Fakultät Informationstechnik

Studiengang Softwaretechnik und Medieninformatik

Bachelorarbeit

Umsetzung einer Optischen Flusserkennung

in einem mobilen Zigbee- Videostream

Kurzbeschreibung

Diese Bachelorarbeit beschäftigt sich mit der Konzeption und Umsetzung eines Sensors, zur Erfassung des Lagezustands über den optischen Fluss und Berechnung daraus resultierender Regelgrößen. Sie bildet damit die Grundlage und Wissensbasis zur Stabilisierung des Flugverhaltens, in erster Linie des Schwebefluges eines Flugapparates. Zur Auswertung der von der Kamera gelieferten Bilder kommt die OpenCV-Bibliothek zum Einsatz, in Folgendem werden alle ihre Möglichkeiten einen Bewegungsfluss zu erkennen aufgezeigt und bewertet. Der Transport der Bilder zu dem stationären Rechenserver wird mit dem ZigBee Funknetz-Standart realisiert, es wird auf seine Möglichkeiten und Grenzen hingewiesen und Vorschläge zur Optimierung seiner Funktionalität gebracht. Neben der Einführung die einem die Grundlagen der einzelnen Technologien näher bringt, beschäftigt sich der Hauptteil der Arbeit mit der Realisierung, Modifikation und Erweiterung der einzelnen Segmente des Systems. Dazu gehört unter anderem die Kommunikation zwischen dem Flugapparat und dem Rechenserver, das Framework der Flusserkennung und das bereitstellen der der Steuerdaten für den Regler. Ein Einblick in die zukünftigen Einsatz- und Erweiterungsmöglichkeiten des Systems rundet die Arbeit ab.

Inhaltsverzeichniss

Kurzbeschreibung 2

Inhaltsverzeichniss 3

1. Grundlagen 4

1.1. Optischer Fluss 4

1.1.1. Bewegung zur Grauwertveränderung 6

1.1.2. Perspektivisches Bewegungsmodell 7

1.2. Erkennen des optischen Flusses 10

1.2.1. Korrelationsbasierte Verfahren (Block Matching) 11

1.2.2. Gradientenverfahren 12

1.2.3. Das Blendenproblem (Aperturproblem) 12

1.2.4. Korrespondenzproblem 13

1.3. Kalman-Filter 15

1.3.1. Dynamical motion 17

1.3.2. Controll motion 17

1.3.3. Random motion 18

1.4. OpenCV 18

1.4.1. Aufbau von OpenCV 19

1.4.2. Installation von OpenCV unter Windows 20

2. Literaturverzeichnis 21

1. Grundlagen

1.1. Optischer Fluss

Als Definition für den optischen Fluss liefert die freie Onlineenzyklopädie, Wikipedia folgende Beschreibung:

Als Optischer Fluss (eng. Optical Flow) wird in der Bildverarbeitung und in der optischen Messtechnik ein Vektorfeld bezeichnet, das die Bewegungsrichtung und -Geschwindigkeit für jeden Bildpunkt einer Bildsequenz angibt. Der Optische Fluss kann als die auf die Bildebene projizierten Geschwindigkeitsvektoren von sichtbaren Objekten verstanden werden.

Optischer Fluss ist ein Phänomen, das uns stets in alltäglichen Leben begleitet. Es ist nichts Anderes als die Bewegung unserer Umwelt, relativ zu unseren Augen, die wir wahrnehmen, wenn wir beispielsweise durch die Stadt laufen, oder mit dem Auto unterwegs sind. Blicken wir aus dem Fenster eines Fahrenden Zuges, so sehen wir verschiedenste Landschaften, ob Gebäude in der Stadt, oder Bäume und Felder auf dem Land, egal ob überfüllt von Menschenmassen oder Tieren, oder scheinbar leer und verlassen, alle diese Objekte rasen in einer scheinbar, fast homogenen Masse an uns vorbei. Diese Bewegung ist der besagte optische Fluss. Zu den Funktionen des optischen Flusses, gehört nicht nur das Erfassen von Bewegungsrichtungen, sondern auch das Erkennen von Entfernungen, so bewegen sich zum Beispiel große, weit entfernte Objekte wie Wolken oder Hochhäuser, in Relation zu näheren und kleineren Objekten wie Bäumen, nur sehr langsam.

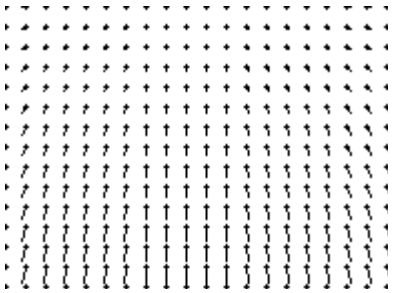

Abbildung 1-1: Vektorfeld für die Vorwertsbewegung

Es gibt eindeutige, mathematische Beziehungen Zwischen der Entfernung zum betrachteten Objekt und dem optischen Fluss. Fahren Sie die doppelte Geschwindigkeit so verdoppelt sich auch die Geschwindigkeit des Flusses. Platziert man jetzt das Objekt in einer doppelten Entfernung, so halbiert sich die sichtbare Geschwindigkeit wieder. Natürlich gibt es auch einen Zusammenhang zwischen dem optischen Fluss und dem Betrachtungswinkel relativ zur Bewegungsrichtung. Der Optische Fluss ist am größten, wenn das betrachtete Objekt sich auf Ihrer Höhe befindet, seitlich, ober- oder unterhalb von Ihnen, also Blickrichtung 90° zur Bewegungsrichtung. Vor oder hinter der 90° Marke nimmt der erkennbare optische Fluss ab.

Platziert man das betrachtete Objekt aber direkt in der Bewegungsrichtung, also vor ihnen, so wird man keinen optischen Fluss erkennen, wir merken, direkt auf der Bewegungslinie ist der optische Fluss gleich 0. Allerdings werden die Kanten eines realen Objekts nie direkt auf der Linie der Bewegung liegen und haben so mit, einen minimalen optischen Fluss, den wir als das Größerwerden des Objekts wahrnehmen können.

Folgende Abbildung veranschaulicht die Verteilung des optischen Flusses bei unterschiedlicher Bewegung. Bei einer gradlinigen Bewegung in eine bestimmte Richtung verteilt sich der optische Fluss, wie oben bereist beschrieben, direkt auf der Bewegungsrichtung ist kein Fluss zu erkennen, auf der Höhe des Betrachters sehen wir den größten optischen Fluss, mit entgegen gesetzter Flussrichtung zur Bewegung des Betrachters. Anders bei einer Drehbewegung, dort sieht man auf den ersten Blick einen konstanten optischen Fluss, auf der zweidimensionalen Abbildung erkennt man allerdings nur die Pole des optischen Flusses. Die Nullpunkte, die sich bei einer gradlinigen Bewegung auf der Bewegungsgeraden befinden, sind bei einer Drehung, ober- und unterhalb der Drehfläche, was einem mehr oder weniger logisch erscheint, wenn man sich an die Rechter-Daumen-Regel oder die Umfassungsregel aus dem Physikunterricht erinnert. Mit der Umfassungsregel wird aus der Drehbewegung wieder eine Gerade, die senkrecht zur Drehfläche steht, als Drehachse bezeichnet werden kann und wie bei gradliniger Bewegung die Nullpunkte des optischen Flusses markiert.

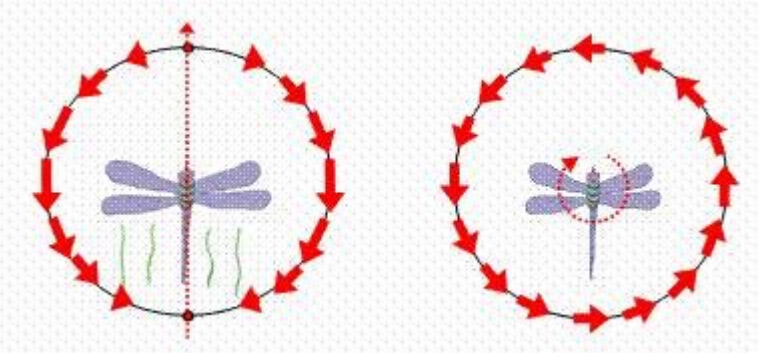

Abbildung 1-2: Charakteristische Vektorbilder für eine Vorwertsbewegung und eine Drehung

Die Eigenschaften der Drehbewegung werden eindeutiger wenn man die nächste Abbildung betrachtet. Es handelt sich dabei um jeweils eine Kugel, im Raum um den Betrachter, die in der Fläche aufgerollt wurde, Pfeile repräsentieren hier mit ihrer Länge, die Geschwindigkeit des optischen Flusses, wobei die Punkte oben und unten, immer einen und den selben Punkt meinen, nämlich den oberen und den unteren Punkt auf der Drehachse.

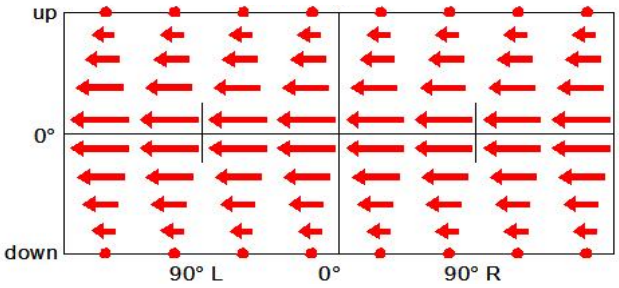

Abbildung 1-3: Aufgeklappte Vektorkugel für die Drehung nach rechts

Das nächste Bild zeigt eine solche Kugel, in einer gradlinigen Bewegung. Der Punkt in der Mitte markiert hier die Bewegungsrichtung, mit den Vektoren des optischen Flusses, die von der Mitte ausgehen, zur 90°-Marke immer größer werden, weiter hinten wieder an Länge verlieren und in der 180°-Marke in einem Punkt münden.

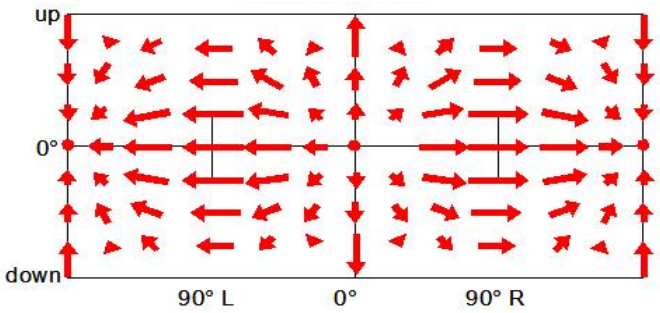

Abbildung 1-4: Aufgeklappte Vektorkugel für die Vorwertsbewegung

1.1.1. Bewegung zur Grauwertveränderung

In der Literatur zur Bildverarbeitung liest man oft den Begriff der Grauwertveränderung/ Grauwertänderung. Der Grauwert ist der Helligkeits- oder Intensitätswert eines Pixels, ohne die Berücksichtigung der Farben. In einem RGB-Bild könnte die Bestimmung des Grauwerts folgendermaßen aussehen:

$$\text{Grauwert} = 0{,}5 \cdot \text{Rot} + 0{,}3 \cdot \text{Grün} + 0{,}2 \cdot \text{Blau}$$

Anhand der Bewegung eben dieser Grauwerte lässt sich der optische Fluss berechnen. Was dabei aber fälschlicherweise, angenommen werden kann, ist die falsche Tatsache, dass die Grauwertveränderung mit der Bewegung in der Szene gleich zu setzen ist. Nun spielen in der Grauwertveränderung die Beleuchtung und die Objekte, eine gleich große Rolle wie die Bewegung selbst. Angenommen eine Kugel mit einer sehr glatten und gleichmäßigen Oberfläche dreht sich vor der Kamera. Die Grauwerte wären für den Betrachter regungslos. Andererseits, würde man eine solche Kugel von einer sich bewegenden Lichtquelle beleuchten lassen, könnte man anhand des bewegten Schattens, zu einer falschen Erkenntnis kommen, dass die Kugel sich bewegt.

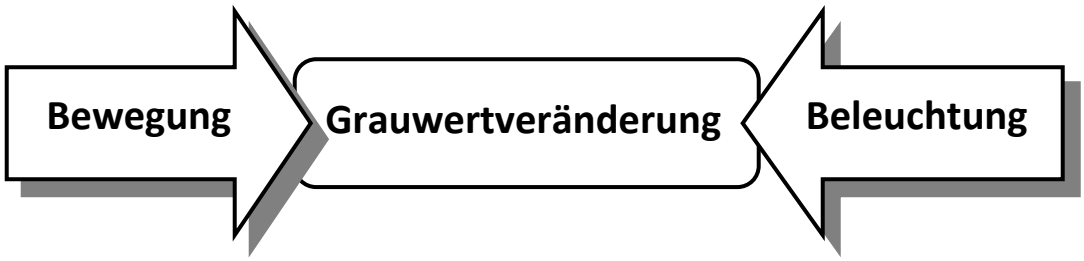

Abbildung 1-5: Zusammenstellung der Grauwertveränderung

So kann man resultierend sagen, dass die Grauwertveränderung ein Ergebnis aus der Verbindung der Bewegung und der Beleuchtung ist, welches seine Uhrsprungskomponente nicht immer eindeutig widerspiegelt.

1.1.2. Perspektivisches Bewegungsmodell

Mit dem Perspektivischen Bewegungsmodell lassen sich Geschwindigkeiten der Szenenpunkte, relativ zur Kamera, als Beobachter, ziemlich anschaulich berechnen. Das Perspektivische Bewegungsmodell beruht auf der dreidimensionalen Starkörperbewegung mit sechs Freiheitsgraden. Zu einem sind es drei Bewegungsrichtungen der Translation mit

$$\vec{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Formel 1-1

und drei weitere Bewegungsrichtungen der Rotation

$$\vec{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

Formel 1-2

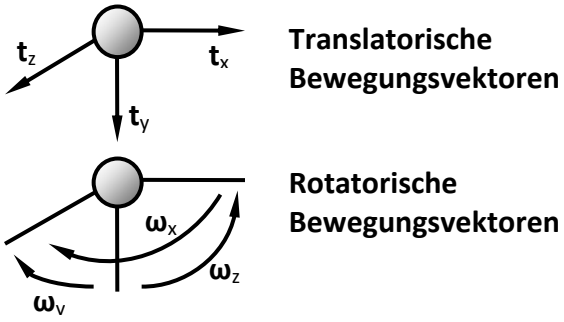

Abbildung 1-6: Freiheitsgrade des Perspektivischen Bewegungsmodells

Ausgegangen von den sechs Freiheitsgraden und den Koordinaten des Szenepunktes $\vec{P}$, mit

$$\vec{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Formel 1-3

läst sich für die Geschwindigkeit eines Punktes der Szene folgende Gleichung aufstellen:

$$\dot{\vec{P}} = -\vec{t} - \vec{\omega} \times \vec{P} = \begin{bmatrix} -t_x - \omega_y \cdot Z + \omega_z \cdot Y \\ -t_y - \omega_z \cdot X + \omega_x \cdot Z \\ -t_z - \omega_x \cdot Y + \omega_y \cdot X \end{bmatrix}$$

Formel 1-4 [3]

Die vielen Minuszeichen kommen daher, da es von der Bewegung der Kamera ausgegangen wird, Bewegungen der Objekte in der Szene wirken dem genau entgegen.

Um die Geschwindigkeit eines Punktes zu bestimmen, sind seine Koordinaten nach der Zeit abzuleiten. Dies kann auf einzelne Komponenten der Bewegungsrichtung explizit angewandt werden. Hier die x-Koordinate, mit

$$x = \frac{f}{Z} \cdot X$$

Formel 1-5

wo f die Brennweite der Kamera und Z den Abstand des Punktes zur Linse darstellen. Abgeleitet nach der Zeit sieht die Gleichung so aus:

$$\frac{dx}{dt} = f \cdot \left(\frac{\partial x}{\partial Z} \cdot \frac{\partial Z}{\partial t} + \frac{\partial x}{\partial X} \cdot \frac{\partial X}{\partial t} \right)$$

Formel 1-6

oder

$$\dot{x} = f \cdot \left(\frac{X}{Z^2} \cdot \dot{Z} + \frac{1}{Z} \cdot \dot{X} \right)$$

Formel 1-7

und für y entsprechend

$$\dot{y} = f \cdot \left(\frac{Y}{Z^2} \cdot \dot{Z} + \frac{1}{Z} \cdot \dot{Y} \right)$$

Formel 1-8

Durch Einsetzen der Formel 1-7 und Formel 1-8 in die Formel 1-4 [3], so erhält man die Formel für das Geschwindigkeitsfeld $\dot{\vec{p}}$:

$$\dot{\vec{p}} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{Z} \cdot \begin{bmatrix} t_z \cdot x - f \cdot t_x \\ t_z \cdot y - f \cdot t_y \end{bmatrix} + \omega_x \cdot \begin{bmatrix} \frac{1}{f} \cdot x \cdot y \\ \frac{1}{f} \cdot y^2 + f \end{bmatrix} - \omega_y \cdot \begin{bmatrix} \frac{1}{f} \cdot x^2 + f \\ \frac{1}{f} \cdot x \cdot y \end{bmatrix} - \omega_z \cdot \begin{bmatrix} -y \\ x \end{bmatrix}$$

Formel 1-9

Diese Darstellung lässt sich durch Normierung weiter vereinfachen. Man setze die Brennweite f=1 und substituiere mit

$$A = \begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \end{bmatrix}$$

und

$$B = \begin{bmatrix} x \cdot y & -x^2 - 1 & y \\ y^2 + 1 & -x \cdot y & -x \end{bmatrix}$$

so erhält man für das Perspektivische Bewegungsmodell, folgende Schreibweise:

$$\dot{p} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = A \cdot \frac{t}{Z} + B \cdot \omega = \begin{bmatrix} \frac{1}{Z} \cdot A & B \end{bmatrix} \cdot \begin{bmatrix} t \\ \omega \end{bmatrix}$$

Formel 1-10 [3]

Man merkt die Abhängigkeit der Geschwindigkeitsverteilung, von der Entfernung Z des Szenenpunktes bis zur linse der Kamera. Es bestätigt sich das Phänomen, dass weit entfernte Objekte sich langsamer zu bewegen scheinen, als näher gelegene. Dieser Zusammenhang Z(p) wird als Tiefenfunktion bezeichnet.

1.2. Erkennen des optischen Flusses

Die Erfassung des Optischen Flusses ist eine Disziplin der künstlichen Intelligenz. Eine Videokamera liefert ca. 30 Bilder in der Sekunde, der Unterschied zwischen den einzelnen Frames stellt hierbei die wichtigste Informationsquelle dar. Bewegt man die Kamera relativ zur Umgebung, so erkennt man am Bildschirm eine dementsprechende Bewegung, den zuvor erwähnten optischen Fluss. Den optischen Fluss stellt man oft als einen Vektorfeld dar, dabei beschreibt Länge und die Richtung jedes Vektors $\vec{V}(x_i, y_i)$ die derzeitige Bewegung, des Pixels $P(x_i, y_i)$. In der Abbildung 1-7: Vektorfelder des Zooms, der Drehung und ihrer Kombination sieht man einige mögliche Vektorfelder und die dazugehörigen Transformationen, Zoom, Drehung und die Kombination as den beiden.

Abbildung 1-7: Vektorfelder des Zooms, der Drehung und ihrer Kombination

Eine Bildrate von 30 Bilder/Sekunde entspricht einem Zeitintervall von $\frac{1}{30}$ Sekunde, zwischen zwei aufeinander folgenden Bildern. Es wird stets versucht die Intervalle so klein wie Möglich zu halten, damit die Bewegung der Bildinhalte nur wenige Pixel ausmacht[1]. Andererseits muss die Zeit, die dem Algorithmus bleibt, um den Fluss der Grauwerte zu berechnen, groß genug sein, um beim nächsten Frame das Ergebnis vorzulegen und mit der nächsten Berechnung fortzufahren.

1.2.1. Korrelationsbasierte Verfahren (Block Matching)

Die korrelationsbasierte Verfahren zur Erfassung des optischen Flusses, versuchen in zwei auf einander folgenden Bildern Bereiche zu finden, die einander eindeutig zuordenbar sind. Dabei vergleichen sie Muster, mittels eines Ähnlichkeitsmaßes, aus hellen und dunkeln Pixeln, in lokalen Bereichen der Bilder (Blöcke). Die zuvor definierten Blöcke müssen so gewählt werden, dass der Algorithmus bis zum nächsten Bildpaar das Ergebnis vorlegen kann, das heißt möglichst kleine Abschnitte mit wenigen Pixeln. Andererseits dürfen die Blöcke auch nicht zu klein gewählt werden, da sonst die Gefahr besteht, dass das Suchmuster sich zum Zeitpunkt in dem das Zweite Bild gemacht wird, schon außerhalb des Blocks befindet und nicht gefunden werden kann.

Da die verfolgten Objekte, an den der optische Fluss bestimmt werden soll, meist größer sind als die Blöcke, tritt beim Suchen der Ähnlichkeiten meist das Aperturproblem auf, da die Objekte nur an ihren Rändern eindeutige Bewegungen erkennen lassen. Dieses Problem wird dadurch gelöst, dass man einfach annimmt, dass die Blöcke die keine eindeutig zu erkennende Muster enthalten, zu einem Objekt mit daneben liegenden Blöcken zusammen gefasst werden können, weiterhin wird angenommen, dass alle Blöcke die zu einem Objekt gehören, gleiche Geschwindigkeiten besitzen, also in einem homogenen Vektorfeld liegen, in dem alle Vektoren die selbe Richtung und Länge haben.

Dazu existieren physikalische Verfahren, zur besseren Vorstellung der Problematik [4], dabei stellt man sich zwischen den benachbarten Blöcken gespannte Federn vor, deren Federkonstante jeweils davon abhängig ist, wie ähnlich die Blöcke sich sind. Zwei ähnlich aussehende Blöcke haben demnach eine Verbindungsfeder mit einer großen Federkonstante, Blöcke die sich stark unterscheiden, oder zwischen denen sich eine Kante befindet, eine mit einer kleinen Federkonstante.

[1] Die meisten Algorithmen zur Bestimmung des optischen Flusses, basieren auf der Annahme, dass die einzelnen Bewegungen im Bild nie eine große Distanz zurücklegen, somit sucht man ein bestimmtes Muster nicht in der gesamten Bildebene, sondern nur in der unmittelbaren Umgebung des gesuchten Grauwerts. Das heißt, umso kleiner der Zeitliche Abstand zweier auf einander folgender Bilder, desto kleiner kann der Suchbereich eingestellt werden, ohne den Verlust der Wahrscheinlichkeit den Vektor richtig zu bestimmen.

Das Ähnlichkeitsmaß ist in aller Regel, eine Korrelation zwischen zwei Folgebildern. Es gibt verschiedene Methoden um die Ähnlichkeit zweier Blöcke zu bestimmen.

- SAD: Summe der Differenzen der Absolutbeträge („Sum of Absolut Differences“)

$$SAD = \sum_{r \in N} \left| I'(p' + r) - I(p + r) \right|$$

- SSD: Summe der quadratischen Differenzen der Absolutbeträge („Sum of Squared Differences“)

$$SSD = \sum_{r \in N} \left(I'(p' + r) - I(p + r) \right)^2$$

- NCC: Normalisierte Kreuzkorrelation („Normalized Cross Correlation“)

$$NCC = \frac{\sum_{r \in N} I'(p' + r) \cdot I(p + r)}{\sqrt{\sum_{r \in N} I'^2(p' + r)} \cdot \sqrt{\sum_{r \in N} I'^2(p + r)}}$$

…..

1.2.2. Gradientenverfahren

1.2.3. Das Blendenproblem (Aperturproblem)

Die Analyse des optischen Flusses beruht also auf räumlichen und zeitlichen Grauwertänderungen. Diese Grauwerte werden innerhalb bestimmter Radien, in zwei aufeinander folgenden Bildern wieder erkannt und einander zugeordnet. Nicht immer sind markante Merkmale dabei die sich sofort zuordnen lassen, doch auch wenn es der Fall ist, kann das Ergebnis mehrdeutig werden. Ein Solches Problem bezeichnet man als das Blendenproblem. Es taucht dann auf, wenn zwar ein Abschnitt des Suchfensters/Maske, Grauwertmäßig sich eindeutig vom Rest der Pixel unterscheidet, doch keine fassbaren Merkmale bittet, an hand derer die Bewegung eindeutig erkannt werden kann, weil das Objekt, zum Beispiel viel größer als das Suchfenster ist.

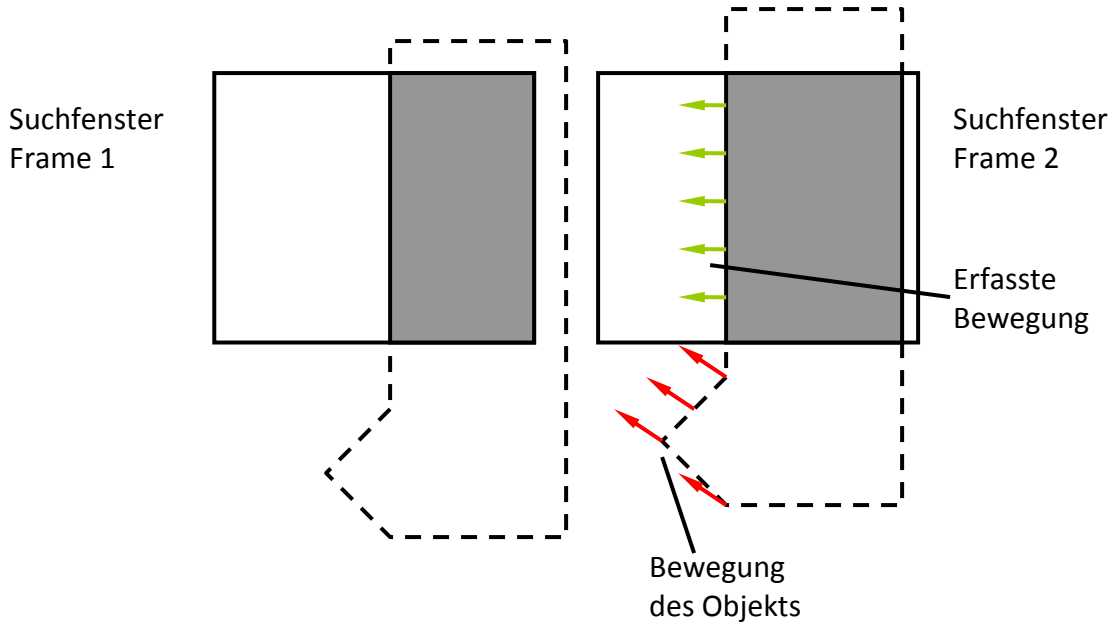

Abbildung 1-8: Veranschaulichung des Blendenproblems

Wir können lediglich die senkrecht zur Kante liegende Komponente des Verschiebungsvektors bestimmen, während die parallel zur Kante liegende unbekannt bleibt. Diese Mehrdeutigkeit (das Blendenproblem) kann z.B. aufgelöst werden, wenn die Operatormaske die Ecke eines Objektes einschließt.

1.2.4. Korrespondenzproblem

Das Korrespondenzproblem beschreibt alle arten des Informationsverlustes einer Bewegung, das zuvor erwähnte Blendenproblem ist ein Spezialfall des Korrespondenzproblems. Das Korrespondenzproblem liegt dann vor, wenn wir keine eindeutig mit einander korrelierender Punkte, in zwei aufeinander folgenden Bildern einer Sequenz, bestimmen können. In folgendem sind einige Beispiele des Korrespondenzproblems aufgeführt.

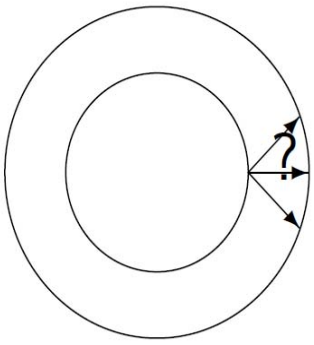

Abbildung 1-9: Korrespondenzproblem 1

Es fehlen eindeutige Merkmale, deren Bewegung man Verfolgen könnte.

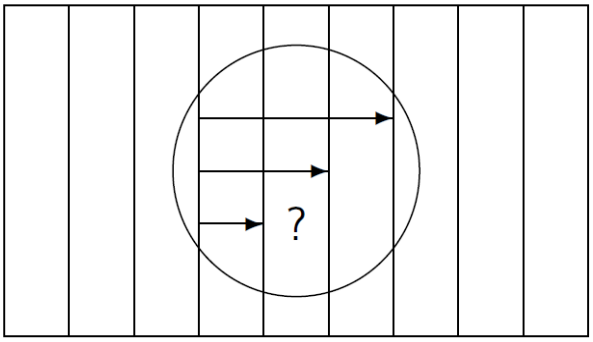

Abbildung 1-10: Korrespondenzproblem 2

Bei Periodischen Texturen, wie Netz oder Gitter, kann die Bewegung nie eindeutig bestimmt werden, solange man nur ein abschnitt des Objekts betrachtet und die Bewegung ein Vielfaches der Maschenweite ist. Man erkennt die Bewegung erst dann eindeutig, wenn man den Rand des Gitters im Suchfenster hat.

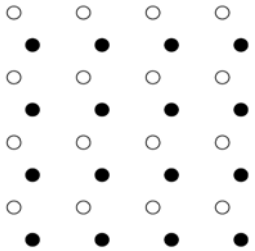

Abbildung 1-11: Korrespondenzproblem 3

Bei Bildern mit vielen Objekten, die eine sehr ähnliche Form haben, kann das korrespondierende Teilchen im nächsten Bild nicht ermittelt werden.

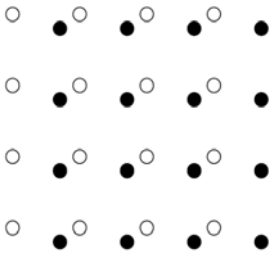

Abbildung 1-12: Lösung durch Verkürzen des Zeitinterwals

Eine Lösung für Solche Probleme wäre das Verkürzen des Zeitinterwals, man würde einen solchen Interwal einstellen, bei dem man sich sicher sein kann, dass in der Zeit ein Teilchen nur einen kleinen Teil des mittleren Teilchenabstandes überbrücken kann.

Um den optischen Fluss erfassen und messen zu können, ist es zwingend notwendig, sich entsprechende Punkte in einem, zu einer bestimmten Zeit gemachten Bild und in dem darauf folgendem Bild zu Finden. Diese Punkte werden Features genannt und sind meist leicht erkennbare, kontrastreiche Bildabschnitte. Es wird versucht, ein und denselben Feature in beiden Bildern zu finden. Dabei spielt das Faktum eine große Rolle, das die Bereiche des Bildes die sich um den jeweiligen Feature befinden, ähnliche Intensitäten aufweisen. Man sucht also nicht einzelne Pixel, sondern größere Pixelfelder, was die Arbeit wesentlich erleichtert. Sucht man einen bestimmten Bereich, mit dem Zentrum im Pixel $P(x_0, y_0)$*, zur Zeit* t_0*, muss man es mit anderen Bildbereichen vergleichen, deren Zentren, potenzielle Trefferpunkte sind, mit den Koordinaten* $(x_0 + \Delta x, y_0 + \Delta y)$ *zur Zeit* $t_0 + \Delta t$ *. Ein gutes Kriterium für die Ähnlichkeit, ist die Summe der Quadratischen Differenzen (Sum of Squared Differences)*

$$SSD(\Delta x, \Delta y) = \sum_{(x,y)} (I(x, y, t) - I(x + \Delta x, y + \Delta y, t + \Delta t))^2$$

Die Koordinaten x und y kriegen ihre Werte aus dem Bereich, mit den Zentrum im $P(x_0, y_0)$*. Gesucht sind Werte für* $(\Delta x, \Delta y)$*, die das kleinste Ergebnis für das SSD liefern. Dann gilt für den Geschwindigkeitsvektor im Punkt* $P(x_0, y_0)$*:*

$$\vec{V}(x_0, y_0) = (v_x, v_y) = (\Delta x / \Delta t, \Delta y / \Delta t)$$

1.3. Kalman-Filter

Um möglichst genaue Messergebnisse vorlegen zu können, bedarf es eines Algorithmus der ohne viel Rechenaufwand, das Rauschen eines Signals heraus filtert und mit Statistischen Methoden an den tatsächlichen Wert möglichst nahe herankommt. Für eine solche Aufgabe bittet sich der, heute wohl am weitesten verbreitete Algorithmus zur Zustandsschätzung linearer und nichtlinearer Systeme an, der Kalman-Filter.

Der Kalman-Filter ist ein nach seinem Entdecker Rudolf E. Kálmán benannter Satz von mathematischen Gleichungen. Obgleich die Benennung des Filters nach Rudolf E. Kálmán erfolgte, wurden bereits zuvor nahezu identische Verfahren durch Thorvald N. Thiele und Peter Swerling veröffentlicht. Auch existierten zur selben Zeit bereits allgemeinere, nichtlineare Filter von Ruslan L. Stratonovich, die das Kalman-Filter und weitere lineare Filter als Spezialfälle

enthalten. Ebenso erwähnenswert sind Vorarbeiten und gemeinsame Publikationen von Kálmán mit Richard S. Bucy, insbesondere für den Fall zeitkontinuierlicher dynamischer Systeme. Daher wird häufig die Bezeichnung Kalman-Bucy-Filter und gelegentlich auch Stratonovich-Kalman-Bucy-Filter in der Fachliteratur benutzt.

Der erste erfolgreiche Einsatz des Filters erfolgte in Echtzeitnavigations- und Leitsystemen, die im Rahmen des Apollo-Programms der NASA unter Federführung von Stanley F. Schmidt entwickelt wurden und in der Navigation und Steuerung der Raumkapsel verbaut wurden.

Mithilfe dieses Filters sind bei Vorliegen lediglich fehlerbehafteter Beobachtungen Rückschlüsse auf den exakten Zustand von vielen der Technologien, Wissenschaften und des Managements zugeordneten Systemen möglich. Die Kernaussage ist: „Ist-Wert des Systems muss gleich Soll-Wert zu einer bestimmten Zeit sein.“ Vereinfacht dient das Kalman-Filter zum Entfernen der von den Messgeräten verursachten Störungen. Dabei müssen sowohl die mathematische Struktur des zugrundeliegenden dynamischen Systems als auch die der Messverfälschungen bekannt sein.

Eine Besonderheit des 1960 von Kálmán vorgestellten Filters bildet seine spezielle mathematische Struktur, die den Einsatz in Echtzeitsystemen verschiedenster technischer Bereiche ermöglicht. Dazu zählen u.a. die Auswertung von Radarsignalen zur Positionsverfolgung von sich bewegenden Objekten (Tracking) aber auch der Einsatz in elektronischen Regelkreisen allgegenwärtiger Kommunikationssysteme wie etwa Radio und Computer.

Mittlerweile existiert eine große Bandbreite von Kalman-Filtern für die unterschiedlichsten Anwendungsgebiete.

Im Gegensatz zu den klassischen FIR- und IIR-Filtern der Signal- und Zeitreihenanalyse basiert das Kalman-Filter auf einer Zustandsraummodellierung, bei der explizit zwischen der Dynamik des Systemzustands und dem Prozess seiner Messung unterschieden wird.

Bevor man einen Kalman-Filter erfolgreich einsetzen kann, sind vom System einige notwendige Bedienungen zu erfühlen, diese wären:

- das System ist linear,
- das Störsignal ist ein weises Rauschen und
- das Störsignal ist gaußscher Natur.

die Erste bedeutet, dass jeder Zustand des Systems zum Zeitpunkt k, ähnlich wie eine Markow-Kette erster Ordnung, nur vom Zustand k-1 abhängt, also aus der Multiplikation einiger Matrizen mit dem Zustand k-1 ermittelt werden kann.

$$X_k = F_{k-1} X_{k-1} + B_{k-1} u_{k-1} + v_{k-1}$$

Die Anderen zwei sagen aus, dass das Rauschen in unserer Messung nicht mit der Zeit korrelieren darf, also Zeitunabhängig ist und mit Zwei Werten exakt beschrieben werden kann, nämlich mit einem Durchschnittswert und einer Kovarianz.

$$v_k \approx WN(\bar{x}, Q_k)$$

Kurz zusammengefasst, erhöhen wir die Genauigkeit der Messwerte nach der eigentlichen Messung, mit Hilfe der vorangegangenen Ergebnisse. Wir berechnen dafür den aktuellen Zustand des Systems unter Berücksichtigung der aktuellen Messwerte, und der Werte von vorherigen Messungen und der dazugehörigen Wahrscheinlichkeiten und Varianzen. Somit bittet der Kalman-Filter eine gute Möglichkeit entweder von mehreren Quellen, oder von einer Quelle zur unterschiedlichen Zeiten erfasste Werte, mit einander zu kombinieren, um zu einem statistisch genaueren Ergebnis zu kommen.

Bis Jetzt war die Rede von Messungen eines ruhenden Systems, d.h. die einzelnen Messwerte wären im Idealfall genau gleich, was ist aber wenn sich das Objekt dessen Zustand erfasst wird, zwischen der ersten und der zweiten Messung bewegt. Um diesen Effekt im griff zu bekommen müssen wir in der so genannten „prediction phase“ eine Vorhersage treffen. In der „prediction phase“ verwenden wir alles was wir über dass System wissen um ihren Zustand zur Zeit der nächsten Messung zu ermitteln. Nehmen wir an unser Quadrokopter bewegt sich mit einer konstanten Geschwindigkeit, so können wir die einzelnen Messwerte (Lage im Raum) nicht einfach auf einander beziehen, doch mit der Kenntnis des Systems und des Zustands in dem es sich zur Zeit *t* befindet, können wir bestimmen wie sich der Ort des Quadrokopters bis zur zweiten Messung zur Zeit *t + dt* ändert. So vergleichen wir die Messung zur Zeit *t + dt* nicht einfach mit dem alten Zustand, sondern mit dem alten Zustand projiziert in die Zeit *t +dt.* Dabei unterscheidet kalmansche Lehre drei grundlegende Bewegungsarten.

1.3.1. Dynamical motion

Dynamical motion oder die Dynamische Bewegung, ist eine Bewegungsart bei der wir die Annahme treffen, dass das System eine gleich bleibende und seit der letzten Messung nicht veränderte Bewegung ausführt. Es sei, zum Beispiel x, die zu letzt gemessene Position des Systems zum Zeitpunkt t und v die aktuelle Geschwindigkeit, dann resultiert daraus, dass zur Zeit $\mathrm{t} + \mathrm{dt}$ das System sich an der Position $\mathrm{x} + \mathrm{v} * \mathrm{dt}$ befinden wird und sich mit der Geschwindigkeit v weiter bewegt.

1.3.2. Controll motion

Controll motion, eine Kontrollierte Bewegung. Wie der Name schon vermuten läst, ist es eine Bewegungsart die wir vorwiegend bei Systemen anwenden, die von Außen gestört werden und geregelt werden müssen. Das bedeuten anders ausgeduckt, wir kriegen eine Reaktion des Systems auf eine beliebige

Bewegungsart die wir selber initiieren. Am Beispiel unseres Quadrokopters heißt es, gibt man zum Zeitpunkt t dem Quadrokopter, der sich an der Position x befindet, ein Befehl zum Beschleunigen, so erwarten wir den Quadrokopter nicht an der Position $x + v * dt$, am Zeitpunkt $t + dt$, sondern ein Stück weiter, wegen der Beschleunigung die wir vorgeben und dem entsprechend auch mit einer höheren Geschwindigkeit $v + dv$.

1.3.3. Random motion

Der Name spricht auch in deisem Fall für sich, die Random motion oder die zufällige Bewegung, schließt alle die Bewegungen mit ein, die wir nicht vorhersehen können. Sei es ein Windstoß oder ein Rottorbruch.

1.4. OpenCV

OpenCV ist eine quelloffene Programmbibliothek unter BSD-Lizenz[1], kann also auch kommerziell frei genutzt werden. Sie ist für die Programmiersprachen C und C++ geschrieben und enthält Algorithmen für die Bildverarbeitung und maschinelles Sehen. Das CV im Namen steht für Computer Vision. Die Entwicklung der Bibliothek wurde von Intel initiiert und wird heute hauptsächlich von Willow Garage gepflegt. Im September 2006 wurde die Version 1.0 herausgegeben, Ende September 2009 folgte nach längerer Pause die Version 2.0.0, welche die Bezeichnung "Gold" trägt. Die Stärke von OpenCV liegt in ihrer Geschwindigkeit und in der großen Menge der Algorithmen aus neuesten Forschungsergebnissen. Die Bibliothek umfasst unter anderem Algorithmen für Gesichtsdetektion, 3D-Funktionalität, Haar-Klassifikatoren, verschiedene sehr schnelle Filter (Sobel, Canny, Gauß) und Funktionen für die Kamerakalibrierung. [1]

OpenCV-Bibliothek ist frei im Internet erhältlich, und eignet sich für die unterschiedlichsten Bildverarbeitungsaufgaben.

Beschäftigt man sich mit OpenCV eine kurze Zeit, merkt man, dass es ein starkes Werkzeug ist, wenn es darum geht Informationen Bildern oder Videosequenzen zu entlocken. Folgende Disziplinen Zählen zu den wichtigsten OpenCV-Aufgaben:

- Mensch-Computer Interaction (HCI)
- Object Identifikation, Segmentierung und Erkennung

[1] BSD-Lizenz, steht für Berkeley Software Distribution - Lizenz und bezeichnet eine Gruppe von Lizenzen aus dem Open-Source-Bereich. Der Urtyp der Lizenz stammt von der University of California, Berkeley. Software unter BSD-Lizenz darf frei verwendet, kopiert, verändert und verbreiten werden. Einzige Bedingung ist, dass der Copyright-Vermerk des ursprünglichen Programms nicht entfernt werden darf. Somit eignet sich unter einer BSD-Lizenz stehende Software auch als Vorlage für kommerzielle Produkte.

- Gesichtserkennung
- Gestenerkennung
- Objektverfolgung
- Structure From Motion (SFM)[1]
- Mobile Robotics

1.4.1. Aufbau von OpenCV

Die OpenCV-Bibliothek läst sich in 4 wesendliche Teile aufgliedern.

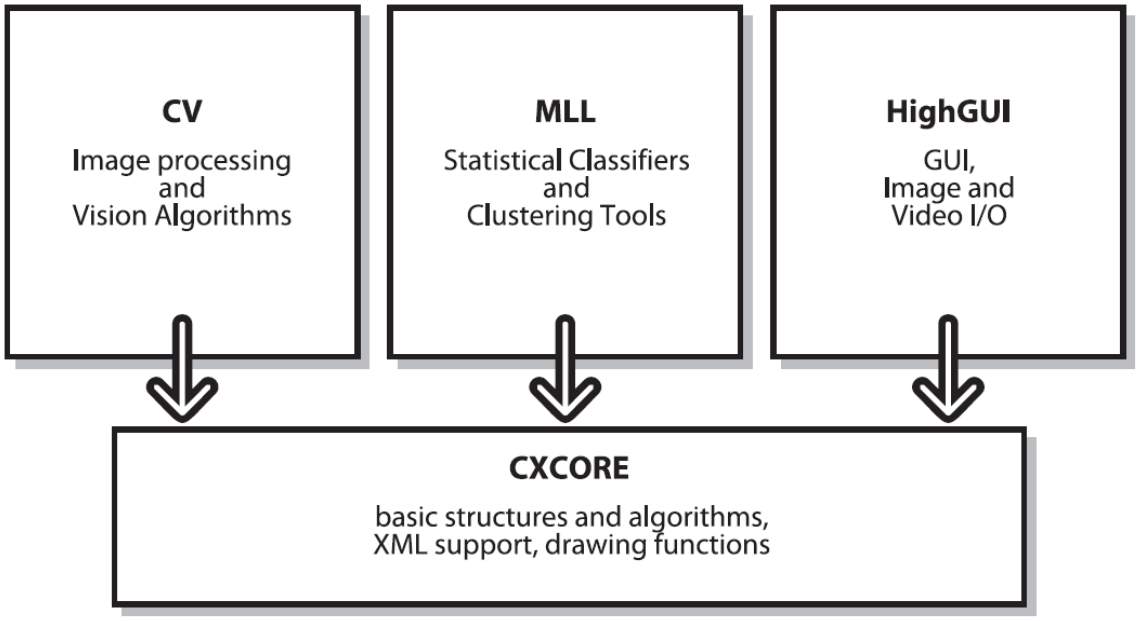

Abbildung 1-13: Aufbau der OpenCV-bibliothek (Quelle[2])

CV steht für *Computer Vision* und benennt alle Algorithmen zur Bild- und Videoverarbeitung, so wie das Verzerren der Bilder (scheren, drehen, skalieren), oder Konvertierungen in verschiedene Bildformate.

MLL heißt *Mashine Learning Library*, es beinhaltet statistische Methoden, Klassifikatoren, Clustering Tools und markiert somit den künstlichen Intellekt von OpenCV.

[1] Structure From Motion, Nennt man in der Wissenschaft das Aufbauen von Dreidimensionalen Räumen, aus Zweidimensionalen Bildsequenzen. Dabei verlässt man sich im Wesentlichen auf die Tatsache, dass sich weit entfernte Gegenstände, optisch langsamer bewegen als diejenigen die sich in der Nähe der Kamera aufhalten.

HighGUI ist für die Kommunikation mit dem Benutzer zuständig, Fensterroutine, Geometrie, aber auch das Laden, das Lesen und Schreiben der Dateien gehört zu ihren Aufgaben.

CXCORE stellt den Datenverkehr und Kommunikation zum Betriebssystem dar und sorgt für einen reibungslosen Arbeitsablauf.

1.4.2. Installation von OpenCV unter Windows

Es existieren sehr viele Beschreibungen der Installation von OpenCV, ob unter Windows oder Mac Systemen, auf Visual Studio oder Eclipse, für C oder C++. Die Meisten von ihnen sind sehr akkurat und detailliert beschrieben und enthalten genügend Bildmaterial um den Benutzer sicher zum Ziel zu führen. Und obwohl die Installation keinerlei Schwierigkeiten bereiten sollte, sind jedoch einige Sachen dabei zu beachten und zu berücksichtigen.

Ich arbeite mit der Version OpenCV 2.0, auf der Entwicklungsplattform Eclipse, mein Betriebssystem ist Windows XP Professional. Unter diesen Bedienungen sieht die Installation wie folgt aus:

S. 356 learning OpenCV

Probleme der Flusserkennung

2. Literaturverzeichnis

[1] http://de.wikipedia.org/wiki/OpenCV, 16.06.2010

[2] Learning OpenCV (Gary Bradski und Adrian Kaehler)

[3] Navigation und Regelung eines Luftschiffes mittels optischer, inertialer und GPS Sensoren, Doktorarbeit, Universität Stuttgart (Martin Fach)

[4] Bewegungserkennung in Videosequenzen zur Erfassung von Verkehrsteilnehmern, Diplomarbeit, Hochschule Esslingen (Ralph Scharpf)