Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Jens Shih

Bildverarbeitungsbibliothek zur Bewegungsdetektion für ein eingebettetes System

Fachbereich Elektrotechnik und Informatik
Department of Electrical Engineering and Computer Science

Jens Shih

Bildverarbeitungsbibliothek zur Bewegungsdetektion für ein eingebettetes System

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik (Bachelor)
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Andreas Meisel
Zweitgutachter: Prof. Dr. Wolfgang Fohl

Abgegeben am 06. Juni 2008

Thema der Bachelorarbeit

Entwicklung einer Bildverarbeitungsbibliothek für Bewegungsdetektion für ein eingebettetes System

Stichworte

ARM9, Embedded Linux, V4L, Bewegungsdetektion, USB Webcam, Browseranwendung u. CGI

Kurzzusammenfassung

Das Ziel dieser Arbeit ist es, für das Produkt TAINY EMOD der Firma Dr. Neuhaus Telekommunikation GmbH, welches einen ARM9 Mikrocontroller besitzt, und auf dem ein Linux- Kernel läuft, eine robuste Bildverarbeitungsbibliothek für Bewegungsdetektion inkl. Userinterface zu entwickeln. Die daraus resultierende Anwendung sollte auch benutzerfreundlich sein und sich fern mittels HTML und Webapplikation parametrieren und steuern lassen. Dazu sollte die Bibliothek auch in der Lage sein, die von der USB- Webcam gelieferte Frames zu dekodieren, verarbeiten, und schließlich als JPEG formatiertes Bild in den Flashspeicher zu sichern.

Title of the paper

Evolution of an image processing library of motion detection for an embedded system

Keywords

ARM9, Embedded Linux, V4L, Motion Detection, USB Webcam, Browser application, CGI

Abstract

The goal of this thesis was to develop a robust image processing library for motion detection, including a user interface for the wireless EDGE Router, which is the new product of the Dr. Neuhaus Telecommunications Company. The TAINY EMOD has an ARM9 micro controller, which a Linux Kernel is running on it. The resulting application should also be user-friendly and controlled and parameterized by using HTML and Web application. In addition, the library shall have the ability to grab frames from the USB Webcam, to decode and process the data and finally to store the images into a JPEG format and into the flash memory.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Vorwort

Im Rahmen eines Industriepraktikums bei der Firma Dr. Neuhaus Telekommunikation GmbH habe ich als vollständiges Teammitglied der Softwareentwicklung mit am Produkt TAINY EMOD gewirkt.
Während dieser Zeit konnte ich meine Unix- Linux- Programmierkenntnisse weiter vertiefen und mich für Embedded Programmierung begeistern. Auch deshalb möchte ich meinen besonderen Dank dem Leiter der Entwicklung Herrn Mathias Nieting aussprechen, der mir auch die hier vorliegende Arbeit angeboten hat.
Des Weiteren gilt meinem Dank den Kollegen der Firma Dr. Neuhaus, die mich mit Rat und Tat unterstützt haben, und vielen Dank auch an Prof. Dr. Meisel für die nützlichen Ratschläge und die gute Betreuung dieser Bachelorarbeit. Ansonsten bedanke ich mich bei meiner Familie und Freunden, die mich voll unterstützt haben, ohne deren Unterstützung diese Arbeit weitaus hürdenreicher anzufertigen gewesen wäre.

2 Einführung

Die Mikrokontrollertechnologie hat in den letzten Jahren enorme Fortschritte in Punkto Leistung, Komptabilität und Performance gemacht. So auch der, der in der TAINY EMOD- Familie eingesetzte ARM9 AT91SAM9260 Mikrokontroller von Atmel, dessen CPU mit über 200 MHz getaktet wird, und im Stande ist über 200 MIPs zu verrichten. Das EMOD wird in erster Linie als ein intelligentes Wireless Router für industrielle Ethernet- Netzwerke eingesetzt, um M2M (Mobil to Mobil) Kommunikation bereitzustellen, mit dem man z. B. digitale Stromzähler fern über GPRS/EDGE auslesen kann.
Die Anwendungsbandbreite eines GPRS/EDGE Wireless Routers ist riesig, besonders wenn dieser für ein mobiles Gerät einen leistungsstarken CPU besitzt, auf dem ein ganzer Linuxkernel läuft und gleichzeitig reichlich mit Speicher, 64MB RAM und ebenso viel ROM, ausgestattet ist. Aus diesen Gründen wurde auch die Idee entwickelt, eine Bildverarbeitungsbibliothek für Bewegungsdetektion zu entwickeln, die standalone auf dem mobilen Router mit angeschlossener USB- Webcam laufen soll. Außerdem, sollte die Bewegungserkennungssoftware über das Internet vom Benutzer parametriert und gesteuert werden können. Die Anwendung sollte auch in der Lage sein, JPEG komprimierte Bilder zu liefern bzw. den Benutzer über detektierte Bewegungen per SMS zu benachrichtigen. Im Hinblick auf die letzte geforderte Eigenschaft, müssen natürlich die Algorithmen und Funktionen so durchdacht entwickelt werden, dass eine Alarm- SMS erst bei aller Wahrscheinlichkeit einer erfolgreichen Detektion verschickt wird, um das Risiko eines Fehlalarms weitgehend zu minimieren. Somit baut diese Arbeit auch auf die Gesamtheit der kompletten Entwicklung auf, die bei der Inbetriebname der USB-Webcam- Treibermodulen für den Linuxkernel anfangen und bei dem Userinterface in Form einer Webapplikation für Benutzereinstellungen enden.

2.1 Problemstellung

Eines der Hauptaugenmerkmale, das sich bei der softwarebasierten Bildverarbeitung stark hervorhebt, ist die Geschwindigkeit des ausführenden Computers. Auch stellen die anfallenden großen Datenmengen als ein großes Problem dar. Erschwerend

kommt hinzu, dass die üblichen auf dem Markt befindlichen USB Webcams keine Schwarz/Weiß- Bilddaten liefern können, die für die Bildauswertung von Bewegungen vollkommen ausreichen, und natürlich auch weniger an Datenmengen bedeuten würden.
Aufgrund, dass die gewöhnlichen Webkameras RGB- anstatt Schwarz/Weiß- Frames produzieren, sind diese auch dreimal so groß und benötigen wiederum dreimal so viel Zeit, um von der Webcam über den USB- Kanal in die Anwendung zu gelangen. Ein zusätzliches großes Hindernis in der Linuxwelt ist wohl die mangelnde Treiberunterstützung der Hardware seitens der Hersteller. So geben nur die wenigsten Hersteller ihre Transportprotokolle frei, mit der man wenigstens einen Treiber nachbauen könnte. Dieses Manko findet man leider auch bei den Webcamherstellern vor.

Außerdem ist es nicht üblich, softwarebasierte Bildverarbeitung in einem eingebetteten Gerät auszuführen, bei dem im Vergleich zu den PCs ein erheblicher Mangel an Performance- und Speicherleistung besteht. Auch deshalb muss hier quasi Neuland beschritten werden, wenn die oben genannten Anforderungen erfüllt sein sollen.

2.2 Gliederung dieser Arbeit

In Kapitel 3 wird die Eigenschaft des EMODS beschrieben. Zudem wird der Aufbau des ARM9 Mikrokontrollers erläutert. Kapitel 4 beschäftigt sich mit dem Linuxbetriebssystem. Dabei wird erklärt, wie man Anwendungen für die Zielmaschine cross- kompiliert und eine Buildrootumgebung anrichtet, um schließlich das Webcam- Treibermodul zu erzeugen. Kapitel 5 hat das Thema V4L (Video for Linux), was eine Applikationsschnittstelle für den Zugriff auf die USB-Webcam- Hardware ist. Dort wird gezeigt, wie man die Webcam initialisiert, und wie man Bilddaten von ihr bekommt.
Die Bibliotheksfunktionen zum Erstellen und Entpacken von JPEG- Bilder werden im Kapitel 6 behandelt. In Kapitel 7 werden die üblichen Verfahren für Bewegungserkennung vorgestellt und erläutert. Später, in Kapitel 8 wird die Realisierung der Bibliothek inkl. eines Userinterfaces in Form einer Webapplikation für die Bewegungsdetektion diskutiert. Dabei wird auch diese Webapplikation für die Benutzereinstellung näher beschrieben. Anschließend wird anhand mehrerer Beispiele, das Ergebnis dieser Arbeit vorgestellt.

3 Aufbau der Hardware

Abbildung 3.1: TAINY EMOD-V2-IO

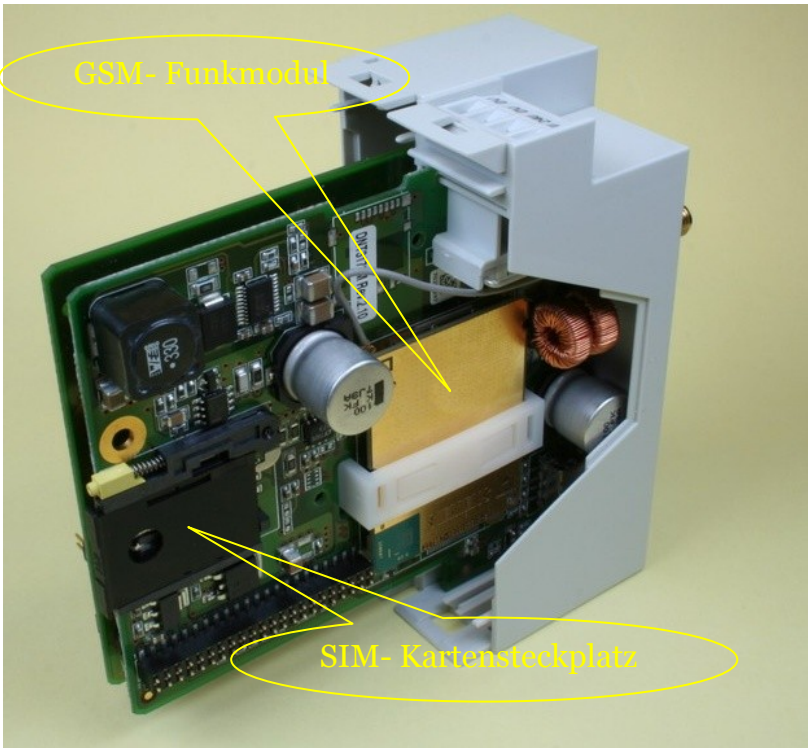

Abbildung 3.2: TAINY EMOD-V2-IO Offen

3.1 TAINY EMOD

Das TAINY EMOD von Dr. Neuhaus Telekommunikation GmbH ist ein mit einem ARM9- Mikrokontroller, einem Netzteil mit variabler Eingangsspannung von 12-60 VDC und einem GPRS/EDGE Funkmodul ausgestattetes Funkwireless Router, der entfernte Stationen durch intelligente Kommunikationsmanagement in ein IP-Netzwerk einbinden kann. Vertrauliche Daten lassen sich so drahtlos und sicher über GSM (Global System for Mobil)- Netz übertragen. Robuste und kompakte Standardbauform kennzeichnet ihn aus, so dass er für die Hutschienenmontage im Schaltschrank im industriellen Umfeld bestens geeignet ist.

3.1.1 Interface

Das TAINY EMOD besitzt verschiedene Interfaces, wie eine 10/100 (Base- T) RJ45-Buchse, die auch als Applikationsschnittstelle, z. B. für Stromzähler, dient. Zudem weist er eine USB-A Schnittstelle aus, an der eine USB- Webcam angeschlossen werden kann. Ein Meldeeingang- und Ausgang (U 5..30 VDC) kann für individuelle Zwecke verwendet werden, um bestimmte Ereignisse auszulösen. Und ein breiter Energieversorgungseingang von 12-60 VDC und 365-92 mA rundet das Spektrum ab.

3.1.2 Funktionalität

Eine integrierte Linux- Stateful Inspection Firewall schützt Anwendungen vor unberechtigtem Zugriff aus dem Internet. Darunter fallen auch diversen Spoofing- bzw. DoS- Angriffe. Die Dienste der NAT (Network Address Translation o. IP Masquerading) und des Port Forwardings ermöglichen es, Rechner innerhalb eines LANs von Außen ansprechbar zu machen und bietet dadurch beste Routerfunktionalität an.
Das TAINY EMOD V2IO, ein wertersteigertes Exemplar der EMOD- Produktfamilie, kann zudem auch als VPN (Virtual Private Network)- Router fungieren.

3.1.3 Radio

Wie bereits erwähnt ist mit dem EMOD eine Kommunikation im EDGE (Enhanced Data Rates for GSM Evolution) Mode möglich. Es unterstützt Class 12 mit bis zu vier Uplinks, vier Downlinks und maximal fünf Slots. Die im EDGE- Mode angewandten Modulations- und Kodierungsverfahren MCS-1 bis 9 erlauben es, Daten mit bis zu 59,2 kbps pro Slot zu übertragen. D. h. es ist theoretisch möglich vier Dowload-Slots (59,2 kbps * 4 = 236,8 kbps) und ein Upload-Slot (59,2 kbps) oder umgekehrt für einen einzelnen Benutzer zu öffnen. Das EMOD hat die Mobile Station Class B Zulassung. Im normalen GPRS (General Packet Radio Service)- Mode können auch bis zu vier Uplinks und bis zu vier Downlinks mit max. 5 Slots geöffnet werden. Hierbei können die Kodierungsverfahren CS-1, CS-2, CS-3 und CS-4 mit max 20 kbps angewandt werden. Dadurch ist eine theoretische Download-Uploadgeschwindigkeit von 20 kbps * 4 = 80 kbps und 20 kbps oder umgekehrt möglich. Im CSD / MTC- Verfahren können Daten mit 2.4, 4.8, 9.6 oder 14.4 kbps transportiert werden. Daneben kann das EMOD auch SMS an beliebige Rufnummer versenden, z.B. um Bestriebsstörungen melden[EMOD].

3.2 ARM9 Mikrocontroller

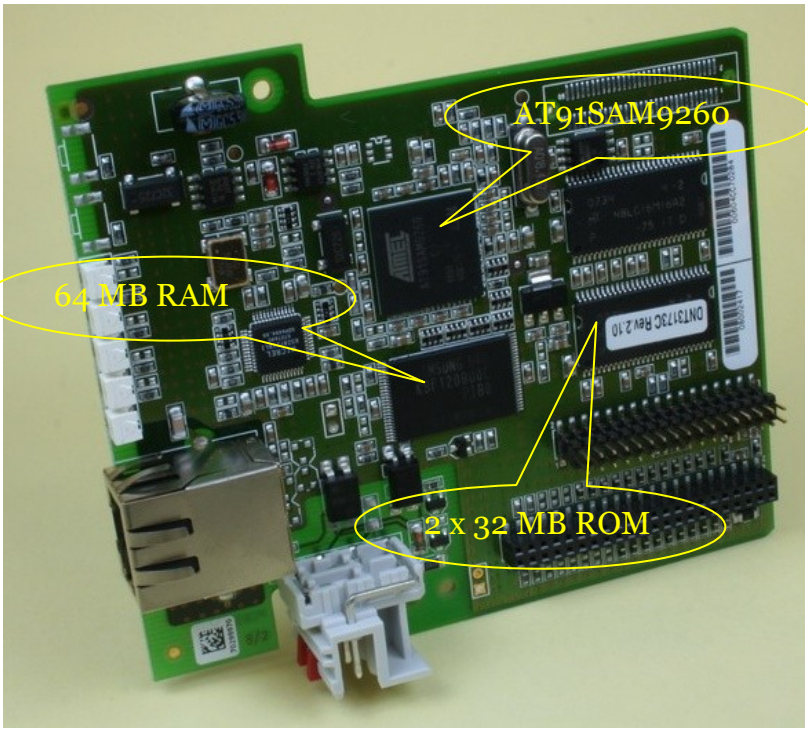

Abbildung 3.3: TAINY EMOD Offen

Das Herzstück des EMODs ist der ARM9 Mikrokontroller mit der Bezeichnung „AT91SAM9260", der auch das Produkt einer pin-kompatiblen ARM9 Mikrocontroller- Familie aus dem Hause ATMEL ist [Atmel 2008]. D. h. zukünftige leistungsfähigere Mikrokontroller dieser Serie können ohne bzw. mit nur geringen Veränderungen der Leiterplatine auf frühe Hardwareentwicklungen aufgesetzt werden, um mehr Performance zu erhalten. Zudem unterstützt der AT91SAM9260 die deterministische Real-Time Programmausführung, um Plattform auch für zeitkritische Anwendungen zu sein. Ebenso können die Betriebssysteme Embdedded Linux oder WindowsCE im dem Mikrokontroller laufen.
Die besondere Eigenschaft des AT91SAM9260 ist wohl seine interne Bus-Bandbreite. Sie besteht aus 24 DMA Kanäle und einer 7 lagigen High- Speed AHB (Bus Matrix). Aufgrund, dass die AHB alle Master und Slaves im Mikrokontroller parallel verbindet, können Daten zwischen den Peripherien und on- sowie Off-Chip Memorys ohne jegliche CPU-Beteiligung transportiert werden, was die Ausführung des Mikrokontrollers so effizient macht. Hiervon profitiert auch die Bildverarbeitungsbiliothek, denn während neue Frames aus der USB- Webcam geholt werden, kann die CPU die bereits eingelesenen Frames verarbeiten.

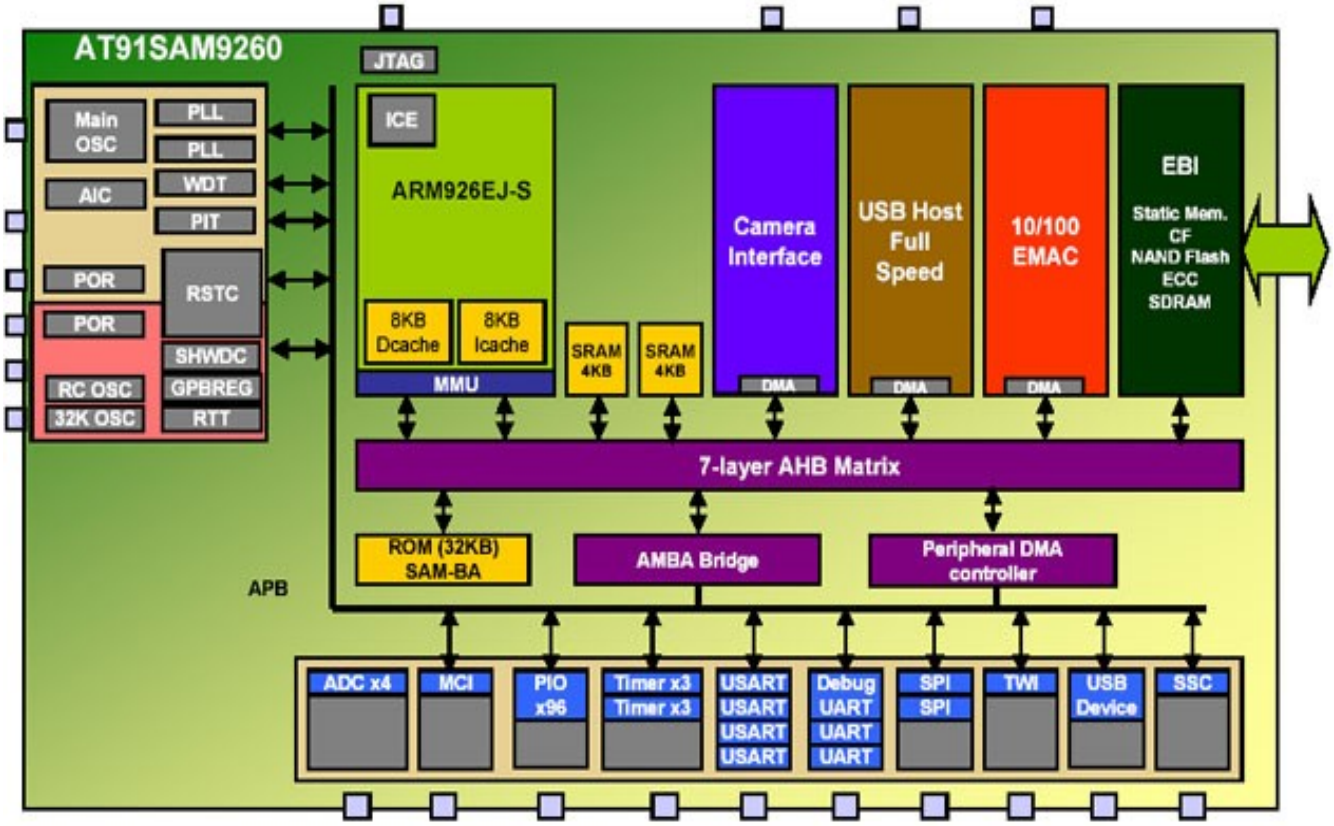

Abbildung 3.4: AT91SAM9260 Block Diagramm

Nebenbei nimmt ein 8kB großer Instruktionscache die Interrupt Routinen entgegen und ermöglicht eine vorhersehbare Interruptantwortzeit. Der System- Controller stellt viele Überwachungsfunktionen zur Verfügung, wie z. B. einen RC-Oszillator, einen 8-Level Prioritäten Vektor-Interrupt Controller, PLLs, einen Real-Time Intervall- und Watchdog- Timer, einen Reset- und Shutdowncontroller und ein Backup Register. Mit Hilfe des Shutdown- Kontrollers kann die MCU in einen Ultra Low Power- Mode mit einem niedrigen Stromverbrauch von ca. 10 µA versetzt werden.

3.2.1 Peripherien

Der AT91SAM9260 besitzt einen CPU- Kern, der über 200 MIPS (Millionen Instruktionen pro Sekunde) verrichten kann. Außerdem ist er mit einem CMOS-Kamera Interface, das in dieser Arbeit zu Gunsten handelsüblicher Webcams nicht benutzt wird, ausgestattet. Daneben zählen sieben USARTs, eine 10/100 Ethernet-Schnittstelle, ein 12 Mbps USB Device und einen Host Controller mit On-Chip Transceiver, eine externe Businterface (EBI) für SDRAM, Flash und NAND- Flash mit eingebauter ECC zu seiner Ausstattung. Auch befinden sich auf der MCU eine SD-, SDIO- und MMC- Interface, drei SSC (Synchronous Serial Controllers), zwei SPI (Master/Slave Serial Peripheral Interfaces), ein 16-bit Timer- Counter, I²C (TWI) und JTAG Boundary Scan.

4 Betriebssystem & Buildumgebung

Die in dem Mikrocontroller befindliche CPU besitzt eine ARM9- Architektur (Acorn Risc Machine). Aufgrund dieses Merkmals versteht die CPU des AT91SAM9260 Mikrokontrollers den Maschinencode der ARM- CPU- Familie.
ARM ist ein Vertreter der RISC- Architektur (Reduced Instruction Set Computing). Die ARM- Architektur besitzt einen effizienten Befehlssatz, und ist für Optimierung im Bereich der Ausführungsgeschwindigkeit und der Stromaufnahme prädestiniert. Da die ARM- CPUs nicht binärcodekompatibel zu üblichen x86 CPUs sind, muss ein Programm, das auf eine ARM- CPU laufen soll, auf einem PC mit Hilfe eine Crosskompilers cross- kompiliert werden. Auch das Embedded Linuxbetriebssystem wird auf diese Weise für den ARM9- Mikrokontroller auf einem Fremdrechner, ebenfalls mit Linus als Betriebssystem, cross- kompiliert.

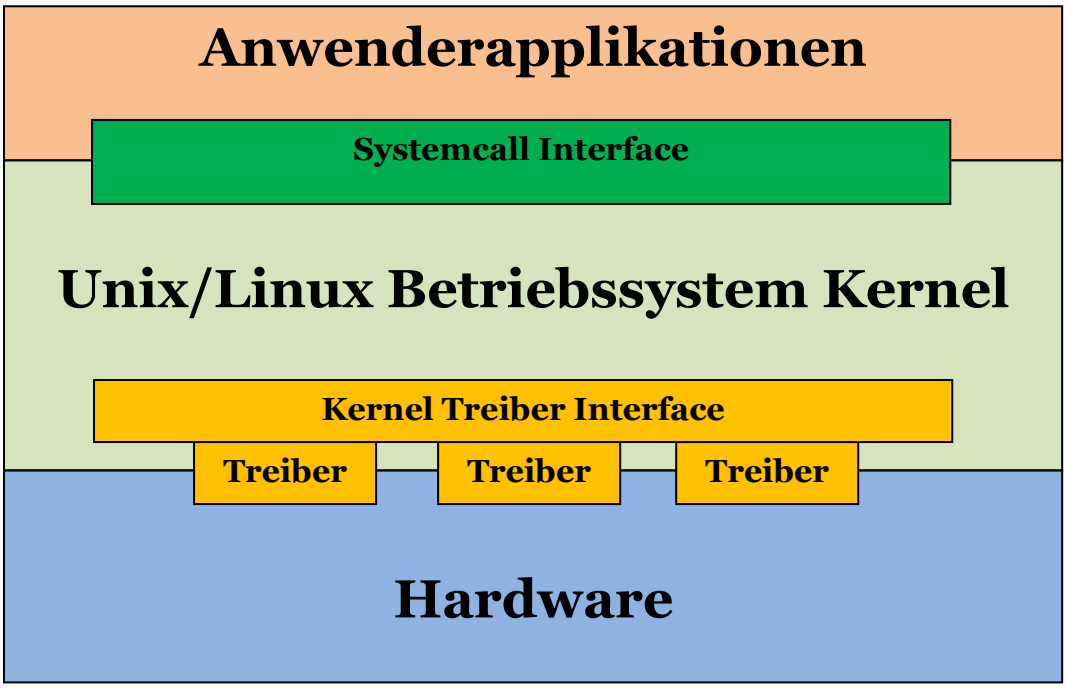

Abbildung 3.5: Unix/Linux Schichtenmodell

Das Linux Betriebssystem lässt sich am Besten als ein Dreischichtenmodell darstellen. Auf der untersten Schicht findet man sämtliche Hardware sowie Peripherie vor. Die zweite Schicht besteht aus dem Linux Kernel, der nachfolgend näher beschrieben wird. In der letzten Schicht des Embedded Linux Betriebssystems verbergen sich verschiedene Standard- Linux- C- Bibliotheken. Viele von denen sind speziell für eingebettete Systeme speicheroptimiert. So gibt es eine Anwendung namens Busybox, in der viele Standarddienstprogramme, wie „find“, „cpy“, „mv“ usw. in einem einzigen Tool integriert sind, was die Speichergröße im Vergleich zu allen Standaloneprogrammen massiv reduziert. Auch sind in dieser Schicht Anwenderprogramme angesiedelt, bei denen die benutzten Bibliotheken vor der Übersetzung entweder statisch oder dynamisch gelinkt werden müssen.

4.1 Der Embedded Linuxkernel

Der Linuxkernel ist auch in drei Schichten aufgeteilt. Die erste Schicht ist eine Low-Level-Schnittstelle, welche für darüber liegende Schichten eine erste API für die

Hardware-Abstraktion zur Verfügung stellt. Dann folgen kleinere Kernel- Module als zweite Schicht, die für die Interpretation von strukturierten Daten aus den Filesystem- und Netzwerkprotokollen zuständig sind. Diese werden von dem Kernel empfangen oder gesendet. Auch der hier verwendete Webcam- Treiber mit der Bezeichnung „GSPCA“ funktioniert als Kernel- Treiber und macht den Zugriff auf die Webcam für die V4L- API verfügbar.
Zum Schluss besteht der Kernel aus einer fast hardwareunabhängigen High-Level-Abstraktionsschicht, die bei vielen Linux- Derivaten und Unix-Systemen gleich sind oder zumindest ähnliche Verhalten aufweisen.
Prozesse, Threads, Dateien, Sockets und Signale können in dieser Schicht generiert oder verarbeitet werden [BrDeKu 2007].

Der Embbeded- Linuxkernel für den Mikrokontroller heißt für gewöhnlich auch zImage (Zipped Image). Der Name kommt daher, da der Linuxkernel, der auf dem nicht- flüchtigen NAND- Speicher in den Arbeitsspeicher geladen wird, eigentlich eine komprimierte und sich selbst extrahierende Programmdatei ist. Wenn das System gestartet wird, entpackt sich das zImage selbstständig in den Arbeitsspeicher, jedoch nur, wenn vorher die benötigen Parameter richtig übergeben worden sind.

Nachdem der Linuxkernel richtig gestartet und die unterstützte Hardware initialisiert worden sind, wird vom Kernel das Rootfilesystem gemountet. Dabei legen die vorher übergebenen Linuxkernel-Startparameter die Art und Ort des Rootfilesystems fest. Da das Betriebssystem nicht nur den Linuxkernel beinhaltet, sondern auch aus anderen Diensten und Programmen besteht, wird der weitere Startvorgang durch die im Rootfilesystem abgelegte Konfigurations- Datei oder Programm in `sbin/init` bzw. `/etc/init` oder `/bin/init` gesteuert. Ist die Datei vorhanden, dann wird sie automatisch ausgeführt und die Dienste und Userprogramme können gestartet werden.

Ein Embedded System wie das EMOD unterscheidet sich in vielen Punkten von einem normalen Personal Computer, wie z. B. niedrigere CPU- Leistung, wenig Arbeitsspeicher, keinen Monitor, keine Festplatte, sondern ein NAND- Flash als nicht flüchtiger Speicher. Daraus ergeben sich auch spezielle Probleme oder Ansätze in der Softwareentwicklung, die später noch erläutert werden sollen.

4.2 Der Crosskompiler

Ein Crosskompiler wird benötigt, um für die verwendete Zielplattform, wie das EMOD mit seiner ARM9- CPU, ARM- Programmiercode übersetzen zu können. Dabei besteht der Crosskompiler aus einer Kombination aus binutils, dem ausführbaren C/CPP- Compilerprogramm und einer C- Laufzeitbibliothek. Eine solche Umgebung wird auch als Crosskompiler- Toolchain genannt.

Der Programmiercode, der für die Zielplattform mit Hilfe des Crosskompilers kompiliert werden soll, kann auf zweierlei Wege gelinkt werden: Dynamisch, wo die C- Laufzeitbibliothek in der Verzeichnisstruktur der Crosskompiler- Toolchain wie auch auf der Zielplattform identisch sein müssen, damit das Programm die benötigten Laufzeitbibliotheken beim Bedarfsfall findet, oder aber statisch, wo bei der Übersetzung einer direkt ausführbaren Binärdatei alle Informationen mitgegeben werden, so dass eine solche Anwendung standalone und mit keinerlei Referenzen auf externe Bibliotheken auskommen kann. Ein auf dieser Weise statisch gelinktes

Programm kann je nach verwendeter Laufzeitbibliothek sehr viel Bytes beanspruchen, hat aber auch den Vorteil, dass es von der Umgebung der Laufzeitbibliothek der Zielplattform unabhängig ist, und dass es mit der Hilfe des GDB- Servers (GNU Debugger) auf dem Target, also dem Zielgerät, remote debuggt werden kann, sofern das Programm vorher mit zusätzlichen Debugginformationen übersetzt worden ist.

4.3 Die Buildrootumgebung

Eine Buidroot- Umgebung ist so etwas wie eine Ansammlung von Verzeichnissen, in denen Programmcodes mit ihren Makefiles zur Steuerung der Kompilierung untergebracht werden. Sie dient dazu, mit Hilfe des Crosskompilers, den Embedded-Linuxkernel mit seinen Kernelmodulen, aber auch eigene Projekte zu erstellen, die dann in ein Rootfilesystem verpackt werden, das danach direkt von der Zielplattform per NFS- Boot gemountet oder als Images in den NAND- Flash kopiert werden können. Das EMOD lässt sich mit beiden Varianten mounten, wo bei man vermerken muss, dass bei der Ersteren der integrierte Webserver nicht laufen kann, da das Netzwerk durch das NFS besetzt wäre.

4.3.1 Die Kernelkonfiguration für den Webcam- Treiber

Bevor der Kernel für den USB- Webcam- Treiber konfiguriert werden kann, muss die Buildroot- Umgebung noch im Entwicklungscomputer konfiguriert werden, so dass durch die Steuerung eines Haupt- Makefiles alle erforderlichen Unter- Makefiles gefunden und ausgeführt werden können. Dadurch wird gleichzeitig das Embedded–Linuxbetriebssystem zusammen mit dem Treibermodul erstellt. Für die Unterstützung des Webcam- Treibers, muss die Konfigurations- Datei `dnt3173_defconfig` alias `.config`, die für die Kompilierung des Linux- Kernels benötigt wird, konfiguriert werden. Hierzu müssen die Multimedia- Kernelmodule eingetragen werden, damit diese beim Kompilieren des Kernels auch mit erstellt werden, die von dem Treibermodul benötigt wird.
Dazu verfügt jeder Linuxkernel einer Standarddistribution über ein grafisches Konfigurationswerkzeug, das die Konfiguration der vielen Kernel- Optionen enorm vereinfacht.
Das Konfigurationswerkzeug kann mit dem Befehl „`make linux_config`“ im Unterverzeichnis „`PS3173`“, bzw. in dem Ordner wo sich das Hauptmakefile befindet, aufgerufen werden. Hierbei kann man wählen, ob die Kernelkomponente statisch oder beim Bedarf als einzelne Module eingebunden werden sollen.
Wurden Komponente als Module erzeugt, so kann man diese in die bestehende Laufzeitumgebung einfügen, ohne den kompletten Kernel austauschen zu müssen. So verhält sich auch der hier verwendete USB- Treiberkernelmodul, der zusammen mit den Mulltimediamodulen in das EMOD- System hin kopiert werden können, um die V4L- API zu benutzen.

4.4 Webcam Treibermodul

Der größte Nachteil von Linux ist wohl, dass es für viele Hardware keinen Treiber hat. Die meisten Hersteller von USB- Kameras bzw. Webcams bieten keine Linux-Treiber an. Auch scheuen sie sich davor, ihre Kommunikationsprotokolle für die von

ihnen auf dem Markt angebotenen Webcams der Öffentlichkeit zur Verfügung zu stellen, damit man wenigstens einen Treiber selber entwickeln kann. Dennoch konnte ich einen USB- Webcamtreiber für Linux auffinden, der nicht nur V4l- konform, sondern auch durch Einpassung für die ARM9- Zielmaschine cross übersetzbar ist.

4.4.1 GSPCA Treibermodul

Auf meinem langen Rechercheweg bin ich auf eine Seite gestoßen, in der der Betreiber auf Eigeninitiative gegen dieses Problem vorgegangen ist.
Herr Dr. Michel Xhaard von der französischen Universität „*Pierre et Marie Curie*“ hat einen kompakten Linux- Webcam- Treiber namens „*GSPCAV1*“ entwickelt, der, in der aktuellen Version, über 240 Webcams von unterschiedlichen Herstellern unterstützt. Eine Liste der unterstützten Webcams ist auf [Xhaard 2008] zu finden. Wie auch aus seiner Seite zu verlauten war, konnte er nicht auf die Unterstützung der Hardware- Hersteller greifen, und hat deshalb über Reverse- Engineering die USB-Kommunikationsprotokolle eines jeden einzelnen Webcams ausgelesen, analysiert und sie in seiner Treibersoftware integriert. So ist der Treiber natürlich nicht in der Lage, alle vom Hersteller eingebauten Funktionen zu unterstützen. Aber ohne technische Dokumentation oder die Kooperation seitens der Hersteller ist dies dennoch eine herausragende Leistung, die sowohl bei vielen Treiber- Nutzer als auch bei mir Anerkennung findet.

Wie bereits erwähnt, ist unter Linux jeder Gerätetreiber ein Teil des Kernels und kann entweder statisch hinzugelinkt oder im Bedarfsfall dynamisch als Kernel-Modul nachgeladen werden. Letzeres trifft auf dem GSPCAV1- Treiber zu. Er kann als Kernel- Modul kompiliert werden, auch nachdem der Kernel bereits übersetzt wurde. Um das Kernel- Modul aus der Treibersoftware zu erzeugen, muss es noch an die Buildroot- Umgebung angepasst werden. Hierzu muss das Makefile mit Angaben der benötigten Verzeichnisadressen, wie z. B. Kernelbuild- Ort mit Makefile des Kernels, und der zu verwendenden Crosskompiler gefüttert werden. Sind alle Einträge richtig eingetragen worden, wird ein Kernelmodul namens gspca.ko erzeugt, welches der eigentliche Webcamtreiber ist. Im Betrieb muss das Modul natürlich noch in den laufenden Kernel registriert bzw. eingebunden werden. Dies geschieht mit dem Befehl:

```
insmod /lib/modules/2.6.20.7/kernel/drivers/usb/media/gspca.ko
```

Aufgrund, dass das „`gspca.ko`“ Kernelmodul noch andere Multimedia-Kernelmodule, die in 4.3.1 erwähnt wurden, verwendet, müssen diese selbstverständlich auch in den Linux-Kernel zur Laufzeit eingefügt werden, sofern diese vorher nicht statisch in den Kernel implementiert worden sind:

```
insmod /lib/modules/2.6.20.7/kernel/drivers/media/video/v4l1-compat.ko

insmod /lib/modules/2.6.20.7/kernel/drivers/media/video/videodev.ko
```

5 Video 4 Linux API

Das Video4Linux- API ist eine tief in den Kernel integrierte Videoaufnahme- API für Linux, das speziell entwickelt wurde, um Multimedia- Devices auch für Linux

verfügbar zu machen [V4L] . Es unterstützt, sofern Treiber- Module vorhanden, USB-Webcams, TV- Karten und diverse andere Multimedia- Ausgabegeräte.
V4L steht aktuell auch in einer zweiten, teilweise abwärtskompatiblen Version zur Verfügung. Grund dafür ist, die erste Version wurde wegen dringenden Einsatznöten neuer Multimedia- Geräte für Linux hastig und mit einem unzureichenden Designentwurf erstellt. So wurde in der zweiten Version mehrere Designfehler korrigiert und das V4L2- API erblickte ab dem 2.5.x-Kernel das Licht der Welt. Der hier verwendete GSPCA- Treiber unterstützt nur die V4L1- API, aber wie vom Ersteller zu verlauten war, soll bereits ein neuer Treiber in der Entwicklung sein, der auch das V4L2- API unterstützen wird.
Nichtsdestotrotz ist der Treiber in der ersten Version und die V4L1- Programmierung für diese Arbeit vollkommen ausreichend.

5.1 Webcam initialisieren

Das Initialisieren dient dazu, die Webcam für das Auslesen des Imageframes vorzubereiten und wird in der Funktion
„`int device_init(struct global_video_struct *vd)`"
in der Datei „`motion_detector.c`" verwirklicht. Als Erstes muss man das Video-Device für den Zugriff öffnen. Da das Kernel- Treibermodul ein integraler Bestandteil des Kernels ist, wurde der direkte Zugriff durch einen laufenden Prozess aus Sicherheitsgründen unterbunden. Aber um dem Benutzer dennoch die Möglichkeit zu geben, um mit einem Gerätetreiber einer Festplatte, einer Soundkarte oder einer USB- Webcam zu kommunizieren, wurde in Unix ein Mechanismus eingeführt, mit dem ein gewöhnlicher Prozess lesend oder schreibend auf deren Hardware-Gerätedatei zugreifen kann. Dabei kann der Prozess über E/A- Funktionen auf den Gerätetreiber zugreifen, als handele es sich um eine gewöhnliche Datei.
Folgender Befehl liefert einen File- Deskriptor für den Webcam- Gerätedatei zurück, mit dem man über die `read()`- und `write()`- Funktionen auf das Gerät lesend oder schreibend zugreifen kann:

```
int fd = open("/dev/video",O_RDWR);
```

Selbstverständlich decken die `read()`- und `write()`- Funktionen bei Weitem nicht alle Funktionalitäten ab, die ein Treiber imstande ist zu leisten. Denn ein Gerät lässt sich auch in seiner spezifischen Art steuern. Dazu kann die Funktion `ioctl` (Input Output Control) verwendet werden, um z. B. die Video- Eigenschaft auszulesen:

```
struct video_capability   video_cap;
ioctl(fd, VIDIOCGCAP, &video_cap);
```

Danach stehen in dem Struct „`video_cap`" die Werte für Helligkeit, Kontrast, Farbintensität, Farbton, Auflösung oder Videomode, sofern sie von der Treibersoftware unterstützt werden.

```
struct video_capability   video_vid;
ioctl(fd, VIDIOCGCHAN, &video_vid);
```

Mit dem vorherigen Aufruf ist es möglich, zu testen, ob der Treiber einen Videokanal unterstützt. Ist es der Fall, dann können Informationen über den Kanal eines

Videogeräts fürs Setzen vorzubereitet werden, um dann wie folgt die eigene Videomode- und Videokanaleinstellung zu setzen:

```
video_vid.norm    = VIDEO_MODE_NTSC;”
video_vid.channel = 0;
ioctl(fd, VIDIOCSCHAN, & video_vid);
```

Möchte man die aktuelle Bildeinstellung, oder Herstellerangaben der Webcam in Erfahrung bringen, so lassen sie sich mit den unten stehenden Befehlen auslesen. Der Counterpart ist der Flagg „`VIDIOCSPICT“,` mit dem man benutzerspezifische Bildeinstellungen in den Treiber laden kann:

```
struct video_picture = video_pic;
ioctl(fd, VIDIOCGPICT, & video_pic);
```

Ferner kann man auch die Framegröße, die für das Videograbbing mittels `read()`-Methode benötigt wird, vorbereiten.

```
int x = 320, y = 240, w = 3;
struct video_window  video_win;
video_win.width  = x;
video_win.height = y;

ioctl(fd, VIDIOCSWIN, &video_win);
```

5.2 Frame Grabbing

Das V4L- API unterstützt zwei Arten, mit denen die Bilddaten gegrabbt, also in ein unsigned Character- Array für weitere Analyse kopiert werden können. Es hat sich bei der Entwicklung dieser Arbeit herausgestellt, dass die Variante des Memory-Mappings für die Bewegungsdetektion nicht geeignet ist. Bei dem Memory- Mapping-Verfahren legt man ein Speicher- Abbild für den Frame ($Breite * Höhe * Weite$) an, wo der Webcam- Treiber kontinuierlich oder auf Befehl neue Bilddaten hineinschreibt. Dabei wird das Speicher- Abbild byteweise aktualisiert bzw. neu überschrieben, so dass zwei zeitlich voneinander verschiedene Bildhälften innerhalb eines Bildes vorkommen können. Das ist zwar für Videostreaming wegen dem Geschwindigkeitsvorteil und stets gefüllte Pixelwerte gut geeignet, jedoch nicht wenn man für das Erkennen von Bewegungen zeitlich intakte Bilder braucht. Zwar gibt es die Möglichkeit, dass das Programm während der Auslesezeit angehalten wird, aber der hier verwendete Treiber scheint diese Funktion nicht zu unterstützen. So habe ich für das Grabbing von Frames die `read()`- Methode angewandt, da sie die Möglichkeit bietet, jeweils einen ganzen und zeitlich intakten Frame auszulesen. Nichtsdestotrotz werden nachfolgend beide Verfahren anhand von C-Codes erläutert, da sie Teil der Entwicklungsarbeit waren.

5.2.1 Memory Mapping

Voraussetzung für Memory- Mapping ist, dass der Treiber Doublebuffering unterstützt. Das heißt, während das Programm Daten aus dem Puffer ausliest, kann der Treiber schon neue Daten in den nächsten Puffer schreiben. Insofern ist es

notwendig, mit folgendem Code zu überprüfen, ob die Treiberunterstützung gegeben ist:

```
struct video_mbuf   video_mbuf;
if(ioctl(fd, VIDIOCGMBUF, &video_mbuf) < 0)
{
     perror("VIDIOCGMBUF");
}
```

War der Rückgabewert der Funktion `ioctl()` nicht -1, so kann man die Mapping-Funktion `mmap()` benutzen, die einen Pionter auf das Speicherabbild zurückliefert.

```
unsigned char *current_frame;
current_frame = mmap(0, video_mbuf.size,
                     PROT_READ|PROT_WRITE, MAP_SHARED, fd,
                0);
```

Um einen Auftrag für das Füllen des Speicherabbildes zu geben, kann die Funktion `ioctl` mit folgenden Argumenten aufgerufen werden:

```
struct video_mmap  video_mmap;
video_mmap.format = VIDEO_PALETTE_RGB24;
video_mmap.frame  = 0;
video_mmap.width  = x;
video_mmap.height = y;

ioctl(fd, VIDIOCMCAPTURE, &video_mmap);
```

Die Funktion `ioctl` ist nicht blockierend, so dass das Programm, ohne auf das Vollenden der Übertragung zu warten, weiter laufen würde. Dieses Verhalten könnte man unterbinden, sofern die obige Funktion mit dem Flagg `VIDIOCSYNC` aufgerufen wird:

```
ioctl(fd, VIDIOCSYNC, & video_mmap);
```

Hierzu wird das Programm solange angehalten, bis die Übertragung abgeschlossen ist, aber auch nur, wenn der Treiber das Memory Mapping unterstützt.

5.2.2 Read Methode

Die Read Methode ist die einfachere Variante, ein ganzes, von dem Webcam- Treiber dekomprimiertes Jpeg- Bild aus der Webcam auszulesen. Auch hier sollte man zuerst überprüfen, ob der Treiber das Auslesen aus der Gerätedatei unterstützt:

```
struct video_window   video_win;
if(ioctl(fd, VIDIOCGWIN, &video_win) == -1)
{
     perror("VIDIOCGWIN");
}
```

War die Prüfung erfolgreich, dann kann das Videoformat für das Auslesen eingestellt werden:

```
video_win.width  = x;
video_win.height = y;
ioctl(fd, VIDIOCSWIN, &video_win);
```

Anschließend wird bei jedem Aufruf der folgenden Funktion einen neuen Frame in den zuvor allokierten Bereich, auf den der Zeiger `current_frame` referenziert, kopiert.

```
int frame_size = x*y*w;
read(fd, current_frame, frame_size);
```

6 Jpeg- Library

Die Verwendung von JPEG- Format (Joint Photographic Experts Group) zum Speichern oder Manipulieren von Bilder, war insbesondere für diese Arbeit wichtig, da die Ressourcen eines Embedded- System doch sehr begrenzt sind. Auch handelt es sich hierbei um ein Wireless- System, bei dem der Kostenfaktor der zu übertragenden Datenmengen zu berücksichtigen sind. Das JPEG- Format erlaubt es, Bilddaten sehr effizient zu komprimieren und durch das Internet zu übertragen. JPEG ist aber eine verlustbehaftete Kompressionsmethode, bei der Daten aus dem ursprünglichen Bild, je nach Kompressionsstufe viel oder wenig, verloren gehen können. Andererseits ist sie sehr weit verbreitet als verlustfreie Kompressionsmethoden wie z. B. PNG (Portable Network Graphics), so das auch ältere Internetbrowser in der Lage sind, das JPEG- Bild im Browserfenster darzustellen. Die JPEG-Library [JPEG] in der verwendeten Version 6b ist eine freiverfügbare und in C geschriebene Bibliothek, die zur Verwendung in den Crosskompiler integriert werden muss. Dazu wurde die Makefile der JPEG_Lib mit dem Kompilerprogramm und Installationsorte für die Headerdateien und die zu erstellenden Objekte modifiziert. Mit dem Befehl „make“ würde die Library in den Crosskompiler integriert werden und ein Embedded-Projekt muss nur noch die Headerdatei „`jpeglib.h`“ einbinden, um die JPEG-Library zu benutzen.

6.1 JPEG- Bild erzeugen

Ein JPEG- Bild kann aus einem eindimensionalen vorzeichenlosen Charakter Array mit Pixelwerten für RGB (Red Green Blue) erzeugt werden, indem Funktionen aus der JPEG Library benutzt werden.

Das Bild lässt sich zum Speicher mit einem Dateizeiger beschreiben, der mit `fopen()` geöffnet werden kann:

```
FILE *outfile;
char *filename= "new_image.jpeg";
outfile =fopen(filename, "w");
```

Der Datentyp `jpeg_compress_struct` enthält alle Informationen, die für das Komprimieren benötigt werden.

```
struct jpeg_compress_struct cinfo;
```

Der JPEG- Error- Handler wird von `jpeg_error_mgr` dargestellt. Dieser ist notwendig, um bei der Kompression auftretende Fehler abzufangen und auszuwerten:

```
struct jpeg_error_mgr jerr;
```

Als erster Schritt muss das JPEG- Kompressionsobjekt allokiert und initialisiert werden:
Dabei werden der zuvor definierte JPEG- Error- Handler sowie das JPEG- Kompressionsobjekt initialisiert:

```
cinfo.err = jpeg_std_error(&jerr);
jpeg_create_compress(&cinfo);
```

Dann müssen noch Daten und Speicherziele spezifiziert werden, wobei `linesize` sich aus $Zeile * Weite$ errechnen lässt.

```
unsigned int linesize = x * w;
jpeg_stdio_dest(&cinfo, outfile);
```

Nun werden Parameter, die für die Kompression benötigt werden, gesetzt:

```
cinfo.image_width  = x;
cinfo.image_height = y;
cinfo.input_components = w;
```

Dann kann das Farbprofil der Bilddaten kann anhand der Tiefe bestimmt werden:

```
if(w == 1)
    cinfo.in_color_space = JCS_GRAYSCALE;
else
    cinfo.in_color_space = JCS_RGB;
```

Dazu muss man aber die Default Kompression wählen, die vom Farbprofil abhängt und automatisch gesetzt werden können:

```
jpeg_set_defaults(&cinfo);
```

Mit der Funktion `jpeg_set_quality`() ist es möglich, eine Qualitätsstufe von 1 bis 100 zu wählen. Ein guter Wert, bei dem die subjektive Qualitätempfindung des Bildes nicht in all zu großer Mitleidenschaft gezogen wird, liegt bei ca. 30:

```
jpeg_set_quality(&cinfo, 30, TRUE);
```

Eine weitere Funktion namens `jpeg_start_compress()` initialisiert den Startzustand für die Kompression, wobei in der nachfolgenden For- Schleife die Bilddaten zeilenweise in den Speicher geschrieben werden:

```
jpeg_start_compress(&cinfo, TRUE);
```

```
line=img;
unsigned int i;
for (i = 1; i <= y; i++) {
   jpeg_write_scanlines(&cinfo, &line, 1);
   line=img + (linesize * i);
}
```

Sind alle Zeilen ausgelesen und in den Speicher gefüllt worden, dann wird der Kompressionszustand für beendet erklärt und die in dem `jpeg_compress_struct- Field` allokierten Objekte können wieder freigegeben werden:

```
jpeg_finish_compress(&(cinfo));
jpeg_destroy_compress(&(cinfo));
```

Zum Schluss wird noch der Filedeskriptor mit `fclose`() geschlossen und das neue JPEG- Bild liegt fertig in dem zuvor benannten Speicherort.

```
fclose(fp);
```

6.2 JPEG- Bild dekodieren

In dieser Arbeit war es auch notwendig, ein JPEG- formatiertes ROI- Bild (Region of Interest) zu dekodieren, das zuvor für die Bewegungsdetektion manuell bearbeitet wurde. Nach der Dekodierung stehen die Rohdaten des Bildes zur Verfügung, aus denen relevante Informationen gewonnen werden können. Aber dazu später mehr.

Auch hier muss man sich, wie beim Erstellen eines JPEG- Bildes, erst einmal einen Filedescriptor besorgen, der auf eine Bilddatei verweist. Nur, dass die Datei durch die Option „`rb`“ nur gelesen und nicht beschrieben wird:

```
char *filename= “image.jpeg”;
FILE *infile;
infile = fopen(filename, "rb");
```

Zum Dekodieren wird das Gegenstück des Datentyps `jpeg_compress_struct` benutzt:

```
struct jpeg_decompress_struct cinfo;
```

Ein JPEG- Error- Handler wird auch hier gebraucht.:

```
struct error_mgr jerr;
```

Der Puffer für eine Bildzeile sieht wie folgt aus:

```
JSAMPARRAY buffer;
```

Hier werden das JPEG- Dekompressionsobjekt sowie der JPEG- Error- Handler initialisiert:

```
cinfo.err = jpeg_std_error(&jerr.pub);
jpeg_create_decompress(&cinfo);
```

Der Anfangszustand des STDIO- Streams wird initialisiert, sofern „`infile`“ vorher geöffnet worden ist:

```
jpeg_stdio_src(&cinfo, infile);
```

Diese Funktion liest bis zum Auftreten des ersten SOS (Start Of Stream) Markers und sichert alle ausgelesenen Informationen aus dem JPEG- Header in das Dekompressionsobjekt:

```
jpeg_read_header(&cinfo, TRUE);
```

Nun kann der Startzustand für die Dekompression initialisiert werden:

```
jpeg_start_decompress(&cinfo);
```

In folgenden Feldern stehen die gesicherten Header- Informationen, wie Breite, Höhe und Tiefe des eingelesenen Bildes:

```
int x = cinfo.output_width;
int y = cinfo.output_height;
int w = cinfo.output_components;
```

Mit diesen Informationen kann der Speicher für das Frame dynamisch allokiert werden:

```
int frame_size = x*y*w;
int row_stride = x * w;

unsigned char *pic =
                malloc(frame_size * sizeof(unsigned char));
```

Auch der Puffer für eine Bildzeile muss noch vorbereitet werden:

```
buffer = (*cinfo.mem->alloc_sarray)
          ((j_common_ptr) &cinfo, JPOOL_IMAGE, row_stride, 1);
```

Anschließend können die Daten zeilenweise in den zuvor allokierten Speicher hineinkopiert werden:

```
int i, zahl=0;
while (cinfo.output_scanline < cinfo.output_height)
{
     jpeg_read_scanlines(&cinfo, buffer, 1);
     for(i= 0; i< row_stride; i++){
          pic[zahl++]= buffer[0][i];
     }
```

```
}
```

Nach Beendigung des Kopierverfahrens kann der Dekompressionszustand für beendet erklärt und alle allokierten Objekte wieder freigegeben werden:

```
jpeg_finish_decompress(&cinfo);
jpeg_destroy_decompress(&cinfo);
fclose(infile);
```

7 Bewegungsdetektion

Die Bewegungsdetektion hat das Ziel, aus einer Szene die dynamischen Aspekte, also die Bewegung von Objekten zu ermitteln. Für das Erkennen von Bewegungen reicht selbstverständlich nur ein einzelnes Bild nicht aus, sondern man benötigt eine Folge von mindestens zwei Bildern, aus denen Bewegung extrahiert werden können.
Auch ist bei der Erkennung im Vorfeld zu klären, ob man nur die Bewegung, als ausreichendes Ereignis identifizieren will, oder ob man auch Informationen über die Art und Richtung der Bewegungen haben möchte. Diese Informationen erfordern aber auch eine rechenintensivere Analyse von der Bildsequenz, die ohne passende Hardware kaum zu bewältigen wäre.
Diese Arbeit jedoch basiert auf das Erkennen von Bewegungen durch ein eingebettetes System, das aufgrund der Knappheit an Systemressourcen keine aufwendige Berechnungen erlaubt, zumal die Berechnungen der Fließkommazahlen auf dem Mikrokontroller nur in der Software realisiert werden können. Dennoch werden nachfolgend die gängigen Methoden zur Bewegungsdetektion vorgestellt und erläutert, wobei einzelne Ideen doch aufgegriffen wurden.
Die Verfahren lassen sich hauptsächlich in drei Kategorien einteilen:

- Differenz- und differenzielle Verfahren
- Zuordnungsverfahren
- Filterverfahren

7.1 Differenz- und differenzielle Verfahren

Der Ansatz bei den Differenz- und differenziellen Verfahren ist der, dass für die Bewegungsdetektion die Veränderungen bzw. Verschiebungen von Grauwerten der Pixelpunkte herangezogen werden [Jähne 2002]. Der Grundgedanke ist, dass sich die Grauwerte eines sich bewegenden Objektes nicht willkürlich verändern, womit die Bedingung erfüllt ist, sie funktional zu beschreiben. Hierzu wird aus den Grauwerten den Differenzwert eines jeden Pixelpunktes errechnet und eine zweidimensionale Matrix erzeugt, aus der eine dynamische Änderung bzw. Bewegung eines Bildpaares abzuleiten ist.

Sei $D(x, y, t_i)$ eine Differenzmatrix zum Zeitpunkt t_i und $g(x, y, t_i)$ eine Funktion, die den Grauwert eines Pixelpunktes an der Stelle $(x, y)^T$ zum Zeitpunkt t_i beschreibt, so lässt sich die Differenzmatrix wie folgt darstellen:

$$D(x, y, t_i) = |g(x, y, t_{i-1}) - g(x, y, t_i)| \tag{7.1}$$

Naturgemäß können bei der obigen Matrixbildung auch unerwünschte Einflüsse, wie z. B. Lichtveränderungen, Schattenwürfe, aber auch Bildrauchen, die durch ein Übersprechen von benachbarten Halbleitersensoren der CCD- Webcams, entstehen, die für die Auswertung nachteilig wirken. Diese Einflüsse lassen sich jedoch durch Einbeziehung eines Schwellenwertes bei der Differenzbilderstellung weitgehend herausfiltern:

$$D(x, y, t_i) = \begin{cases} 0: |g(x, y, t_{i-1}) - g(x, y, t_i)| < thres \\ 1: others \end{cases} \quad (7.2)$$

Um das Bildrauschen oder andere unerwünschte Umwelteinflüsse, wie das Flattern der Blätter an den Bäumen oder kurzzeitige Lichtveränderungen tiefer zu bereinigen, können weitere spezielle Filter benutzt werden, die später im Kapitel Realisierung vorgestellt werden. Dennoch hat dieses einfache und effiziente Differenzverfahren einen gewichtigen Nachteil. Zwar kann eine Bewegung im Bildinhalt identifiziert, aber nicht ihre Richtung bestimmt werden. Hieraus ergibt sich das Problem, dass in dem Differenzbild von Null verschiedene Werte auftreten können, wenn ein Objekt gerade ein Bereich verlässt, als auch wo dieses Objekt einen anderen Bereich neu verdeckt (Abbildung 7.1).

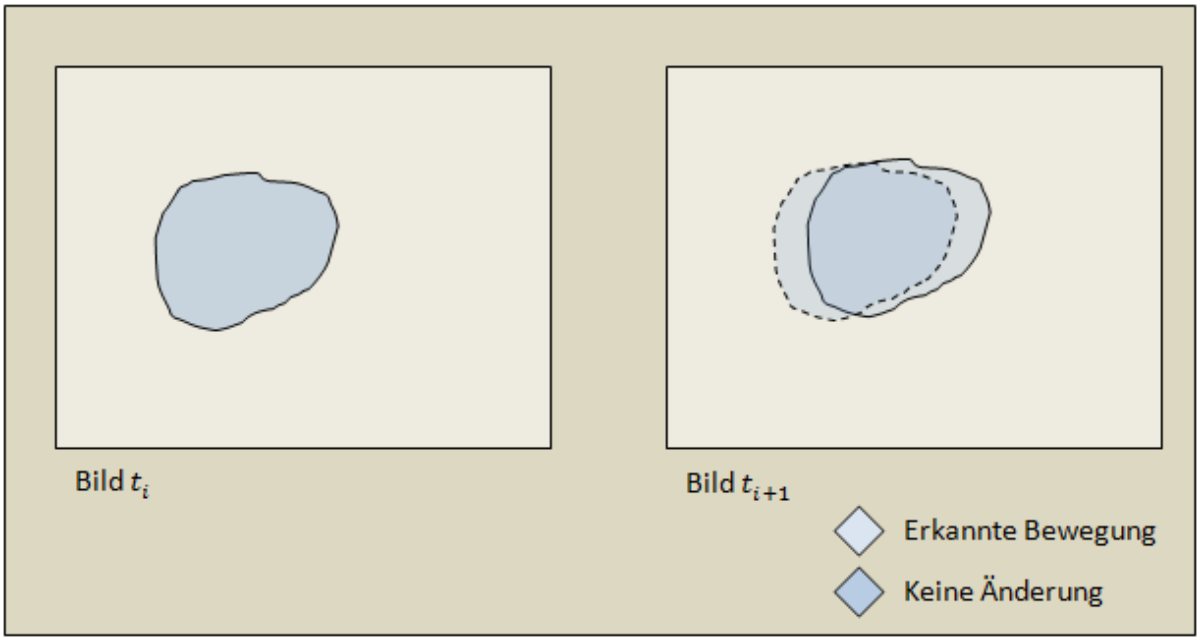

Abbildung 7.1: Problemdarstellung (Bereich verlassen und eingetreten)

Wie bei der Hydrodynamik, wo die Strömung von Fluiden analysiert wird, so lässt sich auch der optische Fluss oder Grauwertfluss einer Bildsequenz untersuchen. Bei den differenziellen Verfahren spielen die sich bewegenden Grauwertmuster aus einer Bildsequenz eine wesentliche Rolle. Sie werden als eine Menge von Bildpunkte betrachtet, die über die Bildfläche strömen. In Erwartung, dass sich die Grauwerte eines Bildobjektes während der Zeit hin nicht verändern, und dass die Beleuchtung ebenfalls nah zu konstant bleibt, lassen sie sich ebenfalls funktional in Abhängigkeit von den Ortskoordinaten des Flusses bestimmen:

$$\mathrm{g}(\mathrm{x}, \mathrm{y}, \mathrm{t}) = \mathrm{g}(\mathrm{x} + d\mathrm{x}, \mathrm{y} + d\mathrm{y}, \mathrm{t} + d\mathrm{t}) \quad (7.3)$$

Eine differenzierbare Funktion $g(x, y, t)$ gibt die Grauwerte eines Bildpunktes an der Stelle $(x, y)^T$ zum jeweiligen Zeitpunkt t an.

Wird ein Bildpunkt in der Bildfläche in einem Intervall dt um $(dx, dy)^T$ verschoben, so gilt die vorstehende Funktion in (7.3). Nun kann mit dem Ansatz der Taylorreihenentwicklung der rechte Teil der Gleichung bestimmt werden:

$$g(x + dx, y + dy, t + dt) = g(x, y, t) + \frac{\partial g}{\partial x} dx + \frac{\partial g}{\partial y} dy + \frac{\partial g}{\partial t} dt + \varepsilon(dx, dy, dt) \tag{7.4}$$

Um die Komplexität der Funktion, und damit den Rechenaufwand stark zu reduzieren, genügt es auch nur den linearen Teil der Funktion, der den Grauwertverlauf beschreibt, zu untersuchen. Hierzu fallen die Glieder der höheren Grade $\varepsilon(dx, dy, dt)$ weg:

$$g(x, y, t) = g(x, y, t) + \frac{\partial g}{\partial x} dx + \frac{\partial g}{\partial y} dy + \frac{\partial g}{\partial t} dt \tag{7.5}$$

Die obige Funktion kann durch Subtraktion von $g(x, y, t)$ und Division durch dt weiter reduziert werden.

$$0 = \frac{\partial g}{\partial x} * \frac{dx}{dt} + \frac{\partial g}{\partial y} * \frac{dy}{dt} + \frac{\partial g}{\partial t} \tag{7.6}$$

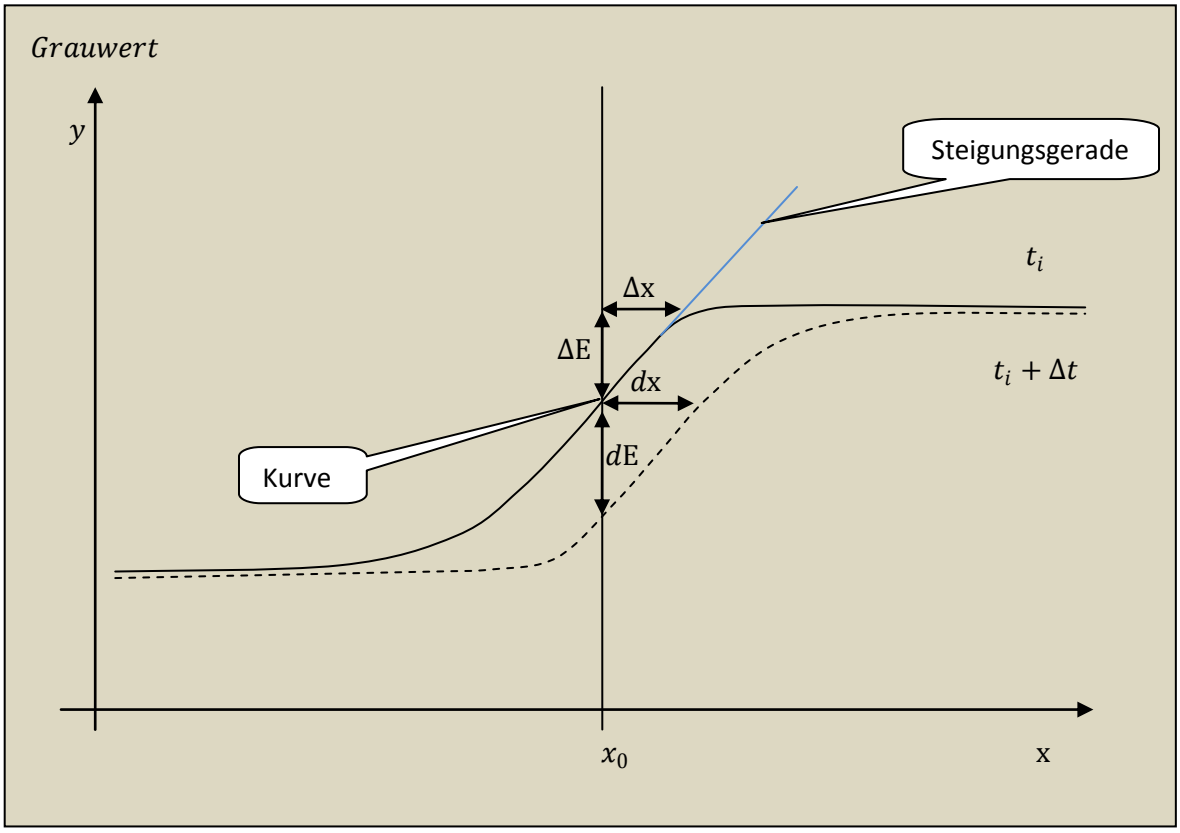

Abbildung 7.2: Differenzielle Methode zur Bewegungsbestimmung im eindimensionalem Fall

Ersetzt man die partiellen Ableitungen durch ihren Funktionsnamen wie z.B. $\frac{dx}{dt} = u$ und $\frac{dy}{dt} = v$, so erhält man eine Geradengleichung $0 = g_x * u + g_y * v + g_t$, die auch als OFC (Optical Flow Constraint) genannt wird, und welche die Menge aller aufgenommenen Grauwertveränderungen $(u, v)^T$ auf eine Gerade im u,-v-Raum einschränkt. Um die differenzielle Methode zur Detektion der Bewegung besser zu veranschaulichen, wird sie im eindimensionalen Fall und mit Hilfe der Abbildung 7.2 erläutert.

$$0 = g_x u + g_t \tag{7.7}$$

Die Funktion (7.7) ist die Einschränkungsfunktion aus (7.6) in eindimensionalem Fall. Die Abbildung 7.2 zeigt z. B. den Verlauf der Grauwertkante, also die Grauwertänderung an der Grenze nebeneinander liegender oder benachbarter Bereiche, über die x- Koordinate an. Dabei beschreibt die obere Linie den Verlauf der Grauwertkante zum Zeitpunkt t_i und die untere Linie den Verlauf nach der Verschiebung um d_x zum Zeitpunkt $t_i + \Delta t$. Um die Steigung des Grauwertgardienten aus dem Steigungsdreieck an der Stelle $x_0$zum Zeitpunkt t_i zu bestimmen, kann die nachfolgende Funktion verwendet werden:

$$\lim_{\Delta x \to 0} \frac{\Delta E}{\Delta x} = E_x \tag{7.8}$$

Wird die Grauwertkante um Δx nach rechts verschoben, so erhält man an der Stelle x_0 die Grauwertänderung dE.

$$dE = -E_x dx \tag{7.9}$$

Durch Umstellung der obigen Gleichung nach dx, kann die Verschiebung dx in x-Koordinate bestimmt werden, nachdem die Grauwertänderung dE und die Steigung des Gradienten an der Stelle x_0 berechnet worden sind.

$$dx = -\frac{dE}{E_x} \tag{7.10}$$

Leider kann die Einschränkungsgleichung bzw. OFC nicht für den zweidimensionalen Fall angewandt werden, da sie für eine Lösung noch unterbestimmt ist. Denn, egal ob sich nur die Kamera oder das Objekt bewegt, es werden stets an einigen Stellen der Szene die Grauwerte wegen der Beleuchtung vom Vorbild unterscheiden. Es müssen noch weitere Bedingungen bzw. Regularisierer, wie z. B. eine Glattheitsbedingung auch Smoothing Constraint genannt, eingeführt werden. D. h. die Änderungen in dem optischen Fluss angrenzender Bildpunkte sollen möglichst glatt sein.

$$V_H(\Delta g, \Delta u, \Delta v) := |\Delta u|^2 - |\Delta v|^2 \tag{7.11}$$

Nun, um sowohl das OFC als auch den Regularisierer (7.11) zu minimieren, können beide zu einem Funktional zusammengefasst werden:

$$E(u, v) := \int_\Omega (g_x u + g_y u + g_t)^2 + \alpha(|\Delta u|^2 - |\Delta v|^2) dx dy \tag{7.12}$$

Wird das obige Funktional weiter minimiert, so führt es zu folgende Form, wobei α lediglich als Gewichtsfaktor dient, der abhängig von jeweiliger Situation zwischen 1 bis 5 gewählt werden kann:

$$E(u,v) \coloneqq \int_{\Omega} G(x,y,u,v,\Delta u,\Delta v)\, dxdy \tag{7.13}$$

Unter Berücksichtigung folgender Euler- Lagrang- Gleichungen kann die Funktion (7.13) zu einer Lösung führen:

$$\partial_x G_{ux} + \partial_y G_{uy} - G_u = 0 \tag{7.14}$$

$$\partial_x G_{vx} + \partial_y G_{vy} - G_v = 0 \tag{7.15}$$

Letztendlich führen die Berechnung der Euler-Lagrange-Gleichungen für das Funktional mit dem Regularisierer auf folgendes partielle Differentialgleichungen:

$$\Delta u - \frac{1}{\alpha} g_x \left(g_x u + g_y v + g_t\right) = 0 \tag{7.15}$$

$$\Delta v - \frac{1}{\alpha} g_y \left(g_x u + g_y v + g_t\right) = 0 \tag{7.16}$$

Leider gleitet dieser Regulierer in jede Richtung des Bildes, so dass auch die Diskontinuitäten, wie z.B. die Bewegungsgrenzen im Optischen Fluss, verwischt werden, die ja eigentlich als solche erkannt werden sollen. Aber dieses Problem kann dank dem bildbasierten Regularisierer behoben werden, der in dieser Arbeit nicht mehr vorgestellt, sondern nur erwähnt wird.

7.2 Zuordnungsverfahren

Die Idee bei diesen Verfahren lautet, charakteristische Merkmale eines Bildes in einem anderen Bild zu suchen und zuzuordnen. Neben dem merkmalsorientierten Zuordnungsverfahren gibt es noch die blockorientierten Zuordnungsverfahren.
Bei den merkmalsorientierten Zuordnungsverfahren können Merkmale, wie z. B. die Konturen eines Objektes extrahiert und diese in den nachfolgenden Bildern gesucht und analysiert werden.
Die korrespondierenden Objekte in den Bildern können mit Einbeziehung des Verlaufs der Orte nach der Zeit in sogenannte Trajektorien zusammengefasst werden, die Auskunft über die Richtung und Geschwindigkeit des sich bewegenden Objektes geben.
Die blockorientierten Zuordnungsverfahren bedienen sich der Suche nach identischen Verschiebungen in den Bildbereichen, um eine anfängliche Extraktion eines ausgezeichneten Objektes zu vermeiden, die viel Rechenleistung in Anspruch nehmen würde. Daneben gibt es bei der Merkmalszuordnung noch eine ganze Reihe von durchdachten und wiederverwendbaren Methoden, die den Focus weiter einschränken und die Rechenzeit verkürzen können:

- **nearest match:** Ein Treffer kann erzielt werden, nur wenn die kürzeste Distanz zwischen der Ursprungsposition und der neuen

Position, die das Objekt eingenommen und neu einnimmt, berücksichtigt wird.

- **reliability:** Sind mehrere mögliche Korrespondenzen für eine Analyse vorhanden, so wird die Korrespondenz mit den meisten Übereinstimmungen ausgewählt.

- **maxi. velocity:** Man nimmt an, dass ein zutreffendes Objekt eine bestimmte Geschwindigkeit nicht überschreiten darf, was zur Folge hat, dass es im nachfolgenden Bild nicht weit von der aktuellen Position entfernt sein kann.

- **small velocity change:** Es wird vorausgesetzt, dass sich die Geschwindigkeit und Richtung eines sich bewegenden Objektes nicht schlagartig verändern wird.

- **block matching:** Die originären Grauwerte eines Objektes in einer lokalen Umgebung werden in eine zweidimensionale Mustermatrix extrahiert und anschließend pixelweise über eine größere Suchmatrix des zweiten Bildes geschoben.
  Mit Hilfe vorher definierenden Übereinstimmungswerte und durch die Erfüllung der Ähnlichkeitsbedingungen kann die Position des Objektes im aktuellen Bild ermittelt werden, die auch Auskunft über Richtung und Geschwindigkeit geben können.

Mit Hilfe der Methoden aus den Zuordungsverfahren ist es möglich, bereits im Vorfeld einer langen und rechenintensiven Analysekette die Entscheidung zu treffen, ob ein Objekt für weitere Untersuchungen verworfen oder durchgelassen werden soll. Zudem haben diese Verfahren auch den Vorteil, dass sie gegenüber Farbschwankungen relativ resistent sind, da zur Identifikation der Objekte ihre Konturen mit Hilfe der Grauwertgradientenbildung bestimmt und analysiert werden können.

7.2 Filterverfahren

Bei den meisten Verfahren zur Objektdetektion werden die Bilder hauptsächlich nur paarweise analysiert und ausgewertet. Diese Einschränkung liegt in der Tatsache begründet, dass die Verarbeitung und Speicherung von Bildern sehr aufwendig und nicht besonders ressourcenschonend ist.
Aber mit zunehmender Rechenleistung der Computersysteme wurde auch vermehrt darauf übergegangen, die Bildfolgenauswertung auch auf Bildsequenzen durchzuführen. Dabei bedient man sich an Filterverfahren zur Bewegungsdetektion, welche die Bilder einer Sequenz in einen dreidimensionalen Raum hintereinander legt und auswertet [Haussecker 1993]. Ein solcher Raum besteht aus zwei Ortskoordinaten und einer Zeitkoordinate.
In Abbildung 7.3 ist der beschriebene Bildfolgeraum dargestellt.
Der schiefe Balken deutet auf die Bewegung hin. Aus der Schräge des Balkens kann die Geschwindigkeit des Objektes abgeleitet werden. Dabei bedeutet es, je größer die Neigung desto größer ist auch die Geschwindigkeit eines Objektes. Hieraus lässt sich

ein Geschwindigkeitsfilter entwickeln, der die Objekte, die sich mit einer bestimmten Geschwindigkeit bewegen, extrahiert bzw. selektiert.
Es existieren viele Verfahren, um einen solchen Filter zu realisieren. Einer der vielen Ansätze ist der Projektionsfilter, bei dem die Bilder mit Hilfe eines Faltungsoperators aufeinander aufaddiert werden.

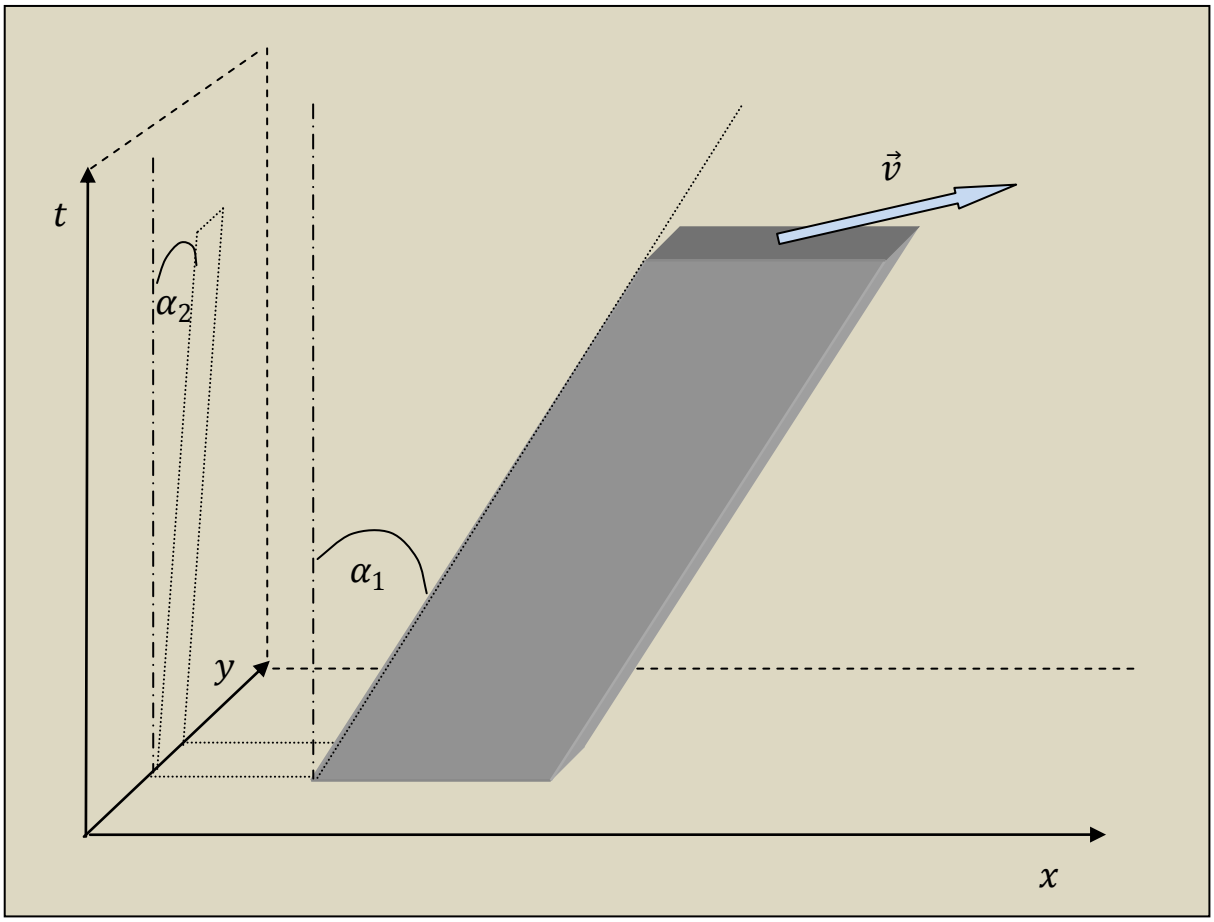

Abbildung 7.3: Bewegung eines Objektes in einem Orts- Zeit- Bild

Aus der Neigung des Orts- Zeit- Objektes gegen die Zeitachse kann die Geschwindigkeit des Objektes direkt berechnet werden:

$$\vec{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = -\begin{pmatrix} tan\alpha_1 \\ tan\alpha_2 \end{pmatrix} \tag{7.17}$$

Die Stapelung der Bildsequenz zu einer 3D-Struktur ermöglicht es auch, dass die Eigenschaften von Objekten im zugehörigen 3D-Fourierraum untersucht werden können.
Dabei lässt sich ein im Ortsraum mit der Geschwindigkeit $\vec{v}$ bewegendes Objekt in den $\vec{k}, \omega$ -Raum darstellen, indem die Fouriertransformierte des Objektes berechnet wird.
Die Gleichung (7.18) beschreibt den Grauwertverlauf im Bildfolgenraum und die Gleichung (7.19) gibt die Fouriertransformierte wieder.

$$g(\vec{x}, t) = g(\vec{x} - \vec{v}t)$$

(7.18)

$$\hat{g}(\vec{k},\omega) = \iint g(\vec{x} - \vec{v} * t)e^{-i(\vec{x}\vec{k}+\omega t)}\, d^2x dt = \hat{g}(\vec{k}) \int e^{-it\vec{v}\vec{k}} e^{-i\omega t}\, dt$$
$$= \hat{g}(\vec{k})\delta(\vec{k} * \vec{v} + \omega)$$

(7.19)

In der Gleichung der Fouriertransformierte liegen alle Objekte, die sich mit der Geschwindigkeit $\vec{v}$ bewegen, auf der δ- Ebene, die durch $\omega = \vec{k} * \vec{v}$ gebildet wird. Die Ebene steht im Fourierraum senkrecht auf dem Objektbalken, wobei aus den Winkeln $\alpha_1$und α_2 zu der k_1- bzw. k_2-Achse, die Geschwindigkeit direkt ermittelt werden kann.

Um alle sich bewegende Objekte aus der homogene Bildszene herauszufiltern, setzt man bei (7.19) für die Geschwindigkeit $\vec{v} = 0$ ein, und man erhält folgende Gleichung:

$$\hat{g}(\vec{k},\omega) = \hat{g}(\vec{k})\delta(\omega)$$

(7.20)

In der k_1- k_2- Ebene im Fourierraum wird dann $\omega = 0$. D. h. multipliziert man die 3D-Fouriertransformierte einer Bildfolge mit dieser Ebene, so wird nach der Rücktransformation eine Bildfolge erzeugt, in der nur noch die ruhenden Strukturen zu sehen sind und alle sich bewegenden Objekte sind verschwunden.

8 Realisierung

Dieses Kapitel befasst sich mit den Lösungsansätzen der jeweiligen Problemstellungen, und gibt einen Überblick darüber, wie die Anwendung der Bewegungsdetektion anzuwenden ist. Es wird aber auch auf die Umsetzung der implementierten Lösungen eingegangen.

Die Anwendung zur Bewegungsdetektion wird auf einem eingebetteten Wireless-Router ausgeführt, der selber keine optische Mittel zur Betrachtung der aufgenommen Bilder bietet. Auch bedarf es diverser individuellen Einstellungen, um die Anwendung und die Webcam auf die jeweilige Situation anzupassen.
Gleichwohl sollte die Bewegungsdetektion auch leicht zu bedienen sein, und möglichst andere Anwendungen, die den Routerbetrieb aufrechterhalten, nicht unnötig behindern, gar sogar zum Absturz bringen.
Um die o.g. Anforderungen zu genügen, wurde die Idee einer auf HTML basierende Webparametrierung und Bedienung entwickelt.
Die Webparametrierung ermöglicht es, die Anwendungen der Bewegungsdetektion und die des Routers parallel auszuführen, und gewährt somit auch die Mobilität, für die der Wireless- Router ausgezeichnet ist.
Ferner kann man die Wireless-Router- Webcam- Kombination auch dazu verwenden, um einzelne Bilder, bei Bedarf, zu schießen und zum Clientrechner übers Internet zu übertragen.
Die Bewegungsdetektion ist auch in der Lage bei Erkennung einer Bewegung eine SMS- Nachricht an eine voreingestellte Telefonnummer zu senden, in der Auskunft darüber steht, wann ein sich bewegendes Objekt entdeckt wurde und wie viele Bilder bereits gespeichert worden sind.

8.1 Aufbau der Anwendung

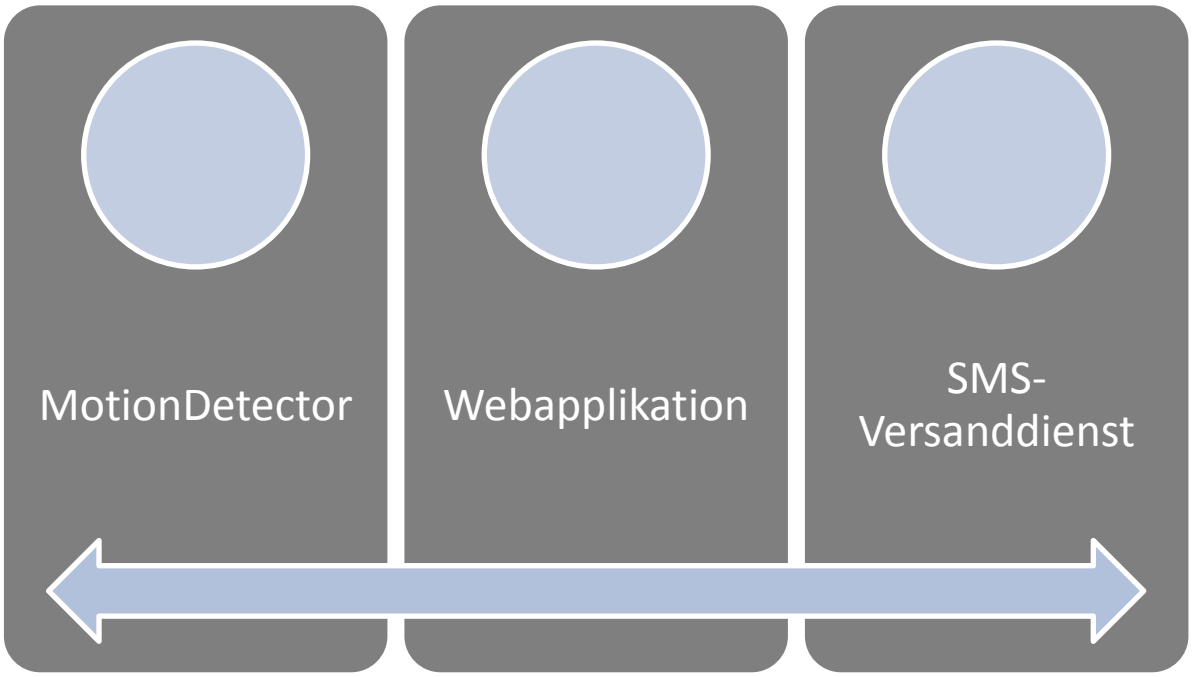

Abbildung 8.1: Aufbau der gesamten Anwendung

Die Konstruktion Motion- Detection besteht aus drei Säulen, die miteinander interagieren, um das Ziel der Bewegungsdetektion zu verwirklichen, und um dem Benutzer eine auf Webclient- und Server basierende Bedienungsoberfläche anzubieten.

Die erste Säule ist die eigentliche Anwendung MotionDetector, die wie die meisten Implementierungen dieser Arbeit in C geschrieben wurde. Grund für den Verzicht einer objektorientierten Programmiersprache war der, dass ein in C geschriebenes Programm gegenüber einer C++ Implementierung weniger Platz benötigt, sofern unter C++ stark objektorientiert programmiert wird. Um bereits im Vorfeld platzsparend und effizient zu programmieren, auch unter Berücksichtigung, dass diese Anwendung in einem noch in Weiterentwicklung befindlichem Umfeld migriert sein wird, wurde die Programmiersprache C gegenüber C++ vorgezogen. Es ist auch von Vorteil, dass die C- Standardbibliotheken bereits in den fertigen Embedded-Linux- Kernel integriert sind, so dass in der Laufzeitumgebung diejenigen dynamisch kompilierten C- Anwendungen auf diese direkt zugreifen können.

Um auf die Gesamtanwendung zurückzukommen, stellt die nächste Säule die Webapplikation dar, mit dessen Hilfe man auf die MotionDetector- Anwendung zugreifen kann. Dieses Userinterface besteht aus den HTML- Dateien, mit eingebauter Ajax-JavaScript- Techniken, die für die Generierung dynamischer Webinhalte mit asynchroner Datenübertragung zuständig sind, und dadurch zur Reduzierung des teuren Datenverkehrs beitragen. Hierzu kommuniziert der Benutzer mit Verwendung eines neueren Java- Ajax- und JavaScriptfähigen Webbrowsers, z.B. Firefox oder Internet Explorer, mit dem bereits auf dem Router vorhandenen Webserver.
Dieser Server leitet die Anfragen des Browsers an ein CGI- Programm (Common Gateway Interface), das auch Teil dieser Arbeit ist, weiter. Das CGI- Programm ist in

der Lage, die von der HTML- Seite ankommenden Parameter oder Informationen im Empfang zunehmen, sie zu verarbeiten, auszuwerten und gegebenenfalls an den MotionDetector zu übergeben.

Der Benutzer kann über den Browser ein JPEG- formatiertes ROI- Bild von der Anwendung schießen lassen, um es dann lokal im Browser zu bearbeiten. Dabei wird ein Java- Applet gestartet, das in der Lage ist, das ROI- Bild vom Server herunterzuladen.
Das Applet dekodiert das JPEG- Bild und stellt es grafisch dar.
Mit den entwickelten Malwerkzeugen ist es für den Benutzer auf einfache Wege möglich, und zwar ohne Hinzuziehen fremder Hilfsprogramme, Bereiche in dem ROI- Bild mit einer schwarzen Farbe zu übermalen, so dass diese Bereiche für die Bewegungsdetektion ignoriert werden. Dabei spielt es keine Rolle, wo in dem Bild und in welcher Form mit dem Schwarzstift hantiert wird, denn durch die effiziente Implementierung können diese Bereiche im Auswertungsalgorithmus übersprungen werden. Zusätzlich kann man auch mit anderen Farben die Grenzen für die mini- und maximale Größe eines zu entdeckenden Objektes durch Zeichnen in das ROI- Bild definieren.
Nach der Bearbeitung kann das ROI- Bild im Applet per Knopfdruck JPEG- kodiert zum Server übertragen werden. Nun ist der MotionDetector in der Lage, die nützlichen Informationen aus dem ROI- Bild zu extrahieren, um benutzerangepasste Bewegungsdetektion durchzuführen. Zuvor kann man noch Einstellungen für die Helligkeit, Auflösung, Zeitstempel, Sensibilität, Startverzögerung etc. einstellen.

Wurde vorher gar die SMS- Funktion gewählt, welche auch als die letzte Säule darstellt, so wird der Benutzer bei erfolgreicher Detektion durch eine automatische SMS- Nachricht an sein Handy benachrichtigt, sofern man auch seine Handynummer korrekt eingegeben hat.

8.2 Motion Detector

Wie in der Abbildung 8.2 zu erkennen ist, besteht der MotionDetector aus vielen Verarbeitungsschritten, die jeweils bestimmte Aufgaben zu erfüllen haben, und auf die nachfolgend eingegangen werden.

Das zentrale Datenelement `global_video_data` ist vom Typ `struct`, indem alle erforderlichen Felder und Referenzen ihren Platz finden, und welches an die meisten implementierten Funktionen mit ihren spezifischen Aufgaben als Argument übergeben wird. Nebenbei wurde bei der Entwicklung Wert darauf gelegt, dass alle speicherintensive Felder erst beim Bedarf dynamisch allokiert und sofort wieder freigegeben werden, wenn sie nicht mehr benötigt werden. Auch sollten die Bilder erst an eine aufwendigere Analyse übergeben werden, wenn durch die Vorfilterung bestimmt wurde, dass sich eine weitere Untersuchung auszahlt. Auf diese Weise können die beschränkten Ressourcen geschont und die Effizienz gesteigert werden.

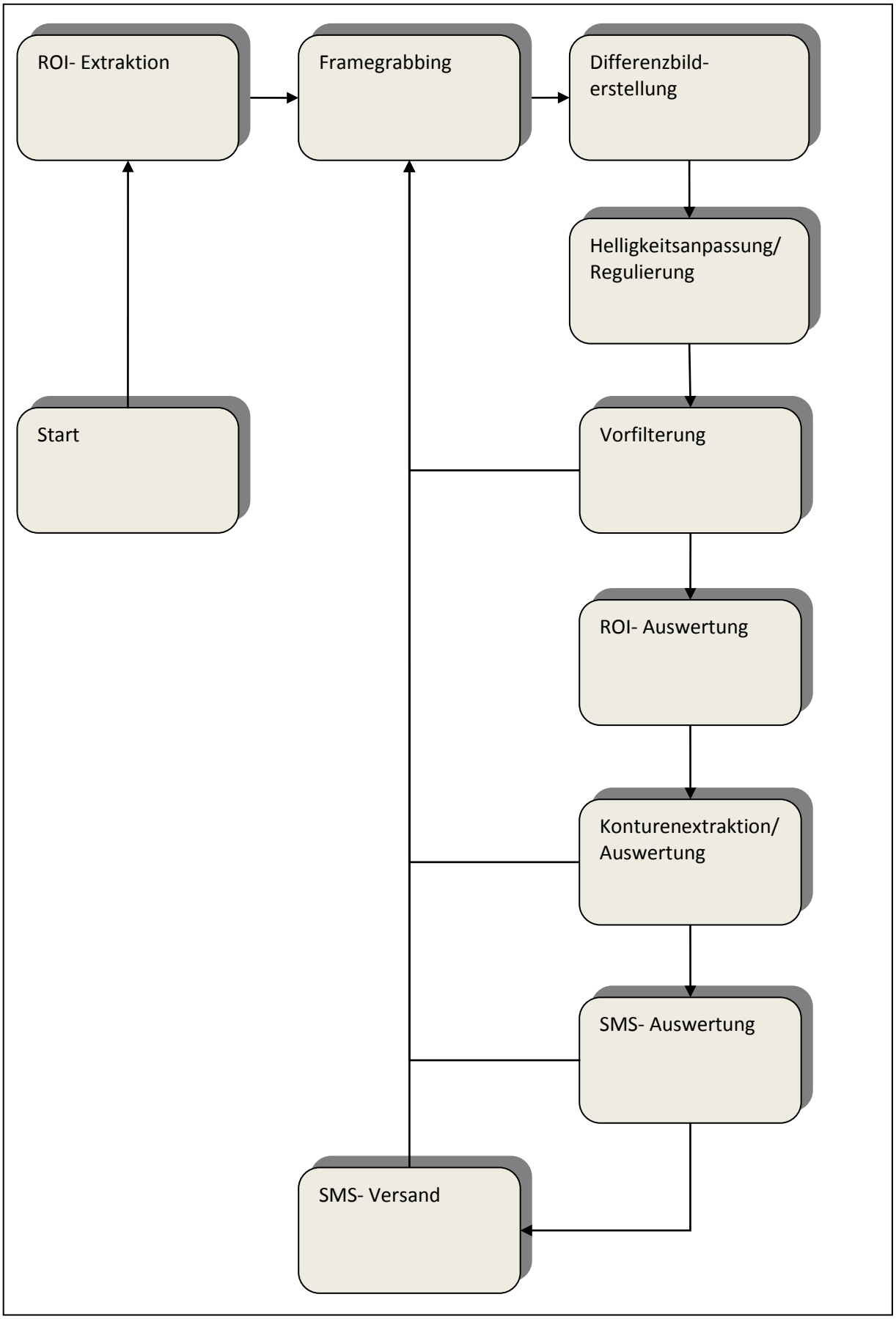

Abbildung 8.2: Ablaufdiagramm MotionDetector

Der Ablauf der Bewegungserkennung ist wie folgt strukturiert. Das Programm MotionDetection wird durch eine Konfigurationsdatei gesteuert, während es in einem regelmäßigen Zyklus den Inhalt der Datei abfragt.
Die Konfigurationsdatei wird von einem CGI- Programm erstellt, auf das ich später eingehe.
Das Program des MotionDetectors arbeitet als ein Hintergrunddämon und liest die Konfigurationsdaten, wie Aktionsbefehl, Auflösung, Devicename, Sensibilität, Handynummer und diverse andere Informationen ein, sofern es vorher durch das CGI- Programm gestartet wurde. Ist die Aktion Bewegungserkennung ausgewählt, so dekodiert die Anwendung als erstes das ROI- Bild, extrahiert die relevanten Informationen aus dem Bild heraus und bereitet zwei Threads vor, die die Erkennungsaufgabe übernehmen.
Der Thread „`getFrameThread`“ ist, wie der Name andeutet, dafür zuständig, die aktuellen Bilder bzw. Frames von der V4L- API zu holen. Dabei stehen ihm bis zu vier Platzhalter für die Frames zum Befüllen zur Verfügung, die in einem FIFO- Verfahren verarbeitet werden. D.h. sind alle Platzhalter mit Frames gefüllt, so wird sich der Frame- Thread schlafenlegen, um die CPU nicht unnötig zu belasten.

Die Put- und Get- Funktionen (`putImagefifo, getImagefifo`) des FIFOs werden durch einen MUTEX vor gegenseitigem Zugriff geschützt und es gibt insgesamt zwei FIFO- Queues, indem der Frame- Thread die eine Queue mit eingelesenen Frames füllt und die verbrauchten Frames aus der anderen Queue entleert. Der Auswertungsthread bedient sich ebenfalls an den zwei Queues, und macht sich erst an die Arbeit, wenn das FIFO- Queue mit mindesten zwei vollen Frames gefüllt ist, aus denen er mit Hilfe der ROI- Informationen und eines Thresholds ein Differenzbild erstellt.
Ein Differenzbild muss eine Vorfilterung durchlaufen, die dafür zuständig ist, Störungen in der Belichtung und korrumpierte Frames herauszufiltern. Dann wird das Differenzbild zu dem Abschnitt der eigentlichen Bewegungserkennung weitergeleitet, und bei erfolgreicher Detektion werden bis zu drei JPEG- formatierte Bilder gespeichert, die beim Bedarf mit Zeitstempel versehen werden können. Am Ende kommt man zu der Stufe, in der ausgewertet wird, ob eine Alarm- SMS versandt werden soll. Je nach eingelesenen Einstellungen des Benutzers, kann der Erkennungszyklus weiterlaufen, oder das Programm beendet sich ordnungsgemäß im Hintergrund.

8.2.1 ROI - Region of Interest

Bei der Verwendung einer „Region of Interest“- Technik ist es möglich, einen freien und verkleinerten Bildausschnitt auszuwählen, in dem sich die informativen Bilddaten befinden. Häufig ist es nutzlos bzw. unerwünscht, wenn allesamt zur Verfügung stehenden Bildbereiche nach Informationen durchsucht werden. Mit anderen Worten, es reicht öfters aus, wenn die Untersuchung auf einen kleineren und ausgezeichneten Bereich des Bildes angewandt und der Rest einfach ignoriert wird. Besonders bei der Bewegungsdetektion, und vor alledem bei Benutzung eines Rechnersystems mit knappen Ressourcen, findet einen solchen Ansatz einen guten Zuspruch.

Abbildung 8.3: ROI- Bild (JPEG, 640 x 480)

Die Abbildung 8.3 zeigt z. B. einen mit der Anwendung und der Webcam (Logitech QuickCam Communicate S) aufgezeichneten Parkplatz, auf dem mehrere Parkplätze zu sehen sind. Möchte der Benutzer nur die Anfahrt bzw. einen bestimmten Parkplatz überwachen lassen, so kann er die Regionen schwarz markieren, die für ihn nicht von Interesse sind. Diese Bereiche wären in diesem Beispiel die gut frequentierte Straße und der Fußgängerübergang vor der Parkplatzeinfahrt.

Abbildung 8.4: ROI- Bild (Straße und Fußgängerübergang markiert)

Durch dieses Verfahren lässt sich auch die aufkommende Datenmenge, die verarbeitet werden müssen, um eine Bewegung zu identifizieren, signifikant verringern.
Ein positiver Nebeneffekt ist auch zu vermerken, denn es wird ebenfalls dadurch eine mühsame Ausrichtung der Kamera vermieden, wobei das Kamerabild, aufgrund fehlender Bildschirmanzeige, erst durch einen PC mit Internetbrowser begutachtet werden kann.

Außerdem, ein weiteres Problem, welches durch das ROI- Bild gelöst wurde, lässt sich erkennen, wenn man die perspektivischen Aspekte der Bewegungsdetektion anhand des Parkplatzbeispiels berücksichtigt.
In welcher Situation kann man behaupten, dass eine Bewegung erfolgreich erkannt werden kann? D. h. wie groß muss ein sich bewegendes Objekt sein, damit Alarm ausgelöst werden kann?
Die Objektgröße ändert sich naturgemäß mit der Entfernung der Kameraposition, und wenn die Kamera keinen mechanischen Zoom aufweist, so muss andere Lösungen her, wenn man nicht möchte, dass das Auftauchen einer Katze oder eines Vögleins auf dem Autodach zu einer Erkennung führt. Um das Problem zu lösen, kann man zusätzlich die minimale und maximale Größe eines zugewarteten Objektes durch die Farben rot und grün angeben, indem man diese jeweils in den unbehandelten Bildregionen markiert.

Abbildung 8.5: ROI- Bild (Min & Max- Objekt 1)

Abbildung 8.5 zeigt in dem Beispiel ein vorbereitetes ROI- Bild, bei dem die Bewegungen eines Objektes, die größer als die rote und kleiner als die grüne Markierungen sind, erkannt werden können. In diesem Falle werden sowohl Menschen als auch Autos erkannt.
Möchte man stattdessen nur Bewegungen von Menschen erkennen, so könnte das nachfolgende ROI- Bild benutzt werden, bei dem das grüne Objekt stark verkleinert wurde.

Abbildung 8.6: ROI- Bild (Min & Max- Objekt 2)

8.2.1.1 ROI – Implementierung

Die von der V4L- API bereitgestellten Bilddaten stehen in einem eindimensionalen und vorzeichenlosen Char- Array, das je nach Bildtiefe, wie bei RGB drei Stellen für einen einzigen Pixelpunkt benötigt.
Bei einem RGB- Bild mit einer Bildauflösung von $320 * 240$ Punkten, muss das Array $320 * 240 * 3$ Stellen aufbringen, um das komplette Bild aufzunehmen.
Bei diesem eindimensionalen Array werden zwei Ansätze vorgestellt, die das markierte ROI- Bild für spätere Analyse der Livebilder auswertet.
Der einfachste, aber gleichzeitig ineffizienteste Weg führt zu einer Schablone, die ebenfalls aus einem Array besteht, das dieselbe Größe wie die Auflösung des ROI-Bildes hat. Dabei werden alle von schwarzen verschiedenen Pixelpunkte des eingelesen ROI- Bildes, bei RGB $Rot * Grün * Blau > thress$, mit 1 versehen, und bei dem Rest mit 0. Später in der Auswertungsphase, können die Arraywerte des jeweiligen Livebildes und der Schablone mit einem &- Operators abgefragt werden, um hieraus ein Differenzbild zu erzeugen, bei dem die schwarzmarkierten Bildpunkte den Wert 0 aufweisen.

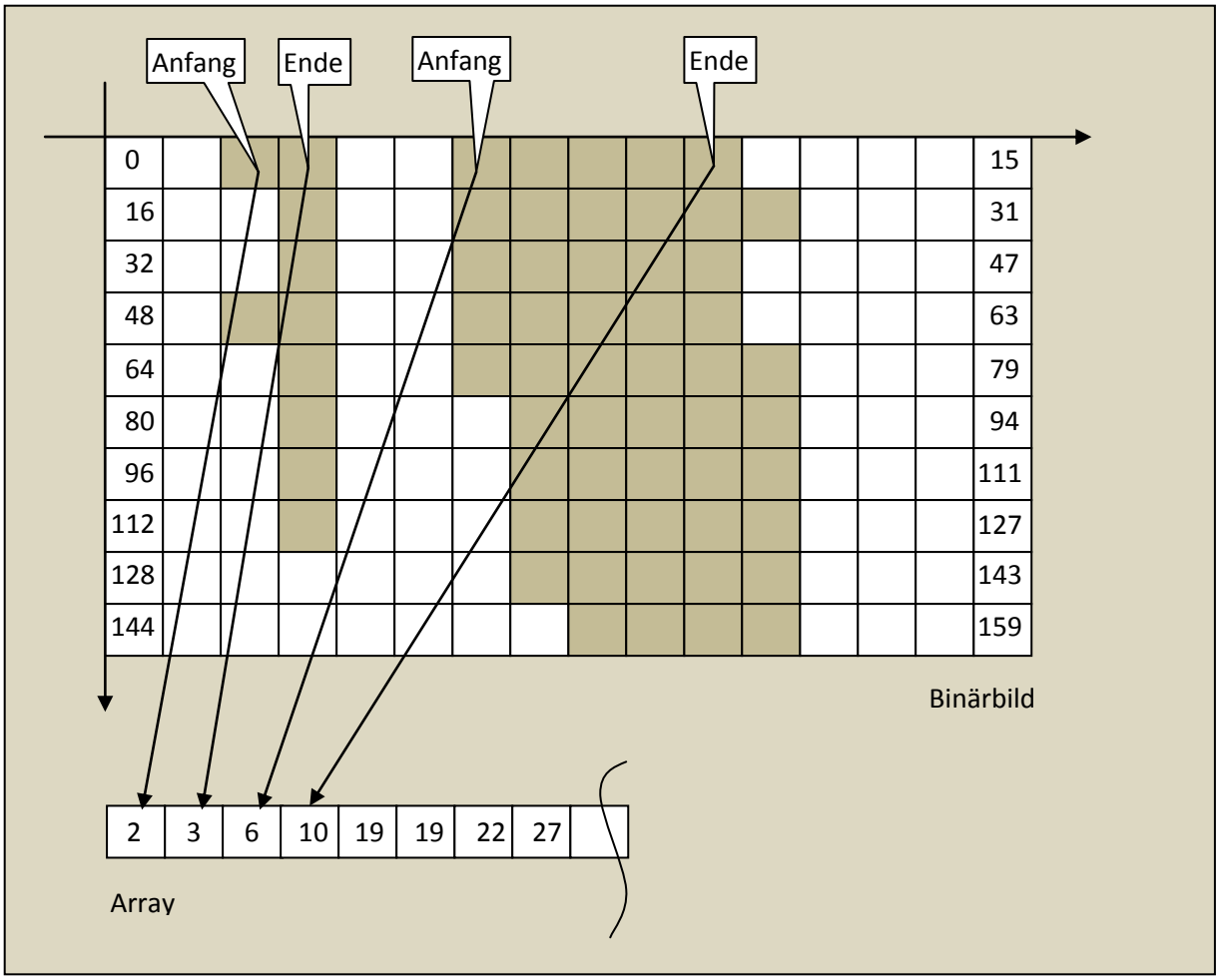

Abbildung 8.7: ROI- Implementierung

Der zweite und effizientere Weg, der auch hier zur Anwendung findet und in Abbildung 8.7 dargestellt ist, bewirkt, dass anstatt alle Stellen einer ausgefüllten Schablone bei der Differenzbilderstellung durchlaufen werden, nur die markierten Stellen berücksichtigt werden. Dabei werden die jeweiligen Anfangs- und Endpositionswerte einer markierten Strecken des ROI- Bildes in ein anderes Array aufgenommen, so dass man danach ein Reduziertes Array vorfindet, in dem nur noch die Positionswerte paarweise stehen, und mit dessen Hilfe alle nicht markierten und fortlaufenden Strecken bei der Analyse leicht übersprungen werden können.

8.2.2 Automatische Helligkeitsregulierung

Die Implementierung einer automatischen Helligkeitsregulierung dient dazu, eine optimale und möglichst gleich bleibende Belichtung für die Bewegungsdetektion zu erzielen.
Insbesondere, weil die Differenzbilder aus den zu unterschiedlichen Zeiten aufgenommenen Frames erstellt werden, ist es wichtig, dass die Helligkeit nur langsam und in kleinen Schritten nachjustiert wird, um das Differenzbild nicht zu korrumpieren.
Je nach verwendeter USB- Webcam, können signifikante Helligkeitsunterschiede im Bild nach der Helligkeitseinstellung durch V4L- API von 0 bis 65000 wahrgenommen werden.

Um die relative Helligkeit eines Bildes herauszufinden, wurde die Funktion `histogramAverage` geschrieben, die das Histogramm des gesamten Bildes berechnet, worauf dann der Durchschnittswert der RGB- Werte ($0 - 255$) ermittelt wird.
Nach mehreren Versuchen mit zwei unterschiedlichen Webcams (Logitech Quickcam STX, und Logitech Communicator) wurde festgestellt, dass der optimale Durchschnittswert sowohl für die menschliche als auch für technische Betrachtung zwischen 110 und 140 liegt.
Kennt man den aktuellen Durchschnittswert der Helligkeit, so kann man die Helligkeit für spätere Bilder korrigieren, falls der Wert nicht zwischen 110 und 140 liegen. Die Funktion `adjustBrightness` übernimmt genau diese Aufgabe.
Wird sie aufgerufen, so ermittelt sie die aktuelle Helligkeit und stellt die Belichtungszeit der Webcam mit einer relativen Schrittgröße, hier 2000 von 65000, nach, je nachdem ob das Bild zu hell oder zu dunkel sei. Diese Funktion wird vom Mainthread nach einer dynamisch definierten Zeit von einigen Sekunden wiederholt aufgerufen. D.h. war eine Nachjustierung der Helligkeit notwendig, so wird diese Funktion in kürzeren Intervallen aufgerufen, um auf konstantem Wege eine rasche Anpassung herbeizuführen.
Auch wenn man die Helligkeit manuell mit der V4L- API einstellen kann, so erzielt man bei nur einem einzigen Fotoschuss nicht immer die gewünschte Helligkeit, vor alledem nicht, wenn die Webcam intern noch irdenwelche Belichtungseinstellungen durchführt, so wie die Quickcam STX von Logitech es tut.

8.2.3 Vorfilter

Jedes Differenzbild muss eine Vorfilterung durchlaufen, in der die ANvB (Anzahl der von Null verschiedenen Bildpunkte) mit der Anzahl der Bildpunkte des minimalen und maximalen Objektes aus dem ROI- Bild verglichen werden. Erst wenn der Wert zwischen den Min- und Maxwerten liegen, kann das Differenzbild gemerkt und das nächste Differenzbild erstellt werden. Dann werden die Gewichtungen, also die ANvB beider Differenzbilder miteinander verglichen, und das Bild mit dem größten Gewicht wird zur weiteren Analyse durchgelassen (Abbildung 8.9).

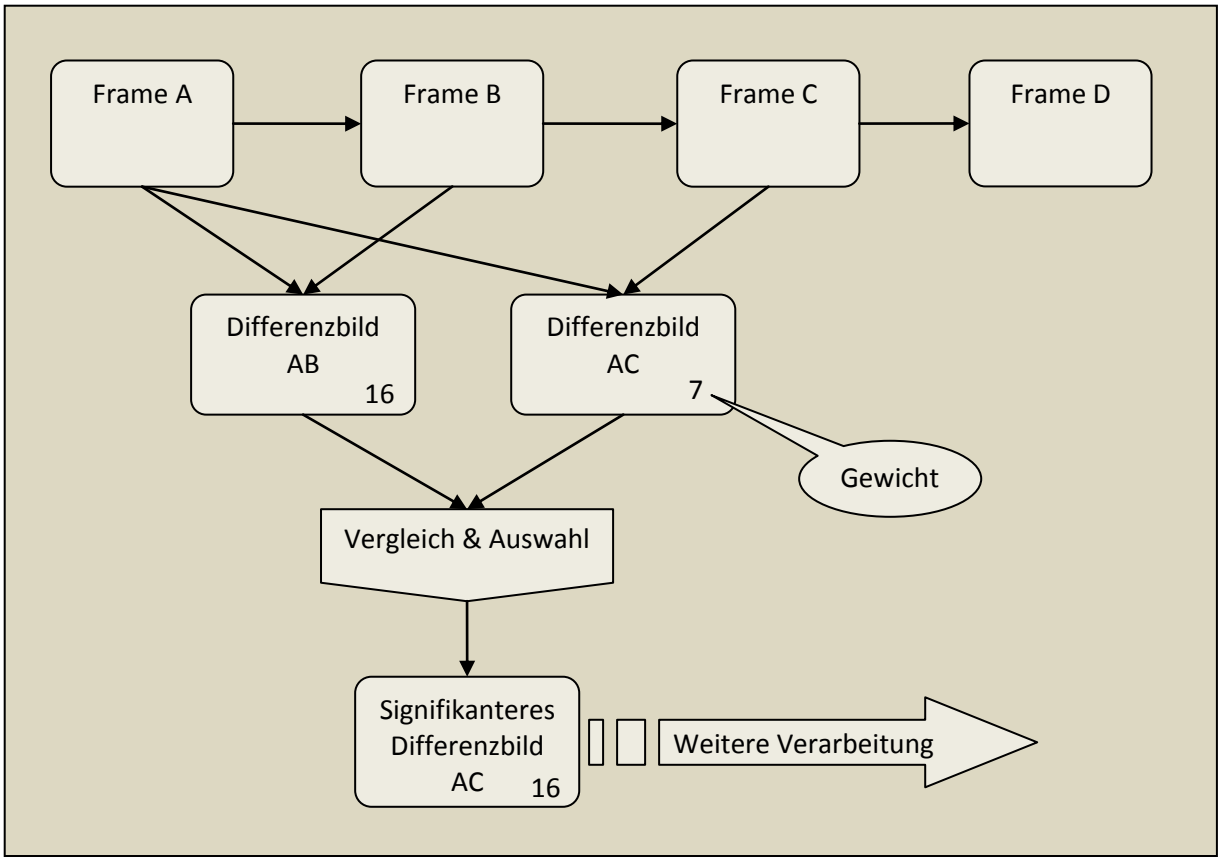

Abbildung 8.8: Vorauswahl eines Differenzbildes

Der Grund dafür ist, es sein kann, dass bei dem ersten Auftauchen einer Bewegung nur ein kleiner Teil eines sich bewegenden Objektes aufgenommen wurde, der zwar die Bedingungen des Vorfilters genügen, aber nicht denen einer rechenaufwendigeren Analyse zur Objekterkennung. Hierbei wäre die Anwendung mit einer zum Scheitern verurteiltem Differenzbild beschäftigt, wobei die kurze Erscheinung des Objektes wieder verwindet, noch bevor die Anwendung mit der Analyse fertig geworden wäre. So hätte man die Bewegung nicht erkannt, und zudem wäre das ganze System mit erfolglosem Berechnen belastet.

Auch werden Differenzbilder stets aus einem Referenzbild, wie das Bild A in der Abbildung 8.8, und einem aktuellen Bild gewonnen, wodurch das bereits beschriebene Problem weitgehend vermieden wird, wobei ein Objekt gerade einen Bildbereich betritt und einen anderen Bildbereich verlässt. Selbstverständlich wird das Referenzbild nach dem Durchlaufen eines definierten Zyklus durch ein aktuelleres Referenzbild ersetzt.

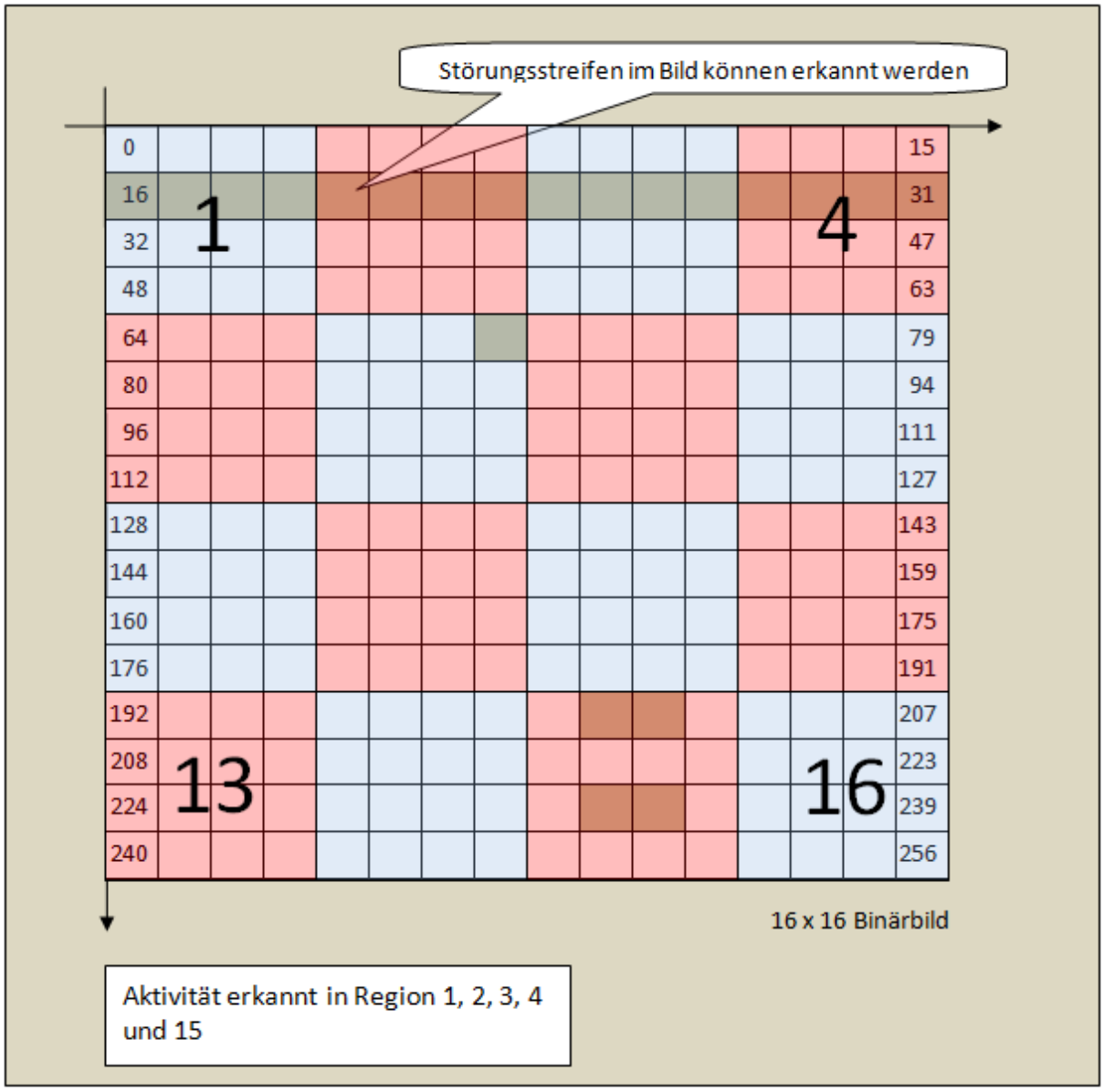

Abbildung 8.9: Bild Segmentierung

Die Aufteilung eines Differenzbildes in Segmenten wird in Abbildung 8.9 dargestellt. Hierbei wird das gewichtigste Differenzbild in 16 gleich großen Segmenten aufgeteilt und jedes Segment für sich weiter untersucht. Die ANvB in dem einzelnen Segment werden mit einem Grenzwert verglichen, und bei Überschreiten dieses Wertes wird das jeweilige Segment als bewegungsaktiv markiert.
Durch gewisse Abfragen der Kombinationen dieser aktiven Teile, lässt Vermutungen darüber geben, ob die Umgebungsbeleuchtung Ursache einer wahrgenommenen Bewegung sei, oder ob sich die Kamera selbst bewegt hätte. Zudem können korrumpierte Zeilen, die wegen Übertragungsfehler unregelmäßig auftauchen, im Bild erkannt und somit ignoriert werden. Im ersten Fall wären mehrere weitverstreute Segmente aktiv, so auch bei einer Bewegung der Kamera und bei im letzten Fall wäre jeweils eine ganze Reihe von Segmenten betroffen.

8.2.4 Bewegungsdetektion mittels Kontorextraktion

Für eine Anwendung, speziell auf einem eigebetteten Gerät, ausgestattet mit weitaus weniger Ressourcen als ein herkömmlicher PC, ist es sehr wichtig, dass eine ausgewogene Lösung gefunden wird, die gleichzeitig ressourcenschonend und verlässig in der Bewegungserkennung ist.
Zumal noch andere Programme auch auf dem EMOD parallel laufen sollen, muss die Anwendung in erster Linie nur Bewegungen erkennen, die Bilder sichern und ggbfs. eine Alarm- SMS schicken können. Sie benötigt daher keine zusätzlichen Informationen, wie Richtung oder Geschwindigkeit des Objektes, die bereits im vorherigen Kapital vorgestellt worden sind.
Auch muss man möglichst sparsam auf intensive Berechnungen von Fließkommazahlen zurückgreifen, da diese aufgrund fehlender Hardwareimplementierung sehr viel langsamer zum Ergebnis führen werden.

Selbstverständlich spielen auch die Art und Form der erwartenden Bewegungen für eine ressourcenschonend Anwendung eine starke Rolle, da die Methoden darauf gezielt entwickelt werden können.
Hier will man nicht die Fluktuationen im gesamten Bildereich einer Bildsequenz erkennen, sonder ein zusammenhängendes Objekt, das z. B. ein Mensch sein kann, der einen bestimmten Bildbereich betritt. Um dieses Objekt in seiner Zusammenhänge zu erkennen, reicht es aus, wenn man nur die Konturen des Objektes ermittelt. D. h. fängt man von einem Konturenpunkt an, die nächsten Konturenpunkte zu traversieren bis man zum Anfangspunkt zurückkehrt. Dann hat der Algorithmus ein zusammenhängendes Objekt in seinem Umfang umrundet und hat es damit gefunden. Zwischendurch können noch die Konturenpunkte gezählt werden, die relative Auskunft über die Größe des Objektes angeben. Stimmt diese relative Größe mit dem aus dem ROI- Bild ermittelten Grenzbereichen überein, so kann es sich um einen Treffer handeln.

Es gibt viele Algorithmen, die die Konturen eines Objektes aus einem Binärbild ermitteln können, so z. B. das in dieser Arbeit verwendete Pavlidis- Verfahren [Meisel 2007]. Es hat gegenüber anderen Verfahren den Vorteil, dass der Algorithmus einen bereits markierten Konturenpunkt nicht noch einmal abfragt, sofern dieser Punkt nicht Teil eines einpixelbreiten Streifens oder der Ausgangspunkt selbst ist.

Der Algorithmus ist richtungsgesteuert, besitzt eine Richtungstafel und sucht, wie Abbildung 8.10 veranschaulicht, zeilenweise nach dem ersten Objektpunkt, welcher auch der Startpunkt ist. Von da an merkt sich der Algorithmus jeden weiteren aktuellen Bildpunkt und traversiert richtungsorientiert an der Objektkante entlang, bis er seinen Ursprungsstartpunkt wieder erreicht hat. Er umkreist gewissermaßen das Objekt und markiert die Objektkante.
Damit der Algorithmus, im Falle, dass ein Objekt nicht vollständig im Bild aufgenommen wurde, nicht im Bereich außerhalb des Bildes sucht, was zu einem Laufzeitfehler führen kann, muss das Differenzbild noch vorbereitet werden. Dazu werden die Ränder des Differenzbildes mit Null überschrieben, was bedeutet, dass der Algorithmus dort nie einen Objektpunkt finden wird und somit auch nicht auslaufen kann.
Selbstverständlich können noch mehrere voneinander unabhängige Objekte in einem Bild sein, so dass das Programm im Selbigen weitersuchen muss, falls das Objekt nicht den Bedingungen genügen sollte.

Damit das Programm ein zuvor untersuchtes Objekt nicht ein zweites Mal in Augenschein nimmt, dienen die Markierungen der Objektkante als Ignorierungsgrund zum Weitersuchen.
Außerdem wird die Detektion in dem aktuellen Differenzbild abgebrochen, sobald ein Objekt gefunden wurde, das größer als das in dem ROI extrahierte Objekt ist. Ansonsten wird das Differenzbild vom Algorithmus bis zum Ende durchlaufen.

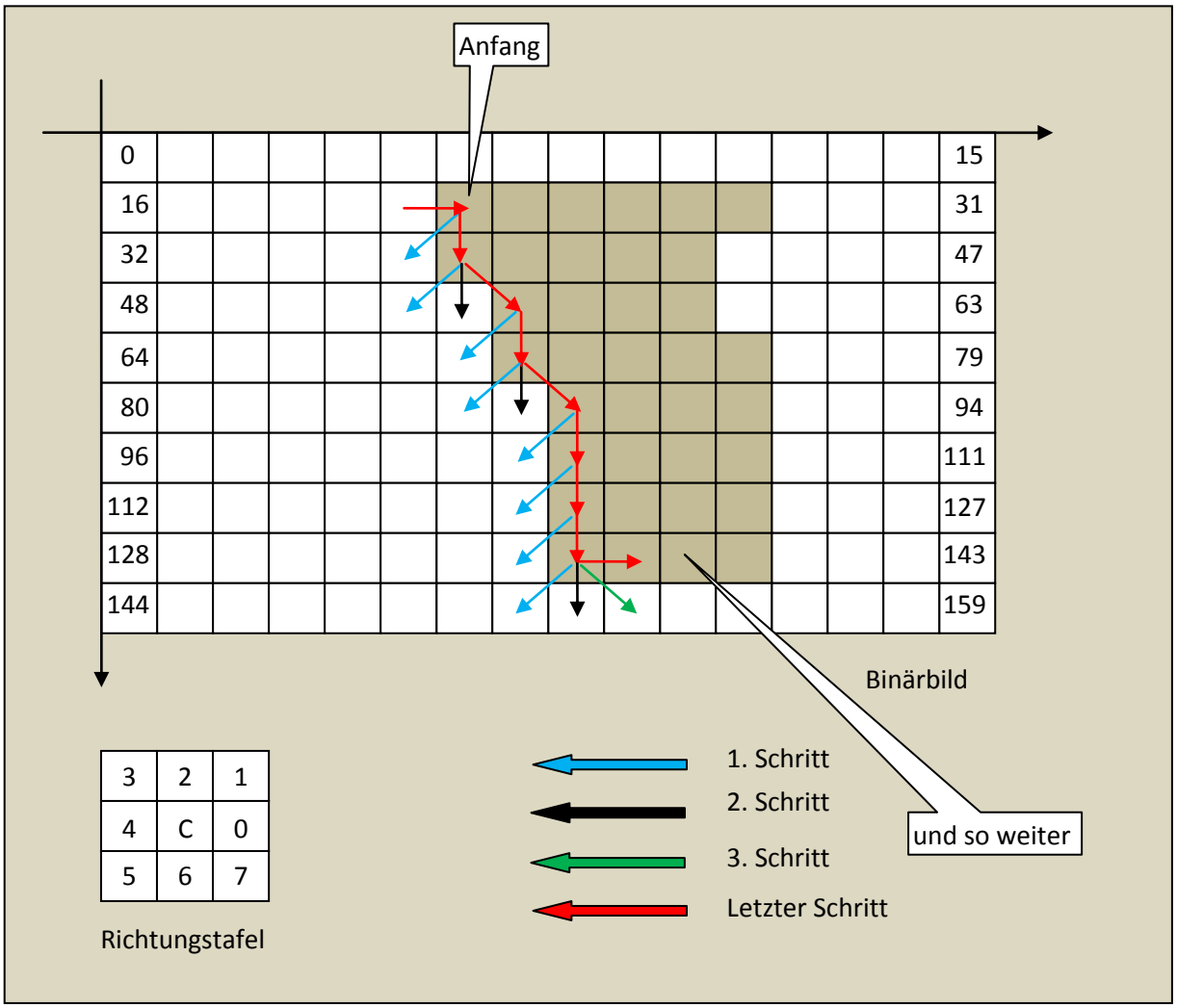

Abbildung 8.10: Konturextraktion nach Pavlidis

Folgende C- Dateien und Funktionen sind entwickelt worden, um die o. b. Eigenschaften zu verwirklichen:

motion_controller.c:

- **writeLine** schreibt einen gegebenen Text in eine bestimmte Zeile in eine Datei.
- **readLine** liest eine bestimmte Zeile aus einer Datei.
- **init_motion_daemon** versetzt den Prozess der Motion_Detector-Anwendung in den Systemhintergrund, und führt ihn als Daemon aus.

- **scheduler** steuert den Ablauf der Anwendung,

- **free_all_memories** gibt alle allokierten Speicher frei.

- **setup_client** fordert eine Message Queue vom Betriebssystem an, um eine SMS- Versandaktion zu veranlassen.

- **sendSMS** veranlasst eine SMS- Versandaktion.

- **main** startet den scheduler und wacht über Änderung der motion.conf Steuerungsdatei.

device_handler.c:

- **device_init** wird benötigt, um die V4L- API zu initialisieren. Dazu wird die Video- Gerätedatei zum Lesen und Beschreiben geöffnet, und es wird die Read- Methode benutzt, um Videodaten aus der Webcam zu lesen.

- **show_webcam_info** schreibt die Einstellungsdaten und die ausgelesen spezifischen Informationen der Webcam in eine Datei, die vom Benutzer betrachtet werden kann.

- **grab_one_frame** benutzt die Read- Methode, um eine Frame aus der Webcam zu lesen. Danach stehen RAW- Bilddaten zur weiteren Verarbeitung bereit.

- **set_adjustment** stellt die Bildeinstellungen, wie Helligkeit, Kontrakt usw. der Webcam über die V4L- API ein.

- **set_channel** setzt den zu verwendenden Videokanal.

- **set_modus** stellt mit Hilfe des zuvor gesetzten Videokanals den Videomodus, wie z. B. NTSC, ein.

- **set_color** dient der nachträglichen Farbkorrektor, um das Videobild ansehnlicher zu machen.

- **adjustBrightness** wird aufgerufen, wenn eine neue Helligkeit übernommen werden sollte.

image_handler.c:

- **renewImage** dient dazu, zu ermitteln, ob die angegebene Anzahl der gespeicherten Bilder erreicht wurde, und überspeichert die anfangsaufgenommenen Bilder Stück für Stück mit Neueren. Dadurch wird erreicht, dass der verfügbare Gesamtspeicherplatz nicht unnötig ausgereizt wird, und dass auch nur die aktuellsten Bilder erhalten bleiben.

- **create_jpeg** codiert die RAW- Bilddaten zu einem JPEG-formatierten Bild und speichert es in den angegebenen Speicherort.

- **read_JPEG_file** decodiert ein JPEG-formatierte Bild und liest so die RAW- Bilddaten ein.

- **printStamp** schreibt mit Hilfe der font_8x8.h Fondatabase Schriftzeichen in ein Bild, und kann auf dieser Weise einen Zeitstempel in das Bild setzen.

command_handler.c:

- **extract_command** extrahiert aus der Steuerungsdatei alle Informationen, und allokiert mit deren Hilfe die Speicher für die gewählte Auflösung und setzt zuletzt die Steuerungsvariablen.

motion_detector.c:

- **setMinMax** setzt die Min- und Maxgrenzwerte für die von Null verschiedenen Bildpunkte.

- **getProcentOfHotRegion** wertet das ROI- Bild aus und berechnet den Prozentwert für die nicht markierten Bildbereiche.

- **getTime** liefert die aktuelle Systemzeit, die auch für den Zeitstempel gebraucht wird.

- **readROIjpeg** initialisiert und bereitet das Einlesen des ROI- Bildes vor.

- **getROIfromImage** erzeugt ein Char Array mit einer dynamischen Größe und füllt es mit Positionswerte der ROI- Bildpunkten auf.
  Erstellt die Grenzen der zu beachteten Bereiche gemäß der in 8.2.2 beschriebenen Methode.

- **getROIDetectSize** berechnet die Min- und Maxwerte für die von Null verschiedenen Bildpunkte aus dem ROI- Bild.

- **analyseExtractedROIData** analysiert das ROI- Bild und setzt ggbfs. die Defaultwerte, falls die extrahierten Informationen nicht mit denen der Benutzereinstellungen kompatibel sind.

- **getROIContourSize** berechnet die Konturengröße eines markierten Bildbereiches.

- **getkonturNextPos** berechnet den Indexwert anhand der aktuellen Suchrichtung und liefert ihn zurück.

- **hasANeighbour** gibt an, ob der im Focus stehende Pixel mindestens einen markierten Nachbarpixel hat.

- **isNotRec** schaut nach, ob ein Pixel bereits für die Untersuchung berücksichtigt worden ist, damit dieser nicht ein weiteres Mal herangezogen wird.

- **setBorderToZero** setzt die sich am Rand befindlichen Pixel im Differenzbild auf Null, damit der Konturalgorithmus nicht ausläuft, was zu einem Systemfehler führen kann.

- **area_analyse** analysiert jede einzelne aufgeteilte Bildregion im Differenzbild, um Veränderungen festzustellen. Anhand bestimmter Muster in der Teilungsmatrix kann Auskunft darüber gegeben werden, ob die Bewegungen sich in einem bestimmten Bildbereich konzentrieren oder weitverstreut sind.

- **area_analyse_init** initialisiert für die Bildteilung die Auswertungsgrößen anhand der gewählten Auflösung.

- **area_analyse_helper** dient der Bestimmung von Veränderungen in einer bestimmten Bildregion.

- **imagefifo_init** initialisiert das FIFO für die von der Webcam geholten Bilddaten.

- **putImagefifo** ist die Implementierung der PUT- FIFO Funktion. Der Aufruf der Funktion wird durch einen MUTEX zum Zwecke der Datenintegrität geschützt.

- **getImagefifo** ist der Counterpart der Implementierung der PUT-FIFO Funktion. Der Aufruf der Funktion wird ebenfalls durch einen MUTEX geschützt.

- **histogramAverage** dient der Berechnung des arithmetischen Mittelwertes der Helligkeit in dem gesamten Bild.

- **grabbingFrames** ist ein Thread, der die Bilder von der Webcam holt und diese in das FIFO packt.

- **motion_contour** ist die eigentliche Funktion zur Bewegungserkennung. Sie funktioniert auch als Thread und benutzt die o. gelisteten Funktionen, um ihre Aufgabe zu erfüllen.

8.2.5 Bildspeicherung und Alarmbewertung

Die Alarmbewertung wird im Haupthread gemacht. Dort wird entschieden, ob eine Alarm- SMS verschickt werden soll. Wie bereits erwähnt, wird die Anwendung durch die Datei `motion.conf` gesteuert. Ist darin der Eintrag einer Telefonnummer zu verlesen, so wird sie zusammen mit einer Textnachricht, bestehend aus dem Zeitpunkt und Anzahl der seit dem Anwendungsstart gespeicherten Bilder, im Messagequeue- Verfahren an einem anderen Router- Prozess geschickt, der in der Lage ist, diese SMS- Nachricht zu versenden.

Die Bilder mit erkannten Bewegungen können je nach Einstellungen begrenzt oder bei Erreichen einer Anzahl erneut überschrieben werden, so dass ein Überlauf des Flash- Speichers weitgehend vermieden wird. Vorher ist es auch möglich, jedes einzelne Bild durch ein aus `cpi2fnt` bekanntes Verfahren, das Teil des Linux-Kernels ist, mit Datum und Nummerierung zu beschriften. Im Falle, dass es die Anzahl der gespeicherten Bilder erreicht wurde, beendet sich die Anwendung mit einem Vermerk über seine letzte Tätigkeit und Informationen über die Aufgezeichneten Bilder, so dass der Benutzer diese Datei im Nachhinein im Browser abrufen kann.

8.3 Webapplikation

Wie bereits erwähnt, dient die Webapplikation dazu, für den Benutzer eine Bedienoberfläche anzubieten, mit dem er die Anwendung im Browser steuern und die im Router gespeicherten Bilder betrachten kann. Die Webapplikation besteht auch aus drei Teilen, den HTML- Dateien, dem Java- Applet und dem CGI- Programm.

8.3.1 Webbedienoberfläche

Mit der Bedienoberfläche ist es möglich, den aktuellen Status der Anwendung zu erfragen, diverse Einstellungen vorzunehmen und in dem mobilen Gerät nach aufgezeichneten Bildern zu suchen und diese ggbfs. wieder zu löschen.

Die Weboberfläche bzw. der Browser des Clients kommuniziert mit dem auf dem Gerät befindlichen Webserver, der die Anfragen an ein CGI- Programm weiterleitet. Normalerweise würde jede noch so kleine Antwort des CGI- Programms zu einem neuen Aufbau der Aufrufseite führen, die zum Einen langsam und zum Zweiten unnötig viel Bytes, die übertragen werden, bedeuten würde. Besonders im mobilen Bereich sind die Datenvolumen ein gewichtiger Kostenfaktor, denen es zu reduzieren gilt. Deshalb wurde in dieser Arbeit bei der Webprogrammierung auf ein Verfahren der asynchronen Datenübertragung, auch Ajax (Asynchronous JavaScript and XML) genannt, zurückgegriffen [HePeSp 2006].
Die Implementierung von Ajax ermöglicht es, innerhalb einer HTML- Seite eine HTTP- Anfrage durchzuführen, ohne die komplette Seite neu laden zu müssen. Dabei können sowohl bestimmte Teile einer HTML- Seite als auch nur reine Nutzdaten nachgeladen werden.

Ajax basiert auf verschiedenen Technogien, wie z. B. DOM (Document Object Model) zur Repräsentation der Daten, JavaScript zur Manipulation von DOM und dynamischen Darstellung der Inhalte. Auch basiert es auf XMLHttpRequest- Objekt zum asynchronen Austausch von Daten zwischen dem Server und Brower.

Nachfolgende Abbildung zeigen und erläutern das Aussehen und die Funktionen der Benutzerschnittstelle in HTML:

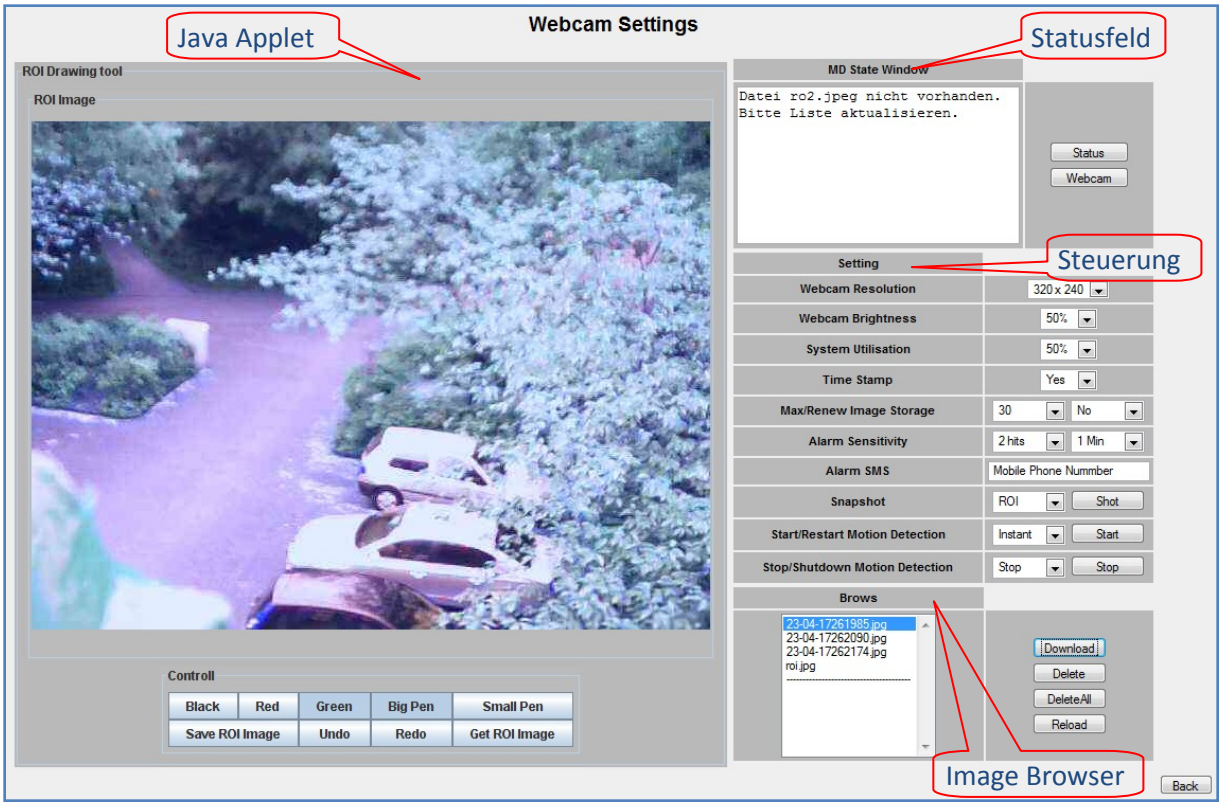

Abbildung 8.11: Bedienoberfläche

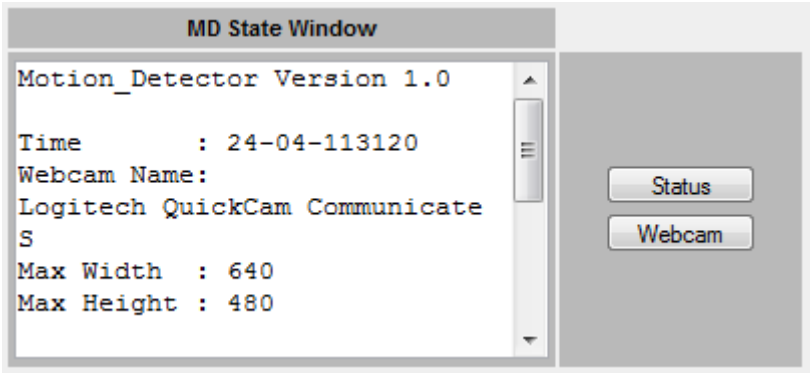

Abbildung 8.12: Statusfeld

Das Statusfeld dient der Anzeige des aktuellen Zustandes von der MotionDetector-Anwendung. Dort drin können alle im Steuerungsfeld beschriebenen Parameter abgefragt und angezeigt werden. Auch liefert es auf Anfrage die spezifischen Informationen der USB- Webcam- Hardware zurück.

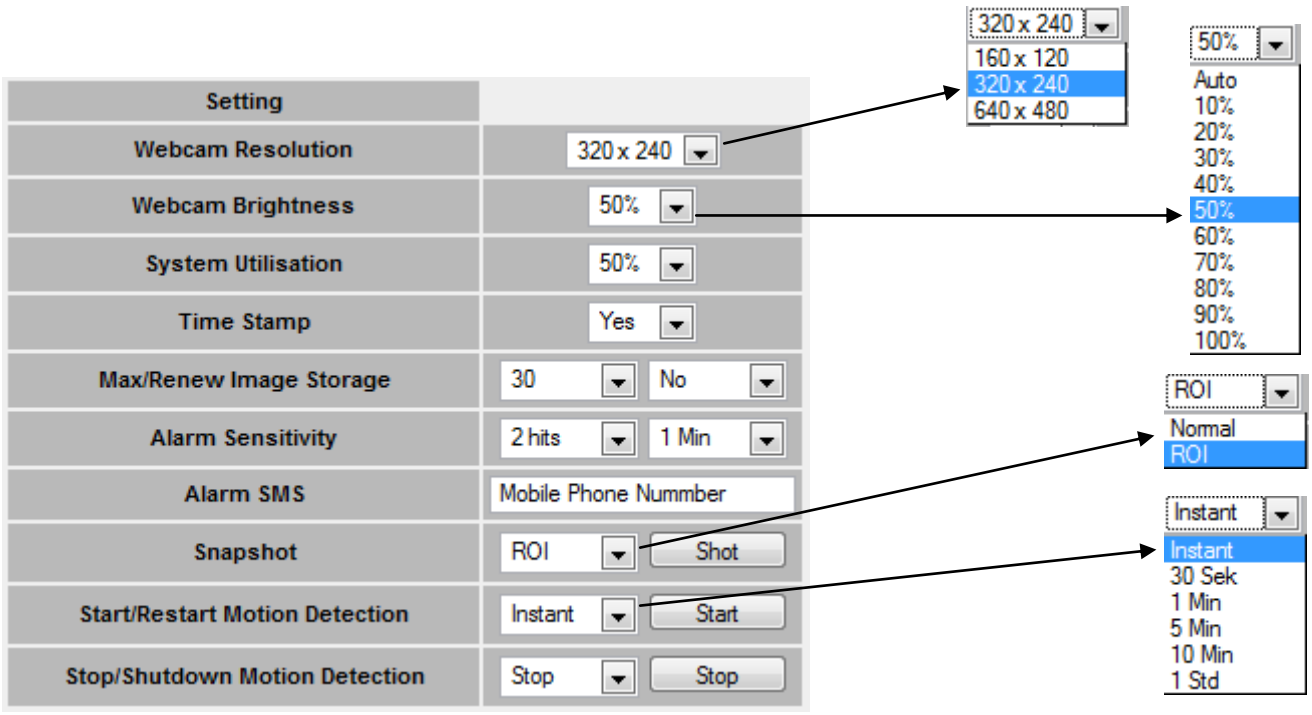

Abbildung 8.13: Steuerungsfeld

Über das Steuerungsfeld kann der Benutzer die Anwendung bedienen. Dazu kann er, wie in der Abbildung 8.13 ersichtlich, verschiedene Einstellungen vornehmen. Es steht ihm drei unterschiedliche Auflösungen zur Verfügung, sofern diese von der Webcam unterstützt werden.
Die Einstellung der Helligkeit kann manuell oder auch automatisch erfolgen und die Systemauslastung der Applikation kann auch entsprechend angepasst werden.
Die Option Alarm Sensitivität erlaubt es dem Benutzer, einzustellen, bei welcher Häufigkeit einer detektierten Bewegung eine Alarm SMS an die im unteren Feld stehende Handynummer verschickt werden soll.
Ansonsten kann man auch ein normales Foto oder ein einzelne ROI- Bild schießen lassen, wo es vom Applet heruntergeladen wird.
Zudem kann die Bewegungsdetektion entweder sofort oder zeitverzögert startet werden. Dies dient dazu, wenn man selbst nach Betätigen des Startknopfes noch das Blickfeld der Kamera kurz betreten muss.

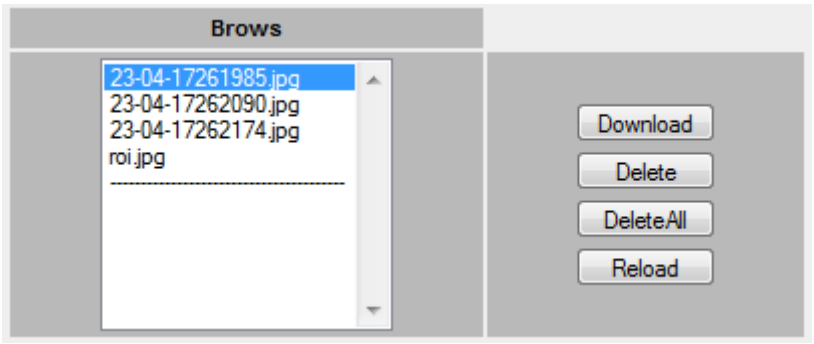

Abbildung 8.14: Image Browserfeld

Im Browserfeld werden alle JPEG- formatierte Bilder im Gerät aufgelistet, wobei diese heruntergeladen oder gelöscht werden können.

8.3.2 Java- Applet

Das Java- Applet wird benötigt, um das ROI- Bild auf einer Webseite eines Browsers zu bearbeiten. Es bietet dem Benutzer auf einfache bequeme Weise ein Bildverarbeitungstool an, das ohne die Hilfe anderer Bildverarbeitungsprogramme auskommt.
Es ist in der Lage das ROI- Bild per Kopfdruck übers Internet vom Webserver des mobilen EMODs herunterzuladen, es im Appletfenster darzustellen, und das Bild nach der Manipulation wieder zum Server hochzuladen, wo es, wie in 8.2.2 beschrieben, zur Informationsgewinnung untersucht werden kann.

Abbildung 8.15: ROI- Bild dargestellt im Applet (Parkplatz & Einfahrt)

Da Applets sicherheitstechnisch nur mit der Adresse kommunizieren dürfen, von wo sie heruntergeladen worden sind, ist es kein Problem die Applet- Implementierung für diese Arbeit nicht zu signieren, das sie weder auf Ressourcen des Client- PCs noch auf andere Internetressourcen zugreifen wird [LeCa 2003].

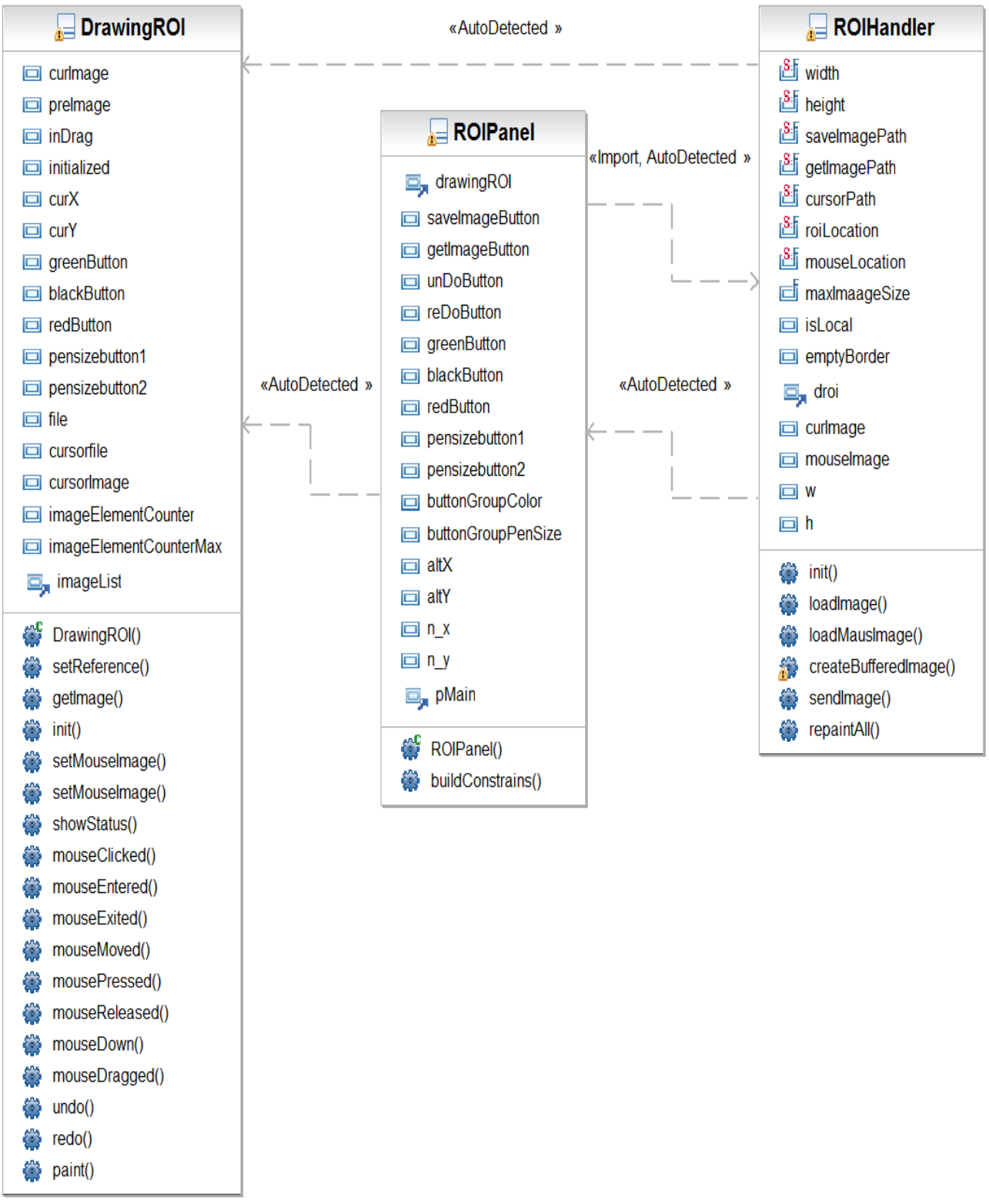

Abbildung 8.16: Java Applet UML

Die Applet- Anwendung besteht aus folgenden Klassen:

- **ROIHandler** ist die Applet- Klasse, welche die anderen Klassen einbindet. Sie besitzt eine `init`- Methode, die aufgerufen wird, sobald das Applet in einem Browser gestartet wird. Die `init`- Methode initialisiert das Applet- Fenster und die Komponenten mit ihren spezifischen Werten, wie z. B. die Größe und Hintergrundfarben. Außerdem enthält sie eine Methode, mit der ein `Image`- Objekt, das das ROI- Bild enthält, zu einem `BufferedImage`- Objekt umwandeln kann, um es beschreibbar zu machen.
  Auch sind hier die Netzwerkfunktionen zum Senden und Empfangen eines ROI- Bildes implementiert worden.
  Die Methode `loadImage` lädt z. B. eine über die URL verwiesene Bilddatei hoch und wandelt sie mit Hilfe von `createBufferedImage` in ein `BufferedImage`- Objekt um.
  Das Senden erledigt die `sendImage`- Methode, die das `BufferedImage`- Objekt in ein `MultiPartFormOutputStream` umwandelt. Hierdurch wird es für das CGI- Programm auf dem EMOD möglich, den Bytestrom des ROI- Bildes vom Applet zu empfangen, um daraus wieder ein JPEG- Bild zu rekonstruieren.

- **ROIPanel** ist eine Komponentenklasse, welche die Buttons erzeugt und diese auch mit `ActionListener`- Objekte für Interaktivität versieht. Hier drin werden die Buttons nach dem Constrains-Verfahren geordnet, d. h. Objekte können tabellarisch sortiert in das Applet- Fenster platziert werden.

- **DrawingROI** ist die Klasse, die für das Zeichnen in dem ROI- Bild verantwortlich ist. Dazu benötigt die Instanz dieser Klasse das `BufferedImage`- Objekt, aus dem sie ein `Graphics`- Objekt erzeugt, welches über die `paint`- Methode aus der Basisklasse von DrawingROI zur Anzeige gebracht wird. Beim Aufruf dieser Methode ist es möglich, in das Graphics- Objekt zu zeichnen.
  Mit Rücksicht auf zu erwartenden und versehentlichen Zeichnungsfehler seitens der Benutzer, wurde jeweils eine `undo`- und `redo`- Funktion eingebaut, die jeden Zeichnungsschritt sowohl rückgängig als auch wiederherstellen kann, sofern das neue Bild nicht erneut vom Server durch das Applet angefragt und übertragen wurde. Hierbei wird vor jedem Zeichnungsschritt, also Betätigen der Maustaste während der Cursor im Bild ist, das `BufferedImage`-Objekt dupliziert und in eine dynamisch wachsende `ArrayList` gesichert. Dabei werden auch diejenigen Objekte, die durch die `undo`-Aktion zum `redo`- Objekte umgewandelt werden, aus der `ArrayList` herausgenommen. Somit wird sichergestellt, dass die `ArrayList` nicht unnötig wachsen wird, und es wird erzielt, dass die Clientressourcen geschont werden.

8.3.3 CGI- Anwendung

Mit der CGI-Schnittstelle (Common Gateway Interface) ist es möglich, Programme im Web bereitzustellen, die von HTML-Dateien heraus aufgerufen werden können. CGI- Programme können ihrerseits Daten oder HTML-Code dynamisch auf dem Server erzeugen, wo diese im Web-Browser angezeigt werden können.

Es gibt zwei Arten Daten an eine CGI- Anwendung zu senden:

- **POST-Verfahren:** Die Daten werden von der HTML-Seite selbst, z. B. über ein Eingabeformular, an den Server gesendet. Dieses Verfahren wird hauptsächlich auch dazu benutzt, um größere Datenmengen, wie Dateien etc., in Empfang zu nehmen.

- **GET-Verfahren:** Hier werden die Daten direkt über die URL übertragen. Diese Methode ist zwar schnell aber auch unsicherer, aufgrund dass die Daten in der URL stehen, sind sie für Jedermann einsehbar.

Das Verfahren der CGI-Schnittstelle ist recht unkompliziert. Die Daten können ganz normal von der `stdin` (Standardeingabe) bei der POST- Methode oder von den Umgebungsvariablen, wie es bei der GET- Methode der Fall ist, empfangen und wenn nötig Antworten in der HTML- Sprache über die `stdout` (Standardausgabe) an den Browser gesandt werden. Sind diese Voraussetzungen erfüllt, können CGI-Anwendungen praktisch mit jeder Programmiersprache implementiert werden.

8.3.3.1 CGI- Implementierung

Die `MDCGIHandler.c` Datei ist die CGI- Implementierung in C, die quasi als zentrale Vermittlungsstelle in dieser Arbeit dient. Sie kommuniziert mit der Webapplikation, nimmt Aufträge von ihr entgegen und steuert den MotionDetector.
Zudem ist sie fähig festzustellen, ob der Prozess MotionDetector gerade läuft, und startet diesen, falls es nicht so sein sollte.
Die MDCGIHandler.cgi ist die übersetzte Anwendung, die bei jedem Aufruf durch die HTML- Seite neu gestartet wird, und die Umgebungsvariable `REQUEST_METHOD` nach der Aufrufmethode abfragt. Hierbei ist zu unterscheiden zwischen Formulareingaben (GET) und Datenstrom (POST).
Bei der GET- Methode werden die Formulareingaben vom Server in die Umgebungsvariable `QUERY_STRING` abgelegt, die ausgelesen und zum Auswerten verarbeitet werden können. Der MDCGIHandler entscheidet anhand der ausgewerteten Befehle, welche Aktionen durchzuführen sind. Dabei kann er die Steuerungsdatei motion.conf, die von dem MotionDetector zur Parametrisierung genutzt wird, neu erzeugen oder einzelne Zeilen überschreiben.
Auch leitet er den Inhalt dieser Datei mit dem aktuellen Prozesszustand des MotionDetectors an die Aufrufseite weiter, damit der Benutzer einen kurzen Überblick über den Status der Anwendung erhalten kann.

Das POST- Einleseverfahren wird dazu benutzt, um das vom Java- Applet modifizierte ROI- Bild serverseitig in Empfang zu nehmen. Hierbei muss geprüft werden, ob die Umgebungsvariable `CONTENT_TYPE` das Übertragungsformat "`multipart`" enthält. Das Verfahren "`multipart/form-data`“ erlaubt den

Transfer ganzer Dateien, egal ob es sich dabei um Bilder, Texte oder andere Daten handelt. Um eine Datei hochzuladen, muss die Byte- Größe des Bodys aus `CONTENT_LENGTH` in Erfahrung gebracht werden, die für diese Arbeit auf maximal 100kB begrenzt ist. Danach muss noch der Headerteil aus `stdin` weggefiltert werden, bis nur noch die Rohdaten zum Auslesen und Abspeichern übrig bleiben.

Folgende Funktionen sind in `MDCGIHandler.c` entwickelt worden, um die o. b. Eigenschaften zu bewerkstelligen:

- **fileUpload** lädt über das POST- Verfahren eine JPEG- Datei vom aufrufenden Applet herunter.
- **printHttpHeader** schreibt den HTTP- Header, der als Einleitungsteil einer Antwort des Servers an den Browser dient.
- **print_location** für den Fall, dass ein alter Browser keine automatische Weiterleitung unterstützt.
- **writeLine** schreibt einen gegebenen Text in eine bestimmte Zeile in eine Datei.
- **readLine** liest eine bestimmte Zeile aus einer Datei.
- **getdata** nimmt die Anfrage des Browsers entgegen, und wertet aus, ob es sich um eine POS oder um eine GET- Anfrage handelt.
- **hex2ascii** Wandelt einzelne Hexzeichen, die übers Netz übertragen worden sind, in ASCII-Zeichen um und kodiert die „+“ Zeichen, die als Leerzeichen dienen, in echte Pluszeichen um.
- **convert** konvertiert einen String von zwei HEX- Zeichen in original ASCII- Zeichen um. Die Funktion wird von hex2ascii gebraucht.
- **creatPairs** erstellt eine Liste von Variablen- und Wertepaaren. Sie dient dem GET- Verfahren.
- **isJpegFile** prüft nach, ob die hochzuladende Datei, vom Typ JPEG ist bzw. dessen Endung aufweist.
- **scanImages** durchsucht in einem Verzeichnis nach allen JPEG- Bilder und übergibt die Namen an die aufrufende HTML- Seite.
- **deleteImage** löscht ein JPEG- Bild anhand des übergebenen Namens und liefert eine entsprechende Nachricht zurück, falls das Bild nicht vorhanden ist.
- **deleteAllImages** löscht alle vorhandenen JPEG- Dateien in einem bestimmten Verzeichnis, sofern diese die Dateinamenendung .jpg oder .jpeg aufweisen.

- **printSeletTag** übermittelt einen HTTP-Select- Tag mit allen vorhandenen JPEG- Dateinamen. Die Funktion unterstützt die asynchrone Übermittlung von AJAX.

- **writeMDConfigFile** erstellt und modifiziert die Konfigurationsdatei motion.conf.

- **printStatus** übermittelt die Statusinformationen an die HTML-Seite, wo diese in den HTTP-TextArea- Tag geschrieben werden.

- **printCap** lässt den MotionDetector die spezifischen Informationen der eingesteckten USB- Webcam erfassen und übermittelt diese an das Statusfenster der HTML- Seite.

- **startMD** startet den MotionDetector- Prozess, falls er gerade nicht laufen sollte, und veranlasst ihn mit der Bewegungserkennung zu beginnen.

- **main** wertet mit Hilfe der beschriebenen Funktionen die Befehle seitens der HTML- Webseite aus und leitet entsprechende Aktion ein.

8.4 SMS-Versanddienst Implementierung

Das TAINY EMOD verfügt vom Hause aus einen SMS- Dienst, der in der Anwendung APL.cpp (Application) für den Routerbetrieb implementiert worden ist. Aber um den Dienst von außen durch ein anderes Programm nutzbar zu machen, ist eine Erweiterung der Nativanwendung nötig, da sonst jeder andere Prozess, der über IPC (Inter Process Communication) mit der APL kommunizieren will, die mqueue.h C-Bibliothek implementieren müsste, wobei die Anwendung des kommunikationswilligen Prozesses, nach dem statischen Übersetzen, zusätzlich 2 MB an Größe anwachsen wird, da die besagte Bibliothek sich nicht in der Laufzeitumgebung befindet.
Deshalb, um die Implementierung sparsam zu halten, habe ich für diese Arbeit eine Erweiterung (MOTION_SMS_HANDLER.cpp) in C++ geschrieben, die von der APL eingebunden wird, und die einen Thread erzeugt. Dieser Thread ist in der Lage, eine MQ- Message (Messag Queue) vom MotionDetector auch über IPC zu empfangen. Hierbei wird anstatt `mqueue.h` die platzsparsamere IPC- Bibliothek `msg.h` in beiden Anwendungen (MotionDetector u. APL- Erweiterung) eingebunden.
Der eingefügte Thread liest aus der Message, die darin enthaltene Telefonnummer inkl. Nachrichtentext und sichert auch die bereits für den Routerbetrieb gespeicherten Informationen, bevor er die neue SMS- Nachricht verschickt. Grund dafür ist, dass die einmal durch die APL verschickte SMS Nachricht, alle alten Einträge mit der Telefonnummer und dem Nachrichtentext überschreibt, die wiederhergestellt werden müssten. Dazu schickt der Erweiterungsthread die notwendigen MQ- Messages zum Auslesen und Einspielen der Benutzerdaten mit Hilfe der Funktionen aus der `mqueue.h` Bibliothek. Nachdem die SMS- Nachricht gesendet worden ist, protokolliert der obige Thread die getätigte Aktion in eine Alarm- Datei, welche von der CGI- Anwendung an die Webapplikation weitergeleitet werden kann.

9 Fazit

Das Ziel dieser Arbeit ist es, eine Bibliothek zur selbstständigen Bewegungsdetektion in Echtzeit zu entwickeln, die auf einem eingebetteten System läuft, und welches automatisch eine USB- Webcam erkennt und unterstützt. Ferner sollte sie auch eine SMS- Nachricht an ein mobiles Telefon senden können, um den Benutzer über eine erkannte Bewegung zu unterrichten. Im Bezug auf dem letzten Punkt ist es wichtig, dass die SMS- Nachricht erst durch den Äther geschickt wird, wenn man mit aller Wahrscheinlichkeit davon ausgehen kann, dass es sich wirklich eine erfolgreich detektierte Fremdbewegung in homogenen Bildfolgen gehandelt hat.

Um den o. stehenden Anforderungen zu genügen, ist ein Softwarepaket entstanden, das aus drei Hauptbestandteilen besteht, in denen eine Kombination aus diversen Technologien zum Einsatz gekommen ist:

- **MotionDetector** ist die eigentliche Anwendung zur Bewegungsdetektion, die mit Hilfe des V4L- APIs und des Linux-Webcam- Treibers die USB- Webcam parametrieren und steuern kann.

- **Webapplikation** dient in erster Linie der Parametrierung der Anwendung MotionDetector und der Betrachtung der aufgenommenen Bilder sowie der Anzeige der Statusinformationen.

- **SMS- Dienst** ist eine Erweiterung der Nativimplementierung des SMS-Versanddienstes. Sie kommuniziert mit dem MotionDetector per IPC und ermöglicht eine Sicherung und Wiederherstellung der alten Eingaben, die zum Normalbetrieb des Wireless- Routers nach einer vom MotionDetector angestoßenen SMS- Nachricht notwendig sind.

9.1 Analyse der Ergebnisse

Das Programm MotionDetector stellt automatisch die Helligkeit der Webcam so ein, dass aus den gewonnenen Folgebildern ein optimales Differenzbild erstellt werden kann, welches für weitere Untersuchungen gebraucht wird. Es trifft auch eine Vorauswahl von zwei zeitlich verschiedenen Differenzbildern, um ein bestmögliches Resultat zu erzielen.

Die Implementierung der ROI- Eigenschaft ermöglicht eine den Anforderungen des Anwenders angepasste Bewegungserkennung, sowie eine ressourcenschonende Berechnung, da Bildregionen explizit für die Auswertung ignoriert werden können.

Mit der Webapplikation wird auch eine benutzerfreundliche User- Interface zur Parametrierung und Betrachtung der Bilder angeboten, die ebenfalls über das Internet aufrufbar ist. Das dazugehörende Java-Applet lädt das ROI- Bild vom mobilen Router herunter und stellt dem Benutzer auf bequeme Art und Weise ein Bildverarbeitungstool zur Verfügung, das standalone in einem javafähigen Browser läuft.
In dem Applet sind auch die Undo- und Redofunktionen implementiert, die ein komfortables Korrigieren der Fehlzeichnungen erlauben.
Die Implementierung der SMS- Erweiterung funktioniert auch wie erwartet, und benachrichtigt den Benutzer über erkannte Bewegungen in Echtzeit. Zudem wird der

Benutzer über den Zeitpunkt und die Anzahl der im Flash gesicherten Bilder informiert.

Die gesamte Anwendung zur Bewegungsdetektion läuft parallel zu den heimischen Programmen des EMODs und durch Einstellung der Systemauslastung kann eine ausgewogene Balance zwischen den laufenden Programmen erzielt werden, so dass diese nicht stark behindert werden.

Zudem wurde durch Zeitmessungen ermittelt, dass das Herunterladen eines 320 ∗ 240 Frames inkl. Dekodieren vom JPEG- Format in Rohdaten durch den Kerneltreiber und je nach verwendeter USB- Webcam ca. 300 ms unter Volllast dauert. Anschließende Filterauswertungen verbrauchen zusätzliche 700 ms. Und zum Schluss kann die Bewegungsdetektion mittels Konturenermittlung je nach Bewegungsfragmentierung im Differenzbild bis zu 3 Sek. beanspruchen.

Abbildung 8.17: ROI- Bild Parkplatzüberwachung 1

In dem ROI- Bild in Abbildung 8.17 ist zu erkennen, dass das rote und das grüne Objekt die relative Größe einer Person entsprechen. Auf diese Weise sollte die Bewegungserkennung nur auf Personen angewandt werden, die in ihrem Umfang zwischen denen der beiden Objekte liegen. Hierbei entspricht ein Auto, das weitaus größer wäre, nicht den Suchkriterien und würde somit auch von der Anwendung ignoriert werden.

Wie erwartet konnte der MotionDetector eine Bewegung feststellen, die wie in Abbildung 8.18 ersichtlich, die Person auf dem fahrenden Fahrrad darstellt.

Abbildung 8.18: Erste Person auf einem Fahrrad wurde erkannt

Abbildung 8.19: Silbernes Auto parkt ein bei t=08:40:38

Obige Abbildung zeigt deutlich, wie das silberne Auto bei der Bewegungsdetektion ignoriert wurde, als es auf den Parkplatz gefahren ist. Erst als das Auto stehen blieb und die Türen aufgingen, konnten die passenden Bewegungen detektiert und die untersuchten Bilder (Abbildung 8.19 – 8.22) gespeichert werden. Dabei haben sich

die Bewegungen wegen dem stehenden Auto stark reduziert, so dass das Differenzbild zur weiteren Analyse durchgelassen wurde.

Abbildung 8.20: Silbernes Auto bei t=08:40:41

Abbildung 8.21: Silbernes Auto bei t=08:40:45

Abbildung 8.22: Silbernes Auto eingefahren t=08:40:56

Abbildung 8.23: ROI- Bild bearbeitet, um Autos zu detektieren

In nächstem Beispiel wird demonstriert, wie die Anwendung nur zur Bewegungserkennung von Autos verwendet wird. Dazu wurde das ROI- Bild aus Abbildung 8.23 verwendet, beidem die Min- und Max- Größe passend zu

der eines Autos gesetzt sind. Analog zum obigen Beispiel sind ebenfalls drei zeitnahe Bilder aufgenommen worden.

Abbildung 8.24: Schwarzes Auto erkannt bei t=08:06:57

Abbildung 8.25: Schwarzes Auto erkannt bei t=08:07:00

Abbildung 8.26: Schwarzes Auto erkannt bei t=08:07:03

9.2 Ausblick

Die Implementierung einer softwarebasierten Bewegungsdetektion auf einem eingebetteten Gerät wird üblicherweise aus Performancegründen nicht angewandt. Aber mit fortschreitender Entwicklung im Bereich der Mikrokontrollertechnologie wird auch dieser Ansatz vermehrt Zuspruch finden.

Mit mehr Rechenpower und entsprechendem Speicherplatz lassen sich auch die in Kapitel 7 vorgestellten und aufwendigeren Verfahren zur Bewegungs- und Objekterkennung im Embedded- Bereich verwirklichen. Dazu ist diese Arbeit so entwickelt worden, dass sie leicht erweitert und modifiziert werden kann. So könnte man später auch die Form der aus dem ROI- Bild extrahierten Objekte über deren Momente bestimmen, um eine berechenbare Objekterkennung zu realisieren. Dies wird insbesondere dann sinnvoll, wenn der hier verwendete Mikrokontroller gegen einen kommenden und leistungsfähigeren Kontroller der AT91SAM- Familie aus dem Hause Atmel ausgetauscht würde.

10 Anhang

Anbei werden alle relevanten Quellcodes, die für diese Arbeit angefertigt worden sind, dargestellt. Aber um die Dimension dieser schriftlichen Arbeit unter Grenzen zu halten, werden die HTML- Dateien und die Makefiles der C- Projekte nicht mehr angezeigt.
Dafür beinhaltet die beiliegende CD- ROM, alle entwickelten Dateien inkl. der vollständigen Projekte, die sowohl in C als auch in Java unter Eclipse erstellt worden sind.

Die CD- ROM beinhaltet:

MotionDetector (Eclipse C- Projekt)
CGI- Anwendung (Eclipse C- Projekt)
Applet- Anwendung (Eclipse Java- Projekt)
HTML- Dateien
APL- Erweiterung (C++)
EMOD GPL Opensource Package
JPEG- Library

Literaturverzeichnis

[Jähne 2002] Berne Jähne, Professor Dr. : Digitale Bildverarbeitung- 5. Überarbeitete und erweiterte Auflage. Springer- Verlag Berlin Heidelberg New York. ISBN 3-540-41260-3

[Wolf, C 2006] Jürgen Wolf: C von A bis Z- 2., aktualisierte und erweiterte Auflage 2006. Galileo Computing- ISBN 3-89842-643-2

[V4L] Video4Linux users and developers framework 2008 (Wiki). http://www.linuxtv.org/v4lwiki/index.php/Main_Page

[Wolf, Linux 2006] Jürgen Wolf : Linux-UNIX-Programmierung- 2., aktualisierte und erweiterte Auflage 2006. Galileo Computing- ISBN 978-3-89842-749-4

[HePeSp 2006] Nick Heinle, Bill Pena & Ulrich Speidel: Webdesign mit JavaScript & Ajax- 2. Auflage Oktober 2006. O' REILY Verlag- ISBN 978-3-89721-471-2

[LeCa 2003] Laura Lemay, Roger Cadenhead: Java 2 in 21 Tagen- 1. Auflage 2003. Markt + Technik Verlag- ISBN 3-8272-6528-2

[KoPl 2006] Michael Kofler, Jürgen Plate: Linux für Studenten- 1. Auflage 2006. Pearson Studiums Verlag- ISBN 13: 978-3-8273-7205-5 & 10: 3-8273-7205-04

[GoBrDa 2002] Joachim Goll, Ulrich Bröckl, Manfred Dausmann: C als erste Programmiersprache (Vom Einsteiger zum Profi)- 4. Auflage 2002. Teubner Verlag- ISBN 3-519-32999-9

[BrDeKu 2007] Brinker, T. , Degenhardt, H. , Kupris, G.: Embedded Linux- Praktische Umsetzung mit uClinux 2007, VDE- Verlag- ISBN 978-3-8007-2716-2

[Münz 2006] Stefan Münz: Professionelle Websites , Programmierung, Design und Administration von Webseiten . Addison-Wesley 2006- ISBN 3-8273-2370-3

[Atmel 2008] Atmel Corporation AT91SAM9260 Technical Reference Manual Atmel Corporation- http://www.atmel.com/dyn/products/product_card.asp?part_id=3870

[Xhaard 2008] Michel & Sylvie Xhaard : Linux kernel webcams Driver GSPCA / SPCA5xx 2008- http://mxhaard.free.fr/spca5xx.html

[SELFHTML 2007] SELFHTML 8.1.2: HTML-Dateien selbst erstellen. http://de.selfhtml.org/

[Meisel 2007] Andreas Meisel, Professor Dr.-Ing.: WP Robot Vision (Vorlesungsscripts) 2007. Hochschule für Angewandte Wissenschaften Hamburg- http://www.informatik.haw-hamburg.de/wp_robot_vision.html

[Haussecker 1993] Horst Haussecker: Mehrgitter-Bewegungssegmentierung in Bildfolgen mit Anwendung zur Detektion von Sedimentverlagerungen. Universität Heidelberg, Diplomarbeit 1993
http://klimt.iwr.uni-heidelberg.de/Projects/BAW/diplhhaus/

[Brandt 2005] Lars Brandt: Entwicklung eines Objekttrackers für Embedded Systeme zur Steuerung mobiler Roboter. Hochschule für Angewandte Wissenschaft Hamburg, Diplomarbeit, 2005. –
http://users.informatik.haw-hamburg.de/~kvl/brandt/diplom.pdf

[EMOD] TAINY EMOD Spezifikation, Dr. Neuhaus Telekommunikation GmbH 2008
http://www.neuhaus.de/frameset.htm

[JPEG] Independent JPEG Group, JPEG- Library 2008-
http://www.ijg.org/

Glossar

ARM9	Advanced RISC Machines- Architektur für 32 Bit Mikroprozessoren.
GSPCA	Ist ein USB- Webcam Treibermodul, das extra Linux geschrieben wurde. Es unterstützt über 240 USB- Webcams.
API	Application Programming Interface- Ist eines Programmier- bzw. Anwendungsschnittstelle.
V4L	Video for Linux API- Ist eine Applikationsschnittstelle für Multimediageräte unter Linux.
LINUX	Ist ein freies Multiplattform und Mehrbenutzer Betriebssystem.
Linux Kernel	Ist der Betriebssystemkern unter Linux. Er legt Prozess- und Datenorganisationen fest und verwaltet Benutzerprogrammen.
JAVA APPLET	Ist ein Computerprogramm in der Programmiersprache Java. Es wird üblicher weise im Browser ausgeführt, um mit Benutzer zu interagieren.
AJAX	Asynchronous JavaScript and XML- Ist ein Konzept zur asynchronen Datenübertragung zwischen Server und Browser. Erlaubt innerhalt einer HTML- Seite eine HTTP- Anfrage dynamisch durchzuführen ohne die Seite komplett laden zu müssen.
ROI	Region of Interest- Untersuchung eines verkleinerten Bildausschnitts der Bilddaten.
CGI	Common Gateway Interface- Ist ein Mittel Webseiten dynamisch bzw. interaktiv zu machen.
EMOD	Wireless GSM Funktrouter für mobile Geräte. Ist auch das Zielplatform für diese Arbeit.

Quellcodes

Motion_controller.c (Eclipse Projekt motion)

```
1  /*
2   * Controlls the Motion Detector, inits the grabbing and analyse threads
3   * Starts the damon
4   */
5
6  #include "motion_controller.h"
7
8  int writeFile(char *path, char *buffer)
9  {
10        FILE *file;
11        file =fopen(path, "w+");
12
13        if(file == NULL)
14        {
15              return -1;
16        }
17
18        fputs(buffer, file);
19        fclose(file);
20        return 1;
21 }
22
23 /*
24  Schreibt eine Zeile in die Datei, angegeben durch row(Zeilennummer beginnt bei 1).
25  path ist der Filename, wo der String buffer in diese Datei geschrieben wird.
26 (Generel nuetzlich um eine bestimmte Zeile in einer Datei durch einen
27  neuen String zu ersetzen)
28  */
29 int writeLine(char *path, char *buffer, int row)
30 {
31        FILE *file;
32        FILE *tmpfile;
33        int strLength;
34        int done=0;
35        int readLineSize = 255;
36        int filefd;
37
38        char str[readLineSize];
39        char temp[readLineSize];
40        char tmpfilePath[] = "/motion/cgiXXXXXX";
41
42        strLength= strlen(buffer);
43
44        strncpy(str, buffer, strLength);
45        str[strLength]='\0';
46
47        file =fopen(path, "a+");
48        filefd = mkstemp(tmpfilePath);
49        tmpfile= fdopen(filefd, "w+");
50
51        if(file == NULL)
52        {
53              return EXIT_FAILURE;
54        }
55        if(tmpfile == NULL)
56        {
57              return EXIT_FAILURE;
58        }
59
60        str[strLength]='\n';
61        str[strLength+1]='\0';
62
63        int i=0;
64        //aus Originaldatei lesen und nach Tmpdatei kopieren.
```

```
65          //An der n-ten Zeile neue Zeile in Tmpdatei einfuegen
66          while(!feof(file)||i<=row)
67          {
68                  if(fgets(temp, readLineSize, file)!=NULL)
69                  {
70                          //n-te Zeile gelesen
71                          if(i==row-1)
72                          {
73                                  fputs(str, tmpfile);
74                                  done=i;
75                          }
76                          else
77                          {
78                                  fputs(temp, tmpfile);
79                          }
80                  }
81                  else
82                  {
83                          //OriginalFile ist leer, anfuegen an erster Stelle
84                          if(done==row-1)
85                          {
86                                  if(done==0&& i==0)
87                                  {
88                                          fputs(str, tmpfile);
89                                  }
90                                  break;
91                          }
92          //wenn Position im Orgfile leer, aber der String dort eingefuegt w. muss
93                          if
94                          (i==row)
95                          {
96                                  //n-te Zeile gelesen
97                                  fputs(str, tmpfile);
98                          }
99                          else
100                         {
101                                 //ein'\n' bereits vorhanden? ja->nichts tun
102                                 if(temp[strlen(temp)-1]=='\n')
103                                 {
104                                         strcpy(temp, "x");
105                                 }
106                                 else
107                                 {
108                                         if(i>0)
109                                                 fputs("\n", tmpfile);
110                                 }
111                         }
112                 }
113                 i++;
114         }
115         fclose(file);
116         remove(path);
117         rename(tmpfilePath, path);
118         fclose(tmpfile);
119         return 1;
120 }
121
122 /*
123  Liest eine Zeile in einer Datei, angegeben durch row beginnend bei 1.
124 (Generel nuetzlich, um eine bestimmte Zeile aus einer Datei zu lesen)
125  */
126 int readLine(char *path, char *buffer, int row)
127 {
128         int buf = 255;
129         int len = 255;
130         int readsize;
131         char temp[len];
132
133         FILE *file;
```

```
134
135         file =fopen(path, "r");
136
137         if(file == NULL)
138         {
139                 printf("...Datei existiert nicht: %s...\n", path);
140                 return EXIT_FAILURE;
141         }
142         int i;
143         //bis zur n-ten Zeile lesen
144         for(i=0; i<row-1; i++)
145         {
146                 if(fgets(temp, buf, file) == NULL)
147                 {
148                         fclose(file);
149                         return EXIT_FAILURE;
150                 }
151         }
152         if(fgets(buffer, buf, file) == NULL)
153         {
154                 fclose(file);
155                 return EXIT_FAILURE;
156         }
157         //n-ten Zeile existiert
158         readsize = strlen(buffer);
159         if(buffer[readsize-1]=='\n')
160                 buffer[readsize-1]='\0';
161         else
162                 buffer[readsize]='\0';
163         fclose(file);
164
165         return EXIT_SUCCESS;
166 }
167
168 typedef void(*sighandler_t)(int);
169
170 static sighandler_t handle_signal(int sig_nr, sighandler_t signalhandler)
171 {
172         struct sigaction neu_sig, alt_sig;
173
174         neu_sig.sa_handler = signalhandler;
175         sigemptyset(&neu_sig.sa_mask);
176         neu_sig.sa_flags = SA_RESTART;
177         if(sigaction(sig_nr, &neu_sig, &alt_sig) < 0)
178                 return SIG_ERR;
179         return alt_sig.sa_handler;
180 }
181
182
183 /*
184  * Runs as a damon
185  */
186 static void init_motion_daemon(const char *log_name, int facility)
187 {
188         int i;
189         pid_t pid;
190
191         /* Elternprozess beenden*/
192         if((pid = fork()) != 0)
193                 exit(EXIT_FAILURE);
194
195         /* Kindprozess wird zum Session-FÃ¼hrer*/
196         if(setsid() < 0)
197         {
198                 printf("%s kann nicht Session-FÃ¼hrer werden!\n", log_name);
199                 exit(EXIT_FAILURE);
200         }
201         /* Signal SIGHUP ignorieren */
202         handle_signal(SIGHUP, SIG_IGN);
```

```
203         /* Oder einfach: signal(SIGHUP, SIG_IGN) ... */
204         /* Kind- Prozess terminieren */
205         if((pid = fork()) != 0)
206               exit(EXIT_FAILURE);
207
208         /* Bitmaske nicht vom Elternprozess erben*/
209         umask(0);
210
211         /*Alle geÃ¶ffneten Filedeskriptoren schlieÃŸen*/
212         for(i = sysconf(_SC_OPEN_MAX); i > 0; i--)
213               close(i);
214   }
215
216
217   //Selects and initialises the operation tasks
218   int scheduler(struct global_video_data *vd)
219   {
220         int ret;
221         if(vd->video_parameters.operate_mode > 0)
222         {
223               if(vd->video_parameters.operate_mode > 2 &&
224                  vd->video_parameters.operate_mode < 5)
225               {
226                     //set an Array for ROI positions
227                     if((vd->roi_size= getROIfromImage(vd, 3)) < 0)
228                     {
229                           //printf("No ROI Image found\n");
230                           return NO_ROI_IMAGE_FOUND;
231                     }
232                     else
233                     {
234                           //printf("ROI Image found %d\n", vd->roi_size);
235                     }
236               }
237
238               if(vd->video_parameters.stamp)
239               {
240                     vd->stamp_pos_x= 20;
241                     vd->stamp_pos_y= vd->y - 20;
242                     vd->stamp_scale_w= 0;
243                     vd->stamp_scale_h=0;
244               }
245
246               //feststellen, ob Datei existiert
247               if(access("/dev/video", F_OK) == -1)
248               {
249                     system("mknod /dev/video0 c 81 0");
250                     system("chmod 666 /dev/video0");
251                     system("ln -s /dev/video0 /dev/video");
252                     writeLine(ALARM_FILE, "make video0", 3);
253                     sleep(1);
254               }
255
256               if(access(ALARM_FILE, F_OK) != -1)
257               {
258                     remove(ALARM_FILE);
259               }
260
261               ret = device_init(vd);
262
263               if(ret == OPEN_VIDEO_DEV_FAILURE)
264               {
265                     system("insmod
266               /lib/modules/2.6.20.7/kernel/drivers/media/video/v4l1-compat.ko");
267                     system("insmod
268               /lib/modules/2.6.20.7/kernel/drivers/media/video/v4l2-common.ko");
269                     system("insmod
270               /lib/modules/2.6.20.7/kernel/drivers/media/video/videodev.ko");
271                     system("insmod
```

```
272                 /lib/modules/2.6.20.7/kernel/drivers/usb/media/gspca.ko");
273                         sleep(1);
274                         ret = device_init(vd);
275                 }
276
277
278                 if(ret == OPEN_VIDEO_DEV_SUCCESS || ret == NO_VIDIOCSWIN_FAILURE)
279                 {
280                         set_channel(vd);
281                         set_mode(vd);
282                 }
283                 else
284                 {
285                         return OPEN_VIDEO_DEV_FAILURE;
286                 }
287
288
289                 if(vd->video_parameters.showc)
290                 {
291                         #ifdef DEBUG
292                         fprintf(stderr,"\n\nvd.v_flags.showc=%d\n",
293                                 vd->video_parameters.showc);
294                         #endif
295                         show_webcam_info(vd);
296                         if(vd->video_device)
297                                 free(vd->video_device);
298                         if(vd->video_format)
299                                 free(vd->video_format);
300                         if(vd->image_fileName)
301                                 free(vd->image_fileName);
302
303                         close(vd->frame_fd);
304
305                         //int munmap(void *addr, size_t len);
306                         munmap(vd->f_grabbing.current_frame, vd->frame_size);
307                         return FALSE;
308                 }
309                 //capture picture
310                 vd->threadshold_limit= 30;
311                 vd->video_parameters.safepic = 0;
312
313                 if(vd->video_parameters.operate_mode == 1)
314                 {
315                         if((vd->f_grabbing.current_frame = malloc(vd->frame_size *
316                         sizeof(unsigned char)))!=NULL)
317                         {
318                                 grab_one_frame(vd);
319                                 adjustBrightness(vd, 140, 100, 2000);
320                                 sleep(1);
321                                 vd->video_parameters.safepic = 1;
322                                 grab_one_frame(vd);
323                                 free(vd->f_grabbing.current_frame);
324                                 vd->f_grabbing.current_frame=NULL;
325                         }
326                 }
327                 else if(vd->video_parameters.operate_mode == 2)
328                 {
329                         if((vd->f_grabbing.current_frame = malloc(vd->frame_size *
330                         sizeof(unsigned char)))!=NULL)
331                         {
332                                 char name[]="roi";
333                                 char *tmp;
334                                 tmp=vd->image_fileName;
335                                 vd->image_fileName=name;
336                                 grab_one_frame(vd);
337                                 adjustBrightness(vd, 140, 100, 2000);
338                                 sleep(1);
339                                 vd->video_parameters.safepic = 1;
340                                 grab_one_frame(vd);
```

```
341                             vd->image_fileName=tmp;
342                             free(vd->f_grabbing.current_frame);
343                             vd->f_grabbing.current_frame=NULL;
344                     }
345             }
346             else if(vd->video_parameters.operate_mode == 3)
347             {
348                     sleep(vd->video_parameters.delay);
349                     pthread_t motion_thread;
350                     pthread_t frame_thread;
351
352                     pthread_cond_init(&(vd->f_grabbing.cond_contour), NULL);
353                     pthread_cond_init(&(vd->f_grabbing.cond_image), NULL);
354                     pthread_mutex_init(&(vd->f_grabbing.mutex), NULL);
355
356                     //Fifo queue init
357                     imagefifo_init(vd);
358
359                     int index;
360
361                     for(index=0; index<4; index++)
362                     {
363                             putImagefifo(vd, &(vd->f_grabbing.dummy_image),
364                                             &(vd->f_grabbing.imf_image[index]));
365                     }
366                     //creat Threads
367                     vd->stop=0;
368                     pthread_create(&motion_thread, NULL, motion_contour, vd);
369                     pthread_create(&frame_thread, NULL, grabbingFrames, vd);
370             }
371             else if(vd->video_parameters.operate_mode == 4)
372             {
373                     sleep(vd->video_parameters.delay);
374                     pthread_t motion_thread;
375                     pthread_t frame_thread;
376                     int brightness = vd->video_parameters.roi_limit;
377                     vd->roi = malloc(vd->frame_size * sizeof(unsigned char));
378                     char name[]="roi.jpg";
379                     char *tmp;
380                     tmp=vd->image_fileName;
381                     vd->image_fileName=name;
382                     creatROIImage(vd, brightness, 0);
383                     vd->image_fileName=tmp;
384                     free(vd->roi);
385                     sleep(1);
386
387                     pthread_cond_init(&(vd->f_grabbing.cond_contour), NULL);
388                     pthread_cond_init(&(vd->f_grabbing.cond_image), NULL);
389                     pthread_mutex_init(&(vd->f_grabbing.mutex), NULL);
390
391                     //Fifo queue init
392                     imagefifo_init(vd);
393
394                     int index;
395                     for(index=0; index<4; index++)
396                     {
397                             putImagefifo(vd, &(vd->f_grabbing.dummy_image),
398                                             &(vd->f_grabbing.imf_image[index]));
399                     }
400                     //creat Threads
401                     vd->stop=0;
402                     pthread_create(&motion_thread, NULL, motion_contour, vd);
403                     pthread_create(&frame_thread, NULL, grabbingFrames, vd);
404
405             }
406             else if(vd->video_parameters.operate_mode == 5)
407             {
408                     show_webcam_info(vd);
409             }
```

```
410                 return OPEN_VIDEO_DEV_SUCCESS;
411         }
412         return IDLE_STATE;
413 }
414
415 /*
416  * To release all allocated memories
417  */
418 void free_all_memories(struct global_video_data *vd)
419 {
420         int index;
421         if(vd->video_device)
422         {
423                 free(vd->video_device);
424                 vd->video_device=NULL;
425         }
426         if(vd->video_format)
427         {
428                 free(vd->video_format);
429                 vd->video_format=0;
430         }
431         if(vd->image_fileName) {
432                 free(vd->image_fileName);
433                 vd->image_fileName=NULL;
434         }
435         if(vd->roi)
436         {
437                 free(vd->roi);
438                 vd->roi=NULL;
439         }
440         if(vd->phone_number)
441         {
442                 free(vd->phone_number);
443                 vd->phone_number=NULL;
444         }
445         if(vd->video_parameters.renew_storage)
446         {
447                 for(index=0; index <= vd->video_parameters.max_storage; index++)
448                 {
449                         free(vd->image_name[index]);
450                 }
451                 free(vd->image_name);
452                 vd->image_name=NULL;
453         }
454         for(index=0; index<4; index++)
455         {
456                 if(vd->f_grabbing.frames[index])
457                 {
458                         free(vd->f_grabbing.frames[index]);
459                         vd->f_grabbing.frames[index]=NULL;
460                 }
461         }
462         close(vd->frame_fd);
463 }
464
465 void printGVS(struct global_video_data *vd)
466 {
467         printf("Resolution: %d %d:\n", vd->x, vd->y);
468         printf("Brightnes: %d:\n", vd->video_parameters.brightness);
469         printf("MaxStorage: %d:\n", vd->video_parameters.max_storage);
470         printf("Renew: %d:\n", vd->video_parameters.renew_storage);
471         printf("Timestamp: %d:\n", vd->video_parameters.stamp);
472         printf("Videodev: %s:\n", vd->video_device);
473         printf("Videoformat: %s:\n", vd->video_format);
474         printf("Videodepth: %d:\n", vd->w);
475         printf("Delay: %d:\n", vd->video_parameters.delay);
476 }
477
478 /*
```

```
 * Register the MSQUEUE
 */
static int setup_client(key_t key, int flag)
{
   int res;
   res = msgget(key, flag);
   if(res == -1)
   {
      return -1;
   }
   return res;
}

/*
 * Funktion induces the APL to send a SMS- Message
 */
int sendSMS(char *phone, char *sms)
{
   int server_id;
   int res;
   client2server c2s;

   /* Eine Message Queues zum Server */
   server_id = setup_client(SMS_SEND_KEY, 0);
   if(server_id < 0)
   {
      return FALSE;
   }
   /* Eine Nachricht an den Server versenden */
   c2s.prioritaet = 2;
   strcpy(c2s.message, MQ_SMS);
   strcpy(c2s.sms_text, sms);
   strcpy(c2s.phone_number, phone);

   res = msgsnd(server_id, &c2s, MSG_LEN, 0);
   if(res == -1)
   {
      return FALSE;
   }

   return TRUE;
}

int main(int argc, char **argv)
{
       int sleepTime = 2;
       int index;
       int ret;
       struct global_video_data global_video;

       signal(SIGINT, _sighandler);

       if(chdir(STOREPATH)!=-1)
       {
              system("mkdir /webserver/pics/");
              chdir(STOREPATH);
       }

       //Als Daemon ausfuehren
       init_motion_daemon("Motion Detector", 1);

       global_video.secUntilAutoAdjustBri=1;
       struct timeval prev_tv;
       struct timeval prev_detected_tv;
       struct timeval *curr_tv=&global_video.tv;

       unsigned char first_run=1;
       unsigned char begin_count_detected=0;
       char commandLine[50];
```

```
548         global_video.video_parameters.sms=0;
549
550         for(index=0; index<4; index++)
551         {
552               global_video.f_grabbing.frames[index]=NULL;
553         }
554         global_video.stop=TRUE;
555         global_video.detected_number=0;
556         gettimeofday(&prev_tv, NULL);
557         gettimeofday(&prev_detected_tv, NULL);
558         gettimeofday(curr_tv, NULL);
559
560         global_video.video_parameters.operate_mode = 0;
561
562         while(global_video.video_parameters.operate_mode > -1)
563         {
564               //aus Zeile 1 lesen
565               if(readLine(CONFIG_FILE, commandLine, 1)==EXIT_SUCCESS)
566               {
567                     if(strncmp(commandLine, "Status ", 6)==0)
568                     {
569                           global_video.stop=TRUE;
570                           if(!first_run)
571                           {
572                                 sleep(2);
573                                 free_all_memories(&global_video);
574                           }
575                           first_run=0;
576
577                           extract_command(&global_video,&
578                                 (global_video.video_parameters));
579                           //printGVS(&global_video);
580                           if((ret = scheduler(&global_video)) == NO_ROI_IMAGE_FOUND)
581                           {
582                                 writeFile(ALARM_FILE, "MD stoped, because ROI Image
583                                       was not found!");
584                                 global_video.video_parameters.operate_mode = -1;
585                           }
586                           else if(ret == OPEN_VIDEO_DEV_FAILURE)
587                           {
588                                 writeFile(ALARM_FILE, "Please, try again, because a
589                                       device error has been detected!");
590                           }
591                     }
592               }
593
594               if(global_video.video_parameters.operate_mode > 2)
595               {
596                     gettimeofday(curr_tv, NULL);
597                     //ausloesen der auto_brightness Funktion
598                     if(((int) curr_tv->tv_sec) >((int) prev_tv.tv_sec) +
599                           global_video.secUntilAutoAdjustBri))
600                     {
601                           adjustBrightness(&global_video, 130, 110, 2000);
602                           gettimeofday(&prev_tv, NULL);
603                     }
604
605                     //Auswertung detektierter Anzahl innerhalb bestimmter Zeit
606                     //Anzahl und Zeit
607                     if(global_video.detected_number>0 && begin_count_detected==0)
608                     {
609                           begin_count_detected=1;
610                           prev_detected_tv.tv_sec = curr_tv->tv_sec;
611                     }
612
613                     if(((int) curr_tv->tv_sec) >((int) prev_detected_tv.tv_sec) +
614                           global_video.s_time))
615                     {
616                           begin_count_detected=0;
```

```
617                             global_video.detected_number=0;
618                     }
619
620                     if(global_video.detected_number>=global_video.s_hits &&
621                             begin_count_detected)
622                     {
623                             char alarm_message[100];
624                             char time[30];
625                             getTime(time, 0);
626
627                             sprintf(alarm_message, "Motion detected at %s and %d
628                                     images have been stored in %s.",
629                                     time, global_video.stored_image_counter, TOREPATH);
630
631                             if(global_video.video_parameters.sms)
632                             {
633                                     global_video.video_parameters.sms=0;
634                                     sendSMS(global_video.phone_number, alarm_message);
635                                     strcat(alarm_message, " Alarm- SMS was send to ");
636                                     strcat(alarm_message, global_video.phone_number);
637                                     strcat(alarm_message, ".\n");
638                             }
639                             writeFile(ALARM_FILE, alarm_message);
640                             begin_count_detected = -1;
641                             global_video.detected_number = 0;
642                     }
643             }
644             sleep(sleepTime);
645     }
646
647     //To record the amount of stored images into a file
648     if(global_video.stored_image_counter > 0){
649             char alarm_message[100];
650             sprintf(alarm_message, "MD finished and total %d images have been
651                     stored.",
652                             global_video.stored_image_counter);
653             writeLine(ALARM_FILE, alarm_message, 2);
654     }
655
656     syslog( LOG_NOTICE, "MD- Damon has finished\n");
657     closelog();
658
659     free_all_memories(&global_video);
660     return 0;
661 }
```

Motion_detector.c (Eclipse Projekt motion)

```
#include "motion_controller.h"

//Setting min and max limits of pixels for the moving object
void getMinMax(struct global_video_data *vd, float minprocent, float maxprocent)
{
	vd->min=(minprocent *(float)(vd->x*vd->y))/100;
	vd->max=(maxprocent *(float)(vd->x*vd->y))/100;
	vd->lower_min=((float)vd->min) * 0.5;
}

//Returns the current system time and date
void getTime(char *str,  int setMilli)
{
	char millisecond[3]="";
	struct timeval tv;
	time_t curtime;

	gettimeofday(&tv, NULL);
	curtime=tv.tv_sec;
	str[0]='\0';
	strftime(str, 25, "%d-%m-%H%M%S", localtime(&curtime));

	if(setMilli==1)
	{
		sprintf(millisecond, "%2d",(int) tv.tv_usec);
		millisecond[2]='\0';
		strcat(str, millisecond);
	}
}

//Returns the procedural number of the unmarked region (ROI)
int getProcentOfHotRegion(struct global_video_data *vd, unsigned char
	blackLimit)
{
	unsigned char pixel;
	unsigned char *roi=vd->roi;
	int index=0, counter=0;

	if(vd->roi!=NULL)
	{
		while(index < vd->frame_size)
		{
			pixel =(roi[index]+roi[index+1]+roi[index+2]) / 3;

			if(pixel<blackLimit)
			{
				counter++;
			}
			index+=3;
		}
		return(counter*100)/(vd->x*vd->y);
	}else
	{
		return 100;
	}
}

//Reads the ROI JPEG image
int readROIjpeg(struct global_video_data *vd)
{
	char *filename;
	char roiFilePath[30];
	int ret;
	//filename sichern
	filename = vd->image_fileName;
	sprintf(roiFilePath, "%sroi.jpg", STOREPATH);
	vd->image_fileName= roiFilePath;
```

```
68
69          //jpec Bild dekordieren
70          ret = read_JPEG_file(vd);
71          vd->image_fileName= filename;
72          return ret;
73  }
74
75  /*
76  *Erzeugt ein Char Array mit einer dynamischen Groesse und fuellt es mit roi
77  Bildpunkten- Positionswerte auf.
78  *Gibt die Grenzen der zu beachteten Bereiche an.
79  */
80  int getROIfromImage(struct global_video_data *vd, unsigned char blackLimit)
81  {
82          int *index;
83          int i, count=0, flag=0, first=0;
84          unsigned char *roi;
85          unsigned char pixel;
86          unsigned char pixel_next;
87
88          //jpec Bild dekodieren
89          if(readROIjpeg(vd))
90          {
91                  roi= vd->roi;
92                  for(i=0; i < vd->frame_size; i+=3)
93                  {
94                          pixel =((roi[i])+(roi[i+1])+(roi[i+2]) ) / 3;
95                          if(pixel<blackLimit)
96                          {
97                                  if(flag==1)
98                                  {
99                                          count++;
100                                         flag=0;
101                                         first=0;
102                                 }
103                         }
104                         else
105                         {
106                                 if(i ==(vd->frame_size-3))
107                                 {
108                                         if(first==1)
109                                         {
110                                                 count++;
111                                         }
112                                 }
113                                 else if(flag==0)
114                                 {
115                                         pixel_next =((roi[i+3])+(roi[i+4])+
116                                                 (roi[i+5]) ) / 3;
117                                         //Nachbarpixel ungleich 0
118                                         if(pixel_next>blackLimit)
119                                         {
120                                                 count++;
121                                                 flag=1;
122                                                 first=1;
123                                         }
124                                 }
125                         }
126                 }
127
128                 if(count>1)
129                 {
130                         //index =(int *)malloc(count * sizeof(int));
131                         index =(int *) calloc(count, sizeof(int));
132                         int k=0;
133                         flag=0;
134                         first=0;
135                         for(i=0; i < vd->frame_size; i+=3)
136                         {
```

```
137                             pixel =((roi[i])+(roi[i+1])+(roi[i+2]) ) / 3;
138                             if(pixel<blackLimit)
139                             {
140                                     if(flag==1)
141                                     {
142                                             index[k++]= i;
143                                             flag=0;
144                                             first=0;
145                                     }
146                             }
147                             else
148                             {
149                                     if(i ==(vd->frame_size-3))
150                                     {
151                                             if(first==1)
152                                             {
153                                                     index[k]= i;
154                                             }
155                                     }
156                                     else if(flag==0)
157                                     {
158                                             if((i+3) < vd->frame_size)
159                                             {
160                                                     pixel_next =((roi[i+3])+
161                                                       (roi[i+4])+(roi[i+5]) ) / 3;
162                                                     if(pixel_next>blackLimit)
163                                                     {
164                                                             index[k++]= i;
165                                                             flag=1;
166                                                             first=1;
167                                                     }
168                                             }
169                                     }
170                             }
171                     }
172             }
173             else
174             {
175                     //if the image is completly white
176                     index =(int *) calloc(2, sizeof(int));
177                     index[0]=0;
178                     index[1]=3;
179                     count=2;
180             }
181             vd->roi_pos = index;
182     }
183     else
184     {
185             return -1;
186     }
187     return count;
188 }
189
190 //Creates a test image with single color
191 void creatTestImageBackground(struct global_video_data *vd)
192 {
193     int index=0;
194     unsigned char farbton=255;
195     unsigned char *pic = malloc(vd->frame_size * sizeof(unsigned char));
196
197     //alle Pixel auf weiss setzen
198     while(index < vd->frame_size)
199     {
200             pic[index]=farbton;
201             pic[index+1]=farbton;
202             pic[index+2]=farbton;
203
204             index+=3;
205     }
```

```
206         vd->roi = pic;
207 }
208
209 //Creates an binary ROI image with threshold limit
210 void creatROIImage(struct global_video_data *vd, int limit, unsigned char farbton)
211 {
212         int index=0;
213         unsigned char *tmp;
214         unsigned char *pic;
215         unsigned char pixel;
216
217         pic= vd->roi;
218         //allokierter Speicher sichern
219         tmp = vd->f_grabbing.current_frame;
220         vd->f_grabbing.current_frame = pic;
221         grab_one_frame(vd);
222
223         int brightness= limit;
224         if(limit<0)
225         {
226                 brightness = abs(limit);
227                 //alle Pixel auf weiss setzen
228                 while(index < vd->frame_size)
229                 {
230                         pixel =(pic[index]+pic[index+1]+pic[index+2]) / 3;
231
232                         if(pixel > brightness)
233                         {
234                                 pic[index]=farbton;
235                                 pic[index+1]=farbton;
236                                 pic[index+2]=farbton;
237                         }
238                         index+=3;
239                 }
240         }
241         else
242         {
243                 while(index < vd->frame_size)
244                 {
245                         pixel =(pic[index]+pic[index+1]+pic[index+2]) / 3;
246                         if(pixel < brightness)
247                         {
248                                 pic[index]=farbton;
249                                 pic[index+1]=farbton;
250                                 pic[index+2]=farbton;
251                         }
252                         index+=3;
253                 }
254         }
255         //allokierter Speicher wiederherstellen
256         vd->f_grabbing.current_frame = tmp;
257         vd->image = vd->roi;
258         create_jpeg(vd);
259 }
260
261 //Counts the amount of detected pixels
262 void getROIDetectSize(struct global_video_data *vd)
263 {
264                 unsigned char *roi;
265                 unsigned char red;
266                 unsigned char green;
267                 unsigned char blue;
268
269                 roi= vd->roi;
270                 if(roi!=NULL)
271                 {
272                         int i, maxCounter=0, minCounter=0;
273                         for(i=0; i < vd->frame_size; i+=3)
274                         {
```

```
275                             red   = roi[i];
276                             green = roi[i+1];
277                             blue  = roi[i+2];
278                             //Green
279                             if(red<8 && green>242 && blue <8)
280                             {
281                                   maxCounter++;
282                             //Red
283                             }
284                             else if(red>242 && green<8 && blue<8)
285                             {
286                                   minCounter++;
287                             }
288                       }
289
290                       vd->max=maxCounter;
291                       vd->min=minCounter;
292                 }
293 }
294
295 //Setting for the default values, if the extracted ROI- data are not compatible
296 void analyseExtractedROIData(struct global_video_data *vd, int *contourDetectedSizeMin,
297 int *contourDetectedSizeMax)
298 {
299       unsigned char set_default=0;
300
301       if(vd->max < vd->min)
302       {
303             if(vd->min<15)
304             {
305                   set_default=1;
306             }
307             else
308             {
309                   vd->max =(vd->x * vd->y);
310             }
311       }
312       else if(vd->min<15)
313       {
314             set_default=1;
315       }
316
317       if(set_default)
318       {
319             vd->max=4000;
320             vd->min=500;
321
322             *contourDetectedSizeMin=100;
323             *contourDetectedSizeMax=500;
324       }
325
326 }
327
328 //Size of the contour of the smallest objekt(red)
329 int getROIContourSize(struct global_video_data *vd, unsigned char red_min,
330           unsigned char red_max, unsigned char green_min,
331           unsigned char green_max, unsigned char blue_min,
332           unsigned char blue_max)
333 {
334       unsigned char *pic_diff;
335       int searchdirection, first, found;
336       int startpoint, currentpoint=0, nextpoint, contourDetectedSize=0;
337       unsigned char *roi;
338       unsigned char cur_red;
339       unsigned char cur_green;
340       unsigned char cur_blue;
341
342       int pic_size= vd->x * vd->y;
343       int pic_diff_index;
```

```
344         int i, max=0;
345
346         roi=vd->roi;
347         if(roi!=NULL)
348         {
349                pic_diff= malloc(pic_size * sizeof(unsigned char));
350                memset(pic_diff, 0, pic_size);
351
352                for(i=0; i < vd->frame_size; i+=3)
353                {
354                       cur_red   = roi[i];
355                       cur_green = roi[i+1];
356                       cur_blue  = roi[i+2];
357
358                       if(cur_red >= red_min   &&   cur_red <= red_max   &&
359                     cur_green >= green_min && cur_green <= green_max &&
360                        cur_blue >= blue_min  &&  cur_blue <= blue_max)
361                       {
362                              pic_diff[i/3]=255;
363                       }
364                }
365
366                pic_diff_index = vd->x;
367                while(pic_diff_index < pic_size - vd->x)
368                {
369                       if(pic_diff[pic_diff_index] == 255 && hasANeighbour(vd,
370                        pic_diff, pic_diff_index) && isNotRec(vd, pic_diff,
371                         pic_diff_index))
372                       {
373                              searchdirection=6;
374                              startpoint= pic_diff_index;
375                              currentpoint=startpoint;
376                              contourDetectedSize=0;
377                              first=1;
378                              while(currentpoint != startpoint || first == 1)
379                              {
380                                     found=0;
381                                     while(found==0)
382                                     {
383                                            if(pic_diff[nextpoint = getkonturNextPos(vd,
384                                                 searchdirection-1, currentpoint)]>253)
385                                            {
386                                                   if(searchdirection==0)
387                                                          searchdirection=6;
388                                                   else
389                                                          searchdirection-=2;
390                                                   found=1;
391                                                   currentpoint= nextpoint;
392                                                   pic_diff[currentpoint]=254;
393                                                   contourDetectedSize++;
394                                            }
395                                            else if(pic_diff[nextpoint =
396                                                     getkonturNextPos(vd, searchdirection,
397                                                         currentpoint)]>253)
398                                            {
399                                                   found=1;
400                                                   currentpoint= nextpoint;
401                                                   pic_diff[currentpoint]=254;
402                                                   contourDetectedSize++;
403                                            }
404                                            else if(pic_diff[nextpoint =
405                                                     getkonturNextPos(vd, searchdirection+1,
406                                                         currentpoint)]>253)
407                                            {
408                                                   found=1;
409                                                   currentpoint= nextpoint;
410                                                   pic_diff[currentpoint]=254;
411                                                   contourDetectedSize++;
412                                            }
```

```
413                                         else
414                                         {
415                                                searchdirection=(
416                                                   searchdirection+2) % 8;
417                                         }
418                                   }
419                                   first=0;
420                             }
421                             if(contourDetectedSize>max)
422                             {
423                                   max=contourDetectedSize;
424                             }
425                       }
426                       pic_diff_index++;
427                 }//end while
428                 free(pic_diff);
429           }
430           return max;
431     }
432
433     //Testfunktion for one spezified pixel
434     void setTestPicPixel(struct global_video_data *vd, unsigned char *pic, int index)
435     {
436           unsigned char *roi_pic;
437           roi_pic = vd->roi;
438           if(roi_pic!=NULL)
439           {
440                 roi_pic[index]   = pic[index];
441                 roi_pic[index+1] = pic[index+1];
442                 roi_pic[index+2] = pic[index+2];
443           }
444     }
445
446     //Returns the next index value(search direction) of the current position (pavlidis)
447     int getkonturNextPos(struct global_video_data *vd, int direction, int index)
448     {
449           switch(direction)
450           {
451                 case 0:
452                       return index + 1;
453                       break;
454                 case 1:
455                       return index -(vd->x - 1);
456                       break;
457                 case 2:
458                       return index - vd->x;
459                       break;
460                 case 3:
461                       return index -(vd->x + 1);
462                       break;
463                 case 4:
464                       return index - 1;
465                       break;
466                 case 5:
467                       return index +(vd->x - 1);
468                       break;
469                 case 6:
470                       return index + vd->x;
471                       break;
472                 case 7:
473                       return index +(vd->x + 1);
474                       break;
475                 case -1:
476                       return index +(vd->x + 1);
477                       break;
478                 case 8:
479                       return index + 1;
480                       break;
481                 default :
```

```
482                     return -1;
483                     break;
484             }
485             return -1;
486
487     }
488
489     /*
490      * Whether the spezified pixel has a neighbor (pavlidis)
491      */
492     int hasANeighbour(struct global_video_data *vd, unsigned char *diff_pic, int index)
493     {
494             if(diff_pic[index+1]==255)
495                     return 1;
496             if(diff_pic[index+(vd->x-1)]==255)
497                     return 1;
498             if(diff_pic[index+(vd->x)]==255)
499                     return 1;
500             if(diff_pic[index+(vd->x+1)]==255)
501                     return 1;
502             return 0;
503     }
504
505     /*
506      * Whether the pixel has already been processed
507      */
508     int isNotRec(struct global_video_data *vd, unsigned char *diff_pic, int index)
509     {
510             if(diff_pic[index-1]>253)
511                     return 0;
512             if(diff_pic[index+1]==254)
513                     return 0;
514             if(diff_pic[index+(vd->x-1)]==254)
515                     return 0;
516             if(diff_pic[index+(vd->x)]==254)
517                     return 0;
518             if(diff_pic[index+(vd->x+1)]==254)
519                     return 0;
520             return 1;
521     }
522
523     //Sets borders to zero, helper for contuoralg.
524     void setBorderToZero(struct global_video_data *vd, unsigned char *pic_diff)
525     {
526             int p, j;
527             int index;
528
529             p=0;
530             j= vd->y-1;
531
532             for(index=0; index < vd->y; index++)
533             {
534                     pic_diff[j]=0;
535                     pic_diff[p]=0;
536                     j+=vd->x;
537                     p+=vd->x;
538             }
539
540             p=(vd->x * vd->y) -(vd->x - 1);
541             for(index=1; index <(vd->x-1); index++)
542             {
543                     pic_diff[index]=0;
544                     pic_diff[index+p]=0;
545             }
546     }
547
548     /*
549      * Helper for analysing of the devided parts of an image
550      */
```

```
551 int area_analyse_helper(struct global_video_data *vd)
552 {
553         int numberOfAreas=0, index, c;
554         for(index=0; index<4; index++)
555         {
556                 for(c=0; c<4; c++)
557                 {
558                         if(vd->areas[index][c] > vd->areas_detect_limits)
559                         {
560                                 numberOfAreas++;
561                                 vd->areas[index][c]=1;
562                         }
563                         else
564                         {
565                                 vd->areas[index][c]=0;
566                         }
567                 }
568         }
569
570         if(numberOfAreas<10)
571         {
572                 //horizontal lines
573                 if(vd->areas[0][0]==1 && vd->areas[0][1]==1 && vd->areas[0][2]==1 &&
574                      vd->areas[0][3]==1)
575                 {
576                       return 0;
577                 }
578                 if(vd->areas[1][0]==1 && vd->areas[1][1]==1 && vd->areas[1][2]==1 &&
579                      vd->areas[1][3]==1)
580                 {
581                       return 0;
582                 }
583                 if(vd->areas[2][0]==1 && vd->areas[2][1]==1 && vd->areas[2][2]==1 &&
584                      vd->areas[2][3]==1)
585                 {
586                       return 0;
587                 }
588                 if(vd->areas[3][0]==1 && vd->areas[3][1]==1 && vd->areas[3][2]==1 &&
589                      vd->areas[3][3]==1)
590                 {
591                       return 0;
592                 }
593                 //vertical lines
594                 if(vd->areas[0][0]==1 && vd->areas[1][0]==1 && vd->areas[2][0]==1 &&
595                      vd->areas[3][0]==1)
596                 {
597                       return 0;
598                 }
599                 if(vd->areas[0][1]==1 && vd->areas[1][1]==1 && vd->areas[2][1]==1 &&
600                      vd->areas[3][1]==1)
601                 {
602                       return 0;
603                 }
604                 if(vd->areas[0][2]==1 && vd->areas[1][2]==1 && vd->areas[2][2]==1 &&
605                      vd->areas[3][2]==1)
606                 {
607                       return 0;
608                 }
609                 if(vd->areas[0][3]==1 && vd->areas[1][3]==1 && vd->areas[2][3]==1 &&
610                      vd->areas[3][3]==1)
611                 {
612                       return 0;
613                 }
614
615                 //4 edge for camera moving
616                 if(vd->areas[0][0]==1 && vd->areas[0][3]==1 && vd->areas[3][0]==1 &&
617                      vd->areas[3][3]==1)
618                 {
619                       return 0;
```

```
620             }
621
622             if(numberOfAreas==0)
623                     return 0;
624             return 1;
625       }
626       return 0;
627 }
628
629
630 /*
631  * Analyses the devided parts of an image
632  */
633 int area_analyse(struct global_video_data *vd, unsigned char *pic_prev, unsigned char
634 *pic_next)
635 {
636       unsigned char pixel_next;
637       unsigned char pixel_prev;
638       int width= vd->x * 3, counter=1;
639
640       int index, c, area_x=0, area_y=0, y_runner=0, x_runner=0;
641
642       for(index=0; index<4; index++)
643       {
644             for(c=0; c<4; c++)
645                   vd->areas[index][c]= 0;
646       }
647
648       for(index=0; index< vd->frame_size; index += vd->areas_step)
649       {
650             pixel_prev =((pic_prev[index])+(pic_prev[index+1])+
651                         (pic_prev[index+2])) / 3;
652             pixel_next =((pic_next[index])+(pic_next[index+1])+
653                         (pic_next[index+2])) / 3;
654
655             x_runner += vd->areas_step;
656
657             if(x_runner >= vd->areas_x)
658             {
659                   x_runner= 0;
660                   if(area_x==3)
661                   {
662                         area_x=0;
663                         y_runner++;
664                         index= width*counter++;
665                   }
666                   else
667                   {
668                         area_x++;
669                   }
670             }
671
672             if(y_runner >= vd->areas_y)
673             {
674                   y_runner=0;
675                   area_y++;
676             }
677
678             if(abs(pixel_next - pixel_prev) > vd->threadshold_limit)
679             {
680                   vd->areas[area_y][area_x]++;
681             }
682       }
683       int d=0;
684       for(index=0; index<4; index++)
685       {
686             for(c=0; c<4; c++)
687                   printf("area%d: %d\n", d++, vd->areas[index][c]);
688       }
```

```
689         return area_analyse_helper(vd);
690 }
691
692 /*
693  * Inits the settings for analysing image regions
694  */
695 int area_analyse_init(struct global_video_data *vd)
696 {
697         int prozent_detection=3;
698         int prozent_analyse_size=5;
699         int analyse_size;
700         vd->areas_x=(vd->x / 4) * 3;
701         vd->areas_y= vd->y / 4;
702
703         //5% of Pixels to analyse
704         analyse_size=((vd->x * prozent_analyse_size) / 100) * vd->areas_y ;
705         vd->areas_step=(vd->areas_x * prozent_analyse_size) / 100;
706         vd->areas_step= vd->areas_step/3;
707         vd->areas_step= vd->areas_step*3;
708
709         vd->areas_detect_limits=(analyse_size * prozent_detection) / 100;
710
711         return 1;
712 }
713
714 /*
715  * Inits the fifo queue and mallocs the images memories
716  */
717 int imagefifo_init(struct global_video_data *vd)
718 {
719         int index;
720
721         for(index=0; index<4; index++)
722         {
723                 if((vd->f_grabbing.frames[index]=(unsigned char *) malloc(
724                                 vd->frame_size * sizeof(unsigned char)))==NULL)
725                         return 0;
726
727                 vd->f_grabbing.imf_image[index].isNewFrame= FALSE;
728                 vd->f_grabbing.imf_image[index].imageNumber=index;
729                 vd->f_grabbing.imf_image[index].image=
730                                                 vd->f_grabbing.frames[index];
731         }
732
733         vd->f_grabbing.dummy_image.previewsImage=NULL;
734         vd->f_grabbing.dummy_analyses.previewsImage=NULL;
735
736         vd->f_grabbing.dummy_image.imageNumber=100;
737         vd->f_grabbing.dummy_analyses.imageNumber=200;
738
739         return 1;
740 }
741
742 /*
743  * Fifo: put an element
744  */
745 int putImagefifo(struct global_video_data *vd, struct imagefifo *dummy_image,
746                         struct imagefifo *imf_New)
747 {
748         struct imagefifo *pointer;
749         //is the first element
750         if(dummy_image->previewsImage == NULL)
751         {
752                 dummy_image->previewsImage = imf_New;
753                 imf_New->previewsImage=NULL;
754         }
755         else
756         {
757                 pointer=dummy_image;
```

```
758                 //searching for the end of the queue
759                 while(pointer->previewsImage!=NULL)
760                 {
761                         pointer=pointer->previewsImage;
762                 }
763                 pointer->previewsImage = imf_New;
764                 imf_New->previewsImage = NULL;
765         }
766         /*
767         //Debug
768         if(dummy_image->imageNumber==100)
769                 printf("<<<<<<<<<Thread put Fifo: %d %d\n", imf_New->imageNumber, imf_New-
770 >isNewFrame);
771         else
772                 printf("<<<<<<<<<Contour put Fifo:%d %d\n", imf_New->imageNumber, imf_New-
773 >isNewFrame);
774         */
775         return 1;
776 }
777
778 /*
779  * Fifo: get the first element, if not exist returns NULL
780  */
781 struct imagefifo *getImagefifo(struct global_video_data *vd,
782                                         struct imagefifo *dummy_image)
783 {
784         struct imagefifo *pointer;
785         //is there something in the queue?
786         if(dummy_image->previewsImage != NULL)
787         {
788                 pointer=dummy_image->previewsImage;
789                 dummy_image->previewsImage=pointer->previewsImage;
790                 pointer->previewsImage=NULL;
791
792                 /*
793                 //Debug
794                 if(dummy_image->imageNumber==100)
795                         printf("<<<<<<<<<Thread get Fifo: %d %d\n", pointer->imageNumber,
796 pointer->isNewFrame);
797                 else
798                         printf("<<<<<<<<<Contour get Fifo: %d %d\n", pointer->imageNumber,
799 pointer->isNewFrame);
800                 */
801                 return pointer;
802         }
803         else
804         {
805                 return NULL;
806         }
807 }
808
809 /*
810  *Average of the histogram
811  */
812 float histogramAverage(struct global_video_data *vd, unsigned char *pic)
813 {
814         unsigned char pixelvalue;
815         unsigned int histogram[256];
816
817         int index;
818         int gh=0, h=0;
819
820         for(index=0; index<256; index++)
821         {
822                 histogram[index]=0;
823         }
824
825         index=0;
826         while(index < vd->frame_size)
```

```
827         {
828                 pixelvalue =(pic[index]+pic[index+1]+pic[index+2]) / 3;
829                 histogram[pixelvalue]++;
830                 index+=3;
831         }
832
833         for(index=0; index<256; index++)
834         {
835                 gh+=index*histogram[index];
836                 h+=histogram[index];
837         }
838
839         return gh/h;
840 }
841
842 /*
843  * Thread which grabbs the image frames
844  */
845 void *grabbingFrames(void *arg)
846 {
847         pthread_attr_t attr;
848         int ret;
849         struct imagefifo *imageQueueElement=NULL;
850
851         ret = pthread_attr_setschedpolicy(&attr, THREADPOLICY);
852         if(ret == 0)
853            fprintf(stderr,"Konnte Scheduling nicht auf SCHED_OTHER stellen\n");
854
855         printf("Status von mit ID %ld : \n",pthread_self());
856
857         struct global_video_data *vd=((struct global_video_data*)arg);
858
859         sleep(2);
860         while(!vd->stop)
861         {
862                 //break;
863                 if(pthread_mutex_lock(&(vd->f_grabbing.mutex)) != 0)
864                 {
865                    //printf("Fehler bei lock in Thread:%ld\n",pthread_self());
866                 }
867
868                 imageQueueElement = getImagefifo(vd,
869                                                  &(vd->f_grabbing.dummy_image));
870                 //printf("thread nach getImagefifo\n");
871
872                 if(pthread_mutex_unlock(&(vd->f_grabbing.mutex)) != 0)
873                 {
874                         //printf("Fehler bei lock in Thread:%ld\n",pthread_self());
875                 }
876
877                 if(imageQueueElement != NULL)
878                 {
879                         vd->f_grabbing.current_frame = imageQueueElement->image;
880                         grab_one_frame(vd);
881
882                         if( pthread_mutex_lock(&(vd->f_grabbing.mutex)) != 0)
883                         {
884                 //printf("Fehler bei lock in Thread:%ld\n",pthread_self());
885                         }
886                         imageQueueElement->isNewFrame=TRUE;
887                         putImagefifo(vd, &(vd->f_grabbing.dummy_analyses),
888                                         imageQueueElement);
889
890                         if(pthread_mutex_unlock(&(vd->f_grabbing.mutex)) != 0)
891                         {
892                 //printf("Fehler bei lock in Thread:%ld\n",pthread_self());
893                         }
894                 }
895                 else
```

```
896                 {
897                         pthread_cond_wait(&(vd->f_grabbing.cond_image),
898                                         &(vd->f_grabbing.mutex));
899
900                         if(pthread_mutex_unlock(&(vd->f_grabbing.mutex)) != 0)
901                         {
902                                 //printf("Fehler bei lock in Thread:%ld\n",pthread_self());
903                         }
904                 }
905                 usleep(vd->video_parameters.sys_util);
906         }//end while
907         pthread_cond_signal(&(vd->f_grabbing.cond_contour));
908         pthread_exit((void *) 0);
909 }
910
911 //Thread which does the detectiong thing
912 void *motion_contour(void *arg)
913 {
914         pthread_attr_t attr;
915         int ret;
916
917         ret = pthread_attr_setschedpolicy(&attr, THREADPOLICY);
918
919         struct global_video_data *vd=((struct global_video_data*)arg);
920         unsigned char *pic_diff_curr;
921         unsigned char *pic_diff_prev;
922         unsigned char *pic_diff_tmp;
923
924         struct imagefifo *imageQueueElementNext=NULL;
925         struct imagefifo *imageQueueElementMiddle=NULL;
926         struct imagefifo *imageQueueElementPrev=NULL;
927         struct imagefifo *imageQueueElementActual=NULL;
928
929         int *roi=NULL;
930         unsigned char pixel_next;
931         unsigned char pixel_prev;
932         unsigned char savedPrevPic=FALSE;
933
934         int begin=0, index, k;
935         int hits;
936         int roi_size;
937         int pic_size= vd->x * vd->y;
938
939         int pic_diff_index;
940         int hits_recorded=0;
941
942         int just_stored=0;
943
944         //cycles to until changing first Picture
945         unsigned char cycles=0, cyclesToChange=5;
946
947         //variables for pavlidis algo.
948         int startpoint, currentpoint=0, nextpoint, contourDetectedSize=0,
949                         contourDetectedSizeMin, contourDetectedSizeMax;
950         int searchdirection, first, found;
951
952         area_analyse_init(vd);
953
954         //set an Array for ROI positions
955         roi_size = vd->roi_size;
956
957         roi = vd->roi_pos;
958
959         //create a diff_pic
960         begin= roi[0];
961         pic_diff_curr= malloc(pic_size * sizeof(unsigned char));
962         pic_diff_prev= malloc(pic_size * sizeof(unsigned char));
963         memset(pic_diff_prev, 0, pic_size);
964         memset(pic_diff_curr, 0, pic_size);
```

```
965
966         //manuel
967         //getMinMax(vd, 2.00, 14.32);
968         //minmax(vd, 0.80, 20.00);
969         //minmax(vd, 0.85, 14.32);
970
971         getROIDetectSize(vd);
972
973         //get contourDetectedSizeMin(red)
974         contourDetectedSizeMin= getROIContourSize(vd, 250, 255, 0, 5, 0, 5);
975         //get contourDetectedSizeMax(green)
976         contourDetectedSizeMax= getROIContourSize(vd, 0, 5, 250, 255, 0, 5);
977
978         analyseExtractedROIData(vd, &contourDetectedSizeMin,
979                                 &contourDetectedSizeMax);
980
981         int roi_steps;
982         int received_frames = -1;
983         vd->video_parameters.pic_ready = FALSE;
984
985         vd->stored_image_counter = 0;
986         while(!vd->stop)
987         {
988                if(pthread_mutex_lock(&(vd->f_grabbing.mutex)) != 0)
989                {
990                   //printf("Fehler bei lock in Thread:%ld\n",pthread_self());
991                }
992
993                imageQueueElementActual = getImagefifo(vd,
994                                             &(vd->f_grabbing.dummy_analyses));
995
996                if(imageQueueElementActual!=NULL)
997                {
998                       if(received_frames<0)
999                       {
1000                             imageQueueElementPrev= imageQueueElementActual;
1001                      }
1002                      received_frames++;
1003               }
1004
1005               if(pthread_mutex_unlock(&(vd->f_grabbing.mutex)) != 0)
1006               {
1007        //printf("Fehler bei unlock in Thread:%ld\n",pthread_self());
1008               }
1009               if(imageQueueElementActual!=NULL && received_frames > 0)
1010               {
1011                      usleep(vd->video_parameters.sys_util);
1012                      if(hits_recorded > vd->min)
1013                      {
1014                             imageQueueElementMiddle = imageQueueElementNext;
1015                             imageQueueElementNext = imageQueueElementActual;
1016
1017                             pic_diff_tmp = pic_diff_prev;
1018                             pic_diff_prev= pic_diff_curr;
1019                             pic_diff_curr= pic_diff_tmp;
1020                             //analyse diff pic with contour algorithm
1021                             vd->video_parameters.pic_ready=TRUE;
1022                      }
1023                      else
1024                      {
1025                             imageQueueElementNext = imageQueueElementActual;
1026                             //do not analyse diff pic with contour algorithm
1027                             vd->video_parameters.pic_ready=FALSE;
1028                      }
1029
1030                      k=1;
1031                      hits=0;
1032                      roi_steps=0;
1033                      index=begin;
```

```
1034                    pic_diff_index=index/3;
1035
1036                    if(area_analyse(vd, imageQueueElementPrev->image,
1037                                    imageQueueElementNext->image))
1038                    {
1039                           //get Differenz Picture
1040                       while(index < vd->frame_size)
1041                       {
1042                           pixel_prev =((imageQueueElementPrev->image[index])+
1043                                  (imageQueueElementPrev->image[index+1])+
1044                                  (imageQueueElementPrev->image[index+2])) / 3;
1045                           pixel_next =((imageQueueElementNext->image[index])+
1046                                  (imageQueueElementNext->image[index+1])+
1047                                  (imageQueueElementNext->image[index+2])) / 3;
1048
1049                                  //wir kommen hier an, jeweils am Ende
1050                                  if(index == roi[k])
1051                                  {
1052                                         //next pos ist am Ende
1053                                         if(++k==roi_size)
1054                                         {
1055                                                index=vd->frame_size;
1056                                         }
1057                                         else
1058                                         {
1059                                                index = roi[k];
1060                                                pic_diff_index = index/3;
1061                                         }
1062                                         k++;
1063                                  }
1064                                  else
1065                                  {
1066                                         index+=3;
1067                                         pic_diff_index++;
1068                                  }
1069
1070                                  //Differenzbild erstellen Threasholdlimit
1071                                  if(abs(pixel_next - pixel_prev) >
1072                                     vd->threadshold_limit)
1073                                  {
1074                                         pic_diff_curr[pic_diff_index]= 255;
1075
1076                                         hits++;
1077                                  }
1078                                  else
1079                                  {
1080                                         pic_diff_curr[pic_diff_index]= 0;
1081                                  }
1082                                  roi_steps++;
1083
1084                    }
1085            //ready to analyse and store pic, means already precompared one time
1086                    if(vd->video_parameters.pic_ready)
1087                    {
1088                           if( hits_recorded > hits )
1089                           {
1090                                  hits= hits_recorded;
1091                                  hits_recorded = 0;
1092                                  pic_diff_tmp=pic_diff_prev;
1093                                  pic_diff_prev=pic_diff_curr;
1094                                  pic_diff_curr= pic_diff_tmp;
1095                           }
1096                    }
1097                    else
1098                    {
1099                           hits_recorded = hits;
1100                    }
1101
1102            //wird benoetigt, damit der Konturalg. nicht aus frame heraus greift
```

```
1103                    setBorderToZero(vd, pic_diff_curr);
1104
1105                    if(hits < vd->max && vd->video_parameters.pic_ready)
1106                    {
1107                           //Contour Alg.
1108                           pic_diff_index = vd->x;
1109                           while(pic_diff_index < pic_size - vd->x)
1110                           {
1111
1112                                  if(pic_diff_curr[pic_diff_index] == 255 &&
1113                                 hasANeighbour(vd, pic_diff_curr,
1114                                                pic_diff_index)
1115
1116              && isNotRec(vd, pic_diff_curr, pic_diff_index))
1117                                  {
1118                                         searchdirection=6;
1119                                         startpoint= pic_diff_index;
1120                                         currentpoint=startpoint;
1121                                         contourDetectedSize=0;
1122                                         first=1;
1123              while(currentpoint != startpoint ||  first == 1)
1124              {
1125                     found=0;
1126                     while(found==0)
1127                     {
1128                           if(pic_diff_curr[nextpoint = getkonturNextPos(vd,
1129                            searchdirection-1, currentpoint)]>253)
1130                           {
1131                                  if(searchdirection==0)
1132                                         searchdirection=6;
1133                                  else
1134                                         searchdirection-=2;
1135                                  found=1;
1136                                  currentpoint= nextpoint;
1137
1138                              pic_diff_curr[currentpoint]=254;
1139                                  contourDetectedSize++;
1140                           }
1141                           else if(pic_diff_curr[nextpoint = getkonturNextPos(vd,
1142                            searchdirection, currentpoint)]>253)
1143                           {
1144                                  found=1;
1145                                  currentpoint= nextpoint;
1146
1147                           pic_diff_curr[currentpoint]=254;
1148                                  contourDetectedSize++;
1149                           }
1150                           else if(pic_diff_curr[nextpoint = getkonturNextPos(vd,
1151                            searchdirection+1, currentpoint)]>253)
1152                           {
1153                                  found=1;
1154                                  currentpoint= nextpoint;
1155
1156                               pic_diff_curr[currentpoint]=254;
1157                                  contourDetectedSize++;
1158                           }
1159                           else
1160                           {
1161                                  searchdirection=(searchdirection+2) % 8;
1162                           }
1163                     }
1164                     first=0;
1165              }
1166              if(contourDetectedSize > contourDetectedSizeMin &&
1167                contourDetectedSize < contourDetectedSizeMax)
1168              {
1169                     just_stored=1;
1170                     if(!savedPrevPic)
1171                     {
```

```
1172                          vd->image= imageQueueElementPrev->image;
1173                            getTime(vd->image_fileName, 1);
1174                            create_jpeg(vd);
1175                            savedPrevPic=TRUE;
1176                        }
1177                        vd->image= imageQueueElementMiddle->image;
1178                        getTime(vd->image_fileName, 1);
1179                        create_jpeg(vd);
1180                        vd->image= imageQueueElementNext->image;
1181                        getTime(vd->image_fileName, 1);
1182                        create_jpeg(vd);
1183                        vd->detected_number++;
1184                        break;
1185                }
1186           }//end if
1187           pic_diff_index++;
1188      }//end while
1189   }//end if Contour Algorithmus
1190
1191              if(pthread_mutex_lock(&(vd->f_grabbing.mutex)) != 0)
1192               {
1193                  //printf("Fehler bei lock in Thread:%ld\n",pthread_self());
1194               }
1195               //Jedesmal, wenn etwas gespeichert wird
1196               if(just_stored)
1197               {
1198                      just_stored=0;
1199                      hits_recorded = 0;
1200                      imageQueueElementPrev->isNewFrame    = FALSE;
1201                      imageQueueElementMiddle->isNewFrame = FALSE;
1202                      imageQueueElementNext->isNewFrame    = FALSE;
1203                      putImagefifo(vd, &(vd->f_grabbing.dummy_image),
1204                              imageQueueElementPrev);
1205                      putImagefifo(vd, &(vd->f_grabbing.dummy_image),
1206                              imageQueueElementMiddle);
1207                      putImagefifo(vd, &(vd->f_grabbing.dummy_image),
1208                              imageQueueElementNext);
1209                      received_frames=-1;
1210                      cycles=0;
1211
1212               }
1213               else if(vd->video_parameters.pic_ready)
1214               {
1215                      hits_recorded=0;
1216                      imageQueueElementMiddle->isNewFrame = FALSE;
1217                      imageQueueElementNext->isNewFrame    = FALSE;
1218                      putImagefifo(vd, &(vd->f_grabbing.dummy_image),
1219                                imageQueueElementMiddle);
1220                      putImagefifo(vd, &(vd->f_grabbing.dummy_image),
1221                                imageQueueElementNext);
1222                      received_frames=0;
1223               }
1224               else
1225               {
1226                      cycles++;
1227                      if(cycles >= cyclesToChange)
1228                      { //erstes Bild neu setzen
1229                             imageQueueElementPrev->isNewFrame=FALSE;
1230                             putImagefifo(vd, &(vd->f_grabbing.dummy_image),
1231                            imageQueueElementPrev);
1232                             imageQueueElementPrev= imageQueueElementNext;
1233                             savedPrevPic=FALSE;
1234                             hits_recorded=0;
1235                             cycles=0;
1236                      }
1237                      else if(hits_recorded <= vd->min)
1238                      {
1239                             imageQueueElementNext->isNewFrame=FALSE;
1240                             putImagefifo(vd, &(vd->f_grabbing.dummy_image),
```

```
1241                                        imageQueueElementNext);
1242                        }
1243                        received_frames--;
1244                }
1245
1246                pthread_cond_signal(&(vd->f_grabbing.cond_image));
1247
1248                if(pthread_mutex_unlock(&(vd->f_grabbing.mutex)) != 0)
1249                {
1250                        //printf("Fehler bei lock in Thread:%ld\n",pthread_self());
1251                        //exit(EXIT_FAILURE);
1252                }
1253        }
1254        else
1255        {
1256                pthread_cond_signal(&(vd->f_grabbing.cond_image));
1257                usleep(vd->video_parameters.sys_util);
1258        }
1259    }//end while
1260
1261        pthread_cond_signal(&(vd->f_grabbing.cond_image));
1262        free(pic_diff_curr);
1263        free(pic_diff_prev);
        pthread_exit((void *) 0);
    }
```

Image_controller.c (Eclipse Projekt motion)

```
#include "motion_controller.h"

/*
 *  signal handler
 */
void _sighandler(int sig)
{
	switch(sig)
	{
		/* ctrl+c */
		case SIGINT:
			fprintf(stderr, "Caught SIGINT - Cleaning \n");
			exit(0);
			break;
	}
}

#ifdef HASJPEG

/*
 * Image renew storing
 */
void renewImage(struct global_video_data *vd, char *fileout)
{
	if(vd->video_parameters.renew_storage)
	{
		if(vd->stored_image_counter > vd->video_parameters.max_storage-1)
		{
			vd->stored_image_counter=0;
			vd->video_parameters.renew_storage=2;
		}
		if(vd->video_parameters.renew_storage==2)
			remove(vd->image_name[vd->stored_image_counter]);
		strcpy(vd->image_name[vd->stored_image_counter++], fileout);
	}
	else if(vd->video_parameters.operate_mode > 2)
	{
	    if(vd->stored_image_counter++ >= vd->video_parameters.max_storage-1)
	    {
			//Shutdown MD
	       vd->video_parameters.operate_mode = -1;
	       char status[100];
	        char time[30];
			getTime(time, 0);

			sprintf(status, "MD reached max storage limit at %s.",
					time);
			writeLine(CONFIG_FILE, status, 1);
		}
	}
}

/*
 * Creates an JPEG- formated image
 */
int create_jpeg(struct global_video_data *vd)
{
    char *fileout= vd->image_fileName;
	  unsigned char *img = vd->image;
	  int lx=vd->x;
	  int ly=vd->y;
	  int lw=vd->w;

	  FILE *fp;
	  unsigned char *line;  // pointer to line
	  unsigned int linesize = lx * lw, i;
```

```
67        struct jpeg_compress_struct cinfo;
68      struct jpeg_error_mgr jerr;
69
70      if(vd->video_parameters.operate_mode != -1)
71      {
72              if(vd->video_parameters.stamp)
73              {
74                      if(vd->y < 150)
75                      {
76                              //do noting
77                      }
78                      else
79                      {
80                              printStamp(vd->stamp_pos_x, vd->stamp_pos_y,
81                               vd->stamp_scale_w, vd->stamp_scale_h, vd, img,
82                                   "%s", fileout);
83                      }
84              }
85
86              strcat(fileout, ".jpg");
87              if(!strncmp(fileout, "-", 1))
88              {
89              #ifdef DEBUG
90                      fprintf(stderr,"File: stdout\n");
91              #endif
92                      fp=stdout;
93              }
94              else
95              {
96                      if((fp=fopen(fileout , "w")) == NULL)
97                      {
98                      perror("fopen");
99                              return -1;
100                     }
101             }
102
103             cinfo.err = jpeg_std_error(&jerr);
104             jpeg_create_compress(&cinfo);
105              jpeg_stdio_dest(&cinfo, fp);
106              cinfo.image_width = lx;
107              cinfo.image_height = ly;
108
109             cinfo.input_components = lw;
110             if(lw == 1)  cinfo.in_color_space = JCS_GRAYSCALE;
111             else cinfo.in_color_space = JCS_RGB;
112
113             jpeg_set_defaults(&cinfo);
114             jpeg_set_quality(&cinfo, FIXEDQUALITY, TRUE);
115             jpeg_start_compress(&cinfo, TRUE);
116
117             line=img;
118
119             for(i = 1; i <= ly; i++)
120             {
121                     jpeg_write_scanlines(&cinfo, &line, 1);
122                     line=img +(linesize * i);
123             }
124
125             jpeg_finish_compress(&(cinfo));
126             jpeg_destroy_compress(&(cinfo));
127             fclose(fp);
128             renewImage(vd, fileout);
129             return 0;
130     }
131        return vd->video_parameters.operate_mode;
132 }
133
134 /*
135  *  Error handling for JPEG
```

```
136  */
137 struct error_mgr
138 {
139       struct jpeg_error_mgr pub;
140       jmp_buf setjmp_buffer;
141 };
142
143 typedef struct error_mgr *my_error_ptr;
144
145 void my_error_exit(j_common_ptr cinfo)
146 {
147
148       my_error_ptr myerr =(my_error_ptr) cinfo->err;
149        (*cinfo->err->output_message)(cinfo);
150         longjmp(myerr->setjmp_buffer, 1);
151 }
152
153 /*
154  * Reads the JPEG- formated image into the system
155  */
156 int read_JPEG_file(struct global_video_data *vd)
157 {
158       char *filename= vd->image_fileName;
159       struct jpeg_decompress_struct cinfo;
160       struct error_mgr jerr;
161
162       FILE *infile;
163       JSAMPARRAY buffer;
164       int row_stride;
165
166       if((infile = fopen(filename, "rb")) == NULL)
167       {
168             fprintf(stderr, "Can't open: %s\n", filename);
169             return 0;
170       }
171
172       cinfo.err = jpeg_std_error(&jerr.pub);
173       jerr.pub.error_exit = my_error_exit;
174
175       if(setjmp(jerr.setjmp_buffer))
176       {
177             jpeg_destroy_decompress(&cinfo);
178             jpeg_destroy_decompress(&cinfo);
179             fclose(infile);
180             return 0;
181       }
182
183       jpeg_create_decompress(&cinfo);
184       jpeg_stdio_src(&cinfo, infile);
185       (void) jpeg_read_header(&cinfo, TRUE);
186       (void) jpeg_start_decompress(&cinfo);
187
188       int x = cinfo.output_width;
189       int y = cinfo.output_height;
190       int w = cinfo.output_components;
191       vd->x = x;
192       vd->y = y;
193       vd->w = w;
194       vd->frame_size= x * y * w;
195       row_stride = x * w;
196
197       //Debug
198       /*
199       fprintf(stderr, "Information about: %s:\n", filename);
200       fprintf(stderr, "Width : %d\n", x);
201       fprintf(stderr, "Height: %d\n", y);
202       fprintf(stderr, "Components: %d\n", w);
203       fprintf(stderr, "Size: %d\n", row_stride*cinfo.output_height);
204        */
```

```
205
206         unsigned char *pic = malloc(vd->frame_size * sizeof(unsigned char));
207
208         //Buffer for one line
209         buffer =(*cinfo.mem->alloc_sarray)((j_common_ptr)
210                     &cinfo, JPOOL_IMAGE, row_stride, 1);
211
212         int i, zahl=0;
213         //read line per line
214         while(cinfo.output_scanline < cinfo.output_height)
215         {
216                jpeg_read_scanlines(&cinfo, buffer, 1);
217
218                for(i= 0; i< row_stride; i++)
219                {
220                       pic[zahl++]= buffer[0][i];
221                }
222
223         }
224
225         vd->roi = pic;
226
227         (void) jpeg_finish_decompress(&cinfo);
228         jpeg_destroy_decompress(&cinfo);
229         fclose(infile);
230         return 1;
231  }
232  #endif
233
```

device_handler.c (Eclipse Projekt motion)

```
#include "motion_controller.h"

//wird benoetigt, um v4l API einzustellen
int device_init(struct global_video_data *vd)
{
    //Videogeraet oeffnen
       if((vd->frame_fd = open((char *) vd->video_device, O_RDWR)) == -1)
       {
              sleep(1);
              if((vd->frame_fd = open((char *) vd->video_device, O_RDWR)) == -1)
              {
                     return OPEN_VIDEO_DEV_FAILURE;
              }
       }
       if(ioctl(vd->frame_fd, VIDIOCGCAP, &(vd->video_cap)) == -1)
       {
              return WRONG_VIDEO_DEV_FAILURE;
       }

       // Informationen ueber den Kanal der Videokarte auslesen
       if(ioctl(vd->frame_fd, VIDIOCGCHAN, &(vd->video_vid))==-1)
       {
#ifdef DEBUG
              perror("init: VIDIOCGCHAN");
#endif
       }
       // default settings
       vd->video_vid.norm=VIDEO_MODE_NTSC;
    vd->video_vid.channel=0;

       // Waehlt aus, von welchem Kanal gelesen und welche Norm genommen wird
       if(ioctl(vd->frame_fd,  VIDIOCSCHAN, &(vd->video_vid))==-1)
       {
#ifdef DEBUG
              perror("init: VIDIOCGCHAN");
#endif
       }

       // setzt alle vd->grab_pic werte auf 0
       memset(&(vd->video_pic), 0, sizeof(struct video_picture));

       // fragt Bildeinstellungen ab
       if(ioctl(vd->frame_fd, VIDIOCGPICT, &(vd->video_pic)) == -1)
       {
              return NO_VIDIOCGPICT_FAILURE;
       }

       vd->frame_size = vd->x * vd->y * vd->w;

       /*************************************************************/
       /******************Memory Mapping Methode*********************/
       /*************************************************************/
       /*
       switch(vd->w)
       {
              case 1:
                     vd->grab_mmap.format = VIDEO_PALETTE_GREY;
                     break;
              case 2:
                     vd->grab_mmap.format = VIDEO_PALETTE_RGB565;
                     break;
              case 3:
              default:
                     vd->grab_mmap.format = VIDEO_PALETTE_RGB24;
                     break;
       }
```

```
67
68          vd->grab_mmap.frame  = 0;
69          vd->grab_mmap.width  = vd->x;
70          vd->grab_mmap.height = vd->y;
71
72          // Ueberprueft, ob Doublebuffering unterstuetzt wird
73          // Waehrend das Programm ausliest, kann die Cam in den naechsten Puffer
74  schreiben
75      if(ioctl(vd->frame_fd, VIDIOCGMBUF, &(vd->grab_mbuf)) < 0)
76      {
77  #ifdef DEBUG
78                  perror("VIDIOCGMBUF");
79  #endif
80                  return NO_VIDIOCGMBUF_FAILURE;
81          }
82
83          //memory mapping => mmap(addr, len, prot, flags, fildes, off)
84          mbufIndexcurrent_frame = mmap(0, vd->grab_mbuf.size,
85  PROT_READ|PROT_WRITE,MAP_SHARED, vd->frame_fd, 0);
86
87          vd->mbufIndex=0; // default frame
88          vd->grab_mbuf.offsets[vd->mbufIndex]=0;
89
90  #ifdef DEBUG
91          fprintf(stderr,"offset %d\n", vd->grab_mbuf.offsets[vd->mbufIndex]);
92  #endif
93
94          */
95
96          /*************************************************************/
97          /************************Read Methode*************************/
98          /*************************************************************/
99          if(ioctl(vd->frame_fd, VIDIOCGWIN, &(vd->video_win)) == -1)
100         {
101 #ifdef DEBUG
102                 perror("read_image: Can't get video window");
103 #endif
104         return NO_VIDIOCGWIN_FAILURE;
105         }
106
107
108         vd->video_win.width  = vd->x;
109         vd->video_win.height = vd->y;
110         //setzt Videoformat
111         if(ioctl(vd->frame_fd, VIDIOCSWIN, &(vd->video_win)) == -1)
112         {
113 #ifdef DEBUG
114         perror("read_image: Can't set video window");
115 #endif
116         return NO_VIDIOCSWIN_FAILURE;
117         }
118
119         return OPEN_VIDEO_DEV_SUCCESS;
120 }
121
122 int update_capability(struct global_video_data *vd)
123 {
124         if(ioctl(vd->frame_fd,VIDIOCGCAP,&(vd->video_cap)) == -1)
125         {
126 #ifdef DEBUG
127                 fprintf(stderr,"couldn't update capabilities.\n");
128 #endif
129                 return UPDATE_CAP_FAILURE;
130         }
131         return UPDATE_CAP_SUCCESS;
132 }
133
134
135 // shows the capabilities
```

```
136 void show_webcam_info(struct global_video_data *vd)
137 {
138         struct video_capability capability;
139         struct video_channel channel;
140         int j;
141         char infoText[500];
142         char tmp[80];
143 
144         getTime(tmp, 0);
145 
146         sprintf(infoText, "Motion_Detector Version %s\n\n"
147                     "Time       : %s\n"
148                     "Webcam Name: \n%s\n"
149                     "Max Width  : %d\n"
150                     "Max Height : %d\n"
151                     "\n"
152                     "Current Settings:\n"
153                     "Format     : %s\n"
154                     "Brightness : %i\n"
155                     "Hue        : %i\n"
156                     "Color      : %i\n"
157                     "Contrast   : %i\n"
158                     "\n"
159                     , VERSION, tmp, vd->video_cap.name, vd->video_cap.maxwidth
160                     , vd->video_cap.maxheight, vd->video_format
161                      , vd->video_pic.brightness
162                     , vd->video_pic.hue, vd->video_pic.colour,
163                      vd->video_pic.contrast);
164 
165         if(ioctl(vd->frame_fd, VIDIOCGCAP, &capability) == -1)
166         {
167               strcat(infoText, "No ioctl VIDIOCGCAP\n");
168         }
169 
170         if(!(capability.type & VID_TYPE_CAPTURE))
171         {
172               strcat(infoText, "Webcam device can't capture\n");
173         }
174 
175         fprintf(stderr, "\tInput names   : \n");
176 
177         for(j = 0; j < capability.channels; j++)
178         {
179          channel.channel = j;
180          if(ioctl(vd->frame_fd, VIDIOCGCHAN, &channel) == -1)
181          {
182             strcat(infoText, "No ioctl VIDIOCGCHAN\n");
183          }
184          else
185          {
186             sprintf(tmp, "Used channel %d: %s\n ", j, channel.name);
187             strcat(infoText, tmp);
188          }
189         }
190         writeFile(CAP_FILE, infoText);
191 }
192 
193 //grab single frame from webcam
194 int grab_one_frame(struct global_video_data *vd)
195 {
196         /*************************************************************/
197         /******************Memory Mapping Methode*********************/
198         /*************************************************************/
199 /*      // gibt den Auftrag eine Frame aus dem Geraet zu uebertragen
200         // in struct video_mmap grab_buf wird das Abbild abgespeichert
201         // Befehl haelt das Programm nicht an
202         if(-1 == ioctl(vd->frame_fd,VIDIOCMCAPTURE,&(vd->grab_mmap))) {
203                     perror("ioctl VIDIOCMCAPTURE");
204                     exit(-1);
```

```
205         }
206
207         // haelt das Programm solange an, bis die Uebertragung
208         // abgeschlossen ist
209         if(-1 == ioctl(vd->frame_fd,VIDIOCSYNC,&(vd->grab_mmap))) {
210                             perror("ioctl VIDIOCSYNC");
211                             exit(-1);
212         }
213 */
214
215         /*************************************************************/
216         /************************Read Methode*************************/
217         /*************************************************************/
218
219         if(read(vd->frame_fd, vd->f_grabbing.current_frame, vd->frame_size) == -1)
220         {
221                 perror("read_image: Error while reading");
222                 return(1);
223         }
224
225         /*************************************************************/
226         /*****************************SW Bild*************************/
227         /*************************************************************/
228 /*
229         vd->f_grabbing.current_frame = malloc((vd->x*vd->y) * sizeof(unsigned char));
230         int index=0, r, c, width= vd->x * 3;
231         for(r=0; r < vd->frame_size; r+=width)
232         {
233                 for(c=0; c < width; c+=3)
234                 {
235                         vd->f_grabbing.current_frame[index++]=((vd->image[c+r]) +(vd-
236 >image[c+1+r]) +(vd->image[c+2+r]) ) / 3;
237                         //vd->f_grabbing.current_frame[index++]= vd->image[c+r];
238
239                 }
240         }
241         vd->frame_size=(vd->x * vd->y);
242         free(vd->image);
243         //vd->image = vd->f_grabbing.current_frame;
244
245 */
246
247         //Farbkorrektur
248         set_color(vd, -25, -15, 30);
249
250         if(vd->video_parameters.safepic)
251         {
252                 vd->image=vd->f_grabbing.current_frame;
253                 create_jpeg(vd);
254          }
255          return 1;
256 }
257
258 //Bildeinstellung der Kamera vornehmen
259 int set_capability(struct global_video_data *vd)
260 {
261
262         if(ioctl(vd->frame_fd, VIDIOCGPICT, &(vd->video_pic)) == -1)
263         {
264                 perror("PICTURE");
265                 return -1;
266         }
267
268         if(vd->video_parameters.hue > -1)
269                 vd->video_pic.hue= vd->video_parameters.hue;
270          if(vd->video_parameters.contrast > -1)
271                  vd->video_pic.contrast= vd->video_parameters.contrast;
272         if(vd->video_parameters.brightness > -1)
273                  vd->video_pic.brightness= vd->video_parameters.brightness;
```

```
274         if(vd->video_parameters.colour > -1)
275                 vd->video_pic.colour= vd->video_parameters.colour;
276
277         if(ioctl(vd->frame_fd, VIDIOCSPICT, &(vd->video_pic)) == -1)
278         {
279                 perror("PICTURE");
280                 return -1;
281         }
282         return 0;
283 }
284
285 //Videokanal waehlen
286 void set_channel(struct global_video_data *vd)
287 {
288         struct video_channel l_chan;
289         l_chan.channel=vd->channel;
290
291         if(ioctl(vd->frame_fd, VIDIOCGCHAN, &l_chan) == -1)
292         {
293                 perror("02 - VIDIOCGCHAN");
294                 return;
295         }
296
297         if(ioctl(vd->frame_fd, VIDIOCSCHAN, &l_chan) == -1)
298         {
299                 perror("02 - VIDIOCSCHAN");
300                 return;
301         }
302
303         set_capability(vd);
304         set_mode(vd);
305 }
306
307 //Videomode einstellen
308 void set_mode(struct global_video_data *vd)
309 {
310     vd->video_vid.channel=vd->channel;
311
312         if(ioctl(vd->frame_fd, VIDIOCGCHAN, &(vd->video_vid)) == -1)
313         {
314                 perror("set_mode: VIDIOCGCHAN");
315                 return;
316         }
317
318         if(strncmp((char *) vd->video_format, "PAL", 3) == 0)
319         {
320                 vd->video_vid.norm = VIDEO_MODE_PAL;
321     }
322         else if(strncmp((char *) vd->video_format, "NTSC", 4) == 0)
323         {
324                 vd->video_vid.norm = VIDEO_MODE_NTSC;
325     }
326         else if(strncmp((char *) vd->video_format, "SECAM", 5) == 0)
327         {
328                 vd->video_vid.norm = VIDEO_MODE_SECAM;
329         }
330         else{
331                 fprintf(stderr,
332                         "%s is not a valid mode. Defaulting to NTSC\n", vd->video_format);
333                 vd->video_vid.norm = VIDEO_MODE_NTSC;
334                 return;
335         }
336
337         if(ioctl(vd->frame_fd, VIDIOCSCHAN, &(vd->video_vid)) == -1){
338                 perror("set_mode: VIDIOCSCHAN");
339                 return;
340         }
341 }
342
```

```
343 //Farb korrektor
344 void set_color(struct global_video_data *vd, int red_correct, int green_correct,
345                int blue_correct)
346 {
347       int r, c, red, blue, green;
348       int width= vd->x * 3;
349
350       for(r=0; r < vd->frame_size; r+=width)
351       {
352             for(c=0; c < width; c+=3)
353             {
354                   red   = vd->f_grabbing.current_frame[r+c];
355                   green = vd->f_grabbing.current_frame[r+c+1];
356                   blue  = vd->f_grabbing.current_frame[r+c+2];
357
358                   //rot
359                   if(red_correct < 0)
360                   {
361                         if(red >= -red_correct)
362                               vd->f_grabbing.current_frame[r+c]= red + red_correct;
363                         else
364                               vd->f_grabbing.current_frame[r+c]= 0;
365                   }
366                   else
367                   {
368                         if((red + red_correct) <= 255)
369                               vd->f_grabbing.current_frame[r+c]= red + red_correct;
370                         else
371                               vd->f_grabbing.current_frame[r+c]= 255;
372                   }
373
374                   //gruen
375                   if(green_correct < 0)
376                   {
377                         if(green >= -green_correct)
378                               vd->f_grabbing.current_frame[r+c+1]= green +
379                               green_correct;
380                         else
381                               vd->f_grabbing.current_frame[r+c+1]= 0;
382                   }
383                   else
384                   {
385                         if((red + red_correct) <= 255)
386                               vd->f_grabbing.current_frame[r+c+1]= green +
387                               green_correct;
388                         else
389                               vd->f_grabbing.current_frame[r+c+1]= 255;
390                   }
391
392                   //blau
393                   if(blue_correct < 0)
394                   {
395                         if(blue >= -blue_correct)
396                               vd->f_grabbing.current_frame[r+c+2] = blue +
397                               blue_correct;
398                         else
399                               vd->f_grabbing.current_frame[r+c+2]= 0;
400                   }
401                   else
402                   {
403                         if((blue + blue_correct) <= 255)
404                               vd->f_grabbing.current_frame[r+c+2] = blue +
405                                blue_correct;
406                         else
407                               vd->f_grabbing.current_frame[r+c+2] = 255;
408                   }
409             }
410       }
411 }
```

```
412
413 //Stampfunktion
414 void printStamp(int pos_x, int pos_y, int font_height, int font_width, struct
415 global_video_data *vd, unsigned char *image, const char *str, ...)
416 {
417       va_list ap;
418       char temp[128];
419       unsigned char *ptr;
420       int i, j, n, len, fnt, c, lines=0, offset=0;
421
422       //default
423       if(font_height == 0) font_height=1;
424       if(font_width  == 0) font_width =1;
425
426       if((vd->y - pos_y) < font_height * 8)
427       {
428             fprintf(stderr, "printStamp(): pos_y out of range!\n");
429             return;
430       }
431
432       va_start(ap, str);
433    memset(temp, 0, sizeof(temp));
434    vsnprintf(temp, sizeof(temp), str, ap);
435    va_end(ap);
436
437    pos_x = pos_x % vd->x;
438    pos_y = pos_y % vd->y;
439    len = strlen(temp);
440
441    for(i = 0; i < 8 ; i++)
442    {
443          //lines=0;
444          offset=(vd->w *(vd->x  *(pos_y + i )) + pos_x);
445          for(c=0; c < font_height ; c++)
446          {
447                   ptr = image + offset + lines;
448
449                   if(image + vd->frame_size < ptr)
450                   {
451                         return;
452                   }
453
454                   for(j = 0; j < len; j++)
455                   {
456                         fnt = font_database[(temp[j] * 8) + i];
457                         for(n = 7; n >= 0; n--) //
458                         {
459                               if(fnt &(1 << n))
460                               {
461                                     ptr[0]=0;
462                                     ptr[1]=0;
463                                     ptr[2]=0;
464                               }
465                               ptr += vd->w  + font_width;
466                         } //n
467                   } //j
468                  lines+=vd->x * vd->w ;
469          } //c
470    } //i
471 }
472
473 //0-65000 Stufen
474 void adjustBrightness(struct global_video_data *vd,
475              unsigned char hAverageMax, unsigned char hAverageMin, int step)
476 {
477       int index=0;
478       float hAverage=0.0;
479
480       if(!vd->stop && vd->video_parameters.auto_brightness)
```

```
481         {
482                 if(pthread_mutex_lock(&(vd->f_grabbing.mutex)) != 0)
483                 {
484                         //printf("Fehler bei lock in Thread:%ld\n", pthread_self());
485                         //exit(EXIT_FAILURE);
486                 }
487
488                 while(!vd->f_grabbing.imf_image[index].isNewFrame && index < 4)
489                 {
490                         index++;
491                 }
492
493                 if(vd->f_grabbing.imf_image[index].image!=NULL)
494                 {
495                         if(!vd->f_grabbing.imf_image[index].isNewFrame)
496                         {
497                                 vd->f_grabbing.current_frame =
498                                                 vd->f_grabbing.imf_image[index].image;
499                                 grab_one_frame(vd);
500                         }
501
502                         hAverage = histogramAverage(vd,
503                                                 vd->f_grabbing.imf_image[index].image);
504
505                         if(pthread_mutex_unlock(&(vd->f_grabbing.mutex)) != 0)
506                         {
507                                 //printf("Fehler bei lock in Thread:%ld\n", pthread_self());
508                                 //exit(EXIT_FAILURE);
509                         }
510                         if(hAverage > hAverageMax && vd->video_parameters.brightness
511                               > step)
512                         {
513                                 vd->secUntilAutoAdjustBri=1;
514                                 vd->video_parameters.brightness-= step;
515                                 set_channel(vd);
516                         }
517                         else if(hAverage < hAverageMin && vd->video_parameters.brightness
518                               <(BRIGHTNESS_MAX-step))
519                         {
520                                 vd->secUntilAutoAdjustBri=1;
521                                 vd->video_parameters.brightness+= step;
522                                 set_channel(vd);
523                         }
524                         else
525                         {
526                                 vd->secUntilAutoAdjustBri=10;
527                         }
528                 }
529                 else
530                 {
531                         if(pthread_mutex_unlock(&(vd->f_grabbing.mutex)) != 0)
532                         {
533                                 //printf("Fehler bei lock in Thread:%ld\n", pthread_self());
534                                 //exit(EXIT_FAILURE);
535                         }
536                 }
537         }
538         else if(vd->f_grabbing.current_frame!=NULL &&
539                   vd->video_parameters.auto_brightness)
540         {
541                 hAverage = histogramAverage(vd, vd->f_grabbing.current_frame);
542                 //printf("adjustBrightness hAverage %f\n", hAverage);
543                 //printf("vor adjustBrightness %lf\n", hAverage);
544
545                 if(hAverage > hAverageMax || hAverage < hAverageMin)
546                 {
547                         vd->video_parameters.brightness =
548                                     BRIGHTNESS_MAX /(hAverage/255 + 1);
549                         set_channel(vd);
```

```
550                 }
551                 //printf("adjustBrightness brightness %d\n", vd->v_flags.brightness);
552                 //printf("nach adjustBrightness %lf\n", hAverage);
553         }
554 }
```

command_handler.c (Eclipse Projekt motion)

```
#include "motion_controller.h"
/*
 *Extracts the motion.conf file and sets up the variables
 */
int extract_command(struct global_video_data *vd, struct global_parameter *parameters)
{
	char commandLine[50];
	char *token;
	char filename[100];
    char status[100];
    char strTemp[50];
    int fileNameLength=35;

	vd->video_device=NULL;
	vd->image_fileName=NULL;
	vd->video_format=NULL;
	vd->image_name=NULL;
	vd->phone_number=NULL;
	vd->x = vd->y = vd->w=0;
	vd->stamp_pos_x=vd->stamp_pos_y=vd->stamp_scale_w=vd->stamp_scale_h=0;

	vd->channel=0;
	vd->stored_image_counter=0;

	parameters->hue= parameters->contrast=
				parameters->brightness=parameters->colour=-1;
	parameters->showc=0;
	parameters->operate_mode=0;
	parameters->sms=0;
	parameters->sys_util=300;

	vd->image_fileName=(char *)malloc((fileNameLength * sizeof(char))+1);
	memset(vd->image_fileName, 0, fileNameLength);
	int i=1, tokenLength;

	while(i<=11)
	{
		if(readLine(CONFIG_FILE, commandLine, i++) == EXIT_SUCCESS)
		{
			token = strtok(commandLine, " ");
			if(token!=NULL)
			{
				if(strncmp(token, "Resolution", 10) == 0)
				{
					token = strtok(NULL, ": ");
					vd->x=atoi(token);
					token = strtok(NULL, ": ");
					vd->y=atoi(token);
				}
				else if(strncmp(token, "Brightnes", 9) == 0)
				{
					token = strtok(NULL, ": ");
					if(strncmp(token, "Auto", 4) == 0)
					{
						parameters->auto_brightness = 1;
						parameters->brightness=20000;
					}
					else
					{
						parameters->auto_brightness = 0;
						parameters->brightness=atoi(token);
					}
				}
				else if(strncmp(token, "MaxStorage", 10) == 0)
				{
					token = strtok(NULL, ": ");
```

```
67                              parameters->max_storage=atoi(token);
68                      }
69                      else if(strncmp(token, "Renew", 5) == 0)
70                      {
71                              token = strtok(NULL, ": ");
72                              if(strncmp(token, "true", 4) == 0)
73                              {
74                                      parameters->renew_storage = 1;
75                                      if(!vd->image_name)
76                                      {
77                                              int i, j;
78                                              if(parameters->max_storage>0 &&
79                                          parameters->max_storage < MAXIMAGESTORAGE)
80                                              {
81                                                      vd->image_name=
82                        (char**)(calloc(parameters->max_storage, sizeof(char*)));
83                                                      for(i=0; i <=
84                                                        parameters->max_storage; i++)
85                                                      {
86                                                              vd->image_name[i]=
87                                              (char *) malloc((fileNameLength * sizeof(char)));
88                                                              if(vd->image_name[i]==NULL)
89                                                              {
90                                                                for(j=i; j>=0; j--)
91                                                                {
92                                                                  free(vd->image_name[j]);
93                                                                }
94                                                                 free(vd->image_name);
95                                                      //Fehler Renew nicht ausfuehrbar
96                                                         parameters->renew_storage = 0;
97                                                                      break;
98                                                              }
99                                                      }
100                                             }
101                                     }
102                             }
103                             else
104                                     parameters->renew_storage = 0;
105                     }
106                     else if(strncmp(token, "Timestamp",9) == 0)
107                     {
108                             token = strtok(NULL, ": ");
109                             if(strncmp(token, "true", 4)==0)
110                                     parameters->stamp = 1;
111                             else
112                                     parameters->stamp = 0;
113
114                     }
115                     else if(strncmp(token, "Videomod", 8) == 0)
116                     {
117                             if(!vd->video_format)
118                             {
119                                     token = strtok(NULL, ": ");
120                                     tokenLength = strcspn(token, "\n");
121                                     vd->video_format=
122                          (unsigned char *)malloc((tokenLength * sizeof(char))+1);
123                                     memset(vd->video_format, 0, tokenLength);
124                                     strcpy((char *) vd->video_format, token);
125                             }
126                     }
127                     else if(strncmp(token, "Videodepth", 9) == 0)
128                     {
129                             token = strtok(NULL, ": ");
130                             vd->w = atoi(token);
131                     }
132                     else if(strncmp(token, "Videodev", 8) == 0)
133                     {
134                             token = strtok(NULL, ": ");
135                             if(!vd->video_device)
```

```
136                             {
137                                     tokenLength = strcspn(token, "\n");
138                                     vd->video_device=
139                      (unsigned char *)malloc((tokenLength * sizeof(char))+1);
140                                     if(!vd->video_device)
141                                     {
142                                             perror("malloc v_device");
143                                     }
144                                     memset(vd->video_device, 0, tokenLength);
145                                     strcpy((char *) vd->video_device, token);
146                             }
147                             token[0]='\0';
148                     }
149                     else if(strncmp(token, "Delay", 5) == 0)
150                     {
151                             token = strtok(NULL, ": ");
152                             parameters->delay = atoi(token);
153                     }
154                     else if(strncmp(token, "SysUtil", 7) == 0)
155                     {
156                             token = strtok(NULL, ": ");
157                             parameters->sys_util =((100 - atoi(token))*50);
158                     }
159                     else if(strncmp(token, "S_hit", 5) == 0)
160                     {
161                             token = strtok(NULL, ": ");
162                             vd->s_hits = atoi(token);
163                     }
164                     else if(strncmp(token, "S_time", 6) == 0)
165                     {
166                             token = strtok(NULL, ": ");
167                             vd->s_time = atoi(token);
168                     }
169                     else if(strncmp(token, "Phone", 5) == 0)
170                     {
171                             token = strtok(NULL, ": ");
172                             if(!vd->phone_number)
173                             {
174                                     tokenLength = strcspn(token, "\n");
175                                     vd->phone_number=
176                              (char *) malloc((tokenLength * sizeof(char))+1);
177                                     parameters->sms=1;
178                                     strcpy(vd->phone_number, token);
179                             }
180                     }
181                     else if(strncmp(token, "Status", 4) == 0)
182                     {
183                             token = strtok(NULL, ": ");
184                             if(strncmp(token, "Shot", 4) == 0)
185                             {
186                                     parameters->operate_mode = 1;
187                                     //Zeile 1 neu schreiben
188                                     getTime(strTemp, 0);
189                                     sprintf(status, "Shot @ %s.", strTemp);
190
191                                     writeLine(CONFIG_FILE, status, 1);
192                                     sprintf(filename, "Shot_%s", strTemp);
193                             }
194                             else if(strncmp(token, "ROI", 3) == 0)
195                             {
196                                     parameters->operate_mode = 2;
197                                     //Zeile 1 neu schreiben
198                                     getTime(strTemp, 0);
199                                     sprintf(status, "ROI @ %s.", strTemp);
200
201                                     writeLine(CONFIG_FILE, status, 1);
202                                     strcat(filename, "roi");
203                             }
204                             else if(strncmp(token, "Start", 5) == 0)
```

```
205                                     {
206                                             parameters->operate_mode = 3;
207                                             //Zeile 1 neu schreiben
208                                             getTime(strTemp, 0);
209                                             sprintf(status, "MD started @ %s.", strTemp);
210
211                                             writeLine(CONFIG_FILE, status, 1);
212                                     }
213                                     else if(strncmp(token, "Stop", 4) == 0)
214                                     {
215                                             parameters->operate_mode = 0;
216                                             //Zeile 1 neu schreiben
217                                             getTime(strTemp, 0);
218                                             sprintf(status, "Stoped @ %s.", strTemp);
219
220                                             writeLine(CONFIG_FILE, status, 1);
221                                     }
222                                     else if(strncmp(token, "Shutdown", 8) == 0)
223                                     {
224                                             parameters->operate_mode = -1;
225                                             //Zeile 1 neu schreiben
226                                             getTime(strTemp, 0);
227                                             sprintf(status, "Shutdown @ %s.", strTemp);
228
229                                             writeLine(CONFIG_FILE, status, 1);
230                                     }
231                                     else if(strncmp(token, "Showcap", 4) == 0)
232                                     {
233                                             parameters->operate_mode = 5;
234                                             getTime(strTemp, 0);
235                                             sprintf(status, "Cap Info @ %s.", strTemp);
236                                             writeLine(CONFIG_FILE, status, 1);
237                                     }
238                             }
239                     }//end if
240             }//end if
241     }//end while
242
243     strcpy(vd->image_fileName, filename);
244
245     //default settings
246     char *strtmp;
247     int strtmpLen;
248
249     if(!vd->video_device)
250     {
251             strtmp = "/dev/video0";
252             strtmpLen= strlen(strtmp);
253             vd->video_device=(unsigned char *)malloc((strtmpLen * sizeof(char))+1);
254             if(!vd->video_device)
255             {
256                     perror("malloc v_device");
257                     exit(1);
258             }
259             memset(vd->video_device, 0, strtmpLen);
260             strcpy((char *) vd->video_device, strtmp);
261     }
262
263     if(!vd->video_format)
264     {
265             strtmp = "NTSC";
266             strtmpLen= strlen(strtmp);
267             vd->video_format=(unsigned char *)malloc((strtmpLen * sizeof(char))+1);
268             memset(vd->video_format, 0, strtmpLen);
269             strcpy((char *) vd->video_format, strtmp);
270     }
271
272     if(vd->w==0)
273     {
```

```
274                 vd->w=3;
275         }
276         return 1;
277 }
```

motion_controller.h (Eclipse Projekt motion)

```
#include <stdio.h>
#include <unistd.h>
#include <syslog.h>
#include <stdlib.h>
#include <string.h>
#include <asm/types.h>
#include <sys/mman.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/msg.h>
#include <sys/time.h>
#include <fcntl.h>
#include <signal.h>
#include <errno.h>
#include <termios.h>
#include <math.h>
#include <malloc.h>
#include <stdarg.h>
#include <pthread.h>
#include <linux/videodev.h>
#include <setjmp.h>

#ifdef HASJPEG
#include <jpeglib.h>
#endif

#include "md_msg.h"
#include "font_8x8.h"


#define VIDEO_FORMAT_RAW  1
#define VIDEO_FORMAT_PNM  2
#define VIDEO_FORMAT_JPG  3
#define VIDEO_FORMAT_TGA_24      4
#define VIDEO_FORMAT_PNG  5

#define MAXLEN 255
#define DOT fprintf(stderr,".");

#define OPEN_VIDEO_DEV_SUCCESS  1
#define OPEN_VIDEO_DEV_FAILURE  2
#define WRONG_VIDEO_DEV_FAILURE 3
#define NO_VIDIOCGPICT_FAILURE  4
#define NO_VIDIOCGMBUF_FAILURE  5
#define NO_VIDIOCGWIN_FAILURE   6
#define NO_VIDIOCSWIN_FAILURE   7
#define NO_ROI_IMAGE_FOUND      8
#define UPDATE_CAP_SUCCESS      9
#define UPDATE_CAP_FAILURE      10
#define UNKNOWN_FAILURE         11
#define IDLE_STATE              12

#define MAXIMAGESTORAGE 500

#define THREADPOLICY SCHED_FIFO

#ifndef CONFIG_FILE
#define CONFIG_FILE "/motion/motion.conf"
#endif

#ifndef ALARM_FILE
#define ALARM_FILE  "/motion/alarm.txt"
#endif

#ifndef CAP_FILE
```

```
67  #define CAP_FILE    "/motion/cap.txt"
68  #endif
69
70  #ifndef STOREPATH
71  #define STOREPATH   "/webserver/pics/"
72  #endif
73
74  #define BRIGHTNESS_MAX 65000
75
76  struct imagefifo
77  {
78        unsigned char *image;
79        unsigned char isNewFrame;
80        unsigned char imageNumber;
81        struct imagefifo *previewsImage;
82  };
83
84  struct framegrabbing{
85        unsigned char*current_frame;
86        unsigned char*prev_frame;
87        unsigned char*frames[4];
88        struct imagefifo dummy_image;
89        struct imagefifo imf_image[4];
90        struct imagefifo dummy_analyses;
91        //Concurrency
92        pthread_mutex_t mutex;
93        pthread_cond_t cond_contour;
94        pthread_cond_t cond_image;
95  };
96
97  struct global_parameter {
98        int contrast, brightness, colour, hue, showc, operate_mode, safepic,
99              pic_ready, roi_limit, max_storage ,renew_storage, stamp, auto_brightness,
100 delay, sys_util, sms;
101 };
102
103 struct global_video_data
104 {
105       //V4L
106     struct video_picture      video_pic;
107     struct video_capability   video_cap;
108     struct video_tuner        video_tun;
109     struct video_mmap         video_mmap;
110     struct video_mbuf         video_mbuf;
111        struct video_channel        video_vid;
112        struct video_window       video_win;
113
114        unsigned char *video_device;
115        unsigned char *video_format;
116
117        int          frame_fd;
118        int    frame_size;
119        int    x, y, w, channel, mbufIndex;
120
121        struct timeval tv;
122
123        unsigned charstop;
124
125        //ROI
126        void   *roi;
127        int          *roi_pos;
128        int          roi_size;
129
130
131        //min max borders of pixels
132        int min, max, lower_min;
133
134        //Auto brightness control
135        unsigned char threadshold_limit;
```

```
136         unsigned char secUntilAutoAdjustBri;
137
138         //Storing control
139         int detected_number;
140         int stored_image_counter;
141         int s_time;
142         int s_hits;
143
144         //Image segmentation Control
145         int areas[4][4];
146         int areas_detect_limits;
147         int areas_x, areas_y, areas_step;
148
149         char **image_name;
150         char *image_fileName;
151         char *phone_number;
152
153         //Stamp control
154         unsigned int stamp_pos_x, stamp_pos_y, stamp_scale_w, stamp_scale_h;
155
156         unsigned char *image;
157
158         struct global_parameter   video_parameters;
159         struct framegrabbing f_grabbing;
160 };
161
162 void _sighandler(int sig);
163
164 #ifdef HASJPEG
165 #define FIXEDQUALITY      30     // JPEG Quality
166 int create_jpeg(struct global_video_data *vd);
167 int read_JPEG_file(struct global_video_data *vd);
168 #endif
169
170 void printStamp(int x, int y, int scale_h, int scale_w,
171              struct global_video_data *vd, unsigned char *image, const char *fmt, ...);
172 void set_channel(struct global_video_data *);
173 void set_mode();
174 void show_webcam_info();
175 void set_color(struct global_video_data *vd, int red_correct, int green_correct,
176                         int blue_correct);
177 int set_capability();
178 int device_init();
179 int update_capability();
180 int grab_one_frame(struct global_video_data *);
181
182
183 ssize_t getline(char **lineptr, size_t *n, FILE *stream);
184 int loadJPEGImage(struct global_video_data *vd);
185 int getROIofImage_All(struct global_video_data *vd);
186 int getROIBorder(int index);
187 int compare_pics(struct global_video_data *vd, int modus, unsigned char limit);
188 void *motion_contour(void *arg);
189 void *grabbingFrames(void *arg);
190 void creatROIImage(struct global_video_data *vd, int limit, unsigned char farbton);
191 int getProcentOfHotRegion(struct global_video_data *vd, unsigned char blackLimit);
192 void adjustBrightness(struct global_video_data *vd,
193 unsigned char hAverageMax, unsigned char hAverageMin, int step);
194
195 int readROIjpeg(struct global_video_data *vd);
196 int hasANeighbour(struct global_video_data *vd, unsigned char *diff_pic, int index);
197 int isNotRec(struct global_video_data *vd, unsigned char *diff_pic, int index);
198 int getkonturNextPos(struct global_video_data *vd, int direction, int index);
199 struct imagefifo *getImagefifo(struct global_video_data *vd,
200                                struct imagefifo *dummy_image);
201 int putImagefifo(struct global_video_data *vd, struct imagefifo *dummy_image,
202                     struct imagefifo *imf_New);
203 int imagefifo_init(struct global_video_data *vd);
204 int writeLine(char *path, char *buffer, int row);
```

```
205 int readLine(char *path, char *buffer, int row);
206 float histogramAverage(struct global_video_data *vd, unsigned char *pic);
207 int setNameOfTime(struct global_video_data *vd);
208 int extract_command(struct global_video_data *vd, struct global_parameter *v_flags);
209 void getTime(char *str, int setMilli);
210 int getROIfromImage(struct global_video_data *vd, unsigned char blackLimit);
211 int writeFile(char *path, char *buffer);
```

md_msg.h (Eclipse Projekt motion)

```
1  #ifndef MD_MSG_H_
2  #define MD_MSG_H_
3  /* msq_header.h */
4  #include <stdlib.h>
5  #include <signal.h>
6  /* Magische Nummer */
7  #define SMS_SEND_KEY 1234L
8  /* Begrenzung der Nachricht */
9  #define MSGHEADER_LEN 50
10 #define SMS_LEN 512
11 #define PHONE_LEN 24
12 #define MSG_LEN 586
13 #define MQ_SMS "MD_SMS"
14
15 /* Zugriffsrechte */
16 #define PERM 0666
17
18 /* Datentypen zum senden und empfangen der Nachrichten */
19 typedef struct {
20    long prioritaet;
21    char message[MSGHEADER_LEN];
22    char sms_text[SMS_LEN];
23    char phone_number[PHONE_LEN];
24 } client2server;
25
26 typedef struct {
27    long prioritaet;
28    char message[MSG_LEN];
29 } server2client;
30
31 #endif /*MD_MSG_H_*/
```

MDCGIHandler.c (Eclipse Projekt motion_cgi)

```c
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <unistd.h>
#include <time.h>
#include <fcntl.h>

#define MAX_PAARE 255
//want to check file extensions? 1=yes 0=no
#define CHECKEXT 1
//define max size of uploads to 100k
#define MAXSIZE 100000
// buffer for storing boundary line and file name
#define MAXLINE 512
char szBoundary[MAXLINE];
// filename to write to
char szFile[MAXLINE];
// basename of uploaded file
char myFile[MAXLINE];
// buffer of arbitrary size for reading from STDIN
#define BUFFSIZE 16*1024

#define FAILURE_PROCESSMESSAGE "Failure: Could not verify, wheather the MD process is
running\n"

char szBuff[BUFFSIZE];

void print_location(char *);
char *getdata();
char *Strdup(const char *);
void hex2ascii(char *);
char convert(char *);
void printf_error(char *);
void printStatus(char *confFilePath, char *alarmFilePath);

void getFilePath(char *buffer, char *path, char *fileName);
int deleteAllImages();
int deleteImage(char *imageName);
void scanImages();
int isJpegFile(char *str);
int writeLine(char *path, char *buffer, int row);
int readLine(char *path, char *buffer, int row);
int md_pid();
void printCap(char *mdCapDir);

char imageDir[]= "/webserver/pics/";
char mdStatusDir[]= "/motion/";
char mdStatusFileName[]= "motion.conf";
char mdAlarmFileName[]="alarm.txt";
char mdCapDir[]="/motion/cap.txt";

struct CGI_DATEN *creatPairs(char *);

struct CGI_DATEN
{
	char *variable;
	char *wert;
	struct CGI_DATEN *next;
};

struct CGI_DATEN *ende= NULL;

/*************************************************************************
```

```
 67  For file upload, CGI variables look like this:
 68  CONTENT_TYPE=multipart/form-data;
 69  boundary=-----------14422580032340
 70  CONTENT_LENGTH=216
 71
 72  STDIN contains the following information:
 73  --------------------14422580032340
 74  Content-Disposition: form-data;
 75  name="file";
 76  filename="/roi.jpg"
 77  Content-Type: text/plain
 78
 79  Data from file goes here
 80
 81  --------------------14422580032340--
 82
 83  ******************************************************************************/
 84
 85 int getext(const char *filename)
 86 {
 87        char *p, *q;
 88        int len, i, isgood = 0;
 89        char *goodext[9] = { "gif", "jpg", "jpeg", "png", "bmp", "tgz", "gz",
 90                        "txt", " tar.gz" };
 91        char ext[7]; //6 chars plus '\0'
 92
 93        if(!(p = strchr(filename, '.')))
 94              return 0;
 95        /*move to first dot */
 96        ++p;
 97        /* move past the dot */
 98        if(!(q = strchr(p, '\0')))
 99              return 0;
100        /*move to end of string */
101        len = q - p;
102        memcpy(ext, p, len);
103        ext[len] = '\0';
104        for(i=0; i<=9; i++)
105        {
106              if(strcmp(ext, goodext[i]) == 0)
107              {
108                    isgood = 1;
109              }
110        }
111        return isgood;
112 }
113
114 void creatFile()
115 {
116        char *s, *post, *tmp;
117        int postlen=0;
118        s=getenv("CONTENT_LENGTH");
119
120        // amount of posted characters
121        if(s!=NULL)
122              postlen=atoi(s);
123
124        // read postlen of characters from stdin post
125        if(postlen>0)
126        {
127              post = malloc((postlen+1)*sizeof(char));
128              tmp  = malloc((postlen+1)*sizeof(char));
129              strcpy(post, "");
130              while(postlen>0)
131              {
132                    strcpy(tmp, "");
133                    fgets(tmp, postlen+1, stdin);
134                    postlen -= strlen(tmp)+1; // man beachte die +1 !!!
135                    strcat(post, tmp);
```

```
136                 }
137         }
138 }
139
140 /*
141  * Uploads the ROI.jpeg file, which was created from the Applet
142  */
143 int fileUpload()
144 {
145         int rc = 0;
146         char *psz1;
147         char *psz2;
148         FILE *out= NULL;
149         long i, total, count;
150         char *ContentLength;
151         /* Pointer to CONTENT_LENGTH environment variable. */
152         long InCount;
153         /* The supposed number of incoming bytes. */
154         char *filename;
155
156         ContentLength = getenv("CONTENT_LENGTH");
157         InCount = atol(ContentLength);
158
159         if(InCount > MAXSIZE)
160         {
161                 rc=7;
162                 goto Error;
163         }
164
165         // null out file name buffer
166         memset(szFile, 0, sizeof(szFile));
167
168         /*----- first line should be MIME boundary, prepend cr/lf ----*/
169         szBoundary[0] = '\r';
170         szBoundary[1] = '\n';
171         szBoundary[2] = '\0';
172
173         if(fgets(&szBoundary[2], sizeof(szBoundary)-2, stdin) == NULL)
174         {
175                 rc = 1;
176                 goto Error;
177         }
178
179         //strip terminating CR / LF
180         if((psz1=strchr(&szBoundary[2],'\r')) != NULL)
181         {
182                 *psz1 = '\0';
183         }
184
185         if((psz1=strchr(&szBoundary[2],'\n')) != NULL)
186         {
187                 *psz1 = '\0';
188         }
189
190         /*-----  second line should contain "Content-Disposition: ----*/
191         if(fgets(szBuff, sizeof(szBuff), stdin) == NULL)
192         {
193                 rc = 2;
194                 goto Error;
195         }
196
197         // get filename keyword
198         if((psz1=strstr(szBuff, "filename=")) == NULL)
199         {
200                 rc = 3;
201                 goto Error;
202         }
203
204         // get pointer to actual filename(it's in quotes)
```

```
205         psz1+=strlen("filename=");
206
207         if((psz1 = strtok(psz1, " \"")) == NULL)
208         {
209               rc = 4;
210               goto Error;
211         }
212
213         // remove leading path for both PC and UNIX systems
214         if((psz2 = strrchr(psz1,'\\')) != NULL)
215         {
216               psz1 = psz2+1;
217         }
218
219         if((psz2 = strrchr(psz1,'/')) != NULL)
220         {
221               psz1 = psz2+1;
222         }
223
224         //psz1 now points to a file name, try to create it in our system
225         sprintf(szFile, "%s%s", imageDir, psz1);
226         //file to actually write
227         sprintf(myFile, "%s", psz1);
228         //basename of file
229         filename= myFile;
230
231         //check for valid extension
232         if(CHECKEXT)
233         {
234               if((getext(filename))== 0)
235               {
236                     rc = 8;
237                     //goto Error;
238               }
239         }
240
241         /*     // determine if file exists, and don't allow overwritting
242         if((out = fopen(szFile, "rb")) != NULL){
243         // file already exists!
244
245         rc = 5;
246         goto Error;
247         }
248         */
249         if((out = fopen(szFile, "w+")) == NULL)
250         {
251               rc = 6;
252               goto Error;
253         }
254
255         /*----- throw away until we get a blank line ----*/
256         while(fgets(szBuff, sizeof(szBuff), stdin)!= NULL)
257         {
258               if(strlen(szBuff) <= 2) {
259                     break;
260               }
261         }
262
263         /*----- copy the file ----*/
264         while((count=fread(szBuff, 1, sizeof(szBuff), stdin)) != 0)
265         {
266               if(count!=0)
267               if((i=fwrite(szBuff, 1, count, out)) != count)
268               {
269                     rc = 7;
270                     goto Error;
271               } //disk write error
272         }
273
```

```
274         // re read last 128 bytes of file, handling files < 128 bytes
275         if((count = ftell(out)) == -1)
276         {
277               rc = 8;
278               goto Error;
279         }
280
281         if(count > 128)
282         {
283               count = 128;
284         }
285
286         if(fseek(out, 0-count, SEEK_END) != 0)
287         {
288               rc = 9;
289               goto Error;
290         }
291
292         // get the new position
293         if((total = ftell(out)) == -1)
294         {
295               rc = 10;
296               goto Error;
297         }
298
299         // and read the data
300         count = fread(szBuff, 1, sizeof(szBuff), out);
301         szBuff[count] = '\0';
302
303         // determine offset of terminating boundary line
304         rc = 11;
305
306         for(i=0; i<=(count-(long)strlen(szBoundary)); i++)
307         {
308               if((szBuff[i] == szBoundary[0]) &&
309                           (strncmp(szBoundary, &szBuff[i], strlen(szBoundary)) == 0))
310               {
311                     total+=i;
312                     rc = 0;
313                     break;
314               }
315         }
316
317         // if rc is still set, we didn't find the terminating boundary line
318         if(rc != 0)
319         {
320               goto Error;
321         }
322
323         if(total == 0)
324         {
325               rc = 11;
326               goto Error;
327         }
328
329         // truncate the file at the correct length by writing 0 bytes
330         fflush(out);
331         fclose(out);
332         Error: if(out != NULL)
333         {
334               fclose(out);
335         }
336
337         switch(rc)
338         {
339               case 0: // success
340               printf("Error %d\n The file <b> %s %ld bytes</b>was uploaded
341                                                   sucessfully.", rc, myFile,InCount);
342               break;
```

```
343
344             case 5: // file exists
345             printf("Error %d\n The file <b> %s </b> already exists and cannot be
346                overwritten.<br>Please try again with a different file.", rc, myFile);
347             break;
348
349             case 7: // file too big
350             printf("Error %d\n The file <b> %s </b> is too big. Try again.",
351                                         rc, myFile);
352             break;
353
354             case 8: // file is not an allowed type
355             printf("Error %d\n The file <b> %s </b> is not an allowed type. Try
356                           again.", rc, myFile);
357             break;
358
359             case 11: // 0 byte file
360             printf("Error %d\n The file <b>%s </b>contains no data.<br>Please try
361                           again with a different file.", rc, myFile);
362             //unlink(szFile);
363             break;
364
365             default: // all other cases
366             printf("Error %d uploading file <b>%s </b>. Please try again!",
367                                         rc, myFile);
368             unlink(szFile);
369             break;
370       }
371       return 0;
372 }
373
374 void printHttpHeader()
375 {
376       printf("Content-type: text/html\n\n");
377 }
378
379 void printHttpError(char *errorStr)
380 {
381       printHttpHeader();
382       printf(errorStr);
383 }
384
385 /*
386  * Forwarding to an URL
387  */
388 void print_location(char *url)
389 {
390       printf("Location: %s\n", url);
391       /* FÃ¼r den Fall, dass ein alter Browser keine
392        automatische Weiterleitung unterstÃ¼tzt */
393       printf("Content-Type: text/html\n\n");
394       printf("<html><head>\n");
395       printf("<title>Weiterleitung zu %s</title>\n", url);
396       printf("</head><body>\n");
397       printf("Weiter gehts <a href=\"%s\">hier</a>", url);
398       printf("</body></html>\n");
399 }
400
401 int writeFile(char *path, char *buffer)
402 {
403       FILE *file;
404
405       file =fopen(path, "w+");
406
407       if(file == NULL)
408       {
409             return -1;
410       }
411
```

```
412         fputs(buffer, file);
413         fclose(file);
414 
415         return 1;
416 }
417 
418 /*
419  * Write a line into the file, which was specified by row
420  */
421 int writeLine(char *path, char *buffer, int row)
422 {
423         FILE *file;
424         FILE *tmpfile;
425         int strLength;
426         int done=0;
427         int readLineSize = 255;
428         int filefd;
429 
430         char str[readLineSize];
431         char temp[readLineSize];
432         char tmpfilePath[] = "/motion/cgiXXXXXX";
433 
434         strLength= strlen(buffer);
435 
436         strncpy(str, buffer, strLength);
437         str[strLength]='\0';
438 
439         file =fopen(path, "a+");
440         filefd = mkstemp(tmpfilePath);
441         tmpfile= fdopen(filefd, "w+");
442 
443         if(file == NULL)
444         {
445                 return EXIT_FAILURE;
446         }
447         if(tmpfile == NULL)
448         {
449                 return EXIT_FAILURE;
450         }
451 
452         str[strLength]='\n';
453         str[strLength+1]='\0';
454 
455         int i=0;
456         //aus Originaldatei lesen und nach Tmpdatei kopieren.
457         //An der n-ten Zeile neue Zeile in Tmpdatei einfuegen
458         while(!feof(file)||i<=row)
459         {
460                 if(fgets(temp, readLineSize, file)!=NULL)
461                 {
462                         //n-te Zeile gelesen
463                         if(i==row-1)
464                         {
465                                 fputs(str, tmpfile);
466                                 done=i;
467                         }
468                         else
469                         {
470                                 fputs(temp, tmpfile);
471                         }
472                 }
473                 else
474                 {
475                         //OriginalFile ist leer, anfuegen an erster Stelle
476                         if(done==row-1)
477                         {
478                                 if(done==0&& i==0)
479                                 {
480                                         fputs(str, tmpfile);
```

```
481                             }
482                             break;
483                     }
484             //wenn Position im Orgfile leer, aber der String dort eingefuegt w. muss
485                     if
486                     (i==row)
487                     {
488                             //n-te Zeile gelesen
489                             fputs(str, tmpfile);
490                     }
491                     else
492                     {
493                             //ein'\n' bereits vorhanden? ja->nichts tun
494                             if(temp[strlen(temp)-1]=='\n')
495                             {
496                                     strcpy(temp, "x");
497                             }
498                             else
499                             {
500                                     if(i>0)
501                                             fputs("\n", tmpfile);
502                             }
503                     }
504             }
505             i++;
506     }
507     fclose(file);
508     remove(path);
509     rename(tmpfilePath, path);
510     fclose(tmpfile);
511     return 1;
512 }
513
514 /*
515  * Read a line from a file, which was specified by
516  */
517 int readLine(char *path, char *buffer, int row)
518 {
519     int buf = 255;
520     int len = 255;
521     int readsize;
522     char temp[len];
523
524     FILE *file;
525
526     file =fopen(path, "r");
527
528     if(file == NULL)
529     {
530             return EXIT_FAILURE;
531     }
532     int i;
533     //bis zur n-ten Zeile lesen
534     for(i=0; i<row-1; i++)
535     {
536             if(fgets(temp, buf, file) == NULL)
537             {
538                     fclose(file);
539                     return EXIT_FAILURE;
540             }
541     }
542     if(fgets(buffer, buf, file) == NULL)
543     {
544             fclose(file);
545             return EXIT_FAILURE;
546     }
547     //n-ten Zeile existiert
548     readsize = strlen(buffer);
549     buffer[readsize]='\0';
```

```
550         fclose(file);
551
552         return EXIT_SUCCESS;
553 }
554
555 /*
556  *  Function reads data from the POST- or GET- methode
557  *  Returntype: String buffer with data
558  *  Failure   : NULL
559  */
560 char *getdata(void)
561 {
562         char *puffer= NULL;
563         char *cont_len;
564         char *cgi_string;
565         char *request;
566         char *content_type;
567         unsigned long length;
568
569         /* Zuerst die Request-Methode Ã¼berprÃ¼fen GET */
570         request = getenv("REQUEST_METHOD");
571         if( NULL == request)
572                 return NULL;
573         else if(strcmp(request, "GET") == 0)
574         {
575                 /* Die Methode GET -> Query String abholen */
576                 cgi_string = getenv("QUERY_STRING");
577                 if( NULL == cgi_string)
578                         return NULL;
579                 else
580                 {
581                         puffer =(char *) Strdup(cgi_string);
582                         return puffer; /* RÃ¼ckgabewert an den Aufrufer */
583                 }
584         }
585         /* Dann die Request-Methode Ã¼berprÃ¼fen POST */
586         else if(strcmp(request, "POST") == 0)
587         {
588                 content_type = getenv("CONTENT_TYPE");
589                 if(NULL == content_type)
590                         return NULL;
591                 if(strstr(content_type, "multipart"))
592                 {
593                         fileUpload();
594                         return NULL;
595                 }
596
597                 /* Die Methode POST -> LÃ¤nge des Strings
598                  * ermitteln(CONTENT_LENGTH) */
599                 cont_len = getenv("CONTENT_LENGTH");
600                 if( NULL == cont_len)
601                         return NULL;
602                 else
603                 {
604                         /* String CONTENT_LENGTH in unsigned long umwandeln */
605                         length =(unsigned long) atoi(cont_len);
606                         if(length <= 0)
607                                 return NULL; /* Keine Eingabe!?!? */
608                 }
609                 /* Jetzt lesen wir die Daten von stdin ein */
610                 puffer =(char *) malloc(length+1);
611                 if( NULL == puffer)
612                         return NULL;
613                 else
614                 {
615                         if( NULL == fgets(puffer, length+1, stdin))
616                         {
617                                 free(puffer);
618                                 return NULL;
```

```
619                     }
620                     else
621                             /* RÃ¼ckgabewerte an dem Ausrufer */
622                             return puffer;
623             }
624     }
625     /* Weder GET-Methode noch die POST-Methode wurden verwendet */
626     else
627             return NULL;
628 }
629
630 /*
631  * an ANSI C-function of strdup() in <string.h>
632  */
633 char *Strdup(const char *str)
634 {
635     char *p;
636     if(NULL == str)
637             return NULL;
638     else
639     {
640             p =(char *)malloc(strlen(str)+1);
641             if(NULL == p)
642                     return NULL;
643             else
644                     strcpy(p, str);
645     }
646     return p;
647 }
648
649 /* Wandelt einzelne Hexzeichen(%xx) in ASCII-Zeichen
650  * und kodierte Leerzeichen(+) in echte Leerzeichen um */
651 void hex2ascii(char *str)
652 {
653     int x, y;
654     for(x=0, y=0; str[y] != '\0'; ++x, ++y)
655     {
656             str[x] = str[y];
657             /* Ein hexadezimales Zeichen? */
658             if(str[x] == '%')
659             {
660                     str[x] = convert(&str[y+1]);
661                     y += 2;
662             }
663             /* Ein Leerzeichen? */
664             else if(str[x] == '+')
665                     str[x]=' ';
666     }
667     /* Geparsten String terminieren */
668     str[x] = '\0';
669 }
670
671 /*
672  * Function converts an String of two hex signs into the original sign
673  */
674 char convert(char *hex)
675 {
676     char ascii;
677     /* erster Hexawert */
678     ascii =(hex[0] >= 'A' ?((hex[0] & 0xdf) - 'A')+10 :(hex[0] - '0'));
679     ascii <<= 4; /* Bitverschiebung schneller als ascii*=16 */
680     /* zweiter Hexawert */
681     ascii +=(hex[1] >= 'A' ?((hex[1] & 0xdf) - 'A')+10 :(hex[1] - '0'));
682     return ascii;
683 }
684
685 /* List of Variables/Value- pairs
686  * Returnvalue: Pointer of the beginning of the list
687  * Failure: NULL
```

```
688  */
689 struct CGI_DATEN *creatPairs(char *str)
690 {
691         char* s;
692         char* res;
693         /*Bis MAX_PAARE*/
694         char *paare[MAX_PAARE];
695         struct CGI_DATEN *ptr_daten= NULL;
696         struct CGI_DATEN *ptr_anfang= NULL;
697         int i=0, j=0;
698         /* Zuerst werden die Variablen/Werte-Paare anhand des Zeichens
699          * '&' getrennt, sofern es mehrere sind */
700         s=str;
701         res=strtok(s, "&");
702         while(res != NULL&& i < MAX_PAARE)
703         {
704                 /* Wert von res dynamisch in char **pair speichern */
705                 paare[i] =(char *)malloc(strlen(res)+1);
706                 if(paare[i] == NULL)
707                         return NULL;
708                 paare[i] = res;
709                 res=strtok(NULL,"&");
710                 i++;
711         }
712         /* Jetzt werden die Variablen von den Werten getrennt und
713          * an die Struktur CGI_DATEN Ã¼bergeben */
714         while(i > j )
715         {
716                 /* Das erste Element? */
717                 if(ptr_anfang == NULL)
718                 {
719                         ptr_anfang =(struct CGI_DATEN *)
720                                                 malloc(sizeof(struct CGI_DATEN *));
721                         if(ptr_anfang == NULL)
722                                 return NULL;
723                         res = strtok(paare[j], "=");
724                         if(res == NULL)
725                                 return NULL;
726                         ptr_anfang->variable =(char *) malloc(strlen(res)+1);
727                         if(ptr_anfang->variable == NULL)
728                                 return NULL;
729                         ptr_anfang->variable = res;
730                         res = strtok(NULL, "\0");
731                         if(res == NULL)
732                                 return NULL;
733                         ptr_anfang->wert =(char *) malloc(strlen(res)+1);
734                         if(ptr_anfang->wert == NULL)
735                                 return NULL;
736                         ptr_anfang->wert = res;
737                         /* printf("%s %s<br>",
738                          * ptr_anfang->variable, ptr_anfang->wert); */
739                         ptr_anfang->next
740                                         =(struct CGI_DATEN *)
741                                                 malloc(sizeof(struct CGI_DATEN *));
742                         if(ptr_anfang->next == NULL)
743                                 return NULL;
744                         ptr_daten = ptr_anfang->next;
745                         j++;
746                 }
747                 else
748                 { /* Die restlichen Elemente */
749                         res = strtok(paare[j], "=");
750                         if(res == NULL)
751                                 return NULL;
752                         ptr_daten->variable =(char *) malloc(strlen(res)+1);
753                         if(ptr_daten->variable == NULL)
754                                 return NULL;
755                         ptr_daten->variable = res;
756                         res = strtok(NULL, "\0");
```

```
757                     if(res == NULL)
758                             return NULL;
759                     ptr_daten->wert =(char *) malloc(strlen(res)+1);
760                     if(ptr_daten->wert == NULL)
761                             return NULL;
762                     ptr_daten->wert = res;
763                     /* printf("%s %s<br>",
764                      * ptr_daten->variable,  ptr_daten->wert); */
765                     ptr_daten->next = (struct CGI_DATEN *)
766                                             malloc(sizeof(struct CGI_DATEN *));
767                     if(ptr_daten->next == NULL)
768                             return NULL;
769                     ptr_daten = ptr_daten->next;
770                     j++;
771             }
772     }
773     ende = ptr_daten;
774     /* Anfangsadresse der Liste struct CGI_DATEN zurÃ¼ckgeben */
775     return ptr_anfang;
776 }
777
778 void deletePairChain(struct CGI_DATEN *daten)
779 {
780     struct CGI_DATEN *next= NULL;
781     while(daten != ende)
782     {
783             next = daten->next;
784             if(daten->variable != NULL)
785                     free(daten);
786             daten=next;
787     }
788 }
789
790 void printf_error(char *str)
791 {
792     printf("Content-Type: text/html\n\n");
793     printf("<html><head>\n");
794     printf("<title>CGI-Fehlermeldung</title>\n");
795     printf("</head><body>\n");
796     printf("%s", str);
797     printf("</body></html>\n");
798 }
799
800 int isJpegFile(char *str)
801 {
802     if(strstr(str, ".jpeg"))
803             return 1;
804     if(strstr(str, ".jpg"))
805             return 1;
806     return 0;
807 }
808
809 /*
810  * Scans all jpeg images and creates the Option- Tag for the browser
811  */
812 void scanImages()
813 {
814     int num_entries, i;
815     struct dirent **namelist, **list;
816     int first=1;
817
818     //Eintraege werden alphabetisch sortiert
819     if((num_entries=scandir(imageDir, &namelist, 0, alphasort)) < 0)
820     {
821             printHttpError("Unknow failure in CGI\n");
822             exit(EXIT_FAILURE);
823     }
824
825     if(num_entries)
```

```
826         {
827                 for(i=0, list=namelist; i < num_entries; i++, *list++)
828                 {
829                         if(isJpegFile((*list)->d_name))
830                         {
831                                 if(first)
832                                 {
833                                         printf("<option selected>%s</option>\n",
834                                                                                 (*list)->d_name);
835                                         first=0;
836                                 }
837                                 else
838                                 {
839                                         printf("<option>%s</option>\n",(*list)->d_name);
840                                 }
841                         }
842                         free(*list);
843                 }
844                 free(namelist);
845         }
846 }
847
848 /*
849  * Deletes specified image by name
850  */
851 int deleteImage(char *imageName)
852 {
853         char home[100];
854         if(isJpegFile(imageName))
855         {
856                 getcwd(home, sizeof(home));
857                 chdir(imageDir);
858                 if((remove(imageName)) < 0)
859                 {
860                         printHttpError("The file ");
861                         printf("%s doesn 't exist. Please refresh the list.\n", imageName);
862                         chdir(home);
863                         return 0;
864                 }
865                 chdir(home);
866         }
867         return 1;
868 }
869
870 /*
871  * Removes all images
872  */
873 int deleteAllImages()
874 {
875         char home[100];
876         int num_entries, i;
877         struct dirent **namelist, **list;
878
879         //Eintraege werden alphabetisch sortiert
880         if((num_entries=scandir(imageDir, &namelist, 0, alphasort)) < 0)
881         {
882                 printHttpError("Unknow failure in CGI\n");
883                 exit(EXIT_FAILURE);
884         }
885
886         if(num_entries)
887         {
888                 getcwd(home, sizeof(home));
889                 chdir(imageDir);
890                 for(i=0, list=namelist; i < num_entries; i++, *list++)
891                 {
892                         if(isJpegFile((*list)->d_name))
893                         {
894                                 if((remove((*list)->d_name)) < 0)
```

```
895                             {
896                                     printHttpError("The file ");
897                                     printf("%s doesn 't exist.
898                                             Please refresh the list.\n",(*list)->d_name);
899                                     chdir(home);
900                                     return 0;
901                             }
902                     }
903                     free(*list);
904             }
905             free(namelist);
906             chdir(home);
907     }
908     return 1;
909 }
910
911 /*
912  * Targeted to WebcamBrows.html. Creates the Select- Tag
913  */
914 void printSeletTag()
915 {
916     printHttpHeader();
917     printf("<select name=\"Auswahl\" size=\"10\" >");
918     scanImages();
919     printf("<option>----------------------------------------</option>\n");
920     printf("</select>\n");
921 }
922
923 void getFilePath(char *buffer, char *path, char *fileName)
924 {
925     /* Vorhandenen Platz in string ermitteln */
926     buffer[0] = '\0';
927     strcpy(buffer, path);
928     strcat(buffer, fileName);
929 }
930
931 void writeMDConfigFile(struct CGI_DATEN *cgi, char *filePath)
932 {
933     char str[500] = "";
934     char status[550]="";
935
936     while(cgi->variable!=NULL)
937     {
938         // Normal or ROI or Start MD
939         if(strncmp(cgi->variable, "bt_startMD", 9)==0 || strncmp(cgi->variable,
940                 "ImageType", 9)==0)
941         {
942             strcat(status, "Status     :");
943             strcat(status, cgi->wert);
944             strcat(status, "\n");
945         }
946         else if(strncmp(cgi->variable, "Showcap", 7) == 0)
947         {
948             sprintf(status, "Status     :%s", cgi->variable);
949             writeLine(filePath, status, 1);
950             return;
951         }
952         else if(strncmp(cgi->variable, "bt_singleShot", 9) != 0)
953         {
954             strcat(str, cgi->wert);
955             strcat(str, "\n");
956         }
957
958
959         cgi = cgi->next;
960     }
961     strcat(status, str);
962     writeFile(filePath, status);
963 }
```

```
964
965 /*
966  * Targeted to WebcamSetting.html. Writes in the TextArea- Tag
967  */
968 void printStatus(char *confFilePath, char *alarmFilePath)
969 {
970        char strBuffer[100];
971        int i=1;
972        int mdpid;
973
974        if(access(mdCapDir, F_OK) != -1)
975        {
976               printCap(mdCapDir);
977               return;
978        }
979
980        printHttpHeader();
981        mdpid= md_pid();
982
983        if(mdpid==-1)
984        {
985               printf(FAILURE_PROCESSMESSAGE);
986        }
987
988        if(mdpid>0)
989        {
990               printf("MD up, PID is %d\n", mdpid);
991        }
992        else
993        {
994               printf("MD down\n");
995        }
996
997        if(readLine(alarmFilePath, strBuffer, 1) != EXIT_FAILURE)
998        {
999               if(strlen(strBuffer)>0)
1000               {
1001                      printf("Last Detection Message:\n");
1002                      printf("%s", strBuffer);
1003               }
1004        }
1005
1006        while(i<=11)
1007        {
1008               if(readLine(confFilePath, strBuffer, i++) != EXIT_FAILURE)
1009               {
1010                      if(strlen(strBuffer)>0)
1011                             printf("%s", strBuffer);
1012               }
1013               else
1014               {
1015                      break;
1016               }
1017        }
1018 }
1019
1020 /*
1021  * Targeted to WebcamSetting.html. Writes in the TextArea- Tag
1022  */
1023 void printCap(char *mdCapDir)
1024 {
1025        char strBuffer[100];
1026        int i=1;
1027
1028        while(i<=20)
1029        {
1030               if(readLine(mdCapDir, strBuffer, i++) != EXIT_FAILURE)
1031               {
1032                      if(strlen(strBuffer)>0)
```

```
1033                            printf("%s", strBuffer);
1034                }
1035                else
1036                {
1037                        break;
1038                }
1039
1040        }
1041  }
1042
1043  int md_pid()
1044  {
1045        FILE *pipe_writer;
1046        char puffer[40];
1047
1048        if((pipe_writer = popen("pidof motion_detector", "r")) != NULL)
1049        {
1050                if(fgets(puffer, 40, pipe_writer) != NULL)
1051                {
1052                        pclose(pipe_writer);
1053                        return atoi(puffer);
1054                }
1055        }
1056
1057        return -1;
1058  }
1059
1060  int startMD()
1061  {
1062        int mdpid;
1063
1064        mdpid= md_pid();
1065
1066        if(mdpid==-1)
1067        {
1068                printf(FAILURE_PROCESSMESSAGE);
1069                return 0;
1070        }
1071
1072        if(mdpid>0)
1073        {
1074                printf("MD up, processid is: %d\n", mdpid);
1075        }
1076        else
1077        {
1078                printf("MD down, restarting.\n");
1079                system("/motion/motion_detector");
1080                mdpid= md_pid();
1081                if(mdpid==-1)
1082                {
1083                        printf(FAILURE_PROCESSMESSAGE);
1084                        return 0;
1085                }
1086                if(mdpid>0)
1087                {
1088                        printf("MD restarted: new processid is %d\n", mdpid);
1089                }
1090        }
1091        return 1;
1092  }
1093
1094  int main(void)
1095  {
1096        char *str;
1097        struct CGI_DATEN *cgi;
1098        struct CGI_DATEN *free_cgi;
1099        char confFilePath[80];
1100        char alarmFilePath[80];
1101        char status[50]= "Status     :";
```

```
1102        int sec=5;
1103
1104        /* Eingabe einlesen */
1105        str = getdata();
1106        if(str == NULL)
1107        {
1108              printHttpError("Error while reading the form input");
1109              return EXIT_FAILURE;
1110        }
1111        /* Hexzeichen in ASCII-Zeichen konvertieren und aus '+'
1112         * Leerzeichen machen */
1113        hex2ascii(str);
1114        /* Liste der Formualardaten erstellen */
1115        cgi = creatPairs(str);
1116
1117        getFilePath(confFilePath, mdStatusDir, mdStatusFileName);
1118        getFilePath(alarmFilePath, mdStatusDir, mdAlarmFileName);
1119
1120        free_cgi = cgi;
1121        if(cgi == NULL)
1122        {
1123              printHttpError("Error while creating the "
1124                    "variable and value- list\n");
1125              return EXIT_FAILURE;
1126        }
1127
1128        if(cgi->wert != NULL)
1129        {
1130              if(strstr(cgi->wert, "Reload"))
1131              {
1132                    printSeletTag();
1133              }
1134              else if(strstr(cgi->wert, "Status"))
1135              {
1136                    printStatus(confFilePath, alarmFilePath);
1137              }
1138              else if(strstr(cgi->wert, "Webcam"))
1139              {
1140                    writeMDConfigFile(cgi, confFilePath);
1141                    startMD();
1142                    sec=5;
1143                    while(sec>0)
1144                    {
1145                          sleep(1);
1146                          if(access(mdCapDir, F_OK) != -1)
1147                          {
1148                                printStatus(confFilePath, alarmFilePath);
1149                                break;
1150                          }
1151                    }
1152              }
1153              else if(strstr(cgi->wert, "DeleteAll"))
1154              {
1155                    if(deleteAllImages())
1156                          printSeletTag();
1157              }
1158              else if(strstr(cgi->wert, "Delete"))
1159              {
1160                    cgi = cgi->next;
1161                    if(deleteImage(cgi->wert))
1162                          printSeletTag();
1163              }
1164              else if(strstr(cgi->variable, "bt_startMD"))
1165              {
1166                    //     writeLine(filePath, cgi->wert, 1);
1167                    //        cgi = cgi->next;
1168                    if(access(mdCapDir, F_OK) != -1)
1169                    {
1170                          remove(mdCapDir);
```

```
1171                    }
1172                    if(access(alarmFilePath, F_OK) != -1)
1173                    {
1174                            remove(alarmFilePath);
1175                    }
1176                    writeMDConfigFile(cgi, confFilePath);
1177                    printHttpHeader();
1178                    startMD();
1179            }
1180            else if(strstr(cgi->variable, "bt_singleShot"))
1181            {
1182                    if(access(mdCapDir, F_OK) != -1)
1183                    {
1184                            remove(mdCapDir);
1185                    }
1186                    writeMDConfigFile(cgi, confFilePath);
1187                    printHttpHeader();
1188                    startMD();
1189            }
1190            else if(strstr(cgi->variable, "Stopoption"))
1191            {
1192                    if(access(mdCapDir, F_OK) != -1)
1193                    {
1194                            remove(mdCapDir);
1195                    }
1196                    strcat(status, cgi->wert);
1197                    writeLine(confFilePath, status, 1);
1198            }
1199    }
1200
1201    /* Speicher wieder freigeben */
1202    deletePairChain(free_cgi);
1203    return EXIT_SUCCESS;
1204 }
1205
```

Java Applet Source Code

ROIHandler.java (Eclipse Projekt AppletDrawing)

```
1  package drawing;
2
3  import java.awt.*;
4  import java.awt.image.BufferedImage;
5  import java.io.BufferedReader;
6  import java.io.File;
7  import java.io.IOException;
8  import java.io.InputStreamReader;
9  import java.net.URL;
10 import java.net.URLConnection;
11 import javax.imageio.ImageIO;
12 import javax.swing.border.Border;
13 import javax.swing.border.CompoundBorder;
14 import javax.swing.border.EmptyBorder;
15 import javax.swing.border.TitledBorder;
16 import javax.swing.JApplet;
17 import javax.swing.JPanel;
18 import java.io.*;
19
20 public class ROIHandler extends JApplet {
21        public final static int width = 670;
22        public final static int height = 670;
23        public final static String saveImagePath = "/rootfs/var/roi.jpeg";
24        public final static String getImagePath = "/home/roi.jpg";
25        public final static String cursorPath = "/home/stift.gif";
26        public final static String roiLocation = "pics/roi.jpg";
27        public final static String mouseLocation = "pics/stift.gif";
28
29        private final int maxImaageSize = 300000;
30        private boolean isLocal = true;
31
32        Border emptyBorder = new EmptyBorder(3, 3, 3, 3);
33        DrawingROI droi;
34
35        Image curImage, mouseImage;
36
37        int w = 640, h = 480;
38
39        public void init() {
40               isLocal = false;
41               // DNT- Farben
42               int fr = 0xB9, fg = 0xB9, fb = 0xB9;
43               int pr = 0xBB, pg = 0xBB, pb = 0xBB;
44               int rgb;
45               try {
46                      String fColor = getParameter("fColor");
47
48                      rgb = Integer.parseInt(fColor, 16);
49
50                      System.out.println(rgb);
51
52                      System.out.println(Integer.toHexString(rgb).toUpperCase());
53
54                      // Zurueckwandeln...
55                      fr = ((rgb >> 16) & 0xFF);
56                      fg = ((rgb >> 8) & 0xFF);
57                      fb = ((rgb >> 0) & 0xFF);
58               } catch (NumberFormatException e) {
59               }
60
61               try {
62                      String pColor = getParameter("pColor");
63
64                      rgb = Integer.parseInt(pColor, 16);
```

```
65
66                      System.out.println(rgb);
67
68                      System.out.println(Integer.toHexString(rgb).toUpperCase());
69
70                      // ZurÃ¼ckwandeln...
71                      pr = ((rgb >> 16) & 0xFF);
72                      pg = ((rgb >> 8) & 0xFF);
73                      pb = ((rgb >> 0) & 0xFF);
74              } catch (NumberFormatException e) {
75              }
76
77              this.setBackground(new Color(fr, fg, fb));
78              GridBagConstraints gbc = new GridBagConstraints();
79              gbc.insets = new Insets(3, 3, 3, 3);
80
81              Container cp = getContentPane();
82              cp.setLayout(new FlowLayout());
83
84              JPanel pMain = new JPanel();
85              pMain.setLayout(new GridBagLayout());
86              pMain.setBackground(new Color(fr, fg, fb));
87
88              JPanel pDraw = new JPanel();
89
90              pDraw.setBorder(new CompoundBorder(new TitledBorder(null,
91                                      "ROI Image",
92                              TitledBorder.LEFT, TitledBorder.TOP), emptyBorder));
93              pDraw.setBackground(new Color(pr, pg, pb));
94              pDraw.setPreferredSize(new Dimension(650, 538));
95              gbc.gridx = 0;
96              gbc.gridy = 0;
97
98              droi = new DrawingROI(w, h);
99              pDraw.add(droi, gbc);
100
101             pMain.setBorder(new CompoundBorder(new TitledBorder(null,
102                   "ROI Drawing tool", TitledBorder.LEFT, TitledBorder.TOP),
103                           emptyBorder));
104
105             setName("ROI Drawing tool");
106             setSize(width, height);
107
108             ROIPanel roip = new ROIPanel(this, droi);
109
110             roip.setBorder(new CompoundBorder(new TitledBorder(null,
111                 "Controll",TitledBorder.LEFT, TitledBorder.TOP),
112                         emptyBorder));
113             roip.setBackground(new Color(pr, pg, pb));
114
115             gbc.gridx = 0;
116             gbc.gridy = 0;
117             pMain.add(pDraw, gbc);
118             gbc.gridx = 0;
119             gbc.gridy = 1;
120             pMain.add(roip, gbc);
121             // setContentPane(pMain);
122             // cp.add(BorderLayout.NORTH, droi);
123             cp.add(BorderLayout.SOUTH, pMain);
124             // loadImage();
125             // loadMausImage();
126             droi.setMouseImage();
127             repaint();
128
129     }
130
131     public void loadImage() {
132             try {
133                     BufferedImage bimg = null;
```

```
134                     if (!isLocal) {
135                             URL url = new URL(getCodeBase(), roiLocation);
136                             InputStream inputstream = url.openStream();
137
138                             int b;
139                             byte[] buffer = new byte[maxImaageSize];
140                             int index = 0;
141                             while ((b = inputstream.read()) != -1) {
142                                     buffer[index++] = (byte) b;
143                             }// end while
144                             int len = index;
145
146                             curImage =
147                     Toolkit.getDefaultToolkit().createImage(buffer, 0, len);
148                             bimg = createBufferedImage(curImage);
149
150                     } else {
151                             File file = new File(getImagePath);
152                             bimg = ImageIO.read(file);
153                     }
154
155                     if (bimg != null) {
156                             droi.init(bimg);
157                             droi.repaint();
158                     }
159             } catch (Exception e) {
160                     // e.printStackTrace();
161             }
162     }
163
164     public void loadMausImage() {
165             try {
166                     BufferedImage bimg = null;
167                     if (!isLocal) {
168                             mouseImage = getImage(getCodeBase(), mouseLocation);
169                             bimg = createBufferedImage(mouseImage);
170
171                     } else {
172                             File file = new File(cursorPath);
173                             bimg = ImageIO.read(file);
174                     }
175
176                     if (bimg != null) {
177                             droi.setMouseImage(bimg);
178                     }
179             } catch (Exception e) {
180                     // e.printStackTrace();
181             }
182     }
183
184     public BufferedImage createBufferedImage(Image image) {
185             try {
186                     Component dummy = new Component() {
187                     };
188                     MediaTracker tracker = new MediaTracker(dummy);
189                     tracker.addImage(image, 0);
190                     tracker.waitForID(0);
191                     if (!tracker.isErrorID(0)) {
192                             if (image instanceof BufferedImage)
193                                     return (BufferedImage) image;
194                             int w = image.getWidth(this);
195                             int h = image.getHeight(this);
196                             BufferedImage bi = new BufferedImage(w, h,
197                                             BufferedImage.TYPE_INT_RGB);
198                             Graphics2D g = bi.createGraphics();
199                             g.drawImage(image, 0, 0, this);
200                             g.dispose();
201                             return bi;
202                     }
```

```
203                 } catch (Exception e) {
204
205                 }
206                 return null;
207         }
208
209         public void sendImage(byte[] buf) {
210                 try {
211                         URL url = new URL(getCodeBase(), "/cgi-bin/MDCGIHandler.cgi");
212                         // create a boundary string
213                         String boundary =
214                            MultiPartFormOutputStream.createBoundary();
215                         URLConnection urlConn =
216                            MultiPartFormOutputStream.createConnection(url);
217                         urlConn.setRequestProperty("Accept", "*/*");
218                         urlConn.setRequestProperty("Content-Type",
219                         MultiPartFormOutputStream.getContentType(boundary));
220                         // set some other request headers...
221                         urlConn.setRequestProperty("Connection", "Keep-Alive");
222                         urlConn.setRequestProperty("Cache-Control", "no-cache");
223                         // no need to connect the getOutputStream() does it
224                         MultiPartFormOutputStream out =
225                          new MultiPartFormOutputStream(urlConn.getOutputStream(),
226                                                                 boundary);
227                         out.writeFile("myFile", "multipart/form-data",
228                                                 "roi.jpg", buf);
229                         out.close();
230                         // read response from server
231                         BufferedReader in =
232                          new BufferedReader(new InputStreamReader(
233                                         urlConn.getInputStream()));
234                         String line = "";
235                         while ((line = in.readLine()) != null) {
236                                 System.out.println(line);
237                         }
238                         in.close();
239
240                 } catch (IOException ex) {
241                         ex.printStackTrace();
242                 }
243         }
244
245         public void repaintAll(BufferedImage curImage) {
246                 w = curImage.getWidth();
247                 h = curImage.getHeight();
248                 init();
249                 repaint();
250         }
251
252 }
253
```

ROIPanel.java (Eclipse Projekt AppletDrawing)

```
1  package drawing;
2
3  import java.awt.GridBagConstraints;
4  import java.awt.GridBagLayout;
5  import java.awt.event.ActionEvent;
6  import java.awt.event.ActionListener;
7  import java.awt.image.BufferedImage;
8  import java.io.ByteArrayOutputStream;
9  import javax.imageio.ImageIO;
10 import javax.swing.ButtonGroup;
11 import javax.swing.JButton;
12 import javax.swing.JPanel;
13 import javax.swing.JToggleButton;
14
15 class ROIPanel extends JPanel {
16        DrawingROI drawingROI;
17
18        private JButton saveImageButton = new JButton("Save ROI Image");
19        private JButton getImageButton = new JButton("Get ROI Image");
20        private JButton unDoButton = new JButton("Undo");
21        private JButton reDoButton = new JButton("Redo");
22        private JToggleButton greenButton = new JToggleButton("Max");
23        private JToggleButton blackButton = new JToggleButton("Ignore");
24        private JToggleButton redButton = new JToggleButton("Min");
25        private JToggleButton pensizebutton1 = new JToggleButton("Big Pen");
26        private JToggleButton pensizebutton2 = new JToggleButton("Small Pen");
27        private ButtonGroup buttonGroupColor = new ButtonGroup();
28        private ButtonGroup buttonGroupPenSize = new ButtonGroup();
29
30        int altX;
31        int altY;
32        int n_x;
33        int n_y;
34
35        ROIHandler pMain;
36
37        public ROIPanel(final ROIHandler pMain, final DrawingROI drawingROI) {
38              super();
39              this.pMain = pMain;
40              this.drawingROI = drawingROI;
41              drawingROI.setReference(greenButton, blackButton, redButton,
42                            pensizebutton1, pensizebutton2);
43              saveImageButton.addActionListener(new ActionListener() {
44                    public void actionPerformed(ActionEvent e) {
45                          try {
46                            BufferedImage bufferedImage =
47                           new BufferedImage(drawingROI.getImage().getWidth(),
48                               drawingROI.getImage().getHeight(),
49                                  BufferedImage.TYPE_INT_RGB);
50                              // Jpeg Image erzeugen
51                              drawingROI.paint(bufferedImage.getGraphics());
52                              ByteArrayOutputStream baos =
53                                        new ByteArrayOutputStream();
54                              ImageIO.write(bufferedImage, "jpeg", baos);
55
56                              byte[] buf = baos.toByteArray();
57                              pMain.sendImage(buf);
58
59                          } catch (Exception ex) {
60                                // ex.printStackTrace();
61                          }
62                    }
63              });
64
65              getImageButton.addActionListener(new ActionListener() {
66                    public void actionPerformed(ActionEvent e) {
```

```
67                      try {
68                              pMain.loadImage();
69                      } catch (Exception ex) {
70                              ex.printStackTrace();
71                      }
72
73              }
74      });
75
76      unDoButton.addActionListener(new ActionListener() {
77              public void actionPerformed(ActionEvent e) {
78                      drawingROI.undo();
79                      // drawingROI.repaint();
80                      // repaint();
81              }
82      });
83
84      reDoButton.addActionListener(new ActionListener() {
85              public void actionPerformed(ActionEvent e) {
86                      drawingROI.redo();
87                      // drawingROI.repaint();
88                      // repaint();
89              }
90      });
91
92      // ButtonGroup add buttons
93      buttonGroupColor.add(greenButton);
94      buttonGroupColor.add(blackButton);
95      buttonGroupColor.add(redButton);
96      buttonGroupPenSize.add(pensizebutton1);
97      buttonGroupPenSize.add(pensizebutton2);
98      pensizebutton1.setSelected(true);
99      blackButton.setSelected(true);
100
101     GridBagLayout gridbag = new GridBagLayout();
102     GridBagConstraints constrains = new GridBagConstraints();
103     this.setLayout(gridbag);
104     // dehnt die Komponenten in beiden richtungen
105     constrains.fill = GridBagConstraints.BOTH;
106
107     buildConstrains(constrains, 0, 1, 1, 1, 100, 100);
108     gridbag.setConstraints(blackButton, constrains);
109     this.add(blackButton);
110
111     buildConstrains(constrains, 1, 1, 1, 1, 100, 100);
112     gridbag.setConstraints(redButton, constrains);
113     this.add(redButton);
114
115     buildConstrains(constrains, 2, 1, 1, 1, 100, 100);
116     gridbag.setConstraints(greenButton, constrains);
117     this.add(greenButton);
118
119     buildConstrains(constrains, 3, 1, 1, 1, 100, 100);
120     gridbag.setConstraints(pensizebutton1, constrains);
121     this.add(pensizebutton1);
122
123     buildConstrains(constrains, 4, 1, 1, 1, 100, 100);
124     gridbag.setConstraints(pensizebutton2, constrains);
125     this.add(pensizebutton2);
126
127     buildConstrains(constrains, 0, 2, 2, 1, 100, 100);
128     gridbag.setConstraints(saveImageButton, constrains);
129     this.add(saveImageButton);
130
131     buildConstrains(constrains, 2, 2, 1, 1, 100, 100);
132     gridbag.setConstraints(unDoButton, constrains);
133     this.add(unDoButton);
134
135     buildConstrains(constrains, 3, 2, 1, 1, 100, 100);
```

```
136                 gridbag.setConstraints(reDoButton, constrains);
137                 this.add(reDoButton);
138 
139                 buildConstrains(constrains, 4, 2, 2, 1, 100, 100);
140                 gridbag.setConstraints(getImageButton, constrains);
141                 this.add(getImageButton);
142 
143                 this.repaint();
144         }// end Constructor
145 
146         private void buildConstrains(GridBagConstraints gbc, int gx, int gy,
147                         int gw, int gh, int wx, int wy) {
148                 gbc.gridx = gx;
149                 gbc.gridy = gy;
150                 gbc.gridwidth = gw;
151                 gbc.gridheight = gh;
152                 gbc.weightx = wx;
153                 gbc.weighty = wy;
154         }
155 }
156 
```

DrawingROI.java (Eclipse Projekt AppletDrawing)

```
1  package drawing;
2  import java.awt.Color;
3  import java.awt.Cursor;
4  import java.awt.Dimension;
5  import java.awt.Graphics;
6  import java.awt.Graphics2D;
7  import java.awt.Point;
8  import java.awt.event.MouseEvent;
9  import java.awt.event.MouseListener;
10 import java.awt.event.MouseMotionListener;
11 import java.awt.image.BufferedImage;
12 import java.io.File;
13 import java.util.ArrayList;
14 import javax.swing.JPanel;
15 import javax.swing.JToggleButton;
16
17 class DrawingROI extends JPanel implements MouseListener, MouseMotionListener {
18        private BufferedImage curImage;
19        private BufferedImage preImage;
20        private boolean inDrag = false;
21        private boolean initialized = false;
22        private int curX = -1, curY = -1;
23        private JToggleButton greenButton, blackButton, redButton, pensizebutton1,
24                    pensizebutton2;
25
26        File file, cursorfile;
27        BufferedImage cursorImage = null;
28        private int imageElementCounter, imageElementCounterMax;
29
30        ArrayList<BufferedImage> imageList;
31
32        // "Constructor" - creates the object
33        public DrawingROI(int width, int height) {
34              super();
35              addMouseListener(this);
36              addMouseMotionListener(this);
37              this.setPreferredSize(new Dimension(width, height));
38        }
39
40        public void setReference(JToggleButton greenButton,
41                    JToggleButton blackButton, JToggleButton redButton,
42                    JToggleButton pensizebutton1, JToggleButton pensizebutton2) {
43              this.greenButton = greenButton;
44              this.blackButton = blackButton;
45              this.redButton = redButton;
46              this.pensizebutton1 = pensizebutton1;
47              this.pensizebutton2 = pensizebutton2;
48        }
49
50        public BufferedImage getImage() {
51              return curImage;
52        }
53
54        public void init(BufferedImage curImage) {
55              initialized = true;
56              setName("Drawing ROI");
57              cursorfile = new File(ROIHandler.cursorPath);
58              this.curImage = curImage;
59              setSize(curImage.getWidth(), curImage.getHeight());
60              preImage = new BufferedImage(curImage.getWidth(), curImage.getHeight(),
61                          curImage.getType());
62              Graphics g = preImage.getGraphics();
63              g.drawImage(curImage, 0, 0, this);
64              imageList = new ArrayList<BufferedImage>();
65              imageElementCounter = 0;
66              imageElementCounterMax = 0;
```

```
67              imageList.add(preImage);
68              repaint();
69      }
70
71      public void setMouseImage(BufferedImage mouseImage) {
72              Cursor c;
73              if (mouseImage != null) {
74                      c = getToolkit().createCustomCursor(mouseImage, new Point(1, 1),
75                                      "Cursor");
76                      setCursor(c);
77              } else {
78                      setMouseImage();
79              }
80      }
81
82      public void setMouseImage() {
83              Cursor c = new Cursor(Cursor.CROSSHAIR_CURSOR);
84              setCursor(c);
85      }
86
87      public void showStatus(String s) {
88              System.out.println(s);
89      }
90
91      // Five methods from MouseListener:
92      /** Called when the mouse has been clicked on a component. */
93      public void mouseClicked(MouseEvent e) {
94      }
95
96      /** Called when the mouse enters a component. */
97      public void mouseEntered(MouseEvent e) {
98      }
99
100     /** Called when the mouse exits a component. */
101     public void mouseExited(MouseEvent e) {
102     }
103
104     public void mouseMoved(MouseEvent e) {
105             showStatus("mouse Moved to " + e.getPoint());
106     }
107
108     /** Called when the mouse has been pressed. */
109     public void mousePressed(MouseEvent e) {
110             if (initialized) {
111                     Point p = e.getPoint();
112                     showStatus("mouse Pressed to " + p);
113                     curX = p.x;
114                     curY = p.y;
115                     inDrag = true;
116                     repaint();
117             }
118     }
119
120     /** Called when the mouse has been released. */
121     public void mouseReleased(MouseEvent e) {
122             if (initialized) {
123                     inDrag = false;
124                     // undo function
125                     while (imageElementCounterMax > imageElementCounter) {
126                             imageList.remove(imageElementCounterMax--);
127                     }
128                     preImage = new BufferedImage(curImage.getWidth(), curImage
129                                     .getHeight(), curImage.getType());
130                     Graphics g = preImage.getGraphics();
131                     g.drawImage(curImage, 0, 0, null);
132                     imageList.add(++imageElementCounter, preImage);
133                     imageElementCounterMax = imageElementCounter;
134             }
135     }
```

```
136
137         public void mouseDown(MouseEvent e) {
138                 Point p = e.getPoint();
139                 showStatus("mouse Down to " + p);
140                 curX = p.x;
141                 curY = p.y;
142         }
143
144         // And two methods from MouseMotionListener:
145         public void mouseDragged(MouseEvent e) {
146                 Point p = e.getPoint();
147                 showStatus("mouse Dragged to " + p);
148                 curX = p.x;
149                 curY = p.y;
150                 if (inDrag && initialized) {
151                         repaint();
152                 }
153         }
154
155         public void undo() {
156                 if (imageElementCounter > 0 && initialized) {
157                         curImage = imageList.get(--imageElementCounter);
158                         preImage = new BufferedImage(curImage.getWidth(), curImage
159                                         .getHeight(), curImage.getType());
160                         Graphics g = preImage.getGraphics();
161                         g.drawImage(curImage, 0, 0, null);
162                         curImage = preImage;
163                         repaint();
164                 }
165         }
166
167         public void redo() {
168                 if (imageElementCounter < imageElementCounterMax && initialized) {
169                         curImage = imageList.get(++imageElementCounter);
170                         preImage = new BufferedImage(curImage.getWidth(), curImage
171                                         .getHeight(), curImage.getType());
172                         Graphics g = preImage.getGraphics();
173                         g.drawImage(curImage, 0, 0, null);
174                         curImage = preImage;
175                         repaint();
176                 }
177         }
178
179         public void paint(Graphics g) {
180                 if (inDrag) {
181                         Graphics2D g2 = (Graphics2D) curImage.getGraphics();
182
183                         if (greenButton.isSelected()) {
184                                 g2.setColor(Color.GREEN);
185                         } else if (blackButton.isSelected()) {
186                                 g2.setColor(Color.BLACK);
187                         } else if (redButton.isSelected()) {
188                                 g2.setColor(Color.RED);
189                         }
190                         if (pensizebutton1.isSelected()) {
191                                 g2.fillOval(curX - 10, curY - 10, 20, 20);
192                         } else if (pensizebutton2.isSelected()) {
193                                 g2.fillOval(curX - 10, curY - 10, 10, 10);
194                         }
195                         g2.dispose();
196                 }
197                 g.drawImage(this.curImage, 0, 0, this);
198         }
199
200 }//end class
201
```

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 05. Juni 2008
Ort, Datum

_____
Unterschrift