MiroSot Vision-System

Bildverarbeitungsmodul des MiroSot-Projektes

HTL-Innsbruck
Abteilung Elektrotechnik

Autor: Strigl Daniel
Klasse: HE5c
Schuljahr: 1999/2000
Leitung: DI. Wesenjak Harald

01. Juni 2000

Inhaltsverzeichnis

MiroSot Vision-System

Bildverarbeitungsmodul des MiroSot-Projektes

1. Allgemeines

Bedingt durch die limitierte Größe der Fußballroboter ist ein vollständiges autonomes Verhalten der Roboter (zumindest mit den zur Zeit verfügbaren Technologien) nicht möglich. Es ist deshalb in den MiroSot-Spielregeln ausdrücklich vorgesehen, zusätzlich Informationen, wie die Position des Balles, der eigenen Spieler und die der Gegenspieler, sowie deren Ausrichtung innerhalb des Spielfeldes über den „Hostcomputer“ (Hauptrechner) an die Roboter weiterzugeben. Das hierzu verwendete Bildverarbeitungssystem kann also als Erweiterung der Sensorausrüstung der Spielroboter angesehen werden.
Die Anwendung von Bildverarbeitungssystemen zur Objekterkennung basiert üblicherweise auf Kantenerkennung oder auf einer Graustufenberechnung. Bei Objekten mit ähnlicher Größe und vergleichbaren Helligkeit/Kontrastwerten, mit beidem ist im Roboterfußball zu rechnen, sind die obengenannten Methoden nicht zielführend. In diesen Fällen muß die Farbe der Objekte als zusätzliche Imformation verarbeitet werden.
Die Fußballroboter müssen deshalb ein definiertes „Trikot“ mit einer speziellen Farbinformation tragen. Als Algorithmus für die Bilderkennung wird für das von uns entwickelte Team die „Pixel-Such-Methode“ verwendet. Es müssen dabei 6 Farben klassifiziert werden: 2 Teamfarben (gelb und blau), 3 Roboterfarben und die Farbe des Spielballes (orange). Bei diesen Algorithmus werden alle Pixel des Spielfeldes untersucht, Pixel mit den entsprechenden Farben werden über ihre Koordinaten registriert und zusammengefaßt. Aus dem „Durchschnitt“ der Koordinatenwerte der einzelnen Farbgruppen kann die Position der Farbflächen und damit also die Position der gesuchten Objekte ermittelt werden.
Auch wenn dieses Verfahren langsam und umständlich erscheint können dennoch Verarbeitungszeiten von ca. 30ms erreicht werden. Natürlich stellt diese Art der Berechnung auch sehr hohe Anforderungen an die Beleuchtung des Spielfeldes.

2. Aufgabe des Systems

Die Aufgabe des Bildverarbeitungssystems besteht nun darin das laufende Spielgeschehen über eine Farb-CCD-Kamera aufzuzeichnen und aus den eingelesenen Bilddaten dann die relevanten Informationen zu extrahieren.

Zu diesen Informationen gehören folgende Eigenschaften:
- Position der Fußballroboter und des Spielballes
- Ausrichtung der eigenen Roboter innerhalb des Spielfeldes
- Geschwindigkeit mit der sich die Spieler und der Spielball im Spielfeld bewegen

Diese ermittelten Eigenschaften sollen dann an alle anderen Module des Projekts, wie Steuer- oder Strategiemodul, übergeben werden. Dies ist für die Planung der nächsten Spielzüge und für die Steuerung der einzelnen Spieler notwendig. Der Austausch der Daten zwischen den einzelnen Modulen erfolgt dabei über einen gemeinsam genutzten Speicher, auf den alle Module des Projekts Zugriff haben.

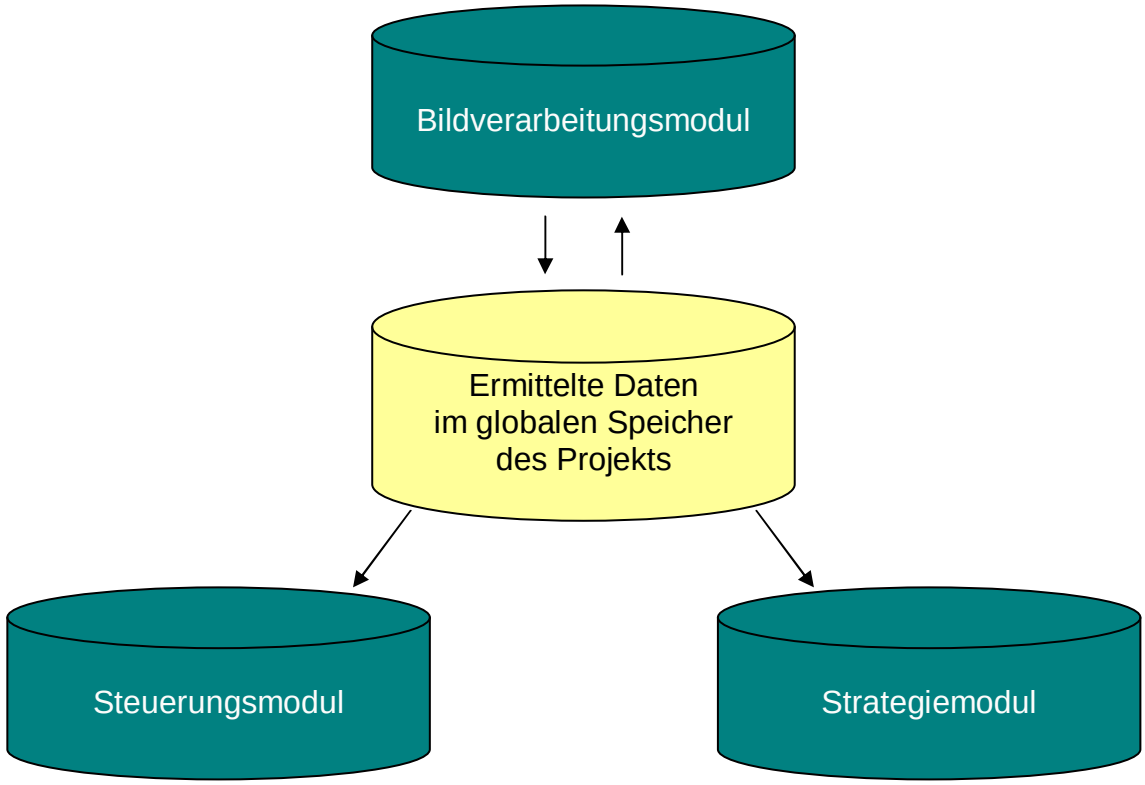

Abb. 1 Datenübergabe zwischen den einzelnen Modulen des Projekts

3. Aufbau des Systems

Um die oben genannten Aufgaben zu bewerkstelligen, sind dabei folgende Hardware-Komponenten erforderlich:

- **Ein handelsüblicher, 586 kompatibler PC als Zentralrechner.** Die Aufgabenbereiche des Zentralrechners sind neben der Bildverarbeitung, die Kommunikation mit den Robotern, sowie die Planung der nächsten Spielzüge und die Steuerung der einzelnen Spieler.
- **Eine Farb-CCD-Kamera** zum Aufzeichnen des Spielgeschehens.
- **Ein Farb-Frame-Grabber** zum Digitalisieren der Videodaten und zum Einlesen der digitalisierten Bilddaten in den Systemspeicher des Computers.

Der Aufbau und die erforderlichen Hardware-Komponenten sind in der nachfolgenden Abbildung dargestellt:

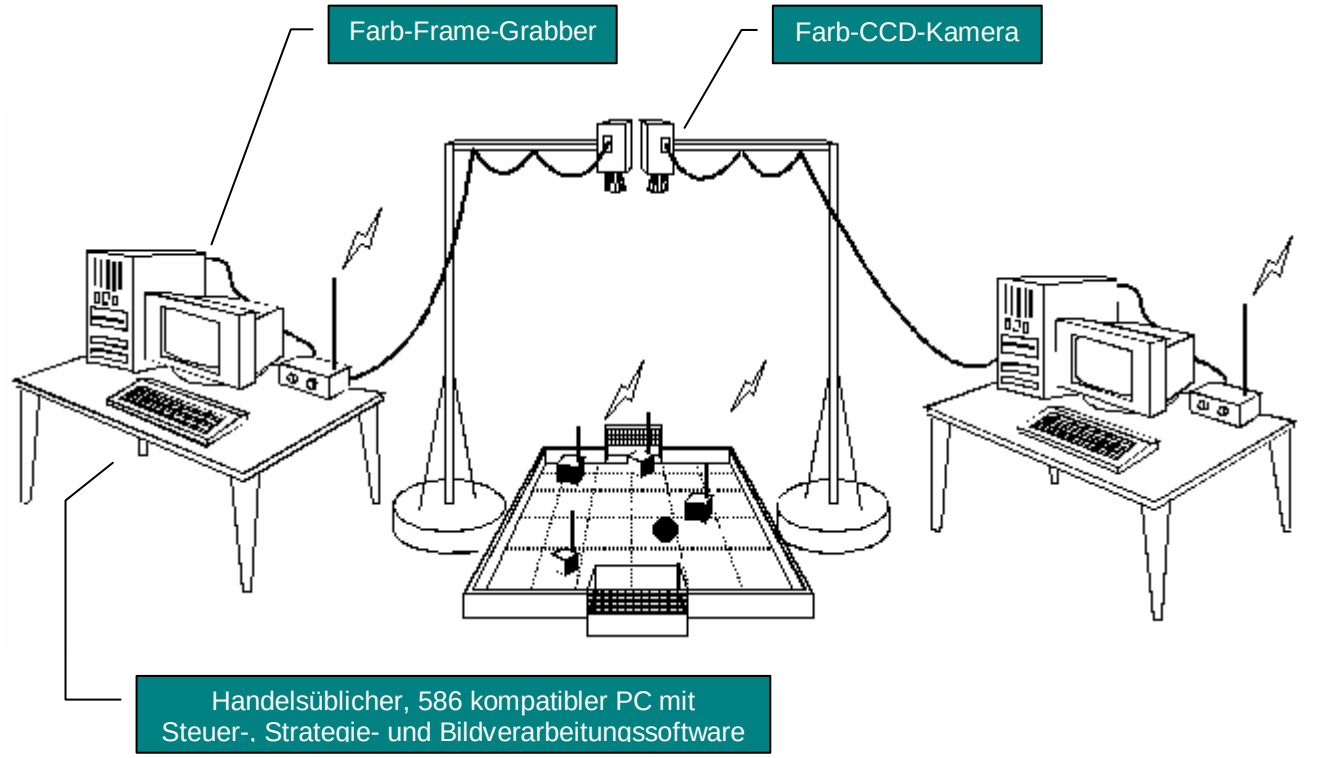

Abb. 2 Erforderliche Hardware und deren Aufbau

Folgende Hardware-Komponenten wurden von uns zur Umsetzung der obengenannten Aufgaben herangezogen:

Video-Kamera:
BLAUPUNKT CC 894 Video Kamera Rekorder

Frame-Grabber:
MuTech M-Vision 500 Video Digitizer

Computer (PC):
COMPAQ PC+ mit AMD-K6™ 3D Prozessor und 64,0 MB RAM

4. Arbeitsweise des Systems

Die Software des Bildverarbeitungssystems wurde in der Sprache C++, unter Verwendung einiger Zeilen Assembler, verfaßt. Die Verwendung von Assembler war notwendig, um eine hohe Verarbeitungsgeschwindigkeit zu erreichen, was oft über den Ausgang eines Spieles entscheidet. Nur die zeitaufwendigsten Funktionen des Programms wurden dabei in Assembler verfaßt, alle anderen Funktionen in C++. Erstellt wurde das Programm mit Hilfe des C++ Builder 4.0 der Firma Borland (Inprise). Als Betriebssystem wurde Windows 98 aus dem Hause Microsoft verwendet.
Der Prototyp der entwickelten Bildverarbeitungssoftware wurde vorerst nur für den Spielball und einen der Spieler ausgelegt. Die Bildverarbeitungssoftware ermittelt also aus den eingelesenen Bilddaten die Position und die Geschwindigkeit eines Spieler und des Spielballs. Neben der Position des Spielers innerhalb des

Spielfeldes wird noch dessen Ausrichtung festgestellt. Dies ist notwendig damit die Steuersoftware den Roboter richtig ausrichten kann. Die Feststellung von Position, Ausrichtung und Geschwindigkeit der anderen Spieler erfolgt dabei analog, wird aber erst später benötigt. Bevor die Spieler zusammen agieren können müssen die Roboter einzeln getestet und deren Verhalten beobachtet werden.
Im folgenden soll die Arbeitsweise zur Ermittlung von Position, Ausrichtung und Geschwindigkeit des einen Roboters und des Spielballs innerhalb des Programms beschrieben werden. Die nachfolgende Abbildung zeigt dabei eine grobe Übersicht über die Arbeitsweise des Programms:

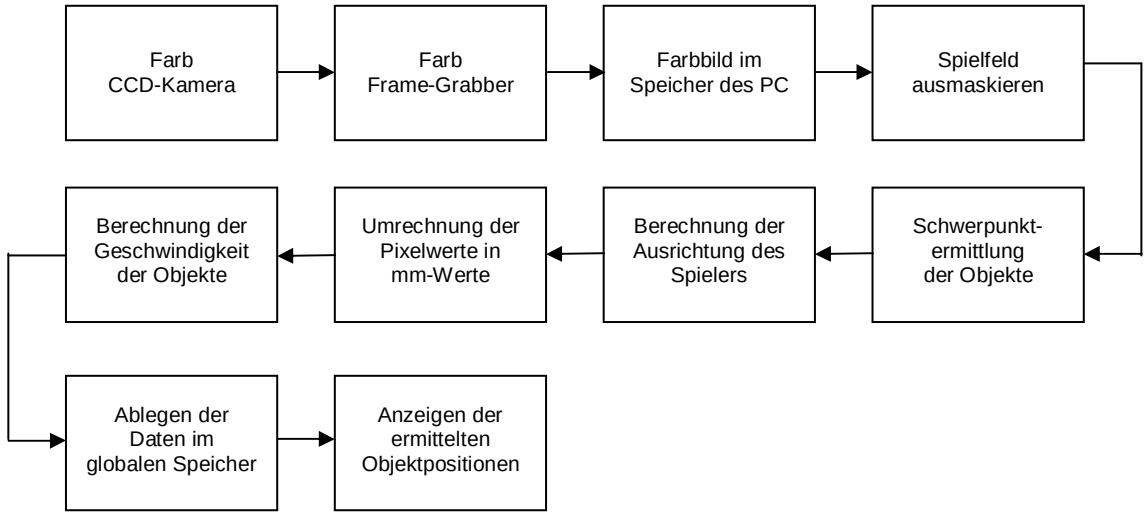

Abb. 3 Übersicht über die Arbeitsweise des Programms

Als nächstes folgt nun eine genauere Beschreibung aller Arbeitsprozesse des Systems.

4.1. Einlesen des Bildes:

Zuerst wird ein Bild des aktuellen Spielgeschehens über die Farb-CCD-Kamera eingelesen und mittels dem Farb-Frame-Grabber (Video-Karte) digitalisiert und im Speicher des Computer abgelegt. Die Digitalisierung der Bilddaten mittels Frame-Grabber ist notwendig, da die Kamera nur analoge Signale liefert, die Software aber nur mit digitalen Werten arbeitet. Das digitalisierte Bild wird als 24-Bit RGB-Farbbild (True-Color Bild) im Speicher des Computers abgelegt. Die Abmessungen des Bildes betragen dabei 320x200 Pixel, wobei jedes der Pixel aus 3 Farbanteilen besteht: einem Rot-, Grün- und Blauanteil (RGB). Jeder Farbanteil nimmt dabei 8 Bit (1 Byte) in Anspruch, wodurch jeder Farbwert eines Pixel aus einem 24-Bit Wert besteht. Jedes Pixel benötigt also 3 Byte, das gesamte Bild also einen Speicherplatz von 320x200x3=192000 Byte.

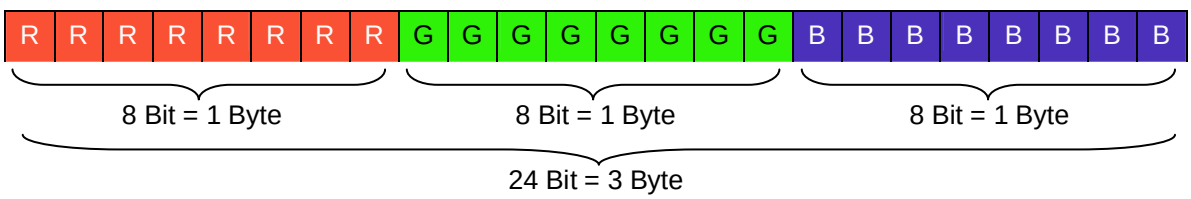

Abb. 4 Aufbau eines Pixels innerhalb eines 24-Bit RGB-Farbbild (True-Color Bild)

4.2. Ausmaskieren des Spielfeldes

Das digitalisierte Bild liegt nun zur weiteren Verarbeitung bereit. Als nächstes wird nun das Spielfeld ausmaskiert, also alle unwichtigen Teile des Bildes bis auf das Spielfeld aus dem Bild entfernt. Dies ist notwendig um das Geschehen außerhalb des Spielfeldes zu ignorieren, also Störungen von außen auszuschließen. Innerhalb des Spielfeldes kommen nur die zuvor definierten Farben vor, außerhalb des Spielfeldes könnten diese Farben allerdings auch auftreten und somit die Suche den Objekten, wie Spielball oder Spieler beeinflussen (z.B. ein oranges Kabel das direkt neben dem Spielfeld verlegt ist, beeinflußt die Suche nach dem orangen Spielball). Deshalb darf sich innerhalb des eingelesenen Bildes nur das aufgezeichnete Spielfeld befinden, alle anderen Pixel außerhalb der Spielfeldgrenzen werden dabei auf Schwarz gesetzt. Somit ist gewährleistet das sich im aufgezeichneten Bild nur das Spielfeld mit den zu suchenden Objekten (Spielball und Spieler) befindet. Das Ausmaskieren des Spielfeldes erfolgt dabei durch eine UND-Verknüpfung mit einer Spielfeldmaske. Die Spielfeldmaske wurde dabei im vorhinein während der Kalibrierung des Programms erstellt.

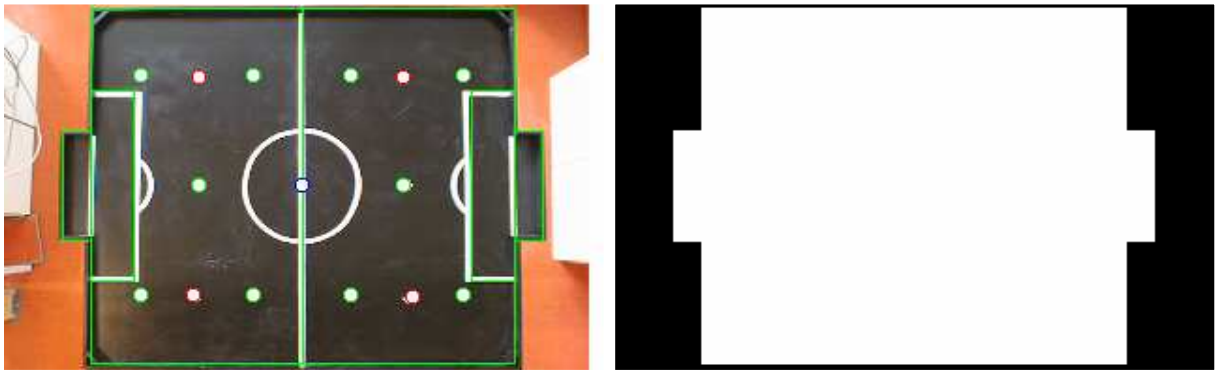

Abb. 5, 6 Die eingezeichneten Spielfeldgrenzen und die daraus entwickelte Spielfeldmaske

Bei der Verknüpfung des eingelesenen Bildes mit der Spielfeldmaske wird jedes Pixel des eingelesenen Bildes mit dem Pixel der Spielfeldmaske mittels einem „logischem UND“ verknüpft. Es werden also alle drei Farbanteile beider Bilder miteinander verglichen.

```
rot_bild   = rot_bild   & rot_maske
gruen_bild = gruen_bild & gruen_maske
blau_bild  = blau_bild  & blau_maske
```

Bei der Verknüpfung des eingelesenen Bildes mit der Spielfeldmaske werden nun alle Pixel beider Bilder durchlaufen und innerhalb der Spielfeldmaske nachgesehen ob das Pixel zum Spielfeld gehört oder nicht. Ist das Pixel teil des Spielfeldes wird es innerhalb des Bildes belassen, handelt es sich jedoch um ein Pixel das außerhalb der Grenzen liegt werden dessen Farbanteile auf den Wert 0 gesetzt, also innerhalb des Bildes schwarz eingezeichnet. Die folgenden Bilder zeigen die Ausmaskierung des Spielfeldes aus dem eingelesenem Bild mittels geeigneter Maske:

Eingelesenes Bild

Spielfeldmaske

Eingelesenes Bild mit ausmaskiertem Spielfeld

Abb. 7, 8, 9 Die Ausmaskierung des Spielfeldes aus dem Bild

4.3. Ermittlung der Schwerpunkte der gesuchten Objekte

Da sich nun im eingelesenen Bild nur noch die relevanten Bildteile befinden kann zur eigentlichen Suche nach den Objekten übergegangen werden. Wie im vorhinein schon erwähnt muß der Spielball orange sein und die Spieler ein definiertes „Trikot“ mit einer speziellen Farbinformation tragen, um innerhalb des Spielfeldes richtig identifiziert werden zu können. Das Trikot besteht dabei aus einer Teamfarbe (gelb oder blau) und einer Spielerfarbe.

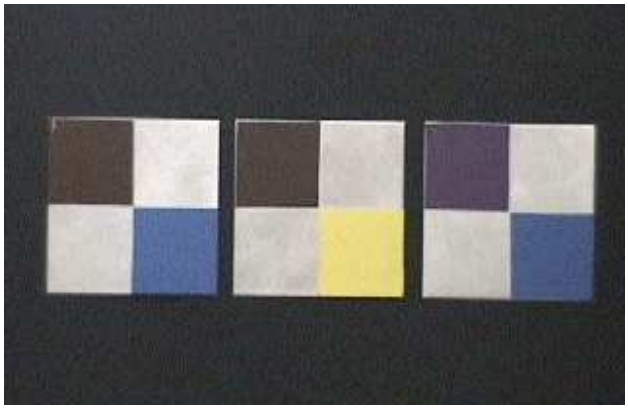

Abb. 10, 11 Farbtrikots der Spieler

Aus diesen beiden Farbmarkierungen kann nun die Position und Ausrichtung der Roboter ermittelt werden. Bei der Positionsbestimmung der Objekte (Farbflächen) werden nun alle Pixel des Bildes untersucht und alle Pixel mit den entsprechenden Farben über ihre Koordinaten registriert und zusammengefaßt. Ob ein Pixel nun zu einem der Objekte (Spielball, Teammarkierung oder Spielermarkierung) gehört oder nicht wird mittels einer oberen und unteren Grenze festgestellt. Liegt der Farbwert des Pixels nun innerhalb dieser Grenzen gehört es zum gesuchten Objekt, ansonsten nicht. Aus dem „Durchschnitt“ der Koordinatenwerte der einzelnen Farbgruppen kann dann die Position der Farbflächen und somit also die Position (eigentlich die Position des Schwerpunktes) der gesuchten Objekte ermittelt werden. Die Bestimmung des Flächenschwerpunktes $S(x_S, y_S)$ zur Positionsbestimmung der gesuchten Objekte erfolgt also nach folgendem Muster, wobei h_{yx} ein Bildpunkt des gesuchten Objektes ist:

Die Fläche des gesuchten Objektes ergibt sich folgerendermaßen:

$$F = \sum_y \sum_x h_{yx}$$

$$h_{yx} = \begin{cases} 1 \text{ wenn } y,x \text{ Bildpunkt des Objektes,} \\ 0 \text{ sonst.} \end{cases}$$

Die Koordinaten x_S und y_S des Flächenschwerpunktes werden berechnet durch:

$$x_S = \frac{1}{F} \cdot \sum_y \sum_x x \cdot h_{yx}$$

$$y_S = \frac{1}{F} \cdot \sum_y \sum_x y \cdot h_{yx}$$

$$h_{yx} = \begin{cases} 1 \text{ wenn } y,x \text{ Bildpunkt des Objektes,} \\ 0 \text{ sonst.} \end{cases}$$

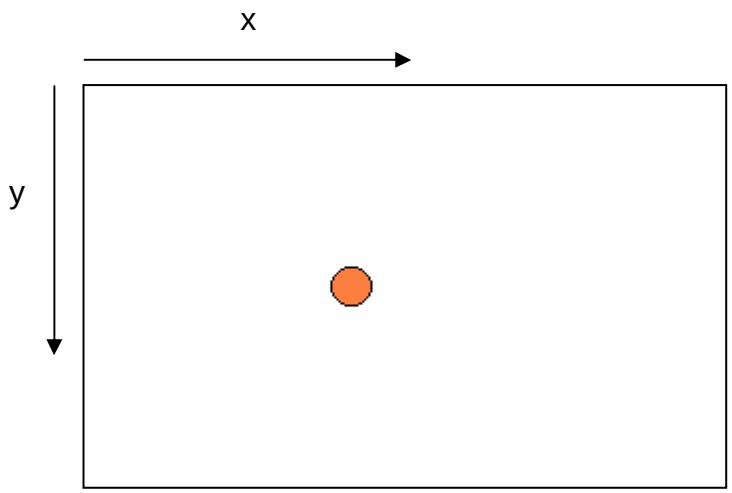

Abb. 12 Indizierung der Bildpunkte

Ist die Position des Spielballes sowie die Position der Team- und Spielermarkierung des „Trikots“ der Spieler bekannt, muß noch der Schwerpunkt des gesamten Spielers ermittelt werden. Das erfolgt durch eine Mittelwertbildung beider Positionen von Team- und Spielermarkierung:

$$x_{SP} = \frac{x_{TM} + x_{SM}}{2}$$

$$y_{SP} = \frac{y_{TM} + y_{SM}}{2}$$

x_{TM} ... x - Koor. der Teammarkierung

y_{TM} ... y - Koor. der Teammarkierung

x_{SM} ... x - Koor. der Spielermarkierung

y_{SM} ... y - Koor. der Spielermarkierung

x_{SP} ... x - Koor. des Spielers

y_{SP} ... y - Koor. des Spielers

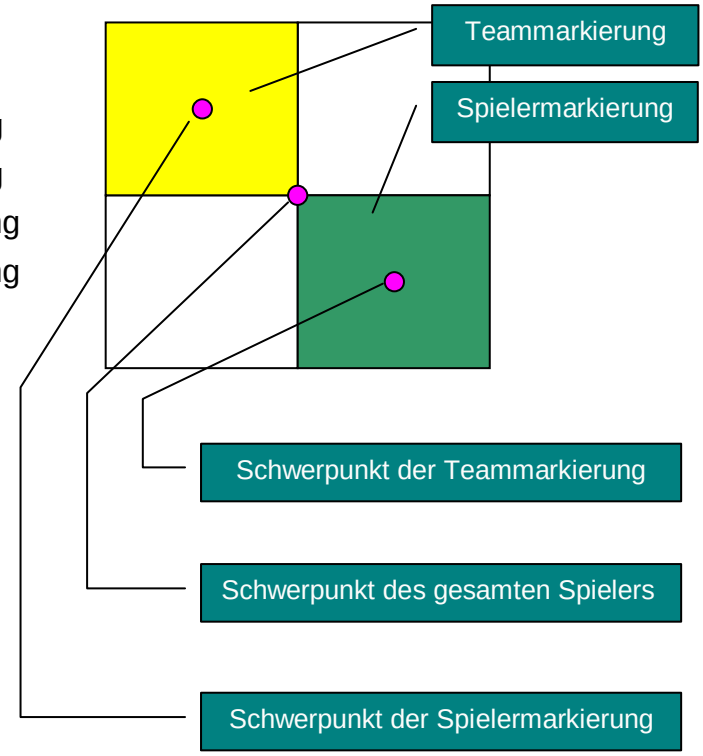

Abb. 13 Farbtrikot mit eingezeichneten Schwerpunkten

4.4. Berechnung der Ausrichtung des Spieles

Mit den ermittelten Objektpositionen kann nun auch die Ausrichtung der Spieler innerhalb des Spielfeldes berechnet werden. Die Ausrichtung der Spieler wird dabei über die Schwerpunkte von Team- und Spielermarkierung des Trikots berechnet. Die nachfolgende Abbildung zeigt wie dies bewerkstelligt wird.

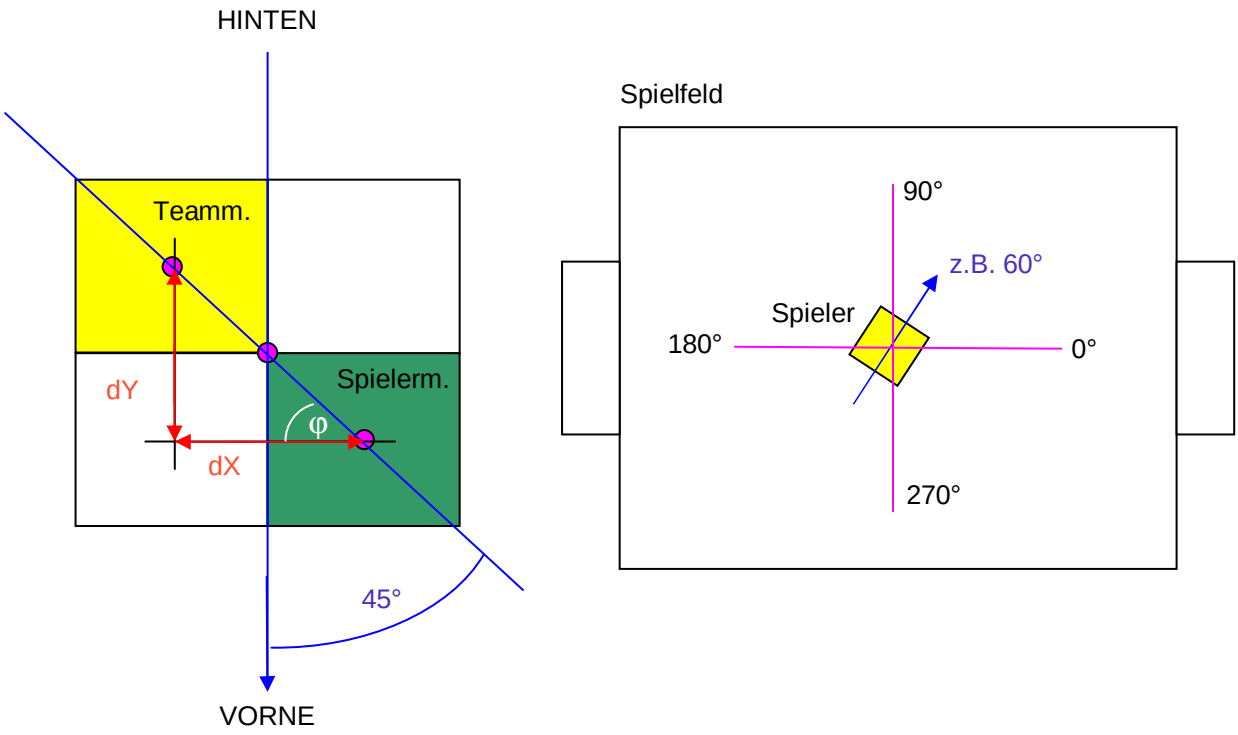

Abb. 14, 15 Algorithmus zur Berechnung der Ausrichtung der Spieler innerhalb des Spielfeldes

Zuerst wird nun der Abstand dX und dY der Teammarkierung zur Spielermarkierung berechnet:

$$dX = |x_{SM} - x_{TM}|$$
$$dY = |y_{SM} - y_{TM}|$$

Mit diesen beiden Abständen läßt sich nun der Winkel φ berechnen:

$$\varphi = \arctan\left(\frac{dY}{dX}\right)$$

Da der Winkel φ innerhalb des Dreiecks nie größer als 90° werden kann, für den Spieler aber eine Ausrichtung von 0° bis 360° möglich ist, muß der Winkel φ noch einem der 4 Quadranten (wie in Abb. 15 zu erkennen ist) zugewiesen werden, also um einen Winkelwert korrigiert werden. Um nun in Erfahrung zu bringen um welchen Wert der Winkel φ korrigiert werden muß, betrachtet man wieder die Koordinaten der Schwerpunkte von Team- und Spielermarkierung. Es wird ein Koordinatensystem durch den Schwerpunkt der Teammarkierung gelegt und nachgesehen innerhalb welchen Quadranten sich der Schwerpunkt der Spielermarkierung befindet, dadurch erhält man den Wert um den man den Winkel φ korrigieren muß. Daraus ergibt sich dann der Winkel φ‘.

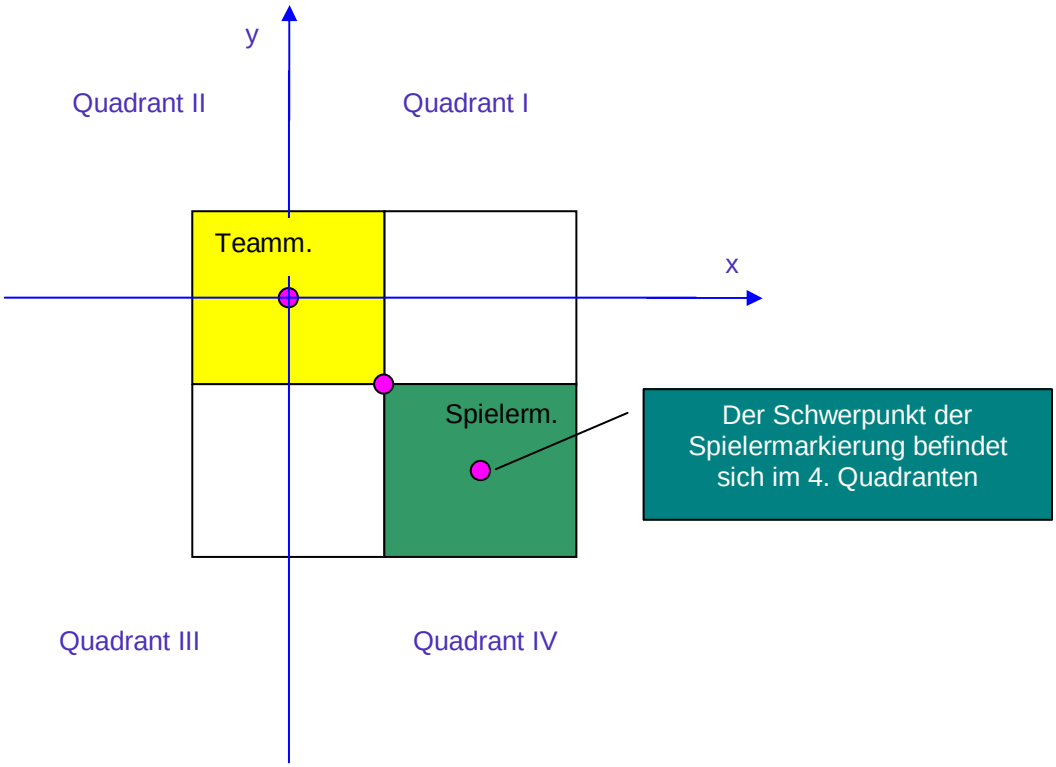

Abb. 16 Ermittlung des Quadranten in dem sich der Schwerpunkt der Spielermarkierung befindet

Der Winkel φ‘ läßt sich also folgendermaßen ermitteln:

$$\varphi' = \begin{cases} 180° - \varphi & \text{wenn } x_{SM} \le x_{TM} \text{ und } y_{SM} < y_{TM}, \\ 180° + \varphi & \text{wenn } x_{SM} < x_{TM} \text{ und } y_{SM} \ge y_{TM}, \\ 360° - \varphi & \text{wenn } x_{SM} \ge x_{TM} \text{ und } y_{SM} > y_{TM}, \\ \varphi & \text{sonst.} \end{cases}$$

Die Ausrichtung des Spielers φ_{SP} läßt sich nun ganz leicht in Erfahrung bringen indem man zum berechneten Winkel φ‘ einen Wert von 45° hinzuaddiert (siehe Abb. 14):

$$\varphi_{SP} = \varphi' + 45°$$

4.5. Umrechnung der Pixelwerte in mm-Werte

Als nächstes müssen nun alle Pixelwerte in mm-Werte umgerechnet werden. Um mit den Positionsdaten sinnvoll arbeiten zu können, müssen die Positionsdaten, die in Pixel angegeben sind, nun bezogen auf das Spielfeld in mm-Werte umgerechnet werden. Als Koordinatenursprung innerhalb des Spielfeldes dient dabei die Spielfeldmitte, die Objekte können sich also innerhalb von 4 Quadranten befinden (rechts oben, links oben, links unten, rechts unten). Dazu werden jedoch die Spielfeldgrenzen innerhalb des eingelesenen Bildes benötigt. Diese wurden jedoch schon beim Erzeugen der Spielfeldmaske während der Kalibrierung des Programms ermittelt und stehen somit zur Weiterverarbeitung bereit (siehe Abb. 5).

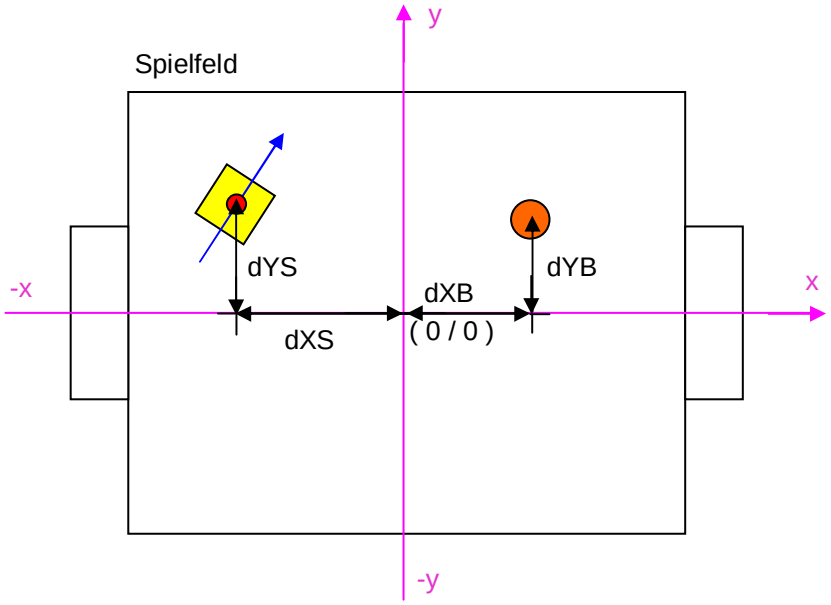

Abb. 17 Spielfeldmitte als Koordinatenursprung aller gesuchten Objekte

Da sich nun der Koordinatenursprung (0/0) nicht mehr links oben befindet, sondern nun die Spielfeldmitte den Ursprung darstellt, müssen zuerst die Abstände der Objekte zur Spielfeldmitte ermittelt werden. Da sich nun die Position der Objekte in alle 4 Quadranten befinden kann können die Koordinaten auch negative Werte besitzen. Die Abstände der Objekte zur Spielfeldmitte (Koordinatenursprung) lassen sich nun wie folgend berechnen:

$$dX = x_{SM} - x_O$$
$$dY = y_{SM} - y_O$$
$$x_{SM}, y_{SM} \ldots \text{Koordinate der Spielfeldmitte}$$
$$x_O, y_O \ldots\ldots \text{Koordinate des Objekts}$$

Als nächstes folgt nun die Umrechnung von Pixelwerten in mm-Werte. Dazu müssen die Abstände dX und dY mit einem Verzerrungsfaktor multipliziert werden. Dieser Verzerrungsfaktor stellt das Verhältnis von Pixelwerten zu mm-Werten innerhalb des Bildes dar. Somit lassen sich nun alle Pixelwerte aus dem Bild in mm-Werte der Realität umrechnen. Um den Verzerrungsfaktor berechnen zu können muß eine beliebige Länge innerhalb des Bildes sowohl in mm als auch in Pixel angegeben sein. Zum Beispiel könnte hier die Länge des Spielfeldrandes verwendet werden. Diese Länge ist sowohl als Pixelwert (aus den Spielfeldgrenzen) als auch als mm-Wert (aus den Abmessungen) bekannt. Der Verzerrungsfaktor läßt sich damit wie folgend ermitteln:

$$f = \frac{\text{Länge in Pixel}}{\text{Länge in mm}}$$

Die Umrechnung der Objektpositionen von Pixelwerten in mm-Werte erfolgt nun folgendermaßen:

$$dX' = dX \cdot f$$

$$dY' = dY \cdot f$$

4.6. Berechnung der Geschwindigkeit der Objekte

Als nächstes folgt nun die Berechnung der Geschwindigkeit, mit der sich die Objekte innerhalb des Spielfeldes bewegen. Bei der Berechnung der Geschwindigkeit wird zwischen jeder Positionsermittlung der Objekte die Zeit gemessen und der Weg berechnet, um den sich die Objekte in dieser Zeit bewegt haben. Das Verhältnis von zurückgelegtem Weg zu vergangenen Zeit entspricht dann der Geschwindigkeit mit der sich die Objekte bewegen. Die Geschwindigkeit wird in der Einheit mm/sek angegeben. Die vergangene Zeit wird dabei einfach über eine von Windows zur Verfügung gestellten Funktion ermittelt, der zurückgelegte Weg muß allerdings berechnet werden. Für den zurückgelegten Weg werden einfach die alte und neue Objektposition verglichen und daraus der Abstand berechnet. Dieser Abstand entspricht dann dem zurückgelegtem Weg. Um daraus dann einen annehmbaren Wert für die Geschwindigkeit zu erhalten, muß das System allerdings eine hohe Verarbeitungsgeschwindigkeit aufweisen und mehrmals pro Sekunde die Position der Objekte ermitteln. Die Geschwindigkeit läßt sich dann aber wie folgend ermitteln:

$$d = \sqrt{(x_{alt} - x_{neu})^2 + (y_{alt} - y_{neu})^2}$$

$$v = \frac{d}{t}$$

d ... zurückgelegter Weg

t ... vergangene Zeit

v ... Geschwindigkeit des Objekts

4.7. Ablegen der Daten im globalen Speicher des Projekts

Am Schluß werden nun die gesamten ermittelten Daten, wie Position, Ausrichtung und Geschwindigkeit im globalen Speicher des Projekts abgelegt. Dadurch haben alle anderen Module des Projekts darauf Zugriff, laufen aber unabhängig vom Bildverarbeitssystem, was einem modularen Aufbau des Projekts gewährleistet. Das Ablegen der Daten im globalen Speicher des Projekts erfolgt dabei durch das Schreiben der Daten in ein sogenanntes „Memory-Mapped-File“. Alle anderen Module können sich nun diesem „Memory-Mapped-File“ bedienen und die ermittelten Daten auslesen.

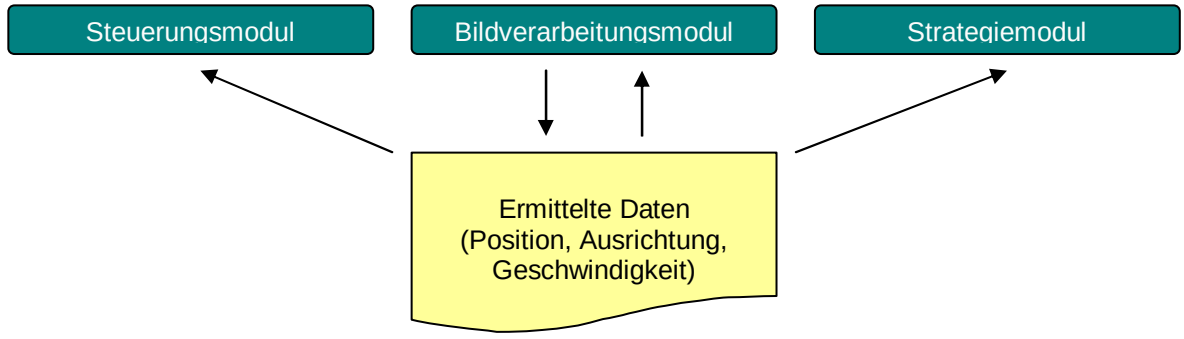

Abb. 18 Datenaustausch zwischen den Modulen des Projekts

4.8. Anzeigen der ermittelten Daten innerhalb des Spielfeldes

Falls gewünscht lassen sich die Position, Ausrichtung und Geschwindigkeit der Objekte in einem Fenster anzeigen. Dies ist oft sehr nützlich um die Richtigkeit der Daten sowie die Arbeitsweise des Programms zu überprüfen.

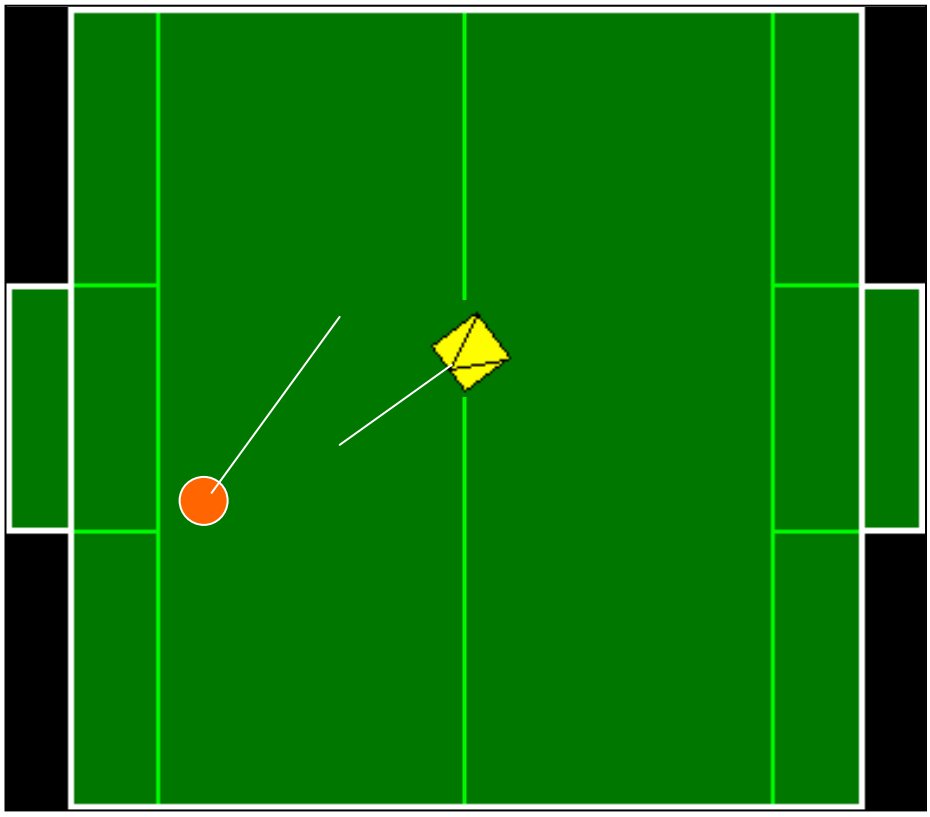

Abb. 19 Spielfeld mit eingezeichneten Objekten

5. Aufbau der Software

Die Software des Bildverarbeitungssystems wurde in modularer Form aufgebaut, sie besteht dabei aus mehreren Modulen die einzeln compiliert werden können und zusammen das gesamte Programm darstellen. Durch den modularen Aufbau des Programms läßt sich die Software leichter pflegen und die einzelnen Module einfacher testen und debuggen.

Die Software besteht dabei aus 5 Modulen:

- einem **Frame-Grabber-Modul** (MuTech M-Vision 500) zum Digitalisieren und Einlesen der aufgezeichneten Videodaten;
- einem **DirectDraw-Modul** zum Darstellen der eingelesenen Bilddaten innerhalb eines Window-Fensters in Echtzeit;
- einem **Bildverarbeitungsmodul**, das die Position, Ausrichtung und Geschwindigkeit der Objekte aus den eingelesenen Bilddaten ermittelt;
- einem **Memory-Mapped-File-Modul (MMF-Modul)** das die ermittelten Daten den anderen Modulen des Projekts zur Weiterverarbeitung bereitstellt;

- einem **Konfigurationsmodul** das alle Einstellungen des Programms, die während der Kalibrierung vorgenommen wurden speichert und bei Gebrauch wieder aufruft;

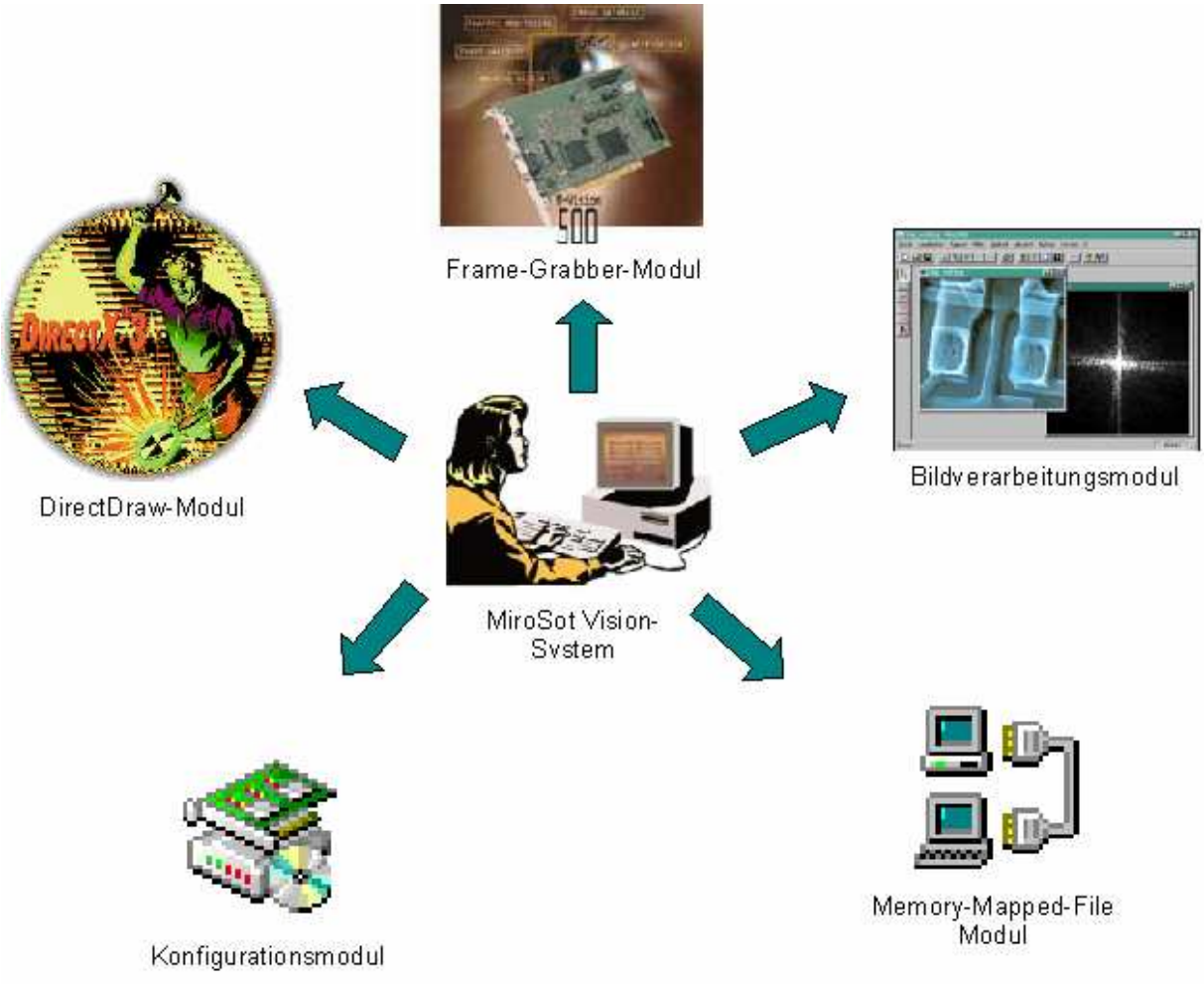

Abb. 20 Übersicht über die Softwaremodule des Systems

5.1. Frame-Grabber-Modul

Der Frame-Grabber übernimmt hierbei die Arbeit des Digitalisieren und Einlesen der aufgezeichneten Bilddaten. Die aufgezeichneten Bilddaten werden dabei von der Kamera zum Frame-Grabber übertragen und dort digitalisiert und anschließend im Speicher des Computers abgelegt. Das Frame-Grabber-Modul stellt hierbei nützliche Funktionen zum Umgang mit dem Frame-Grabber zur Verfügung. Das Modul arbeitet dabei mit den von der Firma MuTech zur Verfügung gestellten Funktionen, die mit dem Frame-Grabber ausgeliefert werden. Um die Funktionen des Frame-Grabbers nutzen zu können, muß nur die mitgelieferte DLL zum Projekt gelinkt werden, und die benötigte Include-Datei ins Projekt eingebunden werden. Damit stehen alle Funktionen des Frame-Grabbers dem Programmierer zur Verfügung.

Das entwickelte Frame-Grabber-Modul beinhaltet dabei folgende Funktionen:
- das Initialisieren des Frame-Grabbers,
- das Zurücksetzen des Frame-Grabbers,
- das einmalige Einlesen eines Farbbildes in den Speicher des Computers,

- das einmalige Einlesen eines Grauwertbildes in den Speicher des Computers,
- das zyklische Einlesen eines Farbbildes in den Speicher des Computers,
- das Speichern eines eingelesenen Bildes in eine BMP-Datei,
- das Laden einer BMP-Datei als Alternative zum Einzug eines Bildes über die Video-Kamera,
- das Aktivieren bzw. Deaktivieren des VCR-Modus des Frame-Grabbers,
- das Einstellen des Helligkeitswertes des eingelesenen Bildes,
- das Einstellen des Kontrastwertes des eingelesenen Bildes,
- das Einstellen des Sättigungswertes des eingelesenen Bildes,
- das Einstellen des zu verwendenden Eingangskanals des Frame-Grabbers,
- das Aktivieren bzw. Deaktivieren des Spiegel-Modus, also ob das Bild gespiegelt eingelesen werden soll oder nicht;

5.2. DirectDraw-Modul

Das DirectDraw-Modul dient zum Darstellen der eingelesenen Bilddaten und der ermittelten Objektpositionen innerhalb eines Window-Fensters in Echtzeit. Um die Darstellung von Objektpositionen und Bilddaten möglichst schnell abwickeln zu können wurden auf die DirectDraw-Funktionen des DirectX-Paketes von Microsoft zurückgegriffen. Bei DirectDraw handelt es sich um einen Teil des DirectX-Paketes, das sich mit dem graphischen Teil von DirectX befaßt. Bei DirectDraw handelt es sich um einen Video-Speicher-Manager, der dem Programmierer das schnelle und effiziente Manipulieren von Bilddaten innerhalb des Video-Speichers der Grafikkarte ermöglicht. DirectDraw bedient sich dabei direkt der Hardware der Grafikkarte und beschleunigt somit die Arbeit mit den Bilddaten erheblich. Die wichtigste Aufgabe des Moduls ist dabei das Darstellen der eingelesenen Bilddaten. Hierbei wird das eingelesene Bild vorerst in einen sogenannten „Hintergrund-Speicher“ geschrieben, der sich entweder im Systemspeicher des Computers oder im Speicher der Grafikkarte befindet. Anschließend wird dieses Bild direkt in den Videospeicher der Grafikkarte kopiert und somit zur Anzeige gebracht.

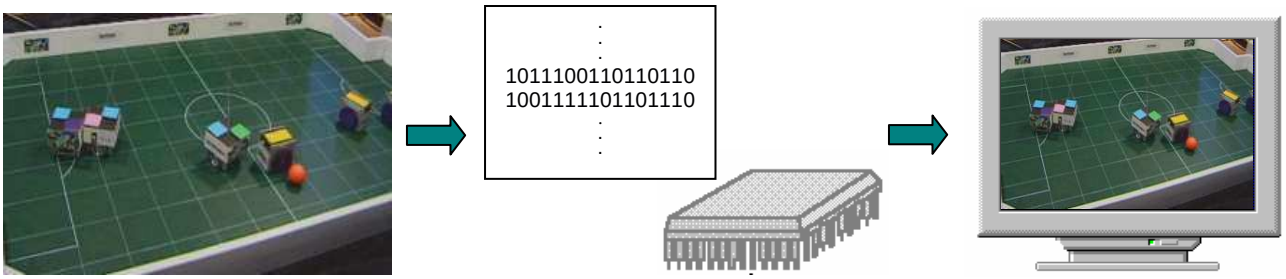

Abb. 21 Darstellen eines Bildes mittels DirectDraw

5.3. Bildverarbeitungsmodul

Das Bildverarbeitungsmodul übernimmt alle Aufgaben die zur Ermittlung der Spielfeldgrenzen sowie zur Ermittlung der Objekteigenschaften (Position, Ausrichtung und Geschwindigkeit) gehören. Innerhalb dieses Moduls steckt die wesentliche Arbeit des Systems, wie die Ermittlung von Position, Ausrichtung und Geschwindigkeit der Objekte. Da diese Funktionen eine hohe

Verarbeitungsgeschwindigkeit aufweisen sollten wurde der Großteil dieser Funktion in Assembler verfaßt.

Zu den wichtigsten Funktionen dieses Moduls gehören unter anderem:
- die Ermittlung der Spielfeldgrenzen,
- die Erzeugung einer Spielfeldmaske aus den zuvor ermittelten Spielfeldgrenzen,
- das Ausmaskieren der Spielfeldmaske aus den eingelesenen Bildern,
- die Ermittlung der Objektpositionen, deren Ausrichtung und Geschwindigkeit;

5.4. Memory-Mapped-File-Modul

Dieses Modul kümmert sich um das Schreiben der ermittelten Daten in den globalen Speicher des Projekts, um sie den anderen Modulen zugänglich zu machen. Der Datenaustausch unter Windows 95/98 zwischen verschiedenen Prozessen (Anwendungen) stellt ein an sich schwieriges Unterfangen dar, angesichts der abgeschotteten Adreßräume der Prozesse unter Windows 95/95. Unter Windows 95/98 erhält nämlich jeder Prozeß einen virtuellen Adreßraum von 4GB auf den er freien Zugriff hat. Aber nur der Prozeß der diesen Adreßraum für sich allokiert hat darf darauf zugreifen. Ein Zugriff auf einen fremden Adreßraum löst ein Exception aus und würde zum Absturz des Programms führen, die Daten können also nicht einfach in den Adreßraum (Speicher) einer anderen Anwendung geschrieben werden. Deshalb kann die Datenaustausch zwischen den Anwendungen nicht so einfach erfolgen, wie angenommen. Die Memory-Mapped-Files (MMF) zeigen einen Ausweg aus diesem goldenen Käfig, weil eine Datei mit ihrer Hilfe gleichzeitig in mehreren Adreßräumen abgebildet werden kann.

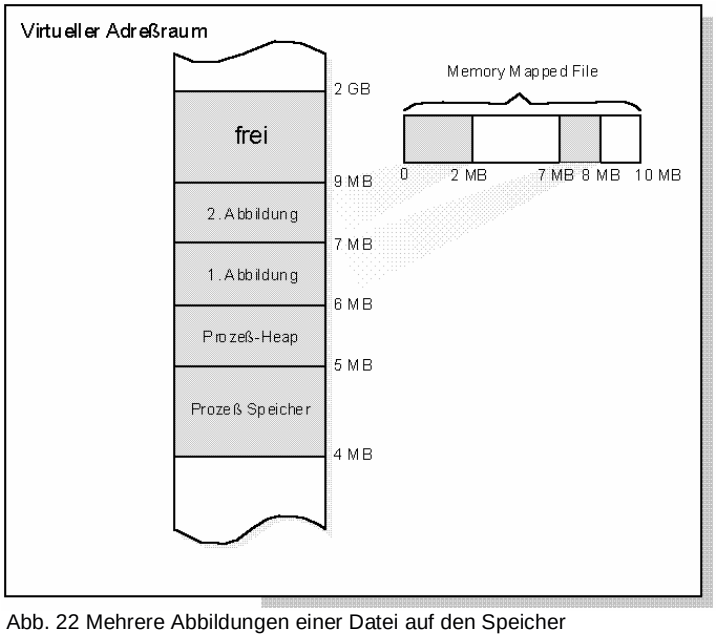

Abb. 22 Mehrere Abbildungen einer Datei auf den Speicher

Mehrere Prozesse partizipieren an einem MMF-Objekt, indem sie ihm einen Namen geben, der allen Beteiligten bekannt ist. Anschließend muß jeder das MMF-Objekt nur noch über diesen Namen öffnen, um Zugang zu erhalten. Jeder Schreibzugriff in den Adreßbereich, auf den die Datei abgebildet ist, erscheint unmittelbar in den Adreßräumen aller beteiligten Prozesse. Somit können die Daten bequem allen anderen Anwendungen zur Verfügung gestellt werden. Dabei muß nicht einmal einer

der beteiligten Prozesse eine Datei bereitstellen, in die der gemeinsam genutzte Speicher ausgelagert wird. Gibt man beim Öffnen eines MMF-Objekts keinen Dateinamen, sondern 0xFFFFFFFF an, wird die Datei aus dem Auslagerungsspeicher von Windows bezogen.

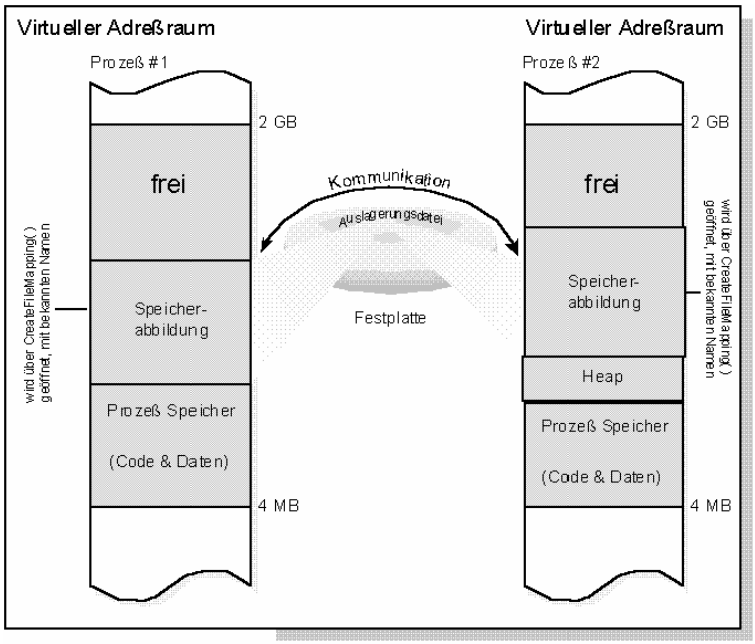

Abb. 23 Interprozeß-Kommunikation über Memory-Mapped-Files

5.5. Konfigurationsmodul

Das letzte Modul des Programms befaßt sich mit dem Speichern der getätigten Einstellungen während der Kalibrierung. Während dieser Kalibrierung des Programms werden etliche Einstellungen, wie Kontrastwert, Helligkeitswert, den zu verwendenden Eingangskanal des Frame-Grabbers, sowie die Farbwerte der zu suchenden Objekte usw. getätigt. Um diese Einstellungen nicht bei jedem Programmstart wiederholen zu müssen, werden diese in eine sogenannte INI-Datei geschrieben und von dort bei jedem Programmstart wieder geladen. Somit ist gewährleistet, daß die Einstellungen erhalten bleiben und nicht wieder von neuen eingegeben werden müssen. Der C++ Builder von Borland bietet hier vorgefertigte Funktion zum Umgang mit den INI-Files und erleichtert die Arbeit somit erheblich.

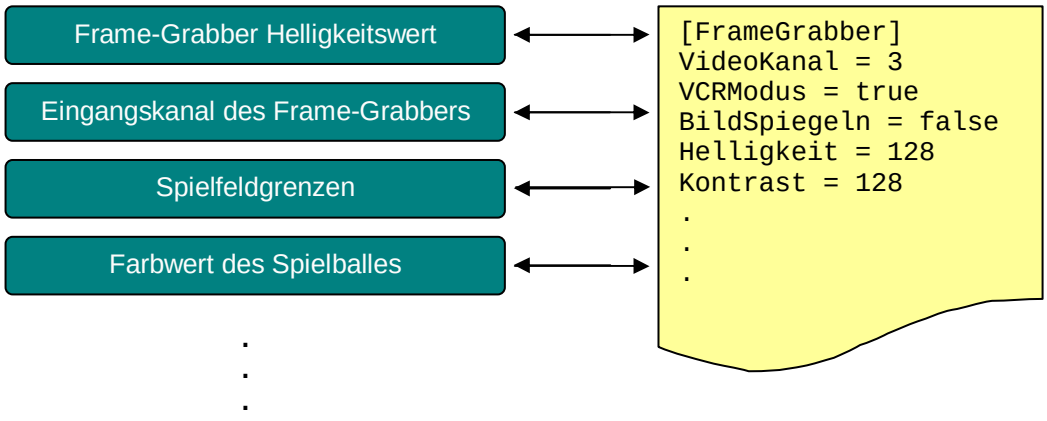

Abb. 24 Speichern der Einstellungen innerhalb einer INI-Datei

6. Bedienung der Software

Wird das Programm gestartet erscheint am Anfang ein kleines Fenster, das zum Warten auffordert. Innerhalb des Fensters ist eine Fortschrittsanzeige zu sehen. Nach Erscheinen des Fensters berechnet das Programm einige Daten die zur Arbeit mit dem Programm benötigt werden. Den Fortschritt der Berechnung ist mit Hilfe der Fortschrittsanzeige zu erkennen. Nach Abschluß aller Berechnungen verschwindet dieses Fenster wieder, es dient also nur der reinen Anzeige des Berechnungsfortschritts.

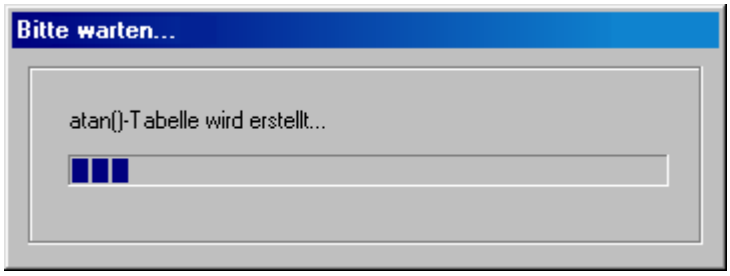

Abb. 25 Startfenster mit Fortschrittsanzeige

Sind alle benötigten Daten berechnet und das Startfenster geschlossen, erscheint das Hauptfenster der Anwendung.

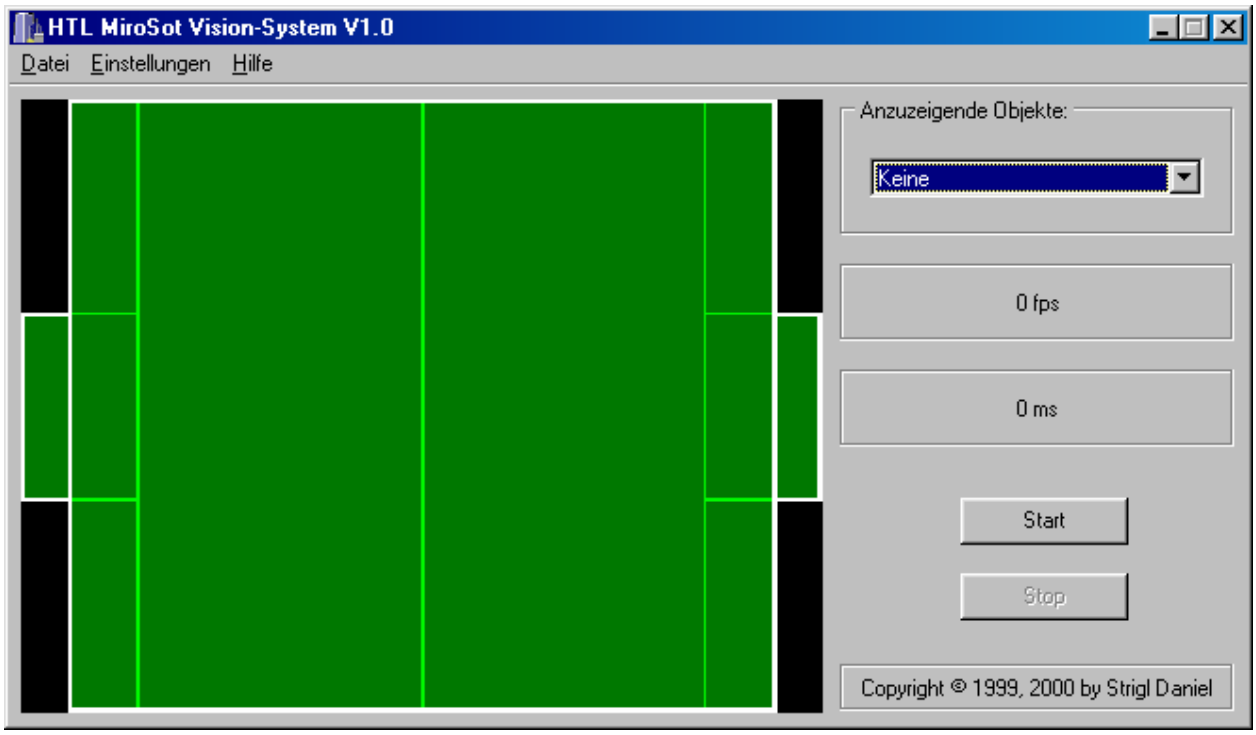

Abb. 26 Hauptfenster der Anwendung

Das Hauptfenster besteht dabei aus einer Menüleiste, einer Abbildung des Spielfeldes, einer Drop-Down Liste, zwei Infoanzeigen, zwei Buttons und einem Copyright-Vermerk. Die Menüleiste besteht dabei aus folgenden Einträgen:

Abb. 27 Die Menüleiste des Hauptfensters mit ihren Einträgen

Mit dem Eintrag Datei-Beenden läßt sich die Anwendung beenden, mit dem Eintrag Einstellungen-Kalibrierung das Kalibrierungsfenster des Systems öffnen. Bevor die eigentliche Positionsermittlung der Objekte gestartet werden kann, müssen noch etliche Einstellungen innerhalb dieses Kalibrierungsfensters getätigt werden. Der Eintrag Hilfe-Info zeigt ein kleines Info-Fenster mit Versionsnummer der Anwendung, Copyright-Vermerk usw. Ein weiterer Teil des Hauptfensters ist die Abbildung des Spielfeldes, innerhalb der die Objektpositionen und deren Ausrichtung angezeigt wird. Es dient also der reinen Kontrolle der ermittelten Positionen, Ausrichtungen und Geschwindigkeiten. Mit der Drop-Down Liste läßt sich bestimmen welche Objekte innerhalb dieser Abbildung des Spielfeldes angezeigt werden sollen. Die beiden Infotexte des Hauptfensters beinhalten dabei die Verarbeitungsgeschwindigkeit (in ms) und die Anzahl der verarbeiteten Bilder pro Sekunde (in fps). Mit den beiden Buttons lassen sich die zyklische Positionsermittlung starten bzw. stoppen. Bevor nun die zyklische Ermittlung von Position, Ausrichtung und Geschwindigkeit gestartet werden kann müssen noch einige Einstellungen innerhalb des Kalibrierungsfensters getätigt werden, welche im nächsten Bild zu erkennen sind.

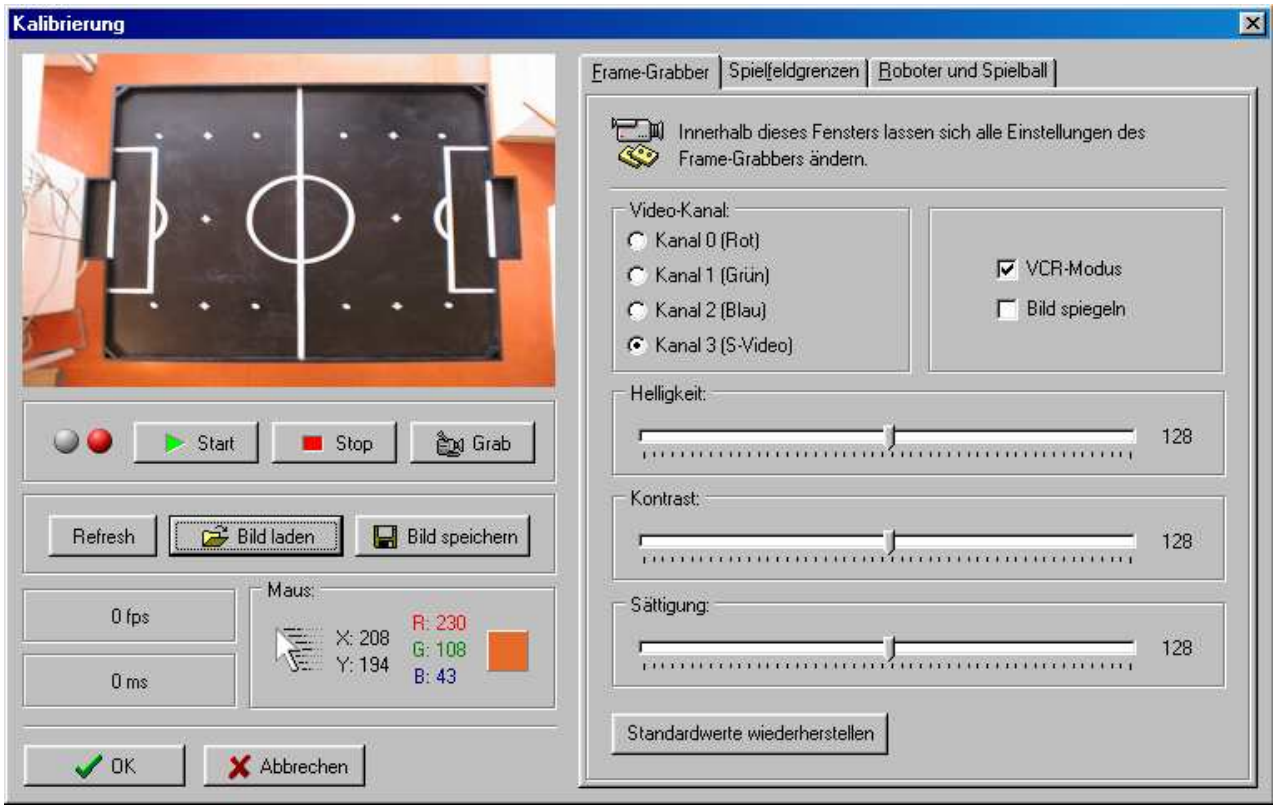

Abb. 28 Kalibrierungsfenster mit den Frame-Grabber Einstellungen

Das Kalibrierungsfenster besteht dabei aus mehreren Komponenten. Links oben ist das Live-Bild zu erkennen. Unterhalb dieses Live-Bildes befinden sich mehrere Buttons zum Starten und Stoppen des Videos, sowie zum einzelnen Einlesen der aufgezeichneten Bilder. Mit den Button „Bild laden“ lassen sich gespeicherte Bilddaten, als Alternative zum Einlesen über die Kamera, laden. Über den benachbarten Button „Bild speichern“ lassen sich die aufgezeichneten Bilder als Bitmap auf der Festplatte des Computers speichern. Mittels dem Button „Refresh“ läßt sich das ürsprüngliche Bild wieder herstellen, falls es zu einer Änderung gekommen ist. Unterhalb dieser ganzen Kontroll-Buttons sind wieder die Verarbeitungsgeschwindigkeit (in ms) sowie die verarbeiteten Bilder pro Sekunde (in

fps) dargestellt. Rechts davon ist der Farbwert, mit den einzelnen Farbanteilen (RGB) unterhalb der Maus dargestellt. Im rechten Teil des Fensters lassen sich die Frame-Grabber Eigenschaften einstellen, sowie deren Standardwerte wiederherstellen. Zu diesen Eigenschaften gehören der verwendete Videokanal, Helligkeit, Kontrast, Sättigung usw.

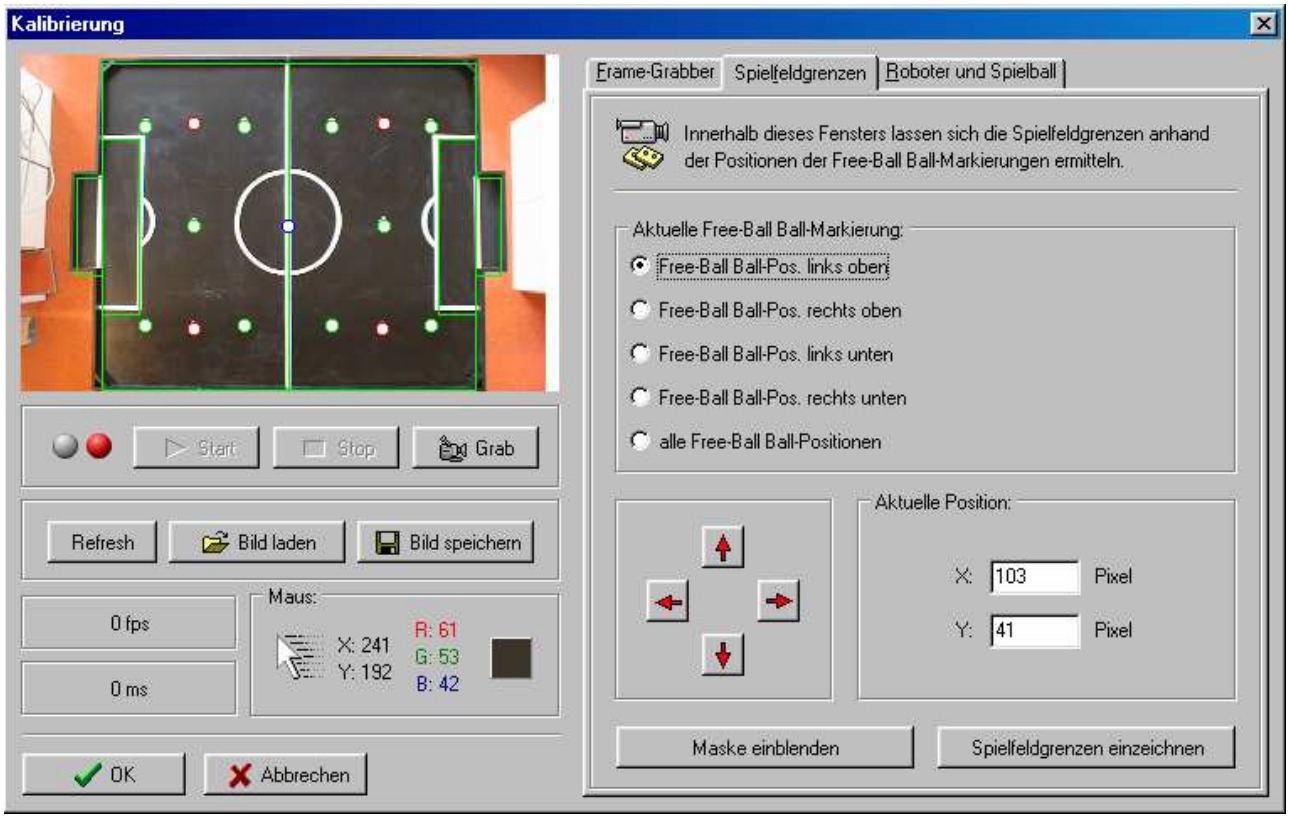

Abb. 29 Kalibrierungsfenster mit dem eingezeichneten Spielfeldgrenzen

Im nächsten Teil des Fensters lassen sich die Spielfeldgrenzen mit Hilfe der Free-Ball Ball-Markierungen des Spielfeldes einstellen. Jedes Spielfeld besitzt 4 Free-Ball Ball-Markierungen, die zum Aufstellen der Roboter benötigt werden. Diese 4 Positionen werden nun verwendet, um daraus die Spielfeldgrenzen zu ermitteln. Die Positionen der 4 Free-Ball Ball-Markierungen lassen sich entweder über die 4 darunterliegenden Buttons positionieren oder über einen „Klick“ mit der Maus in das eingelesene Bild. Mittels dem rechten Button „Spielfeldgrenzen einzeichnen“ lassen sich die Spielfeldgrenzen innerhalb des Live-Bildes einzeichnen. Mittels dem linken Button „Maske einblenden“ läßt sich die dadurch entstandene Maske anzeigen. Im nun letzten Teil des Kalibrierungsfensters lassen sich die Farbwerte der zu suchenden Objekte, sowie deren Toleranzwerte einstellen.

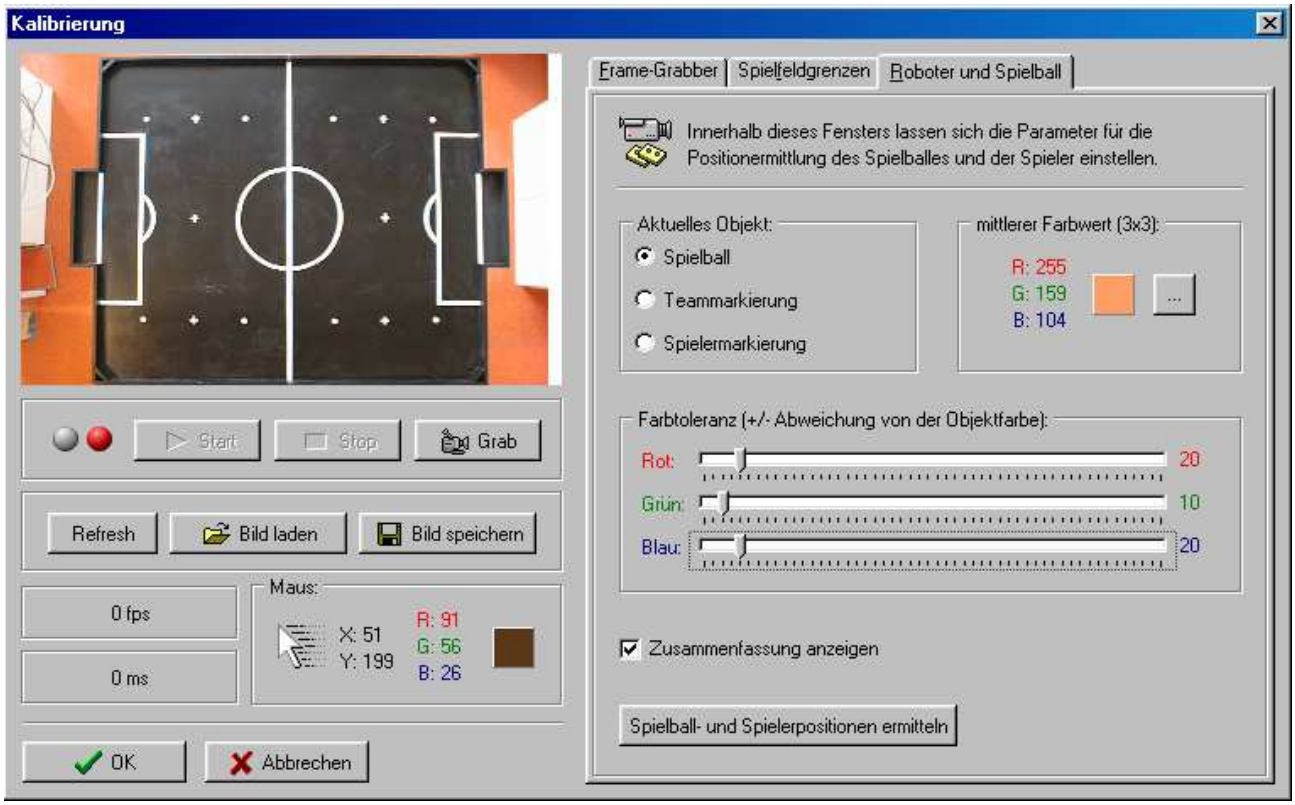

Abb. 30 Kalibrierungsfenster mit eingestellten Farbwerten der zu suchenden Objekte

Innerhalb dieses Teils des Kalibrierungsfensters lassen sich die Farbwerte sowie die Toleranz, die zur Suche der Objekte benötigt wird, einstellen. Mit diesen Farbwerten und den dazugehörigen Toleranzen lassen sich eine obere und untere Grenze festlegen, innerhalb der die Pixel des Objektes sich befinden müssen. Mit Hilfe des darunterliegenden Buttons „Spielball- und Spielerpositionen ermitteln“ lassen sich die Objekteigenschaften (Position, Ausrichtung, Geschwindigkeit) ermitteln und innerhalb des Bildes einzeichnen. Falls gewünscht (Zusammenfassung anzeigen) lassen sich die ermittelten Daten, als Zahlenwerte, innerhalb eines kleinen Fensters anzeigen.

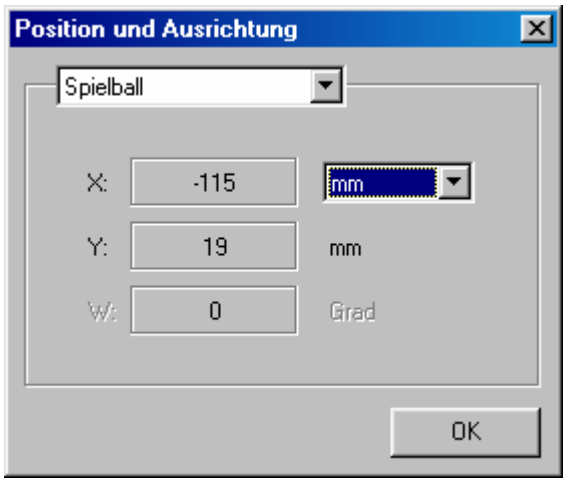

Abb. 31 Die ermittelten Objektdaten (Position und Ausrichtung als Pixel- und mm-Werte)

Innerhalb dieses Fensters sind die Objektdaten (Position und Ausrichtung) sowohl als Pixel- als auch als mm-Werte dargestellt. Somit wäre die Kalibrierung des Programms beendet und das Kalibrierungsfenster kann über einen der beiden

Buttons (OK oder Abbrechen) geschlossen werden. Der Button „OK“ speichert dabei alle Einstellungen, wohingegen der Button „Abbrechen“ alle gemachten Einstellungen verwirft. Das letzte Fenster des Programms ist das Infofenster, das Versionsnummer, sowie Copyright-Vermerk beinhaltet.

Abb. 32 Das Infofenster der Anwendung

7. Probleme bei der Programmentwicklung

Eines der größten Probleme war die Einbindung des Frame-Grabbers in das Projekt. Viele Einstellungen, bei der Einbindung des Frame-Grabbers, führten zu einem verzerrten Bild innerhalb des Speichers des Computers, das nicht verwendet werden konnte. Somit mußten die richtigen Einstellungen bei der Einbindung des Frame-Grabbers gefunden werden, was einige Zeit in Anspruch nahm. Ein weiteres Problem war die DirectX-Programmierung. Da dies die ersten Gehversuche bei der DirectX-Programmierung waren, kam es zu etlichen Probleme, die nur mit viel Zeitaufwand beseitigt werden konnten.

8. Anhang - Aufbau und Funktionsweise von Bildverarbeitungssystemen

Bildverarbeitungssysteme müssen vielseitige und anspruchsvolle Anforderungen erfüllen. Sie sollen z.B.

- Bilder von unterschiedlichen Quellen aufnehmen, speichern und verarbeiten,
- bei Bedarf Bildverarbeitungsoperationen in Echtzeit ausführen,
- wechselnden Einsatzbedingungen flexibel angepaßt werden können,
- eine hochentwickelte graphische Benutzeroberfläche besitzen.

Dementsprechend sind moderne Bildverarbeitungssysteme modular aufgebaute, heterogene Multiprozessorsysteme. Ihre wichtigsten Komponenten sind

- die Kontroll- und Steuereinheit,
- das Bildaufnahme-System,
- der Video-Eingangsteil,
- der Bildspeicher,
- die Bildbearbeitungseinheit und
- der Video-Ausgangsteil.

Die folgende Abbildung zeigt eine Übersicht über die Module eines typischen Bildverarbeitungssystems. Die aus dem Video-Eingangsteil, dem Bildspeicher und dem Video-Ausgangsteil bestehende Einheit wird als Frame-Grabber bezeichnet. Wir

betrachten diese Einheit in den folgenden Abschnitten genauer. Dabei legen mehr Gewicht auf die funktionellen Merkmale als auf die aktuellen technologischen Kenndaten, die sich schnell verändern.

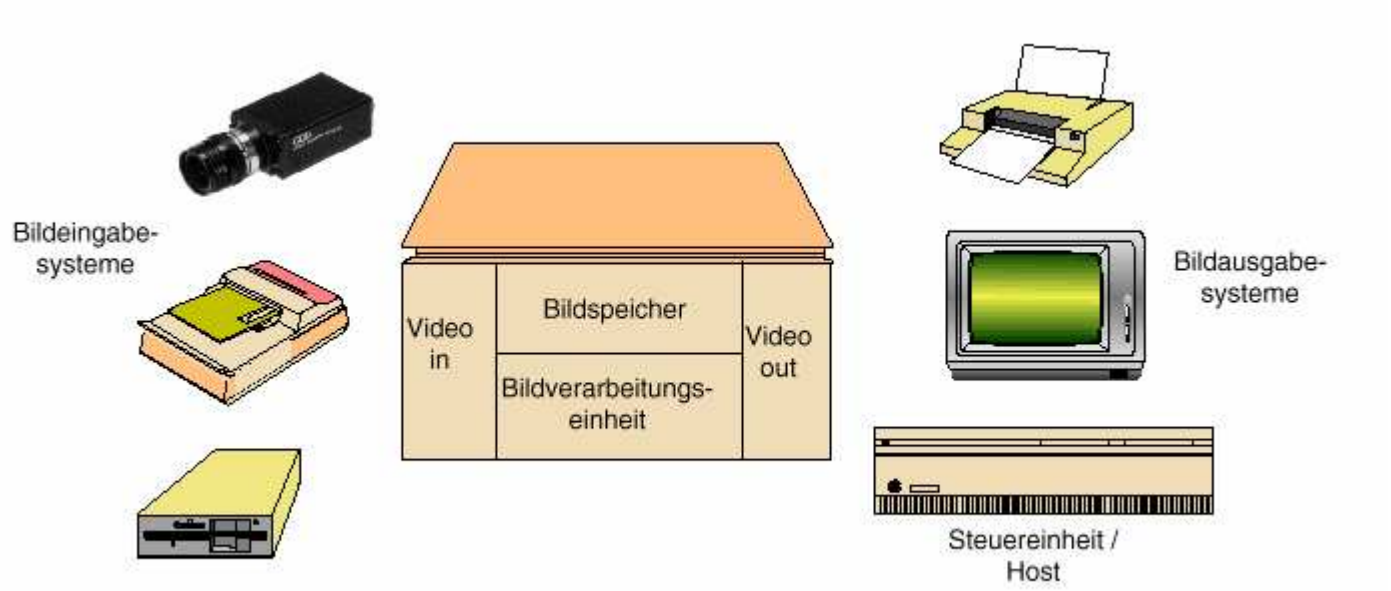

Abb. 33 Modularer Aufbau eines Bildverarbeitungssystems

8.1. Die Kontroll- und Steuereinheit

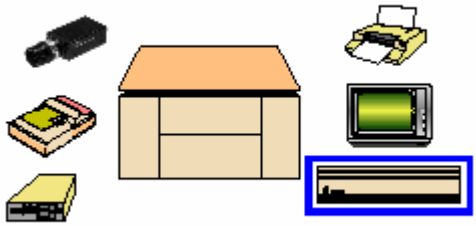

Abb. 34 Die Kontroll- und Steuereinheit

Der Host-Rechner eines Bildverarbeitungssystems stellt die interaktive Schnittstelle zum Benutzer zur Verfügung, steuert die Peripheriegeräte und die Bildverarbeitungseinheit und verwaltet die Bilddateien. Darüber hinaus kann er selbst Bildvararbeitungsoperationen ausführen und die Ergebnisse weiterverarbeiten. In der Regel ist der Host-Rechner ein PC oder eine Workstation.

8.2. Bildaufnahme-Systeme

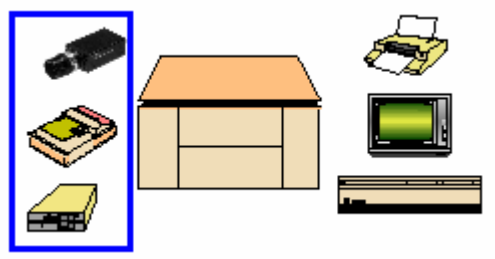

Abb. 35 Die Bildaufnahme-Systeme

Die Wahl eines geeigneten Aufnahme-Systems ist von entscheidender Bedeutung für den Erfolg aller weiteren Bildverarbeitungsprozesse. Häufig eingesetzte Geräte sind z.B.
- Video-Kameras,
- Hand- und Flachbett-Scanner,
- Laser-Scanner,
- Licht-Mikroskope,
- Raster-Elektronenmikroskope,
- Termographie-Kameras,
- Röntgen- und Kerspin-Tomographen.

Sie unterscheiden sich in der Sensor-Technologie, im Aufnahme-Verfahren und im aufgenommenen Spektralbereich.

Sensorik

Die technologische Grundlage all dieser Geräte bilden wenige Sensor-Technologien. Die Sensoren sind letztendlich für die Erfassung der Bildinformationen verantwortlich. Um die Aufnahme in der erforderlichen Qualität bei vorgegebenen Aufnahmebedingungen zu ermöglichen, werden sie in mehr oder wendiger komplexe Aufnahme-Systeme eingebettet. Wichtige Sensoren sind z.B.
- CCD-Arrays und CCD-Zeilen,
- Bild-Aufnahme-Röhren,
- Photodioden,
- Richtantennen,
- Strahlungs-Meßgeräte.

Jeder Sensortyp kann nur eine bestimmte Strahlungs-Art in einem beschränkten Spektralbereich aufnehmen. Dementsprechend unterscheidet man z.B.
- elektromagnetische Sensoren,
- megnetische Sensoren,
- Schall-Sensoren.

Bei elektromagnetischen Sensoren unterscheidet man je nach dem aufgenommenen Spektralbereich Sensoren für Radiowellen, Mikrowellen, für infrarotes, sichtbares und ultraviolettes Licht sowie für Röntgen- und Gammastrahlung.

Aufnahmeverfahren

Bei der Bildaufnahme wird von einer Szene sowohl eine Ortsinformation als auch eine Farb- bzw. Helligkeitsinformation aufgenommen. Beide Informationstypen müssen digitalisiert werden und machen zusammen das digitalisierte Bild aus.

- Ortsinformation

Sie ergibt sich dadurch, daß man die Szene in eine Matrix aus rechteckigen oder quadratischen Zellen zerlegt und für jede dieser Zellen die Farb/Helligkeitswerte abtastet. Die übliche matrixartige Zerlegung in Zeilen und Spalten wie auch die rechteckige oder quadratische Zellenform sind technisch bedingt und haben Konsequenzen für alle weiteren Operationen mit den digitalisierten Bildern. Wichtige

Parameter bei der räumlichen Diskretisierung sind die Auflösung, mit der die Ortsinformationen digitalisiert wird und die Zellen-Geometrie.

Die Abtastverfahren zur Gewinnung der Ortsinformation beruhen auf unterschiedlichen technischen Lösungen, z.B.
- auf mechansichen Kontruktionen, wie bei Zeilenscannern,
- auf der Ablenkung eines Abtaststrahls durch mechansiche Spiegel, etwa bei Laserscannern,
- auf simultaner Aufnahme mit zweidimensionalen Sensor-Arrays, wie bei CCD-Kameras,
- auf elektronischen Linsen, z.B. in der Tomogrphie und in der Raster-Elektronenmikroskopie.

- Intensitäts- und Farbinformation

Die Abtastung leifert für jeden Bildpunkt einen oder mehrere digitale Intensitätswerte. Bei einer normalen Grauwert-Aufnahme genügt ein Aufnahmekanal für die mittlere helligkeit der Zellen, bei Farb-Aufnahmen werden drei Kanäle für die roten, grünen und blauen Bestandteile des sichtbaren Lichts eingesetzt. Die Aufnahmekanäle können aber auch Meßwerte beliebiger Art liefern. Wesentlich dabei ist, daß auch diese Information stets mit einem mehr oder weniger groben digitalen Raster aufgenommen wird.

- Abtastgeschwindigkeit

Wir unterscheiden nach ihrer Arbeitgeschwindigkeit Echtzeit- und Slow Scan-Geräte:
- *Echtzeitsysteme*

können mindestens 25 Bilder in der Sekunde kontinuierlich aufzeichnen. Typische Geräte dieser Kategorie sind Videokameras oder Meßeinrichtungen in der Qualitätskontrolle.
- *Slow Scan-Geräte*

Eignen sich für langsame Bildfolgen oder sogar nur für Einzelbild-Aufnahmen. Eine einzige Aufnahme kann mehrere Sekunden bis Minuten dauern. Geräte dieser Kategorie sind z.B. Zeilenscanner oder Tomographen.

Kameratechnologie

Verglichen mit anderen Aufnahme-Systemen sind CCD-Kameras in der Bildverarbeitung heute die am häufigsten eingesetzten Aufnahmeeinrichtungen (CCD=Charge Coupled Devices). Die wichtigsten Vorteile der dabei eingesetzten CCD-Zeilen und CCD-Arrays sind:
- lange Lebensdauer,
- hohe Empfindlichkeit, es genügen geringe Lichtintensitäten für eine Aufnahme,
- großer Dynamik-Bereich, d.h. sie lassen große Intensitäts-Unterschiede zu,
- hohe geometrische Genauigkeit, auch bei Temperaturschwankungen.

Ds Kernstück jeder CCD-Kamera ist das Sensor-Array. Es besteht aus einer großen Zahl lichtempfindlicher Zellen, die in einer rechteckigen Matrix angeordnet sind (Abbildung 36). Jede Zelle wandelt einfallendes Licht in elektrische Ladungen, die sie in einem MOS-Kondensator speichert. Die Ladungen werden zeilenweise in ein

horizontales Transportregister übernommen und daraus seriell ausgelesen. Am Ausgang der Kamera steht dann ein analoges Video-Signal zur Verfügung.

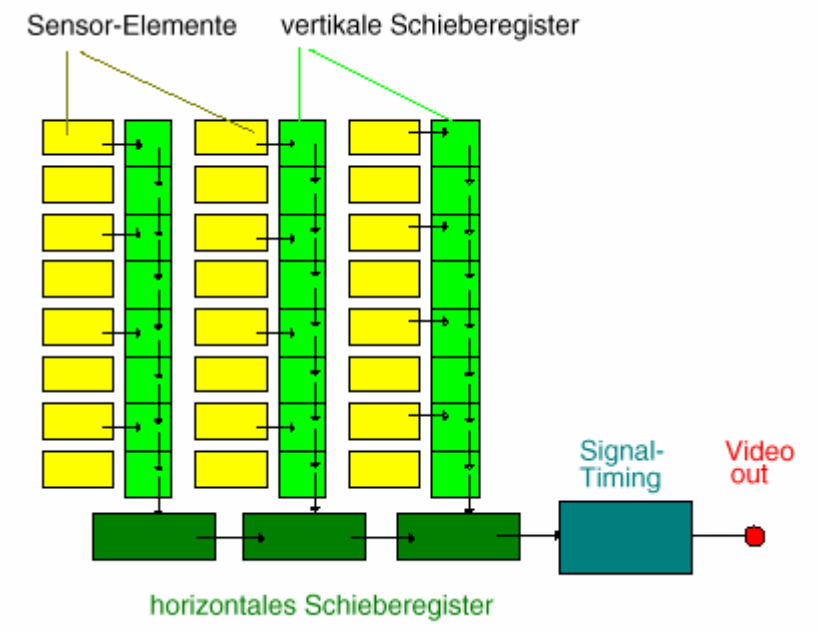

Abb. 36 Aufbau eines CCD-Array

Die Video-Norm der Kamera spezifiziert das Timing und die Pagel des Video-Signals. Neben den Pixel-Informationen enthält das Video-Signal auch Synchronisations-Signale für den Bildanfang und die Zeilenwechsel (Abbildung 37). Die wichtigsten Video-Normen in der Bildverarbeitung sind die europäische CCIR-Norm und die amerikanische RS170-Norm. Beide Verfahren übertragen jedes Bild als zwei aufeinanderfolgende Halbbilder, die nur gerade bzw. ungerade Zeilen enthalten.

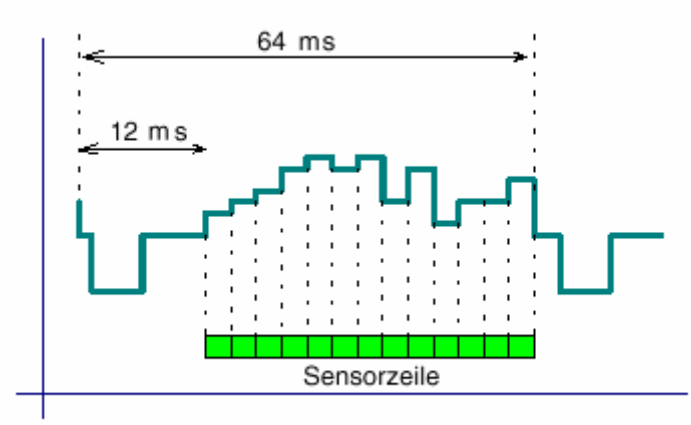

Abb. 37 Schematische Form eines CCIR-Video-Signals

Die folgende Tabelle faßt charakteristische Merkmale beider Signaltypen zusammen:

	CCIR	RS 170
horizontale Abtastzeit	64 µs	63,55 µs
horizontale Abtastfrequenz	15,625 KHz	15,7343 KHz
horizontale Totzeit	11,8414 µs	10,7556 µs
vertikale Abtastzeit	20 ms	16,6833 ms
vertikale Abtastfrequenz	50 Hz	59,9401 Hz
vertikale Totzeit	1,5360 ms	1,2711 ms
Zeilen pro Bild	625	525
davon aktive Zeilen	576	485

8.3. Der Framegrabber

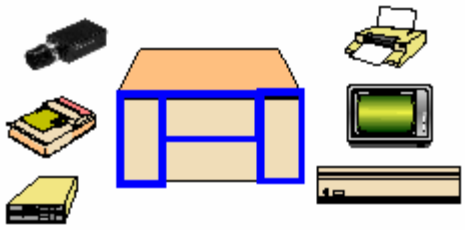

Abb. 38 Der Framegrabber

Abbildung 38 zeigt die innere Struktur eines Framegrabber, der in der Regel
- den Video-Eingangsteil,
- den Bildspeicher und
- den Video-Ausgangsteil

realisiert. Seine erste Stufe bildet der Video-Eingangsteil, der das analoge Video-Signal der Kamera aufnimmt und es digitalisiert. Der Bildspeicher nimmt das digitalisierte Bild auf. Der Video-Ausgangsteil wandelt gespeicherte Bilder wieder um in ein analoges RGB-Signal, das dann auf einem Monitor dargestellt wird.

Aufnahmegeräte, die ein digitales Signal liefern, z.B. Scanner, übertragen ihre Daten über eine digitale Schnittstelle, z.B. einen SCSI-Controller direkt an den Bildspeicher des Bildverarbeitungssystems.

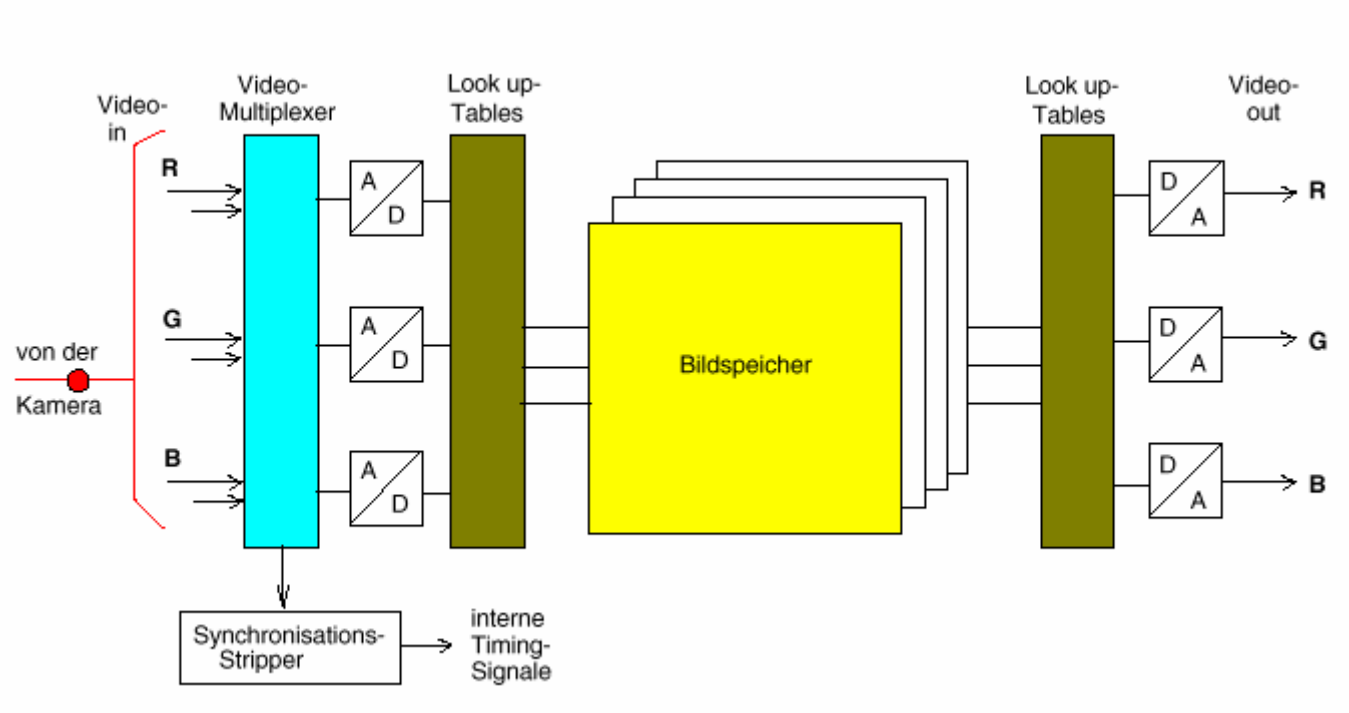

Abb. 39 Struktur eines Framegrabber

Der Video-Eingangsteil

Der Video-Eingangsteil bildet die Schnittstelle zwischen der Aufnahme-Peripherie und dem zentralen Bildverarbeitungssystem. Die ankommenden Video-Signale können unterschiedlicher Art sein:

- digitale Daten, etwa bei Zeilenscannern,
- genormte, analoge Daten, bei Videokameras und Videorecordern, oder
- ungenormte, analoge Daten, z.B. bei Meß-Einrichtungen oder Raster-Elektronenmikroskopen.

Der Video-Eingangsteil muß die Bildquelle mit dem Bildspeicher synchronisieren, analoge Daten verarbeiten und digitalisieren und schließlich die aufbereiteten Daten abspeichern. Dazu besitzt er vier Funktionseinheiten:

- *Videomultiplexer*

Er stellt die erste Verarbeitungsstufe für die ankommenden Analogsignale dar. Seine Aufgabe ist es, die von mehreren Videoquellen gelieferten Datenströme auszuwählen und zu serialisieren.

- *Synchronisations-Stripper*

Genormte Video-Signale enthalten neben der reinen Bildinformation auch Synchronisierungssignale für den Bildanfang und die Zeilenlänge. Der Synchronisations-Stripper extrahiert diese aus dem Datenstrom und erzeugt Triggersignale zur Synchronisation der Bildquelle mit dem Bildspeicher.

- *Analog / Digital-Wandler*

Das analoge Video-Signal muß in einen digitalen Datenstrom umgewandelt werden. Dazu wird jeder Eingangskanal mit einer festen Taktrate abgetastet, wobei die abgetasteten Analogwerte in eine digitale Signalfolge umgewandelt werden. Vile

Standard-Systeme arbeiten mit einer Abtastfrequenz von 10MHz, wobei Bilder mit 512x512 Bildpunkte digitalisiert werden können. Variable Scan-Systeme setzen statt der Analog / Digital-Wandler Signalprozessoren mit variierbarer Abtastfrequenz ein und können so auch variable Bildgrößen generieren.

- *Eingangs-Look-up-Tabelle*

Diese ist die letzte Station des Video-Eingangsteils. Sie transformiert die digitalen Bildpunkt-Werte noch vor der Übertragung in den Bildspeicher. Damit sind einfache Vorverarbeitungsoperationen möglich, z.B. das Ausblenden einzelner Bitebenen. Oft werden mehrere Look-up-Tabellen verwendet, die vom Hostrechner geladen und in Echtzeit selektiert werden können.

Der Bildspeicher

Der Bildspeicher nimmt als zentraler Teil des Bildverarbeitungssystems die digitalisierten Bilder auf zur Verarbeitung. Er dient gleichzeitig als Bildwiederholspeicher für den Video-Ausgangsteil. Bei reiner Grauwert-Verarbeitung genügt eine Tife von 8 Bit pro Bildpunkt, bei Echtfarben-Verarbeitung werden bis zu 32 Bit pro Bildpunkt verwendet. Bei den Bildverarbeitungssystemen im oberen Leistungsbereich sind die freie Konfigurierbarkeit des Bildspeichers, flexible Adressierungstechniken oder Zugriffs-Caches wichtige Leistungsmerkmale. Außer dem Video-Eingangsteil greifen auch der Hostrechner und die Bildverarbeitungsprozessoren auf den Bildspeicher zu. Wir betrachten einige Leistungsmerkmale von Bildspeichern:

- *Speicherkapazität*

In Anbetracht der umfangreichen Bild-Daten wird die Kapazität des Bildspeichers möglichst groß gewählt. Sie kann einige Megabyte bei einfachen PC-Bildverarbeitungssystemen und bis über 100 Megabyte bei Hochleistungssystemen betragen. Systeme mit einer derart hohen Speicherkapazität können Bildfolgen bis zu einigen Minuten resident verarbeiten.

- *Bildspeicher-Struktur*

Die Bildspeicher leistungsfähiger Bildverarbeitungssysteme unterscheiden sich von konventionelle Arbeitsspeichern durch ihre freie Konfigurierbarkeit, unterschiedliche Zugriffsmodi und parallele Zugriffspfade. Oft steht für jeden Eingangskanal eine eigene Speicherbank zur Verfügung. Teile des Bildspeichers können im Adreßraum des Hostrechners liegen.

Die Konfigurierbarkeit des Bildspeichers ermöglicht es, für Bilder beliebiger Größe und Pixel-Tiefe eigene Speicherbereiche zu definieren und diese als autarke Einheiten zu behandeln. Unterschiedliche Zugriffsmodi erlauben den Zugriff auf einzelne Bildpunkte, auf Bit-Ebenen sowie auf ganze Zeilen oder Spalten eines Bildes. Block Move-Operationen können ganze Bildbereiche verschieben.

Bei Echtzeit-Verarbeitung muß der Bildspeicher solche Operationen in sehr kurzer Zeit ausführen, z.B. innerhalb eines einzigen Speicherzyklus. Deshalb werden dafür extrem schnelle Speicher-Bausteine und Adressierungslogik verwendet.

- *Parallel-Zugriff*

Bildspeicher sind als Dual Port Memory konzipiert, d.h. sie können zwei gleichzeitige Zugriffe bedienen. Deshalb kann die Bildaufnahme parallel zur Wiedergabe auf dem Monitor erfolgen. Dabei wird die Ausgabe gegenüber der Eingabe durch interne Pufferung um einen Bildzyklus, d.h, 40 msec verzögert.

Der Video-Ausgangsteil

Der Video-Ausgangsteil erzeugt aus dem digitalisierten Bildspeicherinhalt wieder analoge Video-Signale, die auf einem Monitor ausgegeben oder auf einem Videorecorder aufgezeichnet werden können. Die an diesem Prozeß beteiligten Komponenten sind die Ausgangs-Look-up-Tabelle, die Digital / Analog-Wandler und der Display-Prozessor.

- *Ausgangs-Look-up-Tabellen*

Ebenso wie der Video-Eingangsteil besitzt auch der Video-Ausgangsteil Look-up-Tabellen zur Transformation von Grau- oder Farbwerten. Mit ihrer Hilfe sind homogene Punktoperationen am Bildsignal möglich, ehe es an den Digital / Analog-Wandler weitergeleitet wird. Wir bei den Eingangs-Look-up-Tabellen existieren oft mehrere voneinander unabhängige Look-up-Tabellen, mindestens jedoch für jeden Kanal eine. Sie können vom Host geladen und in Echtzeit selektiert werden.

- *Digital / Analog-Wandler*

Die aus dem Bildspeicher ausgelesenen digitalen Informationen muß nach der Transformation durch die Look-up-Tabellen in ein analoges Video-Signal umgewandelt werden. Diese Aufgabe erfüllen die Digital / Analog-Wandler.

- *Display-Prozessor*

Er steuert die Umsetzung der digitalen Bildinformationen aus dem Bildspeicher in ein analoges Signal, das den Elektronenstrahl des Monitors steuert. Bei einfachen Systemen wird der komplette Bildspeicher-Inhalt auf den ganzen Bildschirm abgebildet. Zusaätzliche Funktionalitäten sind:

- PAN- und Scroll-Funktionen: Dabei wird die Startzeile und Startspalte für die Ausgabe auf dem Bildschirm modifiziert.
- Zoom-Funktionen: Damit kann der Bildspeicherinhalt vergrößert oder verkleinert auf dem Monitor dargestellt werden.

8.4. Die Bildverarbeitungseinheit

Abb. 40 Die Bildverarbeitungseinheit

Im einfachsten Fall benötigt ein Bildverarbeitungssystem gar keine eigene Bildverarbeitungseinheit. Dann müssen alle Verarbeitungsoperationen direkt durch

den Host ausgeführt werden. Falls die Rechenzeit keine ausschlaggebende Rolle spielt, ist dies durchaus akzeptabel. Bei Echtzeit-Verarbeitung sind aber Operationszeiten von mehreren Sekunden oder gar Minuten für ein Bild nicht mehr tragbar. Es müssen daher Spezialprozessoren eingesetzt werden, z.B.

- *Pipeline-Prozessoren für arithmetische Operationen*

Außer Prozessoren zur schnellen Ausführung elementarer arithmetischer Operationen gibt es Module, die Faltungen, FFT-Operationen usw. ausführen.

- *Filterprozessoren*

Filter-Operationen im Ortsraum erfordern die arithmetische Verknüpfung von Bildpunkten im Bereich einer Maske, die über das Bild bewegt wird. Da der Rechenaufwand dür größere Masken sehr schnell steigt, können solche Operationen durch Spezial-Hardware wesentlich beschleunigt werden.

Busorientierte Multiprozessor-Architekturen

Moderne Bildverarbeitungssysteme sind modular als heterogene Multiprozessorsysteme konzipiert. Die einzelnen Prozessor-Module sind oft frei programmierbare, schnelle Spezialrechenwerke, die flexibel für ihre Aufgabe konfiguriert werden können. Die Zusammenarbeit der unterschiedlichen Funktionseinheiten erfolgt direkt über das Bus-System des Hostrechners (VME-Bus, AT-Bus, PCI-Bus) und oft zusätzlich über einen eigenen Image Bus. Abbildung 41 zeigt die Struktur einer solchen Bus-Architektur.

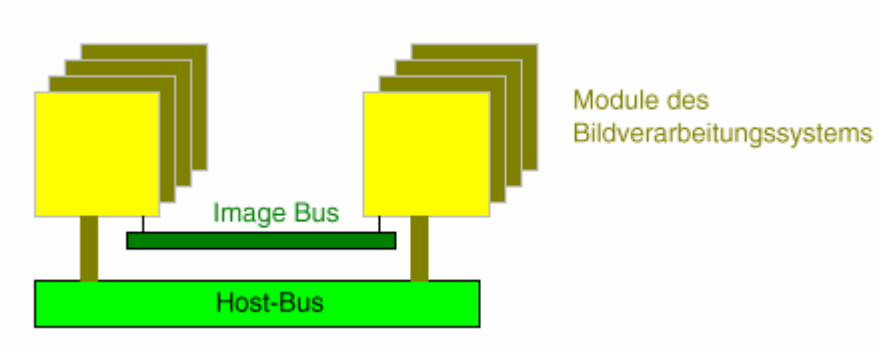

Abb. 41 Bus-orientiertes Bildverarbeitungssystem

9. Anhang – Methoden zur Verarbeitung von Bildinformationen

Bei der Verarbeitung digitalisierter Bilder lassen sich spezifische Aufgabenbereiche unterscheiden:

9.1. Bildverarbeitung

Bei der Bildverarbeitung im engeren sind sowohl die Eingabeinformationen als auch das Ergebnis Bilder. Es ist dabei das Ziel, die Bildinformation so aufzubereiten, daß sie visuelle besser interpretiert werden kann oder daß die anschließende Bildanalyse erleichtert wird. Typische Techniken der Bildverarbeitung sind:

- die Modifikation der Farb- / Graustufenskala durch Skalierung,
- die Beseitigung von Bildfehlern durch Filterung,
- die Korrektur von Aufnahmefehlern durch Kalibrierung,

- die Extraktion von Strukturen durch Kantenextraktion.

Eine typische Beispielanwendung ist die folgende:

Die Interpretation von Luftbild-Aufnahmen wird durch Vorverarbeitungsschritte oft wesentlich erleichtert. So kann man durch eine Skalierung den Kontrast der Aufnahme verbessern, durch Filterung lassen sich die Konturen hervorheben, durch Falschfarbendarstellung können relevante Gebiete, etwa Wälder deutlich hervorgehoben werden.

9.2. Bildanalyse

Die Bildanalyse interpretiert den Inhalt von Bildern. Je nach dem beabsichtigten Resultat liefert sehr unterschiedliche Ergebnisse:
- numerische und geometrische Merkmale, etwa den Flächeninhalt eines Bereich,
- topologische und stereologische Merkmale, etwa die Anordnung von Objekten in einer Szene,
- densitometrische Merkmale, z.B. die Absorption von Strahlung,
- Texturmerkmale, z.B. die kristalline Struktur einer Werkstoffoberfläche.

Eine Bildanalyse liegt etwa bei der folgenden Anwendung vor:

Beulen und Dellen in Blechteilen von weniger als 50 Mikrometer Tiefe sind nach der Lackierung bereits deutlich zu erkennen. Durch den Vergleich eines aufprojizierten Sreifenmusters mit einem Referenzmuster lassen sich solche Beschädigungen schon vorher berührungslos erkennen und lokalisieren.

9.3. Bild-Codierung

Ziel der Bildcodierung ist eine Verringerung des Aufwandes zur Speicherung oder Übertragung von Bildern. Die wichtigsten Ansätze zielen dabei ab:
- auf die Kompression der Information durch Beseitigung von Redundanz, wobei die volle Bildinformation erhalten bleibt,
- auf die Unterdrückung irrelevanter Informationen, was in der Regel mit einem Informationsverlust verbuden ist.

Der folgende Vergleich zeigt die Bedeutung von Codierungsverfahren:

Ein Textzeichen benötigt bei einer Auflösung von 15x20 Pixel 300 Bit zur Speicherung. Durch Kompressionstechniken, z.B. Block-Codierung, kann man eine Reduktion des Speicherbedarfs etwa um den Faktor 6 erreichen und benötigt dann nur noch 50 Bit pro Zeichen. Setzt man Texterkennung ein, so genügen 8 Bit pro Zeichen, das entspricht einer Reduktion um den Faktor 40.

9.4. Bildverarbeitungsprozesse

Die obigen Beispiele zeigen, daß zur Verarbeitung von Bildinformationen ein breites Spektrum von Techniken zur Verfügung steht. Für einen speziellen Anwendungsfall maß man daher eine geeignete Sequenz von Verarbeitungsschritten festlegen, die zu einer Funktionsfolge zusammengefaßt und auf das Bildmaterial angewandt

werden kann. Mit modernen Bildverarbeitungssystemen wird die Verarbeitungsfolge interaktiv entwickelt und aufgezeichnet. Sie kann dann als Makro beliebig oft ablaufen. Abbildung 42 zeigt das Schema einer typischen Funktionsfolge für eine Bildanalyse.

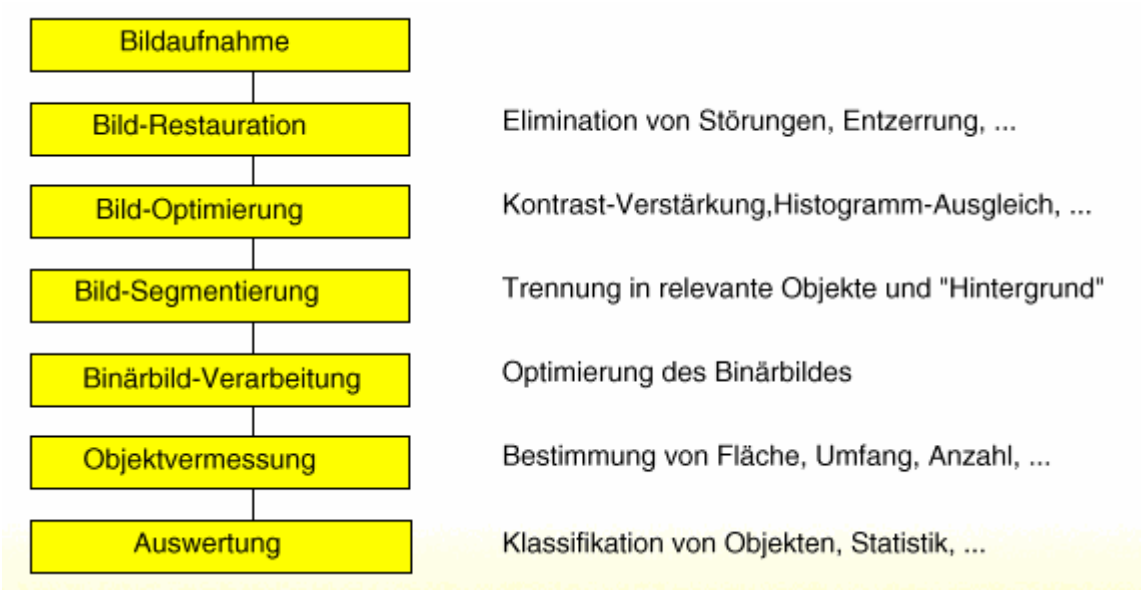

Abb. 42 Schema einer typischen Funktionssequenz bei der Bildanalyse

10. Verwendete Literatur

[1] Peter Haberäcker, Praxis der digitalen Bildverarbeitung und Mustererkennung, Carl Hanser Verlag München Wien, 1995
[2] Herbert Kopp, Bildverarbeitung Interaktiv, B.G. Teubner Stuttgart, 1999
[3] Toolbox Redaktion, Borland C++ Builder 3 in Team, C&L Computer und Literatur, 1999
[4] Wolfgang Link, Assembler Programmierung, Franzis, 7. Auflage, 1995
[5] M-Vision 500 Software Development Guide, MuTech Corporation, 1998
[6] Michael Tischer und Bruno Jennrich, PC Intern 5 Systemprogrammierung, DATA BECKER GmbH & Co. KG, 1995
[7] Microsoft Visual C++ 5.0 Online-Hilfe, DirectDraw-Programmierung
[8] Borland C++ Builder 4 Online-Hilfe und Win32-SDK-Referenz