Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Projekt 1
SoSe 2010

Philipp Teske

Human Fall Detection

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Einleitung

Deutschlandweit kommt, es Schätzungen zufolge, jährlich zu über 100.000 Stürzen von älteren Menschen (>65 Jahren). Die Ursachen hierfür können sowohl körperliche Beeinträchtigungen wie auch Risikofaktoren in der Umwelt sein. Im Wohnbereich sind besonders glatte oder nasse Fußböden, schlechte Beleuchtung und Stolperfallen wie lose liegende Teppiche oder Kabel eine häufige Ursache für Sturzverletzungen. Oftmals kommt die Person mit einem Schrecken davon, nicht selten jedoch führt ein Sturz zu schweren Verletzungen. Diese bedeutet häufig lange Genesungszeiten und Krankenhausaufenthalte. Hinzukommend ist besonders bei älteren Personen festzustellen, dass sie sich aus Angst vor weiteren Stürzen vorsichtiger und weniger bewegen als vorher. Dies hat zur Folge, dass die Muskelkraft nachlässt, sie somit in ihren Bewegungen unsicherer werden und es zu weiteren Stürzen kommt.

Da Stürze in den meisten Fällen aus einer Verkettung mehrerer Faktoren bestehen, lassen sie sich nur schwer verhindern. Daher ist es umso wichtiger, für eine schnelle Genesung gestürzter Menschen zu sorgen und ihnen so weit wie möglich die Angst vor einem erneuten Sturz zu nehmen.

Ein Aspekt ihre Furcht zu lindert ist, für eine schnelle Hilfe nach einem Unfall zu sorgen. Dies gibt besonders alleinlebenden Personen die Sicherheit, im Notfall nicht alleine zu sein.

2 Zielsetzung

Endgültiges Ziel beider Projekte und der Masterarbeit ist die Entwicklung eines Systems, welches mit Hilfe von optischen und kapazitiven Sensoren in der Lage ist, Bewegungen zu analysieren um Stürze und Unfälle von Personen zu erkennen. Das erste Projekt befasst sich mit dem Aufbau der Arbeitsumgebung. Hierzu gehört, sich mit der gegebenen Hardware vertraut zu machen und bereits vorhanden Software auf ihre Verwendbarkeit hin zu testen. Letzteres besteht aus der Evaluierung mehrerer Software-Bibliotheken aus dem Bereich der Bildverarbeitung sowie der Implementierung verschiedener Algorithmen zur Bewegungserkennung.

3 Hardware

Für die optische Bewegungserkennung dient eine Überwachungskamera der Firma *Mobotix*. Diese besitzt neben einer hochauflösenden Optik eine Gegensprechanlage mit Lautsprecher und Mikrofon. Der integrierte Prozessor erlaubt mehrere Sichtfunktionen, darunter eine Panoramasicht sowie eine 360°-Rundumsicht mit stufenlosem Zoom. Anhand der Rundumsichtfähigkeit der Kamera ist es möglich einen Raum mit nur einer,

an der Decke befestigten Kamera vollständig zu überwachen. Die Kamera lässt sich direkt in ein Netzwerk integrieren, über das sie auch mit Strom versorgt wird. Die Bilder lassen sich wahlweise über den integrierten Webserver oder als Videostream auslesen. Die Abfrage erfolgt hierbei über einen HTTP-Request wodurch es möglich ist, für jeden Stream explizite Kameraeinstellungen festzulegen. So lässt sich u.a. die gewünschte Bildgröße und -qualität sowie die Videokodierung angeben.
Die kapazitiven Näherungssensoren, welche später integriert werden, haben den Vorteil, dass sie sich unsichtbar unter/im Fußboden anbringen lassen. Sie bestehen aus einem Schwingkreisgenerator, an dem sich verschiedene Arten von Elektroden anschließen lassen, sowie einer Schnittstelle zum Auswerten der Schwingungsfrequenz. An dieser Stelle sei jedoch auf die Arbeiten von Frank Hardenack und Oliver Dreschke verwiesen, die sich eingehender mit den Verhalten verschiedener Elektroden und Schwingkreisgeneratoren auseinandergesetzt haben.

4 Software

4.1 Bibliotheken zur Bildverarbeitung

Bevor ein Softwaresystem in die Praxis umgesetzt werden kann, muss sich die Frage gestellt werden, wie es am effizientesten realisiert werden kann. So macht es bei einem Bildanalysesystem wenig Sinn, das Rad neu zu erfinden und alle Bildverarbeitungsfunktionen selber zu schreiben. Zudem dies tiefgreifende Kenntnisse über die theoretischen Zusammenhänge der Bildverarbeitung voraussetzt und mit einem erheblichen Zeitaufwand verbunden ist. Stattdessen bietet es sich an, eine der zahlreichen kostenpflichtigen oder kostenlosen Bibliotheken zu verwenden. Diese haben die wichtigsten Funktionen bereits implementiert und können in eigene Systeme integriert werden.

OpenCV

Eine der am weitesten verbreiteten kostenlosen Bibliotheken für Bildverarbeitung ist die *Open Computer Vision Library* (*OpenCV*). Das *OpenCV* Projekt wurde ursprüngliche von *Intel* entwickelt und wurde später von *Willow Garage* übernommen. Die Bibliothek wird unter BSD (Berkeley Software Distribution)-Lizenz veröffentlicht, was bedeutet das sie frei eingesetzt und auch in kommerziellen Projekten verwendet werden darf. Lediglich der Copyright-Vermerk muss erhalten bleiben.
OpenCV ist vollständig in C/C++ programmiert und von Anfang an auf eine hohe Verarbeitungsgeschwindigkeit ausgelegt. Im Laufe der Zeit wurden die

implementierten Algorithmen immer wieder den aktuellen Forschungsergebnissen angepasst und erweitert. Die Bibliothek umfasst in ihrer aktuellen Version (2.1 – April 2010) mehr als 500 Algorithmen, darunter Funktionen zum maschinellen Lernen, Gesichtsdetektion sowie verschiedene Filter [1].

ITK

Eine weitere quelloffene Bibliothek ist das *Insight Toolkit* (*ITK*). Das Projekt entstand 1999 aus einer Ausschreibung der *US National Library of Medicine*. Entwickler der Bibliothek ist ein Konsortium aus mehreren Universitäten und Firmen. Der Entwicklungsschwerpunkt von ITK war die Visualisierung medizinischer Daten wie sie von Computertomographen oder ähnlichen Geräten geliefert werden. Als Programmiersprache diente C/C++, zusätzlich lässt sich die Bibliothek jedoch an verschiedene Programmiersprachen anbinden. ITK wird ebenfalls unter der BSD Lizenz veröffentlicht und unterstützt in der aktuellen Version (3.20 – Juli 2010) Algorithmen zur Segmentierung und Darstellung von Daten, sowie verschiedene Filter (u.a. LaPlace, ColourConversion, Gradienten) [2].

HALCON

Bei *HALCON* handelt es sich um eine weit verbreitete kommerzielle Bildverarbeitungsbibliothek. Entwickler ist die *MVTec Software GmbH*. *HALCON* ist auf die Verwendung im technischen und medizinischen Umfeld ausgelegt. Neben einer hohen Verarbeitungsgeschwindigkeit bietet sie Schnittstellen zu vielen gängigen industriellen Kamerasystemen. Des weiteren lässt sich die Bibliothek mit den Programmiersprachen C/C++, C#, Visual Basic, .NET sowie Delphi verwenden. Die Anbindung erfolgt wahlweise über eine API-Schnittstelle oder der mitgelieferten Entwicklungsumgebung *HDevelop* (Abbildung 1). Da komplexe Abläufe bereits zusammengefasst als sogenannte *procedures* vorliegen, erlaubt *HDevelop* schnelles Prototyping und eine komfortable Erstellung von komplexen Bildverarbeitungssystemen. Nach Abschluss der Entwicklung lassen sich die verwendeten *procedures* direkt in die entsprechende Programmiersprache exportieren.

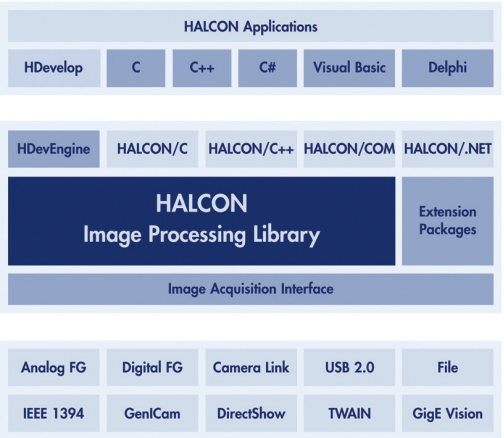

Abbildung 1: HALCON-Architektur
(Quelle: [3])

Neben der normalen Version von *HALCON*, welche für Shared-Memory- und

Multicore-Systeme ausgelegt, steht noch eine spezielle Version für leistungsschwächere Systeme zur Verfügung. *HALCON* embedded erlaubt die Integration von Bildverarbeitungssystemen auf verschiedenen eingebetteten und mobilen Systemen. Neben vielen gängigen Prozessorarchitekturen wird eine große Anzahl an Betriebssystemen, wie Windows CE, VxWorks oder Linux unterstützt. Beide *HALCON* Varianten bieten in der aktuellen Version 9 mehr als 1400 unterschiedliche Operationen und Abläufe [4]. So sind bereits komplexe Vorgänge wie 3D- oder Stereo-Kamera-Kalibrierung, Linienextraktion und Objektmatching fertig hinterlegt. Preislich liegt *HALCON* bei 1.750 Euro für Hochschulen und 5.500 Euro für kommerzielle Kunden (Stand: August 2010). Die Lizenzierung kann wahlweise über die MAC-ID des PCs, was das Produkt an einen einzelnen Rechner bindet, oder über einen USB-Key geschehen.

Alle aufgeführten Bibliotheken bieten einen Funktionsumfang, der das Arbeiten erheblich erleichtert und wären somit für die weitere Verwendung im Projekt geeignet. Jedoch ermöglicht die frei verfügbare Demo-Version von *HALCON* nur einen sehr eingeschränkten Testbetrieb. So lassen sich zwar eigene Programme in der Entwicklungsumgebung *HDevelop* erstellen und Beispielprogramme ausführen, die Export-Funktion sowie die Schnittstellen-API zu verschiedenen Programmiersprache steht leider nicht zur Verfügung.
Ein weiterer entscheidender Faktor, welcher gegen *HALCON* spricht, ist der hohe Anschaffungspreis. *OpenCV* und *ITK* haben den Vorteil, das beide Bibliotheken kostenlos erhältlich sind und in kommerziellen Projekten verwendet werden dürfen. Die Entscheidung viel schlussendlich auf *OpenCV*. Ausschlaggebend hierfür war die weite Verbreitung der Bibliothek und die damit zusammenhängenden umfangreichen Dokumentationen und Anleitungen. Darüber hinaus bietet *OpenCV* bereits eine integrierte Schnittstelle zu der in Kapitel 3 beschriebenen Kamera. Tabelle 1 zeigt eine Übersicht der Vor- und Nachteile der drei Bibliotheken.

	OpenCV	**ITK**	**HALCON**
Positiv	- kostenlos - sehr umfangreich - gute Dokumentation - viele Anleitungen - einfacher Aufbau	- kostenlos	- sehr umfangreich - Anbindung an viele Kamerasysteme
Negativ	- Funktionen teilweise schwer zu finden	- Kamerasysteme nur über Umwege integrierbar	- kostenpflichtig - Test vieler Funktionen in Demo-Version nicht möglich

Tabelle 1: Vergleich OpenCV, ITK und HALCON

4.2 Bewegungserkennung

Bewegungen in Bildern lassen sich leicht durch Veränderungen der Farbwerte in den einzelnen Kanälen erkennen. Hierfür werden üblicherweise zwei oder mehr aufeinanderfolgende Bilder voneinander subtrahiert [5]. Da Pixel, bei denen keine Bewegung stattfand, in allen Bilder den gleichen Farbwert haben, haben diese sich durch die Subtraktion auf. Übrig bleibt ein Bild, welches nur die Pixel enthält, in denen sich der Wert zwischen den verwendeten Bildern verändert hat. Abbildung 2 veranschaulicht die Subtraktion zweier Bilder. Wie im am Unteren Rand und Links neben der Person erkennbar ist, werden auch Bereiche erkannt, welche unwichtige oder sogar überhaupt keine Bewegungen enthält. Schuld daran sind zum einen Lichtschwankungen, wie sie z.B. durch Schatten oder Wolken entstehen, zum anderen macht sich hier das herstellungsbedingte Bildrauschen der bemerkbar.

Abbildung 2: Differenzbild

Durch die einfache Subtraktion der Bilder kann es bei ungünstigen Farbwerten dazu kommen, dass einzelne Pixel oder ganze Bereiche schwarz werden und somit nicht als Bewegung erkannt werden. Deutlich wird dies am Gesicht und Arm der Person im unteren Bild von Abbildung 2. Um dieses Verhalten so weit wie möglich zu unterdrücken, wurden verschiedene Methoden entwickelt. So werden in einigen Verfahren die Bilder aus dem RGB- in den HSI-Farbraum, welcher mehr der menschlichen Farbwahrnehmung ähnelt, konvertiert. Hierin lassen sich Schatten wesentlich einfacher von den eigentlichen Bewegungen unterscheiden, da diese normalerweise nur die Helligkeit *I* eines Pixels verändern, nicht aber seine Farbe *H*. Um das Rauschen der Kamera und Lichtschwankungen zu unterdrücken, wird häufig ein sogenanntes Hintergrund-Modell verwendet. Dieses setzt sich aus einer beliebigen Anzahl an Kamerabildern zusammen. Oftmals jedoch wird nur ein Bild aufgenommen und das jeweils aktuelle Kamerabild zusammen mit einem Gewichtungsfaktor hinzugerechnet. Auf diese Weise passt sich das Hintergrund-Modell mit der Zeit automatisch dem Bildrauschen und aktuellen Lichtverhältnissen an. Zusätzlich lässt sich über den Gewichtungsfaktor die Geschwindigkeit, mit der sich das Modell anpasst, regeln.

Anschließend kann das Differenzbild über eine Schwellwertfunktionen in ein Binärbild konvertiert werden. Hierbei wird der Schwellwert so gewählt, das

Bewegungen noch erkannt werden, das Rauschen der Pixel jedoch herausgefiltert wird. Das Festlegen des Schwellwertes kann entweder manuell geschehen, was nur für Umgebungen mit gleichbleibenden Helligkeitsbedingungen tauglich ist, oder durch einen Algorithmus, welcher den Schwellwert automatisch den jeweiligen Bedingungen anpasst. Das erstellte Binärbild wird jedoch noch vereinzelt Bereiche enthalten, die ein Bildrauschen aufweisen oder in denen ein Teil des Objekts durch die Schwellwertfunktion herausgefiltert wurde (Abbildung 3).

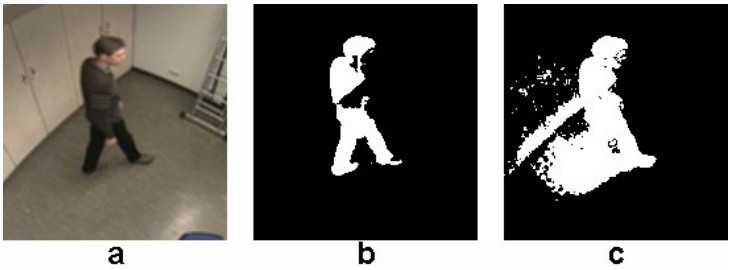

Abbildung 3: Schwellwertverfahren
a: Originalbild, b: Schwellwert zu niedrig, c: Schwellwert zu hoch

Dies lässt sich durch verschiedene Verfahren ausgleichen. Eines hiervon – häufig als *open-close* bzw. *close-open* Methode bezeichnet – entfernt kleinere Bereiche und füllt unerwünschte Lücken aus [6]. Hierfür wird durch verknüpfen der morphologischen Grundoperationen *dilatation* und *erosion,* Funktionen mit der Bezeichnung *open* und *close* erstellt. *Open* ermöglicht es kleinere Bildstrukturen zu entfernen und *close* füllt Zwischenräume und Lücken auf. Das besondere an diesen Funktionen ist, dass die groben Umrisse des Objekts erhalten bleiben.
Nachdem das Binärbild bereinigt und von Störungen befreit wurde, können hierauf aufbauend verschiedene Methoden zur Bewegungsanalyse und Sturzerkennung angewandt werden [7].

5 Fazit

Im Projekt wurden mehrere Software-Bibliotheken aus dem Bereich der Bildverarbeitung evaluiert. Es hat sich herausgestellt, dass die kostenlose Bibliothek *OpenCV* für die Verwendung in dem Projekt am besten geeignet ist. Sie enthält eine Vielzahl von Funktionen, welche die weitere Arbeit erleichtern. Zudem ist bereits eine Schnittstelle zu der in Kapitel 3 genannten Kamera integriert, was eine einfache Anbindung erlaubt. Im weiteren Verlauf des Projektes wurde versucht, verschiedene Verfahren der Bewegungserkennung zu implementieren [8][9][10]. Hierbei hat sich gezeigt, dass ein Großteil der Rechenleistung für die Generierung des Hintergrundmodels benötigt wird. Je nach Komplexität des Models

verzögerte sich bei Bilder mit einer Auflösung von 800x600 Pixeln die Verarbeitung um mehrere Sekunden. Des weiteren hat sich gezeigt, dass das Hintergrund-Model sehr empfindlich auf Bildveränderungen, welche durch die Kamera ausgelöst werden, reagiert. So führen vermutlich bestimmte Änderungen in der Umgebungshelligkeit dazu, dass die Kamera ihre automatische Blendeneinstellung neu justiert und somit das Bild kurz überbelichtet ist. Dies bewirkt im Hintergrundmodel eine starke, sprunghafte Abweichung vom eigentlichen Bild. Zusammen mit den aktuellen Bilddaten der Kamera, welche nach der Neujustierung nun eine andere Helligkeit aufweisen, führt dies dazu, dass nahezu im kompletten Bild Bewegungen erkannt und es somit unbrauchbar wird. Dies und die Verarbeitungs-Geschwindigkeit des Hintergrundmodels gilt es im nächsten Projekt weiter zu optimieren. Zudem soll eine zuverlässige Klassifizierung der erfassten Objekte implementiert werden, um Personen von anderen Dingen unterscheiden zu können.

6 Abbildungsverzeichnis

7 Literaturverzeichnis

1 *Open Computer Vision Library* - URL sourceforge.net/projects/opencvlibrary/

2 *ITK - Segmentation & Registration Toolkit* - URL www.itk.org/

3 *HALCON 9.0 Quick Guide*

4 *MVTec HALCON* - URL www.mvtec.com/halcon/

5 Massimo Piccardi: *Background subtraction techniques: a review*, 2004

6 M. O. Franz: *Morphologische Filter*, 2007

7 Philipp Teske: *Human Fall Detection - Related Work*, 2010

8 Shao-Yi Chien, Shyh-Yih Ma, and Liang-Gee Chen: *Efficient Moving Object Segmentation Algorithm Using Background Registration Technique*. IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 12, NO. 7, 2002

9 Rita Cucchiara, Costantino Grana, Massimo Piccardi, Andrea Prati: *Detecting Moving Objects, Ghosts, andShadows in Video Streams*. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 25, NO. 10, 2003

10 *Blob detection V: growing regions algorithm* - URL geekblog.nl/entry/24

Zugriffsdatum der Webseiten: 28.08.2010