Echtzeitfähige modellbasierte Bilderkennung zur visuomotorischen Kontrolle bewegter Objekte

Diplomarbeit

vorgelegt von:

Michael Bunk

Institut für Informatik
Fakultät für Mathematik und Informatik
Abt. Intelligente Systeme
Universität Leipzig
Betreuung: Prof. R. Der, R. Liebscher

6. Oktober 2004

Inhaltsverzeichnis

Kapitel 1

Einführung

Autonome Roboter müssen sich in einer dynamischen Umwelt zurechtfinden können und zeitnah auf Ereignisse reagieren. Der echtzeitfähigen Verarbeitung visueller Information kommt dabei eine Schlüsselrolle zu. Am Beispiel zweier spezieller Roboter wird die Gewinnung steuerungsrelevanter Daten aus einer Bildsequenz demonstriert. Dazu werden in Form von Modellen umfassende Voraussetzungen aufgestellt von dem, was gesehen werden soll und kann. Es wird ein Erkennungssystem entworfen und implementiert, welches die Extraktion der gesuchten Daten aus den Bilddaten durchführt. Die gewonnenen Daten werden zur Steuerung eines der Roboter verwendet.

In den nächsten Abschnitten werden die grundlegenden Begriffe und vorhandenen Ansätze des Bildverstehens und der Wahrnehmung untersucht. In Kapitel 2 wird der Lösungsweg skizziert. Danach werden die Modelle entwickelt, auf denen die Erkennung basiert. In Kapitel 3 werden die zur Bilderkennung erforderlichen Schritte und die verwendeten Algorithmen vorgestellt. Kapitel 4 beschreibt die Implementation.

1.1 Vorgeschichte

In einem unmittelbar vor dieser Arbeit durchgeführten Robotik-/Kunstprojekt hat der Medienkünstler Julius Popp[1] die zwei „Räder" micro.adam und micro.eva geschaffen. Zielstellung war dabei, die Selbstwahrnehmung von Maschinen zu erforschen.

Die Räder sind auf zwei Rollen gelagert und haben einen bzw. fünf „Arme". Die Arme werden von einem eingebauten Mikroprozessor gesteuert. Durch die Armbewegung wird der Schwerpunkt des Rades verlagert, und es gerät in Bewegung. Mittels einer koordinierten Armbewegung kann ein kontinuierliches Rollen des Rades erzeugt werden.

Selbstwahrnehmung ist in einfacher Weise in einer sensomotorischen Schleife möglich. In einer solchen Schleife werden von den Sensoren aufgenommene Informationen zur Ansteuerung von Aktuatoren verwendet. Aktuatoren sind hier die Motoren, welche die Arme bewegen können. Die Aktuatoren wirken auf die Umwelt und den Roboter selbst. Die Auswirkungen sind dann wieder durch die Sensoren meßbar. Einziger Sensor der Räder ist ein jeweils eingebautes Gyroskop [32], welches einen Wert proportional zur momentanen Drehgeschwindigkeit des Rades liefert. Der hier verwendete Begriff der Selbstwahrnehmung

[1] julius@hgb-leipzig.de, sphericalrobots.com

Abbildung 1.1: micro.adam (Version 1) im Blickfeld einer Webcam. Das an seinem Arm befestigte Kabel liefert Servosteuerungsinformationen vom bildauswertenden Computer.

Abbildung 1.2: micro.eva (Version 1)

(a) micro.adam (b) micro.Eva

Abbildung 1.3: Die „Räder" (Version 1), aus dem Blickpunkt der Kamera

beschränkt sich auf dieser Komplexitätsstufe auf die sensomotorische Schleife und ist von der bewußten Selbstwahrnehmung bzw. bewußten Wahrnehmung überhaupt zu unterscheiden.

Aufgabe der damals implementierten Steuerung der Räder war es, ohne konkrete parametrische Vorgaben das Rad in Bewegung zu halten. D.h. es wurde z.B. nicht vorgegeben, mit welcher Stärke der Armservo (Motor zur Armbewegung) angesteuert werden muß und welches Zeitverhalten dafür zu verwenden ist. Die Steuerung sollte sich adaptiv verhalten und diese Parameter erlernen.

Zu Demonstrationszwecken und zum Test des hier entwickelten Erkennungsprogramms, d.h. um das Programm ausführen zu können ohne die Räder physikalisch anwesend haben zu müssen, existiert ein Video, aus dem Abbildung 1.1 a und b stammen:

- `http://robot.informatik.uni-leipzig.de/~mbunk/raeder.mpg` (17 MB, mpeg4, kein Ton, Wiedergabe mit MPlayer [34] funktioniert, Windows Media Player braucht evtl. zusätzlichen Codec)
- `http://robot.informatik.uni-leipzig.de/~mbunk/raeder.wmv` (25 MB, geringere Kompression)

In dieser Arbeit wird die erste Version der Räder zugrundegelegt. Es wurde inzwischen eine neue Version („Version 2") der Räder entwickelt. Die neuen Räder funktionieren nach denselben Prinzipien, unterstützen jedoch Datenübertragung per Bluetooth, statt serieller Übertragung in Version 1.

Die durch diese Räder gegebene Mikrowelt scheint geeignet, auch eine andere Zielstellung als die oben erwähnte zu verfolgen - die des Bildverstehens. Gleichzeitig könnten für die Steuerung der Räder neue Eingaben zur Verfügung gestellt werden. Dabei würde die sensomotorische Schleife auf einem anderen Weg geschlossen, denn die Informationen des Gyroskops würden durch die Ergebnisse des Bildverstehensprozesses ersetzt.

1.2 Ziel

Um die Leistung des jeweils zur Steuerung des Rades verwendeten Algorithmus bewerten und analysieren zu können und um der Steuerung zusätzliche Daten zu liefern, ist es wünschenswert, die Radbewegung

anhand von Parametern aufzuzeichnen. Werden die aufgezeichneten Parameter zur Steuerung verwendet, muß die Erkennung in Echtzeit erfolgen. Zu spät gelieferte Daten wären wertlos, da auf Ereignisse nicht mehr angemessen reagiert werden kann.

Folgende Parameter sind relevant:

- Der Winkel, um den das Rad aus einer festgelegten Nullposition verdreht ist
- Die Werte, welche die Armstellungen repräsentieren (z.B. -1 für Arm links bis 1 für Arm rechts angewinkelt)

Sind diese Daten über mehrere aufeinanderfolgende Frames (Definition siehe Anfang von Kapitel 3) bekannt, können leicht Drehgeschwindigkeit sowie Beschleunigung des Rades und einzelner Arme berechnet werden. Ein Rad in der Form überwacht werden, daß ein Alarm ausgelöst wird, sobald es selbst oder ein Arm aufhört, sich zu bewegen, z.B. weil sich etwas festgeklemmt hat.

Um diese Parameter zu ermitteln, sollen *Bilderkennungstechniken* eingesetzt werden. Eine Videokamera oder Webcam wird vor dem jeweiligen Rad aufgestellt und ein daran angeschlossener Computer wertet die Bilder aus.

Als zweiter Schritt sollen die gewonnenen Daten zur Steuerung der Räder eingesetzt werden. Dabei soll der bestehende Steuerungsansatz beibehalten werden.

Über die primäre Zielstellung hinaus sollen bei der Durchführung dieser Arbeit Erfahrungen über den Vorgang der Wahrnehmung gesammelt werden. Es soll ein besseres Verständnis für die möglichen Teilprozesse bei Wahrnehmungsvorgängen erreicht werden, indem Wahrnehmungsfunktionen durch eine Computerimplementation realisiert werden. Dabei soll aber nicht von biologischen Vorbildern ausgegangen werden.

1.3 Begriffe

In diesem Abschnitt sollen einige Definitionen bzw. Begriffserklärungen gegeben werden, die später gebraucht werden.

Die *Polardarstellung P_P eines Punktes P* in einer Ebene gegeben in kartesischen Koordinaten $P_K = (x, y)$ ist definiert als:

$$P_P = (\rho, \theta) = \left(\sqrt{x^2 + y^2}, \arctan(y/x)\right) \tag{1.1}$$

Hierbei ist ρ der Abstand des Punktes vom Koordinatenursprung und θ die Richtung. Es gilt $\rho \geq 0$ und $0 \leq \theta < 2\pi$. Falls $\rho = 0$ ist θ nicht eindeutig.

Der *Normalendarstellung einer Geraden $G_P = (\rho, \theta)$* liegt folgende Gleichung zu Grunde:

$$\rho = x * \cos\,\theta + y * \sin\,\theta,\ (x, y) \in \mathbb{R}^2 \tag{1.2}$$

Die Gerade wird also durch den Punkt der Geraden mit dem geringsten Abstand zum Koordinatenursprung parametrisiert. Diese Darstellung hat den Vorteil, daß auch senkrechte Geraden repräsentierbar sind. Sie bildet das Ausgabeformat der Hough-Transformation (siehe Abschnitt 3.2.3).

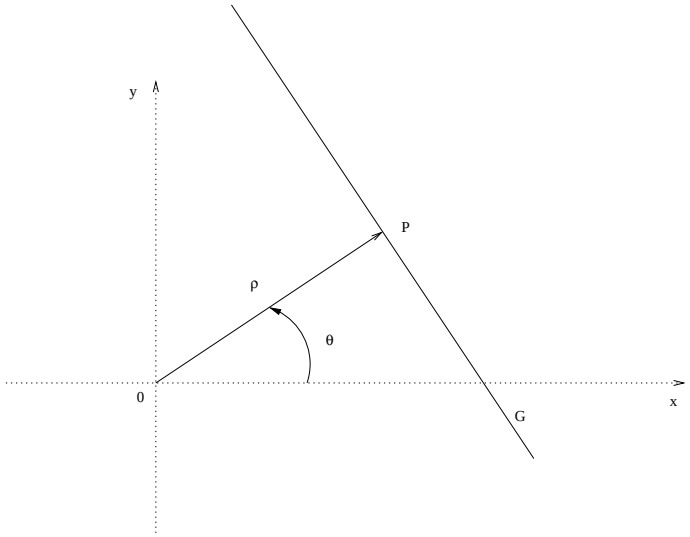


Abbildung 1.4: Polardarstellung von Punkten und Geraden

Als *Angelpunkt* wird der Schnittpunkt zweier Geraden in der Ebene bezeichnet, die in einem bestimmten Winkel zueinander stehen. In Abbildung 1.4 wird der Angelpunkt zwischen der x-Achse und der Gerade, die durch 0 und P verläuft, durch den Koordinatenursprung dargestellt.

Die *Winkeldifferenz* ist einfach $\alpha - \beta$, normiert auf das Intervall $[0, 2\pi)$.

Der *Winkelabstand* $|\alpha - \beta|_{Winkel}$ ist der kleinste Abstand zwischen zwei Winkeln. Er ist stets positiv. Zu beachten sind besonders Fälle wie $\alpha = 350°$ und $\beta = 10°$. Hier beträgt der Abstand 20° und nicht 340°, wie man erwarten könnte. Unter der Voraussetzung, daß $\alpha, \beta \in [0, 2\pi[$:

$$|\alpha - \beta|_{Winkel} = \begin{cases} 2\pi - |\alpha - \beta| & \text{falls} \quad |\alpha - \beta| \geq \pi \\ |\alpha - \beta| & \text{sonst} \end{cases} \tag{1.3}$$

Die *Repräsentation eines Bildes I* erfolgt als zweidimensionale diskrete Funktion über einem begrenzten Intervall, festgelegt durch die Auflösung pro Zeile und Spalte $Res = (x_{Res}, y_{Res})$:

$$I(x, y) = [0, 255], \quad 0 \leq x \leq x_{Res}, \quad 0 \leq y \leq y_{Res} \tag{1.4}$$

Normalerweise ist ein solches Bild das Ergebnis eines Abtastungsprozesses eines analogen Bildes durch eine Kamera.

Eine *Bildsequenz* ist eine Folge von Bildern derselben Auflösung, aufgenommen in äquidistanten Zeitschritten t. Ein *Frame* $I(t)$ ist ein Bild einer Bildsequenz zum Zeitpunkt t.

Ein *Feature* ist ein Anhaltspunkt, ein Bildmerkmal, ein leicht zu identifizierender Bildteil, z.B. eine Ecke. Features können einzelne Pixel, zusammenhängende Bildbereiche oder Eigenschaften einer Menge von Pixeln sein.

Die Funktion

$$\text{argmin}_{v_1, v_2, ..., v_n} F(v_1, v_2, ..., v_n) = (m_1, m_2, ..., m_n) \tag{1.5}$$

liefert das Tupel der Argumente $(m_1, m_2, ..., m_n)$, für das die Funktion F ein globales Minimum hat,

wenn die Variablen v_i in ihren Wertebereichen variiert werden. Dies entspricht einer erschöpfenden Suche. Analog dazu wird argmax definiert.

Eine *Trajektorie* ist eine Funktion des Ortes eines Körpers abhängig von der Zeit. In einem allgemeineren Sinn ist sie eine Funktion des Zustandes eines Objektes abhängig von der Zeit. Dieser Zustand kann neben dem Ort auch Rotationswinkel, Impuls und innere Zustände des Objektes umfassen (z.B. die Stellung eines Gelenks des Objektes).

Verfolgung eines Objektes (Tracking) bedeutet die Bestimmung seiner Trajektorie. Ziel ist also, den Zustand des Objektes zu jedem Zeitpunkt zu kennen.

Ein *Farbblob* oder Blob ist laut [39] „etwas stumpfes, rundes; ein kleiner Tropfen oder Klumpen mit irregulärer Form [...]". Dieser Begriff wird als Bezeichnung für zusammenhängende Bildbereiche ohne fest im Vorhinein gegebene Begrenzung verwendet.

Eine *Heuristik* ist eine „...Daumenregel, eine Vereinfachung oder eine gezielte Vermutung (educated guess), die die Lösungssuche in schwer oder kaum verstandenen Bereichen auf ein kleineres Gebiet eingrenzt. Im Gegensatz zu Algorithmen garantieren Heuristiken keine optimalen oder auch nur brauchbare Lösungen und werden häufig ohne theoretische Garantie eingesetzt" [31].

1.4 Ansätze

In diesem Abschnitt sollen die beiden grundlegenden Begriffe dieser Arbeit - Bildverstehen und Wahrnehmung - dargestellt werden. Es werden Alternativen zum gewählten Ansatz vorgestellt und diskutiert. Aufgrund des begrenzten Platzes kann nur ein Einblick in die Literatur gegeben werden.

1.4.1 Das Bildverstehen

Der *Vorgang des Bildverstehens* hat als Eingabe ein Bild und liefert als Ausgabe eine abstrakte Beschreibung des Bildinhaltes. Er arbeitet in entgegengesetzter Richtung zum Rendering bzw. Raytracing, wo eine dreidimensionale Szenenbeschreibung gegeben ist und ein Bild entsprechend einer gegebenen Kameraposition berechnet werden soll.

Um die visuelle Welt verstehen zu können, ist es häufig notwendig, über das, was wir zu sehen erwarten, weitgehende Annahmen zu treffen. Dies führt zu einem *modellbasierten Ansatz* im Bildverstehen, wobei vordefinierte Modelle verwendet werden, um Objekte in der Szene zu identifizieren und zu verfolgen.

Ein *Modell* ist das Resultat eines Abstraktionsvorganges. Es ist damit ein vereinfachtes Abbild der Realität.

Modelle können nicht nur die planare oder räumliche Struktur der zu sehenden Objekte umfassen, sondern auch deren Oberflächeneigenschaften (Farben, Texturen, Glanzlichter), mögliche Bewegungen, ihre Verformung, die Beleuchtung der Szene u.a.

Zur Abbildung von Bilddaten auf Modellparameter sind im allgemeinen zwei Vorgänge notwendig:

1. Featureextraktion zur Generation von Kandidaten für die Modellparameter

2. Auswahl der besten Kandidaten für das Modell durch den Einsatz von Optimierungstechniken; dazu werden die aus den Kandidaten gebildeten Modellhypothesen mit dem Eingabebild verglichen und bewertet

Die Bewertung von Hypothesen erfolgt meist unter Verwendung von Abstandsmaßen (Metriken). Sie ermöglichen die Abschätzung der Eignung von Hypothesen für Modell und Eingabebild. Je umfassender und detaillierter dabei das Modell ist, desto genauer können Hypothesen für Objektkandidaten geprüft werden, desto genauer sind Objekttrajektorien bestimmbar und desto robuster (unempfindlicher gegenüber Störungen) wird die Erkennung. Jedoch bedeutet eine größere Detailtreue auch einen größeren Erkennungsaufwand [19].

Für eine abstrakte Beschreibung von Bilddaten ist ausschlaggebend, daß bei der Rekonstruktion eines Bildes aus der abstrakten Beschreibung das Wesentliche des Originalbildes wiedergegeben wird. Die mit der Abstraktion erreichte Datenreduktion macht eine große Datenmenge faßbar und handhabbar.

Ein Beispiel für die Verwendung einer Metrik ist die Prüfung einer Hypothese mittels der 2-Norm des Differenzbildes [24]. Das Differenzbild entsteht durch Subtraktion der Bildhelligkeiten des zu analysierenden Bildes I_1 und eines unter Verwendung der Hypothese erzeugten Bildes I_2:

$$d(I_1, I_2)(x, y) = i_1(x, y) - i_2(x, y) \tag{1.6}$$

$$d(I_1, I_2) = \sqrt{\sum_{x,y} \left(d(I_1, I_2)(x, y)\right)^2} \tag{1.7}$$

Diese Norm wird als Fehlermaß verwendet. Stimmen die beiden verglichenen Bilder weitgehend überein, ist obige Summe minimal.

Viel Raum bei der Forschung im Bereich des Bildverstehens nehmen dreidimensionale Ansätze ein. Das Ziel ist, über eine Tiefenrekonstruktion zu einem dreidimensionalen Szenenmodell zu gelangen. Untersucht wird vor allem das Problem der Stereokorrespondenz, das beim binokularen Sehen auftritt. Gegeben sind dabei zwei Bilder mit leicht verschobenen Kamerastandpunkten. Gesucht ist eine Zuordnung der Pixel beider Bilder zu Punkten in der 3d-Szene. Aus der Verschiebung solcher Punkte in beiden Bildern läßt sich die Entfernung zwischen dem jeweiligen Punkt in der dreidimensionalen Szene und dem Beobachter feststellen. Je geringer die Entfernung des Szenenpunktes ist, desto größer ist die Verschiebung.

Über die Erkennung von Objekten aus Bildsequenzen hinaus, erfolgt in der „höheren Bilddeutung" die Auseinandersetzung mit Repräsentationen, Ereignissen und Prozessen, die auf Objekten aufbauen. So werden beispielsweise von einem Fahrassistenzsystem benachbarte Fahrzeuge im Straßenverkehr sowie ihre Bewegungen erkannt. Mittels höherer Bilddeutung lassen sich aus diesen Daten wiederum Spurwechsel, Überholmanöver und Abbiegevorgänge erkennen.

1.4.1.1 Verfolgen

Verfolgung von Objekten ist auf einfachste Weise möglich, indem sie in jedem einzelnen Frame einer Bildsequenz erkannt werden und die Trajektorien mit dem aktuellen Objektzustand aktualisiert werden. Jedoch sind die von der Bilderkennung gelieferten Daten unzuverlässig oder für manche Frames vielleicht nicht vorhanden. In vielen Fällen wird zur optimalen Integration von Zustandsvorhersagen der

Kalman-Filter [27] eingesetzt. Dieser umfaßt die Schritte der Zustandsvorhersage und des Zustandsupdates, wobei jeweils die Erkenntnisse des aktuellen Frames optimal entsprechend ihrer Zuverlässigkeit integriert werden.

Auch bei der Zustandsvorhersage können Modellannahmen über die Objektbewegung genutzt werden. Das Modell eines fahrenden Autos könnte z.B. die möglichen Positionen und Geschwindigkeiten beschränken. Dies ist möglich, weil Autos normalerweise nur vorwärts oder rückwärts, aber nicht seitwärts fahren. Ebenso besteht für die Beschleunigung aufgrund der Motorstärke und Masse des Objektes eine Obergrenze. Sind Ort, Beschleunigung und Geschwindigkeit einmal bis auf ein bestimmtes Maß bekannt, dann ist auch der mögliche Aufenthaltsort des Objektes, weil es in einer gegebenen Zeit nur eine bestimmte Entfernung zurücklegen kann. Liefert der Erkennungsschritt für den aktuellen Frame Daten, die aufgrund der Bedingungen des Bewegungsmodells unmöglich sind, können diese zurückgewiesen und eine andere Hypothese bevorzugt werden.

Ausgehend vom Ansatz des modellbasierten Verfolgens ist es ein kleiner Schritt, einen allgemeinen Algorithmus zu konzipieren, der die Verfolgung für ein beliebiges vorgegebenes zwei- oder dreidimensionales geometrisches Oberflächenmodell (oder für noch allgemeinere Modelle, die eine Rekonstruktion des Aussehens der Objekte ermöglichen) durchführt. Auf dieser Abstraktionsebene existieren jedoch nur spezialisierte Anwendungen für bestimmte Objekttypen, wie Werkstücke von Industrierobotern [20], Gesichter, Hände, Menschen und Autos.

Probleme stellen die Verformbarkeit der Objekte, sowie Okklusionen, also Verdeckung von Objektteilen, dar. Um Okklusionen einzuschränken, werden deshalb zum Tracking von Schauspielern für die Gewinnung von Animationsdaten für Kinofilme und Computerspiele mehrere Kameras gleichzeitig eingesetzt [35].

Was sind die Alternativen zum modellbasierten Verfolgen? Zwei sollen hier genannt werden.

Die erste ist der *Lucas-Kanade-Tomasi-Tracker* (LKT-Tracker) [21, 2]. Er basiert auf der Verfolgung von Featurepixeln, die im Voraus gewählt wurden. Gut eigenen sich dafür Bildpunkte, die Ecken darstellen und einen Punkt auf einem echten Objekt in der Szene repräsentieren. Danach wird die Verschiebung der Features zwischen den Frames unter Verwendung von Techniken zur Berechnung des optischen Flusses berechnet. Ausgabe ist also die Abbildung von Featurepositionen zwischen Frames. LKT-Tracker sind vor allem einsetzbar, wenn über die Szene wenig bekannt ist und überhaupt erst eine Segmentierung in Objekte erfolgen soll. Gut geeignet sind sie auch zur Bildstabilisierung z.B. in Camcordern.

Jedoch ließe sich für unsere Zielstellung ein LKT-Tracker allein nicht verwenden. Ihm könnten zwar Featurepixel vorgegeben werden, aber die Zuordnungen von Featurepixeln zu Punkten auf der Objektoberfläche oder dem Hintergrund müßten zuvor bekannt sein. Diese Zuordnung ist aber gerade der Kern der Erkennung! In jedem Fall muß eine initiale Zuordnung zu den Modellparametern erfolgen. Ist sie bekannt, wäre es möglich von der Featurebewegung auf unsere Modellparameter Rückschlüsse zu ziehen. Ein weiteres Problem wäre, daß die Drehung des Kreisringes im optischen Flußbild nicht sichtbar ist, da die Oberfläche des Kreisrings im Kamerabild während der Raddrehung nur eine Rotation ausführt und keine nennenswerte innere Struktur besitzt, an der Features festgemacht werden könnten. Es blieben also nur die Arme und bei diesen vor allem die Eckpunkte und die auf ihnen montierten LEDs.

Ein weiterer Tracking-Algorithmus ist *CamShift* [3]. Dieser Algorithmus bekommt eine Zielfarbe (hue) sowie Position und Größe eines Suchfensters vorgegeben. Das Suchfenster entspricht dem zu verfolgenden Blob und ermöglicht die Verfolgung unabhängig von anderen Blobs derselben Zielfarbe im Bild. Für eine Bildregion, die entsprechend der erwarteten Bewegung des Blobs über das Suchfenster hinausgeht, wird

mittels eines Histogramms und Farben-Rückprojektion ein Farbwahrscheinlichkeitsbild berechnet. Darin haben Pixel der Zielfarbe eine sehr hohe Wahrscheinlichkeit und unähnliche Farbtöne eine sehr niedrige. Durch Gewichtung der Pixel im Suchfenster entsprechend ihrer Wahrscheinlichkeit wird der Schwerpunkt des Blobs berechnet. Aus der Fläche bzw. Anzahl der Objektpixel wird die Größe des neuen Suchfensters bestimmt.

Für die vorliegende Aufgabenstellung ließe sich CamShift nur zur Verfolgung der Leuchtdioden einsetzen, da sie die einzigen Farb-Blobs (außer dem gesamten Rad) darstellen.

Überraschenderweise scheint es zur Verfolgung anhand eines Modells letztendlich keine Alternativen zu geben. Nicht-modellbasiertes Verfolgen erscheint unmöglich, selbst wenn man nur Features wie mit dem LKT-Tracker oder Blobs wie mit CamShift verfolgt. Selbst in diesen Fällen verwendet man Modelle, wenn gleich sehr einfache. Diese basieren nicht auf physikalischen Objekten, sondern auf optischen Eigenschaften von Bildregionen.

1.4.2 Die Wahrnehmung

Die biologische Wahrnehmung einschließlich physiologischer Grundlagen und der Verarbeitungsprozesse der Sinnesreize wird in der Wahrnehmungspsychologie untersucht. Der Vorgang der Wahrnehmung wird von ihr folgendermaßen erfaßt [44]:

> „In der (kognitiven) Psychologie und in der Physiologie bezeichnet Wahrnehmung die Summe der Schritte Aufnahme, Interpretation, Auswahl und Organisation von sensorischen Informationen - und zwar nur jener Informationen, die zum Zwecke der Adaption des Wahrnehmenden an die Umwelt oder ihrer Modifikation aufgenommen werden. Gemäß dieser Definition sind also nicht alle Sinnesreize Wahrnehmungen, sondern nur genau jene, welche geistig auch verarbeitet werden."

Gemäß dem Symbolverarbeitungsansatz [14] der künstlichen Intelligenz, muß o.g. geistige Verarbeitung durch symbolverarbeitende Systeme erfolgen. Die Theorie der Wahrnehmung muß also eine Antwort auf die Frage liefern, wie symbolverarbeitende Systeme mit der Welt verbunden werden können. Diese Frage wird *symbol grounding* genannt und zielt auf die Verankerung der Symbole in der Welt [15]. Der Vorgang der Wahrnehmung entspricht einer Analyse, die die Symbolverarbeitung erst ermöglicht. Denn einerseits erfolgt eine starke Datenreduktion und andererseits werden so überhaupt erst manipulierbare Symbole mit einem Bedeutungsinhalt erzeugt.

In der Logikprogrammierung z.B. mit der Sprache Prolog oder dem System *smodels* [22] erfolgt das Grounding durch die Belegung der Variablen mit Werten. Die Bestimmung der Werte wiederum erfolgt meist durch direkte Eingabe. Es werden aber auch komplexe Systeme gebaut, die durch Wahrnehmungskomponenten mit der Welt verbunden sind. Die Wahrnehmungskomponenten verwenden häufig neuronale Netze. Neuronale Netze sind aus biologisch inspirierten konnektionistischen Ansätzen entstanden und arbeiten nach dem Vorbild des Gehirns. Sie stehen im Gegensatz zu allein mathematisch inspirierten Ansätzen, wie dem in dieser Arbeit verwirklichten.

1.4.2.1 Struktur vs. Inhalt

Nimmt man analog zur Medium-Botschaft-Dichotomie[2] eine analytische Unterteilung der Wahrnehmung in Wahrnehmungstrukturen und Wahrnehmungsinhalte vor, ergeben sich folgende Fragestellungen, die auch Gegenstand der Erkenntnistheorie sind:

- Wie sieht die Unterteilung im konkreten Fall aus?
- Wie funktionieren Wahrnehmungsstrukturen? Wie könnten sie funktionieren? Welche Teilprozesse gibt es? Auf welche Weise laufen sie ab?
- Wie entsteht die Unterteilung im Subjekt? Anders formuliert: Wie entstehen Wahrnehmungstrukturen? Für diese Arbeit bedeutet das: „Woher kommen die Modelle?" Gibt es Anlässe zur Entstehung neuer Strukturen?

Den ersten beiden Punkten wird in der vorliegenden Arbeit im Falle einer visuellen Fragestellung nachgegangen. Das entwickelte Erkennungssystem stellt dabei mit seiner Hard- und Software die Wahrnehmungsstruktur dar. Diese verkörpert einerseits „das Wahrnehmende", indem es das der Erkennung zugrundeliegende Modell implementiert. Andererseits wird durch dieses Modell gleichzeitig „das Wahrnehmbare" vorgegeben.

Die Wahrnehmungsinhalte sind die Eingabebilder, die bei der Erkennung entstehenden Daten und die vom Erkennungssystem ausgegebenen Parameter der Räder. Diese Parameter stellen Parameter des Modells dar.

Platon schrieb einst „Erkennen ist Wiedererkennen" als Grundüberlegung für seine Ideenlehre[41], nach der neben unserer sichtbaren, sinnlich wahrnehmbaren Welt eine zweite, die unsichtbare, aber ungleich seinsmächtigere Geist-Seele des Menschen steht, in welcher die grundlegenden geistigen "Muster" vorhanden sind, nach denen wir die Welt erkennen. Diese "Muster" werden von ihm Ideen genannt. Die Ideen haben nach Platons Überzeugung ein geistiges Sein, deren Substanz durch Teilhabe im Sein, dem Grund schlechthin gründet. Erkenntnis ist deshalb nicht Abstraktion, wie später Aristoteles behauptet, sondern wir erkennen beim Betrachten der Welt die von unserer Geist-Seele im Moment der Erkenntnis erinnerten "Ideen" (Anamnesis), in jeweils unterschiedlicher Ausprägungsform dem sinnlichen Eindruck angepaßt.

Diese Lehre setzt voraus, daß alles Erkennbare schon im voraus bekannt sein muß. Wenn nun die Wahrnehmung dem Wiedererkennen oder der Erinnerung entspricht, wird also ein Modell vorausgesetzt. Platon begründete das Vorhandensein der Erinnerungen durch die Theorien der Ideenschau und der Wiedergeburt. Vom Standpunkt des Empirismus, der alle Erkenntnis aus Sinneserfahrungen ableitet, ist diese Begründung nicht zu akzeptieren. Es stellt sich erneut die epistemologische Frage, wie die Erfahrung neuer Erkenntnisse möglich ist, wenn nicht durch Wahrnehmung?

Einen Ausweg verspricht der Vorgang des Lernens bzw. der Modellbildung. Wie läuft er ab? Gibt es „Regeln" zur Modellerstellung? Dieser Frage (die dem dritten Punkt oben entspricht) wird hier nicht weiter nachgegangen. Sie führt über die Wahrnehmung hinaus.

[2] In einem entfernteren Sinne kann man auch eine Analogie zum Gegensatz Neuron-Erregung sehen.

1.4.2.2 Der Wahrnehmungsprozeß

Die Funktionsweise der Wahrnehmung läßt sich aus ihren Teilprozessen ableiten. Diese sind nach obiger Definition „Aufnahme, Interpretation, Auswahl und Organisation von sensorischen Informationen". Bei der Betrachtung als informationsverarbeitenden Prozeß sind dabei Ein- und Ausgabe vorgeschrieben: Die Eingabe bilden von den Sensoren gelieferte Informationen. Die Ausgabe erfolgt auf einer abstrakteren Ebene, der Ebene des der Erkennung zugrundeliegenden Modells.

Den Kernprozeß scheint die *Interpretation* darzustellen. So wie ein Übersetzer die Verbindung zwischen zwei Sprachen herstellt und dafür Kenntnis beider Sprachen haben muß, erfordert die Übersetzung von Sensordaten in „Modelldaten" (Modellparameter) „Kenntnisse" beider Seiten. Da auf der Ebene der Wahrnehmungsstrukturen kein wahrnehmendes Subjekt lokalisierbar ist, können diese Kenntnisse nur durch die Wahrnehmungsstrukturen selbst und ihre Funktionsweise dargestellt werden.

Kenntnisse über die Sensordaten ermöglichen eine *Analyse* der aufgenommenen Informationen. Sie werden gruppiert und in Zusammenhänge gestellt. Die dabei verwendeten Grenzen sind durch die Wahrnehmungsstruktur festgelegt.

Kenntnisse über das Modell ermöglichen eine *Konstruktion* von Modellhypothesen. An dieser Stelle entstehen durch Unvollständigkeit der Sensordaten Probleme. Zusätzliche Modellkenntnisse ermöglichen den Ausschluß ungültiger Konstruktionen.

Die genannte *Auswahl* bestimmter sensorischer Informationen wird einerseits durch das Modell vorgegeben - nur im Modell zulässige sensorische Informationen können ausgewählt werden. Andererseits weist dieser Teilprozeß auf die aktive Steuerung der Wahrnehmung durch die Aufmerksamkeit des wahrnehmenden Subjekts bzw. der wahrnehmenden Struktur hin. Wenn dieselben Eingaben verschiedene gültige Ausgaben zulassen, also ein Fall von Mehrdeutigkeit vorliegt, kann eine Auswahl erfolgen.

Im Zuge der Abstraktion erfolgt eine starke Datenreduktion.

Im hier entwickelten Programm ist das „generate & test"-Paradigma wiederzufinden. Dieses wird häufig bei bottom-up-Lösungsstrategien eingesetzt. Es besagt: „Generiere eine (willkürliche) potentielle Lösung und verifiziere anschließend, ob die Lösung auch den Anforderungen genügt" [26, 1].

Die in Abschnitt 1.4.1 genannten Schritte „Featureextraktion zur Generation von Parameterkandidaten" und „Optimierung zur Abbildung der Kandidaten auf das Modell" stellen die Verwirklichung dar. Die Hough-Transformation beispielsweise generiert Kandidaten für die Armaußenkanten von micro.adam. Danach werden diese Kandidaten auf bestimmte Eigenschaften getestet (siehe Abschnitt 3.1.2).

Constraints (Ausschlußbedingungen) können bei beiden Schritten aktiv sein: Einmal sollten nur „sinnvolle" Kandidaten generiert werden, also Kandidaten, die vorgegebenen Constraints nicht offensichtlich widersprechen. Andererseits sind Constraints die Grundlage der Hypothesentests.

Beispielsweise werden später bei der Erkennung des Arms von micro.adam (siehe Abschnitt 3.1.2) nur sinnvolle Kandidaten für die Armaußenkanten aufgestellt, denn die Hough-Transformation liefert Geraden, die mit hoher Wahrscheinlichkeit im Bild wiederzufinden sind. Danach wirken Constraints, weil sie aus den Kandidaten die Geraden auswählen, die im richtigen Abstand und parallel zueinander sind.

Die Generierung „sinnvoller" Kandidaten ist gleichbedeutend mit dem Einsatz einer Heuristik.

Die abgeschwächte Form dieses Prinzips ist „trial and error", wie es in genetischen Algorithmen angewendet wird. Dabei werden die Kandidaten nicht gezielt aufgestellt.[3]

[3] Andererseits erfolgt die Auswahl der überlebenden Individuen zielgerichtet, so daß auch genetische Algorithmen keine

1.5 Anwendungen

Ziel der wissenschaftlichen Auseinandersetzung mit den Themen des Bildverstehens und der Wahrnehmung ist ein besseres Verständnis der zugrundeliegenden Prozesse. Gleichzeitig gibt es ein breites Anwendungsspektrum für die dabei entwickelten Techniken und Systeme. Dieses Spektrum reicht von Robotik (Steuerung autonomer und nichtautonomer Roboter, „Visual Servoing") und Videoüberwachung zu „Augmented Reality" (s.u.), medizinischer Bildverarbeitung und industriellen Anwendungen, z.B. der optischen Materialkontrolle.

„Visual Servoing" setzt sich mit der Steuerung von Robotern auseinander. Ziel ist die Ansteuerung von Aktuatoren auf der Basis visueller Wahrnehmung der zu manipulierenden Objekte (z.B. Werkstücke) und der Position des Roboters selbst. Es schließt die Verwendung einer oder mehrerer Videokameras und eines Bilderkennungssystems ein, um die Position des „Roboterarms" relativ zum Werkstück je nach Arbeitsaufgabe zu überwachen.

„Augmented Reality" ist ein Teilbereich der „Virtual Reality"-Forschung. Ein AR-System generiert für seinen Benutzer eine zusammengesetzte Sicht. Diese ist eine Kombination einer realen Szene mit einer virtuellen computergenerierten Szene, die die reale Szene mit zusätzlicher Information anreichert. Die zusätzliche Information verbessert die Wahrnehmung und Leistung des Benutzers. Anwendungen lägen z.B. in der Fahrzeugnavigation und der Medizin. Ein Chirurg könnte durch ein AR-System unterstützt werden, indem die aus präoperativen Scans gewonnenen Informationen mit der realen Sicht verbunden werden. Diese Sicht könnte die Leistung des Chirurgen verbessern und vielleicht sogar andere Darstellungsformen ersetzen.

reine Realisierung von „trial and error" sind.

Kapitel 2

Herangehensweise

In diesem Abschnitt soll eine Eingrenzung der Aufgabenstellung erfolgen, sowie erklärt werden, wie der Lösungsweg aussehen kann (Methodologie).

2.1 Eingrenzung der Aufgabenstellung

Die Parameter (Rad- und Armpositionen) könnten durch einfachere Techniken als Bilderkennung gemessen werden oder von der Steuerung selbst geliefert werden. Einsetzbar sind z.B.:

- Kontakte: Die Kontakte könnten ein Signal liefern, nachdem sich das Rad um einen bestimmten Winkel weitergedreht hat. In Version 2 der Räder wurde diese Möglichkeit bereits eingebaut.
- Odometrie: Odometrie ist die Mitführung eines Wertes innerhalb eines globalen Bezugssystems (z.B. Position eines Roboters auf einer Karte oder die Armposition von micro.adam) auf Basis lokaler Messungen (z.B. zurückgelegte Wegstrecke und Richtung pro Zeiteinheit oder Werte, mit denen die Motoren zur Armbewegung von micro.adam angesteuert werden). Durch ungenaue Messungen und Fehlerfortpflanzung ist aber keine große Genauigkeit erreichbar. Zusätzlich ist der Ausgangszustand unbekannt, was eine weitere Schwierigkeit darstellt.

Es sollen jedoch Bilderkennungsalgorithmen eingesetzt werden. Die o.g. alternativen Informationsquellen werden höchstens zur Evaluation herangezogen.

Die Vorgabe der speziellen Objekte micro.adam und micro.eva ermöglicht die Festlegung spezifischer Modelle und Ausgaben. Die „Allgemeine Sehaufgabe", also die Konstruktion eines Sehsystems, das so flexibel wie das des Menschen ist, konnte vor allem deshalb bisher nicht gelöst werden, weil die Erfüllung der Aufgabe die Entscheidung erfordert, festzulegen was interessant ist und gesehen werden soll.

Die Echtzeitanforderung ergibt sich aus der Zielstellung der Steuerung der Räder. Dazu ist nicht unbedingt die volle Framerate der Kamera erforderlich. Ausschlaggebend für die Erfüllung der Echtzeitanforderung ist, daß die Radsteuerung mit der erreichten effektiven Framerate funktioniert. Im übrigen erfüllt sich diese Forderung im Zuge der Weiterentwicklung der Rechentechnik fast von allein.

Die Verwendung der gewonnen Daten zur Steuerung wird unter Übernahme des alten Steuerungsansatzes anhand von micro.adam demonstriert. Hier existieren noch weitgehende Ausbaumöglichkeiten.

2.2 Lösungsweg

Schon durch das Thema ist ein Großteil des Vorgehens festgelegt.

1. Es muß ein Modell entworfen werden, das die Wahrnehmungsinhalte entsprechend der geforderten Weiterverarbeitungsmöglichkeiten abbildet.
2. Danach muß durch die passenden Algorithmen eine Abbildung der Bilddaten auf das Modell erfolgen. Der schwierige Schritt ist dabei die Suche nach geeigneten Algorithmen. Sie müssen die erforderlichen Abstraktionsschritte unterstützen, dabei zuverlässig sein und mit verrauschten oder unvollständigen Daten zurechtkommen. Die übliche Vorgehensweise zur Verarbeitung visueller Daten ist es, von den konkreten Pixeln auszugehen und sie zu abstrakteren Einheiten wie Kanten und Flächen zusammenzufassen. Auf einer höheren Ebene werden dann Objektumrisse und letztendlich Objekte repräsentiert.
3. Letztendlich muß eine Implementation erfolgen. Die eingesetzten Algorithmen müssen verkettet werden. Das erfordert meist ein kleines Framework zum Aufbau von Datenstrukturen und zur Initialisierung und die Umwandlung verschiedener Werte (technisches Wissen).
4. Die Erkennungsergebnisse sollten bewertet werden (Evaluation). Eine formale Evaluation der Erkennungsergebnisse wurde nicht durchgeführt, da durch eine Sichtkontrolle ihre Genauigkeit und Zuverlässigkeit in subjektiv ausreichendem Maße ersichtlich war.
5. Die erkannten Parameter müssen zur Steuerung verwendet werden. Da es möglich ist die erkannten Radparameter auf die Sensorwerte des Gyroskops abzubilden, kann der alte Steuerungsansatz fast direkt übernommen werden.

Oberstes Ziel ist es zu zeigen, daß die Erkennung überhaupt möglich ist. Zwar bestreitet kaum jemand, daß maschinelle Bilderkennung möglich ist. Erinnert sei an erfolgreiche Anwendungen wie Fingerabdruckerkennung, optische Zeichenerkennung und die Erkennung von Nummernschildern durch das Mautsystem, das nun doch bald eingeführt wird. Jedoch sind diese Anwendungen - außer der optischen Zeichenerkennung - noch keine alltägliche Selbstverständlichkeit. Gleichzeitig ist das Wissen um die Funktionsweisen maschineller Bilderkennung sehr begrenzt.

An zweiter Stelle wird dann die Geschwindigkeit der Erkennung optimiert. Die Messung der effektiven Framerate, sowie der Verarbeitungszeiten der einzelnen Erkennungsschritte liefern Anhaltspunkte für die Optimierung.

2.3 Die Modelle

Modellbasierte Verfolgung bedeutet, daß im Computer Wissen über das zu verfolgende Objekt gespeichert sein muß. Ein Teil dieses Wissens ist das Modell, das der Erkennung zugrunde liegt. Es ist quasi die Vorstellung des Computers, von dem, was er sieht.

Die im Folgenden gewählten Modelle sind nicht dynamisch, d.h. sie lassen die Entwicklung der Parameter im Verlaufe der Zeit außer Betracht. Sie sind erweiterbar durch Radgeschwindigkeit und -beschleunigung. Daraus ließen sich zusätzliche Constraints ableiten, die eine zuverlässigere Entscheidung zulassen, ob die

gefundenen Parameter konsistent, also im gegebenen Modell möglich sind. Es bieten sich z.B. Obergrenzen für Geschwindigkeit und Beschleunigung an.

Eine Radgeschwindigkeit von schätzungsweise 5 Umdrehungen pro Sekunde ist aufgrund der Bauweise der Räder nicht erreichbar. Auch für die Beschleunigung gibt es Grenzen. Die positive Beschleunigung ist begrenzt, da die Räder nur durch ihre mit begrenzter Motorleistung erzeugten Armbewegungen an Geschwindigkeit gewinnen können. Die negative Beschleunigung ist zusätzlich dadurch begrenzt, daß die Räder keine Bremsen besitzen und nirgends anschlagen können.

2.3.1 Das Modell von micro.adam

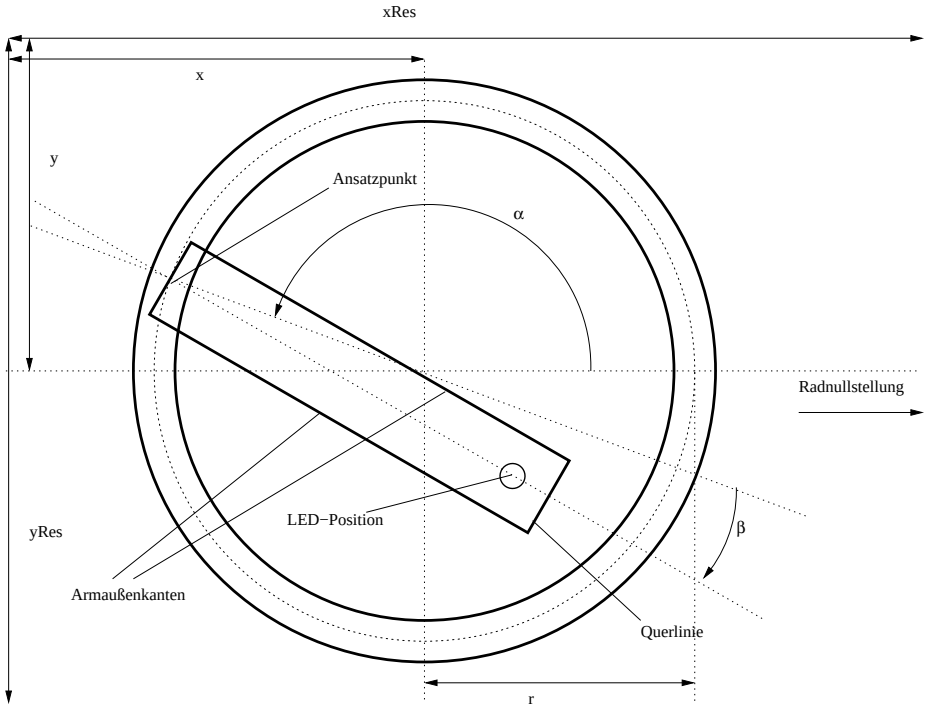


Abbildung 2.1: Skizze von micro.adam mit eingezeichneten Modellparametern

Dieses Modell wird durch 4 Parameter charakterisiert :

1. r - Radius des Rades in Pixeln. Da es aus einem Kreisring besteht, wird hier die Mitte von Außen- und Innenradius verwendet. Das Rad sollte vollständig im Bild sichtbar sein.
   $0 < r < \frac{\min(x_{res}, y_{res})}{2}$ (Hierbei sind x_{res} und y_{res} die Breite und Höhe des Kamerabildes)
2. $M = (x_M, y_M)$ - Koordinaten des Mittelpunkts des Rades im Kamerabild in Pixeln.
   $r < x_M < x_{res} - r$, $r < y_M < y_{res} - r$
3. α - Winkel, um den das Rad aus der Nullstellung verdreht ist; bestimmt den Ansatzpunkt des Armes; $0 \leq \alpha < 2\pi$

4. β - Winkel, um den der Arm aus seiner Nullstellung bewegt wurde; Angelpunkt ist der Ansatzpunkt des Armes; $-20° < \beta < 20°$; Wenn $\beta < 0$ ist, dann befindet sich der Arm vom Armansatzpunkt aus in Richtung Radmitte gesehen, links

Als Nullstellung wird die Radstellung angesehen, bei der der Armansatzpunkt genau rechts im Bild liegt ($\alpha = 0$) und der Arm von seinem Ansatzpunkt zur Radmitte zeigt ($\beta = 0$).

Des weiteren werden folgende allein vom Radradius abhängige Verhältnisse genutzt:

1. Abstand von Außen- und Innenradius (Radringdicke): $dicke = 0.21\ r$
2. Armlänge: $armlaenge = 1.65\, r$
3. Armbreite: $breite = 0.2\, armlaenge$
4. Position der LED auf dem Arm: $0.92 * armlaenge$ (Die LED wird in der Armmitte am Armansatzpunkt gegenüberliegenden Armende mit einem Radius von $0.05 * r$ eingezeichnet.)

Zusammen liefern diese Parameter und Verhältnisse eine eindeutige abstrakte Beschreibung des Aussehens des Radsystems in einem Bild.

2.3.2 Das Modell von micro.eva

Dieses Modell wird durch 8 Parameter charakterisiert:

1. r - Radius wie bei micro.adam
2. M - Mittelpunkt wie bei micro.adam
3. $\alpha \in [0, 2\pi)$ - Winkel, um den das Rad aus der Nullstellung verdreht ist. Da keiner der Arme sichtbar ausgezeichnet ist, kann prinzipiell der Winkel des Ansatzpunktes eines beliebigen Armes als α verwendet werden.
4. bis 8. $\beta_i \in [0, 1]$ - Maß zur Beschreibung der Armstellung, kein Winkel. Eine Zahl reicht aus, da ein Arm einen Freiheitsgrad hat. 0 bedeutet „Arm am Rad angelegt" und 1 bedeutet „maximal nach innen gefahren". $i \in \{0, 1, 2, 3, 4\}$ bezeichnet die Armnummer.

Wie bei micro.adam wird der Abstand von Außen- und Innenradius (Radringdicke) verwendet.

Über die Arme werden folgende Annahmen getroffen:

1. Sind alle Arme „am Rad angelegt" ($\beta_i = 0$), so sind ihre LED-Positionen im Abstand von $\frac{360°}{5} = 72°$ zueinander relativ zum Radmittelpunkt angeordnet.
2. Wird ein Arm „maximal nach innen gefahren" ($\beta_i = 1$), so verschiebt sich seine LED-Position um 20° entgegen dem Uhrzeigersinn relativ zum Mittelpunkt.

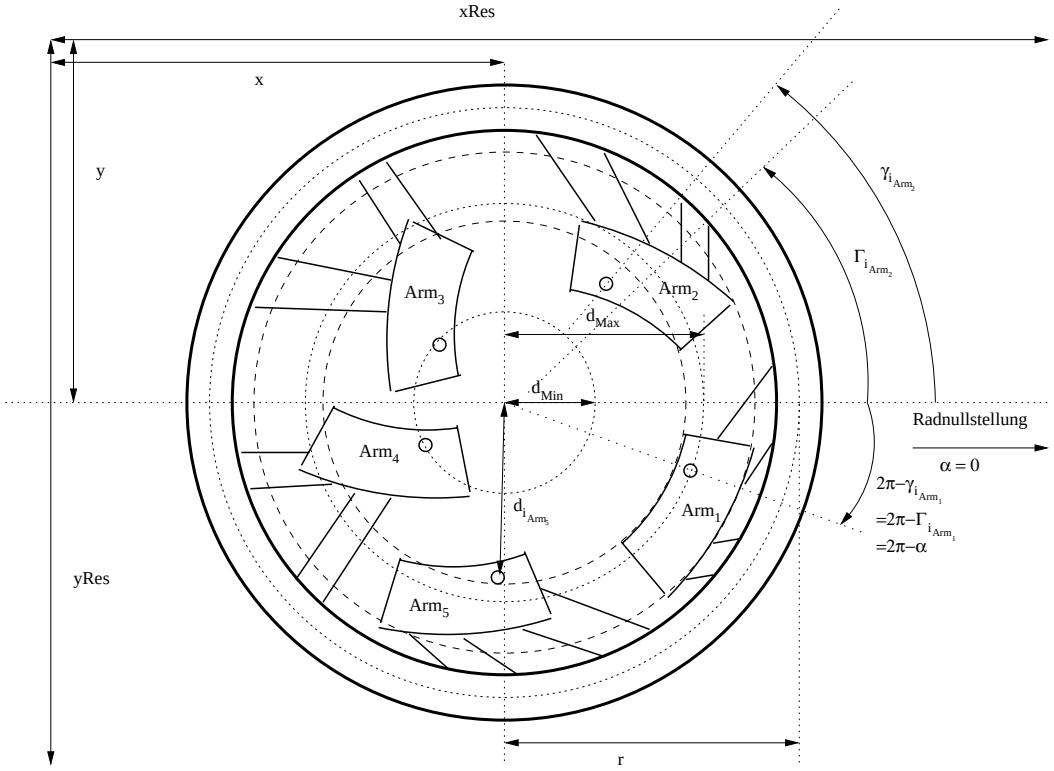


Abbildung 2.2: Skizze von micro.eva

Kapitel 3

Ablauf der Bilderkennung

Die Bildverarbeitung erfolgt allein mit mathematisch motivierten Algorithmen. Die Erkennungsaufgabe wird als nur zweidimensional behandelt, da dies ausreicht. Für beide Radvarianten erfolgt eine ähnliche Verarbeitung der Bilddaten.

Ein Videogerät liefert *Frames* $F(t)$ in äquidistanten diskreten Zeitschritten $t \in \mathbb{N}$ in einer bestimmten Auflösung (x_{res}, y_{res}). Eine Webcam oder Videokamera liefert meist 25 Bilder/s. Nach einer festen Anzahl Frames wird ein Frame als *Keyframe* bezeichnet und gesondert behandelt. Für Keyframes erfolgt eine Neubestimmung des Radmittelpunkts, zusätzlich zur Bestimmung der Armstellungen. Für normale Frames wird nur der durch Radradius und Radmittelpunkt beschriebene Bildausschnitt betrachtet. Dies bringt für normale Frames eine starke Reduktion der zu verarbeitenden Bilddaten - je nach Radgröße im Bild etwa 60%.

Ziel der Verfolgung von Objekten in Bildsequenzen ist, eine *Trajektorie* zu gewinnen, also die Kurve, welche das Objekt im Raum beschreibt. Die Berechnung einer Trajektorie kann einerseits erfolgen, indem sie aus den Positionen/Konfigurationen des Objekts in den Einzelbildern zusammengesetzt wird. Oder - was zur Verringerung des Rechenaufwandes sinnvoller ist - man nutzt die bereits gewonnene Trajektorie, um mit ihrer Hilfe den Suchraum einzuschränken. Beispielsweise erfolgt die Suche eines Objektes im nächsten Frame nur in dem Bildbereich, in dem es erwartet wird. Größe und Position dieses Bildbereiches sind durch Bewegungsgeschwindigkeit und -richtung des Objektes aus der Trajektorie bekannt.

Die gesamte Erkennungsprozedur berechnet die Funktion

$$F_{adam} : F(t) \mapsto (M, r, \alpha, \beta) \tag{3.1}$$

bzw.

$$F_{eva} : F(t) \mapsto (M, r, \alpha, \beta_i) \tag{3.2}$$

3.1 Erkennungsziele

3.1.1 Bestimmung von Radradius und Radmittelpunkt

Der Erkennung beider Radvarianten gemeinsam sind Bestimmung von Radradius und Radmittelpunkt. Diese Bestimmung wird versucht, wenn Radius oder Radmitte noch unbekannt sind. Für alle Keyframes erfolgt eine Neubestimmung des Mittelpunkts, um eventuelle Veränderungen auszugleichen, wenn die Kameraposition relativ zum Rad leicht geändert wurde.

Das farbige Eingabebild wird in ein Graustufenbild umgewandelt. Darauf wird der *Canny-Algorithmus* (siehe Abschnitt 3.2.1) angewandt, um eine Kantensegmentdarstellung zu erhalten (siehe Abbildung 3.1a). Diese wird im nächsten Schritt der Bildkorrelation gebraucht.

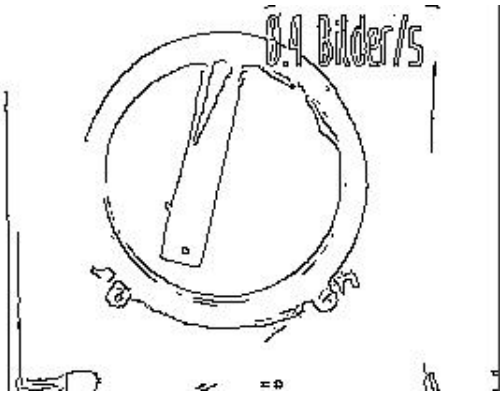
Alternativ zur Kantensegmentdarstellung wurde versucht, eine Objekt-Hintergrund-Trennung mit einem globalen Schwellwertverfahren vorzunehmen, indem aus dem Eingabebild ein Binärbild berechnet wurde. Darin wurden zum Objekt gehörige Pixel gesetzt und zum Hintergrund gehörige Pixel gelöscht. Nachteil dieses Verfahrens war jedoch, daß die Entscheidung, ob ein Pixel zum Objekt gehört oder nicht, allein aufgrund des Grauwerts getroffen wurde. Diese globale Entscheidung hatte zur Folge, daß teilweise Hintergrund und Objekt nicht auseinandergehalten werden konnten, weil sie dieselben Grauwerte aufwiesen (Abbildung 3.1b).

Zu gegebenem Radius r kann mittels *Bildkorrelation* (optische Korrelation, siehe Abschnitt 3.2.2) der Mittelpunkt des Rades im Bild berechnet werden. Korreliert werden hierbei das in ein Kantensegmentbild B umgewandelte Kamerabild in voller Auflösung, sowie eine nach vorgegebenem Radius gezeichnete *Vorlage* V_r. Bei direkter Umsetzung der Bildkorrelation nach Formel 3.36 wird die Vorlage dabei in allen möglichen Verschiebungen über das Kantensegmentbild gelegt und die Korrelation übereinanderliegender Pixel berechnet. In der Implementation erfolgt aus Geschwindigkeitsgründen die Korrelation unter Verwendung der Fourier-Transformation.

Die Vorlage besteht lediglich aus den Kreisringen des Rades. Bei micro.adam kann wahlweise auch eine detailliertere Vorlage mit dem Arm verwendet werden. Für letzteres müssen jedoch zusätzlich α und β vorgegeben werden. Außerdem ergaben sich störende Maxima im Korrelationsbild (Abbildung 3.1f) für den Fall, daß der Arm im Vorlagebild genau waagerecht oder senkrecht verlief, so daß die Vorlage am besten ohne Arme gezeichnet wird. Stimmen Vorlage und Eingabebild nicht exakt überein, ist im Allgemeinen trotzdem ein deutliches Maximum im Korrelationsbild zu finden.

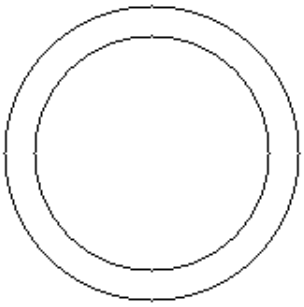
Ausgabe der Bildkorrelation ist ein *Korrelationsbild* $K_r = C(B, V_r)$. Darin sind die Bildhelligkeiten entsprechend der Korrelationsstärke der Eingabebilder bei Verschiebung der Vorlage an den jeweiligen Punkt gesetzt. An der Stelle, an der die Bildhelligkeiten des Korrelationsbildes ein globales Maximum bilden $G_r = (x_G, y_G) = \text{argmax}_{x,y} K_r(x, y)$, passen die Eingabebilder „am besten übereinander". Bei richtig gewählter Vorlage (siehe Abschnitt 3.2.2) kann die Position dieses Maximums direkt als Radmittelpunkt M verwendet werden. Die Korrelationsstärke im Maximum $K_r(x_G, y_G)$ wird als *Korrelationsstärke von Korrelationsbild und Vorlage* bezeichnet.

Wird nun für verschiedene Radien r_i eine Bildkorrelation versucht, kann aus den Absolutwerten der globalen Maxima der Korrelationsbilder $K_{r_i}(x_G, y_G)$ der Radius r des Rades im Eingabebild berechnet werden. Denn für den Fall, daß die Vorlage denselben Radius wie das Rad im Kamerabild hat ($r_i = r$), ist $K_{r_i}(x_G, y_G)$ maximal. Das entsprechende r_i wird als Radius des Rades im Kamerabild angenommen.

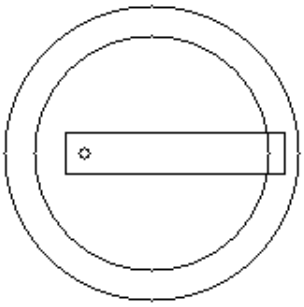


(a) Kantensegmentbild

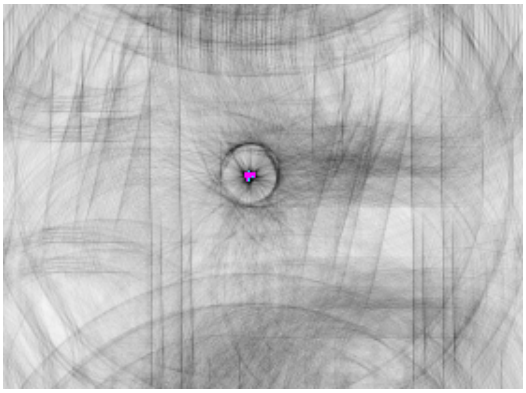
(b) Anwendung eines globalen Schwellwertverfahrens auf die Grauwerte des Eingabebildes



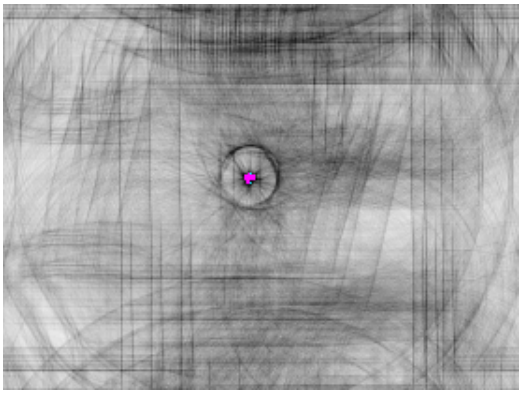
(c) Vorlage ohne eingezeichneten Arm bei r=71



(d) Vorlage mit eingezeichnetem Arm bei $\alpha = 0$ und $\beta = 0$



(e) Korrelationsbild von (a) und (c). Etwa in der Bildmitte ist ein Maximum zu erkennen. Dieses wird als Position des Radmittelpunkts angenommen.



(f) Korrelationsbild von (a) und (d). In der Nähe von oberem und unterem Bildrand sind waagerechte störende lokale Maxima, die größer als das Maximum im Radmittelpunkt werden können

Abbildung 3.1: Radmittelpunktbestimmung von Abbildung 1.1a (invertiert). Trotz der störenden Schrift rechts oben im Anfangsbild wird der Mittelpunkt zuverlässig gefunden.

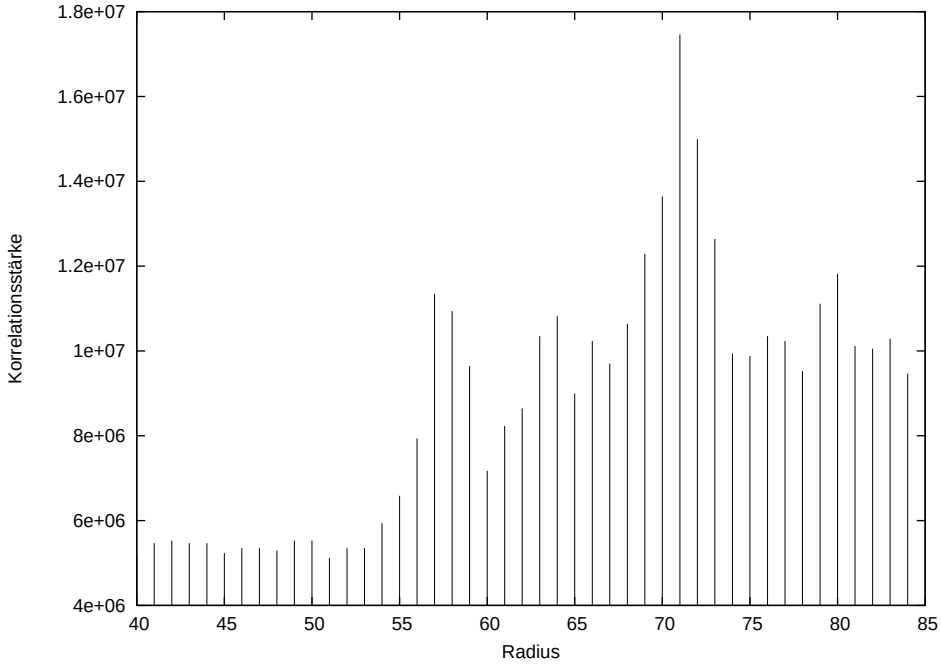


Abbildung 3.2: Die Korrelationsstärke für micro.adam aus Bild 1.1a in Abhängigkeit vom Radius der Vorlage. Bei r=71 ist das globale Maximum.

Um eventuelle Ausreißer auszufiltern wird die Korrelationsstärke der globalen Maxima der Korrelationsbilder über die verschiedenen Radien folgendermaßen geglättet:

$$K_r^{geglaettet} = 0.25\ K_{r-1} + 0.5\ K_r + 0.25\ K_{r+1} \tag{3.3}$$

In diesem geglätteten Korrelationsstärken wird das globale Maximum gesucht und der zugehörige Radius der Vorlage als Radius des Rades im Kamerabild angenommen.

Zwischen der Korrelationsstärke und der Anzahl der Pixel in der Vorlage - und damit dem Radradius in der Vorlage - besteht ein proportionaler Zusammenhang. Es erfolgt keine Korrektur dafür, da das Maximum bei einer Übereinstimmung in Vorlage und Kantensegmentbild immer noch ein globales Maximum ist.

Alternativ zur Korrelation ließen sich Kreise oder Ellipsen aus den Kantensegmentbildern extrahieren. Dies ermöglicht beispielsweise die Hough-Transformation unter Verwendung einer parametrischen Kreisgleichung (siehe Abschnitt 3.2.3). Es würde damit nach dem Außen- und Innenkreis des Rad-Kreisringes gesucht. Da das Verhältnis von Außen- und Innenradius skalierungsinvariant ist, hätte man hier eine gute Ausschlußbedingung.

Sind Radius und Radmittelpunkt bestimmt, betrachten nachfolgende Verarbeitungsschritte nur die im Inneren des Rades liegende Bildfläche, um die Arm- und Radstellung zu berechnen.

3.1.2 Bestimmung der Arm- und Radstellung bei micro.adam

Bei micro.adam werden mittels *Hough-Transformation* die im Radinneren verlaufenden Geraden bestimmt. Dazu wird die Hough-Transformation auf die Kantensegmentdarstellung des Eingabebildes an-

gewendet. Ausgabe der Hough-Transformation ist eine Liste von Geraden $g_i = (\theta_i, \rho_i)$ in Polardarstellung.

Aus diesen Geraden werden mittels Fehlerminimierung zwei Geraden g_{a_1}, g_{a_2} ausgewählt, die

- parallel zueinander stehen und
- in einem bestimmten, zum Radius proportionalen Abstand voneinander verlaufen - der Breite des Armes.

Diese beiden Geraden stellen mit hoher Wahrscheinlichkeit die beiden Außenkanten des Armes von micro.adam dar. Folgende Fehlerterme werden berechnet:

$$E_\theta(m,n) = (\theta_m - \theta_n)^2 \tag{3.4}$$
$$E_\rho(m,n) = (|\rho_m - \rho_n| - armbreite_r)^2 \tag{3.5}$$
$$E_{gesamt}(m,n) = 30\ E_\theta + 0.15^2\ E_\rho \tag{3.6}$$

Die Konstanten in der letzten Gleichung dienen der Gewichtung der Fehlerterme und sind experimentell ermittelt worden. Die ebenfalls experimentell bestimmten Ausschlußbedingungen, die alle Paare (m,n) erfüllen müssen, um nicht von vornherein ausgeschlossen zu werden, lauten:

$$E_\theta(m,n) \leq 0.04 \tag{3.7}$$
$$E_\rho(m,n) \leq 30 \tag{3.8}$$

Aus den verbleibenden Paaren wird das ausgesucht, für das E_{gesamt} minimal ist:

$$(a_1, a_2) = \underset{m,n}{\operatorname{argmin}} E_{gesamt}(m,n) \tag{3.9}$$

Der Arm kann nun als die zwischen den beiden gefundenen Geraden verlaufende Gerade $g_{Arm} = (\theta_{Arm}, \rho_{Arm})$ repräsentiert werden. Dabei ist

$$\theta_{Arm} = \theta_{a_1} \approx \theta_{a_2} \tag{3.10}$$
$$\rho_{Arm} = \frac{\rho_{a_1} + \rho_{a_2}}{2} \tag{3.11}$$

Ist ρ_{Arm} nach dieser Berechnung negativ, muß es normalisiert werden mit:

$$\text{normalize}(\theta_{Arm}, \rho_{Arm}) = (\theta_{Arm} + \pi, -\rho_{Arm}) \tag{3.12}$$

Danach werden die beiden Schnittpunkte s_1, s_2 zwischen g_{Arm} und dem den Radring repräsentierenden Kreis mit Radius r und Mittelpunkt M berechnet. Dazu wird folgende quadratische Gleichung gelöst, die aus der Gleichsetzung der parametrischen Gleichungen für eine Gerade nach (3.41) und einen Kreis

mit $x^2 + y^2 = r$ entsteht. Die Punkte werden im Koordinatensystem berechnet, das den Radmittelpunkt als Koordinatenursprung hat:

$$p = -2\ r\ \sin(\theta_{Arm}) \tag{3.13}$$
$$q = \rho_{Arm}^2 - r^2\ \sin(\theta_{Arm})\ \sin(\theta_{Arm}) \tag{3.14}$$
$$y_{1/2} = \frac{-p}{2} \pm \sqrt{\frac{p^2}{4} - q} \tag{3.15}$$
$$x_{1/2} = \frac{r}{\cos(\theta_{Arm})} - \frac{y_{1/2}}{\tan(\theta_{Arm})} \tag{3.16}$$
$$s_{1/2} = (x_{1/2}, y_{1/2}) \tag{3.17}$$

Einer dieser beiden Schnittpunkte stellt den Ansatzpunkt $s_{Ansatzpunkt}$ des Armes dar. Damit kann α als Winkel zwischen positiver x-Achse und $s_{Ansatzpunkt}$ mit dem Radmittelpunkt als Angelpunkt berechnet werden:

$$\alpha = \begin{cases} \arccos(x_{Ansatzpunkt}/r) & \text{falls} \quad y \geq 0 \\ 2\pi - \arccos(x_{Ansatzpunkt}/r) & \text{falls} \quad y < 0 \end{cases} \tag{3.18}$$

Zur Entscheidung, ob s_1 oder s_2 der richtige Ansatzpunkt ist, wird zuerst das α des letzten Frames herangezogen und mit den zu $s_{1/2}$ gehörenden $\alpha_{1/2}$ verglichen. Es wird angenommen, daß sich das Rad nur wenig gedreht hat. Also wird das α_i ausgewählt, für das die Winkeldifferenz zum im letzten Frame für α gefundenen Winkel $\alpha(t-1)$ minimal ist:

$$i_{Ansatzpunkt} = \underset{i}{\operatorname{argmin}} |\alpha_i - \alpha(t-1)|_{Winkel} \tag{3.19}$$

Voraussetzung hierbei ist, daß $\alpha(t-1)$ gültig ist. Um diese Voraussetzung für die meisten Frames zu erfüllen, wird die vom Floodfill-Algorithmus (siehe Abschnitt 3.2.4) gelieferte Position der LED p_{LED} herangezogen. Aus ihr wird der Winkel α_{LED} berechnet, der die Richtung der LED ausgehend von M relativ zur positiven x-Achse darstellt. α_{LED} wird ähnlich berechnet, wie α in Gleichung (3.18), nur daß $\arccos\left(x_{LED} / \left\|p_{LED} - M\right\|\right)$ berechnet wird. Die Idee ist, daß die LED dem Armansatzpunkt meist in etwa gegenüber liegt. Die entsprechende Bedingung lautet:

$$|\alpha_{i_{Ansatzpunkt}} + \pi - \alpha_{LED}|_{Winkel} \leq 30^\circ \tag{3.20}$$

Die Obergrenze von 30° ergibt sich aus der Überlegung, daß der Arm beweglich ist und sein Drehpunkt nicht dem Radmittelpunkt entspricht. So liegt α nur genau gegenüber von α_{LED}, wenn $\beta = 0$. Aus der Kenntnis von α, β, r und der Radgeometrie läßt sich bereits p_{LED} und eine exakte Obergrenze für obige Ungleichung berechnen. Letzteres war aber nicht erforderlich, da die Erkennung auch mit der direkt umgesetzten Idee gut funktioniert.

Ist diese Bedingung nicht erfüllt, wird α auf die andere Seite verlagert. Damit müßte sie im nächsten Frame erfüllt sein. Falls keine oder mehr als eine LED gefunden werden, wird α unverändert gelassen, da keine sinnvolle Entscheidung möglich ist. Blobs, die weniger als $r/2$ vom Mittelpunkt entfernt gefunden werden, werden nicht betrachtet, da sie keine LED darstellen können.

Eine Mittelung der Erfüllung der Verlagerungsbedingung über mehrere Frames könnte sich vorteilhaft auswirken, um die Verlagerung störungsunempfindlicher zu machen.

Ohne die LED einzubeziehen wäre die Ansatzpunktunterscheidung ebenso möglich:

- Einmal mittels der Querlinie des Armes am „LED-Ende". Sie könnte relativ leicht aus der Menge der gefundenen Linien heraussortiert werden, denn ihre Ausrichtung und Länge sind bekannt. Allerdings ist sie zu kurz, um zuverlässig mittels Hough-Transformation gefunden zu werden.
- Eine weitere Alternative wäre, die Kantenpixel der Armaußenkanten im Armansatzbereich zu zählen. Dort, wo Pixel sind, wäre der Ansatzpunkt. Hier liegt die Schwierigkeit darin, möglichst nur die Pixel der Armaußenkanten zu zählen und nicht etwa zum Kreisring gehörige. Trotzdem wird diese Verfahrensweise wahrscheinlich einen geringeren Rechenaufwand bedeuten als das momentan durchgeführte Floodfill für die gesamte Radinnenfläche.

Die Berechnung von β ist der letzte Schritt. Es ist die Differenz aus α und der Ausrichtung θ_{Arm} der den Arm repräsentierenden Gerade g_{Arm}. Das einzige Problem ist eine aufwendige Fallunterscheidung, da die Ausrichtung der Geraden durch einen Winkel aus dem Intervall $[0, \pi)$ gegeben ist.

$$\beta = \begin{cases} \alpha - \theta_{Arm} & \text{falls} \quad \alpha < \frac{\pi}{2}, & \theta_{Arm} < \frac{\pi}{2} \\ \alpha - \theta_{Arm} + \pi & \text{falls} \quad \alpha < \frac{\pi}{2}, & \theta_{Arm} \geq \frac{\pi}{2} \\ \alpha - \theta_{Arm} - \pi & \text{falls} \quad \frac{\pi}{2} \leq \alpha < \pi, & \theta_{Arm} < \frac{\pi}{2} \\ \alpha - \theta_{Arm} & \text{falls} \quad \frac{\pi}{2} \leq \alpha < \pi, & \theta_{Arm} \geq \frac{\pi}{2} \\ \alpha - \theta_{Arm} - \pi & \text{falls} \quad \pi \leq \alpha < \frac{3}{2}\pi, & \theta_{Arm} < \frac{\pi}{2} \\ \alpha - \theta_{Arm} & \text{falls} \quad \pi \leq \alpha < \frac{3}{2}\pi, & \theta_{Arm} \geq \frac{\pi}{2} \\ \alpha - \theta_{Arm} - \pi & \text{falls} \quad \frac{3}{2}\pi \leq \alpha < 2\pi & \end{cases} \tag{3.21}$$

Eventuell muß β noch normalisiert werden:

$$\beta_{Normalisiert} = \begin{cases} \beta + \pi & \text{falls} \quad \beta < -\frac{\pi}{2} \\ \beta - \pi & \text{falls} \quad \beta > \frac{\pi}{2} \\ \beta & \text{sonst} \end{cases} \tag{3.22}$$

Schon allein aus Kenntnis von LED-Position, Radradius und -mittelpunkt sind α und β berechenbar. Der Arm hat wie die Arme von micro.eva nur einen Freiheitsgrad in der Bewegung relativ zu seinem Ansatzpunkt. Da die Armlänge im Bild durch den Radradius bekannt ist, kann also die Position des Ansatzpunktes berechnet werden. Die Gerade, die durch LED-Position und Armansatzpunkt verläuft, ist nun wieder g_{Arm}. Aus ihr können auf bekannte Weise α und β berechnet werden.

3.1.2.1 Alternativer Weg mittels Polartransformation

Alternativ zur Verwendung der Hough-Transformation wurde die Ermittelung von α durch Transformation des Radinneren in eine polare Entwicklung um den Radmittelpunkt und nachfolgende Korrelation versucht.

Bei der Transformation eines Bildes in Polarkoordinaten wird es um den Mittelpunkt herum „aufgerollt". Die Polartransformation eines in kartesischen Koordinaten gegebenen Bildes $B = (x_K, y_K),\ x \in \{0, 1, ..., X\},\ y_K \in \{0, 1, ..., Y\}$ erhält man durch

$$x_K = \rho \frac{X}{2} \cos(\theta) \tag{3.23}$$

$$y_K = \rho \frac{Y}{2} \sin(\theta) \tag{3.24}$$

mit

$$\rho = \frac{r}{R} \tag{3.25}$$

$$\theta = 2\pi \frac{t}{T} \tag{3.26}$$

Ausgabe ist ein Bild $P = (t, r),\ t \in \{0, 1, ..., T-1\},\ r \in \{0, 1, ..., R-1\}$. Hierbei erfolgte die Entwicklung um den Mittelpunkt von B.

In einem solchen Bild bewirkt die Raddrehung eine Verschiebung. Die Suche nach dem Armansatzpunkt könnte auf einen rechteckigen Bereich beschränkt werden. In Abbildung 3.3b und c handelt es sich dabei um den unteren Bildrand.

Als Voraussetzung für eine brauchbare Ausgabe der Polartransformation muß der Mittelpunkt der Entwicklung (also der Radmittelpunkt) exakt bekannt sein. Ist das nicht der Fall, ist das Ergebnis vergleichbar Abbildung 3.4.

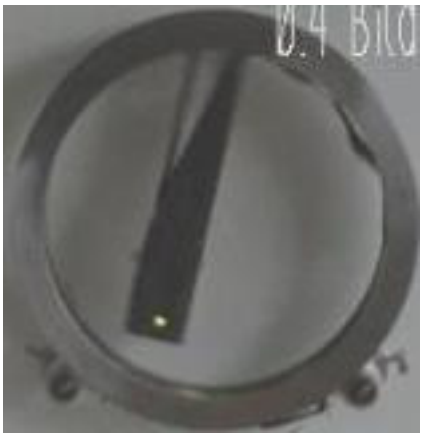
Die Suche nach dem Armansatzpunkt und damit nach α ließe sich wiederum mittels Bildkorrelation von Kantensegmentdarstellungen durchführen. Ein mögliches Eingabebild und eine Vorlage sind in Abbildung 3.5 zu sehen. Das durch die Vorlage realisierte „T-Stück" soll auf die Kanten im Bereich des Armansatzpunktes passen.

Für die Korrelation wäre an dieser Stelle besser, keine Kantensegementdarstellung zu verwenden, sondern ein Bild, das in Vorder- und Hintergrund getrennt wurde. Dann wäre das genaue Aussehen der Vorlage weniger bedeutend. Diese Trennung gestaltet sich aber aufgrund des schlechten Kontrasts im Bild schwierig. Der schlechte Kontrast war schon bei der Suche nach Radmittelpunkt und -radius in Abschnitt 3.1.1 der Grund gewesen, die Korrelation anhand der Kantensegmentdarstellung vorzunehmen.

Dieser Weg wird im Folgenden nicht mehr verfolgt. Er scheint weniger effizient als die Hough-Transformation zu sein. Ein zweiter Nachteil ist, daß zur Feststellung des Parameters β zusätzliche Berechnungen angestellt werden müßten.

3.1.3 Bestimmung der Armstellungen bei micro.eva

Bei micro.eva werden mittels *Floodfill*-Algorithmus die wahrscheinlichsten Positionen der LEDs $p_i,\ i \in [0, 1, .., i_{max}]$ ermittelt. Voraussetzung für die Brauchbarkeit des Ergebnisses ist, daß im Bild des Radinneren diese Farben nicht außerhalb von LED-Abbildern vorkommen. Die Anzahl der zurückgelieferten Positionen i_{max} wird zunächst unabhängig von der Anzahl der im Bild enthaltenen LEDs betrachtet. Erst die Erfüllung verschiedener Kriterien läßt mehr und mehr den Schluß zu, daß es sich um die gesuchten LEDs handelt.



(a) Bildausschnitt mit dem Innenbereich von micro.adam

(b) Nach der Polartransformation

(c) Transformation des weitergedrehten Rades

Abbildung 3.3: Transformation des Radinneren in eine Polardarstellung

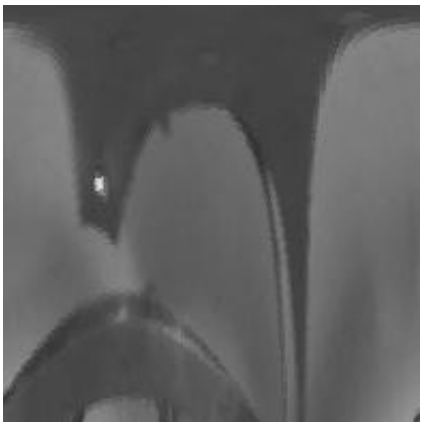


Abbildung 3.4: Polartransformation mit einem leicht verschobenen Mittelpunkt

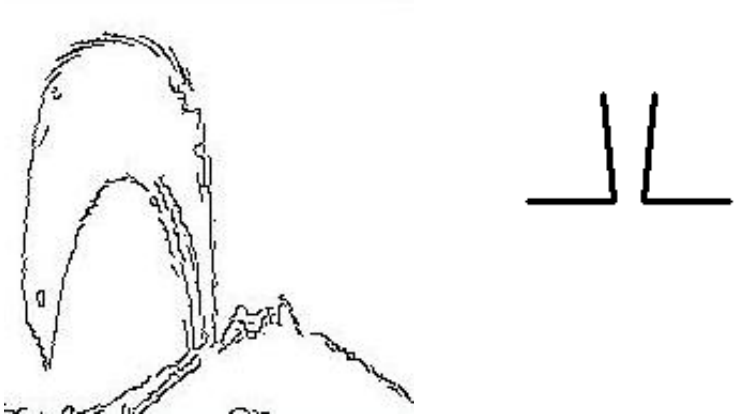


Abbildung 3.5: Eingabebild und Vorlage für die Bildkorrelation zur Ermittelung von α

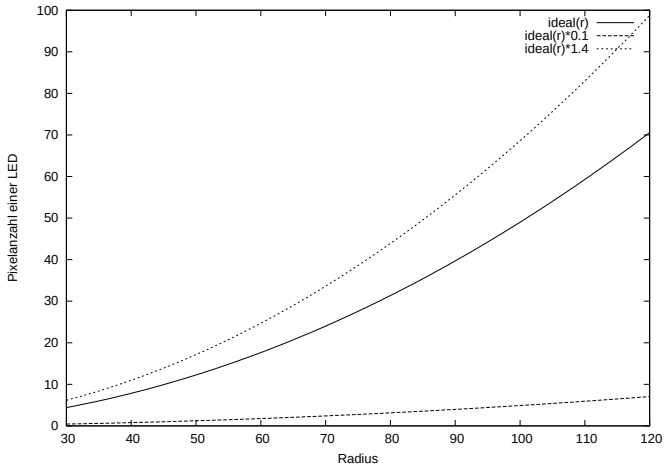


Abbildung 3.6: Minimale, ideale und maximale Pixelanzahl einer LED in Abhängigkeit vom Radradius

Die p_i werden gesiebt: Sie müssen im Radinneren liegen ($d_i = \|p_i - M\| < r$) und sollten aus einer passenden Anzahl von Pixeln bestehen. Die ideale erwartete Pixelzahl wird abhängig vom Radradius berechnet. Aus ihr werden minmale und maximale Grenzen für die Anzahl der beitragenden Pixel der gefundenen Schwerpunkte abgeleitet:

$$pix_{ideal} = (0.07\ r)^2 \tag{3.27}$$

$$pix_{minimal} = 0.1\ pix_{ideal} \tag{3.28}$$

$$pix_{maximal} = 1.4\ pix_{ideal} \tag{3.29}$$

Die verwendeten Konstanten wurden experimentell bestimmt. Insbesondere wird mit dieser Maßnahme erreicht, daß bei hoher Auflösung keine einzelnen Pixel als LED angesehen werden.

Zu jedem Schwerpunkt wird der Winkel γ_i berechnet, der den Winkel des Schwerpunktes p_i relativ zum

Angelpunkt Radmitte M und positiver x-Achse angibt:

$$\gamma_i = \begin{cases} \arccos(x_{p_i}/d_i) & \text{falls} \quad y_{p_i} \geq 0 \\ 2\pi - \arccos(x_{p_i}/d_i) & \text{falls} \quad y_{p_i} < 0 \end{cases} \tag{3.30}$$

Da ein Arm nur einen Freiheitsgrad hat, stellt der Abstand d_i ein Maß für den Zustand eines Armes dar, dessen LED an Position p_i im Bild ist. Ist $d_i = d_{min}$ minimal, ist die LED nahe dem Radmittelpunkt, der Arm also „ausgefahren". Ist $d_i = d_{max}$ maximal, ist der Arm weit vom Radmittelpunkt entfernt, liegt also „am Radring an". Nach Konstruktion der Arme gilt $0 < d_i < r$. $\frac{d_i}{r}$ ist nun ein skalierungsinvariantes Maß, welches die Armstellung repräsentieren kann. Wird der Schwerpunkt i im Folgenden einem Arm j zugeordnet, wird das auf [0,1] normierte, invertierte Maß als β_j übernommen:

$$\beta_j = 1 - \frac{d_i - d_{min}}{d_{max} - d_{min}} \tag{3.31}$$

Die fünf Arme von micro.eva sind im gleichmäßigen festen Abstand von 72° montiert ($\frac{360°}{5}$). Diese Eigenschaft wird im Folgenden genutzt, um aus den p_i die herauszusuchen, die mit LEDs von micro.eva korrespondieren. Dazu werden die Winkel der Arme berechnet (wieder relativ zum Angelpunkt M), wenn sie sich in Nullstellung befinden, also $d_i = d_{max}$:

$$\Gamma_i = \gamma_i + \frac{20}{180}\pi \ \beta_i \tag{3.32}$$

Es wurde beobachtet, daß sich der Winkel der LED-Position beim „Hineinfahren" des Armes zur Radmitte um einige Grad entgegen dem Uhrzeigersinn verschiebt. Bei Annahme eines linearen Zusammenhangs und einer Schätzung der Verschiebung von 20° kommt obige Formel zustande. Daß diese Annahmen nicht zutreffen, läßt sich leicht aus Abbildung 3.7 ersehen. Offenbar besteht ein nichtlinearer Zusammenhang, welcher durch genauere Modellierung auf der Grundlage der Konstruktion von micro.eva nachvollziehbar wäre. Ebenso ist eine Messung der exakten Verschiebung möglich. Trotz dieser Ungenauigkeiten funktioniert die Erkennung gut.

Anhand der Γ_i werden nun per Fehlerminimierung die maximal 5 Indizes $i_{Arm_0}, i_{Arm_1}, ..., i_{Arm_4}$ herausgesucht, die am Besten im 72°-Abstand voneinander angeordnet sind. Jedes i wird einmal als Ausgangspunkt der Suche verwendet, also als i_{Arm_0} verwendet. Dann wird links davon nach i_{Arm_1} gesucht. Es muß gelten, daß $|\Gamma_{i_{Arm_0}} + 72° - \Gamma_{i_{Arm_1}}|$ minimal ist. Die Suche wird bis zum fünften Arm fortgesetzt, wobei also das allgemeine Fehlermaß lautet:

$$E_{Arm}(k) = |\Gamma_{i_{Arm_k}} + 72° - \Gamma_{i_{Arm_{k+1}}}|, \ k \in 0, 1, 2, 3 \tag{3.33}$$

Die Menge der i_{Arm_j}minimiert also den Fehler:

$$E_{Arme} = \sum_{k=0}^{3} E_{Arm}(k) \tag{3.34}$$

Ist $E_{Arm}(k) > E_{Arm}^{Max}$ (derzeit etwa 50°), wird angenommen, daß für diese Armposition kein Schwerpunkt gefunden wurde und es wird ein Strafwert zum Gesamtfehler addiert.

Abbildung 3.7: LED-Suche bei micro.eva. Blau eingekreist sind die LED-Positionen p_i. Aus ihnen wurden die Γ_i berechnet, die auf dem Radaußenkreis eingezeichnet wurden. Man kann hier sehen, daß zwischen $\Gamma_{i_{Arm_4}}$ und $\Gamma_{i_{Arm_5}}$ ein Abstand von mehr als 72° besteht.

Die Leuchtdioden von micro.eva sind ideale Features, denn sie sind mit geringem Aufwand zu erkennen. Ohne sie müßten andere Features erkannt werden, die schwerer wahrnehmbar sind, so daß der Aufwand für die Erkennung zunehmen würde. Eine Möglichkeit bieten die „Kreisringstücke" der Arme (die Teile auf denen die Leuchtdioden montiert sind, ohne die Streben). Dies würde einen translations- und rotationsinvarianten Erkennungsalgorithmus erfordern. Dies ist z.B. die starre Registrierung (rigid image registration), basierend auf der Fast-Fourier-Mellin-Transformation [10, S. 155 ff]. Diese wäre sogar skalierungsinvariant. Sie kann aber nicht auf das Gesamtrad angewendet werden, da es nicht starr ist. Außerdem ist sie aufgrund ihrer Komplexität, die mindestens das doppelte der FFT beträgt, weniger effizient als Floodfill berechenbar.

Eventuell ließe sich auch Bildkorrelation zur Suche nach den LEDs einsetzen.

3.1.4 Bestimmung der Radstellung bei micro.eva

Ergebnis der Bestimmung der Armstellungen waren die Indizes der bis zu fünf Schwerpunkte i_{Arm_j}. Je nachdem, ob $\alpha(t-1)$ bekannt ist, wird nun verschieden verfahren.

Ist $\alpha(t-1)$ unbekannt, wird $\alpha = \Gamma_{i_{Arm_0}}$ gesetzt, also i_{Arm_0} als erster Arm festgelegt. Er muß dann auch in den folgenden Frames wieder als erster Arm behandelt werden. Danach werden die zu den entsprechenden i_{Arm_j} gehörigen $\beta_{i_{Arm_j}}$ als β_0 bis β_4 übernommen.

Ist $\alpha(t-1)$ bekannt, muß nun eine Zuordnung der i_{Arm_j} zu den fünf Armen erfolgen. Es wird analog zum Verfahren bei micro.adam angenommen, daß sich das Rad nur wenig bewegt hat. Es wird das i_{Arm_j} herausgesucht, für das gilt:

$$\Gamma_{i_{Arm_j}} \approx \alpha(t-1) \tag{3.35}$$

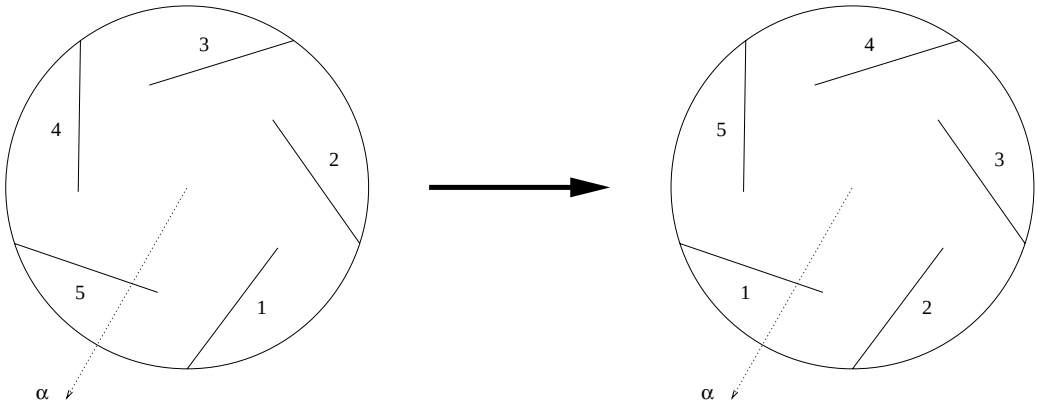


Abbildung 3.8: Rotation der Armnumerierung um Zuge der Bestimmung von α. Hier erfolgt eine Rotation um einen Schritt im Uhrzeigersinn.

und als $\alpha(t)$ gesetzt. Ausgehend von diesem ersten Arm erfolgt nun eine Zuordnung der restlichen Armpositionen und die Übernahme der entsprechenden β_i. Diese Zuordnung entspricht einer Rotation der Numerierung, da die i_{Arm_j} schon in einer fortlaufenden Reihenfolge angeordnet sind und nur der „Startarm" neu gesetzt wird.

3.2 Erkennungsalgorithmen

3.2.1 Canny-Algorithmus (Kantensegmenterkennung)

Der Canny-Algorithmus [4] [37, Kap. 3-12 ff] findet Kanten an den Stellen im Eingabebild, an denen ein Sprung der Bildhelligkeit zwischen benachbarten Pixeln auftritt. Im Normalfall heben sich gesuchte Objekte vom Hintergrund ab, so daß mit diesem Algorithmus vor allem die Objektumrisse gefunden werden.

Im Folgenden wird der Canny-Algorithmus kurz skizziert.

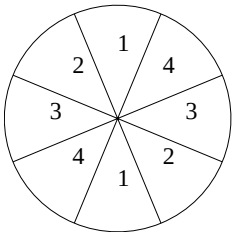


Abbildung 3.9: Quantisierung von Gradientenrichtungen

Algorithmus 1: Canny-Algorithmus

1. Bildglättung: Das Eingabebild wird mittels einer Gaußfunktion geglättet, um Rauschen auszufiltern. Je größer die Standardabweichung für den Gauß-Kern gewählt wird, desto geringer ist die Empfindlichkeit gegenüber Rauschen. Jedoch gehen dabei Bilddetails verloren. Auch steigt der Lokalisierungsfehler, d.h. die detektierten Kanten können gegenüber den wirklichen Kanten verschoben sein.

2. Differentiation: Das geglättete Bild wird nach x und y differenziert. Dies ist ebenfalls durch eine diskrete Faltung realisierbar. Aus den Gradienten in x- und y-Richtung werden Betrag und Richtung des Gradienten in jedem Bildpunkt berechnet.

3. Nicht-Maxima-Unterdrückung: An den Stellen, an denen der Betrag der Gradienten ein lokales Maximum aufweist, können Kanten zu finden sein. Es werden alle Nicht-Maxima unterdrückt, also alle Punkte an denen der Gradientenbetrag kein lokales Maximum aufweist. Jedoch müssen nur die Nicht-Maxima senkrecht zur Kantenrichtung unterdrückt werden, denn die Kante verläuft mit gewisser Stärke entlang einer Kontur. Die Gradientenrichtungen werden entsprechend der Intervalle in Abbildung 3.9 in 4 Richtungen quantisiert. Danach wird für jeden Bildpunkt entsprechend des Richtungsintervalles der Betrag der Gradienten der Nachbarpixel in der 3×3-Umgebung geprüft. Ist er nicht größer als der Betrag seiner zwei geprüften Nachbarn, wird er unterdrückt.

4. Anwendung eines Hysteresis-Schwellwertverfahrens auf die Kantenpixel: Nachteil eines Schwellwertverfahrens mit nur einem globalen Schwellwert ist, daß Fluktuationen auftreten können, was zu unschönen unterbrochenen Linien führen kann. Bei einem Hysteresis-Schwellwertverfahren existieren zwei Schwellwerte - eine obere und eine untere Grenze. Liegt der Gradient eines Kantensegments über der oberen Grenze, wird er sofort akzeptiert. Analog, wird er sofort gelöscht, wenn er unter der unteren Grenze liegt. Liegt ein Gradient zwischen diesen beiden Grenzen, wird er akzeptiert, wenn er mit Pixeln benachbart ist, die schon als Kantenpixel akzeptiert wurden. Die Wahrscheinlichkeit von unterbrochenen Linien wird dadurch drastisch reduziert, denn die Beträge der Gradienten müssen über die obere und unter die untere Grenze fluktuieren, damit Flackern auftreten kann. J. Canny schlägt in [4] vor, daß das Verhältnis von oberer zu unterer Grenze im Bereich zwei oder drei zu eins liegt, je nach vermutetem Signal-Rausch-Abstand.

Durch Einsatz der Nicht-Maxima-Unterdrückung und des Hysteresis-Schwellwertverfahrens auf die Gradienten ist dieses Verfahren lokal bzw. adaptiv. Ausschlaggebend dafür, ob ein Kantensegment erkannt wird, ist der Grauwertanstieg zwischen benachbarten Pixeln und nicht der absolute Grauwert.

Im Programm können die Hysteresis-Schwellwerte angepaßt werden. Der größere wird verwendet, um Anfangssegmente von Kanten zu finden. Der kleinere, um Kantensegmente zu verbinden. Er kann verringert werden, falls unterbrochene Kanten auftauchen. Eine Glättung wird aus Geschwindigkeitsgründen nicht durchgeführt.

3.2.2 Bildkorrelation im Frequenzraum

Wird ein unbekanntes Bild mit einem bekannten Bild als Vorlage korreliert, so können im berechneten Korrelationsbild durch Maximumsuche die Stellen gefunden werden, an der die Vorlage im unbekannten Bild wiederzufinden sein müßte.

Die direkt durchgeführte Bildkorrelation im Ortsraum ist eine sehr aufwendige Operation. Doch sie läßt sich auch mittels Fourier-Transformation ausdrücken. Für diese sind schnelle Berechnungsverfahren bekannt.

3.2.2.1 Bildkorrelation im Ortsraum

Eingabe für die Bildkorrelation bilden ein Graustufenbild B und ein Vorlagebild V (ebenfalls ein Graustufenbild). Die Vorlage hat dabei eine Größe von $J \times L$ Pixeln.

Bei einer Verschiebung der Vorlage V um (m, n) Pixel gegenüber B ist die Korrelationsstärke $K(m, n)$ definierbar als

$$K(m,n) = \sum_{j=0}^{J-1} \sum_{l=0}^{L-1} V(j,l)\ B(m+j, n+l) \tag{3.36}$$

Es werden also die Helligkeiten der übereinanderliegenden Punkte von B und V miteinander multipliziert und aufsummiert.

Wird nun die Korrelationsstärke für alle möglichen (sinnvollen) Verschiebungen (m, n) berechnet, ergibt sich das Korrelationsbild K.

Abbildung 3.1 auf Seite 23 verdeutlichte die Bildkorrelation an konkreten Bildern.

Die Komplexität einer einfachen Implementierung der Bildkorrelation im Ortsraum nach (3.36) ist sehr hoch. Für einen quadratischen Kern K×K angewendet auf ein Bild der Größe N×N ist sie $O(K^2)$, wobei die Komplexität pro Pixel auf Grundlage der Anzahl der Multiplikationen und Additionen bemessen wurde.

3.2.2.2 Korrelation im Frequenzraum

Nach dem Korrelationstheorem [13, Kap. 3] gilt

$$f(x,y) \circ g(x,y) \Leftrightarrow F^*(u,v)G(u,v) \tag{3.37}$$

D.h. die Korrelation im Ortsbereich $f \circ g$ entspricht einer *Multiplikation* im Frequenzbereich, also einer komplexen Multiplikation pro Pixel. F^* steht dabei für das konjugiert-komplexe der Fourier-Transformierten von f. Zusammen mit der schnellen Fourier-Transformation läßt sich dieser Umstand zur Konstruktion einer schnellen diskreten Bildkorrelation nutzen.

Die *diskrete Fourier-Transformation* [30] ist definiert als:

$$\mathfrak{F}(\omega) = C \sum_{k=0}^{K-1} f(kT)\ e^{-i\omega kT} \tag{3.38}$$

C ist eine Konstante und T der Abstand zweier aufeinanderfolgender Zeitpunkte/Pixel.

Ausgedehnt auf den zweidimensionalen Fall haben wir:

$$\mathfrak{F}(\omega_x, \omega_y) = C \sum_{j=0}^{J-1} \sum_{k=0}^{K-1} f(kT_x, jT_y)\ e^{-i(\omega_x kT_x + \omega_y jT_y)} \tag{3.39}$$

Zumeist wird $C = \frac{1}{JK}$ gewählt, da dann die Rücktransformation die Ausgangswerte wieder ergibt.

Der Algorithmus der schnellen diskreten Bildkorrelation im Frequenzbereich lautet nun:

Algorithmus 2: Bildkorrelation im Frequenzbereich

1. Anwendung der 2d-DFT auf Eingangsbild und Vorlage
2. elementweise Multiplikation der konjugiert-komplexen Fourier-Transformierten vom Eingangsbild und der Fourier-Transformierten der Vorlage
3. Rücktransformation des Ergebnisses

Die Korrelation im Frequenzbereich unterscheidet sich von der Korrelation im Ortsbereich in dem Punkt, daß die Funktionen jeweils als periodisch fortgesetzt betrachtet werden.[1]

Die Komplexität dieses Ansatzes pro Pixel für ein Eingangsbild der Größe N×N und einen Kern der Größe K×K ist $O(\log N)$ komplexe Multiplikationen und Additionen - unabhängig von K. Es wird vorausgesetzt, daß N eine bestimmte Struktur hat (z.B. daß es eine Potenz von 2 oder Produkt möglichst kleiner Primzahlen ist), so daß schnelle FFT-Algorithmen wie Radix2-FFT anwendbar sind. Ebenso sollte $K^2 \gg logN$ sein, damit die Korrelation im Frequenzbereich wirklich schneller als die direkte Implementation von (3.36) ist.

3.2.2.3 Berechnung der Vorlage

Für die Lokalisierung des Radmittelpunktes ist es relevant, daß die Maxima im Korrelationsbild an der richtigen Stelle auftauchen. Da die Größe des Vorlagebildes gleich der Größe des Kamerabildes ist, müssen die Inhalte des Vorlagebildes an eine bestimmte Stelle gezeichnet werden.

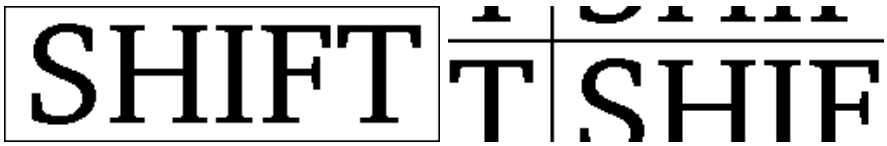
Sie wird so gezeichnet, daß als Mittelpunkt von Radinnen- und Außenkreis die Stelle $(\frac{x_{Res}}{2}, \frac{y_{Res}}{2})$ verwendet wird. Ohne eine Verschiebung der Vorlage würden alle Punkte im Korrelationsbild jeweils um $(-\frac{x_{Res}}{2}, -\frac{y_{Res}}{2})$ Pixel verschoben auftauchen.

Deshalb erfolgt nun eine *zirkuläre Verschiebung*. Die zirkuläre Verschiebung einer Funktion $f(x)$ diskreter Variablen um $d_x \in \mathbb{Z}$, wobei $0 \leq x < X$, ist definiert als:

$$g(x) = f\left((x + d_x) \mod X\right) \tag{3.40}$$

Für den zweidimensionalen Fall bedeutet das, daß alles, was an einem Rand aus dem Bild geschoben wird, auf der gegenüberliegenden Seite wieder auftaucht (siehe Abbildung 3.10).

[1] Deshalb heißt die zur Korrelation ähnlich definierte Faltung im Frequenzbereich auch zirkuläre Faltung [18].



(a) Bild mit X=251, Y=78 (b) Verschiebung um $d_x = 60$, $d_y = 20$

Abbildung 3.10: Zirkuläre Verschiebung

Für unsere Zwecke muß also das Zentrum des in die Vorlage gezeichneten Objektes an die Stelle $(0, 0)$ verschoben werden. Dabei darf natürlich durch die Verschiebung kein Teil der Vorlage wegfallen. Dies wird durch die zirkuläre Verschiebung um $(\frac{x_{Res}}{2}, \frac{y_{Res}}{2})$ bzw. äquivalent $(-\frac{x_{Res}}{2}, -\frac{y_{Res}}{2})$ sichergestellt. Und wirklich ist die schnelle Faltung im Frequenzraum eine zirkuläre Faltung, d.h. der Kern wird an seinen Rändern periodisch fortgesetzt. Somit wurde also das Zentrum des Objektes in der Vorlage ohne Verluste an die Stelle $(0, 0)$ verschoben.

Die Verschiebung ist optional, denn an dieser Stelle bietet sich eine Beschleunigungsmöglichkeit an. Bei Einsparung der zirkulären Verschiebung müssen nur die gefundenen Maximumpositionen nachträglich umgerechnet werden.

3.2.3 Hough-Transformation (Geraden-Extraktion)

Eingabe der Hough-Transformation [8] ist ein Binärbild der Kanten, Ausgabe sind die wahrscheinlich im Bild enthaltenen Geraden.

Die Hough-Transformation ist eine Technik, mit der sich Bildmerkmale bestimmter Formen aus Bildern isolieren lassen. Wichtige Eigenschaften dabei sind, daß die Erkennung gegenüber Leerstellen in der Umrißdarstellung von Objekten tolerant ist, daß sie auch bei verrauschten Daten verwertbare Ergebnisse liefern kann und daß die Anzahl der im Bild enthaltenen Merkmale nicht im Voraus bekannt sein muß. Können die Bildmerkmale in Gleichungsform angegeben werden, läßt sich die klassische Hough-Transformation anwenden. Sie ist vor allem zur Erkennung regelmäßiger Formen geeignet, die mit wenigen Parametern spezifiziert werden können, wie Geraden, Kreise und Ellipsen. Ist eine Angabe der Form als Gleichung nicht möglich, kann die verallgemeinerte Hough-Transformation [7] eingesetzt werden.

Dahinter steckt die Idee, daß jedes Eingabepixel zu einer global konsistenten Lösung (z.B. einer Geraden) beiträgt.

Die Erkennung von Geraden mittels Hough-Transformation funktioniert nun wie folgt:

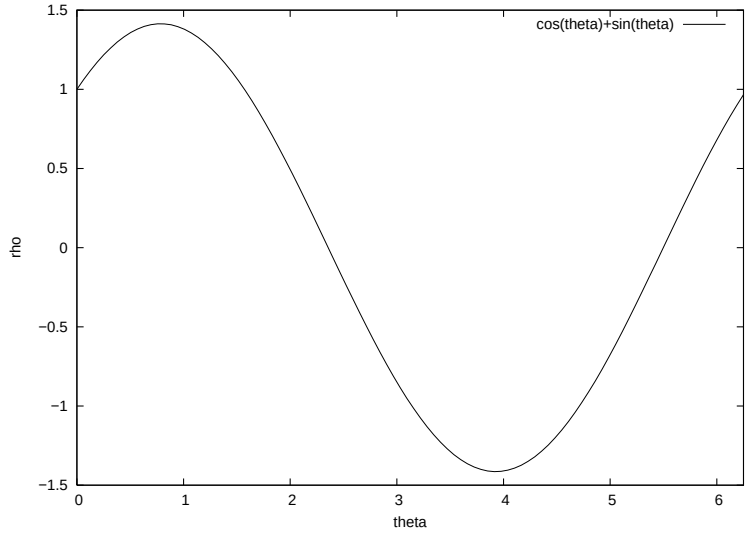


Abbildung 3.11: Eine zu einem Pixel an Position $(1, 1)$ gehörige Sinuskurve. Der Definitionsbereich von θ ist auf das Intervall $[0, 2\pi)$ beschränkt.

Algorithmus 3: Hough-Transformation

Die Beschreibung einer Geraden ist u.a möglich in Normalendarstellung:

$$\rho = x \ \cos\theta + y \ \sin\theta \tag{3.41}$$

Bei der Bildanalyse sind x und y bekannt, denn sie sind durch die gesetzten Pixel im zu analysierenden binären Bild gegeben. ρ und θ sind die gesuchten Variablen. Lösung dieser Gleichung ist eine eindimensionale Lösungsmenge. Ein Graph dieser Lösungsmenge würde eine Sinuskurve ergeben (siehe Abbildung 3.11). Punkte im kartesischen Eingaberaum des zu analysierenden Bildes werden also zu Sinuskurven im polaren Parameterraum transformiert.
Führt man diesen Transformationsschritt für alle gesetzten Pixel des Eingabebildes durch, ergibt sich ein Überlagerungsbild ähnlich Abbildung 3.12. Betrachtet man nun den Parameterraum, läßt sich gut erkennen, welche Punkte kollinear - d.h. auf einer Geraden - liegen: Sie werden auf Kurven abgebildet, die sich in einem gemeinsamen Punkt schneiden.

Die Transformation wird implementiert, indem der Parameterraum in endliche Intervalle quantisiert wird. Diese Intervalle werden auch Akkumulatorzellen genannt. Für jede Lösungsmenge von Gleichung (3.41) werden die entsprechenden Akkumulatorzellen erhöht. Zum Schluß werden die maximalen Akkumulatorzellen bestimmt. Sie repräsentieren Geraden, die mit hoher Wahrscheinlichkeit im Bild wiederzufinden sind.

Die Anzahl der Akkumulatorzellen bestimmt die erreichbare Auflösung. Stehen zu wenige Zellen zur Verfügung, kann es passieren, daß nahe beieinander verlaufende Geraden zur Akkumulation in denselben Zellen führen. Je höher die Anzahl der Akkumulatorzellen ist, desto besser können beieinanderliegende Geraden aufgelöst werden. Aber es erhöht sich auch der Erkennungsaufwand, denn es müssen mehr Zellen verwaltet werden.

Bei der Hough-Transformation unter Verwendung anderer Darstellungen erhöht sich meistens die Komplexität der Berechnung. Denn schon bei der Hough-Transformation für Kreise ist der Parameterraum

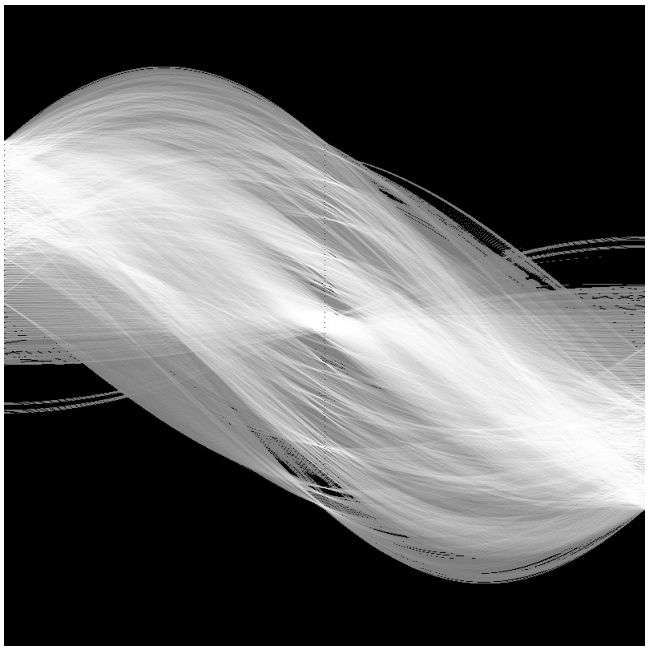


Abbildung 3.12: Der Akkumulator nach einer Hough-Transformation. Zu erkennen sind die sich überlagernden Sinuskurven und die von ihnen gebildeten Maxima (Bild mit freundlicher Genehmigung von Prof. Robert B. Fisher [9])

dreidimensional. Variieren können dabei die zwei Koordinaten des Kreismittelpunkts x_M, y_M, sowie der Kreisradius r:

$$r = \sqrt{(x - x_M)^2 + (y - y_M)^2}$$

3.2.4 Floodfill (Farb-Blob-Extraktion)

Die Funktion von Floodfill ist das Labeling verbundener Komponenten.

Eingabe für diesen Algorithmus ist ein farbiges Bild sowie der Ausschnitt des HSV-Farbraums, der als Objektfarbe gewertet werden soll. Ausgegeben werden 0 bis j „Schwerpunkte" zusammenhängender Objektpixel mit ihren Positionen (x,y) und der Anzahl an Pixeln n, aus denen sie gebildet werden.

Es wird ein *globales Schwellwertverfahren* verwendet, also eine Funktion, die einer gegebenen Farbe entweder die Eigenschaft „Objektfarbe" oder die Eigenschaft „Hintergrund" zuweist. Sie wird durch jeweils maximale und minimale Parameter im HSV-Farbraum bestimmt. Damit können z.B. rote oder grüne LEDs gefunden werden.

Das *HSV-Farbmodell* ist eine Alternative zum in der Computerwelt gebräuchlichen RGB-Farbmodell (Rot-Grün-Blau). Seine Parameter H (Hue, Farbton), S (Saturation, Sättigung) und V (Value, Helligkeit) sind an der menschlichen Farbwahrnehmung angelehnt. Somit ist es relativ leicht, bestimmte Farbtöne zu isolieren, weil der Farbton hauptsächlich durch den Parameter H repräsentiert wird. S und V können bei der Suche nach Pixeln eines bestimmten Farbtons fast ignoriert werden.

Von Floodfill existieren zwei praktisch relevante Varianten [17]. Die stackbasierte Variante erfordert nur einen Durchgang durch das Bild. Dabei wird das Bild gescannt. Tritt ein Objektpixel auf, werden alle seine

1	2	3
4	5	6
7	8	9

Abbildung 3.13: Numerierte Nachbarpixel des Pixels an Position 5, um Nachbarschaftsbeziehungen besser beschreiben zu können.

Umgebungspixel auf einem Stack abgelegt und das Pixel markiert. Dann wird der Stack abgearbeitet. Tritt ein noch unmarkiertes Objektpixel auf, wird es mit derselben Markierung versehen wie das erste Pixel und seine Umgebung wird auf dem Stack abgelegt.

Die Scanline-Variante von Floodfill erfordert zwei Durchgänge. Im ersten wird das Bild von links oben nach rechts unten gescannt. Tritt ein nicht markiertes Objektpixel auf, wird es abhängig von seinen rechten unteren Nachbarn markiert (Position 6, 7, 8, 9). Tragen die Nachbarn keine Marke, wird eine neue vergeben. Tragen sie maximal eine Marke, wird damit auch das aktuelle Pixel markiert. Tragen sie zwei verschiedene Marken, wird das aktuelle Pixel mit einer davon markiert und die Äquivalenz der beiden Marken gespeichert. Danach müssen die Äquivalenzen aufgelöst werden, indem ihr transitiver Abschluß gebildet wird und im zweiten Durchgang dann alle Marken durch einen Repräsentanten ihrer Äquivalenzklasse ersetzt werden.

Aus Geschwindigkeitsgründen wird eine eingeschränkte Scanline-Variante eingesetzt, bei der nur ein Durchgang erfolgt. Durch die Fehlende Auflösung der Äquivalenzen werden dabei selbst für direkt benachbarte Bereiche verschiedene Marken vergeben. Die Begründung ist darin zu suchen daß in Schritt 4 nur rechts und unterhalb des aktuellen Pixels nach weiteren zu markierenden Pixeln gesucht wird - nicht oberhalb. Eine Anpassung des Algorithmus wäre leicht und würde die Komplexität nur geringfügig erhöhen. Trotzdem wurde die vorgestellte Variante eingesetzt. Die zu erkennenden Leuchtdioden sind meist rund, so daß pathologische Fälle - also die Vergabe verschiedener Marken für zusammenhängende Bereiche - selten auftreten sollten. Außerdem erfolgt während der späteren Verarbeitung eine Zusammenlegung sehr naher Schwerpunkte, so daß letztendlich in den meisten Fällen doch eine Marke für einen zusammenhängenden Objektpixelbereich verwendet werden sollte.

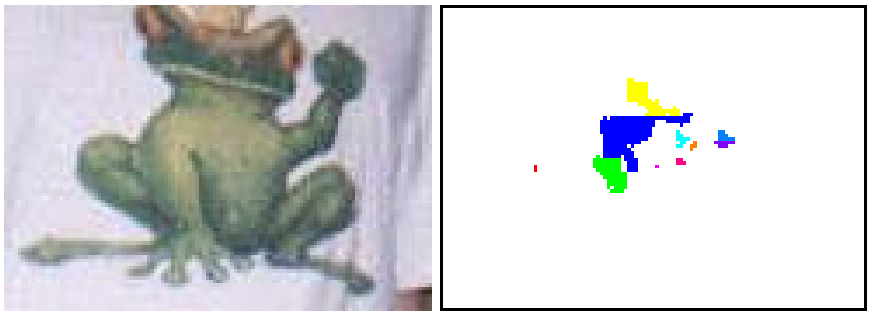


Abbildung 3.14: Ein- und Ausgabebilder von Floodfill

Algorithmus 4: Verwendete Scanline-Variante von Floodfill

1. Das Bild wird von links nach rechts, oben nach unten gescannt, bis ein noch nicht markiertes Objektpixel auftritt.

2. Tritt eines auf, wird zuerst geprüft, ob es rechts oder unterhalb von schon markierten Objektpixeln benachbart wird (Dies erfolgt in der Implementation in der willkürlichen Reihenfolge 6, 8, 7, 9). Sobald eine Marke auftritt, wird diese im nächsten Schritt weiterverwendet. Der Fall, daß das aktuelle Pixel von Pixeln benachbart wird, die zwei unterschiedliche Marken tragen, wird damit ignoriert. Links und oberhalb (Positionen 1, 2, 4) muß nicht geprüft werden, da dort keine Objektpixel sind - wären Objektpixel dort zu finden, würden sie und auch dieses Pixel schon mit einer Markierung versehen worden sein). Position 3 wird nicht geprüft.

3. Wurde bei der Prüfung eine andere Marke gefunden, wird dieses Pixel mit derselben Markierung versehen. Ist das nicht der Fall, wird eine neue Markierung vergeben.

4. In beiden Fällen wird nach der Markierung rekursiv rechts und unterhalb des Pixels (Positionen 6, 8, 7, 9) nach weiteren Objektpixeln gesucht. Diese werden mit derselben Markierung versehen. Während des Markierungsprozesses werden für jede Marke die Anzahl der beitragenden Pixel und der Schwerpunkt mitberechnet.

5. Am Ende dieses Prozesses wurden j Markierungen vergeben.

In Abbildung 3.14 wird die Ausgabe von Floodfill veranschaulicht. Die Eingabe bildete ein Bild von einem „Froschkönig", in dem Grüntöne vorkommen, die denen der Leuchtdioden entsprechen. Gesucht wurde nach Grüntönen gegeben durch einen Ausschnitt des HSV-Farbraums, wobei $h_{Min} = 40$, $h_{Max} = 110$, $s_{Min} = 55$, $s_{Max} = 255$, $v_{Min} = 50$ und $v_{Max} = 255$. Es wurden 10 Marken vergeben.

Dieser Algorithmus ist rauschanfällig. Tritt ein einzelnes Pixel ohne Nachbarn in der Objektfarbe auf, wird es als eigener Schwerpunkt behandelt. Erst in einer nachfolgenden Stufe kann es anhand der Anzahl der zum Schwerpunkt beitragenden Pixel identifiziert und eventuell aussortiert werden.

Ein weiterer Nachteil ist, daß bei der Implementierung für die maximale Anzahl vergebbarer Marken sinnvollerweise eine Obergrenze festgelegt wird. Werden mehr Schwerpunkte gefunden als Marken zur

Marke	Schwerpunktkoordinaten	Anzahl der Pixel	Farbe
1	(27,46)	2	rot
2	(48,47)	68	grün
3	(54,36)	181	blau
4	(59,26)	70	gelb
5	(62,46)	1	cyan
6	(69,38)	11	magenta
7	(68,44)	5	cyan
8	(72,40)	4	pink
9	(81,39)	9	violett
10	(81,37)	10	hellblau

Tabelle 3.1: Die von Floodfill vergebenen Marken für das Beispiel in Abbildung 3.14

Verfügung stehen, muß der Algorithmus abgebrochen werden.

Unter der Annahme, daß die Anzahl der Objektpixel viel kleiner als die Anzahl der Hintergrundpixel ist, beträgt die Komplexität pro Pixel O(1). Das ist nicht zu unterbieten und die Begründung, warum die vielen Nachteile in Kauf genommen werden.

Kapitel 4

Implementation

Es wird nun ein Erkennungssystem beschrieben, welches die in Abschnitt 2.3 entwickelten Modelle verwendet. Für das Programm sind sie fest vorgegeben. Gleichzeitig werden auch Bedienhinweise gegeben.

Die Architektur des Erkennungssystems zergliedert sich auf oberster Ebene in die Bibliothek *libseemicro* und das Frontend *seemicro*. Plattform für die Implementation ist Linux.

Die wichtigsten Klassen der Bibliothek libseemicro sind in Abbildung 4.1 zu sehen. Die Klasse **Micro** und die von ihr abgeleiteten Subklassen dienen zur Repräsentation der Modellparameter. Die wesentlichen Klassen wurden von der Klasse **RecognitionModule** abgeleitet. Sie dienen der gezielten Bilderkennung von micro.adam und micro.eva. Zur Unterstützung des Frontends und zur Steuerung einzelner RecognitionModules dienen die von **OutputModule** abgeleiteten Klassen. Sie ermöglichen eine leichte Ausgabe von Zwischen- und Endergebnissen, sowie die Parametrisierung der Erkennungsmodule. Die Erkennungsmodule lassen sich direkt und ohne OutputModules verwenden. Sie sind jedoch gedacht zur Verwendung durch eine Instanz der Klasse **MicroController**. Diese Klasse verwaltet die Module in jeweils einer Liste und ruft sie nacheinander auf. Ebenso können die Ausgabemodule durch einen MicroController verwaltet werden.

Im Frontend werden entsprechend dem Erkennungsmodus (also ob micro.adam oder micro.eva erkannt werden soll) die Erkennungsmodule zu einer Kette zusammengebaut. Ebenso werden die angeforderten Ausgabemodule instantiiert. Durch die Verwendung von MicroController muß nun nur noch die Erkennung und Ausgabe jedes Frames angestoßen werden.

Ziel der Modularisierung war, leichte Austauschbarkeit der Erkennungsalgorithmen zu ermöglichen. Ebenso können auf einfache Weise alternative Frontends entworfen werden, die z.B. ohne grafische Oberfläche auskommen.

Konkrete Implementationen der meisten verwendeten Algorithmen, sowie des GUI sind in der Bibliothek OpenCV [36] zu finden. Zur diskreten Fourier-Transformation wird FFTW [11] verwendet.

Die detaillierte Quelltextdokumentation wird mittels des Dokumentationssystems „Doxygen" im Verzeichnis „libseemicro/doc" aus dem Quelltext erzeugt. Sie ist online zu finden unter:

`http://robot.informatik.uni-leipzig.de/~mbunk/doc/html/main.html`

Die Quellen von libseemicro weisen zwei Besonderheiten auf. Zum einen sind die meisten Membervariablen öffentlich. Auf Zugriffsbeschränkungen wurde verzichtet, um während der Entwicklung unkompliziert auf

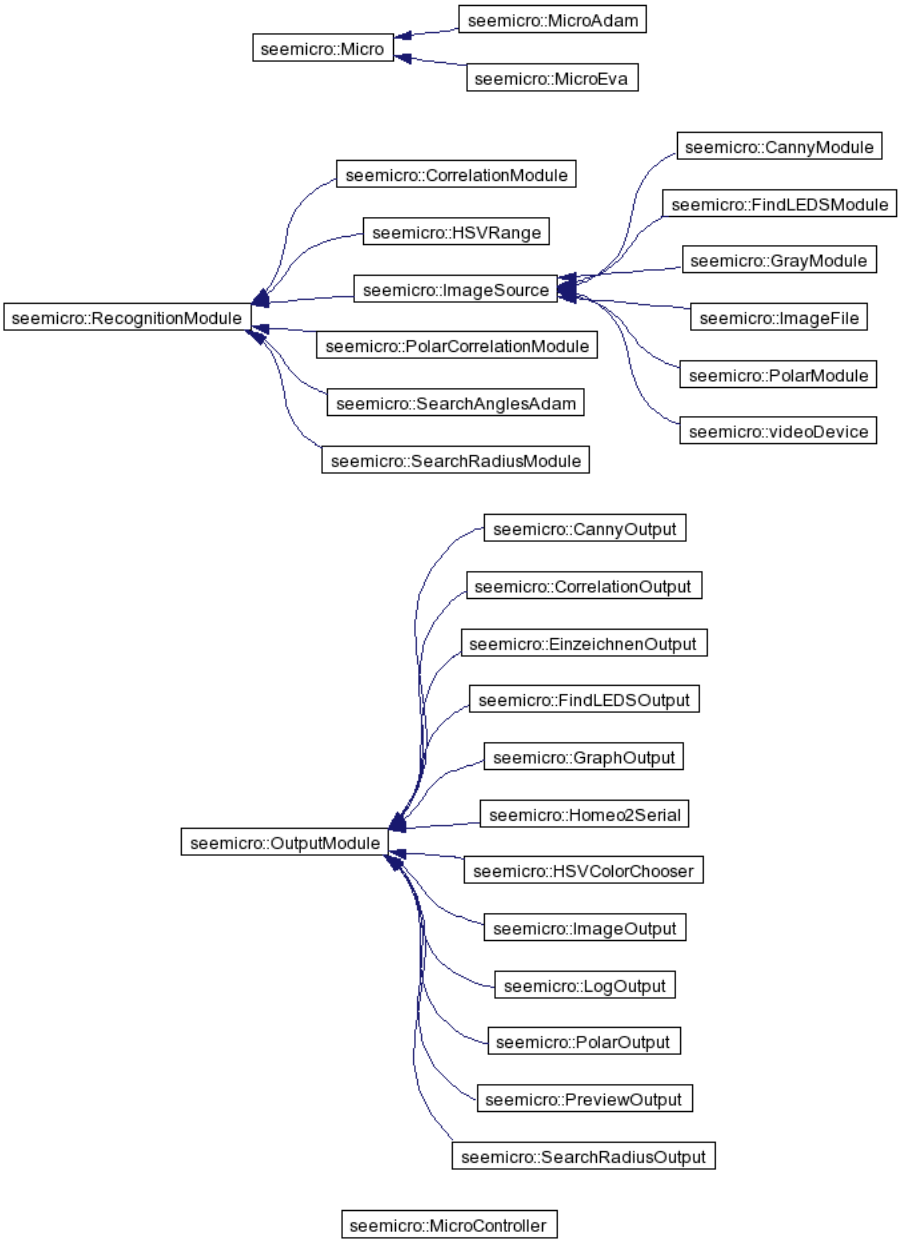


Abbildung 4.1: Die wichtigsten Klassen in libseemicro

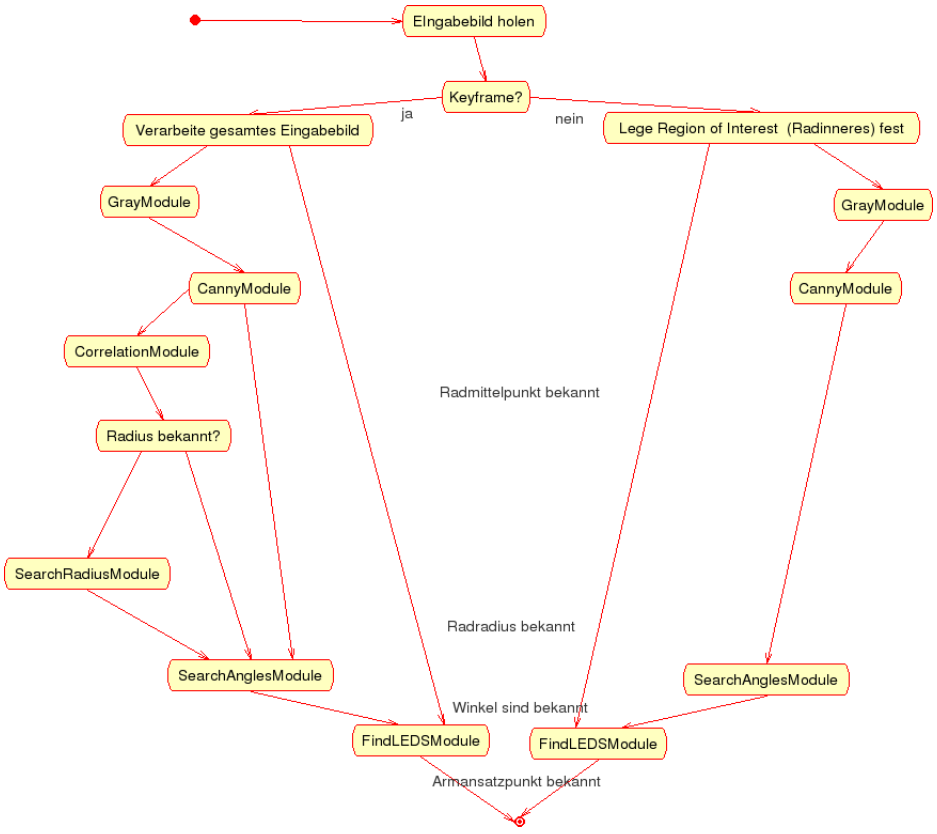


Abbildung 4.2: Überblick über die zur Erkennung von micro.adam durchgeführten Schritte

alle Variablen zugreifen zu können. Des weiteren sind die meisten Funktionen „inline", um eine schnellere Ausführung zu ermöglichen. Dieser Umstand erfordert aber, daß die Implementierung dieser Funktionen in den „.h"-Dateien steht und nicht wie üblich den „.cpp"-Dateien. Das heißt auch, daß Frontends neu kompiliert werden müssen, sobald die Implementierung in einer „.h"-Datei geändert wird!

4.1 Aufruf

Um das Programm in der Linux-Konsole zu starten, können dem Programm `seemicro` folgende Optionen übergeben werden:

```
./seemicro [--help | --image=file.jpg | --dev=/dev/video0 [--width=x --height=y]]
[--mode=adam|eva] [--debug=level] [--log=seemicro.log]

 --help
 --image=file.jpg - load image from file.jpg and recognize
        OpenCV supports:
                Windows bitmaps - BMP, DIB
                JPEG files - JPEG, JPG, JPE
                Portable Network Graphics - PNG
                Portable image format - PBM, PGM, PPM
                Sun rasters - SR, RAS
                TIFF files - TIFF, TIF
 --dev=/dev/video0 - open this video device without changing preset image size
 --width=x - request video of x pixels width
 --height=y - request video of y pixels height
 --log=file.log
 --mode=adam|eva - Set initial recognition mode
 --debug=level - Set debug level to 0, 1, 2, 3
 also: standard x toolkit options are accepted, eg. -fg green, see X(7) OPTIONS
```

Das Programm kennt zwei Modi - Adam-Modus (`--mode=adam`) und Eva-Modus (`--mode=eva`). Der Modus bestimmt, für welche Variante der Räder die Erkennung versucht wird. Als Bildquelle kann entweder ein Videogerät oder eine Bilddatei (`--image=adam.jpg`) angegeben werden. Die Videoquelle wird durch die entsprechende video4linux-Gerätedatei angegeben (z.B. mittels `--dev=/dev/video0`). In diesem Fall kann die gewünschte Auflösung angegeben werden. Je nachdem, ob das Gerät die Auflösung unterstützt, wird diese oder die nächstkleinere Auflösung verwendet - so erfordert es die Video4Linux-Spezifikation[45]. Das Debuglevel bestimmt, wieviele Meldungen während des Programmlaufs ausgegeben werden. Je größer das Level, desto mehr Meldungen werden ausgegeben. Optional können die erkannten Parameter in eine Logdatei oder einen FIFO ausgegeben werden. Das dabei verwendete Format wird in Abschnitt 4.3.9 beschrieben. Nähere Informationen zu den „standard X toolkit options" erhält man mit dem Befehl „`man 7 X`".

4.2 Tastenfunktionen

Nach dem Start wird folgender Text ausgegeben. Darin werden die über die jeweiligen Tasten erreichbaren Funktionen genannt. Die meisten Funktionen beziehen sich auf das Öffnen und Schließen der Fenster der Ausgabemodule. Bei der Tastatureingabe muß darauf geachtet werden, daß ein OpenCV-Fenster aktiv ist (nicht das xterm, aus dem seemicro gestartet wurde und kein gnuplot-Fenster!):

```
Keys to enter into windows (not this xterm!):
'a' for adam mode
'e' for eva mode
'q' to quit
'm' for radius search
's' for angles search (adam)
'i' in video mode: save input frame as frameX.pbm (X starts at 0)
'v' for video mode
'h' to get this help again
'p' toggle preview window
'S' toggle searchradius window
'c' toggle canny window
'C' toggle correlation window
'H' toggle hsvcolorchooser window
'f' toggle SearchLEDS window
'F' select SearchLEDS background
'g' toggle graph window
'o' toggle Homeo2Serial window & module
'E' toggle SearchAnglesAdam window
Numpad 1,2,3 load Homeo2Serial parameters
Shift+Numpad 1,2,3 save Homeo2Serial parameters
number to select video device.
```

4.3 Die Ausgabe- und Parametrisierungsmodule

Dieser Abschnitt erläutert, was in den jeweiligen Fenstern ausgegeben wird und welche Bedeutung die einstellbaren Parameter haben.

4.3.1 PreviewOutput - Vorschau

In diesem Fenster wird das unverarbeitete Eingabebild dargestellt. Falls als Bildquelle eine Bilddatei verwendet wird, können keine Parameter angepaßt werden.

Wird ein Videogerät als Bildquelle verwendet, lassen sich hier die Kameraparameter Helligkeit, Kontrast, Farbkontrast und Weißpunkt einstellen. Diese Einstellungen müssen vom Gerätetreiber unterstützt werden, damit sie etwas bewirken (vloopback unterstützt beispielsweise keinen der Parameter). Die hier eingestellten Werte bleiben auch nach dem Programmende erhalten, da sie vom video4linux-Treiber gespeichert werden.

Abbildung 4.3: Das PreviewOutput-Fenster

Die Kameraparameter können dazu verwendet werden, um grobe Fehleinstellungen bei der Videoaufnahme zu vermeiden. Beispielsweise sollte darauf geachtet werden, daß das Kamerabild nicht übersteuert ist, so daß weite Bereiche des Bildes weiß sind.

Auch die Keyframerate, die v.a. bestimmt, wie oft die Bildkorrelation durchgeführt wird, kann angepaßt werden. Sie wird bei Programmstart auf 25 voreingestellt. Das angezeigte Videobild wird nur für Keyframes aktualisiert.

4.3.2 CannyOutput - Canny-Algorithmus

Angezeigt wird das Kantensegmentbild, das vom Canny-Algorithmus berechnet wird.

Eingestellt werden können die Parameter „Canny1" und „Canny2" - die obere und untere Grenze für das Hysteresis-Schwellwertverfahren, welches Teil des Canny-Algorithmus ist (siehe Abschnitt 3.2.1). Je kleiner die Schwellen sind, desto mehr Kantensegmente werden gefunden. Größere Schwellen führen zu entsprechend weniger Kantensegmenten.

4.3.3 CorrelationOutput

Angezeigt wird das Korrelationsbild der in Abschnitt 3.2.2 beschriebenen Bildkorrelation.

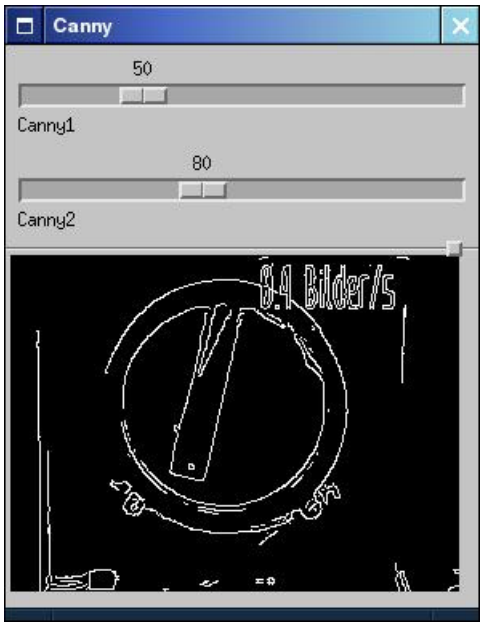


Abbildung 4.4: Das CannyOutput-Fenster

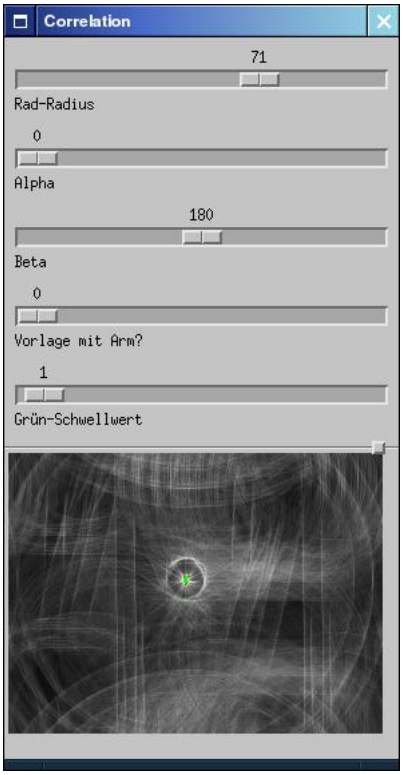


Abbildung 4.5: Das CorrelationOutput-Fenster

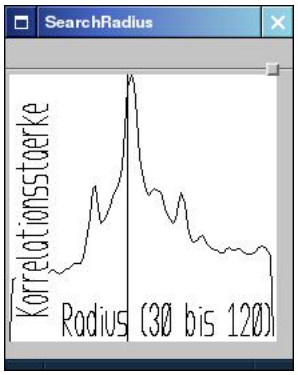


Abbildung 4.6: Das SearchRadius-Fenster

Einstellbar ist hier, ob die Vorlage von micro.adam mit dem „Arm" gezeichnet werden soll oder nicht. Ebenso können Radius, sowie - falls der Arm eingezeichnet wird - α und β der Vorlage eingestellt werden. Im Eva-Modus sind die Schieberegler „Vorlage mit Arm?", sowie „Alpha" und „Beta" nicht verfügbar.

Nach der Radiussuche beim Programmstart wird der Radius der Vorlage automatisch gesetzt. Eine manuelle Radiuseinstellung ist also normalerweise nicht erforderlich. Bei der manuellen Radiuseinstellung für micro.adam bzw micro.eva können die im SearchAnglesAdam- bzw. FindLEDSOutput-Fenster eingezeichneten Modelle als Vergleichsmöglichkeit herangezogen werden.

Ein weiterer Parameter ist der „Grün-Schwellwert" GSW. Er hat keine Auswirkungen auf die Berechnung, sondern nur auf die Ausgabe des Korrelationsbildes. Eigentlich ist das Korrelationsbild grau. Jedoch werden alle Pixel mit einem Betrag größer als 255 rot angezeigt. Alle Pixel, die über dem Schwellwert $255\ e^{0.05\ GSW}$ liegen, werden grün dargestellt. Durch Variation des Grün-Schwellwertes kann das globale Maximum experimentell im Korrelationsbild lokalisiert werden.

4.3.4 SearchRadius

In diesem Fenster wird der Zusammenhang zwischen Radius und Korrelationsstärke dargestellt. Der aktuell eingestellte Radius wird durch eine senkrechte Linie markiert.

4.3.5 SearchAnglesAdam

Dieses Fenster ist nur im Adam-Modus verfügbar. Hier wird die erkannte Radstellung von micro.adam über das invertierte Kantensegmentbild gezeichnet.

Folgende Parameter sind hier einstellbar:

- „Hough: Distance Resolution": Anzahl der Zellen, die im Akkumulator zur Entfernungsauflösung (für ρ) zur Verfügung stehen. Je mehr Zellen zur Verfügung stehen, desto genauer kann die Auflösung erfolgen. Aber desto mehr Speicher und Rechenzeit werden gebraucht.
- „Hough: Angle Resolution": Anzahl der Zellen, die im Akkumulator zur Winkelauflösung (Ausrichtung θ der erkannten Geraden) zur Verfügung stehen. Die Anzahl der Zellen wirkt sich analog zur Entfernungsauflösung aus.

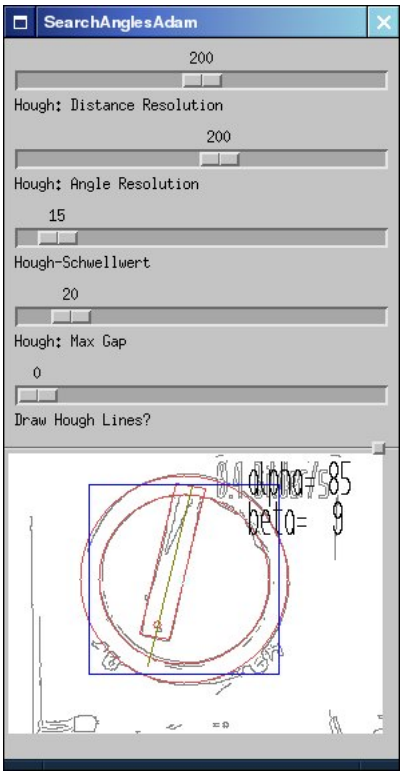


Abbildung 4.7: SearchAnglesAdam. Das blaue Rechteck kennzeichnet die „region of interest". Die den Arm repräsentierende Gerade g_{Arm} ist in Gelb eingezeichnet. Sie endet jeweils in den Schnittpunkten mit dem Kreis mit Radius r und Mittelpunkt M. Die erkannte Radstellung ist in Rot eingezeichnet.

- „Hough-Schwellwert": Schwellwert, ab dem angenommen wird, daß eine Akkumulator-Zelle eine wirkliche Gerade im erkannten Bild repräsentiert. Je kleiner dieser Wert ist, desto mehr Geraden werden ausgegeben.
- „Hough: Max Gap": Gibt die maximale Größe von Lücken zwischen Liniensegmenten an. Besteht zwischen zwei Segmenten eine kleinere Lücke als „Max Gap", so werden die Segmente verbunden.
- „Draw Hough Lines?": Es ist möglich, die von der Hough-Transformation gefundenen Linien einzeichnen zu lassen. Dies ermöglicht eine gezieltere Einstellung der Hough-Parameter. Die in OpenCV verwendete HighGUI-Bibliothek bietet keine Möglichkeit CheckBoxen zu verwenden. Deshalb wird an dieser Stelle ein Schiebebalken verwendet, mit dem nur zwei Werte einstellbar sind - 0 und 1.

4.3.6 HSVColorChooser

An dieser Stelle kann der Ausschnitt des HSV-Farbraums (siehe Abschnitt 3.2.4) bestimmt werden, der als Objektfarbe für die LED-Suche verwendet werden soll. Dazu können für H, S und V jeweils die Ober- und Untergrenze eingestellt werden ($H_{Max}, H_{Min}, S_{Max}, S_{Min}, V_{Max}, V_{Min}$). Für die grünen Leuchtdioden von micro.adam und micro.eva Version 1 sollten sinnvolle Werte voreingestellt sein.

Das Fenster ist unterhalb der Schieberegler in zwei Bereiche unterteilt. Im oberen Bereich wird eine SV-Ebene (x-Achse: S, y-Achse: V; links oben befindet sich (0,0)) für den Farbton $H_{Mitte} = \frac{H_{Max}+H_{Min}}{2}$

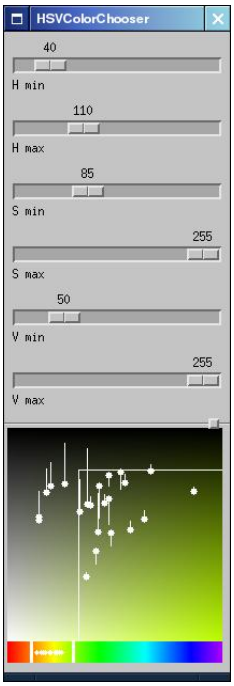


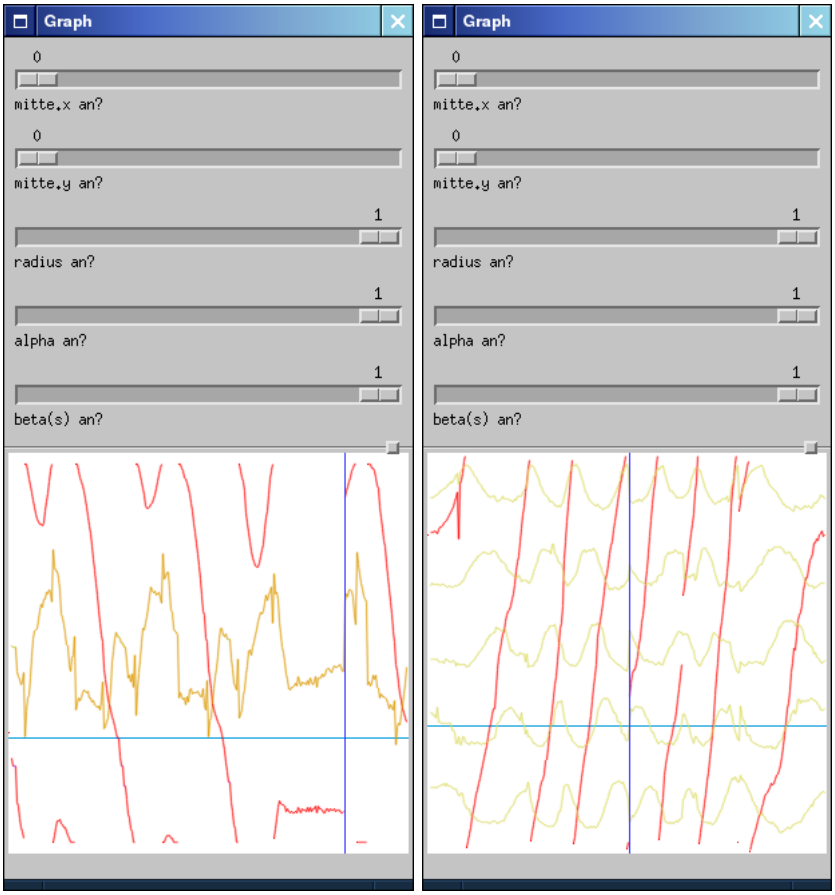
Abbildung 4.8: HSVColorChooser

angezeigt, also die Mitte des ausgewählten H-Bereichs. Auf dieser Farbebene sind für einige im Quelltext fest vorgegebene Farbwerte kleine weiße Kreise eingezeichnet. Diese Farbtöne stammen von grünen Leuchtdioden wurden mittels eines Bildbearbeitungsprogramms aus vom Programm zu verarbeitenden Bildern extrahiert. Von diesen Kreisen führt eine Linie entweder nach oben oder nach unten. Die Linie repräsentiert die Entfernung dieses Farbwertes von seinem idealen Hue-Wert. Zeigt die Linie nach oben, so ist der Hue-Wert kleiner als H_{Mitte}. Zeigt die Linie nach unten, ist die ideale Farbebene größer als H_{Mitte}. Durch Variation von H_{Min} und H_{Max} sollten alle Linien möglichst kurz eingestellt werden, um den gesuchten Farbton einzugrenzen. Außerdem wird auf der SV-Ebene ein Rechteck dargestellt, das durch die Grenzen von S und V bestimmt ist. Alle Farbtöne, die innerhalb des Rechtecks liegen, werden als Objektfarben angesehen.

Im untersten Bereich des Fensters ist eine Farbleiste, die alle Hue-Werte von 0 bis 359 repräsentiert. Auf ihr sind ebenfalls weiße Kreise an den Stellen eingezeichnet, an denen die vorgegebenen Farbwerte zu finden sind. Außerdem werden zwei senkrechte weiße Linien eingezeichnet, die H_{Min} und H_{Max}repräsentieren. Somit kann ein Farbtonintervall festgelegt werden. Nachteil dieser Methode ist, daß es nicht möglich ist ein Farbtonintervall wie [350°,10°] einzustellen, obwohl der Wertebereich von H zyklisch ist.

4.3.7 SearchLEDS

In diesem Fenster werden die als LEDs erkannten Bildbereiche eingezeichnet. Im Eva-Modus werden zusätzlich die gefundenen Armansatzpunkte (Γ_i) eingezeichnet. Ein Screenshot ist in Abbildung 3.7 zu sehen. Als Hintergrund kann entweder das Eingabebild oder das Ausgabebild von Floodfill gewählt werden. Das Eingabebild kommt meist direkt von der Bildquelle. Das Ausgabebild von Floodfill entspricht den durch Floodfill vergebenen Marken, verdeutlicht durch bestimmte Farben.



(a) Im Adam-Modus. Links des blauen Balkens wurden die neuesten Parameter eingezeichnet.

(b) Im Eva-Modus. Die Unstetigkeit im Graph von α rechts des blauen Balkens weist auf einen Erkennungsfehler hin.

Abbildung 4.9: Das Fenster des GraphOutput-Moduls

Bei sehr hellen Eingabebildern kann starkes Rauschen auftreten, da in vielen weißen Pixeln der gesuchte Farbton erkannt wird. In einer solchen Situation empfiehlt es sich, S_{Min} mit dem HSVColorChooser zu vergrößern.

4.3.8 GraphOutput

In diesem Fenster werden die erkannten Modellparameter graphisch dargestellt. Die Schiebebalken ermöglichen das An- und Abstellen einzelner Graphen. Eine Pixelbreite entspricht einem Frame. Das Fenster wird nur für Keyframes neu gezeichnet. Jeweils links des blauen senkrechten Balkens erschienenen die neuesten Daten.

Interessant sind vor allem die Graphen von α und β (bzw. der $\beta_{i_{Arm_j}}$). α wird dabei auf die gesamte Plotfläche skaliert und rot dargestellt, die Nulllinie ist am oberen Bildrand, 2 π am unteren Bildrand. β wird im Adam-Modus auf $\pm 20°$ skaliert, die Nulllinie ist in der Bildmitte. Bei dieser Skalierung ist erkennbar, daß die für β gewonnenen Werte relativ ungenau sind. Im Eva-Modus werden alle fünf $\beta_{i_{Arm_j}}$

untereinander dargestellt.

Die Graphen von α und β (bzw. der $\beta_{i_{Arm_j}}$) sollten im Rahmen der Erkennungsgenauigkeit stetig sein. Treten Sprünge auf, so liegt wahrscheinlich ein Erkennungsfehler vor.

4.3.9 LogOutput

Durch Verwendung des LogOutput-Moduls können die erkannten Parameter in eine Datei geschrieben werden. Es kann auch ein zuvor mit dem Unix-Befehl `mkfifo` angelegter FIFO („First In, First Out") als Ausgabedatei verwendet werden. Von dort können die Daten sofort von einem anderen Programm weiterverarbeitet werden.

Für jeden Frame wird eine Zeile ausgegeben.

Im Adam-Modus wird folgendes Format verwendet, wobei das Symbol ⇝ bedeuten soll, daß die Zeile an dieser Stelle fortgesetzt wird, aber das Symbol selbst nicht in den Logdaten auftaucht:

`frame=`<Framenummer> `mittevalid=`<0-oder-1> `mitte=(`<x> <y>`) radiusvalid=`<0-oder1->⇝
`radius=`<n Pixel> `alphavalid=`<0-oder-1> `alpha=`<float> `beta=`<float>`\n`

Die Wertebereiche der Winkel sind wie in Abschnitt 2.3 festgelegt. Ihre Werte werden in radians angegeben.

Eine vollständige Log-Zeile sieht beispielsweise so aus:

```
frame=7 mittevalid=1 mitte=(130 93) radiusvalid=1 radius=71 alphavalid=1 alpha=1.50032 beta=0.143488\n
```

Die Framenummer beginnt bei 0. Die `valid`-Werte geben an, ob die entsprechenden Parameter gültig sind. Wurden in einem Frame bei der Hough-Transformation keine geeigneten Armaußenkanten gefunden, konnten also α und β nicht bestimmt werden, wird `alphavalid=0` ausgegeben. Radradius und Radmitte sind nur zu Beginn der Erkennung und bei expliziter Aufforderung zur Radius- oder Mittelpunktesuche unbekannt.

Im Eva-Modus wird im Prinzip dasselbe Format verwendet, außer daß die 5 β-Werte der Arme statt des einen β von micro.adam ausgegeben werden:

`frame=`<Framenummer> `mittevalid=`<0-oder-1> `mitte=(`<x> <y>`) radiusvalid=`<0-oder1->⇝
`radius=`<n Pixel> `alphavalid=`<0-oder-1> `alpha=`<float> `beta0=`<float> `beta1=`<float>⇝
`beta2=`<float> `beta3=`<float> `beta4=`<float>`\n`

Das im nächsten Abschnitt vorgestellte Programm `logdata2speed.c` kann als Beispiel dienen, wie eine Weiterverarbeitung der Logdaten erfolgen kann.

4.4 Berechnung der Raddrehgeschwindigkeit

Mit dem Programm `logdata2speed.c` können aus Logdaten im Adam-Format die Drehgeschwindigkeit von micro.adam und die von seemicro gelieferte Erkennungsrate berechnet werden.

Ein Aufruf ist folgendermaßen möglich:

Ein FIFO wird angelegt:
`cd seemicro`
`mkfifo fifo.log`
In einem xterm wird das Erkennungsprogramm gestartet, wobei die Logdaten in einen FIFO ausgegeben werden:
`./seemicro --dev=/dev/video0 --log=fifo.log`
In einem anderen xterm wird das Meßprogramm gestartet, wobei Logdaten aus dem FIFO gelesen werden:
`cd seemicro`
`../logdata2speed fifo.log`

Folgende Ausgabe wird dann geliefert:

```
Framerate=25.0 Frames/Sekunde
Aufsummierung der Geschwindigkeit über die letzten 10 Frames.
Aufsummierung der Erkennungsrate über die letzten 100 Frames.
Opening 'adam.log'...
Opened f.
Buffering mode: line-buffered
Alpha= 18° speed= 16.4°/frame= 410.9°/s= 1.1u/s mittevalid=100% radiusvalid=100% alphavalid=100%
```

Aufgrund der stark schwankenden Geschwindigkeit wäre eine Angabe in u/min wenig aussagekräftig[1].

4.5 Steuerung von micro.adam

Zum Schließen der sensomotorischen Schleife soll anhand der erkannten Parameter micro.adam gesteuert werden.

Die Steuerung erfolgt ebenso wie im dieser Arbeit vorausgehenden Projekt (siehe Abschnitt 1.1) mit einem nach dem Ansatz der Homeokinesis arbeitenden Neuron [6]. Dabei wird versucht, selbstorganisiertes, emergentes Verhalten zu erzeugen, indem der Agent (in unserem Fall micro.adam) mit einem adaptiven Modell seines eigenen Verhaltens versehen wird. Die Aufgabe der Steuerung ist es, den internen Zustand stabil zu halten. Gleichzeitig werden nun das Modell des eigenen Verhaltens und die Steuerung durch das Fehlersignal zwischen dem aus den Sensordaten gewonnenen wirklichen Verhalten und dem durch das Modell vorhergesagten Verhalten angepaßt.

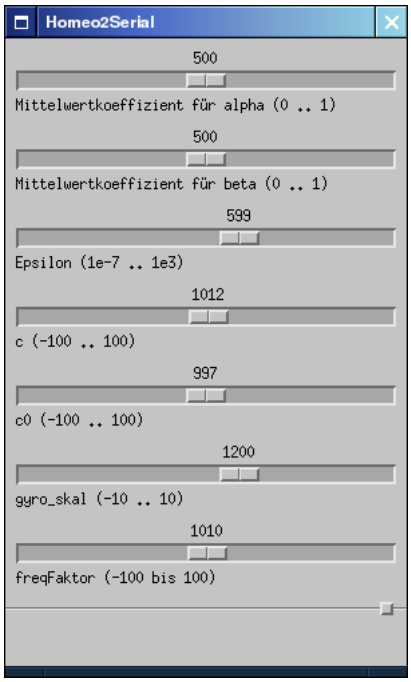
In diesem Abschnitt wird zwar Homeokinesis verwendet, es erfolgt jedoch keine Erklärung der Funktionsweise, Zusammenhänge oder Formeln. Dazu sei noch einmal explizit auf die Quellen verwiesen.

Als Eingabe werden im Gegensatz zu früher nicht die Meßwerte des Gyroskops verwendet, sondern die durch die Bilderkennung für α und β gewonnenen Parameter. Der Wert zur Ansteuerung der Armservos wird über eine serielle Verbindung vom bildauswertenden PC zum Mikrocontroller auf der Radhardware gesendet.

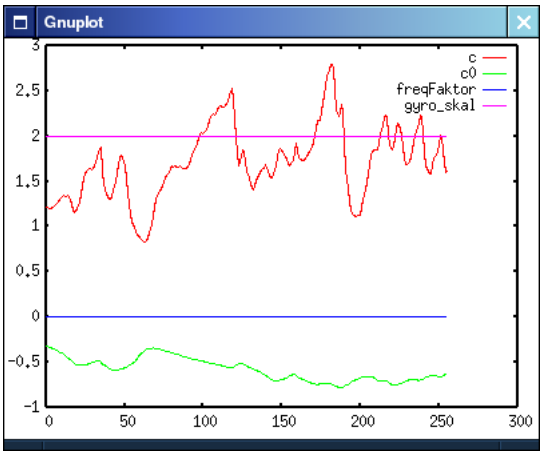
Die Implementation erfolgte in der von OutputModule abgeleiteten Klasse **homeo2serial**.

Im „Homeo2Serial"-Fenster sind die Parameter der Homeokinesis einstellbar. Da die Wertebereiche der Schieberegler nur ganze Zahlen von 0 bis zu einem Maximalwert umfassen können, müssen die eingestellten Parameter umgerechnet werden. Die dadurch erlangten Wertebereiche sind im Fenster in Klammern angegeben. Die Parameter bedeuten:

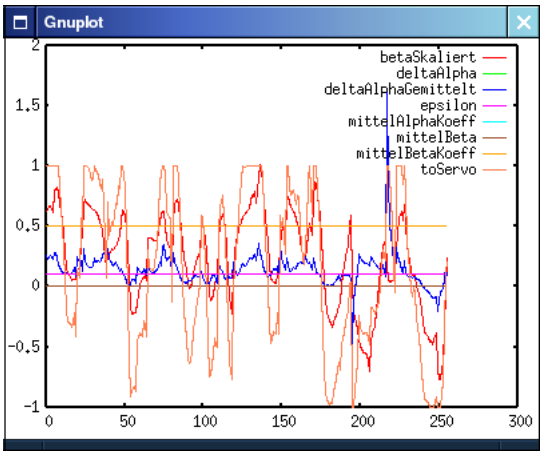
[1] Die verwendete Einheit „u" für Umdrehungen kann weggelassen werden: $1\frac{u}{s} = 1/s$



(a) Einstellung der Parameter der Homeokinesis



(b) In den dazugehörigen gnuplot-Fenstern werden die Parameter und Ausgaben angezeigt.



(c) Im zweiten gnuplot-Fenster werden die Variablen angezeigt, die sich nur etwa im Intervall [-1,1] bewegen

Abbildung 4.10: Die Fenster des Homeo2Serial-Moduls

- Mittelwertkoeffizient mw_α für $\Delta\alpha$: Es erfolgt eine Glättung der $\Delta\alpha$ mit folgender Formel:

$$\widetilde{\Delta\alpha}(t) = 0.001\ mw_\alpha\ (\Delta\alpha - \widetilde{\Delta\alpha}(t-1)) \tag{4.1}$$

  Ist mw_{alpha} klein, erfolgt eine starke Glättung. Ist es auf dem Maximalwert 1000, wird $\Delta\alpha$ direkt übernommen und es erfolgt keine Glättung mehr.

- Mittelwertkoeffizient mw_{beta} für β: Es erfolgt eine Glättung nach demselben Prinzip:

$$\widetilde{\beta}(t) = 0.001\ mw_\beta\ (\beta - \widetilde{\beta}(t-1)) \tag{4.2}$$

- Epsilon: Lernkoeffizient für c und $c0$
- c: synaptische Stärke des Eingangssignals für $\Delta\alpha$
- c0: Bias des Neurons
- gyro_skal: s.u.
- freqFaktor: Legt fest, wie schnell das Umlernen erfolgt

Zu beachten ist, daß beim Verstellen eines Schiebebalkens alle anderen von den Schiebebalken kontrollierten Werte ebenfalls neu gesetzt werden. Dabei werden c und $c0$ gerundet!

Die Ausgabe des Neurons wird folgendermaßen berechnet:

$$y = 1.2\ \tanh(c\ \widetilde{\beta} + c_0) + 0.01\ \sin\ t \tag{4.3}$$

Der erste Summand realisiert dabei die sigmoide Funktion des Neurons. Der zweite Summand simuliert ein Rauschen.

Danach werden die Parameter entsprechend ihrer Lernregeln angepaßt:

$$c(t+1) = c(t) - 2\ \epsilon\ (y^2\ c(t) - 1) \tag{4.4}$$

und

$$c_0 = c_0 - (freqFaktor\ \epsilon\ \widetilde{\Delta\alpha}^2 + 0.01)\ y \tag{4.5}$$

Dann wird der letztliche Ausgabewert berechnet:

$$toServo = y + \widetilde{\Delta\alpha}\ gyro\_skal \tag{4.6}$$

Der aktuelle Zustand aller Parameter läßt sich mit den Tasten Shift + „1" bis „3" des numerischen Tastenblocks speichern. Laden eines Zustandes ist ebenfalls mit den Tasten „1" bis „3" des numerischen Tastenblocks möglich (aber ohne Shift).

Experimentell wurde folgende „funktionierende Parameterkombination" gefunden (Funktionierend bedeutet dabei, daß micro.adam ein ihm angemessenes Verhalten ausführte - er führte vollständige Umdrehungen aus):

Parameter	Wert des Schiebereglers
Mittelwertkoeffizient $\Delta\alpha$	500
Mittelwertkoeffizient β	500
Epsilon	599
c	1012
c0	1000
gyro_skal	1200
freqFaktor	1696 (1512 ging auch)

Das Modul sendet die Steuerungsdaten über die serielle Schnittstelle `/dev/ttyS0`. Der Pfad zu diesem Gerät ist derzeit nur im Quelltext änderbar. Gesendet wird ein maximal 10 Bytes langes Kommando mit einer Baudrate von 2400 bit/s.

4.6 Ergebnisse

4.6.1 Erkennungsbedingungen und -raten

In diesem Abschnitt soll die Qualität der unter verschiedenen Erkennungsbedingungen gelieferten Ausgabewerte eingeschätzt werden. Dazu wird teilweise die erreichte Erkennungsrate herangezogen, wie sie von `logoutput2speed.c` geliefert wird.

1. Eine Abhängigkeit vom Kameratyp besteht nicht. Die Erkennung funktionierte mit einer Webcam „Logitech Quickcam" ebenso gut wie mit einer an eine Frame-Grabber-Karte angeschlossenen Videokamera.

2. Die Erkennung funktioniert weitgehend unabhängig von den Lichtverhältnissen. Dies wird zum einen durch die Detektion von Kantensegmenten im Canny-Algorithmus mittels Nicht-Maximum-Unterdrückung ermöglicht. Entscheidend sind Grauwertkontraste, nicht die Grautöne. Andererseits sind die Leuchtdioden selbstleuchtend. Also ist ihr Farbton nicht davon abhängig, ob künstliches oder natürliches Umgebungslicht vorhanden ist. Jedoch kann der automatische Weißabgleich von Videokameras und Webcams die Farbtöne verfälschen.

3. Die maximale und minimale Kameraentfernung sind nun allein von der Kameraoptik abhängige Parameter. Sie bestimmen sich aus dem Öffnungswinkel der Kamera.

4. Die Verwendung eines blauen statt eines weißen Hintergrundes bewirkte stärkere Kontraste im Bild. Schatten verschwanden und die Erkennung funktionierte besser.

5. Das Rad muß nicht im Bild zentriert sein. Es reicht, wenn es vollständig im Bild ist.

6. Es muß eine minimale Größe von 30 Pixeln haben. Diese Untergrenze ist im Quelltext so festgelegt. Jedoch macht ein viel kleinerer Radius wenig Sinn, denn je kleiner der Radradius im Bild ist, desto weniger genau kann die Radstellung aufgelöst werden - für α und β (bzw. β_i bei micro.eva) stehen dann nur wenige diskrete Werte zur Verfügung.

   Die maximale Radgröße ist einerseits durch die Auflösung festgelegt. Größer als $\min(x_{Res}, y_{Res})/2$

Pixel darf der Radius nicht sein, denn sonst ist das Rad nicht mehr vollständig im Bild. Andererseits ist im Quelltext für die Radiussuche eine maximale Obergrenze von 120 festgelegt.

7. Zu klein gewählte Hough-Auflösungen verringern die Erkennungsrate. Um eine Winkelauflösung zu erreichen, bei der zwei Linien aufgelöst werden können, die sich im Winkel von 1° schneiden, sind 360 Akkumulatorzellen für den Parameter θ erforderlich. Ähnlich verhält es sich mit der unter „Distance Resolution" für die Hough-Transformation angegebenen Zellenanzahl. Werden die Parameter „Hough Schwellwert" zu groß und „Max Gap" zu klein gewählt, werden nicht genügend Linien erkannt. Die Armaußenkanten von micro.adam müssen erkannt werden, sonst können α und β nicht bestimmt werden (angleValid=0).

Die Radgeschwindigkeit von micro.adam wurde mit `logdata2speed.c` gemessen. Sie betrug maximal etwa 1.2 u/s. Die maximal meßbare Radgeschwindigkeit ist abhängig von der Framerate (s.u.).

In einem Experiment wurde micro.adam relativ zur Kamera schräg aufgestellt. Bei der Erkennung betrug der Radius 72 Pixel (Auflösung: 320×240). Dabei erschien der Kreisring ellipsenförmig. Bis etwa zu einem Winkel von 20° funktionierte die Erkennung trotzdem. Radius- und Mittelpunktsuche ergaben zwar keine völlig korrekten Werte, meist wurde jedoch der Vorlagenkreisring über ein Ellipsensegment im Kamerabild gelegt, so daß der Bereich im Radinneren korrekt weiterverarbeitet werden konnte. Festgehalten werden kann also, daß keine genaue Ausrichtung von Kamera und Rad zueinander erforderlich ist.

Die erreichbare Genauigkeit für α ist erstaunlich. Man wird an eine Messung erinnert, welche als „... quantitative Bestimmung des Wertes einer Meßgröße..." [42] verstanden wird. Es stellt sich sofort die Frage nach dem Meßfehler. Sie soll hier nicht beantwortet werden. Jedoch soll die Anwendungsmöglichkeit der Bilderkennung für Messungen festgehalten werden. Diese Anwendung ist erstaunlich, da wir gewohnt sind, von unseren Augen keine quantitativen Daten zu bekommen.

Verfolgt man die Armnummern von micro.eva, die in das Ausgabebild von FindLEDS eingezeichnet werden, so kann man bisweilen ein Umspringen der Armnummern beobachten. Für diese Erkennungsfehler gibt es zwei wahrscheinliche Ursachen:

1. Die LED von Arm #1 wurde nicht erkannt
2. „lost tracking", siehe unten

4.6.2 Echtzeitfähigkeit

Echtzeitfähigkeit ist wichtig für eine zeitnahe Steuerung von Robotern. Fährt ein Roboter gegen ein Hindernis und stellt erst nach der Kollision fest, daß er hätte ausweichen sollen, erfolgt die Reaktion zu spät. Im Falle von micro.adam und micro.eva muß die Armsteuerung so zeitnah erfolgen, daß auf die anstehenden Ereignisse wie „Ausrollen" und „Fallen" rechtzeitig reagiert werden kann. Das Auftreten dieser Ereignisse ist je nach Radgeschwindigkeit unterschiedlich. Dreht sich das jeweilige Rad schneller, müssen auch die Arme schneller koordiniert werden.

Um die Verarbeitungsgeschwindigkeit der Implementierung einschätzen zu können, wurden zwei Funktionen implementiert. Zum einen wird im PreviewOutput-Fenster die effektiven Framerate eingeblendet. Das ist die Framerate, mit der die Verarbeitung erfolgt.

Zum anderen werden, wenn seemicro auf spezielle Weise übersetzt wurde (siehe Makefile), die Verarbeitungszeiten pro Frame und pro Keyframe in Millisekunden angezeigt, z.B.:

```
msecs:: recognize/output/waitkey Normal: 57 0 5 Key: 134 9 5 rec:19 0 4 0 0 24 8
```

Dabei stehen die ersten drei Werte (57 0 5) für die Anzahl der Millisekunden, die zur Verarbeitung eines normalen Frames für alle Erkennungs- und Ausgabemodule sowie den Aufruf der cvWaitKey-Funktion gebraucht wurden. Letztere Funktion gehört zu OpenCV und ist einerseits zur Abfrage von Tastaturereignissen zuständig. Andererseits werden in ihr aber auch anstehene X-Events verarbeitet. Die nächsten drei Werte (134 9 5) schlüsseln nach der gleichen Einteilung den Zeitaufwand für die Verarbeitung eines Keyframes auf. Aufgrund der Korrelation und der Verarbeitung des vollen Bildes ist der Zeitaufwand hierfür deutlich größer. Die nächsten sieben Werte schlüsseln den Zeitaufwand zur Verarbeitung eines normalen Frames noch einmal für die einzelnen Erkennungsmodule auf. Die verwendete Reihenfolge ist: VideoDevice, GrayModule, CannyModule, CorrelationModule, SearchRadiusModule, SearchAnglesAdam und FindLEDSModule.

Ist zusätzlich DEBUGLEVEL > 0, so warnt das Programm, wenn nicht genug Rechenleistung zur Bilderkennung bei voller Framerate zur Verfügung steht.

Die Echtzeitfähigkeit bei einer Bildauflösung von 320x240 Pixeln auf einem 1 GHz Intel-Rechner ist gegeben. Dabei wird aber nicht die volle Framerate von 25 Frames/s erreicht, sondern nur etwa 10 Frames/s. Trotzdem kann der Arm von micro.adam angemessen gesteuert werden. Bei langsameren Rechnern können bisweilen „lost tracking"-Situationen auftreten, wenn der Folgezustand nicht richtig dem letzten bekannten Zustand zugeordnet werden konnte, weil die Bildverarbeitung eines Keyframes (oder auch wegen anderer auf dem Computersystem aktiver Prozesse) zu lange gedauert hat, die Verarbeitung einiger Zwischenframes übersprungen wurde und sich das Rad zu weit weiterbewegt hatte.

Bei micro.eva reichen dafür $72/2°$, denn sonst erfolgt in Gleichung (3.35) die Bestimmung des ersten Armes anhand von $\alpha(t-1)$ falsch. Es ergibt sich:

$$v_{micro.eva} < \frac{72}{2}°/Frame = 36°/Frame \tag{4.7}$$

Bei einer Framerate von 25 Frames/s sind das:

$$v_{micro.eva} < 36°/Frame = \frac{1}{10}\ u/Frame * 25\ Frames/s = 2,5\ u/s \tag{4.8}$$

Bei einer aufgrund unzureichender Rechenleistung effektiven Framerate von 10 Frames/s ist eine maximale Radgeschwindigkeit von 1 u/s verfolgbar.

Die mit der implementierten Verfahrensweise maximal erkennbare Drehgeschwindigkeit von micro.adam kann größer sein, weil er nur einen Arm besitzt und deshalb keine Zuordnung zu benachbarten Armen erfolgen kann. Es gilt:

$$v_{micro.adam} < 180°/Frame = \frac{1}{2}\ u/s * 25\ Frames/s = 12,5\ u/s \tag{4.9}$$

Werden Radgeschwindigkeit und Beschleunigung mitgeführt, können höhere Radgeschwindigkeiten bei niedriger Framerate verfolgt werden.

Kapitel 5

Ausblick

Der offensichtlichste nächste Schritt ist die Steuerung von micro.eva mittels der erkannten Daten.

Unter Beibehaltung des Themas sind Verbesserungsmöglichkeiten gegeben und teilweise schon angesprochen worden.

Bei der Zuordnung der Armnummern von micro.eva ließe sich die Erweiterung des Modells um die Raddrehgeschwindigkeit vorteilhaft anwenden, um eine Vorhersage für α zu erhalten. Diese kann zur Detektion nicht oder inkorrekt erkannter Frames, sowie von „lost tracking" Situationen verwendet werden. Letzteres erfordert zusätzlich die Verwendung von Zeitmarken für die Frames. Unter der Annahme einer konstanten Radgeschwindigkeit ließe sich die Vorhersage für den aktuellen Frame zur Zuordnung nutzen.

Eine weitergehende Verwendung des Wissens aus früheren Frames würde durch eine Erweiterung des Modells um Beschleunigung und Masse des Rades ermöglicht. Es könnten Radgeschwindigkeit und -position noch genauer geschätzt werden. Somit könnte die Zuverlässigkeit der Erkennung erhöht werden.

Ebenso könnte über zuverlässigere Alternativen oder Ergänzungen zur Hough-Transformation nachgedacht werden. Bei der Verfolgung von micro.adam muß Floodfill nicht für jeden Frame aufgerufen werden.

Interessante neuen Fragestellungen ergeben sich aber insbesondere aus den Grenzen des Ansatzes.

Allem voran fällt die Inflexibilität der Erkennung ins Auge. Das fest vorgegebene Modell legt uns auf eine spezielle Anwendung fest. Wie könnte eine allgemeinere Erkennung funktionieren, bei der die Modelle nicht fest vorgegeben, sondern ersetzbar sind? Wie können Modelle erlernt werden?

Die interessanteste zukünftige Fragestellung ist aber die der Anwendbarkeit des bei der Wahrnehmung stattfindenden Erkennungsvorganges zur Umsetzung höherer kognitive Fähigkeiten wie Planung und Problemlösung. Wahrnehmungsvorgänge könnten im Denkprozeß eine grundlegendere Bedeutung haben als nur Sinneswahrnehmung. Zum Beispiel könnten Problemlösungen direkt, „intuitiv" wahrgenommen werden.

Dieses „intuitive Denken" ergänzt das „symbolmanipulierende Denken". Ein intuitiver Denkvorgang ist nicht durch Introspektion beobachtbar, wohl aber die Lösungssuche während der Symbolmanipulation. Beispielsweise denkt man beim Schachspiel über die möglichen nächsten Züge nach. Auf der Ebene der Züge werden Symbole manipuliert, indem Stellungen bewertet und optimale Züge ausgewählt werden. Mögliche Züge werden zwar durch die Regeln vorgegeben, der Suchraum ist dadurch aber sehr groß. Das

„intuitive Denken" kann an dieser Stelle helfen, aus allen möglichen Zügen erfolgversprechende auszuwählen und sinnlose Züge zu vernachlässigen. Dies garantiert wie auch beim Einsatz von Heuristiken keine optimalen Lösungen und kann dazu führen, daß wichtige Züge „übersehen werden". Es eröffnet aber auch eine alternative Sichtweise, die ohne Symbole auskommt.

Piagets Stadientheorie [23] unterteilt die geistige Entwicklung des Menschen in eine sensomotorische Periode, eine Periode der konkreten Operationen und eine Phase der formalen Operationen. Symbolisches Denken entwickelt sich dabei schrittweise. Da es also erst mit einem bestimmten Alter beim Menschen auftritt, stellt sich hier die Frage nach seinen Grundlagen. Für die Bedeutung der Wahrnehmung als Grundlage kognitiver Fähigkeiten spricht auch, daß sie in der Evolution dem bewußten Denken vorausging.

Kapitel 6

Zusammenfassung

Diese Arbeit befaßte sich mit dem Problem des echtzeitfähigen modellbasierten Verfolgens von zweidimensionalen Objekten in monokularen Bildsequenzen. Es wurden die Roboter micro.adam und micro.eva als Mikrowelt verwendet, um eine sensomotorische Schleife zu konstruieren. Dazu wurde eine Kamera auf die Roboter gerichtet, so daß sie sich „selbst wahrnehmen" konnten. In den zur modellbasierten Erkennung aufgestellten Modellen wurden Radradius, Radmittelpunkt und Armstellungen durch Parameter realisiert, welche vom Erkennungsprozeß geliefert werden mußten. Dazu wurden der Canny-Algorithmus, Bildkorrelation im Frequenzraum, Hough-Transformation und ein Komponentenlabelingalgorithmus verwendet. Die für das Modell extrahierten Parameter wurden dann zur Steuerung vom micro.adam verwendet.

Es wurde ein weiterer praktischer Beweis der nicht angezweifelten, aber doch nicht als selbstverständlich akzeptierten spezialisierten Echtzeitbilderkennung mit Computern erbracht. Das implementierte System verarbeitet die Sensordaten zielgerichtet, denn verschiedene Algorithmen wurden verkettet. Die Verwendung abstrakter Modelle erlaubt die Formulierung von Ausschlußbedingungen. Sie sind wichtige Bausteine im hier durchgeführten Erkennungsprozeß. Die Erkennung funktionierte so zuverlässig und schnell, daß eine angemessene Steuerung von micro.adam erfolgen konnte.

Index

Literaturverzeichnis

[1] Roman Barták „Online Guide to Constraint Programming", First Edition, 1998, http://kti.ms.mff.cuni.cz/~bartak/constraints/index.html

[2] Stan Birchfield „KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker", http://www.ces.clemson.edu/~stb/klt/

[3] G.R. Bradski „Computer vision face tracking as a component of a perceptual user interface" in Workshop on Applications of Computer Vision, S. 214-219, Princeton, NJ, Okt. 1998, http://www.intel.com/technology/itj/q21998/articles/art_2.htm

[4] J. Canny „A Computational Approach to Edge Detection", IEEE Trans. on Pattern Analysis and Machine Intelligence, 8(6), 1986, pp. 679-698

[5] Peter Corke „Visual Servoing Home Page", 1999, http://www.cat.csiro.au/cmst/staff/pic/vservo.htm

[6] Der, R., Steinmetz, U., and Pasemann, F. „Homeokinesis - a new principle to back up evolution with learning". In Computational Intelligence for Modelling, Control, and Automation, volume 55 of Concurrent Systems Engineering Series, 1999, S. 43 47. IOS Press. Siehe auch http://www.informatik.uni-leipzig.de/~der/Forschung/homeoki.html

[7] John Mc Donald „The Hough Transform: Explained and Extended", 1998, http://www.cs.may.ie/~johnmcd/SNHT/sld026.htm

[8] Robert Fisher, Simon Perkins, Ashley Walker, Erik Wolfart „Hypermedia Image Processing Reference", Department of Artificial Intelligence in the University of Edinburgh, 2003, http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm

[9] http://homepages.inf.ed.ac.uk/rbf/HIPR2/images/sff1hou1.gif

[10] Wolfgang Förstner, Joachim M. Buhmann, Annett Faber, Petko Faber (Eds.): Mustererkennung 1999, 21. DAGM-Symposium, Bonn, 15.-17. September 1999, Proceedings. Informatik Aktuell Springer 1999, ISBN 3-540-66381-9, http://www.informatik.uni-trier.de/~ley/db/conf/dagm/

[11] Matteo Frigo and Steven G. Johnson „The Fastest Fourier Transform in the West", http://www.fftw.org/

[12] J. R. Parker „Algorithms for Image Processing and Computer Vision", Wiley Computer Publishing, 1997

[13] Rafael C. Gonzalez, Richard E. Woods „Digital Image Processing", Addison-Wesley, 1992/2000

[14] Günther Görz (Hrsg.) „Einführung in die künstliche Intelligenz", Addison-Wesley, 1995, S. 300f

[15] S. Harnad „The Symbol Grounding Problem", 1990, Physica D 42: 335-346 http://www.ecs.soton.ac.uk/~harnad/Papers/Harnad/harnad90.sgproblem.html

[16] Douglas R. Hofstadter „Metamagicum", Basic Books, New York, 1985, Ausgabe: dtv/Klett-Cotta,1994, S. 701ff

[17] Mariusz Jankowski, Jens-Peer Kuska „Connected Components Labeling - algorithms in Mathematica, Java, C++ and C#", http://www.ims2004.com/ (siehe CD)

[18] Ramiro Jordán, Roberto Lotufo, http://www.dca.fee.unicamp.br/dipcourse/html-dip/c6/s2/front-page.html in „Digital Image Processing (DIP) with Khoros 2", 1997

[19] Dieter Koller „Model-Based Object Tracking in Road Traffic Scenes", 1996, http://www.vision.caltech.edu/koller/ModelTracking.html

[20] Danica Kragic, 2002, http://www.nada.kth.se/~danik/articles/iros02/node3.html

[21] B.D. Lucas, T. Kanade „An iterative image registration technique with an application to stereo vision" In Proc. Int. Joint Conference on Artificial Intelligence, 1981, S. 674-679

[22] I. Niemelä and P. Simons „An implementation of the stable model semantics for logic programs", http://www.tcs.hut.fi/Software/smodels/

[23] Jean Piaget „Meine Theorie der geistigen Entwicklung", Beltz, 2003

[24] Frederic Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, David Salesin „Synthesizing Realistic Faces", http://grail.cs.washington.edu/projects/realface/tracking.html

[25] I. Pitas „Digital image processing algorithms and applications", Wiley, 2000

[26] Amrudee Sukpan „A Survey on Constraint Satisfaction Problems", http://www2.cs.uregina.ca/~sukpan1a/csp/csp.htm#2.1.1%20Generate-and-Test

[27] Greg Welch „The Kalman Filter", http://www.cs.unc.edu/~welch/kalman/

[28] I.T. Young, J.J. Gerbrands, L.J. van Vliet „Image Processing Fundamentals", 1995-2003, http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-Convolut-2.html

[29] s.o., http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-.html

[30] http://de.wikipedia.org/wiki/Diskrete_Fourier-Transformation

[31] „The Free Online Dictionary of Computing", Version vom 09. Feb 2002, Schlüsselwort "heuristic"

[32] http://www.mathe-schule.de/Physik/links_kreisel.htm

[33] http://www.teachsam.de/psy/psy_kog/lernth/wiss/wiss_2_2_1_2.htm

[34] http://mplayerhq.hu/, erfordert zusätzlichen Patch zur Ausgabe von Video auf ein video4Linux-Device

[35] Starbreeze Animation Studio, `http://www.starbreeze.com/sbz/EN_PlainPage.asp?id=2198`

[36] „Intel® Open Source Computer Vision Library”, `http://www.sourceforge.net/projects/opencvlibrary`. Version beta3.1 wurde in dieser Arbeit verwendet.

[37] „Open Source Computer Vision Library Reference Manual”, 2001, Intel Corporation, docs/opencvman_old.pdf im[36]-Paket

[38] http://motion.sourceforge.net/vloopback/index.html

[39] „Webster’s Revised Unabridged Dictionary Version”, C. & G. Merriam Co. Springfield, Mass., 1913, `http://dict.org/`

[40] Wikipedia, `http://de.wikipedia.org/wiki/Faltung_(Mathematik)`

[41] Wikipedia, `http://de.wikipedia.org/wiki/Menon`

[42] Wikipedia, `http://de.wikipedia.org/wiki/Messung`

[43] Wikipedia, `http://de.wikipedia.org/wiki/Modell`

[44] Wikipedia, `http://de.wikipedia.org/wiki/Wahrnehmung`

[45] Alan Cox „Video4Linux API Specification”. Teil der Linux-Kernelquellen unter `Documentation/video4linux`.

Anhang A

Erklärung

Ich versichere, meine Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Datum:

Unterschrift:

Anhang B

Danksagung

Besonderer Dank geht an Dr. J.-P. Kuska für den Hinweis auf die Bildkorrelation und den Floodfill-Algorithmus und die Beharrlichkeit bei der Klärung eines Verständnisproblems mit der Fourier-Transformation. An zweiter Stelle obwohl zeitlich weitaus aufwendiger möchte ich Rene Liebscher meinen Dank aussprechen für seine Idee zur Aufgabenstellung sowie die vielen Hinweise und Verbesserungsmöglichkeiten im Laufe der Arbeit. Prof. Der danke ich für die gegebene Freiheit bei der Bearbeitung des Themas.

Ebenso danken möchte ich meiner Familie dafür, mir mein Studium ermöglicht zu haben. Danken möchte ich auch meinen Freunden, die mir durch ihre Hinweise geholfen haben, mich verständlicher und klarer auszudrücken. Dank auch an Pyndap für ihre Geduld.

Anhang C

Kontakt

Michael Bunk
Hohle Gasse 23
04159 Leipzig
micha@luetzschena.de, micha137@users.sourceforge.net

Dieses Dokument, der Quelltext, die Online-Dokumentation und die Videos sind unter folgender Adresse zu finden:

`http://robot.informatik.uni-leipzig.de/~mbunk`