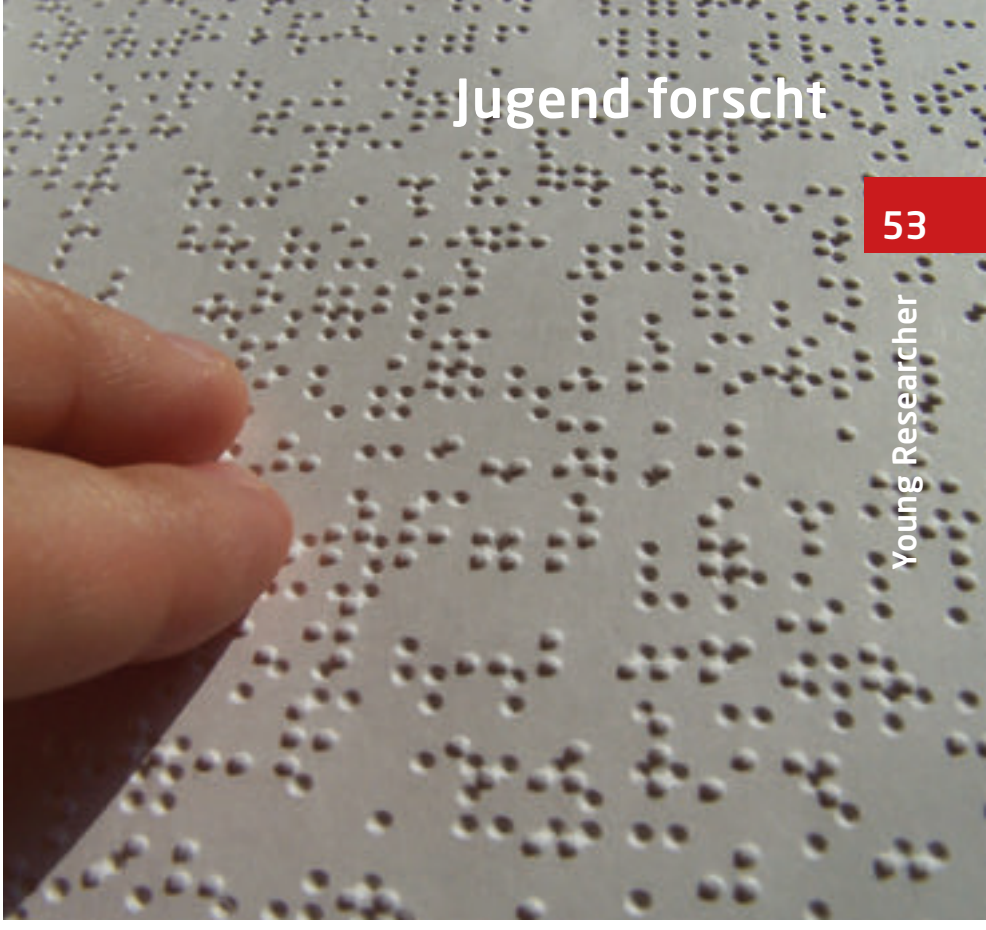
Andreas Sonntag * 1990
Saarlouis

Schule:
Saarlouiser Gymnasium am Stadtgarten,
Saarlouis

Eingang der Arbeit:
März 2009

Zur Veröffentlichung angenommen:
Mai 2009



Bilder begreifen

Ein neues Konzept für eine Tastatur, die Bilder für Blinde begreifbar macht

Die hier entwickelte Blindentastatur soll blinden und sehbehinderten Menschen helfen, Bilder zu begreifen. Dafür wurde eine Software entwickelt, die Bilder analysiert und anschließend die wichtigsten Strukturen auf einer vibrierenden Platte begreifbar macht. Einfache Strukturen wie Kreise, Quadrate und Sterne können Blinde damit bereits eindeutig erkennen.

1 Themenwahl

Hilfsprogramme von Windows Vista und anderen Betriebssystemen erleichtern sehbehinderten Menschen das Arbeiten mit dem Rechner, indem sie mit einer Bildschirmlupe Ausschnitte vergrößern oder Texte vorlesen. Aus Interesse an solchen Hilfsprogrammen recherchierte ich im Internet, ob es weitere Möglichkeiten gibt, sehbehinderten Menschen die Arbeit am Computer zu erleichtern.
Hierbei fand ich eine Tastatur, die Braillezeichen greifbar macht, indem kleine Stäbchen aus ihr heraustreten. Diese an Arbeitsplätzen und Schulen häufig eingesetzten Geräte sind sehr teuer und bedürfen guter Kenntnisse über die Brailleschrift, die man schnell und genau lesen können muss.

Weiterhin gibt es die Möglichkeit, Braillezeichen als fühlbares Relief zu drucken, um sie auf Papier lesen zu können. Im Bereich der Mathematik und Physik helfen Hilfsprogramme wie „LaTex“, Formeln oder komplexe Zeichen zu drucken.
Um graphische, mathematische Inhalte zu vermitteln, gibt es eine Maus mit integriertem „rumble pack“, die vibrieren kann, um ein Feed-back an die Hand zu übermitteln. Dieses soll Informationen über Position und Art der Zeichen bzw. Graphen liefern. Dies ist sehr schwierig, da die Maus sich nicht linear bewegt, sondern je nach Haltung verschieden stark beschleunigt.

Bei meinen Recherchen fiel mir also auf, dass es keine Software gibt, um ganze Bilder oder gar Videos für Sehbehinderte oder Blinde begreifbar zu machen.
Anfänglich überlegte ich mir aus Neugierde, wie man so etwas Komplexes wie ein Bild in eine begreifbare Darstellung für Blinde umsetzen könnte und arbeitete an einer Software, die zweidimensionale Bilder in dreidimensionale Darstellungen übertragen konnte. Nach ersten Ansätzen beschäftigte ich mich dann mit der tatsächlichen Umsetzung dieser Bildinformationen durch einem Apparat. Nach einem Besuch der Blindenschule in Lebach und Gesprächen mit blinden Schülern habe ich schließlich eine Apparatur gebaut, die die Umsetzung der Bildinformationen für Blinde tatsächlich in einer einfachen und kostengünstigen Art und Weise ermöglicht.

2. Problemstellung

Im Folgenden werde ich ausgehend von der Erlebniswelt eines Blinden, die Probleme in Soft- und Hardwareentwicklung aufzeigen, die für das Erreichen des Ziels gelöst werden mussten.

2.1 Erlebniswelt eines Blinden

Blinde und Sehbehinderte nehmen ihre Umwelt anders wahr. Sie verlagern ihre Wahrnehmungsprioritäten auf alle anderen Sinne, vor allem auf den Tastsinn.

Sie sind daher nicht daran gewöhnt, geometrische Einheitsobjekte wie Kreise, Vierecke oder Dreiecke als Grundelemente der Umgebung anzusehen. Sie wenden deshalb eine völlig andere Strategie beim Erfassen ihrer Umwelt an: Sie versuchen, einzelne Formen zu ertasten und vergleichen diese mit denen in ihrer Erfahrung und kombinieren diesen Eindruck mit ihrem Gehöhr und dem allgemeinen Wissen um diesen Bereich.

Wenn man Blinden nun Bilder begreifbar machen will, muss man also ein System schaffen mit der Möglichkeit, eindeutige Formen zu bilden, die unabhängig vom Gesamtinhalt identifizierbar sind.

2.2 Bilderkennung- Anforderungen an die Software

Das Problem ist die Ausgabe der wichtigsten Strukturen eines Bildes auf eine Blindentastatur. Um dieses Problem zu lösen, muss das Bild zuerst analysiert werden.

Der Mensch ist durch seine Erfahrungen, die er im Laufe seines Lebens gemacht hat, in der Lage, in einem Bild oder im Raum Strukturen zu erkennen und diese dahingehend zu deuten, ob diese zweidimensional oder dreidimensional sind. Er ist somit in der Lage, zwischen Vordergrund und Hintergrund eindeutig zu unterscheiden und somit das Objekt, als solches zu identifizieren, zu klassifizieren und zu definieren. Er kann also unterscheiden, ob er eine Flasche oder eine Person sieht; ob diese blonde Haare, weiße Schuhe oder blaue Kleidung trägt. Diese Differenzierungen kann er allerdings nur beschreiben, weil er auf viele Jahre des Erfahrungssammelns zurückblicken kann. Er hat im Laufe seines Lebens all diese Objekte schon einmal gesehen bzw. er hat erfahren, was diese darstellen.

Ein moderner Computer ist nicht in der Lage komplexe Unterscheidungen im visuellen Bereich zu machen. Der Computer verfügt über keinerlei Erfahrungen wie der Mensch. Er ist somit nicht in der Lage, die Transferaufgabe zu meistern, aufgrund der bekannten Definitionen der Objekte, diese in einem neuen Kontext zu sehen. Dies kann nur der Mensch. Es muss also ein einfacheres Verfahren gefunden werden, das die Komplexität des Bildes einschränkt und dem Computer eine Interpretation ermöglicht.

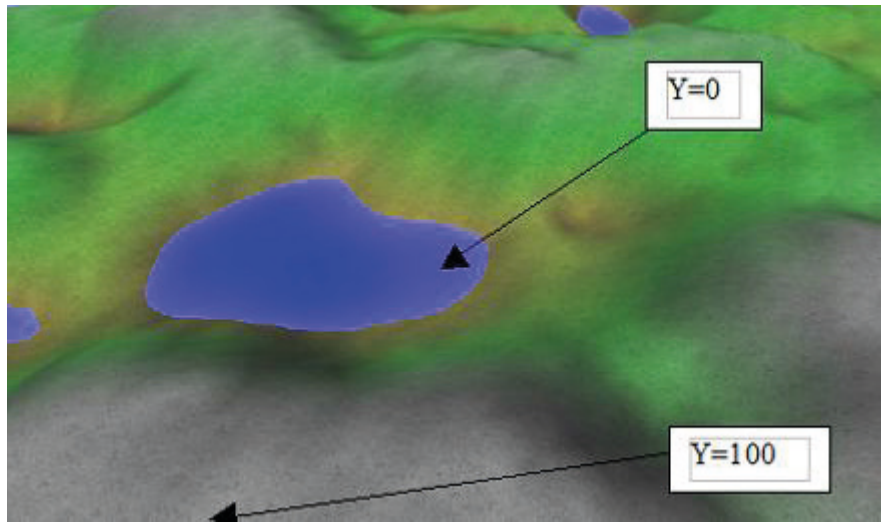


Abb. 1: Drahtgitterlandschaft mit verschiedenen Höhen, die von Blau bis Grau eingefärbt sind

2.3 Bilddarstellung- Anforderungen an die Hardware

Es gibt bereits technische Hilfsmittel, die Blinden ihre Umgebung erfahrbar machen können. Doch keines dieser Hilfsmittel kann ganze Bilder so darstellen, dass sie Blinde interpretieren können.
Bedenkt man nun das Prinzip der Komplexitätsverringerung, so liegt die Idee nahe, ein Gerät zu konstruieren, welches aus verschiedenen Feldern besteht, die unabhängig von einander ihre Eigenschaften fühlbar verändern können.
Allgemein benötigt man genauso viele Felder wie Pixel im Bild. Durch die Komplexitätsverringerung jedoch bräuchte man nur einen Bruchteil an Feldern, um die Informationen des Bildes dennoch beizubehalten.

Um dieses Prinzip nun zu realisieren, braucht man also ein Gerät mit verschiedenen Feldern, die sich fühlbar verändern lassen.

3 Softwarelösung zur Umsetzung von Bildinformationen für Blinde

3.1 Die Idee

Zunächst erscheint es logisch, dem Computer „beizubringen“, wie ein jedes Objekt in einem Bild aussehen kann. Da es aber viel zu viel Zeit in Anspruch nähme, dem Computer alle alltagsüblichen Objekte zu erklären, habe ich mich bemüht einen anderen Weg zu finden, um die wichtigsten Strukturen herauszufiltern: Der Computer soll über das Bild gehen und eine Häufigkeitsverteilung der Farben anlegen und diese dann auswerten. Aufgrund dieser Bewertung soll er dann alle "unwichtigen" Farbinformationen löschen, um sicherzustellen, dass auf dem Ausgabebild nur noch "wichtige" Flächen vorhanden sind. Der Anwender soll nicht mit für ihn unwichtigen Informationen konfrontiert werden, die ihn nur unnötig belasten würden.
Da also durch das Filtern auf die Komplexität des Bildes verzichtet werden muss und nur noch die wichtigsten Strukturen begreifbar gemacht werden sollen, ist es zunächst das Ziel, einfache Bilder zu nehmen, auf denen möglichst einfache Objekte sind.
Um zu prüfen, ob diese Vorgehensweise möglich ist, erstellte ich ein Modell in Form einer virtuellen Matrix, die die Bildinformationen vereinfacht darstellt. Anhand dieser soll dann eine reale Lösung gefunden werden.

3.2 Grundlagen zur Bilddarstellung:

Ein Bild wird wie folgt definiert:
Der Befehl „load image(datei,image1)“ liest die Bilddatei als Speicherblock ein, für den gilt:

```
image1.breite =
MEMBLOCK DWORD(image1, 0)
image1.hohe =
MEMBLOCK DWORD(image1, 4)
image1.tiefe =
MEMBLOCK DWORD(image1, 8)
```

Alle weiteren Daten sind Farbinformationen (image1.farbe = MEMBLOCK DWORD(image1, 8 + position)), 0,4,8 sind Bytepositionen.

Definition der als 3D-Objekt dargestellten Matrix:

Die Mengen: V = {1,2,..maxV} (Vertex- = Knotenmenge) und E = {(1,maxV) , .. (maxV,1)} (Edge- = Kantenmenge) [7] beschreiben die Grundlage jeder graphischen Darstellung.
Ein jeder Graph kann mit diesen Mengen beschrieben werden, da jeder Punkt des Graphen definiert ist. Es ist daher notwendig, Vertizes (Knotenpunkte) zu deklarieren, deren Verknüpfung das Gesamtobjekt beschreibt. Ein 3D-Objekt ist also wie folgt definiert:
Sei O eine Relation für die gilt:
pos : int -> (real,real,real,real,real,real,dword,int,int);
O(pos) := (X(pos),Y(pos),Z(pos), NX(pos),NY(pos),NZ(pos),rgb(r,g,b),u,v)
Wobei pos die Position im Speicherblock(Datenfeld aller O(pos)), X, Y, Z die Koordinaten und NX,NY,NZ die Normalenkoordinaten der Vertizes sind. Rgb gibt den Farbwert an der Stelle an und u, v dienen zur Mitberechnung von Texturen. (X(pos),Y(pos),Z(pos)) ist Vertexinformation (Scheitelpunkt) und pos somit Meshindex (Drahtgitterposition) an der Koordinate (X,Y,Z). Die Menge aller O(pos) bildet, wie später ausgeführt wird, einen Speicherblock, der das dreidimensionale Objekt vollständig beschreibt und über die DirectX Schnittstelle angezeigt werden kann.
Eine Beispielmatrix folgt in Abbildung 1. Diese „Drahtgitterlandschaft“ dient als Grundlage für die Speicherung und Visualisierung der Bildkonturen als Y – Höhen. Diese Bildkonturen sollen durch ein noch zu findendes Prinzip erkannt werden und dann in der 3D-Matrix gespeichert werden. Somit kann sie die Bildinhalte veranschaulichen.

3.3 Erste Versuche zur Prinziperkennung

Die Abb. 2. zeigt beliebig gewählte geometrische Formen mit einem Schwarzweiß Kontrast in fünf Stufen: von 100% Weiß bis 0% Weiß. Der Computer soll nun in Relation zur maximalen Höhe, dies entspricht der Farbe 100% Weiß, ein dreidimensionales Abbild bzw. Relief der verschiedenfarbigen Flächen schaffen.
Die verwendete Programmiersprache ist „DarkBasic Professional“, die einen C-Compiler verwendet und einige der komplexen C-Befehle in einfachere Prozeduren zusammenfasst. Außerdem verfügt er über einige DirectX-Bibliotheken für Ein- und Ausgaben.

Das Datenfeld " Matrix" ist zweidimensional, wobei xsegments und ysegments jeweils für die Breite und Höhe des Bildes stehen, das vorher mit dem Befehl "load image" geladen und mit "paste image" angezeigt wurde. Diese beiden Befehle stammen aus der Bibliothek des Compilers "Dark basic professional", der für das gesamte Projekt verwendet wird. Ebenso aus dessen Bibliothek, welche auf die Microsoft Direct X 9.0 c zugreift, ist die Funktion "point(x, y)", die die Farbe des Punktes (x, y) ausgibt. Die Funktion "rgb(g,r,b)" gibt die Farbe aus, die sich aus den Grün- Rot- und Blauwert ergibt. In diesem Fall 100% Weiß.

Das Datenfeld Matrix enthält also nun die relative Farbe des Punktes (x,y). Dieser Wert ist der Ausgangswert für die Berechnung der dritten Dimension des Bildes, welche der Anwender später nachfühlen soll. Der Graph der Matrix ist in Abb. 3 dargestellt.
Jedoch stellt sich nun das Problem, das nicht in jedem Bild die Farbe Weiß als Maßstab bzw. maximale Höhe dienen kann. Aufgrund der Häufigkeitsverteilung in komplexen Bildern ergibt es sich, dass die Farbe Weiß kaum oder gar nicht vorkommt.
Es muss also ein allgemeiner Maßstab gefunden werden. Am nächsten liegend scheint da die Farbtiefe zu sein, da sie in jedem Bild anders ist und somit als Maßstab dienen kann.
Ich erweiterte also den Algorithmus und setzte einen Filter davor. Der Filter „glättet“ das Bild, sodass die Farbdifferenzen nicht so komplex sind.

Das Datenfeld matrix(x,y) speichert zu jedem Punkt (x,y) des Bildes den ermittelten relativen Farbwert. matrix(x,y) ist also die dritte Dimension des Bildes, welche ihm durch eine 3D-Matrix mit dem Befehl „set matrix height“ hinzugefügt wird.
Ich stelle also fest, dass das ursprüngliche Problem der Räumlichkeitsanalyse indirekt durch die Funktion: Matrix(x,y) = rgbg(point(x,y)) / image1.tiefe gelöst wird, wobei ihr dabei der Filteralgorithmus hilft, da dieser das Bild in Layer einteilt und alle Farbbereiche des Bildes gleichschaltet und überschreibt. Im Weiteren wird zur Vereinfachung des Bildes die hellste Farbfläche gesucht und auf 100 % Weiß gesetzt.

3.4 Das Konzept der Wichtigkeit

Ziel dieses Projektes ist es, dass der Anwender die Möglichkeit bekommt, zu begreifen bzw. zu erfahren, welche Objekte sich auf einem Bild befinden. Im Vordergrund stehen dabei nicht die Details, sondern die groben Strukturen. Der Anwender soll nicht mit der Komplexität des Bildes überflutet werden, sondern lediglich die wichtigsten Informationen bekommen. Es ist also notwendig, das eingegebene Bild zu vereinfachen. Wir beschränken uns dabei auf die wichtigsten Flächen.

Dies geschieht durch obigen Algorithmus, der für eine vertikale Wichtigkeitsaufteilung sorgt, indem er Flächen mit lokalen Farbabweichungen gleichschaltet und die Farben im Bild von der nun dunkelsten zur hellsten neu orientiert. Dies schafft Graustufen bzw. einheitliche

Abb. 2: Farbfelder mit Graustufen als exemplarisches Beispiel eines einfachen Bildes

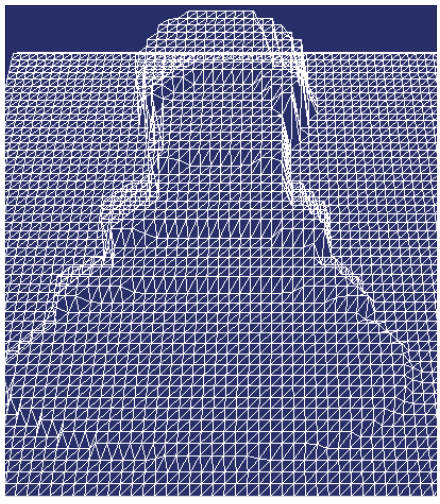


Abb. 3: Dreidimensionaler Graph der in Abb. 2 dargestellten Graustufen

Farbebenen, deren Helligkeitsdifferenz zu benachbarten Ebenen einen Kontrast in der dritten Dimension erzeugt: Der Farbe. Dieser Kontrast wird durch die nun folgende Funktion der Wichtigkeitsfeststellung ermittelt:
„ini1“ definiert im Folgenden die Wichtigkeit an der Mausposition (von 0 bis 100, Einheit in Prozent). Durch das Einbeziehen der Mausposition kann das Bild in Echtzeit gelesen werden und es entsteht kein Laufzeitverlust, da die Feststellung der Mausposition sowie der Farbwert an dessen Stelle von DirectX übernommen wird.
Es sei „mousex()“ die X-Koordinate, „mousey()“ die Y-Koordinate des Mauszeigers, „heightSize#“ die Bildtiefe des auf dem Bildschirm angezeigten Bildes, und point(x,y) eine Funktion, die die Farbe am Punkt (x|y) als rgb Triple darstellt. int(x#) wandelt x# : real in x : integer um.
ini1 = int(point(mousex(),mousey()))
ini1 wird zu dem Farbwert an der Mausposition ausgewertet. Dieser Wert ist jedoch unverhältnismäßig hoch und nicht zwischen 0 und 1. Außerdem berücksichtigt er nicht die Farbverteilung und Kontraste im ganzen Bild.

Hierzu muss das Bild - wie bereits gesagt – vorbereitet und gefiltert werden mit einer Funktion, die bereits durch obigen Algorithmus implementiert ist. Dieser teilt das Bild in Layer auf und orientiert es der hellsten Farbe nach neu.
Diese Vorbereitung des Bildes spart Rechenzeit, da nicht bei jeder Mausbewegung das Verhältnis zum Gesamtbild neu berechnet wird. Es muss lediglich die Einteilung von 0 bis 100 vorgenommen werden. Dies geschieht durch die Funktion: rgbg(rgbcolor) = (rgbcolour&&0x0000FF00)>>8, die aus dem in Graustufen vorliegendem, gefilterten Bild den Grünwert liest, der gleich der Helligkeit an dieser Stelle ist, da alle Kanäle bei dem gefilterten Bild gleichgeschaltet wurden.
rgbg(point(x,y)) gibt also einen Wert von 0 bis 255 aus, da für jede Farbe color gilt: color = rgb(r:byte,g:byte,b:byte), also auch für point(x,y).
Für die Wichtigkeit I(x,y) gilt:
I(x,y) = int (rgbg(point(x,y)) / heightSize# / 10) * 12.25
und für die Wichtigkeit I(mousex(), mousey()) = ini1 gilt:
ini1 = int (rgbg(point(x,y)) / heightSize# / 10) * 12.25

Da diesem Wert „ini1“ eine besondere Bedeutung zukommt, gebe ich ihm die Einheit „AS“ wie folgt:
[importance] = 1 AS

Dieser Wichtigkeitswert gibt an, als Ersatz zur Räumlichkeit, wie wichtig ein Punkt oder eine Fläche ist. Die durch die Funktion neugewonnene Y Koordinate = ini1 wird als Höhenachse der 3D-Matrix projiziert:
Id=1
Setmatrixheight(id,x,y,matrix(x,y))
Der Befehl "setmatrixheight(id,x,y,height)" setzt die Höhenachse der 3D-Matrix. Das bedeutet, dass der X-und Z-Koordinate Matrix(x,z) zugeordnet wird. Die neue Bezeichnung x, z statt x, y ist erforderlich, da im 3D-Raum x und z die Fläche beschreiben, also Breite und Tiefe und y die Höhe. Der Graph des in Abb. 2 gezeigten Bildes sieht also nach Anwendung des Algorithmus wie in Abb. 3 aus. Dieses zeigt das plastische und dreidimensionale Objekt, das verhältnismäßig genauso breit und hoch ist wie das Originalbild. Die oben gezeigte Matrix soll eine Blindentastatur simulieren. Sie schafft die Grundlage, die Höheninformationen einfach abrufbar für spätere Apparaturen zu verwalten. Der Anwender soll ein Gerät vorgesetzt bekommen, welches ihm die Möglichkeit bietet, jene durch die obige Matrixfunktion berechneten Werte zu begreifen, wobei mit Begreifen hier ein tatsächlicher physischer Kontakt gemeint ist.

3.5 Die Implementierung der Bilderkennung

Nun fassen wir all die obigen Prozeduren in ein Programm zusammen, welches dann nach Eingabe eines Bildes, den Graph der virtuellen Matrix ausgibt und die Wichtigkeit jedes einzelnen Punktes im Bild in Echtzeit berechnet und angibt. In Abb. 4 ist beispielhaft links ein reales Bild dargestellt und rechts der Graph der virtuellen Matrix.
Der Graph der virtuellen Matrix zeigt sehr schön die Strukturen des Bildes. Diese Kontraste machen die Konturen der Tasse erkennbar, indem sie sie räumlich von dem „unwichtigen“ Tisch trennen.

4 Hardware zur Umsetzung der Bildinformationen für Blinde

4.1 Mögliche Ideen zur Lösung

Stellen wir uns nun vor, wir könnten die virtuelle Matrix als ein tatsächlich existierendes Objekt fühlen. Dabei könnten wir ohne Zweifel nach einiger Zeit des Ausprobierens am Beispiel aus Abb. 4 eine tassenähnliche Plastik erkennen. Wir können also ohne das Bild gesehen zu haben mit einer relativ hohen Wahrscheinlichkeit sagen, dass es sich um eine Tasse handelt.

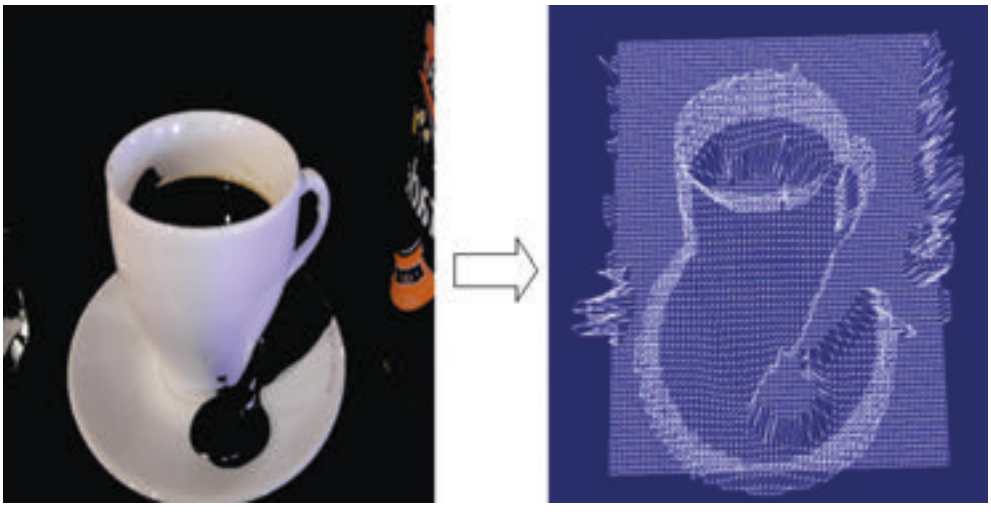


Abb. 4: Links: Bild einer Tasse; Rechts: Dreidimensionaler Graph der Tasse nach Berechnung der „Wichtigkeit“

Es wurde gesagt, dass die 3D-Matrix die Blindentastatur simulieren soll. Eine Möglichkeit wäre also, ein Gerät zu entwickeln, das entsprechend viele Felder hat, die individuell verschieden hoch angehoben werden können. Die technische Umsetzung sieht schematisch wie in Abb. 5. aus.
Die Umsetzung, so ein Gerät

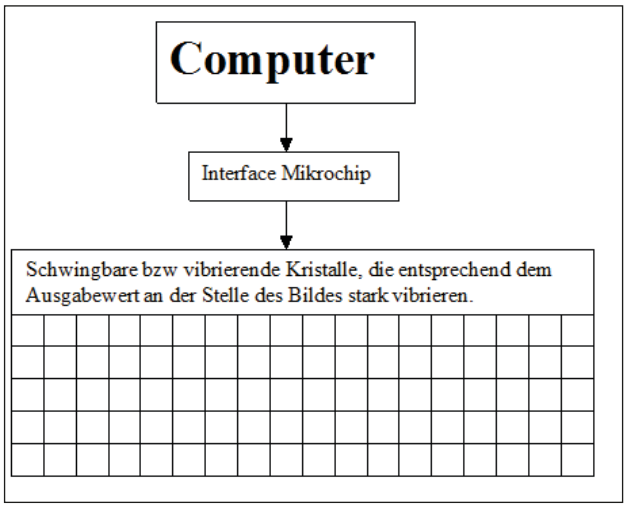


Abb. 5: Schematischer Plan eines idealen Gerätes zur Ausgabe der dreidimensionalen Informationen

tatsächlich zu bauen, würde sich jedoch als sehr teuer und aufwendig erweisen. da weder ein geeigneter Steuerchip noch schwingbare Kristalle im Handel zu erwerben sind. Auch eine elektrotechnische Umsetzung mit beweglichen Stäben an Stelle der Kristalle würde sich als sehr schwierig erweisen.

Es stellt sich also nun die Frage, wie die Wichtigkeit spürbar an die Hand des Anwenders geleitet werden kann. Hierzu wird ein Motor oder Schwingungserzeuger benötigt, der 100 verschiedene Impulsstärken haben muss. Prädestiniert als Schwingungserzeuger ist ein Lautsprecher, dessen Membranaufhängung entsprechend schnell der Eingabefrequenz schwingt oder die Kombination eines „Force-feedback-vibration“ Controllers und einer normalen Computermaus (ähnlich der in Kapitel 1 angeführten).
Hierbei wird die Ausgabe des Höhenwertes an der Stelle des Bildes, wo sich die Maus befindet, als Vibrationsimpuls an den Controller gegeben. Diese Methode erfordert Training, ist aber weitaus günstiger und einfacher als einen Lautsprecher zu verwenden, der als Eingabe transponierbare Töne bräuchte, die kompliziert zu arrangieren wären.

4.2 Eine erste Tastatur

Das erste Modell einer Blindentastatur wurde wie folgt aufgebaut: Eine rechteckige, 0,8 mm dicke Stahlplatte wurde mit ihren vier Ecken weich und mit kleiner Auflagefläche gelagert, sodass auf die Platte auftreffende Vibrationsimpulse mit möglichst wenig Energieverlusten übertragen werden können. Als Impulsgeber wurde ein Vibrationsmotor eines Force-Feedback-Controllers fest montiert (sieh Abb. 6). Der Motor, auf dessen Ankerwelle ein Extender montiert ist, bildet radiale Vibrationsimpulse. Er wurde daher mit seiner Achse parallel zur Stahlplatte montiert.

Das eingesetzte Computerprogramm gibt den Befehl, den Force Feedback Controller mit der berechneten Stärke ini1 vibrieren zu lassen, wobei hier durch die Konstanten „vibration1“ und „vibrationsangle1“ eine optimale Koordination der beiden Motoren ermöglicht wird, was für eine gleichmäßige Vibrationsstärke auf der ganzen Platte sorgt. Der Anwender kann also nahezu in Echtzeit mit der Maus über das Bild gehen und den Impuls bekommen.

Bei der Anwendung des obigen Bilderkennungsalgorithmus stellte sich heraus, dass mit zunehmender Impulsfrequenz die auf der Stahlplatte erzeugten Vibrationsstufen im höheren Frequenzbereich weniger spürbar erkannt werden konnten. Trotz dieser Tatsache und einigen Selbsttests war ich zuversichtlich, dass ein Anwender mit hoher Tastsensibilität in der Lage wäre, mindestens 70 von 100 Intensitätsstufen zu spüren.

Daher bat ich einen sehbehinderten Schüler der Louis-Braille Schule in Lebach, das Gerät zu testen. Nach einer Einführung in die Bedienung der Tastatur wurden dem Schüler einige Bilder vorgeführt. Er war jedoch - bis auf eine Fläche - nicht in der Lage, die einfachen geometrischen Objekte in Position und Form zu erkennen, und dies obwohl ihm die geometrischen Objekte bekannt waren.

Ich folgerte aus diesem Versuch, dass Sehbehinderte nicht die gleiche räumliche Vorstellung haben wie normal Sehende. Der besagte Schüler war nämlich nicht in der Lage, die Metallplatte als Repräsentant des digitalen Bildes anzusehen. Dies wurde deutlich, als er meine Fragen bezüglich der genauen Position einzelner Bestandteile widersprüchlich beantwortete.

Der Schüler riet mir, Änderungen bezüglich der Maussteuerung zu machen. Diese bewegt den Mauszeiger mit einer beschleunigten Bewegung und kann somit nicht im Verhältnis zur Platte gesteuert werden. Dem Anwender war es also nicht möglich, den Punkt und den Impuls einander zuzuordnen. Es musste also eine andere Eingabemöglichkeit gefunden werden.

4.3 Die Vierpunkt-3D-Tastatur

Um den Aufbau zu optimieren, ersetzte ich die Maus durch einen Touchscreen, der mit den Maßen der Metallplatte kalibriert wurde. Durch den Touchscreen wird nun die Maus gesteuert, was die Bewegung des Fingers und des Mauszeigers synchronisiert, da jede Stelle des Touchscreens genau einer Stelle des Bildschirms entspricht und da das Bild auf diesen gedehnt ist, kann man das gesamte Bild „abfühlen“. Jedem Punkt auf der Platte entspricht also ein Punkt im Bild.

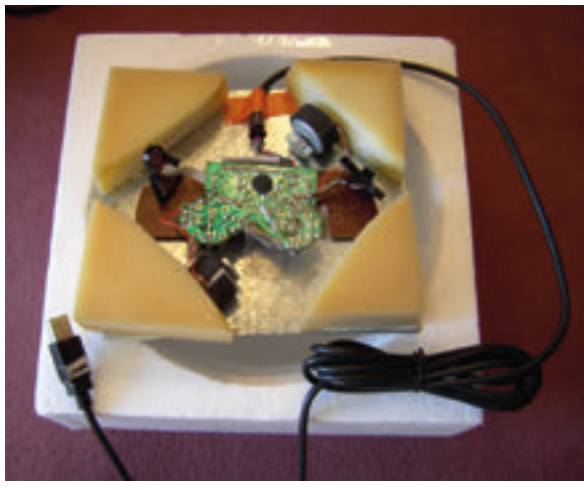


Abb. 6: Unter der Stahlplatte befindet sich die Elektronik und die zwei Vibrationsmotoren

Der Aufbau ist in Abb. 7 dargestellt: Anknüpfend an die erste Tastatur wurde wiederum eine rechteckige, 0,8 mm dicke Stahlplatte mit ihren vier Ecken weich und mit kleiner Auflagefläche gelagert. Als Impulsgeber wurden nun zur Leistungsverstärkung zwei Vibrationsmotoren eines Force-feedback-Controllers fest an der Unterseite montiert. Diese werden gegenüberliegend angebracht, so dass ihre radialen Impulse (auch durch die Software unterstützt, die die Einsätze der einzelnen Motoren koordiniert) zu einer konstruktiven Interferenz mit maximaler Intensität führen. Die dazu notwendigen Positionen wurden durch zahlreiche Versuche festgestellt.

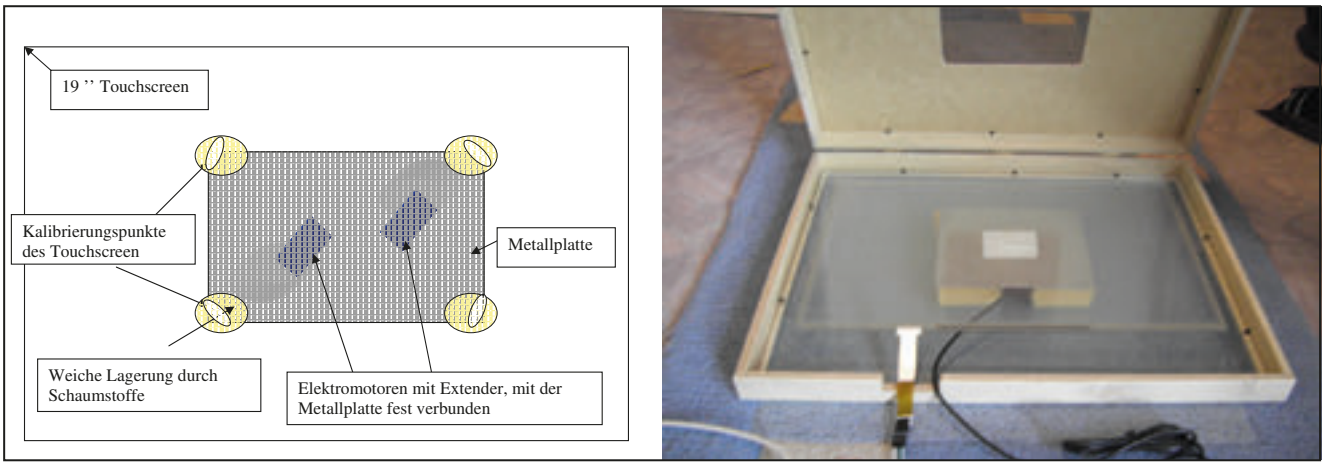


Abb. 7 Aktuelle Version der Vierpunkt-3D-Tastatur: a) schematisch b) Realisierung

Auf die Metallplatte wurde nun ein 19 Zoll resistiver Touchscreen geklebt und entsprechend den Eckpunkten der Platte nach kalibriert. Damit nur noch der Teil des Touchscreens betastbar ist, der auf der Metallplatte liegt, wurde ein Holrahmen mit Deckel über die ganze Konstruktion gelegt. Der Deckel wurde entsprechend ausgeschnitten, sodass die Metallplatte betastet werden kann.

Dieser neue Apparat wurde dem bereits erwähnten Schüler aus der Louis-Braille Schule nun vorgesetzt. Es wurden die gleichen Tests wie bei dem vorherigen Apparat durchgeführt.

Diesmal konnte sich der Schüler mit dem Touchscreen viel besser im Bild zurechtfinden und erkannte vier von fünf der einfachen geometrischen Objekte. Weiterhin erkannte er sogar Strukturen eines Koordinatensystems und konnte den Verlauf des Graphen nachfühlen. Gesichter und andere sehr komplexe Bilder sind für ihn weiterhin nicht nachvollziehbar.

Abb. 8 Die Buchstaben Ö und X in Brailleschrift bilden die Form einer Tasse

4.4 Bilder für Blinde drucken

Dieses Ergebnis ermutigte mich, ein noch besseres und leistungsfähigeres System zu entwickeln, dass besser nachfühlbare Plastiken automatisch kreiert und die Bedienung z.B. im Unterricht einfacher und Hardware-unabhängiger macht.

Hierzu verwendete ich den bereits zu Beginn erwähnten Braille-Drucker, der in das Papier hineinstanzt. Dieser ist allerdings nur für Braillezeichen ausgelegt und daher muss die Eingabe entsprechend aus Buchstaben bestehen, was es notwendig macht einen neuen Konvertierungsalgorithmus zu entwickeln.

Die Funktion soll ein Bild laden, das möglichst bereits gefiltert wurde und es nun noch einmal in Schwarz, Hellgrau und Weiß aufteilen. Diese Farben ermöglichen es später bei der Ausgabe auf dem Papier, einfache Koordinatenachsen von Graphen zu unterscheiden, was auch ein Ablesen der Koordinaten möglich macht.
Im nächsten Schritt wird nun eine Textdatei erstellt, die entsprechend zu den Pixeln im Bild, die Buchstaben Ö, A und X speichert.
Diese Textdatei wird nun mit der zum Drucker gehörigen Software in Brailleschrift konvertiert und ausgedruckt. Die genannten Buchstaben sind als Braillezeichen so verschieden, dass sie auf dem gestanzten Papier sehr gut differenzierbar sind. Somit können Konturen leicht erkannt werden und sogar Punkte auf dem Graphen genau identifiziert werden. Zur Veranschaulichung dient Abb. 8.

5 Zusammenfassung und kritische Analyse

Mein Lösungsansatz basiert auf der Vereinfachung des Eingabebilds auf das „Wichtigste“. Dieser Ansatz ist aber nur im Idealfall eine perfekte Lösung, da in der Wirklichkeit bei einigen Bildern mit starken Lichtquellen sowie Vorder- und Hintergrundvermischungen kaum Strukturen hervorgehoben werden können. Diese Tatsache, dass die Informationen des Bildes zu komplex sind, als dass ein Computer sie jemals derart vereinfachen könnte, brachte mich davon ab, an einer Optimierung des Bilderkennungsalgorithmus zu arbeiten.

Ich konzentrierte mich stattdessen auf einfache Objekte und schematische Bilder (z.B. für Mathematik-Lehrbücher), die aufgrund ihrer Eindeutigkeit in der Farbverteilung sehr viel einfacher von dem Bilderkennungsalgorithmus erkannt werden können.

Bei der technischen Umsetzung konnte ich aus zeitlichen Gründen leider keine Experimente mit einem Lautsprecher als Motorersatz durchführen. Theoretisch wäre dieses Prinzip jedoch besser, da ein Lautsprecher sehr viel schneller auf die Befehle vom Bilderkennungsalgorithmus reagieren würde.

Das heißt also, dass das Hauptziel nicht vollständig gelöst werden konnte, da es nach wie vor nicht möglich ist, jedes beliebige Bild für Blinde begreifbar auszugeben, sondern lediglich solche, die entweder sehr eindeutig in ihrer Farbverteilung sind oder extra für den Bilderkennungsalgorithmus ausgelegt sind.
Außerdem haben die Tests mit dem blin-

den Schüler aus Lebach gezeigt, dass die Bedienung der Apparatur große Konzentration erfordert. Angesichts der normalen Lesemethoden Sehbehinderter ist eine große Umgewöhnung nötig, da nichts wirklich greifbar ist, sondern lediglich punktuell erfasst wird. Dennoch hat sich erwiesen, dass der blinde Schüler die meisten der ihm auf der Apparatur „gezeigten" Bilder erkannt hat.

Als Einsatzmöglichkeiten der Blindentastatur sehen der blinde Schüler und ich den didaktischen Kontext: Lerninhalte aus Architektur, Mathematik, Biologie, Chemie oder Kunst sollten so an Blinde zu vermitteln sein.

Konkret könnte man also z.B. ein mathematisches Lehrbuch nehmen, den Text in Brailleschrift und eventuelle Bilder daneben in Brailleschrift konvertiert als Relief abdrucken. Dies ersetzt aufwendige Verfahren und kann Lehrern wie Schülern helfen, effektiver und anschaulicher, sogar graphische Inhalte zu verstehen.

Literatur:

[1] http://de.wikipedia.org/wiki/DirectX [Stand vom 11.10.07]
[2] http://de.wikipedia.org/wiki/CCD-Sensor [Stand vom 11.10.07]
[3] http://www.caretec.at/ [Stand vom 11.01.2008]
[4] http://www.blindnews.eu/hilfsmittel/10157.html [Stand 07.01.2009]
[5] http://www.tzi.de/~visser/GIS/Tutorial/anhang/glossar/v.htm [Stand 11.10.2007]
[6] http://userserv.hochschule-reutlingen.de/~hug/Lehrveranstaltungsmaterial/I3/Vortraege/05-06-WS/Liedtke/3D-Grafik_mit_DirectX.pdf [Stand 11.10.2007]
[7] Smolka, Gert: Programmierung – Eine Einführung in die Informatik. Fachrichtung Informatik, Universität des Saarlandes, September 2007