Hochschule für Angewandte Wissenschaften

Hamburg

Fachbereich Elektrotechnik und Informatik

Studienarbeit

Konzeption eines Frameworks für Kamerabildanalyse

Ilia Revout Matr.Nr.: 1560533

Betreuer : Herr Prof. Dr. Von Luck

Inhalt

1 PROLOG 4

1.1 BILDBEARBEITUNG 4

1.1.1 Geschichte der Bildbearbeitung 4

1.1.2 Ebenen der Bildbeschreibung 4

1.1.3 Image understanding 5

1.2 PROBLEM-LÖSUNG SITUATION 6

1.3 WIE ICH ZU DIESEM THEMA GEKOMMEN BIN 6

2 DIPLOMARBEIT VON RAINER BALZEROWSKI 7

2.1 KURZBESCHREIBUNG 7

2.2 ERSTE ZIELE 8

2.3 DAS SCHLUSSERGEBNIS 8

3 VISIONEN 9

3.1 DIE UMGEBUNG 9

3.2 FILTER 10

3.2.1 Was sind Filter? 10

3.2.2 Filter in der Umgebung 10

4 FRAMEWORK 11

4.1 WAS MUSS FRAMEWORK LEISTEN 11

4.1.1 Erleichterung der Filterentwicklung 11

4.1.2 Filterkombinationen 11

4.1.2.1 Filter nacheinander ausführen 12

4.1.2.2 Filter mit mehreren Quellen 12

4.1.3 Filtereinstellungen 13

4.1.4 Leichte Integration 14

4.1.5 Automatische Darstellung 15

4.1.6 Originalbildquelle 16

4.2 RAHMENBEDINGUNGEN FÜR DIE IMPLEMENTIERUNG 17

4.2.1 Software 17

4.2.1.1 Betriebssystem 17

4.2.1.2 Entwicklungsumgebung 17

4.2.2 Hardware 18

4.2.2.1 Minimale Voraussetzungen 18

4.2.2.2 Auswahl der Kamera 18

4.3 SOFTWAREARCHITEKTUREN (VORSCHLAG) 19

4.3.1 Entwurfsmuster 19

4.3.1.1 Was ist Entwurfsmuster? 19

4.3.1.2 Entwurfsmuster im Framework 19

4.3.2 UML *20*
4.4 WOZU KANN FRAMEWORK VERWENDET WERDEN? 22
4.4.1 Testumgebung für Filter *22*
4.4.2 Eigenständiges Programm *22*
5 RESÜMEE 23
6 LITERATURVERZEICHNIS 24

1 Prolog

1.1 Bildbearbeitung

1.1.1 Geschichte der Bildbearbeitung

Die Geschichte der Bildbearbeitung beginnt um 1960. Grosse Firmen und Forschungszentren fangen damals an, für die Grafikberechnung Computer zu benutzen. Es hat aber sehr lange gedauert, bis die Computeranimation in der Lage war zu zeigen, dass sie mehr kann, als zuvor ohne Computer möglich war. Die siebziger Jahre waren eine Zeit intensiver Forschung. Viele fundamentale Lösungen wurden damals gefunden. In den achtziger Jahren wanderten die Entwicklungen aus den Forschungslabors in die Industrie. Es entstanden neue Methoden in Architektur, Konstruktion, Medizin, Computergrafik und Animation. Da die Rechner damals nicht schnell genug waren, musste man stundenlang rechnen lassen, um einfache Ergebnisse zu erzielen.

Ein Blick auf gegenwärtige Anwendungen und Entwicklungstendenzen digitaler Bildbearbeitung macht deutlich, wie weit der Prozess technologischen Wandelns bereits fortgeschritten ist. Ein Leben ohne digital erzeugte, bearbeitete und per Datenleitung rund um den Globus versendbare Bilder scheint unmöglich zu sein. Die Bildverarbeitung ist sehr populär geworden. Man braucht nicht viel Zeit zu investieren, um die Berichte, die Forschungsbeschreibungen, die Programme aus dem Bereich zu finden. Das Internet boomt von solcher Information.

Unter [7] und [8] findet man hunderte von Links, die sich alle mit dem Thema beschäftigen.

1.1.2 Ebenen der Bildbeschreibung

Die Bildbearbeitung kann man in drei Ebenen unterteilen: Ikonische -, Geometrische - und die Bildinterpretation – ebene.

Ikonische Ebene: Das ist die Ebene der Bilder als physikalisches Signal, die Beschreibung von Helligkeiten und Farben als Funktion des Ortes im Bild.

Die geometrische Ebene: Die geometrische Szenenbeschreibung schildert die Anordnung von den im Bild

abgebildeten Szenenkomponenten und die Form der Szenenkomponenten.

Die Bildinterpretation: Auf der semantischen Ebene wird die Bedeutung einer im Bild dargestellten Szene beschrieben: Welche Objekte gibt es? Was tun sie? Welche Ziele verfolgen sie?

Die Teilgebiete der Computergrafik betreffen unterschiedliche Ebenen dieser Hierarchie und unterschiedliche Verarbeitungs-richtungen innerhalb der Hierarchie.
Die Prozesse der graphischen Datenverarbeitung sind also im Wesentlichen top-down Prozesse von der Beschreibung zum Bild, während die Prozesse des Bildverstehens bottom-up Prozesse vom Bild zur Bildbeschreibung sind. Das größte Problem beim Bildverstehen ist die mangelnde Eindeutigkeit: Jedes Bild kann das Abbild von unendlich vielen verschiedenen Szenen sein. Bei der grafischen DV dagegen ist die Lösung eindeutig, aber die Aufgabe besteht darin, Algorithmen zu entwickeln, die diese eindeutigen Lösungen in vertretbarer Zeit finden. [6]

1.1.3 Image understanding

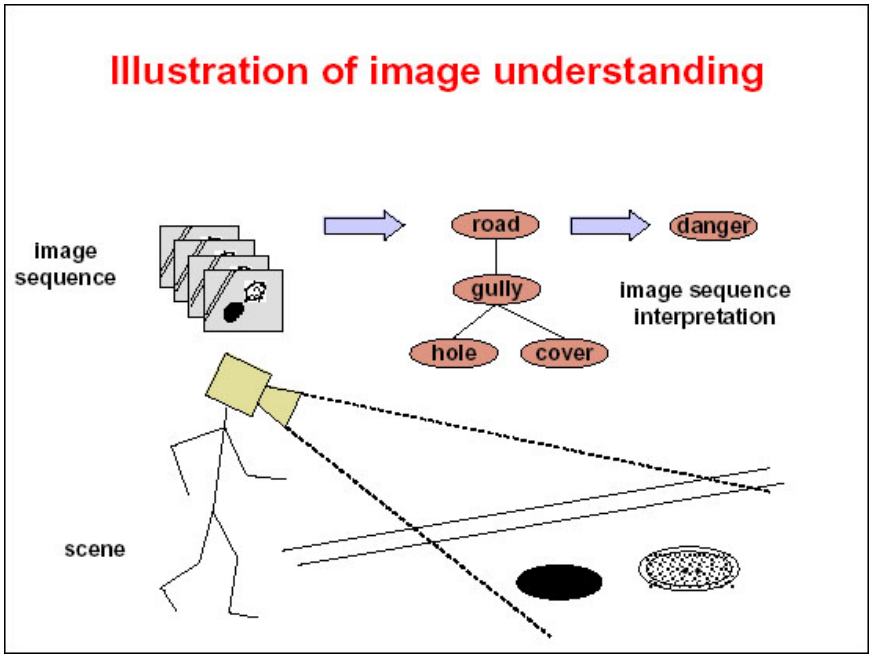

Abbildung 1 © Prof. Dr. Bernd Neumann

In der Abbildung 1 zeigt uns Prof. Dr. Bernd Neumann, was man unter ‚image understanding' verstehen kann. Wir haben Bilderfolge

(Image sequence) zur Verfügung. Gesucht ist ein Algorithmus, welche die Bilder analysiert und uns mitteilen kann, sobald bestimmter Muster entdeckt ist. Die Lösung dieser Aufgabe ist nicht einfach. Die befasst sich mit unterschiedlichen Ebenen der Bildbearbeitung.

1.2 Problem-Lösung Situation

Es ist offensichtlich, dass der Bereich der Bildbearbeitung sehr groß ist. Neue Anforderungen bringen mit sich auch neue Probleme. Zurzeit werden diese Probleme oft einzeln gelöst, was zu den Insellösungen führt. Die Programme sind meistens nicht anpassungsfähig, so dass es nicht möglich ist, andere Probleme damit zu lösen.

1.3 Wie ich zu diesem Thema gekommen bin

Im Wintersemester 2002-2003 habe ich ein Wahlpflichtfach „3-dimensionale Bilderfassung für den industriellen Einsatz “ absolviert, wo ich mich mit Bildbearbeitung auseinander gesetzt habe. Der Schwerpunkt lag bei den Filtern. Meine Aufgabe war bestimmte Filter zu schreiben und als übersichtliche API zur Verfügung zu stellen. Als schwierigste hat sich nicht das Filterschreiben, sondern das Testen herausgestellt. Ich musste mich mit solchen Problemen wie Bild laden, Pixel im Bild suchen, Veränderungen anzeigen auseinandersetzen. Auch andere Kommilitonen mussten die gleichen Probleme bekämpfen.

Da mich die Bildbearbeitung interessiert, habe ich mich entschlossen in die Richtung auch die Studienarbeit zu schreiben. Als Grundvorlage sollte dabei die Diplomarbeit von Rainer Balzerowski[1] dienen.

2 Diplomarbeit von Rainer Balzerowski

2.1 Kurzbeschreibung

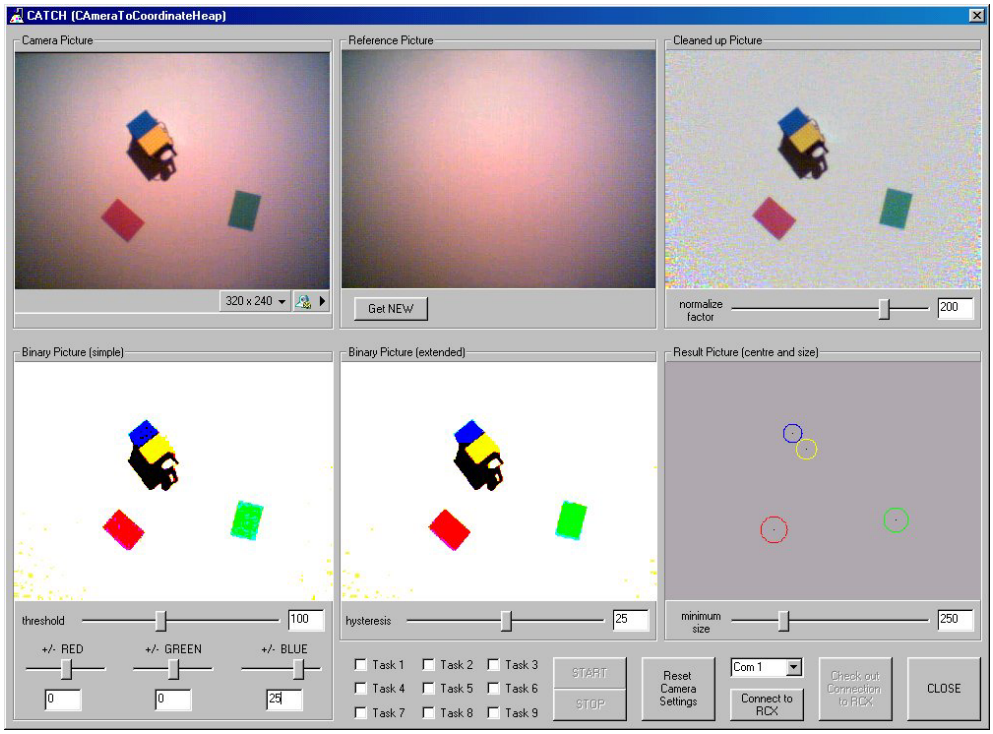

Abbildung 2

Dieser Abschnitt ist aus der Diplomarbeit von Rainer Balzerowski [1] entnommen. „Die Softwarelösung, die im Rahmen dieser Diplomarbeit entwickelt wird, sollte in der Lage sein, jeweils ein Objekt einer bestimmten Farbe zu beobachten. Die zu erkennenden Objekte können von einer der sechs Farben Rot, Grün, Blau, Gelb, Cyan oder Magenta sein. Flächen dieser Farben werden bestimmt und ihre Größe sowie ihr Schwerpunkt berechnet. Die Größe wird benötigt, um die größte Fläche einer Farbe zu bestimmen, falls mehrere Flächen einer Farbe vorhanden sind. Da nur ein Koordinatenpaar pro Farbe verwaltet wird, werden die Informationen einer kleineren Fläche der gleichen Farbe verworfen. Die Koordinatenpaare der einzelnen Farbflächen werden bezüglich der Auflösung der Kamera ermittelt und auf Anforderung, per Infrarotkommunikation an den RCX des Roboters geschickt.“

2.2 Erste Ziele

Das Programm von Rainer Balzerowski basiert sich auf **Logitech® QuickCam SDK.** Mit Hilfe von diesem SDK ist der Zugang zu allen Kameradaten sehr schnell und einfach. Aber nicht nur Vorteile bringt das SDK mit sich. Die Nutzung von nicht Logitech® - kompatiblen Kameras ist nicht möglich.

Meine Aufgabe war es also, das **Logitech® QuickCam SDK** zu ersetzen. Es sollte dabei nur für das Programm notwendige Funktionalität von SDK nachgebildet werden.

2.3 Das Schlussergebnis

Leider musste ich sehr schnell feststellen, dass das Programm sehr große Abhängigkeiten von SDK besitzt und die Umschreibung meiner Schätzung nach mehr Zeit in Anspruch nehmen wird, als die Neuimplementierung.

3 Visionen

3.1 Die Umgebung

Ich habe mir überlegt, was man nun implementieren muss, wenn man das Programm neu schreiben würde. Es wäre doch sehr nett, wenn man eine Umgebung hätte, wo man sich ausschließlich mit eigentlicher Arbeit beschäftigen könnte. Wenn man nicht darüber nachdenken braucht, wie man das Image von der Kamera in den Computer übernimmt, wie man das Image auf dem Bildschirm darstellt. Eine grobe Skizze solcher Umgebung könnte vielleicht wie folgt aussehen:

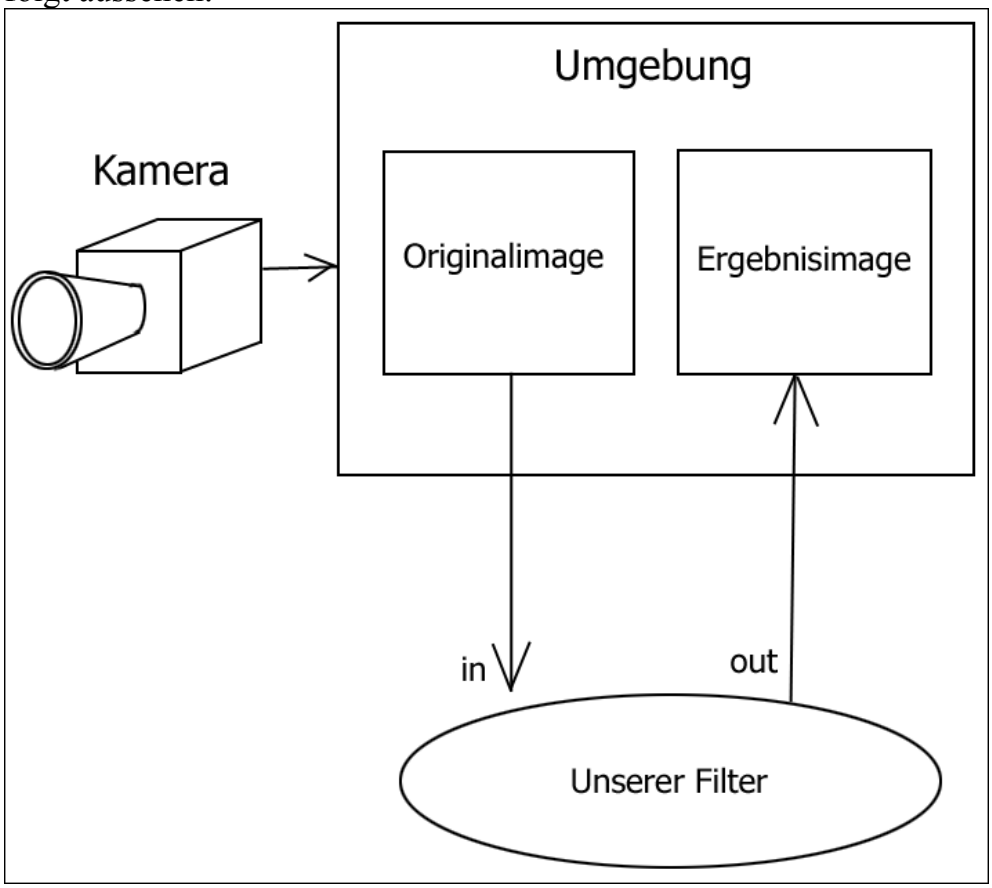

Abbildung 3

Wie man aus der Abbildung 3 gut sehen kann, brauchen wir uns nicht mehr um die Darstellung, sowie um Imagegrabbing[1] zu kümmern.

[1] Grabbing ist der engl. Ausdruck für die exklusive Zuordnung eines Betriebsmittels im X-Window System.

3.2 Filter

3.2.1 Was sind Filter?

Bevor ich etwas ausführlicher über Umgebung berichten werde, müssen wir uns klar werden, was man eigentlich unter Filter zu verstehen hat.

Da dieser Studienarbeit sich mit der Bildbearbeitung beschäftigt, werde ich den Filter wie folgt definieren:

1. Filter kann aus einem Eingabebild mittels eines Algorithmus ein Ausgabebild erzeugen.

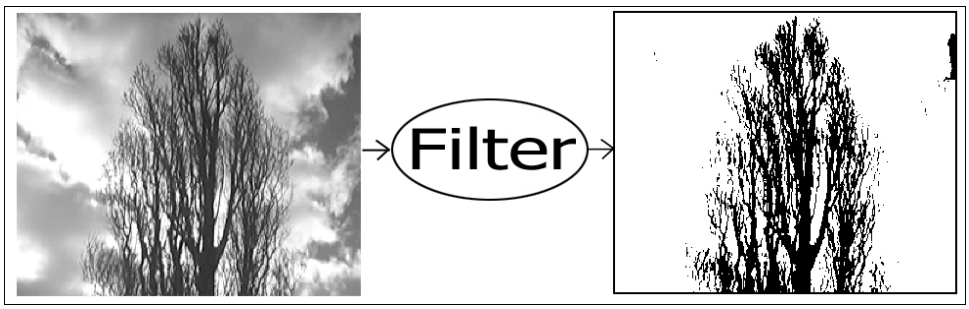

Abbildung 4

2. Es muss nicht unbedingt ein Ausgabebild am Ende rauskommen. Der Filter kann bestimmte Information aus einem Eingabebild gewinnen und die zur Verfügung bereitstellen.

Abbildung 5

3.2.2 Filter in der Umgebung

Welchen Filter soll die Visionsumgebung enthalten? Die Antwort lautet keinen! An dieser Stelle kommen wir etwas näher zu der oben dargestellten Umgebung. Die Idee besteht darin, eine Umgebung, ein Framework, zu entwickeln, so dass man beliebige Filter damit testen kann.

4 Framework

4.1 Was muss Framework leisten

4.1.1 Erleichterung der Filterentwicklung

Die Hauptaufgabe solches Frameworks sollte die Erleichterung der Implementierung sein. Das bedeutet, dass ganz bestimmte Grundwerkzeuge müssen sofort zur Verfügung stehen. Unter Grundwerkzeuge verstehe ich die Methoden, die z.B. mit Hilfe von X/Y – Koordinaten ein Pixel holt, bzw. ein Pixel setzt, Die Information über Eingabebild wie die Breite und die Hohe zurückliefert.
Auch über das Endbild sollte der Entwickler keine Gedanken machen. Das Framework soll automatisch die Vorlage erzeugen, so dass man diese Vorlage einfach mit den Daten fühlen kann.

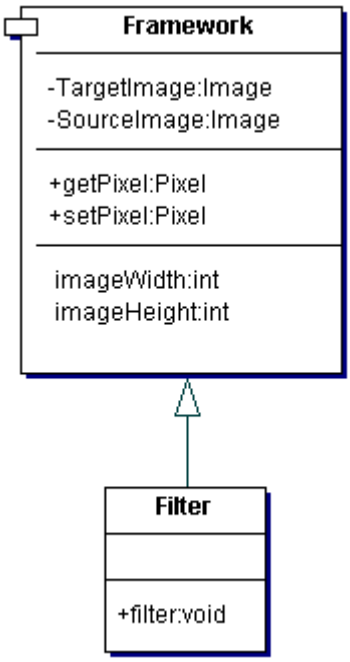

Abbildung 6

4.1.2 Filterkombinationen

Betrachten wir unsere Filter als Lego – Bausteine, so sollte es auch möglich sein, die schon vorhandenen Filter zusammen zu kombinieren.

4.1.2.1 Filter nacheinander ausführen

Als Grundfunktion sollte es möglich sein, alle im Framework vorhandene Filter in beliebiger Reihenfolge anzuwenden. Dabei soll jeder Filter sich auf dem Ergebnis von vorherigem Filter basieren. Der erste Filter nimmt als Quelle der Originalbild aus der Kamera. Man soll in der Lage sein, den gleichen Filter mehrmals an verschiedenen Stellen anzuwenden. Die Abbildung 7 demonstriert eine mögliche Filterkombination.

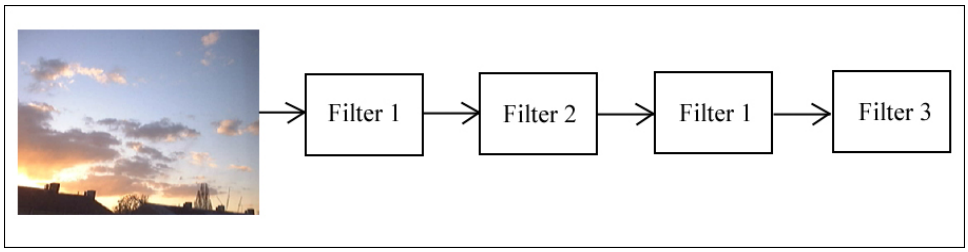

Abbildung 7

4.1.2.2 Filter mit mehreren Quellen

Oft braucht man ein Filter, der in der Lage ist mehrere Quellen zu analysieren und daraus ein Ergebnis zu liefern. Auch das muss Framework unterstützen. Das typische Beispiel ist die Farbfindung. Ausgegangen vom Originalbild werden mehrere Filter angewendet. Jeder Filter basiert sich auf dem Originalbild und hat die Aufgabe, bestimmte Farbe im Bild zu suchen, um nur die gesuchte Farbe darzustellen. Am Ende sollen alle Ergebnisse zusammengefügt werden, so dass nur ein Endbild, welches alle gesuchten Farben enthält, zur Verfügung steht. Die Abbildung 8 demonstriert dieses Verhalten.

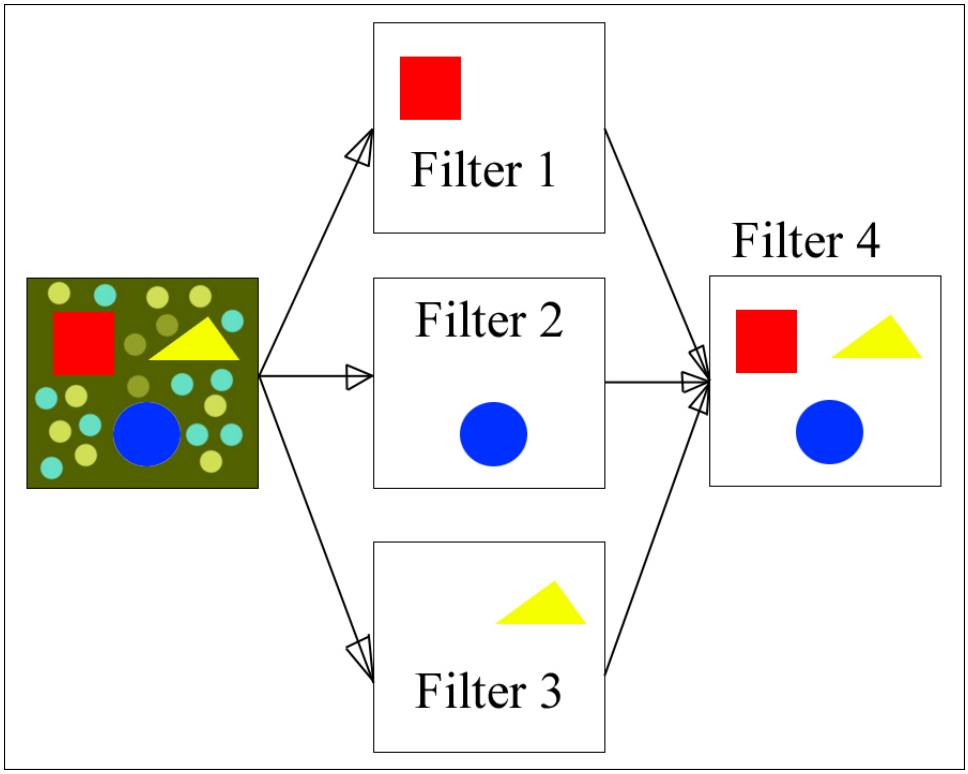

Abbildung 8

4.1.3 Filtereinstellungen

Nachdem man die Abbildung 8 näher angeschaut hatte, fragt man sich, ob es Sinn macht drei unterschiedliche Filter (Filter 1, Filter 2 und Filter 3) zu bauen. Alle drei Filter machen dasselbe und unterscheiden sich nur in ein Paar Suchkriterien. An dieser Stelle kommen wir zu nächster Anforderung, welche das Framework bewerkstelligen soll.

Es muss also möglich sein dasselbe Filter mit unterschiedlichen Einstellungen ausführen zu können. Die Abbildung 9 demonstriert dann das neue Verhalten.

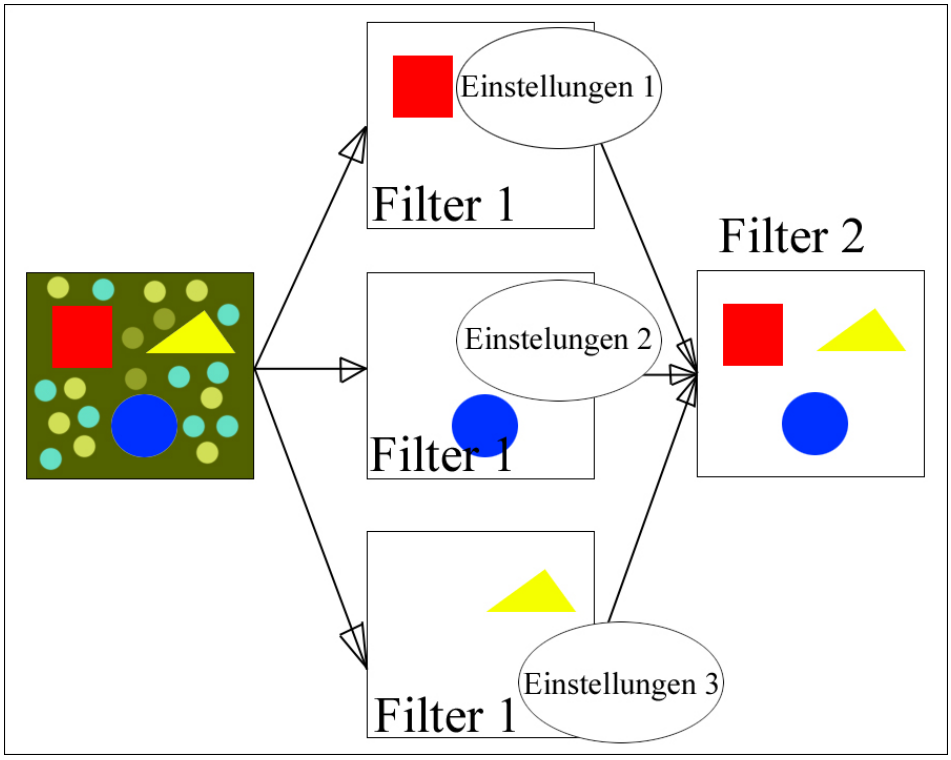

Abbildung 9

Weitere Problematik liegt darin, dass die Filter, in der Abbildung 9 sind es drei ‚Filter 1', nicht gleichzeitig ausgeführt werden und entsprechend auch nicht gleichzeitig fertig werden. Der ‚Filter 2' soll aber erst dann weitermachen, wenn alle Filter davor ihre Arbeit abgeschlossen haben.

Auch diese Überlegungen sollen den Entwickler nicht belasten, sondern vom Framework erledigt werden.

4.1.4 Leichte Integration

Wenn wir schon über Arbeitserleichterung, die das Framework leisten soll, sprechen, dann sollen wir ein Paar Wörter über die Integration verlieren. Unter Integration verstehe ich die Anbindung der Filter ins Framework. Es reicht nicht, dass das Framework uns bei der Entwicklung unterstüzt, es muss auch sehr einfach sein, fertig geschriebene Filter zu testen. Es soll möglich sein, den Filter im Framework zu registrieren. Alle registrierten Filter sollen dann zur Verfügung stehen, so dass man mit der Zeit eine Sammlung mit Filter bekommt, die alle miteinander kombinierbar sind. Damit wird es

möglich sein, neue Ideen schnell zu testen, ohne eine Zeile Code geschrieben zu haben.

4.1.5 Automatische Darstellung

Wie testet man einen Filter? Am einfachsten geht es, wenn man das Ergebnis anschauen könnte. Deswegen soll das Framework dies auch können. Es soll möglich sein, das Originalbild, sowie das Ergebnis von jedem einzelnen Filter, der angewendet wird, anzuschauen. Wie soll es aber funktionieren? Es gibt mehrere Vorgehensweisen. Zuerst kann man für jeden Filter eigenes Fenster erzeugen. Nachteil dabei besteht darin, dass man sehr schnell den Überblick verlieren kann. Die zweite Lösung wäre dann eine Art Weiche zu haben, mit deren Hilfe es möglich wäre, einfach zwischen Filtern zu wechseln. Die Abbildung 10 demonstriert die zweite Lösung.

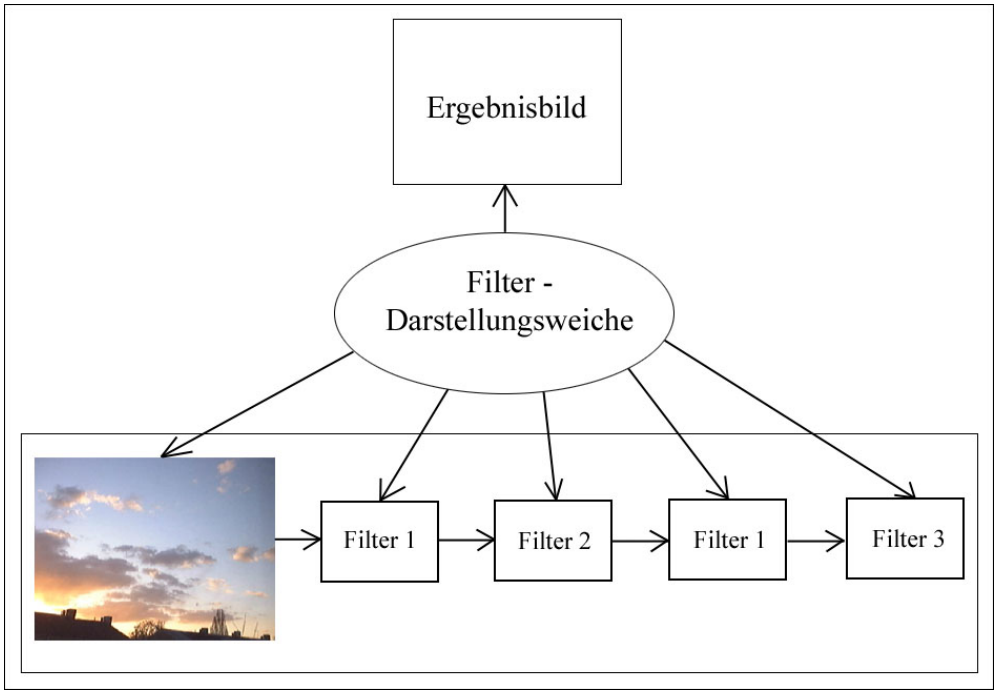

Abbildung 10

4.1.6 Originalbildquelle

Da es um die Echtzeitbildbearbeitung geht, müssen wir noch die Videoquelle besprechen. Meiner Meinung nach soll es für den Entwickler gleich sein, wie das Bild geholt wird. Man geht einfach davon aus, dass das Quellbild schon zur Verfügung steht und man damit sofort loslegen kann.

Weil das Framework universell eingesetzt werden soll, muss es in der Lage sein möglichst viele Kameras anzusprechen. Es wäre zum Vorteil, wenn auch unterschiedliche Schnittstellen (USB1.0, 1.1,2.0; FireWire etc.) unterstützt würden.

4.2 Rahmenbedingungen für die Implementierung

4.2.1 Software

4.2.1.1 Betriebssystem

Es geht um das Betriebssystem, unter welchem das Framework implementiert sein kann. Die Anforderung, möglichst viele Kameras, sowie die Schnittstellen ansprechen zu können, lassen uns nicht viel zum Auswahl. Das Framework soll möglichst einfach eingesetzt werden. Das heißt, dass wir an den Treibern für die Schnittstellen und für die Kameras angewiesen sind. Leider werden viele Kameras immer noch nur mit Windows®-Treiber ausgeliefert.

Weiteres Pluspunkt für Windows® (98,Me, 2000, XP) ist die von Microsoft® entwickelte „Video for Windows®“ (VfW) – Schnittstelle, die den Zugriff auf beliebige sich im System befindende Kamera ermöglicht, falls diese von VfW® unterstützt wird.

4.2.1.2 Entwicklungsumgebung

Unter Windows kommen zwei Entwicklungsumgebungen in Frage. Das erste ist Borland® C++ Builder und das zweite Visual C++ von Microsoft®. Beide Entwicklungsumgebungen erlauben die Verwendung von MFC (Microsoft Foundation Class). Diese Library bietet alle notwendigen Klassen für die GUI (Graphical User Interface), sowie erleichtert die Arbeit mit Dateien und stellt unterschiedliche Collections zur Verfügung.

Beide Entwicklungsumgebungen eignen sich gut für diese Aufgabe. Leider muss man die Entscheidung ganz am Anfang treffen, da die Kompatibilität nicht gewährleistet ist und die Migration sehr viel Zeit in Anspruch nehmen kann.

4.2.2 Hardware

4.2.2.1 Minimale Voraussetzungen

Wir brauchen nur die minimalen Voraussetzungen für die Hardware anzuschauen. Die obere Grenze existiert nicht. Die Bildbearbeitung braucht sehr viel Rechenkapazität, so dass davon nie genug vorhanden sein kann.

Damit die Implementierung, sowie das Testen jedoch möglich sind, brauchen wir eine entsprechende Hardwareumgebung. Diese minimalen Hardwarevoraussetzungen hängen direkt von Betriebssystem, Entwicklungsumgebung sowie Kamera ab. Jeder Hersteller von diesen Produkten hat eine Beschreibung solcher Hardwarevoraussetzungen. Nachdem die Entscheidung über die oben genannten Komponenten getroffen ist, müssen die entsprechenden Voraussetzungen verglichen werden, und davon die maximalen als unsere minimalen Voraussetzungen genommen werden.

4.2.2.2 Auswahl der Kamera

Eine Kamera soll für die Implementierung minimalen Voraussetzungen unterliegen. Entscheiden ist, dass diese Kamera, genauer gesagt, ihrer Treiber, VfW – kompatibel ist. Die Auflösung bzw. die Anzahl der Bilder pro Sekunde ist für die Implementierung unwichtig. Das Framework muss kameraunabhängig geschrieben werden, so dass der Umstieg auf eine andere Kamera dem ‚Plug-And-Play' – Prinzip unterliegen wird. Besonders eignen sich für diesen Zweck die WebCams. Diese Kameras sind preiswert und liefern sehr vernünftige Auflösung.

Oft werden die Kameras mit eingebauten Filterfunktionen ausgeliefert. So wird z.B. der Weisabgleich oder die Helligkeit des Bildes schon in der Kamera durchgeführt. Um die Übertragung zu dem Rechner zu beschleunigen, werden die Daten auch noch komprimiert, so dass man am Ende ein sich von RGB24 unterscheidendes Format zur Verfügung bekommt. Dieses Problem kann mit Hilfe von einer zusätzlichen Programmschicht (Adapter) gelöst werden. Solche Adapter können für unterschiedliche Kameras geschrieben werden, so das Framework unabhängig vom Bildformat der Kamera bleibt.

4.3 Softwarearchitekturen (Vorschlag)

Wie kann man so etwas programmieren? Gibt es schon erprobte Technologien? Diese und auch andere Fragen versuche ich in diesem Kapitel zu beantworten.

4.3.1 Entwurfsmuster

4.3.1.1 Was ist Entwurfsmuster?

Christoper Alexander schreibt: "Jedes Muster beschreibt ein in unserer Umwelt beständig wiederkehrendes Problem und erläutert den Kern der Lösung für dieses Problem, so dass Sie diese Lösung beliebig oft anwenden können, ohne sie jemals ein zweites Mal gleich auszuführen." [2] Obwohl Christoper Alexander über Muster in der Architektur geschrieben hat, pass es ausgezeichnet auch für die Softwareentwicklung. Sehr oft wiederholen sich die Probleme. Auch die Lösungsansätze für sich wiederholende Probleme sind identisch. Alles spricht dafür, solche Lösungswege abstrakt zu beschreiben und für jede Lösung einen bestimmten Namen zu vergeben.

Diese Arbeit hat unter anderen auch Erich Gamma gemacht. In seinem Buch [2] beschreibt er 23 häufigsten Probleme und deren Lösungen.

4.3.1.2 Entwurfsmuster im Framework

Die Aufgabe der Frameworks ist unter anderem auch die universelle Einsetzbarkeit zu ermöglichen. Dabei ist man sofort an viele Probleme gebunden. Viele davon passen genau unter Muster des „Gamma-Pattern". Ein Beispiel dafür ist das Problem mit den neuen Filtern. Wie wird der neu geschriebene Filter automatisch aufgerufen? Das Framework kann doch nicht wissen, wie die Schnittstelle bei neuen Filtern aussehen wird. Doch, das kann er! Die Lösung wird ausführlich bei ‚Visitor-Pattern' beschrieben.

Die Entwurfsmuster sind sehr große Hilfe für jede Art Programmierung. Besonders aber, wenn es Framework entwickelt wird.

4.3.2 UML

Eine grobe UML - Skizze demonstriert den möglichen Framework-aufbau.

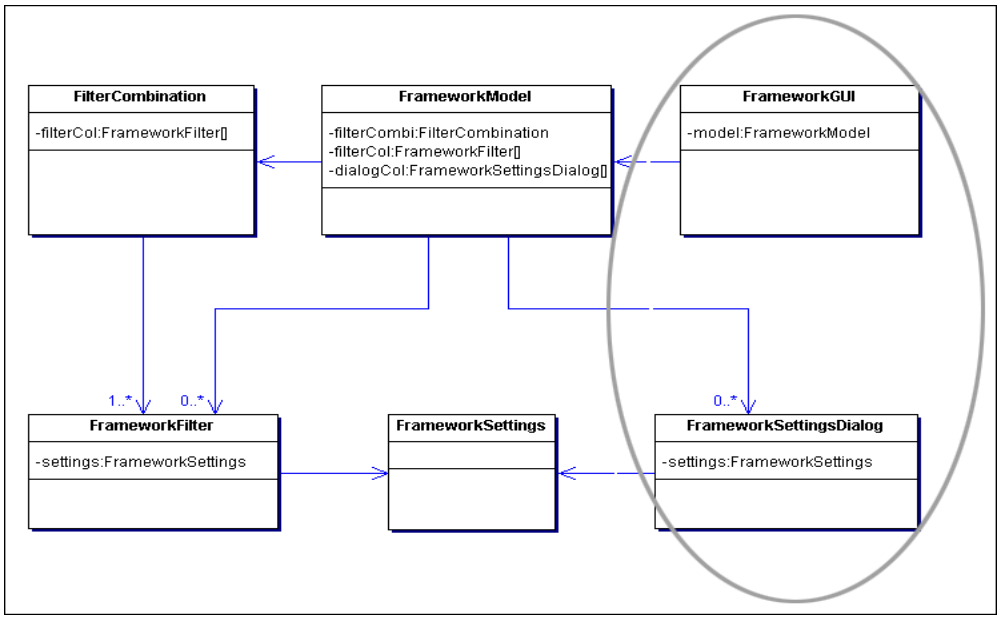

Abbildung 11

Die Klassen, die in der Abbildung in der unteren Reihe stehen, sind abstrakt und bieten damit die Schnittstelle für die Nutzung des Frameworks. Das Framework bietet die grafische Oberfläche an (die im Oval abgebildeten Klassen), die man aber nicht zu nutzen braucht.

Die Abbildung 12 zeigt die Implementierung der konkreten Klassen.

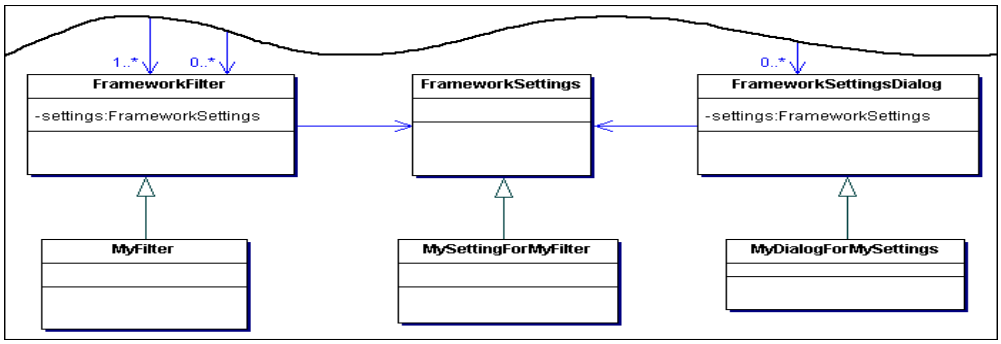

Abbildung 12

Fertiggeschriebene Klassen werden dann mit Hilfe von ‚register' – Methode, die vom Framework zur Verfügung gestellt wird, im System registriert. Beispielweise:

```
register(myFilter,mySettingsForMyFilter,MyDialogForMySettings)
```

Nach dem Registrieren sind die Klassen im Framework bekannt und stehen sofort zur Verfügung. Das Framework bietet die Methoden an, mit deren Hilfe man die vorhandenen Filter miteinander beliebig kombinieren kann. An dieser Stelle ist zu erwähnen, dass das Framework in der Lage sein soll, neue Instanzen von bekanten Filtern zu erzeugen. Das ermöglicht den gleichen Filter mehrmals an verschiedenen Stellen anzuwenden.

4.4 Wozu kann Framework verwendet werden?

4.4.1 Testumgebung für Filter

Die Hauptaufgabe des Frameworks besteht in der Unterstützung bei der Entwicklung der Filter. Oft hat man eine Idee, die man schnell ausprobieren möchte. Eine komplexe Anwendung dafür zu schreiben, lohnt sich meistens nicht. An der Stelle kommt das Framework zum Einsatz. Da es sehr einfach ist, ein Filter damit zu implementieren, dauert es nicht lange, bevor man das Ergebnis schon ansehen kann. Dazu kommt die Möglichkeit, neu entwickelter Filter zusammen mit schon vorhandenen zu testen.

4.4.2 Eigenständiges Programm

Natürlich kann man das Framework auch als selbständiges Programm nutzen. Dabei braucht man es nur um eigene Komponenten erweitern. Der Anwendungsbereich kennt wie die Fantasie keine Grenzen. Angefangen von Webcam - Anwendung, in der man die Bilder einfach zum Webserver schickt, damit die ganze Welt die anschauen kann, bis zu Steuerprogramm für Roboter. Dabei muss nur die Steuer, sowie ‚Logik' Module ins Framework eingebaut werden.

5 Resümee

Zum Schluss möchte ich gerne alles zusammenfassen. Wie ich schon ganz am Anfang erwähnt habe, gibt es schon ganze Menge von solchen Programmen. Man stellt sich sicherlich die Frage, wozu man dann das Framework braucht.

Der Bereich der Bildverarbeitung ist riesig groß. Es ist nicht möglich, ein globales Framework zu bauen, so dass keine Wünsche offen bleiben. Jedes Programm richtet sich meistens an die Lösung von bestimmten Problemen und kann nur mit Umständen an andere angepasst werden. Auch von mir vorgeschlagenes Framework ist für den bestimmten Bereich anwendbar. Als die Grundvoraussetzung für das Framework hat die am Anfang beschriebene Arbeit von Rainer Balzerowski [1] gedient. In dieser Studienarbeit wurde das Framework beschrieben, welches helfen soll, solche Aufgaben zu lösen. Das Hauptszenario ist also die Bilder aufzunehmen und in der Echtzeit bestimmte Muster daraus zu erkennen. Dabei sollte es keine Rolle spielen, ob die Kamera fest fixiert, oder am Roboter befestigt ist.

Ich hoffe, dass man mit Hilfe von diesem Framework noch eine kleine Lücke im Bereich der Mustererkennung für die Roboter schließen kann.

6 Literaturverzeichnis

[1] BALZEROWSKI, Rainer: *Realisierung eines Webcam basierten Kamera-Systems für mobile Roboter*. online. 2002. – URL http://www.informatik.haw-hamburg.de/~robots/papers.html – Zugriffsdatum: 11.03.2003

[2] Gamma, Erich : Entwurfsmuster : Elemente wiederverwendbarer objektorientierter Software. ISBN 3-89319-950-0. 1996

[3] Kahlbrandt, Bernd : Software-Engineering:Objektorientierte Software-Entwicklung mit der Unified Modeling Language. ISBN 3-540-63309-x. 1998

[4] Kruglinski, David : Inside Visual C++ 6.0. ISBN 3-86063-461-5. 1998

[5] Eine kurze Geschichte der digitalen Bildgenerierung und -bearbeitung URL http://lily.rz.hu-berlin.de/visuelle/thomas/historie.htm - – Zugriffsdatum: 11.04.2003

[6] Leonie S. Dreschler-Fischer und Werner Hansmann : Computergrafik, Sommersemester 2000 URL http://kogs-www.informatik.uni-hamburg.de/ ~graphik/cg/skript/cg-s01.pdf Zugriffsdatum: 13.04.2003

[7] The computer vision homepage. URL http://www-2.cs.cmu.edu/~cil/vision.html Zugriffsdatum: 29.04.2003

[8] „german robotics server“. URL http://wwwiaim.ira.uka.de/germrob/ Zugriffsdatum: 29.04.2003