normalen Arbeit der Kamera besteht.

Mit einer Kamera, die eine höhere Bildwiederholrate gewährleistet, könnte man das System in Zukunft schneller gestalten. Hier muss allerdings berücksichtigt werden, dass eine höhere Bildwiederholrate auch einen größeren Datenstrom verursacht und damit die Bandbreite von USB 1.1 (circa 12 MBit/s) nicht mehr ausreichend ist[15]. Da aber die derzeit verfügbaren Kameras mit 60 Bildern pro Sekunde bereits relativ Preiswert sind – 100 € bis 200 € – kann über die Verwendung dieser, zukünftig sicher ein echtzeitfähiges System noch besser angenähert werden.

7.2 Test der Filter

Die Erklärung der einzelnen Filter ist in Kapitel 6.3 schon ausführlich vorgenommen worden. Bei den Tests ergab sich allerdings, dass einige Filter teils auch negativen Einfluss auf das Gesamtsystem haben können. Dies ist hauptsächlich von verschiedenen Beleuchtungen und das Auftreten von Schatten beeinflusst.

Bei zu schlechter Ausleuchtung der Szene kommt es dazu, dass das aufgenommene Bild eher viele dunkle Grauwerte – das Bild wird sofort in Grauwerte transformiert (vgl. Kapitel 6.3) – aufweist. Dies kann meist mit einer Anpassung des Kontrastes und der Helligkeit behoben werden. Diese Einstellungen müssen aber mit bedacht gewählt werden. Bei zu wenig Kontrast kommt es beispielsweise zum verschwimmen an den Kanten der einzelnen Bildobjekte. Bei zu hohem sind teilweise Punkte hervorgehoben, die eigentlich überhaupt nicht wichtig für die Szene sind. Hierbei können sogar Staub oder andere Verschmutzungen als markanter

[15] Bei einer Auflösung von 320 x 240 Pixel, einer Bildwiederholrate von 60 Frames pro Sekunde und einer Farbtiefe von 3 Byte (RGB) entstehen Datenmengen von 320 Pixel x 240 Pixel x 60 Bilder pro Sekunden x 24 Bit pro Pixel = 108.000 KBit pro Sekunde. Dies entspricht einer Datenmenge von rund 13,2 Megabyte pro Sekunde, die über den Bus transferiert werden muss.

Punkt gefunden werden. Diese Spielen zwar für die Konzentrationsfindung dann meist keine Rolle, sind allerdings störend für das Gesamtsystem.

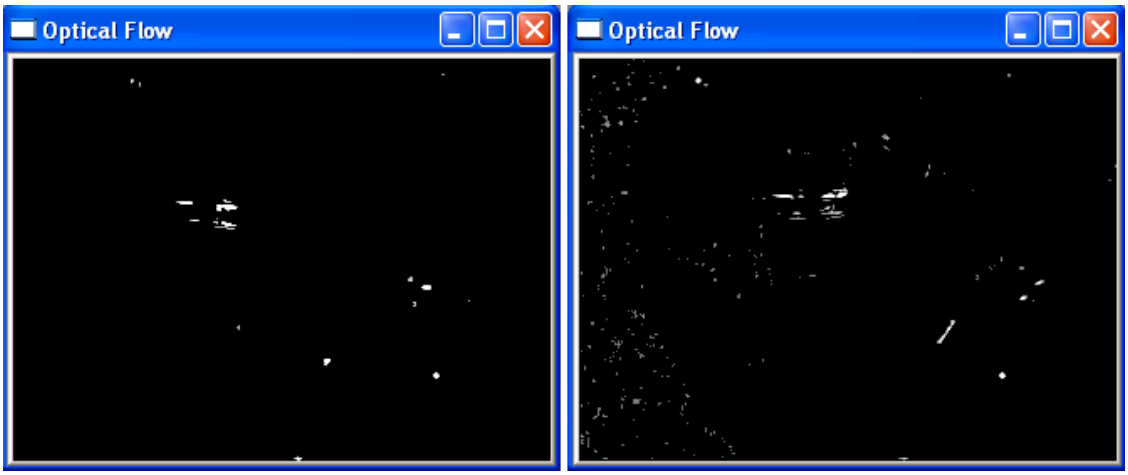

Abb. 7.1: Eigenwert-Bild ohne (links) und mit Schwarzfilter (rechts)

Der oben erwähnte Filter für annähernd schwarze Flächen hat ebenfalls, neben seiner beschriebenen positiven Auswirkung, einige Schwächen. Diese kommt vor allem bei Bewegungen zum tragen (siehe Abbildung 7.1). Im linken Bild ist das Eigenwert-Bild relativ eindeutig, welches zur Findung der besonderen Merkmale in einem Bild dient. Die gleiche Bewegung, mit aktiviertem Filter, zeigt im rechten Bild deutlich mehr Eigenwerte. Diese entstehen vornämlich durch Schatten und sich schnell verändernde Lichtverhältnisse. Man sieht hieran auch, dass das hervorstechendste Merkmal zusätzlich noch ein wenig verwischter ist. Durch die Umrechnung der Grauwerte, die nahe an Schwarz sind, kommt es auch an Kanten bei denen Schatten ist dazu, dass hier ebenfalls größere Kontrastunterschiede errechnet werden – da die sonst fließend dunkler werdenden Schatten komplett Schwarz werden – und so mehr Punkte gefunden werden. Diese werden zwar über die Konzentrationsfindung wieder verworfen, aber in zusätzlichen Schritten erstmal mitgeführt, so dass eine minimal längere Bearbeitungszeit besteht.

Generell lässt sich für die einzelnen Filter damit sagen, dass sie unter den richtigen Lichtverhältnissen gut funktionieren. Bei schlechter werdenden Lichtverhältnissen und prägnanterer Ausprägung von Schatten kommt es allerdings zu mehr Berechnungen. Dadurch entsteht aber kein großer Nachteil für den Nutzer, da durch schnelle Systeme die Berechnungszeit nur minimalen Mehraufwand bedeutet.

7.3 *Test der Zielpunkterkennung*

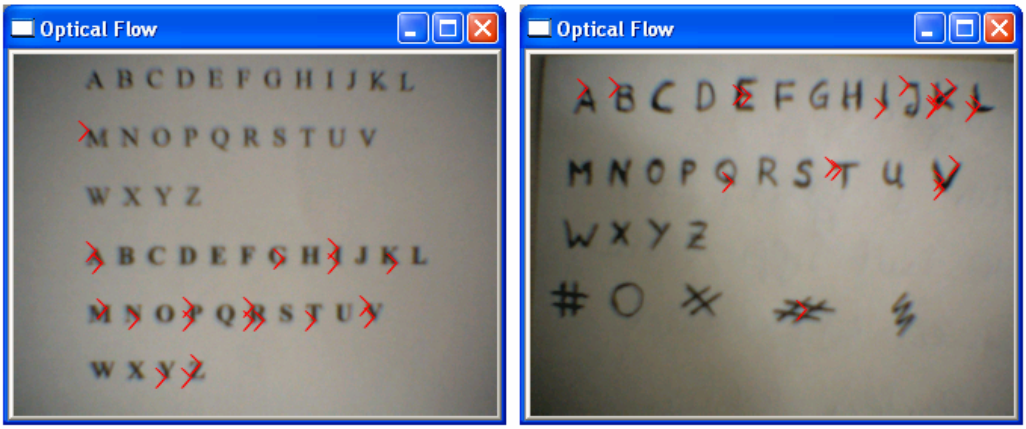

Abb. 7.2: Zielpunkterkennung unterschiedlicher Muster beim Testsystem 1

Die Zielpunkterkennung ist eines der wichtigsten Bestandteile des Algorithmus. Die Hand allein bietet nur wenige markante Punkte oder diese so verstreut, dass es sehr schwer ist, genau einen Punkt zu filtern, der auf den Monitor übertragen werden kann. Deshalb wurden drei Testreihen mit verschiedenen Formen durchgeführt, um herauszufiltern, welche Strukturen sich am besten für eine Verfolgung eignen. Dazu wurde zunächst das Alphabet einmal mit normalen und einmal mit dicken Buchstaben ausgedruckt. Der *Optical-Flow* ist laut seiner Definition besonders auf scharfe Strukturen und scharfe Kanten angewiesen. Das Testsystem 1 liefert dabei folgende Ergebnisse, die in Abbildung 7.2 dargestellt sind. Die Buchstaben A, I, P, R, V und Z wurden besonders deutlich gefunden. Es wird auch offensichtlich, dass eine dickere Struktur der Buchstaben ebenfalls positiven Einfluss auf deren Findung hat. Im zweiten Test wurden handschriftlich Buchstaben des Alphabetes getestet. Zusätzlich wurden noch einige Strukturen untersucht, die theoretische ebenfalls hohe Kantenwerte aufweisen müssten. Hier stellte sich heraus, dass letztere fast gar nicht berücksichtigt wurden. Die Buchstaben selbst zeigen vor allem bei K und V besonders hohe Trefferquoten.

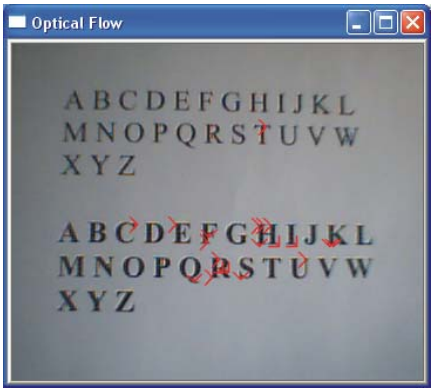

Abb. 7.3 Zielpunkterkennung des Alphabets beim Testsystem 2

Bei dem Testsystem 2 allerdings, stellten sich folgende Ergebnisse dar (siehe Abbildung 7.3). Hier kann man erkennen, dass vorwiegend die Buchstaben F, H, K und R bevorzugt gefunden werden. Bauchige Strukturen, wie B, C, D, O und Q sind hingegen kaum gefunden worden. Es ist also offensichtlich, dass Buchstaben mit besonders eckigen Strukturen aufgefunden werden und eine breitere Ausprägung der Struktur ebenfalls unterstützend wirkt, was die Erkennungswahrscheinlichkeit betrifft. Da bei den Tests doch deutliche Unterschiede aufgetreten sind, sollte jeder Nutzer bei seinem System diesen Test ebenfalls durchführen und anschließend das bei ihm am Besten gefundene Merkmal benutzen.

Erkennungswert	**Testreihe**		
	1 (Abb. 7.2 links)	**2** (Abb. 7.2 rechts)	**3** (Abb. 7.3)
sehr gut	A, M, I, P, R, V, Z	E, K, L, T, V	F, H, K, R
befriedigend	G, K, N, T, Y	A, B, I, J, Q	C, E, I, Q, S, T, U
kein	B-F, H, J, L, O, Q, S, U, W, X	C, D, F-H, M-P, R, S, U, W-Z	A, B, D, G, J, L, M-P, V-Z

Tabelle 1: Übersicht über die verschiedenen Testreihen mit Testergebnissen

Ein Grund für die unterschiedlichen Testergebnisse liegt am Fokusbereich der Kamera. Die Stelle des schärfsten „Sehens“ der Kamera liegt in der Objektivmitte und nimmt zum Rand hin ab (von Kamera zu Kamera unterschiedlich). Deshalb werden Buchstaben, die sich näher an der Mitte des Bildes befinden besser detektiert. Ein weiterer Grund ist darin zu sehen, dass die Kamera nicht optimal auf das zu betrachtende Objekt eingestellt ist und das empfangene Bild somit unscharf und verschwommen ist. Aus einem unscharfen Bild kann der *Optical-Flow*-Algorithmus nur schwierig besondere Merkmale extrahieren, da keine bzw. wenige scharfe Kanten und hohen Kontrastdifferenzen vorhanden sind. Dies sind auch die Hauptgründe, weshalb zum Beispiel der Buchstabe X, der einer der kantenreichsten Buchstaben des Alphabetes ist, nicht erkannt wird.

Weiterhin wurde getestet, ob das System sich auch mittels eines Laserpointers bedienen läst. Dieser Test ergab, dass wenn man die zu findende Konzentration verringert, auch dieser sehr feine Punkt gefunden wird und eine gute Steuerung des Systems möglich ist. Die Mausklicks werden hierdurch allerdings erschwert, da man den Laserpointer hierfür nur sehr kurz ausschalten darf. Eine Anpassung der Werte, wie lange der Punkt mindestens und maximal Verschwunden sein darf, um als Mausklick interpretiert zu werden, bietet sich hier an. Ebenfalls nachteilig wirkte sich bei diesem Test die umständliche Handhabung des Laserpointers aus. Dies ist darauf zurück zu führen, da der Punkt direkt von oben auf die betrachtete Unterlage projiziert werden muss und eine hohe Konzentration dem Nutzer abver-

langt, weil ein ruhiges halten des Punktes erforderlich ist, um die Maus ebenfalls ruhig an einer Position zu halten.

Bei der Zielpunkterkennung stellte sich weiterhin heraus, dass Objekte, die sich ebenfalls im Bild befinden, ablenkend für den *Optical-Flow* wirken. Wenn diese Objekte starke Strukturen oder hohe Kontraste aufweisen, kann es passieren, dass diese als Zielpunkte identifiziert werden. Deshalb wird eine unifarbene Unterfläche als Unterlage definiert, über welche allein die Hand als Objekt gesetzt werden darf. Bei Tests wurde hierzu eine schwarze Platte benutzt. Diese Platte hat je nach Beleuchtung allerdings die Eigenschaft leichte Reflexionen wiederzugeben, so dass ein gewisses Rauschen innerhalb des Bildes immer noch vorhanden ist. Dies hat zum einen den Vorteil, dass trotz der Restriktion der unifarbenen Unterfläche, das System auf kleinere Störungen getestet werden konnte. Der Nachteil liegt darin, dass so wieder Störungen ins System gelangten, die den Idealfall verhinderten. Es zeigte sich allerdings, dass das System doch relativ robust auf diese Problematik reagiert. Das Rauschen wird teils gefiltert und spätestens bei der Suche nach einer Konzentration von Punkten eliminiert. Das System ist unabhängig von der Farbe des Untergrundes, da dass besondere Merkmal an der Hand befestigt ist und über die genannten Eigenschaften immer die höchste Konzentration aufweist.

7.4 Test der Positionierungsarten

Innerhalb des Systems sind zwei verschiedene Positionierungsarten implementiert. Es werden hier die absolute Positionierungsart, die auf die aktuelle Auflösung abgebildet wird, und die relative Positionierungsart bei beiden Testsystemen überprüft (vgl. Kapitel 5.4.2).

Die absolute Positionierungsart hat den Vorteil, dass direkt auf die Bewegung reagiert wird. Ein schnelles wechseln von Ecke zu Ecke ist beispielsweise möglich, da der neue Mauspunkt direkt aus dem Bild berechnet wird. Probleme ergeben sich bei dieser Art und Weise in zwei Bereichen. Zum einen kann es bei einer sehr hohen Monitorauflösung dazu kommen, dass der Mauszeiger springt. Dies wurde verhindert, indem ein Bereich definiert wurde, bei dem keine Bewegung stattfindet. Ursache für dieses Problem ist vor allem die Umrechnung von dem kleinen Kamerabild zur relativ hohen Monitorauflösung. Der genutzte Kompromiss, einen nahen Bereich bei der Bewegung zu ignorieren, führt dazu, dass ein genaues Zielen auf Schaltflächen oder der gleichen erschwert wird. Das zweite Problem ist der Kantenbereich bei dem System. Da es eine absolute Positionierungsart ist, ist das Ansteuern der Monitorkanten besonders schwierig, da nur wenn das besondere Merkmal ebenfalls sehr nah an einer Kante des Kamerabildes ist, dieser Fall ein-

tritt. Es kann dabei sehr leicht passieren, dass man aus dem Aufnahmebereich der Kamerabild heraus geht.

Die relative Positionierungsart, ist im Gegensatz zur absoluten Positionierung, besonders gut was die Arbeit im Kantenbereich angeht. Dahingegen ist es relativ schwierig im normalen Bereich mit dem System zu arbeiten. Dies beruht hauptsächlich auf die sehr ungewohnte Art der Positionierung. Weiterhin erschwert die hohe Verzögerung der Kamera diese Positionierungsart, da man beispielsweise bei einer Rechtsbewegung den markanten Punkt bereits früher wieder in der Deadzone haben muss, als er an seinem Ziel angelangt ist. Dieser Umstand kann vor allem mit ein wenig Übung wettgemacht werden. Teilweise ist mit diesem System ein deutlich leichteres anspringen einzelner Schaltflächen verzeichnet worden. Durch die schwierige und teils herausfordernde Handhabbarkeit, kann es einigen Menschen allerdings lieber sein, die relative Bewegung zu nutzen. Deshalb werden beide Arten im System erhalten und dem Nutzer die freie Wahl darüber gelassen, welches System eingesetzt werden soll.

7.5 Test der Mausklicks

Die Mausklicks sind über eine Art Time-Shifting-Verfahren implementiert (vgl. Kapitel 5.4.4). Bei schlechter Ausleuchtung der Szene oder zu schnellen Bewegungen kann es dazu kommen, dass kurzzeitig kein Merkmal gefunden wird. Wenn nun das Merkmal in einer der angegebenen Zeiten wieder erscheint, wird automatisch ein Mausklick ausgeführt. Da dieses in diesem Fall nicht gewollt ist, ist, wirkt diese Eigenschaft sehr störend auf das System. Dieser Fall kann beispielsweise auch eintreten, wenn der Nutzer das Kamerabild mit seiner Hand für nur sehr kurze Zeit verlässt. Über die Bedingung, dass das Merkmal länger unauffindbar sein muss, als die in Kapitel 5.4.4 implementierten Werte, könnte man dieses Problem lösen. Auf der anderen Seite ist es auch wieder sehr störend, wenn man für die Ausführung eines einfachen Mausklicks mehr als zwei Sekunden benötigt, da man bessere Werte von einer normalen Maus gewohnt ist.

Eine Eigenschaft der Mausklicks, stellte sich allerdings bei den Tests als besonders Positiv heraus. Da sich die Position gemerkt wird, an der der Mauszeiger verschwunden ist, kann ein Mausklick deutlich präziser durchgeführt werden. Ohne diese Funktion war es so, dass wenn man ein Mausklick auf einer Schaltfläche auslösen wollte, diese oft verfehlt wurde. Das war darauf zurückzuführen, dass manchmal das Merkmal einige Pixel neben der Schaltfläche wieder detektiert wurde und somit nicht auf die Schaltfläche der Mausklick durchgeführt wurde. Da nun aber die Werte mitprotokolliert werden, an dem das Merkmal verschwindet, muss nur noch die Schaltfläche getroffen werden. Egal wo das Merkmal erneut

auftritt, an der Position des Verschwindens wird der Mausklick durchgeführt. Die Genauigkeit der Mausklicks konnte allein über die Verwendung der gespeicherten Positionswerte und deren Benutzung für die Mausklicks um ein vielfaches gesteigert werden.

7.6 Test des Gesamtsystems

Die Performance des Gesamtsystems ist ein wichtiger Indikator zur Beurteilung des Systementwurfs. Folgende Systemauslastungen wurden mit Hilfe des Windows Task-Managers ermittelt. Dabei wurde darauf geachtet, dass keine Störeinflüsse und somit Systemressourcen, zum Beispiel durch andere geöffnete Applikationen entstehen. Bei mehreren Tests ergab sich, dass bei dem oben erwähnten Testsystem 1 die Kamera alleine, das heißt ohne das Starten des Programms, eine prozentuale Auslastung von 15% bis 30% aufweist. Bei laufendem Programm verursacht die Kamera allein eine Systemlast von 25%. Diese Werte sind bei sehr guter Ausleuchtung des Systems entnommen worden. Bei schlechterer Ausleuchtung oder dunklerer Umgebung benötigt die Kamera weniger Systemressourcen, liefert allerdings auch weniger Bilder. Beim zweiten Testsystem, welches eine deutlich schnellere CPU mit HT-Technologie besitzt und dadurch mehr Ressourcen zur Verfügung stellen kann, wurde eine geringere Systemlast gemessen. Die Messwerte lagen hier durchschnittlich bei 2%, maximal bei 5%. Die Implementierung selbst benötigt ebenfalls je nach Ausleuchtung der Szene und der Systeme unterschiedlich viele Systemressourcen (CPU-Zeit). Da bei guter Beleuchtung allgemein mehr Daten in Form von zu verarbeitenden Bildern anfallen. Maximale Werte für das Programm liegen bei 20%, im Schnitt bewegt sich dieser Wert allerdings eher um die 15% Systemauslastung bei Testsystem 1, Testsystem 2 hat eine CPU-Auslastung von maximal 10% . Durchschnittlich gesehen, verursacht die komplette Implementierung samt Kamera also eine 40%-ige Systemlast beim Ersten und 12% beim zweiten Testsystem, bei idealen Bedingungen. Diese Auslastung ist relativ niedrig, verglichen mit dem vorgestelltem System *Nouse* (vgl. Kapitel 3.2.2), dessen Werte hier meist bei 90% für das Erste und 55% beim zweiten Testsystem liegen.

	Testsystem 1	**Testsystem 2**
Systemauslastung durch Kamera	▪ 15% - 30% ▪ ø 23%	▪ 2% - 5% ▪ ø 3%
Systemauslastung durch Programm	▪ 12% - 20%	▪ 6% - 10%
Gesamtsystemauslastung	▪ 27% - 50%	▪ 8% - 15%

Tabelle 2: Vergleich der erzeugten Systemlasten der unterschiedlichen Testsysteme

Die Systemlast lässt sich wie folgt erklären. Zum einen werden hier große Datenmengen verarbeitet. Das Kamerabild an sich ist jeweils 225 kB groß, was bei einer anhaltenden Datenrate relativ viel ist. Hinzu kommt, dass *cMouse* drei Threads benötigt, um zu arbeiten. Der Haupt-Thread ist für die graphische Benutzeroberfläche verantwortlich, welcher nur wenige Systemressourcen benötigt. Der *Optical-Flow*-Thread für das Holen der Bilder und Verarbeiten des *Optical-Flow*-Algorithmus benötigt die meisten Ressourcen. Der Maus-Thread, welcher zuständig für die Berechnung des neuen Mauspunktes, die Bewegung und die Mausklicks ist, benötigt weniger CPU-Zeit gegenüber den anderen beiden Threads. Dadurch, dass diese Threads sich auf einem Ein-Prozessor-System ständig abwechseln, entsteht ebenfalls eine Verzögerung.

7.7 Fazit der Testergebnisse

Allgemein lässt sich sagen, dass das Gesamtsystem doch teilweise stark von der Ausleuchtung abhängig ist. Es werden zwar auch bei schlechter Beleuchtung noch befriedigende Ergebnisse erreicht, da das System sehr robust implementiert wurde, es macht sich allerdings doch bemerkbar. Der Faktor der Umrechnung von Kamera- zur Bildschirmauflösung ist ebenfalls nicht unerheblich, zu mal das zielgenaue Ansteuern einzelner Schaltflächen somit ein wenig Geschick erfordert. Die Implementierung der Mausklicks ist bei genügender Ausleuchtung ebenfalls sehr gut. Bei schlechter Beleuchtung kommt es allerdings vereinzelt zu ungewollten Mausklicks, was für den Benutzer frustrierend sein kann. Durch die umfassenden Einstellungen, die der Nutzer auswählen kann, können einige dieser Probleme behoben werden. Diese müssen dann je nach Umgebung allerdings individuell vorgenommen werden.

8 Zusammenfassung und Ausblick

Innerhalb dieser Studienarbeit wurden viele Aspekte der Maus und neuerer darauf aufbauender Technologien betrachtet. Die Gegenüberstellung aktueller Forschungsarbeiten und verschiedene Untersuchungen zur Umsetzbarkeit eines dieser Konzepte ergaben, dass innerhalb dieser Studienarbeit eine Implementierung einer kontaktlosen Maus auf der Grundlage von Kameradaten angestrebt wurde. Dieses System wurde gewählt, da es eines der interessantesten Bereiche ist und diese Thematik viele verschiedene Möglichkeiten zu seiner Umsetzung zulässt. Diese Ansätze betreffen vor allem die verschiedenen Arten, wie ein Bild ausgewertet werden kann. Beispiele für das Finden von Objekten oder das Verfolgen von Punkten wurden hierzu erläutert. Weiterhin bieten die verschiedenen Probleme, wie unterschiedliche Beleuchtungen der Szene oder die Möglichkeit Mausklicks zu realisieren etc., viele unterschiedliche und komplexe Lösungsmöglichkeiten, was den Reiz dieser Thematik erhöhte.

Durch die Implementierung von *cMouse*, konnten weitere sehr gute und umfassende Einblicke in diese Problematik gewonnen werden. Der Programmentwurf unter Verwendung des *Optical-Flow*-Algorithmus zog ein sehr robustes, erweiterbares und flexibles System nach sich. Dieses System ist in der Lage, selbst mit einer preiswerten und damit qualitativ durchschnittlichen Webcam ein Ergebnis zu liefern, welches mit der Bewegung einer normalen Maus vergleichbar ist. Es ist hierüber möglich die Maus auf verschiedenste Arten zu steuern, Mausklicks durchzuführen und sich die einzelnen Schritte des *Optical-Flow* anzeigen zu lassen.

Des Weiteren konnten über die Implementierung von *cMouse* viele Tests unternommen werden. Diese Untersuchungen waren teils allgemein, wie die Problematik der Ausleuchtung, und sind damit auch auf andere kameraunterstützte Systeme anwendbar, und teils speziell auf den vorgestellten und implementierten *Optical-Flow*-Algorithmus gestützt. Hier wurden vor allem unterschiedliche Positionierungsarten sowie der Einfluss der Kamera untersucht und unterschiedliche Methoden und Filter für eine Optimierung erarbeitet.

Aus zeitlichen Gründen, konnte das System allerdings nur über diese eine Methode implementiert werden. Es sind viele weitere teils sehr unterschiedliche Ansätze denkbar. Eine kontaktlose Maus könnte zum Beispiel auch über die Auswertung von Farbe in einem Bild oder gar einer kompletten Handerkennung realisiert werden, was jeweils neue Vor- und Nachteile mit sich bringt.

Die Implementierung über Farb- oder Helligkeitswerte ermöglicht einen interessanten Ansatz. Wenn man dem Nutzer die Auflage macht eine markante Farbe zu benutzten und diese aus dem Bild herausfiltert, kann man hierüber ebenso einen Punkt gewinnen, der als neuer Mauspunkt dienen kann. Problematisch an diesem Konzept sind allerdings wieder die hohen Datenmengen, da die Farbinformationen des Bildes weiter mitgeführt und ausgewertet werden müssen. Weiterhin könnten verschiedene Tests durchgeführt werden, ob sich ein Farbton besonders gut zur Verfolgung eignet, wofür genauere Untersuchen erforderlich sind.

Eine Umsetzung über eine komplette Handerkennung und anschließende Filterung des nördlichsten Punkts, ist ebenfalls denkbar. Das größte Problem hierbei ist allerdings die komplexe Erkennung der Hand. Weiterhin muss sich ein komplett anderes System für die Mausklicks überlegt werden. Reizvoll an diesem Aspekt ist vor allem der Punkt, dass man hierdurch komplett unabhängig von einem zusätzlichen Merkmal ist und somit jeder ad hoc mit dem System ohne jegliche Vorbereitungen arbeiten könnte.

Ein anderer Optimierungs- beziehungsweise Arbeitszweig könnte auch in der Verbesserung der Mausbewegung liegen. Derzeit wird die Mausbewegung durch Funktionsaufrufe getätigt. Wenn man die maschinennahste Programmiersprache Assembler für diese Problematik benutzt und nicht mehr C++, könnte theoretisch eine noch saubere und schnellere Form der Mausbewegung realisiert werden. Ob dieses über Assembler umsetzbar und wirklich vorteilhafter ist, müsste in diesem Zusammenhang allerdings erst untersucht werden.

Allein bei der Betrachtung dieser drei Arbeitsbereiche wird offensichtlich, dass das System einer kontaktlosen Maus über die Auswertung von Kameradaten noch lange nicht erschöpft ist. Es gibt viele Ansätze und Möglichkeiten, die weitere Forschungen und Erarbeitungen auf diesem Gebiet ermöglichen und erfordern. Deshalb wird bis zu einer allgemeinen Anwendung im PC-Bereich noch viel Zeit vergehen. Gerade dieser Punkt ist und sollte eine Herausforderung und ein Ansporn für weitere Projekte dieser Art sein.

Literatur- und Quellverzeichnis

[AY02] AMIT, Y.: *„2D Object Detection and Recognition Models, Algorithms, and Networks"*, The MIT Press, Cambridge Massachusetts, 2002.

[BIB02] BIBLIOGRAPHISCHES INSTITUT & F. A. BROCKHAUS AG. Mannheim, 2002.

[BJY99] BOUGUET J.-Y.: *„Pyramidal Implementation of the Lucas Kanade Feature Tracker."*, Intel Corporation, Microprocessor Research Labs. OpenCV Documents, 1999.

[FB02] BAUMGARTNER, F.: *„Piezoresistive Beschleunigungssensoren"*, Hochschule für Technik Buchs, Buchs (Schweiz), 2002.

[HKHV04] HALAMA, C.; KISTEL, T.; HANSCHE, S.; VENSKE, N.: *„Virtual Reality. Datenhandschuh und Datentastatur."*, Technische Fachhochschule Wildau - Bereich Telematik, Wildau, 2004.

[IS05] VOSSEN, P.H.: *„Einführung in Interaktive Systeme - Begriffsdefinition, Historie und Entwurfsprinzipien"*, Vorlesungsskript Interaktive Systeme, Mannheim, 2005.

[KWT87] KASS, M.; WITKIN, A.; TERZOPOULOS, D.: *„Snakes: Active Contour Models"*, IEEE Proceedings of International Conference on Computer Vision, S. 259-268, London, 1987.

[LK81] LUCAS, B.; KANADE, T.: *„An Iterative Image Registration Technique with an Application to Stereo Vision"*, Proc. of 7th International Joint Conference on Artificial Intelligence (IJCAI), S. 674-679, Vancouver, British Columbia, 1981.

[MM00] MEKHAIEL, M.: *„Videounterstützte, robuste Merkmalsextraktion für die Spracherkennung in Echtzeit."*, Dissertation an der Fakultät für Elektrotechnik der Universität Fridericiana Karlsruhe, S. 38-48, Karlsruhe, 2000.

[SC04] SIEGL, C.: „*Erstellen eines Messprogramms für einen mobilen Belastungsprüfstand*“, Bericht des 3. Praxissemesters, DLR Braunschweig, 2004.

[SD05] STAVENS, D.: „*The OpenCV Library: Computing Optical Flow*”, Stanford Artificial Intelligence Lab - CS223b Project, Stanford, 2005.

[ST03] SIEGL, C.; TAPPERT, E.: „*Die PC-Maus: Arten und Funktion*“. Referat für Praktische Datenverarbeitung, Mannheim, 2003.

[ST94] SHI, J.; TOMASI, C.: „*Good Features to Track.*”, IEEE Conference on Computer Vision and Pattern Recognition (CVPR94), Seattle, 1994.

Internetreferenzen:

[BR04] BAYRISCHER RUNDFUNK: „*Wissen und Bildung. Joystick, Maus & Co. Mit der Macht der Gedanken.*”,
URL: br-online.de/wissen-bildung/thema/mensch-computer/hirn.xml
Aktualisierungsdatum: 29.09.2004

[BW04] BEEPWORLD: „*Daniel’s 1st Homepage. Maus.*”.
URL: http://www.beepworld.de/members7/daniel21/maus.htm
Aktualisierungsdatum: 17.12.2004

[CT04] COMPUTER-TUTORIAL.DE: „*Tastatur und Maus.*“,
URL: http://www.computer-tutorial.de/inout/tastmaus.html.
Aktualisierungsdatum: 15.12.2004

[FF05] FRAUNHOFER FIRST „*Intelligente Datenanalyse. Projekte BCCI*”,
URL: http://www.first.fraunhofer.de/bbci
Aktualisierungsdatum: 08.05.2005

[FW05] FISCHER-WICKENBURG, U.: „*EyeToy sieht nun auch dreidimensional.*”,
URL: diepresse.com/Artikel.aspx?channel=&ressort=hg&id=475631
Aktualisierungsdatum: 11.04.2005

[GI04] GYRATION, INC.: *„Gyration Ultra Cordless Optical Mouse"*,
URL: http://www.gyration.com/ultra.htm
Aktualisierungsdatum: 20.09.2004

[HT05] HERMES, T..: *„Bildverarbeitung. Vorverarbeitung."*,
URL: http://www.tzi.de/~hermes/lectures/ss05/21.04.05.pdf
Aktualisierungsdatum: 21.04.2005

[HU05] NEWTON, I.: *„Lex II"*, URL: http://www.culture.hu-berlin.de/ck/lehre/seminare/relativitaet/newton.html
Aktualisierungsdatum: 30.05.2005

[IC04] INTEL CORPORATION: *„Open Source Computer Vision Library"*,
URL: http://www.intel.com/research/mrl/research/opencv/
Aktualisierungsdatum: 27.09.2004

[KW03] KNUTZON, J.; WALTER, B.: *„Object Recognition in Image Processing"*,
URL: ttp://www.public.iastate.edu/~knutzonj/ee424projectMain.htm
Aktualisierungsdatum: 03.12.2003

[PGU05] PLAYSTATION2 GAMING-UNIVERSE: *„Screengalerie"*, URL:
http://playstation2.gaming-universe.de/screengalerie/18854.jpg
Aktualisierungsdatum: 08.02.2005

[PVT04] PERCEPTUAL VISION TECHNOLOGY AT THE NATIONAL RESEARCH COUNCIL: *Nouse "Nose as mouse"*,
URL: http://www.cv.iit.nrc.ca/research/Nouse/index2.html
Aktualisierungsdatum: 24.09.2004

[S4P05] STATIC 4PLAYERS: *„Screenshots"*, URL:
http://static.4players.de/premium/Screenshots/49/f9/122194-medium.jpg
Aktualisierungsdatum: 18.05.2005

[SF05] SOURCEFORGE, *„Project: Open Computer Vision Library: Summary"*,
URL: http://sourceforge.net/projects/opencvlibrary/
Aktualisierungsdatum: 27.05.2005

[SI04] SONY INC., Deutschland: *„EyeToy"*,
URL: http://www.eyetoy.com/german/index.html
Aktualisierungsdatum: 19.10.2004

[TEG04] FRAUNHOFER TEG: „*Beschleunigungsmaus*", URL: http://www.teg.fraunhofer.de/german/projekte/index54.html Aktualisierungsdatum: 26.02.2004

[WDE05] WISSEN.DE: „*Exoskelett*", URL: http://www.wissen.de Aktualisierungsdatum: 28.05.2005

[WE05] WIKIPEDIA: „*EyeToy*", URL: http://de.wikipedia.org/wiki/EyeToy Aktualisierungsdatum: 25.03.2005

[WE05a] WIKIPEDIA: „*Java Standard: Einleitung*", URL: http://de.wikibooks.org/wiki/Java_Standard:_Einleitung#Geschwindigkeit_von_Java-Programmen Aktualisierungsdatum: 15.05.2005

[WE05b] WIKIPEDIA: „*Microsoft Foundation Classes*", URL: http://de.wikipedia.org/wiki/Microsoft_Foundation_Classes Aktualisierungsdatum: 24.05.2005

[WE05c] WIKIPEDIA: „*Eigenwertproblem*", URL: http://de.wikipedia.org/wiki/Eigenwert Aktualisierungsdatum: 23.05.2005

[WE05d] WIKIPEDIA: „*RSI-Syndrom*", URL: http://de.wikipedia.org/wiki/RSI-Syndrom Aktualisierungsdatum: 23.05.2005

[YK05] KURITA, Y.: „*Snake. OpenCV*", URL: robotics.aist-nara.ac.jp/~yuuich-k/HowToIpl-euc/node65.html Aktualisierungsdatum: 02.05.2005

Anhang

A OpenCV-Einstellungen für Visual-Studio 6.0

In diesem Abschnitt werden ein paar Einstellungen beschrieben und dargestellt, um die Verwendung der Bibliothek *OpenCV* einfacher zu gestalten. Nachfolgend werden die wichtigsten Änderungen, sowohl Einstellungen die das Betriebssystem betreffen, als auch Einstellungen in der Entwicklungsumgebung Visual Studio 6.0 beschrieben und dargestellt. Für die Programmierung und Verwendung von *OpenCV* ist es wichtig die benötigten Module einzubinden. Dies geschieht folgender Maßen:

Unter dem Menüpunkt Extras und Optionen kann man Einstellungen für die gesamte Entwicklungsumgebung festlegen. So auch die Verzeichnisse in denen sich die Header-Dateien, Bibliotheken und ausführbare Dateien liegen. Dazu wählt man im Fenster Optionen den Karteikartenreiter Verzeichnisse aus und gibt hier unter den jeweiligen Rubriken die Verzeichnisse an, in denen sich die *OpenCV*-Module befinden, die für die Programmierung benötigt werden (siehe Abbildung A1).

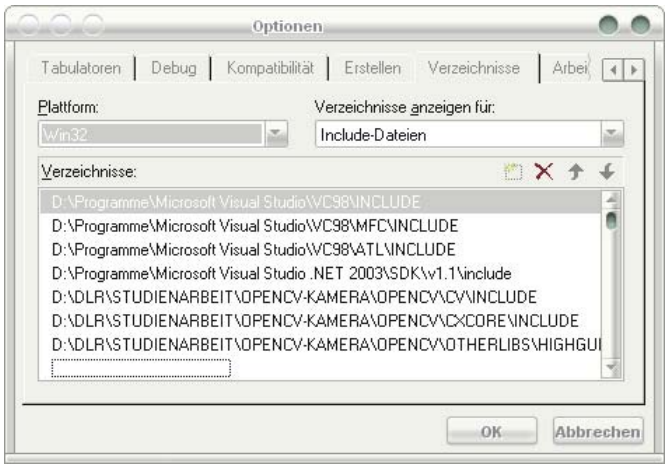

Abb. A1: Optionsmenu Verzeichnisse von Visual C++ 6.0

Nachfolgend müssen diverse Einstellungen für das verwendete Projekt vorgenommen werden. Unter dem Menüpunkt „Projekt“, „Einstellungen“ und anschließend im Karteikartenreiter „Linker“ müssen im Feld „Objekt-/Bibliothek-Module“ die zugehörigen Bibliotheken angeben werden, die verwendet werden sollen (siehe Abbildung A2). Diese Bibliotheken sind: *cvcam.lib, cxcore.lib, cv.lib* und *highgui.lib*. Damit sind die wichtigsten Parameter eingestellt und der Implementierung von *OpenCV*-Modulen sollte nun nichts mehr im Wege stehen.

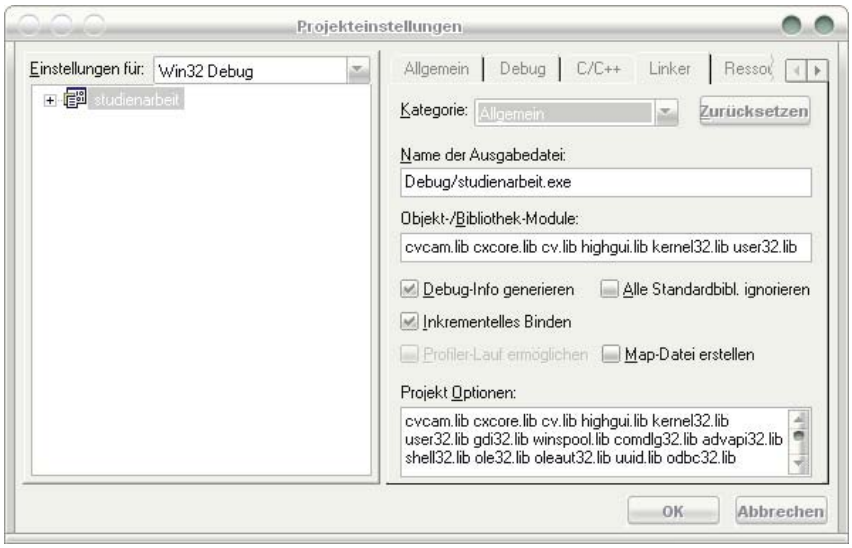

Abb. A2: Linkereinstellung eines Projektes in Visual C++ 6.0

Hat man seine Implementierungen abgeschlossen und möchte den Quellcode testen, wird man feststellen, dass die Programmausführung abbricht. Dies hat die Ursache, dass die benötigten Bibliotheken nicht mit in dem Ordner des erstellten ausführbaren Programms sind. Es gibt zwei Möglichkeiten dieses zu beheben. Entweder man kopiert die benötigten Bibliotheken per Hand in das Debug-Verzeichnis oder man ändert die Systemvariable „Path“ in Windows und erweitert diese um den *OpenCV*-Pfad. Um letzteres durchzuführen geht man folgendermaßen vor: Man öffnet in der Systemsteuerung die Option „System“, „Erweitert“ und wählt dort die Option „Umgebungsvariablen“ (siehe Abbildung A3).

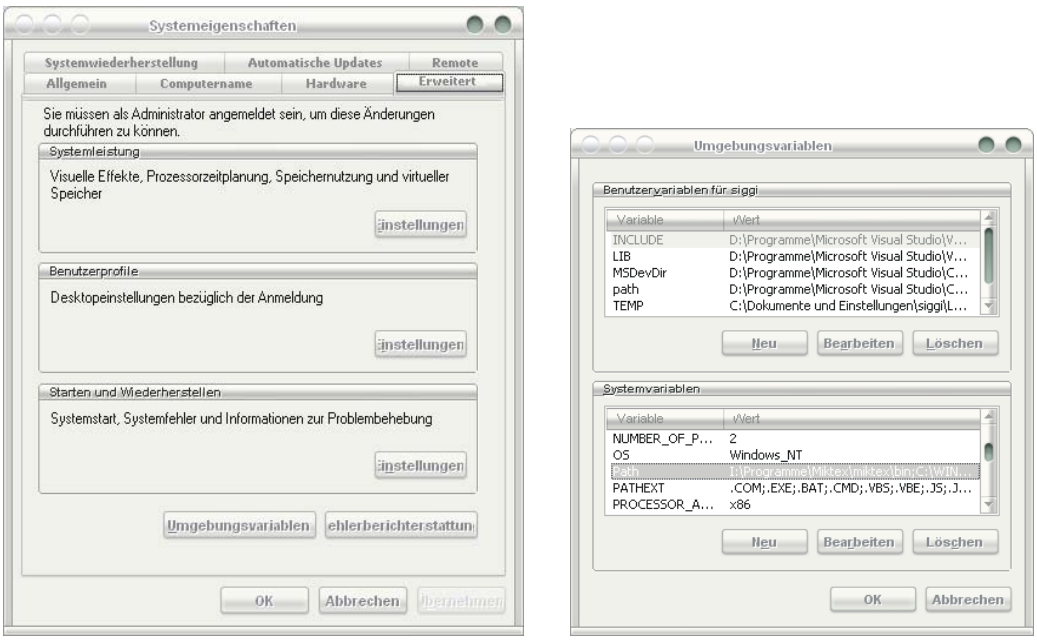

Abb. A3: Anpassen der Systemvariable Pfad in Windows XP (1)

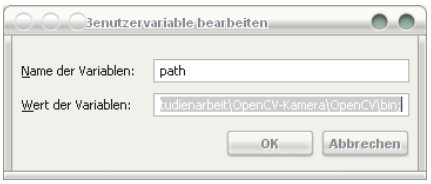

Abb. A4: Anpassen der Systemvariable Pfad in Windows XP (2)

Im Fenster „Umgebungsvariablen“ (siehe Abbildung A4) wählt man nun unter „Systemvariablen“ den Eintrag „Path“ aus und wählt anschließend die Schaltfläche „Bearbeiten“. Nun bekommt man ein nächstes Fenster (Abbildung A4), hier muss im Feld „Wert der Variablen“ der Ordner „OpenCV\bin“ angegeben und mit „OK“ bestätigt werden. Nun kann man die erstellte EXE-Datei ausführen, ohne dass man von Fehlermeldungen gestört wird. Sollte doch noch eine Fehlermeldung über eine fehlende Bibliothek erscheinen, muss man den Pfad, wo sich diese Datei befindet ebenfalls noch im Systempfad angeben.

B Berechnung der Punktkonzentration

Die Formel für die Berechnung der Punktkonzentration ist wie folgt:

$$M_{ix} = \frac{\left(\sum_{j=1}^{n} X_{i,j}\right)}{\left(\sum_{j=1}^{n} K_{i,j}\right)} \qquad M_{iy} = \frac{\left(\sum_{j=1}^{n} Y_{i,j}\right)}{\left(\sum_{j=1}^{n} K_{i,j}\right)} \qquad 1 \le i \le n$$

Hierbei gilt für die einzelnen Komponenten:

$$K_{i,j} = 1 \qquad X_{i,j} = \left(P_{jx}\right) \qquad Y_{i,j} = \left(P_{jy}\right)$$

für $\left|P_{ix} - P_{jx}\right| \le d$ und $\left|P_{iy} - P_{jy}\right| \le d$

$$K_{i,j} = X_{i,j} = Y_{i,j} = 0$$

für $\left|P_{ix} - P_{jx}\right| > d$ oder $\left|P_{iy} - P_{jy}\right| > d$

n ist die Anzahl der zu bearbeitenden Elemente
d ist die maximale Distanz zwischen dem Referenzpunkt und dem jeweils verglichenem Nachbarn
P_i ist der Referenzpunkt mit dem alle anderen Punkte verglichen werden
P_j ist jeweils der Punkt, der mit dem Referenzpunkt verglichen wird
M_i ist der Mittelpunkt der Konzentration an der Stelle i mit dem maximalen Abstand d

Zahlenbeispiel:

Es seien folgende fünf Punkte gegeben:
P_1 (10;12) P_2 (100;10) P_3 (90;15) P_4 (110;20) P_5 (120;5)

Der Wertebereich für die Punkte liegt bei $1 \le x \le 320$ und $1 \le y \le 240$, da die Kamera nur Bilder in einem Format von 320 x 240 wiedergeben kann.

Bei d = 20 gilt dann:

$$M_{1x} = \frac{(10+0+0+0+0)}{1+0+0+0+0} = \frac{10}{1} = 10 \qquad M_{1y} = \frac{(12+0+0+0+0)}{1+0+0+0+0} = \frac{12}{1} = 12$$

$$M_{2x} = \frac{(0+100+90+110+120)}{0+1+1+1+1} = \frac{420}{4} = 105 \qquad \text{etc.}$$

P_1 hat keine Nachbarn in seiner Nähe, die eine Distanz kleiner gleich 20 aufweisen. Die Konzentration bezüglich dieses Punktes ist dementsprechend nur der Punkt selbst.
P_2 hat vier Nachbarn, und zwar P_3, P_4, P_5 sowie sich selbst in einer geringeren Distanz als d. Deshalb wird über diese das arithmetische Mittel gezogen.

Für die Konzentrationen M_1 bis M_5 ergeben sich so folgende Punkte:

M_1 (10; 12)	mit einem Nachbarn (P1)
M_2 (105; 12,5)	mit vier Nachbarn (P_2, P_3, P_4, P_5)
M_3 (100; 15)	mit drei Nachbarn (P_2, P_3, P_4)
M_4 (105; 12,5)	mit vier Nachbarn (P_2, P_3, P_4, P_5)
M_5 (110; 11,67)	mit drei Nachbarn (P_2, P_4, P_5)

Hieraus wird anschließend derjenige Punkt mit den meisten Nachbarn als Punktkonzentration für das ganze Bild genommen und die anderen Konzentrationen werden verworfen. Sollte es mehrere Konzentrationen mit der gleichen Anzahl von Nachbarn geben, wird die erste Punktkonzentration genommen, da sie mit den anderen identisch ist, wie im Beispiel erkennbar

C Berechnung der Bewegungsgeraden

Die beiden Punkte, zwischen denen die Bewegung interpoliert werden soll, sind bekannt. Zum einen sind dies die aktuelle Position des Mauszeigers (P_M) und zum anderen die Position des berechneten Zielpunktes (P_Z) auf dem Bildschirm.

Hierüber kann eine lineare Gleichung der Form:

$$y = m \cdot x + n$$

berechnet werden.

Für den Anstieg m gilt folgender Zusammenhang der beiden Punkte:

$$m = \frac{P_{Zy} - P_{My}}{P_{Zx} - P_{Mx}}$$

Der Wert für n ergibt sich dann durch das Einsetzen eines Punktes (P_M oder P_Z) in die Formel:

$$n = P_{My} - m \cdot P_{Mx}$$

Zahlenbeispiel:

Es seien die Punkte P_M (100; 100) und P_Z (200; 150) gegeben.

$$m = \frac{150 - 100}{200 - 100} = \frac{50}{100} = 0{,}5$$

Wird nun der Punkt P_M eingesetzt, ergibt sich für n folgender Wert:

$$n = 100 - 0{,}5 \cdot 100 = 50$$

Es entsteht folgende lineare Gleichung, die zur Interpolation aller Punkte zwischen den gegebenen Start- und Endpunkt genutzt werden kann:

$$y = 0{,}5 \cdot x + 50$$

ENTWURF UND REALISIERUNG EINER KONTAKTLOSEN MAUS ALS EINGABEGERÄT

Autor:	Christian Hasselbach	Christian Siegl
Matrikelnummer:	141056	148858
Kurs:	TIT02BNS	TIT02BNS

Ziel dieser Studienarbeit ist es, eine aktuelle Übersicht über Forschungen und Entwicklungen auf dem Gebiet der kontaktlosen Maus vorzustellen und eines dieser Konzepte selbstständig zu realisieren. Zunächst werden hierfür die unterschiedlichen Forschungen, wie Auswertung von Sensoren, Kameradaten und Gehirnströme aufgezeigt und erläutert. Diese werden anschließend bewertet und ein System herausgefiltert, welches interessante und realisierbare Ansätze einer Implementierung bietet. Innerhalb dieser Studienarbeit wird auf die Auswertung von Kameradaten zurückgegriffen, die mit einer möglichst preiswerten Kamera umgesetzt wird. Hier ergeben sich folgende Teilaspekte für eine eigenständige Realisierung:

- Empfangen des Kamerabildes,
- Auswertung der Kameradaten und finden eines geeigneten Punktes,
- Übertragen dieses Punktes auf den Bildschirm und
- Tests und Testergebnisse.

Besonderer Wert wird bei der Ausarbeitung auf die verschiedenen Möglichkeiten der Auswertung von Kameradaten gelegt und anschließend der *Optical-Flow*-Algorithmus angewandt. Bei der Übertragung des Zielpunktes auf den Bildschirm werden weiterhin mehrere Positionierungsverfahren aufgezeigt, beispielsweise relative und absolute Positionierung, und ihre Implementierung dargestellt. Die abschließenden Tests und Testergebnisse zeigen die aufgetretenen Probleme auf und die entsprechend gefundenen Lösungen sowie weitere Punkte, die für zukünftige Projekte dieser Art relevant sind.

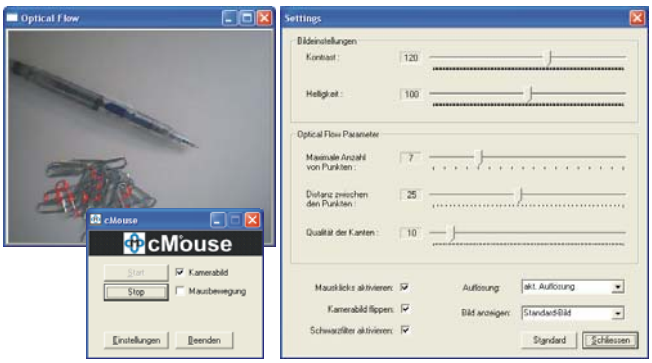

Abb. 1: Oberfläche des implementierten Systems

DESIGN AND IMPLEMENTATION OF A CONTACTLESS MOUSE AS INPUT DEVICE

Author:	Christian Hasselbach	Christian Siegl
Matriculation number:	141056	148858
Course:	TIT02BNS	TIT02BNS

This student research project aims to present an overview of developments in the field of contactless mice and to realize one of these concepts single-handedly. To begin with, different aspects of current research are presented such as interpretation of sensor and camera data as well as brain waves. Afterwards, these are evaluated and one of the methods is selected for implementation. The selection is based on interest and feasibility. Within this student research project the evaluation of camera data will be used. The implementation is divided into the following components:

- Capturing image data,
- Evaluating the image data and finding a suitable tee,
- Mapping the tee to the screen and
- Tests and test results

Particular interest during the project is put on the various methods of interpreting image data. Specifically, the optical-flow algorithm will be used later on. To map the tee to the screen, the advantages and disadvantages of relative and absolute positioning are compared to each other on the basis of their implementation. The final tests and there results will show problems encountered during the project. This paper will then provide solutions to these problems as well as further aspects with relevance for future projects.

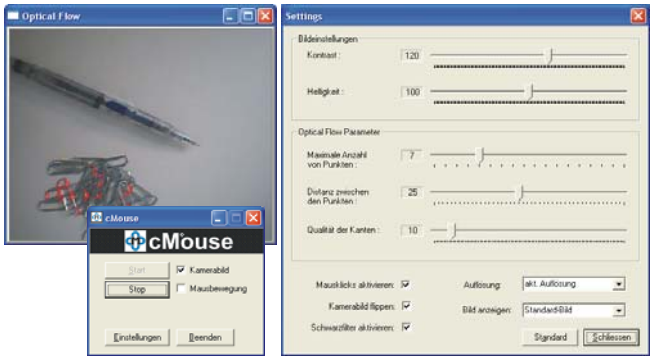

Figure 1: Interface of the realised system