Eberhard Karls
Universität
Tübingen

Kamera-basierte Detektion sich individuell bewegender Hindernisse für die 3D-Navigation eines Flugroboters

Diplomarbeit

vorgelegt von
Isabell Schwab

Tübingen, März 2009

Betreuer:
Prof. Dr. Hanspeter A. Mallot
Lehrstuhl Kognitive Neurowissenschaft
Prof. Dr. Andreas G. Schilling
Lehrstuhl Graphisch-Interaktive Systeme

Eidesstattliche Erklärung

Ich erkläre hiermit, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Tübingen, den 15. März 2009

Kurzfassung

In dieser Diplomarbeit wurde ein Verfahren zur Detektion sich unabhängig bewegender Objekte im Blickfeld einer mobilen Kamera anhand von optischem Fluss entwickelt. Die Eigenbewegung der Kamera wurde dabei berücksichtigt. Aus dem optischen Fluss werden die Eigenbewegungsparameter der Kamera geschätzt. Es wird für jeden Flussvektor die Differenz zwischen beobachtetem und anhand der Eigenbewegungsparameter geschätztem Fluss ermittelt. Anhand dieses Fehlers werden durch dynamisches *thresholding* die durch Eigenbewegung verursachten Flussvektoren (EM) von den durch unabhängige bewegte Objekte verursachten Flussvektoren (IDM) getrennt. Den Kern des Verfahrens bildet dynamisches *thresholding* anhand des Fehlers. Da dynamische *thresholding*-Verfahren für unterschiedliche Anwendungen sehr verschiedene Ergebnisse liefern, wurden drei *thresholding*-Verfahren implementiert und ihre Eignung zur Trennung von EM und IDM verglichen und bewertet. Zwei der getesteten Verfahren, das Kittler-Illingworth-Verfahren und die Methode von Yuan, eignen sich gut für diese Anwendung. Durch unterschiedliche Charakteristika sind ihre Anwendungsschwerpunkte jedoch verschieden.

Danksagung

Diese Diplomarbeit fand am Lehrstuhl Kognitive Neurowiswsenschaft unter der Leitung von Prof. Dr. Mallot statt. Ihm und Prof. Dr. Schilling vom Lehrstuhl für Graphisch-Interaktive Systeme des Wilhelm-Schickard-Instituts für Informatik möchte ich recht herzlich für die Betreuung dieser Diplomarbeit danken.

Mein Dank gilt ebenso meinen beiden Betreuern Dr. Chunrong Yuan und Fabian Recktenwald, die mir stets mit Rat und Tat beiseite standen. Fabian danke ich ausserdem für die Versorgung mit nervenberuhigenden Gummibärchen.

Dem Lehrstuhl für Kognitive Neurowissenschaften möchte ich für das freundliche und angenehme Arbeitsklima danken. Und den lieben Kicker-Kollegen für entspannende Runden am bzw. unterm Tisch. Ausserdem dem Streitwagenkonstrukteur Zottel, sowie den unermüdlichen Kistenwacklern Chris, Dom, den Raketen Sonja und Johannes, Zottel und Fabian.

Auch der ganzen Familie Hrabal möchte ich an dieser Stelle für ihre Unterstützung und ihre Freundschaft danken.
Meinem lieben Freund danke ich von ganzem Herzen für Rückhalt, Aufmunterung und Fürsorge.
Last but not least möchte ich mich bei meinen Eltern bedanken, die mir dieses Studium ermöglicht haben. Und für ihre stete liebevolle Unterstützung in jeder Hinsicht.

Inhaltsverzeichnis

1 Einleitung und Aufgabenstellung **11**

2 Grundlagen **14**

2.1 Optischer Fluss 14

2.1.1 Zentralprojektion 14

2.1.2 Definition von optischem Fluss 15

2.1.3 Das dreidimensionale Vektorfeld bei Eigenbewegung des Betrachters 17

2.1.4 Projektion des dreidimensionalen Bewegungsfelds in die Bildebene 18

2.1.5 Optischer Fluss bei Rotation 19

2.1.6 Optischer Fluss bei Translation 20

2.1.7 Die *smoothness*- und die *brightness*-Annahme 21

2.1.8 Der Lucas-Kanade-Algorithmus 22

2.1.9 Der GDIM-Algorithmus 24

2.1.10 Pyramidaler Ansatz 26

2.2 Egomotion-Schätzung 28

2.3 Dynamisches *thresholding* 30

2.3.1 Methode von Brink und Pendock 31

2.3.2 Methode von Kittler und Illingworth 33

2.3.3 Methode von Yuan 36

2.4 *Clustering* 37

2.4.1 KMeans 37

2.4.2 *Learning the k* : der *G-means*-Algorithmus 38

2.4.3 Clusterinitialisierung 41

2.5 Statistische Tests: Die Anderson-Darling Statistik 42

2.6 RANSAC 44

3 Design und Implementierung **46**
3.1 Motivation für das dynamische *thresholding* 46
3.2 Wahl der *thresholding*-Verfahren 47
3.3 Überblick über die Implementierung 50
3.3.1 Berechnung des optischen Flusses in Vor- und Rückrichtung 50
3.3.2 Filtern von Ausreißern über den Vergleich von Vor- und Rückfluss 51
3.3.3 Egomotion-Schätzung 53
3.3.4 Histogramm der Qualitätswerte erstellen 54
3.3.5 Dynamisches *tresholding* 54
3.3.6 Filtern der als IDM klassifizierten Vektoren durch Überprüfung der Nachbarschaft 55
3.3.7 Clustern anhand der Position im Bild 56
3.3.8 Filtern der Cluster 56
3.3.9 Eventuelle Fusionierung von Clustern bei zu feiner Unterteilung 57
3.4 Aufnahme der Testsequenzen 58
3.5 Ermittlung der *ground truth* 60

4 Ergebnisse und Diskussion **63**
4.1 Wahl des optimalen *thresholds* 63
4.2 Bewertung der Lage der dynamisch gelegten *thresholds* im Vergleich zum idealen *threshold* 66
4.3 Bewertung der Gesamtverfahren 70
4.3.1 Bewertung der Verfahren als Klassifikatoren 71
4.3.2 Eignung der Verfahren zum Filtern von Ausreißern . . . 75
4.4 Erklärung der Inkongruenz 76
4.5 Übergeordnete Bewertungskriterien 79
4.6 Gesamtbewertung 81
4.6.1 Kittler 81
4.6.2 Brink 81
4.6.3 Yuan 82

5 Schlussfolgerung und Ausblick **83**

A Anhang **89**
A.1 Weiteres Beispiel zur Wahl des optimalen *thresholds* 89

Kapitel 1

Einleitung und Aufgabenstellung

Diese Diplomarbeit entstand an der Universität Tübingen am Lehrstuhl für Kognitive Neurowissenschaften im Rahmen des EU-Projekts μDrones [1].
Das Ziel des EU-Projekts μDrones ist die Entwicklung einer autonom navigierenden kleinen Flugdrohne (VTOL micro-UAV=*vertical take-off and landing micro unmanned aerial vehicle*) für automatische Überwachungs- und Erkundungsaufgaben. Die Drohne soll dabei in unbekannter Umgebung sicher navigieren, sowohl statischen als auch sich bewegenden Hindernissen ausweichen, sowie auf unerwartete Ereignisse dynamisch reagieren können. Die Drohne ist hierzu mit verschiedenen Sensoren ausgestattet. Um dieses Ziel zu erreichen integriert das Projekt verschiedene Forschungsgebiete: die Entwicklung eines *mission-control*-Systems mit einem intuitiven Mensch-Maschinen-Interface für die Missionsplanung, die Entwicklung von Perzeptions- und Steuerungsalgorithmen, welche die Flugautonomie ermöglichen sollen, die Entwicklung eines VTOL micro-UAV-Prototyps. An dem Projekt sind fünf internationale Projektpartner beteiligt: das französische Forschungszentrum für Technik CEA-LIST [2], Thales Security Systems [3] - ein internationaler Konzern für Sicherheitssysteme, AirRobot [4] - ein auf die Entwicklung von Mini-UAV-Systemen spezialisiertes Unternehmen, das griechische Unternehmen Lisippos S.A., sowie der Lehrstuhl für Kognitive Neurowissenschaften der Universität Tübingen [5].
Der Lehrstuhl für Kognitive Neurowissenschaften ist hierbei für die Entwicklung von Komponenten zur automatischen Hindernisvermeidung verantwortlich. Die Hindernisvermeidung soll dabei inspiriert durch die Natur ohne explizite Objekterkennung, basierend auf optischem Fluss geschehen. Von einigen Insekten ist bekannt, dass sie das durch Eigenbewegung entstehende Flussfeld zur Navigation nutzen. Honigbienen beispielsweise zentrieren ihre Flugrichtung

[1] `http://www.ist-microdrones.org`
[2] `http://www-list.cea.fr`
[3] `http://www.thalesgroup.com/security-services`
[4] `http://www.airrobot.com/`
[5] `http://www.cog.uni-tuebingen.de/`

so, dass der optische Fluss auf beiden Augen ausgeglichen ist. Da durch Translation entstehender optischer Fluss von der Tiefe der passierten Objekte abhängt, hat ein Angleichen des optischen Flusses der beiden Augen automatisch eine Zentrierung der Flugbahn zur Folge [Srinivasan et al., 1996]. Dieses Verhalten wurde mit mobilen Robotern erfolgreich imitiert, beispielsweise durch Coombs und Roberts [Coombs and Roberts, 1993]. Bei Flugrobotern, wie dem kleinen Quadrokopter des μDrones Projekts AR100 der Firma AirRobot ist eine solche Imitation tierischen Verhaltens noch komplexer. Während mobile Roboter ihre Eigenbewegung durch Odometrie zu bestimmen vermögen, ist dies bei Flugrobotern nicht möglich. Tatsächlich ist es eine nicht-triviale Aufgabe, bei fliegenden Objekten die Geschwindigkeit ohne externe Signale wie beispielsweise GPS, basierend nur auf externen Sensoren zu bestimmen. Die Flugdrohne des μDrones-Projekt ist ein AR100-Quadrokopter. Der visuelle Sensor der Flugdrohne ist eine auf der Drohne montierte digitale Farbkamera. Die Bildsignale werden per Funk an die Bodenstation geschickt und dort verarbeitet.

In dieser Diplomarbeit sollte nun versucht werden, anhand des optischen Flusses sich unabhängig bewegende Objekte (IDM =*independent motion*) im Blickfeld einer sich bewegenden Kamera zu detektieren. Die Versuche wurden mit einer Webcam durchgeführt, die Flugdrohne wurde nicht verwendet.
Da sich die Kamera bewegt, entsteht durch die Eigenbewegung optischer Fluss. Befindet sich im Blickfeld der sich bewegenden Kamera ein IDM-Objekt, so addieren sich die durch Eigenbewegung und durch die IDM entstehenden Flussvektoren. Abbildung 1.1 zeigt ein Beispiel für ein durch Eigenbewegung entstandenes Flussfeld, in dem sich aber auch unabhängige Bewegung befindet.

Es sollte ein Verfahren entwickelt werden, mit dem sich durch Egomotion entstandene Flussvektoren (=EM) und durch sich unabhängig bewegende Objekte entstandene Flussvektoren (=IDM) automatisch trennen lassen. Die Auftrennung von durch EM und IDM entstandenen Flussvektoren ist aus folgendem Grund wichtig: Die Navigation und die Hindernisvermeidung der Drohne basieren auf optischem Fluss. Aus dem Flussfeld werden die Eigenbewegungsparameter geschätzt. Wurden die Eigenbewegungsparameter aus dem Flussfeld geschätzt, so lässt sich für jeden Flussvektor eine geschätzte Tiefe berechnen. Diese Tiefenschätzung gilt jedoch nur für durch EM entstandene Flussvektoren. Für durch IDM entstandene Vektoren ist sie also falsch. Daher ist es wichtig zu wissen, ob sich im Blickfeld des Roboters IDM befindet, damit man ihr rechtzeitig ausweichen kann.

Es wurde der Ansatz gewählt, die EM und die IDM anhand ihrer Reprojektionsfehler aufzutrennen. Den Kern des implementierten Verfahrens bilden dynamische *thresholding*-Verfahren. Da die Leistung eines automatischen *thresholding*-Verfahrens stark der Anwendung abhängt [Burgoyne et al., 2007] und

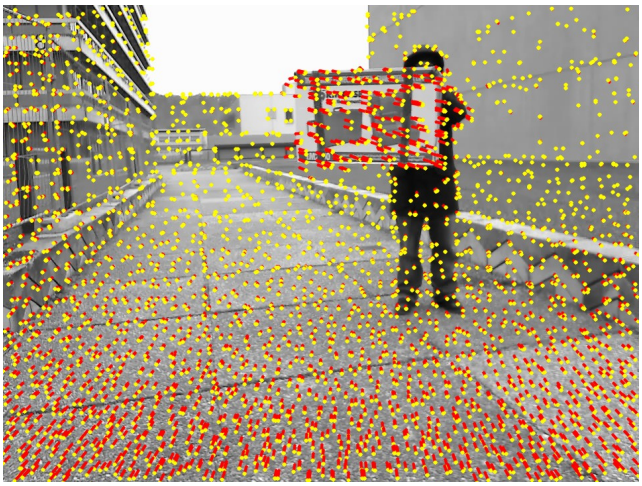


Abbildung 1.1: Beispiel für ein durch Eigenbewegung entstandenes Flussfeld, in dem aber auch *independent motion* vorhanden ist. Die Kamera bewegt sich geradeaus ins Bild, die Kiste - das IDM-Objekt - wird nach links bewegt.

es bislang keine Literatur zur Anwendung von automatischem *thresholding* auf die Trennung von EM und IDM anhand der Reprojektionsfehler gibt, wurden drei Verfahren implementiert und jeweils ihre Leistungen für diese Aufgabe bewertet.

Diese Arbeit ist in fünf Teile untergliedert. Im Grundlagen-Teil (Kapitel 2) werden die zum Verständnis der Arbeit benötigten Konzepte erläutert, sowie die Algorithmen beschrieben, die in der Arbeit verwendet werden. Das Kapitel Design und Implementierung (Kapitel 3) enthält die Motivation für dynamisches *thresholding* zur Klassifizierung von EM und IDM, sowie die Wahl speziell der verwendeten *thresholding*-Methoden. Die implementierte Pipeline wird beschrieben, die einzelnen Arbeitsschritte erläutert. Im Teil Ergebnisse und Diskussion (Kapitel 4) werden die Ergebnisse, die sich mit den implementierten Verfahren erzielen ließen, vorgestellt. Diese werden auf unterschiedliche Arten bewertet und diskutiert. Am Ende des Abschnitts befindet sich eine Gesamtbewertung der drei Verfahren, sowie eine Beschreibung der Stärken und Schwächen. Der Teil Schlussfolgerung und Ausblick (Kapitel 5) erörtert Möglichkeiten, die Leistung des implementierten Algorithmus zu verbessern, sowie Anregungen für das weitere Vorgehen.

Kapitel 2

Grundlagen

2.1 Optischer Fluss

2.1.1 Zentralprojektion

Um die Enstehung von optischem Fluss zu begreifen, benötigt man ein Modell der Funktionsweise einer Kamera. In Abbildung 2.1 ist ein allgemeines Kameramodell dargestellt, das nach dem Prinzip der Zentralprojektion (Abbildung 2.2) funktioniert.

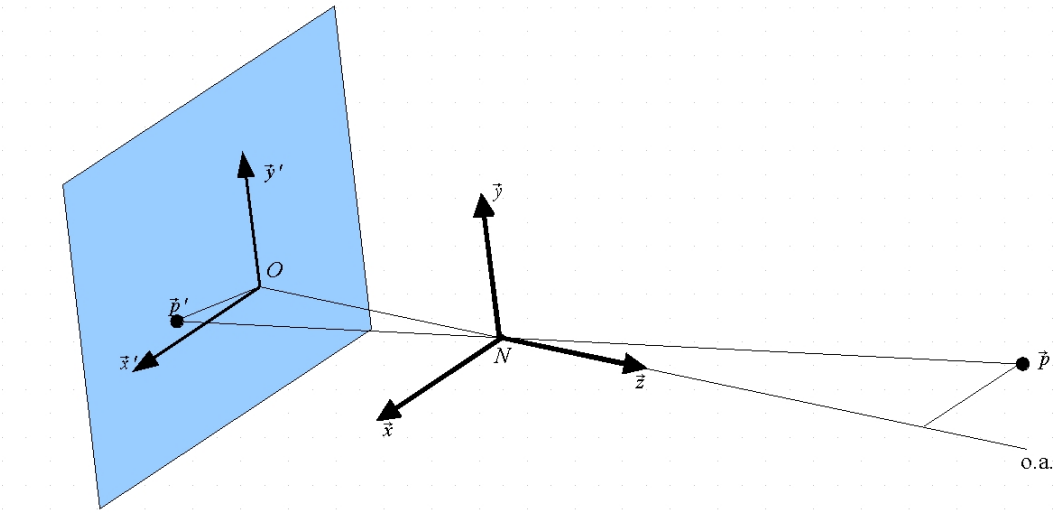


Abbildung 2.1: Allgemeines Kameramodell. Die Bildebene steht orthogonal auf der optischen Achse. Das Projektionszentrum ist der optische Drehpunkt N. Dieser liegt im Abstand $f = \overline{NO}$ zur Bildebene. (nach [Mallot and Allen, 2000])

Damit lässt sich die Abbildung dreidimensionaler Objekte und Szenen auf

zweidimensionale Bilder erklären. Die Kamera, genauer gesagt deren optisches Zentrum, das Projektionszentrum, befindet sich im Punkt N. Das Kamerakoordinatensystem ist so justiert, dass die z-Achse in Blickrichtung zeigt, die y-Achse orthogonal dazu die Höhe angibt, und die x-Achse orthogonal zu beiden zur Seite weist. Die Bildebene hat vom Projektionszentrum den Abstand f. Normalerweise wird $f = 1$ angenommen. Um zu ermitteln, wohin im Bild ein 3D-Punkt abgebildet wird, denkt man sich einen Projektionsstrahl durch den Punkt N und den abzubildenden 3D-Punkt $\vec{\mathbf{p}} = (p_1, p_2, p_3)$. Der Schnittpunkt des Projektionsstrahl mit der Bildebene kennzeichnet den Bildpunkt $\vec{\mathbf{p}^{\mathbf{i}}} = (p_1^i, p_2^i)$.
Die Abbildungvorschrift, gemäß derer man 3D-Punkte in die Bildebene projizieren kann lautet:

$$\vec{\mathbf{p}} = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \mapsto \begin{pmatrix} p_1^i \\ p_2^i \end{pmatrix} := -\frac{f}{p_3} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \tag{2.1}$$

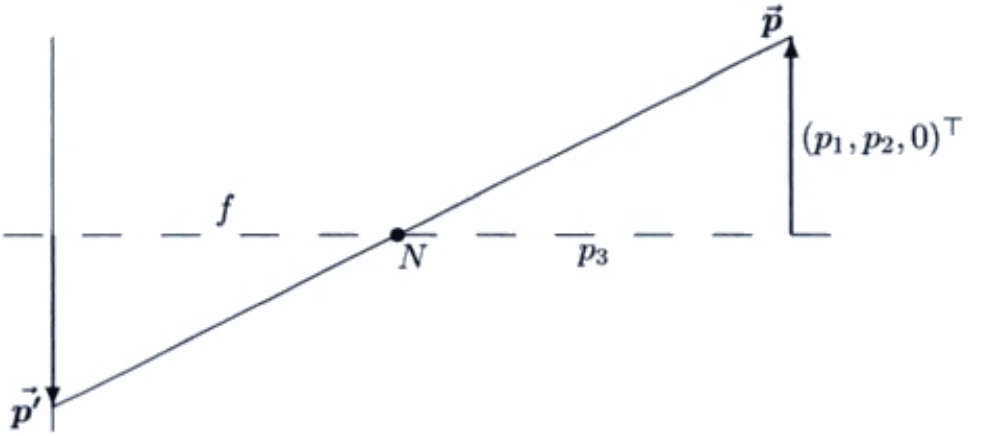


Abbildung 2.2: Verdeutlichende Abbildung zur Herleitung der Formel zur Zentralprojektion: $\vec{\mathbf{p}}$ (aus [Mallot and Allen, 2000])

2.1.2 Definition von optischem Fluss

Optischer Fluss ist definiert als die sichtbare Verschiebung von Helligkeitsmustern in einer Bild-Sequenz [Horn, 1986]. Er kann einerseits durch die relative Bewegung zwischen Kamera und aufgenommener Szene hervorgerufen werden, als auch durch die Bewegung von Objekten im Blickfeld der Kamera.

Die Entstehung von Optischem Fluss im Falle der Bewegung von Objekten in der aufgenommenen Szene wird an Abbildung 2.3 erläutert. In diesem Fall führt die Bewegung eines Punktes P_0 in der Szene zu einer Bewegung des Bildpunktes P_i in der Bildebene. Betrachtet wird ein Zeitintervall δt. Der 3D-Punkt P_0 bewegt sich mit Geschwindigkeit v_0, im Zeitintervall δt also um die Strecke $\delta t v_0$. Dies führt zu einer Bewegung des Bildpunktes mit der Geschwindigkeit v_i, der sich somit um $\delta t v_i$ bewegt.

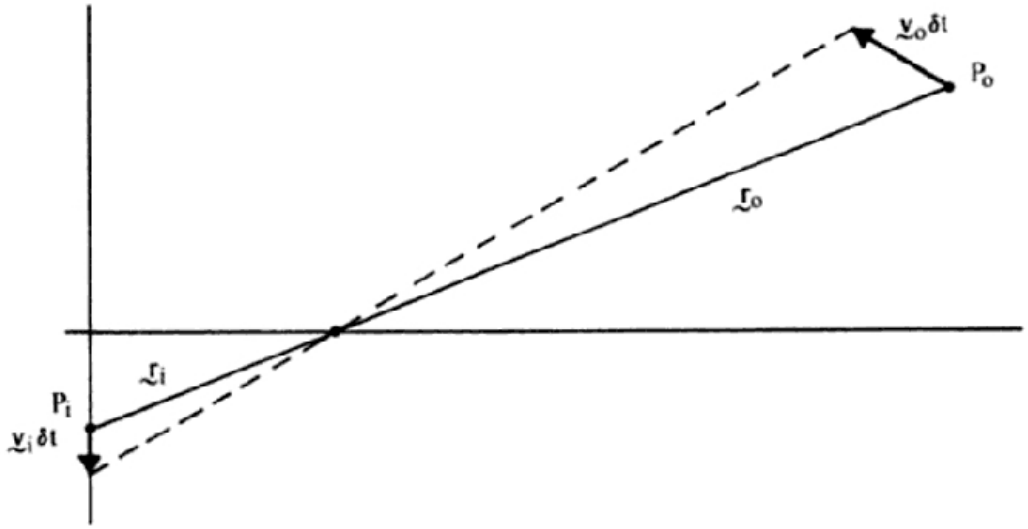
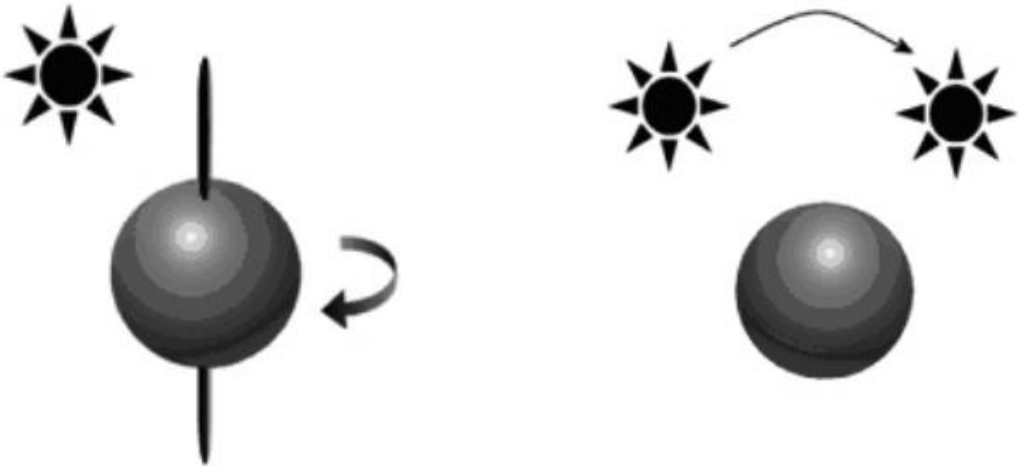


Abbildung 2.3: Entstehung von optischem Fluss (aus [Horn, 1986])

Betrachtet man diese Projektion für alle Punkte im Bild, so erhält man das sogenannte Bewegungsfeld (engl.: *motion field*). Das Bewegungsfeld ordnet jedem Punkt im Bild einen Geschwindigkeitsvektor zu.

Wenn sich die Bewegung sichtbar im Bild niederschlägt, so ist das Bewegungsfeld gleich dem Optischen Fluss. Dies ist aber der Idealfall, der nicht unbedingt zutreffen muss, wie sich an folgenden zwei Beispielen zeigt: In Abbildung 2.4a bewegt sich eine ebenmäßige, ungemusterte Kugel. Obwohl eigentlich Bewegung vorhanden ist, entsteht doch kein Optischer Fluss, da die Bewegung nicht wahrnehmbar ist. In Abbildung 2.4b steht die Kugel still. Jedoch wandert die Beleuchtungsquelle und somit die Schatten auf der Kugel. Das Bewegungsfeld der Kugel ist gleich null, trotzdem entsteht ein Optischer Fluss aus der Bewegung der Schatten auf der Kugel.



(a) Bewegungsfeld $\neq 0$, aber Optischer Fluss$= 0$

(b) Bewegungsfeld $= 0$, aber Optischer Fluss $\neq 0$

Abbildung 2.4: Vergleich von Optischem Fluss und Bewegungsfeld (aus [Naderi, 2007])

2.1.3 Das dreidimensionale Vektorfeld bei Eigenbewegung des Betrachters

Bewegung im Raum besitzt sechs Freiheitsgrade und lässt sich in einen Rotations- und einen Translationsanteil zerlegen. Der Translationsanteil wird durch den Translationsvektor des optischen Drehpunktes der Kamera $\vec{\mathbf{k}}\,(t)$ beschrieben. Dieser besitzt drei Freiheitsgrade. Die Rotation kann durch Multiplikation des Positionsvektors $\vec{\mathbf{k}}\,(t)$ mit der Rotationsmatrix $\mathbf{A}$ beschrieben werden ($\mathbf{A}$ ist eine orthonormale Matrix mit Determinante $= 1$). Die Rotation besitzt ebenfalls drei Freiheitsgrade, entsprechend den drei eulerschen Drehwinkeln um die Koordinatenachsen. Sowohl Rotation als auch Translation sind zeitabhängig. Die Position eines festen Punktes $\vec{\mathbf{p_0}}$ zum Zeitpunkt t bezüglich des (sich bewegenden) Kamera-Koordinaten-Systems ist gegeben durch:

$$\vec{\mathbf{p}}\,(t) = \mathbf{A}\,(t)\left(\vec{\mathbf{p_0}} - \vec{\mathbf{k}}\,(t)\right) \tag{2.2}$$

Leitet man Gleichung 2.2 nach der Zeit ab, so erhält man das dreidimensionale Bewegungsfeld $\vec{\mathbf{w}}\,(t)$:

$$\vec{\mathbf{w}}\left(\vec{\mathbf{p}}\,(t)\right) = \mathbf{A}'\,(t)\left(\vec{\mathbf{p_0}} - \vec{\mathbf{k}}\,(t)\right) - \mathbf{A}\,(t)\,\vec{\mathbf{k}'}\,(t) \tag{2.3}$$

Da man an der momentanen Bewegung interessiert ist, betrachtet man den Zeitpunkt $t = 0$:

$$\vec{\mathbf{w}}\left(\vec{\mathbf{p_0}}\right) = \mathbf{A}'(0)\,\vec{\mathbf{p_0}} - \vec{\mathbf{k}}'(0) \tag{2.4}$$

Dabei ist $\vec{\mathbf{k}}'(0)$ die zeitliche Ableitung der Bewegungsrichtung. Sie entspricht der Translationskomponente der Bewegung und wird mit $v\,\vec{\mathbf{u}}$ dargestellt, wobei $\vec{\mathbf{u}}$ der Einheitsvektor in Translationsrichtung ist.
$\mathbf{A}'(0)$ ist die Ableitung der Rotation zum Zeitpunkt $t = 0$, Diese ist gegeben durch

$$\mathbf{A}' = \begin{pmatrix} 0 & w_3 & -w_2 \\ -w_3 & 0 & w_1 \\ w_2 & -w_1 & 0 \end{pmatrix} \tag{2.5}$$

Für die Herleitung sei auf [Mallot and Allen, 2000] S.205 verwiesen.
Das dreidimensionale Vektorfeld, das durch Egomotion entsteht ist also

$$\vec{\mathbf{w}}\left(\vec{\mathbf{x}}\right) = -v\,\vec{\mathbf{u}} - w\,\vec{\mathbf{r}} \times \vec{\mathbf{x}} \tag{2.6}$$

2.1.4 Projektion des dreidimensionalen Bewegungsfelds in die Bildebene

Betrachte den Punkte $\vec{\mathbf{p}}(t)$ mit den zeitabhängigen Koordinaten $\vec{\mathbf{p}}(t) = (p_1(t), p_2(t), p_3(t))^T$ bezüglich des sich bewegenden Kamera-Koordinatensystems. Die momentane Geschwindigkeit zum Zeitpunkt $t = 0$ ist dann gleich dem Wert des dreidimensionalen Vektorfelds $\vec{\mathbf{w}}$ im Punkt $\vec{\mathbf{p}}(0)$. Das heißt

$$\frac{d}{dt}\,\vec{\mathbf{p}}(0) = \vec{\mathbf{w}}\left(\vec{\mathbf{p}}(0)\right) \tag{2.7}$$

Die Zentralprojektion $\vec{\mathbf{p^i}}(t)$ des sich bewegenden Punktes $\vec{\mathbf{p}}(t)$ mit den Koordinaten $\vec{\mathbf{p}}(t) = (p_1(t), p_2(t), p_3(t))^T$ ist nach Gleichung 2.1

$$\vec{\mathbf{p^i}}(t) = \frac{1}{p_3(t)}\begin{pmatrix} p_1(t) \\ p_2(t) \end{pmatrix} \tag{2.8}$$

Leitet man Gleichung 2.8 nach der Zeit ab, so erhält man die Projektionsformel, mit der man den dreidimensionalen Bewegungsvektor in die Bildebene projizieren kann. Für die x-Komponente ergibt sich mit der Quotientenregel

$$\frac{d}{dt}p_1^i(t) = \frac{d}{dt}\left(-\frac{p_1(t)}{p_3(t)}\right) = -\frac{1}{p_3}\left(\frac{dp_1}{dt} + p_1^i\frac{dp_3}{dt}\right) \tag{2.9}$$

Für die y-Komponente ergibt sich analog

$$\frac{d}{dt}p_2^i(t) = \frac{d}{dt}\left(-\frac{p_2(t)}{p_3(t)}\right) = -\frac{1}{p_3}\left(\frac{dp_2}{dt} + p_2^i\frac{dp_3}{dt}\right) \tag{2.10}$$

Da $\frac{d}{dt}\vec{\mathbf{p}}(t) = \vec{\mathbf{w}}$ erhält man

$$\begin{pmatrix}\frac{d}{dt}p_1^i(t)\\ \frac{d}{dt}p_2^i(t)\end{pmatrix} = -\frac{1}{p_3}\left(\begin{pmatrix}w_1\\ w_2\end{pmatrix} + w_3\begin{pmatrix}p_1^i\\ p_2^i\end{pmatrix}\right) := \vec{\mathbf{v}} \tag{2.11}$$

Diese Projektion der 3D-Bewegung nach 2D entspricht genau dem optischen Fluss $\vec{\mathbf{v}}$.

2.1.5 Optischer Fluss bei Rotation

Wir betrachten eine Kamera, welche um die Achse $\vec{\mathbf{r}}$ rotiert. Die Drehachse $\vec{\mathbf{r}}$ schneidet dabei den optischen Drehpunkt (= Nodalpunkt, engl.: *nodal point*) der Kamera. Die Rotationsgeschwindigkeit beträgt ω. Nimmt man an, dass die Länge der Drehachse auf 1 normiert ist, dann ist das dreidimensionale Bewegungsfeld $\vec{\mathbf{w}}(p_1, p_2, p_3)$ gegeben durch:

$$\vec{\mathbf{w}}(p_1, p_2, p_3) = -\omega\,\vec{\mathbf{r}} \times \vec{\mathbf{p}} = -\omega\begin{pmatrix}r_2p_3 - r_3p_2\\ r_3p_1 - r_1p_3\\ r_1p_2 - r_2p_1\end{pmatrix} \tag{2.12}$$

Im Falle einer reinen Rotation um den optischen Drehpunkt ändert sich die Perspektive nicht. Daher enthält ein reines Rotationsfeld keine Informationen über den dreidimensionalen Aufbau der betrachteten Szene. Projiziert man die 3D-rotationale Bewegung $\vec{\mathbf{w}}\left(\vec{p}\right)$ gemäß Gleichung 2.11 in die Bildebene, so erhält man:

$$\vec{\mathbf{v}} = \frac{\omega}{p_3}\left(\begin{pmatrix}r_2p_3 - r_3p_2\\ r_3p_1 - r_1p_3\end{pmatrix} + (r_1p_2 - r_2p_1)\begin{pmatrix}x'\\ y'\end{pmatrix}\right) \tag{2.13}$$

Man sieht, dass die räumlichen Koordinaten (p_1, p_2, p_3) nur in Form der Quotienten $x' = -p_1/p_3$ und $y' = -p_2/p_3$ auftauchen. Daher erzeugen alle Punkte, die entlang eines 3D-Strahls durch den optischen Drehpunkt der Kamera gehen, denselben Vektor $\vec{\mathbf{v}}$ im Bild. Daher kann man die 3D-Koordinaten durch Bildkoordinaten ersetzen. Es gilt also $(x', y') = -(p_1/p_3, p_2/p_3)$. Unter Verwendung dieser Gleichung kann man Gleichung 2.13 folgendermaßen umformen:

$$\vec{\mathbf{v}}(x', y') = \omega\left(-r_1\begin{pmatrix}x'y'\\ 1 + y'^2\end{pmatrix} + r_2\begin{pmatrix}1 + x'^2\\ x'y'\end{pmatrix} + r_3\begin{pmatrix}y'\\ -x'\end{pmatrix}\right) \tag{2.14}$$

In Abbildung 2.5 sind zwei Spezialfälle einer reinen Rotationsbewegung des Betrachters dargestellt, sowie die Projektion des durch Rotation entstehenden Flussfeldes auf eine Kugel.

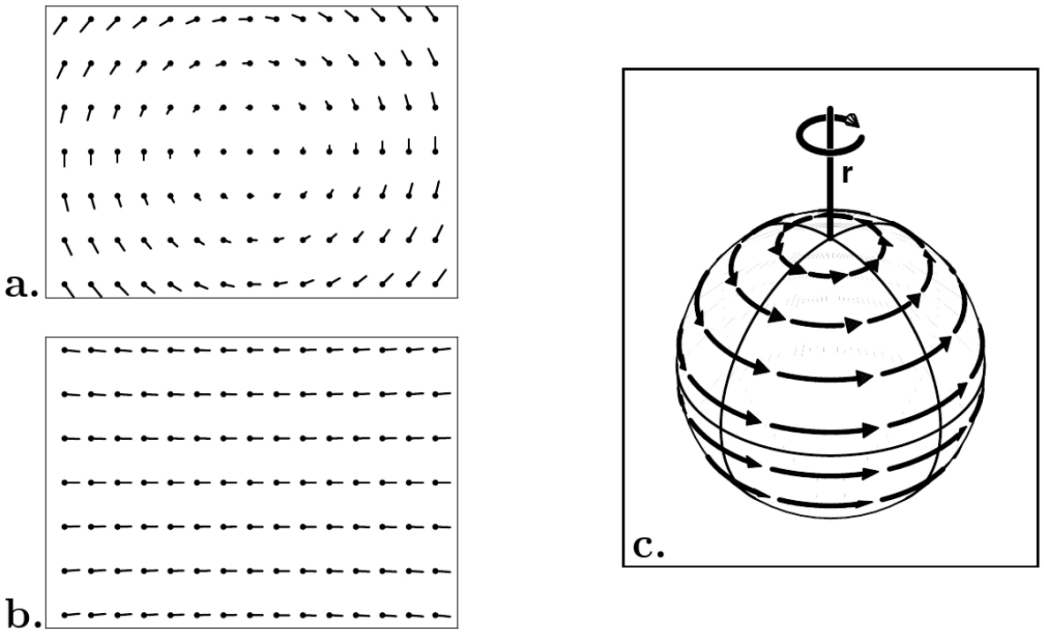


Abbildung 2.5: Abbildung **a.** zeigt den Optischen Fluss, der bei Rotation um die optische Achse entsteht, also Rotation um die Achse der Blickrichtung. Abbildung **b.** zeigt den optischen Fluss, der bei Rotation um die y-Achse entsteht. Abbildung **c.** zeigt das Flussfeld auf die Oberfläche einer Kugel um den Beobachter projiziert. Die Rotationsachse dabei ist $\vec{\mathbf{r}}$(aus [Mallot and Allen, 2000])

2.1.6 Optischer Fluss bei Translation

Wenn man eine rein translatorische Bewegung der Kamera mit dem Translationsvektor $v\ \vec{\mathbf{u}}$ betrachtet - v ist dabei die Geschwindigkeit, der auf 1 normierte Vektor $\vec{\mathbf{m}}$ gibt die Translationsrichtung an - so gilt für das dreidimensionale Bewegungsfeld $\vec{\mathbf{w}}\ (p_1, p_2, p_3)$

$$\vec{\mathbf{w}}\ (p_1, p_2, p_3) = -v\ \vec{\mathbf{u}} = -v\,(u_1, u_2, u_3)^T \tag{2.15}$$

Setzt man dies in die Projektionsformel für sich bewegende Punkte (Gleichung 2.11) ein, so erhält man:

$$\vec{\mathbf{v}}\ (x', y') = \frac{v}{p_3}\left(\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + u_3 \begin{pmatrix} x' \\ y' \end{pmatrix}\right) \tag{2.16}$$

Im Falle von translationaler Bewegung hängt das Flussfeld also von der Tiefe p_3 des betrachteten Objekts ab. Daher kann man aus dem Translationsfeld Informationen über die dreidimensionale Struktur der Umgebung gewinnen. Eine Besonderheit des Translationsfeldes, das unabhängig von der räumlichen Struktur der betrachteten Szene ist, ist der sogenannte *focus of expansion* (f.o.e.): Für $u_3 \neq 0$ gibt es einen Punkt in der Bildebene, in dem das projizierte Vektorfeld verschwindet, beziehungsweise auftaucht, das heißt es gilt dort $\vec{\mathbf{v}}\ (x', y') = 0$. Daher gilt für den *focus of expansion* $\vec{\mathbf{F}}$:

$$\vec{\mathbf{v}} = 0 \ \Rightarrow \ u_1 + u_3 x'_0 = 0 \ , \ u_2 + u_3 y'_0 = 0 \tag{2.17}$$

$$\Rightarrow \ \begin{pmatrix} x'_0 \\ y'_0 \end{pmatrix} = \frac{1}{u_3} \begin{pmatrix} u_1 \\ u_3 \end{pmatrix} =: \vec{\mathbf{F}} \tag{2.18}$$

Der *focus of expansion* $\vec{\mathbf{F}}$ liegt in der Verlängerung der Bewegungsrichtung des Betrachters. Abbildung 2.6 zeigt zwei Beispiele für translatorischen optischen Fluss und den *focus of expansion*, sowie die Projektion des translatorischen Flussfelds auf eine Kugeloberfläche um den Betrachter.

2.1.7 Die *smoothness*- und die *brightness*-Annahme

Die *brightness constancy assumption* beruht auf der Annahme, dass die Helligkeit eines verfolgten Pixel-Patches über die Zeit konstant bleibt. Diese Annahme ist richtig, wenn die Beleuchtungsbedingungen zwischen zwei aufeinanderfolgenden Bildern unverändert bleiben, wenn die Objektoberflächen nicht spiegeln, und wenn zwischen den Bildern nur eine kleine Bewegung stattfindet.

Die *smoothness assumption* resultiert aus der Beobachtung, dass wahrnehmbare Bewegung normalerweise auf einheitliche Bewegung diskreter Objekte zurückzuführen ist. Daher erfahren benachbarte 3D-Punkte zumeist eine ähnliche Bewegung, und wenn diese benachbarten Szenenpunkte dann auf benachbarte Bildpunkte projizieren, so besitzen diese benachbarten Bildpunkte auch ähnliche Flussvektoren. Aus diesem Grunde sollte das Flussfeld zum größten Teil glatt sein. Ausnahmen bilden Kanten, an denen Objekte einander verdecken. An solchen Stellen entsprechen benachbarte Bildpixel nicht benachbarten Szenenpunkten und müssen somit auch keine ähnlichen Flussvektoren besitzen. Die *smoothness assumption* wird für gewöhnlich dadurch verwirklicht, dass eine Nachbarschaft des aktuell betrachteten Pixels mitbetrachtet wird. Dies erleichtert zusätzlich das Auffinden des korrespondierenden Pixels im zweiten Bild.

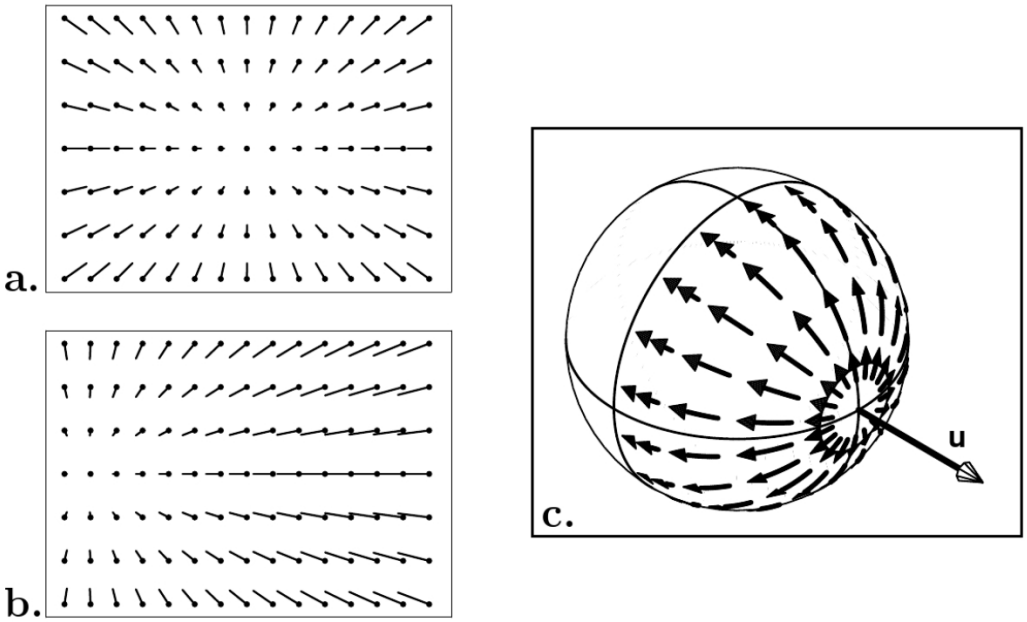


Abbildung 2.6: Abbildung **a.** und **b.** zeigen den optischen Fluss, der bei Translation entsteht. Der f.o.e. liegt in Abbildung **a.** in der Bildmitte, das heißt die Translationsrichtung ist geradeaus nach vorne. In Abbildung **b.** liegt er links außen, das heißt die Bewegungsrichtung ist nach vorne links. Abbildung **c.** zeigt das translatorische Flussfeld auf die Oberfläche einer Kugel um den Beobachter projiziert. Translationsrichtung ist $\vec{\mathbf{u}}$. (aus [Mallot and Allen, 2000])

2.1.8 Der Lucas-Kanade-Algorithmus

Die Lucas-Kanade-Methode zur Berechnung des optischen Flusses geht zurück auf Bruce Lucas und Takeo Kanade [Lucas and Kanade, 1981] und wurde durch Tomasi und Kanade weiterentwickelt [Carlo Tomasi and Takeo Kanade, 1991].

Eingabe für die Lukas-Kanade-Methode sind zwei grauwertige Bilder $A(x, y)$ und $B(x, y)$. Man betrachtet einen bestimmten Punkt $\vec{u}$ im ersten Bild A und möchte nun den korrespondierenden Punkt $\vec{w} = \vec{u} + \vec{v} = [u_x + v_x, u_y + v_y]^T$ im zweiten Bild B finden. Der gesuchte Vektor $\vec{v}$ ist der optische Fluss.

Der gesuchte optische Fluss $\vec{v}$ soll die folgende Zielfunktion minimieren:

$$\epsilon\left(\vec{v}\right) = \epsilon\left(v_x, v_y\right) = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{x=p_x-w_x}^{p_x+w_x} \left(A(x, y) - B(x + v_x, y + v_y)\right)^2 \quad (2.19)$$

Bei der Lukas-Kanade-Methode gelten die *smoothness*- und *brightness*-Annahme. Die *smoothness*-Annahme wird verwirklicht, indem eine Nachbarschaft des ak-

tuellen Pixels betrachtet wird. Man nennt diese Nachbarschaft das Integrationsfenster der Größe $w = (w_x, w_y)$.
Man sucht also das Optimum der Funktion ϵ in Abhängigkeit vom gesuchten optischen Fluss $\vec{v}$. Da die erste Ableitung einer Funktion an ihrem Optimum gleich Null ist, betrachtet man also die erste Ableitung von ϵ nach $\vec{v}$:

$$\frac{\partial \epsilon\left(\vec{v}\right)}{\partial\, \vec{v}} = -2 \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{x=p_x-w_x}^{p_x+w_x} \left(A\left(x,y\right) - B\left(x+v_x, y+v_y\right)\right) \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} \tag{2.20}$$

Da man von kleinen Verschiebungen zwischen Bild A und Bild B ausgeht, kann man $B\left(x+v_x, y+v_y\right)$ durch die Taylor-Entwicklung erster Ordnung im Punkt $(0,0)$ approximieren. Diese lautet:

$$B\left(x+v_x, y+v_y\right) \approx B\left(x,y\right) + \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} \cdot \vec{v} \tag{2.21}$$

Setzt man Gleichung 2.21 in Gleichung 2.20 ein, so erhält man

$$\frac{\partial \epsilon\left(\vec{v}\right)}{\partial\, \vec{v}} = -2 \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{x=p_x-w_x}^{p_x+w_x} \left(A\left(x,y\right) - B\left(x,y\right) - \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} \vec{v}\right) \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} \tag{2.22}$$

Die Differenz $A\left(x,y\right) - B\left(x,y\right)$ kann man in Kurzform schreiben als

$$A\left(x,y\right) - B\left(x,y\right) = \delta I\left(x,y\right) \tag{2.23}$$

$\begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix}$ ist der Vektor der räumlichen Bildgradienten. Man kann ihn schreiben als:

$$\nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix} = \begin{bmatrix} \frac{\partial B}{\partial x} \\ \frac{\partial B}{\partial y} \end{bmatrix} \;\Leftrightarrow\; \nabla I^T = \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} \tag{2.24}$$

Setzt man Gleichung 2.23 und 2.24 in Gleichung 2.22 ein, so erhält man

$$\frac{\partial \epsilon(\left(\vec{v}\right)}{\partial\, \vec{v}} \approx -2 \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{x=p_x-w_x}^{p_x+w_x} \left(\nabla I^T \cdot \vec{v} - \delta I\left(x,y\right)\right) \cdot \nabla I^T \tag{2.25}$$

Umformen und Ausmultiplizieren ergibt

$$\frac{1}{2} \frac{\partial \epsilon(\left(\vec{v}\right)}{\partial\, \vec{v}} \approx \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{x=p_x-w_x}^{p_x+w_x} \left(\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \cdot \vec{v} - \begin{bmatrix} \delta I \cdot I_x \\ \delta I \cdot I_y \end{bmatrix}\right) \tag{2.26}$$

Setzt man

$$\sum_{x=p_x-w_x}^{p_x+w_x} \sum_{x=p_x-w_x}^{p_x+w_x} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} =: G \tag{2.27}$$

und

$$\sum_{x=p_x-w_x}^{p_x+w_x} \sum_{x=p_x-w_x}^{p_x+w_x} \begin{bmatrix} \delta I \cdot I_x \\ \delta I \cdot I_y \end{bmatrix} =: \vec{b} \tag{2.28}$$

so erhält man

$$\frac{1}{2} \frac{\partial \epsilon \left(\vec{v} \right)}{\partial \vec{v}} \approx G \vec{v} - \vec{b} \tag{2.29}$$

Gleichung 2.25, die erste Ableitung von Gleichung 2.19, ist am Optimum gleich Null. Daher lässt sich der optische Fluss berechnen durch

$$\vec{v} = G^{-1} \vec{b} \tag{2.30}$$

Um den optischen Fluss berechnen zu können, muss die Matrix G invertierbar sein. Ist sie das nicht, so lässt sich kein Fluss berechnen. In der Nähe von Singularitäten können auch unsinnige große oder kleine Ergebnisse, das heißt Ausreißer unter den Flussvektoren, entstehen. Die Matrix G ist dann nicht invertierbar, wenn das Bild zu wenig Gradienteninformationen in x- oder y-Richtung der Nachbarschaft des aktuell betrachteten Pixels aufweist.

2.1.9 Der GDIM-Algorithmus

In [Negahdaripour, 1998] stellt Shariar Negahdaripour einen Algorithmus zur Berechnung von Optischem Fluss nach einer erweiterten Definition vor. Diese neue Definition von optischem Fluss nennt Negahdaripour *Generalized Dynamic Image Model*(GDIM). Negahdaripour stellte diese neue Definition erstmals in [Negahdaripour and Lanjing, 1995] vor. Die neue Definition hat den Vorteil, dass sie Helligkeitsänderungen zwischen den Bildern erlaubt. Die *brightness-consistency*-Annahme wird verworfen. Stattdessen wird der optische Fluss als die gesamte wahrnehmbare Transformation zwischen zwei Bildern $[\delta x, \delta y, \delta e]$ definiert. Dabei ist $[\delta x, \delta y]$ wie bisher die Verschiebung im Bild, die Erweiterung $[\delta e]$ beschreibt ein radiometrisches Feld, welches die Helligkeitsänderung von einem Pixel in Bild1 zum entsprechenden Pixel in Bild 2 beschreibt $\delta e = e(r + \delta r) - e(r)$.

Diese neue Definition beruht also statt wie bisher auf der Verschiebung von Helligkeitsmustern zwischen aufeinanderfolgenden Bildern, nun auf der gesamten beobachtbaren Transformation.

Als radiometrische Transformation wird eine lineare Transformation der Helligkeitswerte zwischen den Bildern angenommen. Mathematisch kann das folgendermaßen formuliert werden (nach [Gennert and Negahdaripour, 1987]):

$$E\left(r+\delta r, t+\delta t\right) = M\left(r,t\right) E\left(r,t\right) + C\left(r,t\right) \tag{2.31}$$

Für kleine δt erwartet man nur kleine Helligkeitsänderungen, daher sollte M nahe 1 sein und C nahe 0. Da es sich um inkrementelle Änderungen handelt, kann man schreiben: $M = 1 + \delta m$ und $C = \delta c$. Für $\delta t \to 0$ gilt sowohl $\delta m \to 0$ als auch $\delta c \to 0$. Daher kann man die zeitlichen Ableitungen m_t und c_t folgendermaßen definieren:

$$m_t = \lim_{\delta t \to 0} \frac{\delta m}{\delta t} \tag{2.32}$$

$$c_t = \lim_{\delta t \to 0} \frac{\delta c}{\delta t} \tag{2.33}$$

Gleichung 2.31 wird so zu

$$E\left(r+\delta r, t+\delta t\right) = \left[1+\delta m\left(r,t\right)\right] E\left(r,t\right) + \delta c\left(r,t\right) \tag{2.34}$$

Die Taylor-Entwicklung erster Ordnung von $E\left(r+\delta r, t+\delta t\right)$ im Punkt r lautet:

$$E\left(r+\delta r, t+\delta t\right) = r\left(r,t\right) + \frac{\partial E}{\partial r}\delta r + \frac{\partial E}{\partial t}\delta t + O\left(\epsilon\right) = E_r \delta r + E_t \delta t + O\left(\epsilon\right) \tag{2.35}$$

Setzt man dies Gleichung 2.34 ein und vereinfacht, so erhält man:

$$E_r \delta r + E_t \delta t - E\delta m - \delta c + O\left(\epsilon\right) = 0 \tag{2.36}$$

Division durch δt und Betrachtung von $\delta t \to 0$ führt zu

$$E_t + E_r r_t - E m_t - c_t = 0 \tag{2.37}$$

Dies ist die von Gennert und Negahdaripour vorgeschlagene *optical flow constraint equation*. Im Falle $M = 1$ und $C = 0$ (dh $m_t = c_t = 0$) entspricht das Formulierung von Horn & Schunck [Horn and Schunck, 1981].

Die *smoothness*-Annahme wird im *generalized dynamic image model* beibehalten. Sie wird dadurch verwirklicht, dass man annimmt, dass die gesuchten

Flussfelder innerhalb kleiner Bereiche um jedes Pixel konstant sind. Eine Lösung für δx, δy, δm und δc kann man unter dieser Annahmen dann aus

$$\sum_W \begin{bmatrix} E_x^2 & E_xE_y & -E_xE & -E_x \\ E_xE_y & E_y^2 & -E_yE & -E_y \\ -E_xE & -E_yE & E^2 & E \\ -E_x & -E_y & E & 1 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \\ \delta m \\ \delta c \end{bmatrix} = \sum_W \begin{bmatrix} -E_xE_t \\ -E_yE_t \\ EE_t \\ E_t \end{bmatrix} \tag{2.38}$$

bestimmen. $E_t = \frac{\partial E}{\partial t}$ ist die Helligkeitsänderung zwischen den beiden Bildern, E_x ist der Gradient in x-Richtung, E_y der Gradient in y-Richtung und E ist die Helligkeit des ersten Bildes in diesem Pixel. W ist die Nachbarschaftsregion (ein 9x9-Fenster um jedes Pixel). Diese Annahme von *smoothness* ist richtig für Bildbereiche die glatte Oberflächen darstellen. Dort ergibt sie robuste Ergebnisse. Sie ist nicht richtig für Kanten an Bereichen mit 3D-Tiefenunterschieden oder unterschiedlicher 3D-Bewegungsparameter. Die Herleitung von Gleichung 2.38 geht analog zur Herleitung der Lukas-Kanade-Methode zur Bestimmung des optischen Fluss (Abschnitt 2.1.8). Um den optischen Fluss bestimmen zu können, muss auch hier die Matrix in Gleichung 2.38 invertierbar sein.

2.1.10 Pyramidaler Ansatz

Die in den beiden vorigen Abschnitten erläuterten Verfahren zur Berechnung von optischem Fluss können pyramidal implementiert werden. Eine pyramidale Repräsentation eines Bildes bedeutet, das Bild in L_m rekursiv ermittelten Vergröberungen darzustellen. Pro Vergröberungsschritt halbiert sich die Auflösung des Originalbildes. Dies geschieht, indem man iterativ einen Tiefpassfilter auf das Bild anwendet. Die unterste Ebene L_m der pyramidalen Repräsentation ist dann die am stärksten vergröberte Ebene, die oberste Ebene 0, das originale Bild. In Abbildung 2.7 ist eine schematische Darstellung der pyramidalen Bildrepräsentation gegeben.
Die rekursive Berechnungsvorschrift für Ebene I^{L-1} aus Ebene I^L ist gegeben durch:

$$\begin{aligned} I^L(x,y) \;=\; & \frac{1}{4}\, I^{L-1}(2x,2y) \\ & +\frac{1}{8}\, \left[I^{L-1}(2x-1,2y) + I^{L-1}(2x+1,2y)\right] \\ & +\frac{1}{8}\, \left[I^{L-1}(2x,2y-1) + I^{L-1}(2x,2y+1)\right] \\ & +\frac{1}{16}\, \left[I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y+1)\right] \\ & +\frac{1}{16}\, \left[I^{L-1}(2x-1,2y+1) + I^{L-1}(2x+1,2y-1)\right] \end{aligned} \tag{2.39}$$

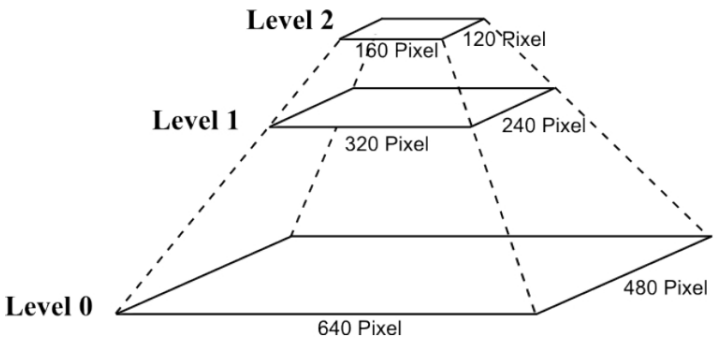


Abbildung 2.7: Konzept der pyramidalen Repräsentation (aus [Blazek, 2009])

Der Vorteil, der sich für die Bestimmung des optischen Flusses $\vec{v}$ anhand der pyramidalen Repräsentation der Eingabebilder ergibt, ist folgender: Man berechnet den optischen Fluss zuerst auf unterster pyramidaler Ebene. Das Ergebnis propagiert man in Form einer initialen Schätzung für den optischen Fluss auf die nächsthöhere Bildebene. Bei der Berechnung des optischen Fluss auf jeder Ebene wird die Größe des verwendeten Integrationsfensters w konstant gehalten. Relativ zur sich verkleinernden Auflösung des Bildes innerhalb der Pyramide (ausgehend vom Originalbild) nimmt die Größe des Integrationsfensters somit zu. Der auf untersten Pyramidenebene berechnete Fluss ist klein. Propagiert man ihn aber in der Pyramide nach oben wird er immer größer. Dadurch lassen sich durch eine pyramidale Implementation viel größere Pixelverschiebungen handhaben als durch die Berechnung des Flusses nur auf den Originalbildern.

Wenn man annimmt, dass jede elementare Fluss-Berechnung Pixel-Bewegungen der Größe d_{max} bewältigen kann, so kann die gesamte Pyramidale Berechnung Pixelbewegungen der Größe $d_{maxfinal} = \left(2^{L_m+1} - 1\right) d_{max}$ bewältigen. Für eine Wahl der Anzahl der pyramidalen Ebenen von beispielsweise $L_m = 3$ bedeutet das eine Vergrößerung der Länge des berechenbaren Flusses um Faktor 15.

Ein pyramidal implementierter Fluss-Algorithmus arbeitet in folgenden Schritten:

1. Berechne den optischen Fluss $\vec{v}^{L_m}$ auf der untersten pyramidalen Ebene L_m

2. Propagiere das dort berechnete Ergebnis auf Level L_{m-1} in Form einer

initialen Schätzung $\vec{g}^{L_m-1}$ für den Fluss $\vec{v}^{L_m-1}$:

$$\vec{g}^{L_m-1} = 2\left(\vec{g}^{L_m} + \vec{v}^{L_m}\right) \tag{2.40}$$

3. Mit Hilfe dieser initialen Schätzung wird der verfeinerte optische Fluss $\vec{v}^{L_m-1}$ auf Ebene L_{m-1} berechnet
4. Propagiere das Ergebnis auf Level L_{m-2}
5. ...etc. bis Level L_0 (=das Originalbild)

Den letztlich berechneten Optischen Fluss erhält man dann als Ergebnis der obersten Ebene $\vec{v} = \vec{g}^0 + \vec{v}^0$.
Der berechnete Gesamtfluss $\vec{v}$ ergibt sich also durch die Summe der Flussberechnungen auf jeder ebene jeweils multipliziert mit dem Vergrößerungsfaktor 2^L:

$$\vec{v} = \sum_{L=0}^{L_m} 2^L \vec{v}^L \tag{2.41}$$

2.2 Egomotion-Schätzung

Die Egomotion-Schätzung besteht darin, aus einem von einer bewegten Kamera aufgenommenen Bildpaar die Bewegungsparameter der Kamera zu ermitteln, das heißt die 6 Freiheitsgrade der dreidimensionalen Bewegung der Kamera zu bestimmen. Zur Lösung dieses Problems gibt es verschiedene Ansätze. (Für eine Auswahl an Ansätzen sei auf [Tian et al., 1996] verwiesen.) Der in dieser Diplomarbeit verwendete Ansatz ist der Kanatani-Algorithmus [Kanatani, 1993]. Da die Herleitung des Verfahrens sehr umfassend ist und die verwendete Implementierung außerdem schon vorlag, soll hier nur ein kurzer Überblick über das Verfahren gegeben werden. Eine ausführliche Herleitung ist in [Kanatani, 1993] zu finden.

Um die Bewegungsparameter der Kamera zu bestimmen, löst Kanatani die bezüglich der Essentialparameter (engl.: *essential parameters*) $(\mathbf{K}, \mathbf{v})$ und dem sogenannten „gedrehten“ optischen Fluss $\dot{\mathbf{m}}^*$ (engl.: *twisted optical flow*) formulierte Epipolargleichung:

$$(\dot{\mathbf{m}}^*, \nu) + (\mathbf{m}, \mathbf{Km}) = 0 \tag{2.42}$$

Dies geschieht durch eine Fehlerminimierung nach der Methode der kleinsten Quadrate. Die so erhaltene Lösung für die Translation ν enthält jedoch einen systematischen Fehler. Kanatani analysierte diesen Fehler mittels eines Gauß'schen Rauschmodels und entwickelte eine Methode (Kanatani nennt sie Renormalisierungs-Methode, engl.: *renormalization method*), die den unbekannten Fehler automatisch ausgleicht.

Die in dieser Diplomarbeit implementierte Trennung der EM- und IDM-Vektoren basiert auf dem Reprojektionsfehler, der sich nun für jeden Flussvektor bestimmen lässt.
Eine Kamerabewegung mit Rotationsgeschwindigkeit ω und Translationsgeschwindigkeit ν induziert folgenden optischen Fluss $\dot{\mathbf{m}}$:

$$\dot{\mathbf{m}} = -\omega \times \mathbf{m} - \frac{\mathbf{P}_m \nu}{r\,(\mathbf{m})} \tag{2.43}$$

Dabei ist $\mathbf{m}$ der 3D-Punkt, r(m) seine Tiefe und $\mathbf{P}_m = \mathbf{I} - \mathbf{m}\mathbf{m}^T$ die Projektionsmatrix.
Die Differenz zwischen beobachtetem und gemessenem Fluss, die hier *quality* ($\mathbf{m}$) genannt wird, beträgt daher:

$$quality\,(\mathbf{m}) = \dot{\mathbf{m}} - \left[-\omega \times \mathbf{m} - \frac{\mathbf{P}_m \nu}{r\,(\mathbf{m})}\right] \tag{2.44}$$

Je kleiner die Qualität, das heißt der Reprojektionsfehler, desto besser passt der Flussvektor zur geschätzten Egomotion.

Wichtig ist noch zu bemerken, dass der optische Fluss hierbei als zeitliche Ableitung des sogenannten N-Vektors [Kanatani, 1991a] [Kanatani, 1993] dargestellt wird. Dies ist der normierte Vektor, der vom Ursprung des Kamera-Koordinatensystems auf den auf den Startpunkt des Flusses im Bild weist (in Kamerakoordinaten). Dies wird auch sphärische Darstellung des Flusses genannt.
Um den Bildfluss in eine sphärische Darstellung zu überführen, muss ein in Bildkoordinaten gegebener Fluss mit Startpunkt $\mathbf{p}^i$ und Flussvektor $\mathbf{v}^i$ zuerst in Kamerakoordinaten umgewandelt werden:
$\mathbf{v}^i \mapsto \mathbf{v} = (v_x, v_y, 0)$
$\mathbf{p}^i \mapsto \mathbf{p} = (p_x, p_y, p_z)$
Zur Transformation der Bildkoordinaten benötigt man eine Kalibrierung der Kamera. Hier wurde die Kamerakalibrierung nach Tsai [Tsai, 1986] verwendet.
Der optische Fluss $\dot{\mathbf{m}}$ in sphärischer Darstellung ist dann gegeben durch:

$$\dot{\mathbf{m}} = \frac{1}{|\mathbf{p}|} \begin{pmatrix} v_x \\ v_y \\ 0 \end{pmatrix} - \frac{p_x v_x + p_y v_y}{|\mathbf{p}|} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \tag{2.45}$$

Da sich die Reprojektionsfehler auf die sphärische Darstellung beziehen, befinden sich die Werte meist zwischen Null und Eins.

2.3 Dynamisches *thresholding*

Unter *thresholding* versteht man die Auftrennung von Daten anhand eines Merkmals, typischerweise eines Zahlwerts, in zwei Klassen. Es wird ein Schwellwert festgelegt, der sogenannte *threshold*; alle Datenpunkte mit Merkmal kleiner oder gleich dem *threshold* werden der einen, alle Datenpunkte mit Merkmal größer dem *threshold* der anderen Klasse zugeordnet.
Eine typische Anwendung für *thresholding*-Methoden ist die Bildsegmentierung. Zumeist sollen Grauwert-Bilder in Vorder- und Hintergrund aufgetrennt werden.
Dynamische *thresholding*-Methoden wählen den *threshold* automatisch, abhängig von bestimmten Kriterien. Es gibt eine Vielzahl Dynamischer *thresholding*-Methoden, die verschiedene Eigenschaften des Bildes und seiner Grauwertverteilung zur Ermittlung des *thresholds* verwenden.
Sezgin und Sankur erfassten 2004 vierzig *thresholding*-Verfahren in einer kategorisierenden und vergleichenden Studie [Sezgin and Sankur, 2004]. Sie ordneten die Verfahren in sechs Gruppen - Histogramm-basierte, Clustering-basierte, Entropie-basierte, Ähnlichkeits-Merkmal-basierte, räumliche und lokale Verfahren - basierend auf dem Informationstyp, den ein Verfahren für das *thresholding* nutzt. Jede Gruppe lässt sich dann noch feiner untergliedern (siehe [Sezgin and Sankur, 2004]).
In dieser Diplomarbeit wurden Dynamische *thresholding*-Verfahren verwendet, um die EM von der IDM zu trennen. Drei Dynamische *thresholding*-Verfahren wurden implementiert: [Kittler and Illingworth, 1986], [Brink and Pendock, 1996] und [Yuan, 2004]. Die Methoden werden jeweils so beschrieben, wie sie die Autoren entwickelten und anwendeten, das heisst die ersten beiden zur Segmentierung von Graustufenbildern, die dritte Methode zur Segmentierung anhand von Grauwertdifferenzen. Eine Erläuterung der Wahl des Dynamischen *thresholdings* für die Trennung von EM und IDM und speziell der Auswahl dieser drei Verfahren wird in 3.1 gegeben.

2.3.1 Methode von Brink und Pendock

Die Methode von Brink und Pendock [Brink and Pendock, 1996] gehört zur Kategorie der Entropie-basierten *thresholding*-Methoden [Sezgin and Sankur, 2004]. Bei der Brink-Pendock-Methode wird die Kreuzentropie (engl.: *cross-entropy*) zwischen dem Eingabe-Grauwert-Bild und dem Ausgabe-Binärbild minimiert. Die Kreuzentropie wird dabei als ein Maß für die Datenkonsistenz zwischen dem originalen und dem binarisierten Bild interpretiert.

Die Kreuzentropie wird auch Relative Entropie oder Kullback-Leibler-Divergenz [Kullback, 1959] genannt und ist ein Maß für die Unterschiedlichkeit zweier Wahrscheinlichkeitsverteilungen desselben Ereignishorizonts. Sie erreicht ihr Minimum für $q = p$. Typischerweise repräsentiert p Beobachtungen oder eine präzise Wahrscheinlichkeitsverteilung, während q ein Modell oder eine Approximation darstellt.

Die Kreuzentropie H_{CE} zwischen einer diskreten *a priori*-Wahrscheinlichkeitsverteilung p_x und einer diskreten *a posteriori*-Wahrscheinlichkeitsverteilung q_x ist gegeben durch

$$H_{CE}(q,p) = \sum_{x=1}^{N} q_x log \frac{q_x}{p_x} \tag{2.46}$$

mit

$$\sum_{x=1}^{N} p_x = \sum_{x=1}^{N} q_x \,(=1) \tag{2.47}$$

dabei ist N die Anzahl der diskreten möglichen Realisationen der Verteilung. Die so definierte Kreuzentropie ist ein keine Metrik, da $H_{CE}(p,q) \neq H_{CE}(q,p)$. Sie lässt sich in eine Metrik umformulieren, die wir hier jedoch nicht benötigen (für nähere Informationen siehe [Brink and Pendock, 1996]).

Die *a priori*-Verteilung p entspricht hier dem originalen Grauwert-Bild, die *a posteriori*-Verteilung q dem zweiwertigen Ergebnis der Segmentierung.

Ein digitales Bild kann als diskrete Wahrscheinlichkeitsverteilung betrachtet werden, in der der Grauwert jedes Pixels die Wahrscheinlichkeit, dass ein Photon dieses spezielle Pixel trifft, widerspiegelt. Für die Auswahl des *thresholds* betrachten wir das originale Grauwertbild als die *a priori* Verteilung, jeder Grauwert g repräsentiert eine Häufigkeit (bzw eine Wahrscheinlichkeit, falls die Grauwerte normalisiert sind). Der Mittelwert des Bereiches unterhalb des *thresholds* heißt $\mu_0(T)$, der Mittelwert des Bereiches oberhalb $\mu_1(T)$. Diese beiden kann man als die beiden Grauwerte des binarisierten Bildes verwenden. Die Mittelwerte sind gegeben durch:

$$\mu_0(T) = \sum_{g=a}^{T} g \frac{f_g}{N_a^T} \tag{2.48}$$

und

$$\mu_1(T) = \sum_{g=T+1}^{b} g \frac{f_g}{N_{T+1}^b} \tag{2.49}$$

Diese beiden Formeln wurden korrigiert, da die von Brink und Pendock aufgeführten Formeln fehlerhaft waren.

wobei a und b den kleinsten bzw. größten im originalen Grauwertbild vorhandenen Grauwert repräsentieren, T ist der Wert des *thresholds*, f_g gibt die Häufigkeit an, mit der der Grauwert g vorkommt, und N_a^T beziehungsweise N_{T+1}^b die Anzahl der Pixel mit Grauwerten von a bis T beziehungsweise $T+1$ bis b.

Wenn man die Mittelwerte aus den Gleichungen 2.48 und 2.49 für das Binärbild verwendet, so erfüllt dies die Bedingung für Gleichung 2.46. Das heisst

$$\sum_{i=1}^{N} g_i = \left[\sum_{i=1}^{N} \mu_0(T)\right]_{g_i \leq T} + \left[\sum_{i=1}^{N} \mu_1(T)\right]_{g_i > T} \tag{2.50}$$

Für digitale Bilder kann man Gleichung 2.46 dann folgendermaßen schreiben:

$$H_{CE}(T) = \left[\sum_{i=1}^{N} \mu_0(T)\, log \frac{\mu_0(T)}{g_i}\right]_{g_i \leq T} + \left[\sum_{i=1}^{N} \mu_1(T)\, log \frac{\mu_1(T)}{g_i}\right]_{g_i > T} \tag{2.51}$$

Anstatt über alle Pixel zu summieren kann man auch ein Histogramm der Grauwert-Frequenzen verwenden. Dies erhöht die Berechnungsgeschwindigkeit. Jedes *bin* eines solchen Histogramms stellt dann die Häufigkeit f_g dar, mit der dieser Grauwert im Bild auftaucht, das heisst die Anzahl der Pixel mit dem Grauwert g.

Gleichung 2.51 unter Verwendung eines Grauwert-Histogramms lautet:

$$H_{CE}(T) = \sum_{g=a}^{T} f_g \mu_0(T)\, log \frac{\mu_0(T)}{g} + \sum_{g=T+1}^{b} f_g \mu_1(T)\, log \frac{\mu_1(T)}{g} \tag{2.52}$$

Die einfachste Art und Weise, diesen Term zu minimieren ist es, den *threshold* über alle im Bild vorhandenen Grauwerte zu iterieren. Diese Methode wurde auch in der vorliegenden Diplomarbeit gewählt. Für eine kleine Anzahl an *bins* ist dieses Vorgehen auch durchaus praktikabel.

2.3.2 Methode von Kittler und Illingworth

Das Verfahren von Kittler und Illingworth [Kittler and Illingworth, 1986] gehört zur Kategorie der *clustering*-basierten *thresholding*-Methoden [Sezgin and Sankur, 2004]. Bei diesen Methoden wird eine Clusteranalyse mit zwei Clustern durchgeführt. Innerhalb der *clustering*-basierten Methoden gehört die Methode von Kittler und Illingworth zur Untergruppe der *minimum error thresholding*-Methoden. Diese Methoden nehmen an, dass das Bild über eine Misch-Verteilung der Vorder- und Hintergrundgrauwerte beschrieben werden kann.

Man betrachtet das Histogramm $h(g)$ der Grauwerte g im Bild, setzt einen beliebigen *threshold* T und modelliert jede der dadurch entstehenden beiden Pixelpopulationen durch eine Normalverteilung $h(g|i,T)$ mit Mittelwert $\mu_i(T)$ und Standardabweichung $\sigma_i(T)$ und einer *a priori*-Wahrscheinlichkeit $P_i(T)$ für $i = \{1,2\}$, gegeben durch:

$$P_i(T) = \sum_{g=a}^{b} h(g) \tag{2.53}$$

$$\mu_i(T) = \left[\sum_{g=a}^{b} h(g)\, g\right] / P_i(T) \tag{2.54}$$

$$\sigma_i^2(T) = \left[\sum_{g=a}^{b} \{g - \mu_i(T)\}^2 h(g)\right] / P_i(T) \tag{2.55}$$

mit

$$a = \begin{cases} 0 & i = 1 \\ T+1 & i = 2 \end{cases} \tag{2.56}$$

und

$$b = \begin{cases} T & i = 1 \\ n & i = 2 \end{cases} \tag{2.57}$$

Dann ist die bedingte Wahrscheinlichkeit $e(g,T)$, dass ein Pixelgrauwert g der richtigen Klasse zugewiesen wird, gegeben durch:

$$e(g,T) = \frac{h(g|i,T) \cdot P_i(T)}{h(g)} \qquad i = \begin{cases} 1 & g \leq T \\ 2 & g > T \end{cases} \tag{2.58}$$

Da der Nenner $h(g)$ von t und i unabhängig ist, lässt man ihn ausser acht und betrachtet nur den Zähler. Nimmt man davon den Logarithmus und multipliziert das Ergebnis mit -2, so erhält man

$$\epsilon(g,T) = \left[\frac{g - \mu_i(T)}{\sigma_i}\right]^2 + 2log\sigma - i(T) - 2logP_i(T) \quad i = \begin{cases} 1 & g \leq T \\ 2 & g > T \end{cases} \tag{2.59}$$

Man kann dies als Maß für die Güte der Klassifizierung des Grauwerts g betrachten.
Summiert man dieses Maß über alle vorkommenden Grauwerte multipliziert mit der Häufigkeit ihres Auftretens, so erhält man die Zielfunktion

$$J(T) = \sum_g h(g) \cdot \epsilon(g,T) \tag{2.60}$$

Diese Zielfunktion spiegelt für einen bestimmten *threshold* T die Größe des überlappenden Bereichs zwischen den beiden modellierten Normalverteilungen des Vorder- und des Hintergrunds wieder (siehe Abbildung 2.8). Je besser die Modelle die Beschaffenheit der Daten zu beschreiben vermögen, desto kleiner sollte der überlappende Bereich sein. Daher muss die Zielfunktion minimiert werden.
Trennt man $J(T)$ in zwei Summanden bezüglich der beiden Modelle und setzt man die Gleichungen 2.53, 2.54 und 2.55 ein, so kommt man zu folgender Form der Zielfunktion.

$$\begin{aligned} J(T) = 1 &+ 2\left[P_1(T)\, log\, \sigma_1(T) + P_2(T)\, log\, \sigma_2(T)\right] \\ &-2\left[P_1(T)\, log\, P_1(T) + P_2(T)\, log\, P_2(T)\right] \end{aligned} \tag{2.61}$$

Diese Zielfunktion lässt sich nach [Kittler and Illingworth, 1986] über folgenden iterativen Algorithmus minimieren:

- Wähle einen beliebigen initialen *threshold* T
- Berechne $\mu_i(T)$, $\sigma_i(T)$ und $P_i(T)$ für $i = \{1,2\}$.

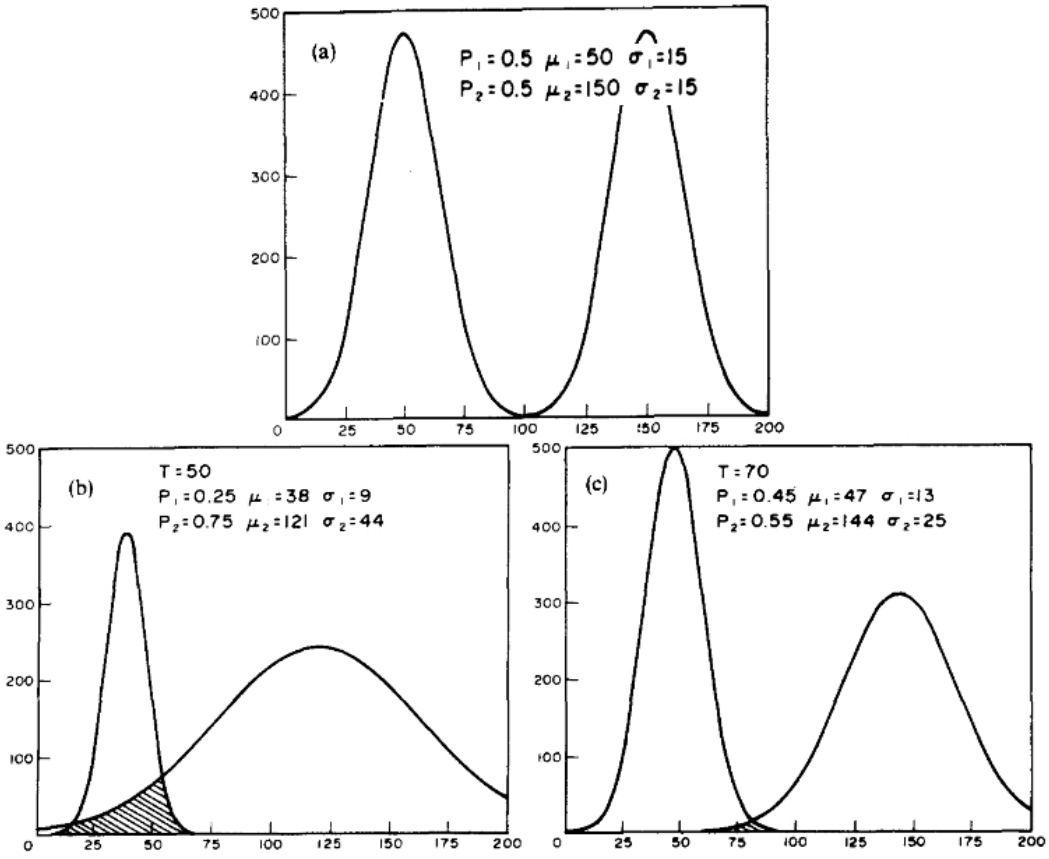


Abbildung 2.8: (a) Grauwert-Histogramm aus zwei Gauss-Verteilungen mit denselben Parametern. Der optimale *threshold* liegt bei $T = 100$ (b,c)Die Modell-Verteilungen für $T = 50$ und $T = 70$. Der Binarisierungsfehler entspricht dem schraffierten Bereich, in dem die Modell-Dichtefunktionen überlappen. (aus [Kittler and Illingworth, 1986])

- Berechne den verbesserten *threshold* durch Lösen von Gleichung 2.62

$$g^2\left[\frac{1}{\sigma_1^2(T)}-\frac{1}{\sigma_2^2(T)}\right]-2g\left[\frac{\mu_1(T)}{\sigma_1^2(T)}-\frac{\mu_2(T)}{\sigma_2^2(T)}\right]$$
$$+\frac{\mu_1^2(T)}{\sigma_1^2(T)}-\frac{\mu_2^2(T)}{\sigma_2^2(T)}+2\left[log\,\sigma_1(T)-log\,\sigma_2(T)\right] \qquad (2.62)$$
$$-2\left[logP_1(T)-logP_2(T)\right]=0$$

- Falls der neue *threshold* gleich dem alten *threshold* ist, so ist der optimale *threshold* gefunden, ansonsten gehe zu Schritt 2.

Der Algorithmus konvergiert in unserem Fall meist nach wenigen Iterationen.

2.3.3 Methode von Yuan

Das Verfahren von Yuan [Yuan, 2004] lässt sich den Histogramm-basierten Methoden zuordnen. Yuan verwendete ein automatisches *thresholding*-Verfahren, um die Grauwert-Differenz zwischen zwei aufeinanderfolgenden Bildern aufgenommen von einer statischen Kamera in Vorder- und Hintergrund zu unterteilen. Der Hintergrund wird dabei als statisch angenommen, der Vordergrund ist als sich bewegend definiert. Um die Trennung durchzuführen wird ein zweikomponentiges Modell für die Grauwert-Differenz-Dichtefunktion angenommen, nämlich aus dem statischen Hintergrund, und dem sich bewegenden Vordergrund. Diese beide Komponenten können durch ein statistisches Verfahren getrennt werden. Für jedes Pixel soll bestimmt werden, welcher Komponente es angehört.
Die Trennung wird anhand eines Histogramms der Differenz-Werte durchgeführt. Für jedes Bild-Pixel wird die Differenz zwischen dem Grauwert des ersten Bildes an Bildposition und dem Grauwert des zweiten Bildes an dieser Bildposition berechnet. Aus diesen Grauwert-Differenzen wird ein Histogramm erstellt. Das Histogramm der Grauwert-Differenzen zwischen den beiden Bildern ist gegeben durch $p_i = \frac{n_i}{N}$, wobei N die Anzahl der Pixel im Bild und n_i die Anzahl der Pixel, deren Grauwert-Differenz gleich i ist. Das Unterscheidungs-Problem zwischen Hinter- und Vordergrund besteht nun darin, den optimalen Wert k zu finden, so dass alle Pixel, die zum Vordergrund gehören einen Wert kleiner als k und alle Pixel die zum Hintergrund gehören einen Wert größer als k besitzen.
Sei Ω_0 die Klasse der Hintergrund-Pixel, Ω_1 die Klasse der Vordergrund-Pixel. Die Wahrschlichkeitsdichtefunktionen der beiden Klassen lauten:

$$P(\Omega_0) = \sum_{i=0}^{k} p_i \tag{2.63}$$

$$P(\Omega_1) = \sum_{i=k+1}^{m} p_i \tag{2.64}$$

Wobei m die maximal vorkommende Grauwert-Differenz ist.
Der Mittelwert der beiden Klassen lässt sich dann folgendermaßen berechnen:

$$\mu(\Omega_0) = \sum_{i=0}^{k} \frac{ip_i}{P(\Omega_0)} \tag{2.65}$$

$$\mu(\Omega_1) = \sum_{i=k+1}^{m} \frac{ip_i}{P(\Omega_1)} \tag{2.66}$$

Der Mittelwert des gesamten Bildes ist gegeben durch:

$$\mu = \sum_{i=0}^{m} ip_i \tag{2.67}$$

Eine optimale Trennung der beiden Klassen sollte die mittlere quadratische Differenz zwischen den beiden Klassen maximieren. Also:

$$J_i = P(\Omega_0)(\mu(\Omega_0) - \mu)^2 + P(\Omega_1)(\mu(\Omega_1) - \mu)^2 \tag{2.68}$$

Den optimalen Wert k erhält man dann als

$$k = \text{argmax}_i \{J_i\} \tag{2.69}$$

Wieder ist die einfachste Art und Weise, diesen Term zu minimieren, den *threshold* über alle m möglichen Werte zu iterieren.

2.4 *Clustering*

2.4.1 KMeans

Der *kmeans*-Algorithmus ist ein *clustering*-Verfahren bei dem m Datenpunkte in einem N-dimensionalen Raum in k Partitionen unterteilt werden sollen. Jede Partition wird durch ihren Mittelwert repräsentiert. Die Anzahl der Cluster k muss vorher festgelegt werden. Beim *kmeans*-Algorithmus wird die Gesamt-Summe der Distanzen der Datenpunkte zu ihrem jeweils zugeordneten Zentrum minimiert. Oder anders ausgedrückt: die Varianz zwischen den Partitionen soll maximiert, die Varianz innerhalb der Partitionen minimiert werden. Es können verschiedene Distanz-Maße verwendet werden. In dieser Arbeit wurde die euklidische Distanz verwendet.

Der grundlegende *kmeans*-Algorithmus verläuft folgendermaßen [Hartigan, 1975]:

Gegeben: Datenmenge X, Clusteranzahl k

1. Initialisiere k Cluster [1].
2. Ermittle zu jedem der k Cluster das Clusterzentrum c_k als Mittelwert der dem Cluster zugeordneten Datenpunkte x_i.
3. Ermittle zu jedem Datenpunkt das nächstgelegene Zentrum und ordne ihn diesem Cluster zu.

[1] Es gibt verschieden Verfahren zur Clusterinitialisierung. Standard-*kmeans* verwendet Zufallsinitialisierung. Ein Einblick in andere Initialisierungsmethoden wird in Abschnitt 2.4.3 gegeben

4. Berechne die neuen Clusterzentren.

5. Gehe zu 2.
   Solange wiederholen bis keine Zuordnungsänderungen mehr auftreten.

Die Standard-Version verwendet eine Zufallsinitialisierung der Cluster, das heisst, jeder Datenpunkt wird zufällig einem der k Cluster zugewiesen. Da die Güte des Ergebnisses des *kmeans*-Algorithmus aber stark von der Initialisierung abhängt, kann das Ergebnis bei Zufallsinitialisierung stark variieren. Es gibt diverse andere Methoden, die Cluster zu initialisieren (siehe Abschnitt 2.4.3, [He et al., 2004], [Redmond and Heneghan, 2007], [Pena et al., 1999], [Al-Daoud and Roberts, 1996]).

2.4.2 *Learning the k* : der *G-means*-Algorithmus

„*Learning the k in kmeans*“ ist der Titel eines Artikels von Hamerly und Elkan [Hamerly and Elkan, 2003]. Häufig ist die passende Anzahl k der Partitionen beim *clustering* nicht bekannt. Wählt man die Anzahl unpassend, so wird das Ergebnis suboptimal (Abbildung 2.9). Hamerly und Elkan entwickelten 2003 einen Algorithmus, der durch wiederholtes Durchführen eines Hypothesentests die passende Anzahl der Cluster während dem *clustering* ermittelt [Hamerly and Elkan, 2003]. Dabei wird angenommen, dass bei richtiger Partitionierung der Daten jedes Cluster einer Gauss-Verteilung um sein Zentrum folgt. Der Algorithmus startet mit einer kleinen Anzahl von Clustern. Das *clustering* wird mit *k-means* durchgeführt. In jeder Iteration werden diejenigen Cluster, die nicht einer Gauss-Verteilung zu folgen scheinen, in zwei Cluster aufgeteilt.

Die Clusteranzahl wird erhöht, indem ein Cluster in zwei Cluster aufgeteilt wird. Ob diese Aufteilung sinnvoll ist, wird mittels eines Hypothesentests, dem Anderson-Darling-Test, überprüft. Es finden solange Aufteilungen statt, wie dies mit dem Hypothesentest als sinnvoll erachtet wird. Man kann die Methode prinzipiell mit jedem *clustering*-Verfahren kombinieren, um die richtige Anzahl von Clustern zu ermitteln.

Der *G-means* (*Gaussian-means*) - Algorithmus nach Hamerly und Elkan:
gegeben: Datenmenge X

1. Wähle Startwert für Clusteranzahl k

2. Partitioniere X mit *kmeans* und Clusteranzahl k

3. Sei $\{x_i \mid \text{Klasse von } x_i = j\}$ $j = 1, \cdots, k$ die Menge der Datenpunkte, welche Cluster j zugeordnet wurden.

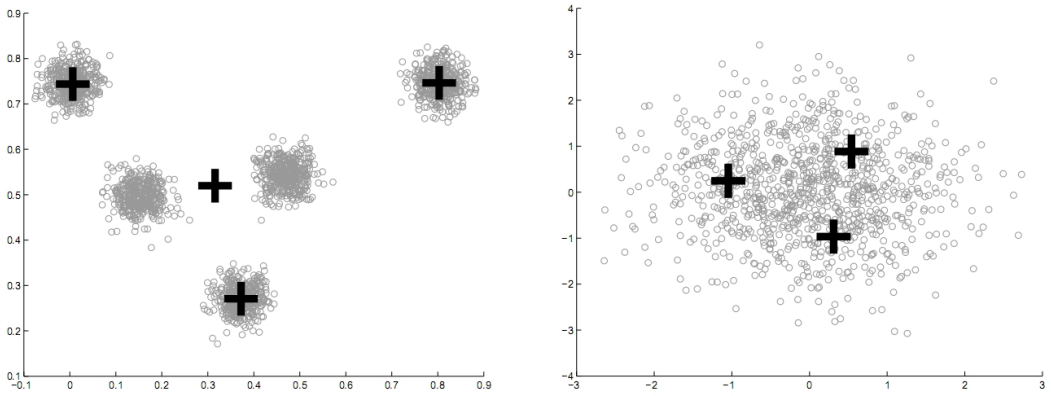


Abbildung 2.9: Zwei *kmeans*-Ergebnisse, bei denen die Clusteranzahl k unpassend gewählt wurde. Die schwarzen Kreuze sind die von *kmeans* ermittelten Clusterzentren. Links wurde die Clusteranzahl zu klein gewählt, $k = 5$ wäre optimal gewesen. Rechts wurde die Clusteranzahl zu groß gewählt, $k = 1$ wäre optimal gewesen.

4. Überprüfe mittels eines statistischen Tests, ob alle $\{x_i \mid \text{Klasse von} x_i = j\}$ einer Normalverteilung um ihr Zentrum c_j genügen.

5. Sind die Daten normalverteilt, so behalte c_j bei, ansonsten ersetze c_j durch zwei neu initialisierte Clusterzentren.

6. Gehe zu Schritt 2, bis keine neuen Clusterzentren mehr hinzukommen.

Der statistische Test, der hierbei verwendet wird, ist der Anderson-Darling-Test. Mit dem Anderson-Darling-Test kann man prüfen, ob gegebene Daten aus einer bestimmten gewählten Verteilung, in unserem Fall aus einer Normalverteilung stammen. Der Anderson-Darling-Test wird im Abschnitt 2.5 ausführlich beschrieben.

Der Test zur Überprüfung, ob ein ermitteltes Cluster C mit Datenpunkten X und Clusterzentrum c weiter unterteilt werden sollte, verläuft folgendermaßen (siehe auch Abbildung 2.10):

1. Lege das Signifikanzniveau α für den statistischen Test fest (Abschnitt 2.5).

2. Initialisiere zwei neue Zentren für Cluster C.

3. Lasse *kmeans* mit diesen beiden Zentren laufen. Dann sind c_1 und c_2 die Zentren der durch *kmeans* ermittelten Partition.

4. Sei $v = c_1 - c_2$ der d-dimensionale Vektor, welcher die beiden Zentren verbindet. Entlang dieser Richtung kann am besten diskriminiert werden.

Projiziere X orthogonal auf v: $x_i^{'} =< x_i, v > /||v||^2$. Dann ist X' eine 1-dimensionale Repräsentation der Daten projiziert auf v. Transformiere X' in eine Standardnormalverteilung [2].

5. Sei $z_i = F(x'_{(i)})$ [3]. Wenn $A^2_*(Z)$ [4] im nicht-kritischen Bereich des Konfidenzintervalls mit Signifikanzniveau α liegt, so nehme eine Normalverteilung der Clusterpunkte um ihr Zentrum an, behalte das ursprüngliche Zentrum bei und verwerfe c_1 und c_2. Ansonsten verwerfe das ursprüngliche Zentrum und behalte c_1 und c_2.

In Abbildung 2.10 ist eine schematische Darstellung der Arbeitsweise des *G-means*-Algorithmus gegeben. Alle roten Datenpunkte gehören zu einem Cluster, dargestellt durch die blaue Ellipse (**a)**). Man möchte testen, ob die Verteilung der Datenpunkte aus einer Normalverteilung stammt und die momentane Partitionierung somit gut ist, oder ob das Cluster weiter unterteilt werden sollte. Zuerst ermittelt man die zur Zuordnung der Datenpunkte aussagekräftigste Richtung: Man lässt das Cluster in **a)** durch *kmeans* mit $k = 2$ partitionieren. **b)** zeigt das Ergebnis der Partitionierung. Der violette und der blaue Punkt zeigen die Clusterzentren an. Nun legt man eine Gerade durch die beiden Zentren (Abb. **c)**). Dies ist die Richtung, in der am besten getrennt werden kann. Man projiziert die Datenpunkte orthogonal auf diese Gerade (**d)**). Dadurch ist eine Dichteverteilung der projizierten Datenpunkte auf der Geraden gegeben, approximiert darstellbar durch ein Histogramm. Die Verteilung wird zu Mittelwert 0 und Standardabweichung 1 transformiert. Nun kann man mit einem statistischen Test abschätzen, wie wahrscheinlich es ist, dass die Daten aus einer Normalverteilung stammen. (**f)** zeigt das Histogramm zusammen mit der Gauss-Kurve, mit der verglichen werden soll. Im Falle der Abbildung **f)** stammen die Daten, wie man leicht sieht, mit Sicherheit nicht aus einer Normalverteilung. In diesem Falle wird das Zentrum des Ursprungs-Clusters durch die beiden von *kmeans* ermittelten Zentren ersetzt. Falls die projizierten Daten aus einer Normalverteilung stammen, so behalte das alte Cluster (**a)**) bei. Auf diese Art und Weise werden rekursiv alle Cluster, deren Datenpunkte nicht normalverteilt sind, weiter aufgesplittet.

[2] Transformation normalverteilter Daten X in standardnormalverteilte Daten Z: Sei $\text{mean}(X)$ der Mittelwert, $\text{stdDev}(X)$ die Standardabweichung der Daten X. Dann sind die transformierten Daten Z mit $z_i = \frac{x_i - \text{mean}(X)}{\text{stdDev}(X)}$ normalverteilt mit $\text{mean}(Z) = 0$ und $\text{stdDev}(Z) = 1$

[3] F ist die $N(0,1)$ - Kumulative Verteilungsfunktion (*cumulative distribution function* = CDF), z_i gibt dann die Wahrscheinlichkeit an, dass eine reellwertige Zufallsvariable S einen Wert kleiner oder gleich x_i annimmt.

[4] Der durch den Anderson-Darling-Test ermittelte Testwert (Erläuterung siehe Abschnitt 2.5)

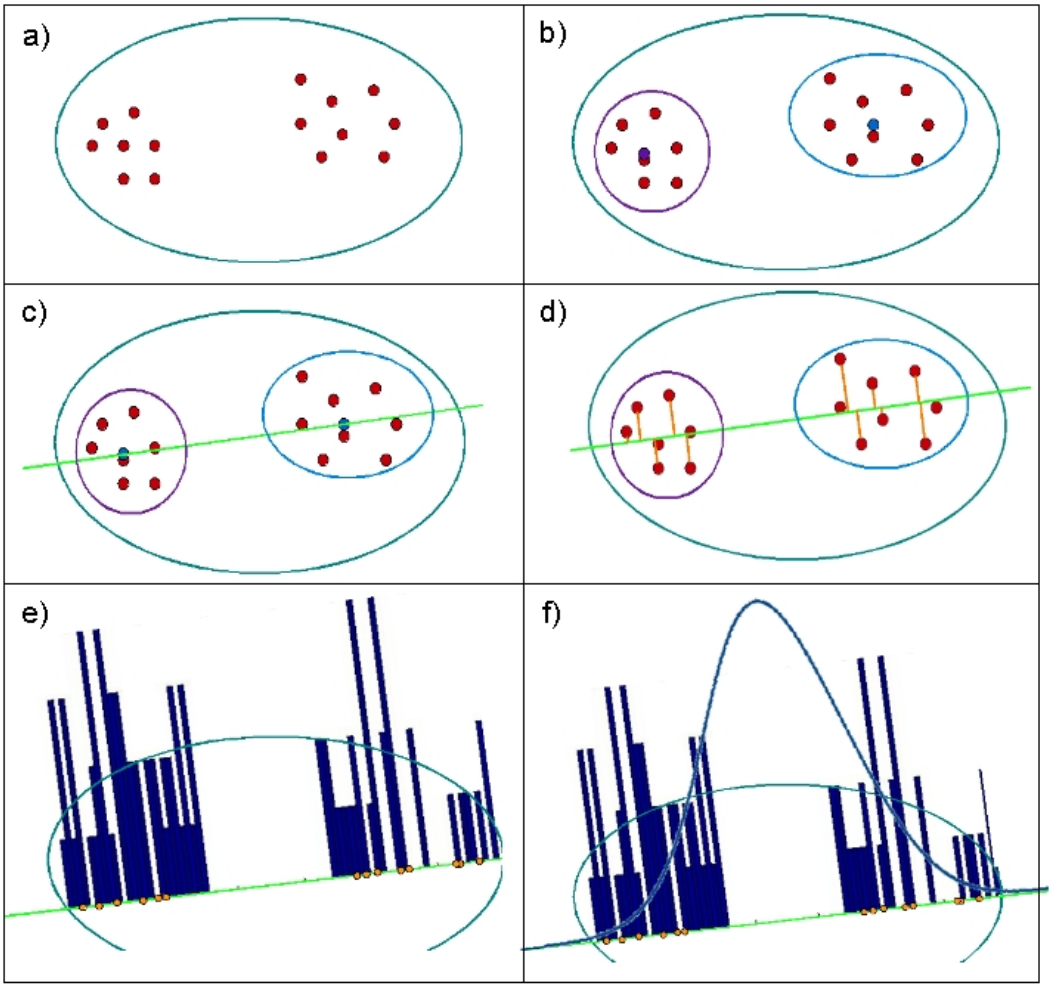


Abbildung 2.10: Schematische Darstellung der Arbeitsweise des *G-means*-Algorithmus. Beschreibung siehe Text.

2.4.3 Clusterinitialisierung

In der Standard-Version von *kmeans* werden die Cluster zufallsinitialisiert. Dies hat zur Folge, dass die Ergebnisse je nach Initialisierung sehr unterschiedlich sein können. Es gibt verschiedene andere Methoden zur Cluster-Initialisierung. Diese können das Ergebnis verbessern und die Güte der Klassifikation stabilisieren.

He et. al. untersuchten 2004 in einer Studie verschiedene Clusterinitialisierungsmethoden in Kombination mit *kmeans* und evaluierten, wie sehr sie das Ergebnis von *kmeans* zu verbessern vermögen [He et al., 2004]. Sie ermittelten zwei Methoden, die in Kombination mit *kmeans* die besten Ergebnisse lieferten: *Simple Cluster Seeking* (SCS) und die Methode nach Katsavounidis, Kuo

und Zhang [Katsavounidis et al., 1994] (KKZ, wie sie seit [Al-Daoud and Roberts, 1996] genannt wird). He et. al. testeten die Initialisierungsmethoden in Kombination mit *kmeans* an verschiedenen artifiziellen, sowie *real-life*, *benchmark*-Datensätzen.
SCS besitzt gegenüber KKZ den Nachteil, dass es einen sensitiven Parameter besitzt, welcher das Ergebnis je nach Initialisierung beeinflusst. Aus diesem Grund habe ich mich für die KKZ-Methode entschieden, die im Folgenden beschrieben wird.

Katsavounidis et. al. [Katsavounidis et al., 1994] entwickelten ihren Algorithmus zur verbesserten Initialisierung eines Verfahrens zur Vektorquantisierung.

Der KKZ-Algorithmus verläuft folgendermaßen:

- Wähle denjenigen Datenpunkt x mit maximaler Norm $\|x\|_{max}$ als erstes Zentrum.
- Derjenige Datenpunkt, der den größten Abstand zum ersten Zentrum besitzt, wird als zweites Clusterzentrum gewählt.
- Das dritte Clusterzentrum wird dasjenige, das den größten Abstand zum näherliegenden der beiden ersten besitzt
- usw.

2.5 Statistische Tests: Die Anderson-Darling Statistik

Der Anderson-Darling-Test ist ein statistischer Test, mit dem es möglich ist, zu überprüfen, ob eine Menge von Datenpunkten aus einer bestimmten vorgegebenen Verteilung stammen.

Der Anderson-Darling-Test geht auf T.W. Anderson und D.A. Darling zurück, die ihn 1952 entwickelten [Anderson and Darling, 1952].

Seien $x_1, x_2, \cdots, x_n$ unabhängige Zufallsvariablen einer kontinuierlichen Verteilung $G(x)$, und sei $F_n(x)$ die empirische kumulative Verteilungsfunktion (engl.: *empirical cumulative distribution function* ECDF). Die Nullhypothese nimmt an, dass die Daten X aus der Verteilung $G(x)$ stammen, die Alternativhypothese besagt das Gegenteil.

Die Anderson-Darling-Statistik berechnet den Testwert A^2.

$$A^2 = -n - \frac{1}{n}\sum_{i=1}^{n}\left[(2i-1)\,ln\left(\phi\left(y_i\right)\right) + \left(2\left(n-i\right)+1\right)ln\left(1-\phi\left(y_i\right)\right)\right] \quad (2.70)$$

ϕ ist dabei die kumulative Verteilungsfunktion.

Falls dieser kleiner als der kritische Wert auf einem bestimmten Signifikanzniveau ist, so wird die Nullhypothese mit diesem Signifikanzniveau angenommen.

Die kritischen Werte sind beim Anderson-Darling-Test unterschiedlich, je nach Verteilung, auf die getestet werden soll. Die kritischen Werte liegen tabelliert vor für Normalverteilung, Logarithmische Normalverteilung, Exponentialverteilung, Weibull-Verteilung, Logistische Verteilung, sowie Extremwertverteilung vom Typ1.

Die Anderson-Darling-Statistik A^2 legt im Vergleich zu zwei anderen statistischen Tests auf Normalverteilung, dem Kramer-van-Mises-Test und dem Kolmogorov-Smirnoff-Test, mehr Gewicht auf die Randbereiche der Verteilung und ist so besser als diese beiden dazu geeignet, Abweichungen in den Randbereichen zu detektieren [Stephens, 1974].

Je nachdem, wieviel von $G(x)$ bekannt ist, müssen vier Fälle unterschieden werden [Stephens, 1974]:

- Fall 0: $G(x)$ ist kontinuierlich, alle Parameter sind bekannt.
- Fall 1: $G(x)$ ist eine Normalverteilung, die Standardabweichung σ^2 ist bekannt, der Mittelwert μ muss aus den Daten als deren Mittelwert geschätzt werden.
- Fall 2: $G(x)$ ist eine Normalverteilung, der Mittelwert μ ist bekannt, die Standardabweichung $\sigma^{2'}$ muss aus den Daten geschätzt werden durch: $\sigma^{2'} = \sum_i (x_i - \mu)^2 / n$
- Fall 3: Fall 1: $G(x)$ ist eine Normalverteilung, sowohl Mittelwert μ als auch die Standardabweichung σ^2 müssen aus den Daten geschätzt werden durch μ ist der Mittelwert der Daten, die Standardabweichung $\sigma^{2'}$ ist $\sigma^{2'} = \sum_i (x_i - \mu)^2 / n - 1$

Je nachdem, welcher der obigen Fälle auf die Testsituation zutrifft, muss der berechnete Testwert A^2 korrigiert werden [Stephens, 1974].

In der Anwendung des Anderson-Darling-Tests in dieser Diplomarbeit trifft Fall 3 zu. Man muss A^2 zu

$$A_*^2 = A^2 \cdot \left(1 + \frac{3}{4n} + \frac{9}{4n^2}\right) \tag{2.71}$$

korrigieren (nach [Duller, 2008]).

Der Testalgorithmus läuft folgendermaßen ab:

- Sortiere die Daten X in aufsteigender Reihenfolge.
- Berechne Mittelwert und Standard-Abweichung der Daten X
- Transformiere die Daten X nach Z mit Mittelwert 0 und Standardabweichung 1
- Berechne den Testwert nach Gleichung 2.70
- Korrigiere den Testwert nach Gleichung 2.71.
- Falls A_*^2 kleiner ist als 0.752 (der kritische Wert für die modifizierte Statistik mit Signifikanzniveau 5% (nach [Duller, 2008])), so akzeptiere H_0, ansonsten H_1.

2.6 RANSAC

RANSAC steht für *Random Sample Consensus*. Der RANSAC-Algorithmus ist eine von Fischler und Bolles entwickelte Methode, um die Parameter eines gegebenen Modells aus einer Menge fehlerbehafteter Daten zu schätzen [Fischler and Bolles, 1981]. Ein Vorteil des RANSAC-Algorithmus ist, dass dabei Ausreißer aus den Daten entfernt werden können. Der Algorithmus arbeitet folgendermaßen: Es wird zufällig eine kleine Untermenge aus den Gesamtdaten ausgewählt und damit das Modell initialisiert. Nun wird festgestellt, wieviele der übrigen Datenpunkte innerhalb eines vorher festzulegenden Toleranzbereichs zum aktuellen Modell passen. Diese passenden Datenpunkte werden *consensus set* genannt. Besitzt das *consensus set* eine Mindestgröße T, so wird es gespeichert. Dieser Vorgang wird iterativ N-mal wiederholt (N ist dabei vorher festzulegen). Aus dem größten *consensus set* werden dann die endgültigen Modellparameter bestimmt. Alle Datenpunkte, die nicht im letztendlichen *consensus set* liegen, sind mit hoher Wahrscheinlichkeit Ausreißer.

Der RANSAC-Algorithmus:

1. Wähle eine Untermenge von s Datenpunkten aus der Datenmenge S und initialisiere damit die Modellparameter
2. Bestimme diejenigen Datenpunkte S_i, die innerhalb eines festgelegten Toleranzbereichs K zum Modell liegen. Die Menge S_i nennt man *consensus set*. Sie enthält die zum Modell passenden Datenpunkte (die *inliers*).
3. Wenn die Anzahl der Datenpunkte in S_i größer ist, als T, so speichere das *consensus set*.
4. Gehe zu Punkt 1.

5. Nach N Iterationen werden mit den Datenpunkten aus dem größten *consensus set* S_i die endgültigen Modellparameter bestimmt.

Kapitel 3

Design und Implementierung

3.1 Motivation für das dynamische *thresholding*

Die Flussvektoren sollen durch dynamisches *thresholding* anhand ihrer Qualitätswerte in Egomotion und IDM klassifiziert werden. Die Qualitätswerte entsprechen hierbei der Differenz zwischen berechnetem und tatsächlich beobachteten Fluss (siehe auch 2.2). Wenn man den *threshold* für diese Unterteilung manuell in einige Bilder legt (siehe Abschnitt 4.1), stellt man fest, dass der korrekte *threshold* variiert. Die Unterschiedlichkeit beruht unter anderem auf Unterschieden in der Beleuchtung im Bild und des dargestellten Motivs. Beides beeinflusst die Güte des optischen Flusses, somit die Güte der Egomotion-Schätzung und daher wiederum die Verteilung der Qualitätswerte bzw. den durchschnittlichen Qualitätswert. Einige Beispiele für die Verschiedenheit der *threshold* sind in Abbildung 3.1 gegeben.

Der absolute Unterschied zwischen den *thresholds* erscheint auf den ersten Blick gering. Jedoch kann bereits eine kleine absolute Verschiebung des *thresholds* zu einer deutlichen Änderung in der Klassifizierung führen. Ein typisches Beispiel ist in Abbildung 3.2 gegeben: Das Bild oben zeigt die Klasse der als IDM klassifizierten Vektoren wenn der *threshold* auf $= 0.0037$ gesetzt ist. Die beiden unteren Bilder zeigen die Klasse der IDM-Vektoren bei einer leichten Verschiebung des *thresholds* nach unten (3.2 unten links) beziehungsweise nach oben (3.2 unten rechts). Ein Unterschied des Qualitätswertes von 0.001 macht schon 4% des angenommenen Wertebereichs aus. Daher kann eine geringe Verschiebung des *thresholds* bereits eine deutlich schlechtere Kassifikation der Vektoren zur Folge haben.
Wählt man im Beispiel in Abbildung 3.2 den *threshold* um 0.0004 kleiner (links unten), so wird eine Gruppe von Punkten auf der linken Seite fälschlicherweise als IDM identifiziert, wählt man ihn um 0.0023 größer, so erhält man das Ergebnis rechts unten: man sieht, dass die als IDM klassifizierten Flüsse schon ausdünnen. Und was schwerwiegender ist: die hier nicht mehr erscheinenden

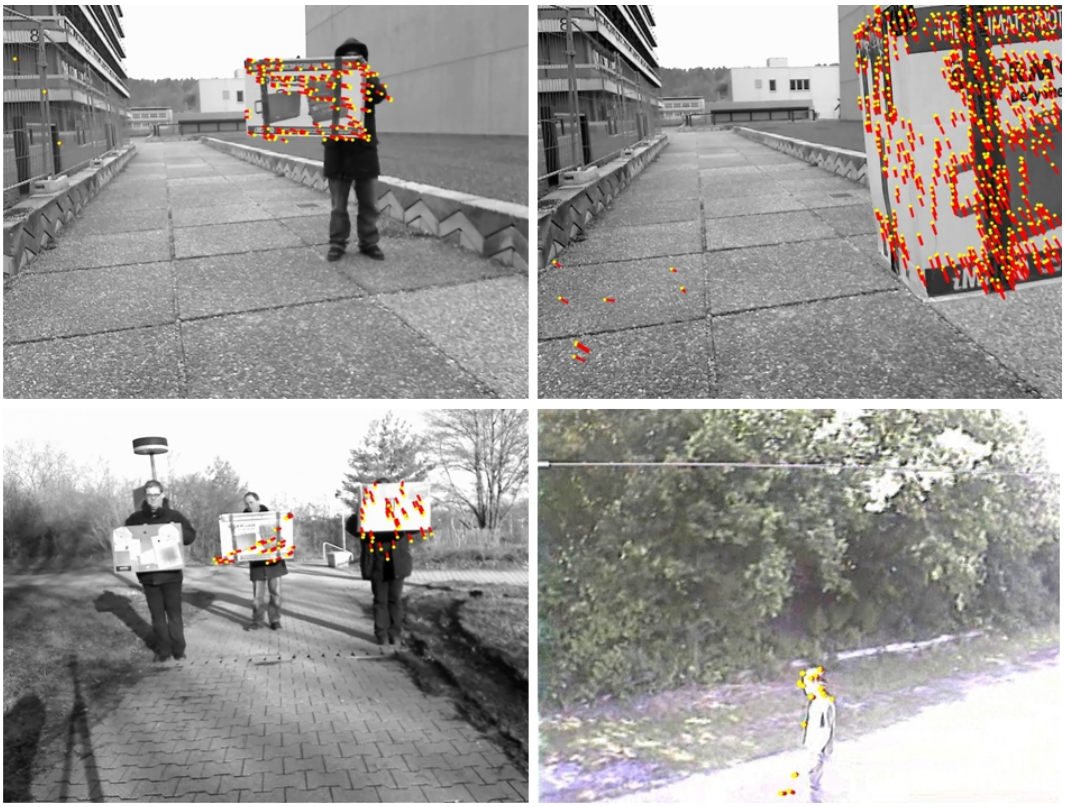


Abbildung 3.1: Die als IDM klassifizierten Flussvektoren.
oben links: *threshold*= 0.037 ; oben rechts: *threshold*= 0.0113 ;
unten links: *threshold*= 0.00555 ; unten rechts: *threshold*= 0.004

Flüsse werden jetzt fälschlich der Egomotion zugeordnet.

Aus diesem Grunde erscheint es angebracht, ein dynamisches *thresholding*-Verfahren zu wählen, um den *threshold* zur Trennung zwischen EM und IDM für jedes Bildpaar dynamisch festzulegen.

3.2 Wahl der *thresholding*-Verfahren

Sezgin und Sankur [Sezgin and Sankur, 2004] testeten 40 *thresholding*-Methoden und bewerteten sie in einem komplexen Wertesystem anhand ihrer Leistung an zwei Klassen von Bildern. Die erste Klasse beinhaltete 40 Textdokumente verschiedener Schriftarten- und Größen und mit verschieden starkem Rauschen überlagert, bei denen die *thresholding*-Methoden angewandt wurden, um die Schrift zu extrahieren. Die zweite Klasse, die Sezgin und Sankur NDT (*nondestructive testing* = Zerstörungsfreie Werkstoffprüfung)-Bilder nennen, beinhaltet 40 Bilder aus verschiedenen Bereichen der Zerstörungsfreien Werkstoffprüfung, u.a. Ultraschallprüfung, Thermografie, Leitfähigkeitsprü-

Abbildung 3.2: Unterschiede in der Klassifizierungsleistung bei winziger Verschiebung des *thresholds*:
Oben Mitte: Klassifikation bei *threshold*= 0.0037.
Unten links: der *threshold* um 0.0004 verkleinert.
Unten rechts: der *threshold* um 0.0023 erhöht.
Mit der grünen Ellipse sind Bereiche markiert, an denen durch die Änderung des *thresholds* Flussvektoren im Vergleich zum Original (oben Mitte) hinzukommen oder wegfallen.

fung, Mikroskopie, etc. Das Ziel des *thresholdings* bei den NDT-Bildern ist es, durch Binarisierung der Bilder eine Darstellung der Bilder zu erhalten, in dem das Qualitätskriterium der Werkstoffprüfung besonders gut zu erkennen ist.
Zu jedem der Testbilder lagen *ground truth*-Daten vor. Zur Bewertung der *thresholding*-Methoden führten Sezgin und Sankur fünf verschiedene Maße ein, die unterschiedliche Aspekte der Klassifizierungsleistungen der verschiedenen Methoden abdecken und bewerten sollten (für eine genauere Erläuterung der Maße siehe [Sezgin and Sankur, 2004]). Alle Maße verglichen Original-Bild und binarisiertes Bild auf Pixel-Ebene.
Die Bewertung der Methoden wurde für jedes Kriterium einzeln durchgeführt, die Methoden wurden dann gemäß ihrer Leistung in einer Rangfolge angeordnet. Der letztliche Gütewert für jede einzelne Methode beinhaltet dann nicht mehr direkt das berechnete Maß, sondern sondern wird vielmehr so erhalten, dass die Ränge jeder Methode in den fünf Leistungskategorien gemittelt wurde, so dass man als Bewertung der verschiedenen Methoden einen durchschnittlichen Rang erhielt.

Nachteilig an dem Artikel von Sezgin und Sankur ist, dass nur die letztendlich ermittelte Bewertung der Methoden angegeben werden, jedoch weder die Leistungen der Methoden in den einzelnen Maßen, noch die Leistungen der Methoden an einzelnen Bildern. Dies wäre hilfreich gewesen, die Leistung der vorgestellten Methoden bezüglich der in der vorliegenden Arbeit gewollten Anwendung, auf die Trennung von EM und IDM hin, abzuschätzen.

Burgoyne et. al. untersuchten 20 der von Sezgin und Sankur als gut bewerteten Algorithmen, sowie fünf weitere, im Hinblick auf die spezielle Anwendung der Binärisierung antiker Musikdokumente (OCM (*optical music recognition*)) [Burgoyne et al., 2007]. Bei alten Musikdokumenten kommt es häufig vor, dass verschiedene Tintenarten auf demselben Dokument verwendet werden, und die Noten auf der Rückseite durchschlagen. Dies erschwert eine automatische Binärisierung. Sie stellten fest, dass diejenigen Algorithmen, die von Sezgin und Sankur als die mit der besten Leistung bewertet wurden, bei der Binärisierung von Musikdokumenten nicht unbedingt gut abschnitten. Beispielsweise landete der laut Sezgin und Sankur überragende Algorithmus von Kittler und Illingworth [Kittler and Illingworth, 1986] bei den Versuchen von Burgoyne nur auf Platz 15. Während der von Sezgin und Sankur nur als mittelmäßig bewertete Algorithmus von Brink und Pendock [Brink and Pendock, 1996] bei Burgoyne überragende Ergebnisse lieferte.

Burgoyne et. al. verwendeten zur Bewertung der getesteten Methoden eine zielorientierte Evaluierungsmethode (*goal-directed evaluation technique*). In komplexen Computervisions-Systemen ist die Binärisierung von Bildern häufig der erste von vielen Prozessierungsschritten. Zielorientierte Evaluierung von Binärisierungsmethoden heisst dann, dass man die Binärisierungsmethoden danach beurteilt, wie gut sie für die Weiterverarbeitung durch die späteren Schritte geeignet sind, das heisst, wie sehr sie diese Ergebnisse zu verbessern vermögen. Der Ansatz einer zielorientierten Bewertung wurde bspw. auch von Belkasim zur Bewertung von Binärisierungsmethoden in der *optical character recognition* [Belkasim et al., 1991], sowie von Trier und Jain zur Evaluierung von Binärisierungsmethoden zur Zahlerkennung [Trier and Jain, 1995]) verwendet.

In der Evaluierung von Trier und Jain schnitt ein lokal adaptives Verfahren, das von Niblack [Niblack, 1985], am besten ab. Lokal adaptive Methoden erscheinen für die Trennung von EM und IDM jedoch ungeeignet. Die Egomotion ist im ganzen Bild dieselbe, der *threshold* sollte deshalb einmal global für das ganze Bild gewählt werden.

Die unterschiedlichen Ergebnisse bezüglich der Güte der verschiedenen Binärisierungsmethoden für verschiedene Anwendungen legen nahe, auch für die Anwendung in dieser Arbeit, der Trennung von EM und IM verschiedene *thresholding*-Methoden zu testen. In dieser Diplomarbeit wurde der Algorithmus

von Kittler und Illingworth [Kittler and Illingworth, 1986] gewählt, der in den Experimenten von Sezgin und Sankur am besten abschnitt, der von Brink und Pendock [Brink and Pendock, 1996], der in den Experimenten von Burgoyne et al [Burgoyne et al., 2007] die besten Ergebnisse lieferte, sowie die Methode nach Yuan [Yuan, 2004], die schon in der Identifikation von IDM-Objekten in 3D-Szenen zur Anwendung kam [Yuan, 2004].

3.3 Überblick über die Implementierung

Der implementierte Algorithmus zur Trennung der IDM- von den EM-Flussvektoren arbeitet in neun Schritten. Abbildung 3.3 zeigt ein Flussdiagramm . Eingabe sind zwei in kurzem zeitlichen Abstand aufgenommene Bilder. Zwischen diesen wird der optische Fluss berechnet. Anhand des Flusses soll dann festgestellt werden, ob sich IDM-Objekte im Blickfeld der Kamera befinden. Die Implementierung der Algorithmen zur Berechnung des optischen Fluss und der Egomotion-Schätzung lag schon vor ([Recktenwald, 2009]). Die neun Prozessierungsschritte werden im Folgenden beschrieben. Schritt 5) (Abschnitt 3.3.5), das dynamische *thresholding*, kann wahlweise mit einer der vorgestellten automatischen *thresholding*-Methoden (Abschnitt 2.3) durchgeführt werden. Die beschriebenen Teile sind zumeist modular implementiert und können je nach Anwendung unterschiedlich kombiniert werden.

3.3.1 Berechnung des optischen Flusses in Vor- und Rückrichtung

Der optische Fluss wird mittels der pyramidalen Implementierung des GDIM-Algorithmus berechnet (siehe Abschnitt 2.1.9 und 2.1.10), die Implementierung stammt von Recktenwald [Recktenwald, 2009]). Ist ein Fluss mit GDIM nicht berechenbar, weil die benötigte inverse Matrix nicht berechnet werden kann, so wird probiert, den Fluss über den pyramidalen Lucas-Kanade-Algorithmus zu bestimmen. Als Parameter sind die Anzahl der pyramidalen Ebenen, sowie die Körnigkeit eines Gitternetzes festzulegen. Dieses Gitternetz mit der angegebenen Anzahl von Feldern wird in der Größe den Eingabebildern angepasst und darüber gelegt. Pro Feld wird ein Flussvektor berechnet. Man erhält so also ein *sparse flow field*, das relativ gleichmäßig über das Bild verteilt ist, jedoch kann in manchen Feldern kein Fluss berechnet werden, falls dort zu wenige Gradienteninformationen vorhanden sind. Die Anzahl der pyramidalen Ebenen wurde in allen hier durchgeführten Test auf fünf gesetzt. Das Gitternetz wurde auf je 60 Felder in x- und y-Richtung festgelegt. Abbildung 3.4a zeigt ein Beispiel für ein Ergebnis einer Flussberechnung. Die Flussvektoren sind gleichmäßig über das ganze Bild verteilt, nur in Bereichen, in denen keine Helligkeitsunterschiede vorhandenen sind, wie am Himmel und der gleichmäßigen

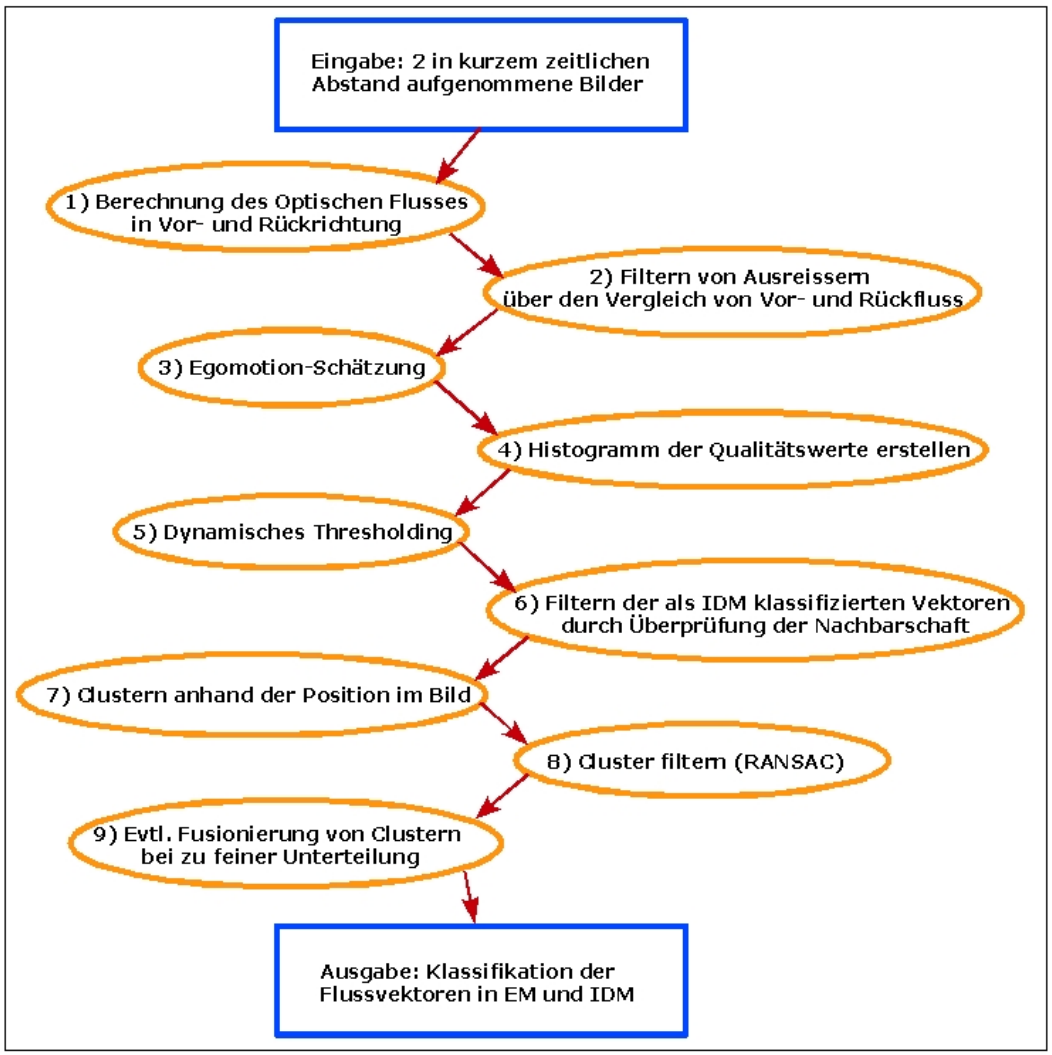


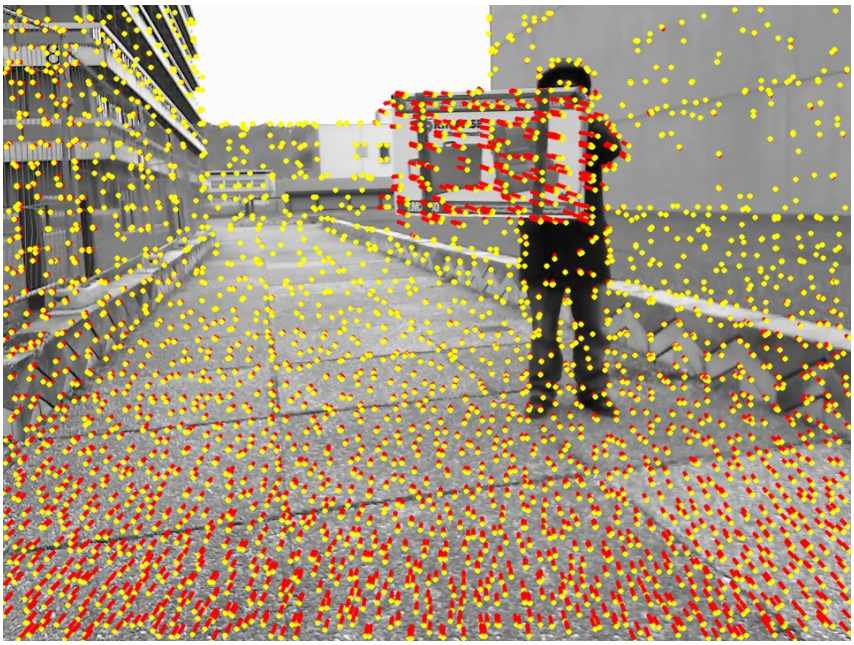
Abbildung 3.3: Das Flussdiagramm des implementierten Algorithmus

Wand des Gebäudes rechts, konnte kein Fluss bestimmt werden.

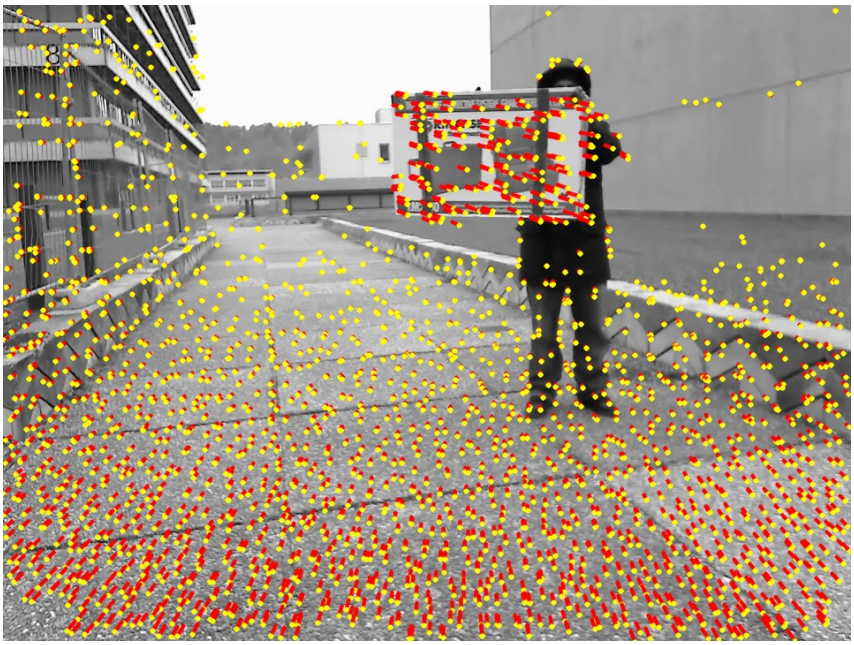
3.3.2 Filtern von Ausreißern über den Vergleich von Vor- und Rückfluss

Der Vergleich von Hin- und Rückfluss ist eine gängige Methode, um Ausreißer unter den Flussvektoren herauszufiltern. Es werden die Längen von Hin- und Rückfluss verglichen, sowie deren Winkel. Beträgt die Pixel-Differenz von Hin- und Rückfluss mehr als 7 Pixel, so wird dieser Flussvektor verworfen. Beträgt der Winkel zwischen Hinfluss und dem inversen Rückfluss mehr als $2°$[1], so

[1] Verschiedene Testläufe wiesen auf diese Parameter als gute Wahl hin



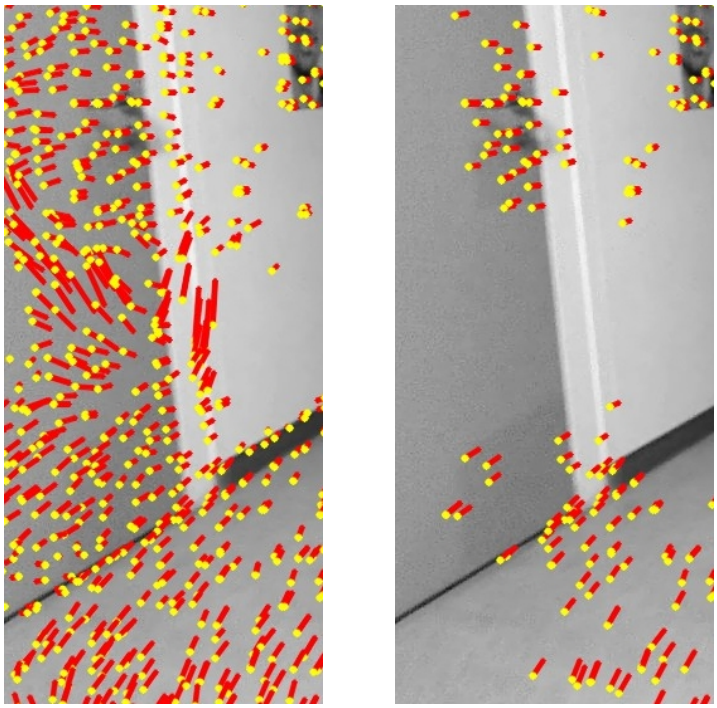
(a) Vor dem Filtern



(b) Nach dem Filtern

Abbildung 3.4: (a): alle berechneten Flussvektoren (b): die Flussvektoren nach dem Filtern über Hin- und Rückfluss

wird der betreffende Flussvektor ebenfalls verworfen. Auf diese Art und Weise können schon viele Ausreißer entfernt werden. Ein Beispiel ist in Abbildung 3.5 und 3.4 zu sehen.



(a) Vor dem Filtern (b) Nach dem Filtern

Abbildung 3.5: Links die Flussvektoren vor dem Filtern über Hin- und Rückfluss, rechts die Flussvektoren nach dem Filtern: diejenigen Flussvektoren, deren Vor- und Rückfluss nicht innerhalb der gesetzten Parameter (siehe 3.3.2) übereinstimmten, wurden herausgefiltert und verworfen.

3.3.3 Egomotion-Schätzung

Die Egomotion-Schätzung wird mit dem in Abschnitt 2.2 vorgestellten Kanatani-Algorithmus durchgeführt. Für jeden Flussvektor wird anhand der ermittelten Egomotion ein Qualitätswert berechnet, welcher der Differenz zwischen aus geschätztem und tatsächlich beobachteten Fluss entspricht (2.2). Grundsätzlich wird angenommen, dass die Anzahl der EM-Vektoren unter den Flussvektoren überwiegt. Dies ist die Voraussetzung, damit die Egomotion zuverlässig bestimmt werden kann. Nur wenn die Schätzung der EM zuverlässig ist, ist es möglich, anhand des Reprojektionsfehler EM und IDM zu trennen. Die Annahme, dass EM-bedingt Flussvektoren überwiegen ist durchaus plausibel, da bei der Navigation der Flugdrohne Objekten immer frühzeitig ausgewichen wird, so dass der Abstand immer groß genug ist, damit ein Objekt nicht das gesamte Blickfeld ausfüllt.

3.3.4 Histogramm der Qualitätswerte erstellen

Aus den Qualitätswerten der Flussvektoren wird dynamisch ein Histogramm aus 100 äquidistanten *bins* erstellt, das für das dynamische *thresholding* im nächsten Schritt benötigt wird. Als Ober- und Untergrenze des Wertebereichs des Histogramms werden der maximale und minimale Qualitätswert unter den Qualitätswerten der Flussvektoren des jeweiligen Bildpaares gewählt. Der Bereich dazwischen wird in 100 gleich große Bereiche, die *bins* unterteilt. Jedes *bin* enthält die Anzahl derjenigen Flussvektoren, deren Qualitätswert in den Bereich des *bins* fällt. Ein typisches Histogramm ist in Abbildung 3.6c dargestellt.

3.3.5 Dynamisches *tresholding*

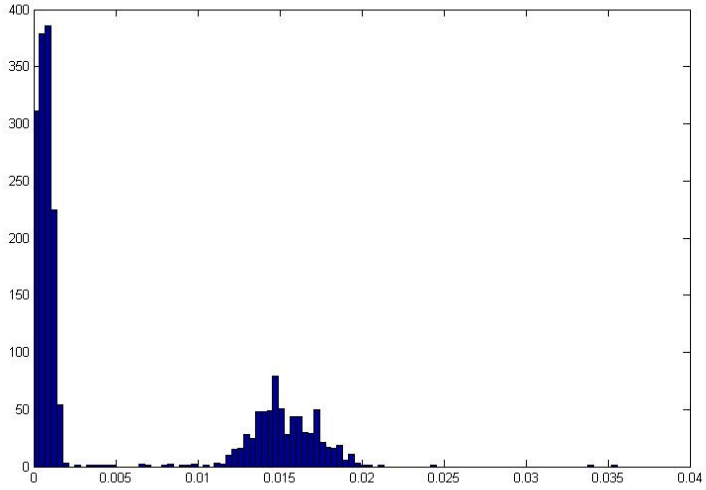
Das dynamische *thresholding* kann wahlweise mit einer der drei vorgestellten Methoden (Abschnitt 2.3) durchgeführt werden. Im Abschnitt 2.3 wurden die Methoden in ihrer ursprünglichen Anwendung, der Grauwert-Segmentierung bzw. der Grauwert-Differenzen-Segmentierung vorgestellt. Hier werden sie nun angewandt, um anhand eines Histogramms der Reprojektionsfehler der Flussvektoren die EM von der IDM zu trennen. Grundsätzlich kann man IDM und EM anhand der Flussvektoren nur dann trennen, wenn sich die durch die IDM entstehenden Flussvektoren deutlich von den durch EM entstandenen abheben. Das heisst, im Falle dass EM und IDM in Bezug auf den f.o.e. und die Länge der Flussvektoren ähnliche Vektoren erzeugen, sind sie anhand des optischen Flusses nicht zu trennen. Erzeugt die IDM aber von der EM verschiedene Flussvektoren, so sollte sich dies in den Reprojektionsfehlern widerspiegeln. EM-Vektoren sollten, da sie mit den geschätzten Egomotion-Parametern verträglich sind, Reprojektionsfehler nahe 0 aufweisen, während IDM-Vektoren einen größeren Reprojektionsfehler ergeben. Man erwartet also im Histogramm einen durch die EM-Vektoren erzeugten Peak nahe null. Bei der IDM geht man davon aus, dass Punkte auf demselben Objekt eine ähnliche 3D-Bewegung ausführen und somit auch ähnliche Flussvektoren erzeugen. Diese Vektoren eines IDM-Objekts werden dann alle einen ähnlichen Reprojektionsfehler aufweisen und sollten so ebenfalls einen Peak im Histogramm ergeben. Ist das Objekt recht klein, oder weist es wenig Helligkeitsunterschiede oder Textur auf der Oberfläche auf, so kann der erzeugte Peak sehr klein sein. Von den Peaks kann man annehmen, dass sie in die Form einer Gaussverteilung besitzen, da die Flussvektoren fehlerbehaftet sind [2] und Fehler häufig einer Normalverteilung genügen. Die dynamischen *thresholding*-Verfahren liefern also einen Fehlerwert, den man als Grenze zwischen EM und IDM wertet. EM-Flüsse haben einen Qualitätswert kleiner als der *threshold*, IDM-Flüsse größer als der *threshold*.

[2] Die Bilder, auf denen der optische Fluss berechnet wird, sind verrauscht. Aus diesem Grunde sind auch die berechneten Flussvektoren fehlerbehaftet.

Abbildung 3.6 zeigt ein Beispiel für ein Histogramm, an dem die Peaks für EM und IDM deutlich zu unterscheiden sind.

(a)

(b)



(c)

Abbildung 3.6: (a) und (b): Ein Bildpaar mit EM und IDM. Die Kamera bewegt sich in einer reinen Translationsbewegung geradeaus in Blickrichtung. Die Kiste rechts im Bild bewegt sich unabhängig von der Kamera nach rechts vorne. (c): Das Histogramm der Reprojektionsfehler, wenn aus dem Optischen Fluss zwischen 3.6a und 3.6b die Egomotion bestimmt wird, und für alle Flussvektoren der Reprojektionsfehler berechnet wird. Der hohe Peak links kennzeichnet die EM, der etwas flachere in der Mitte die IDM.

3.3.6 Filtern der als IDM klassifizierten Vektoren durch Überprüfung der Nachbarschaft

Es wird für jeden Flussvektor die euklidische Distanz vom Startpunkt des Flusses bis zum nächstliegenden Startpunkt eines als IDM klassifizierten Flussvek-

tors im Bild ermittelt. Liegt ein IDM-Vektor allein in einem gewissen Umkreis, so kann man davon ausgehen, dass es sich um einen Ausreißer handelt. Vektoren, deren nächster Nachbar weiter als 50 Pixel entfernt liegt, werden als Ausreißer klassifiziert und aus der Menge der IDM-Vektoren entfernt.

3.3.7 Clustern anhand der Position im Bild

Die verbleibenden IDM-Vektoren werden anhand ihrer Position im Bild geclustert. Dies geschieht mit dem in Abschnitt 2.4.2 beschriebenen *G-means*-Algorithmus.
Da der *G-means*-Algorithmus die ermittelten Cluster auf eine Normalverteilung der Datenpunkte um das ermittelte Clusterzentrum hin überprüft und so lange weiter unterteilt, bis alle Cluster einer Normalverteilung um ihr Zentrum genügen, kommt es relativ häufig vor, dass die IDM-Objekte in eine größere Anzahl von Clustern aufgeteilt werden, als es nötig wäre.

Es ist theoretisch möglich, die IDM-Vektoren nicht nur nach ihrer Position im Bild zu clustern, sondern den Reprojektionsfehler als dritte Dimension hinzuzunehmen. Der *G-means*-Algorithmus wurde so implementiert, dass die Dimension der zu clusternden Datenpunkte beliebig ist. Es ist vorstellbar, dass man so nahe beieinanderliegende oder sich gegenseitig verdeckende Objekte in unterschiedliche Cluster zu klassifizieren könnte. Vorläufige Versuche diesbezüglich waren jedoch wenig vielversprechend. Da unter den IDM-Vektoren noch zahlreiche Ausreißer vorhanden sind, ist es eher hinderlich, die Qualität als dritte Dimension beim Clustern hinzuzunehmen, als dass es das Ergebnis verbessern würde.

3.3.8 Filtern der Cluster

Falls man zu feine Cluster wieder fusionieren möchte, so muss man die durchschnittlichen Vektoren zweier möglicherweise zu fusionierender Cluster vergleichen, um festzustellen, ob die Cluster zum selben IDM-Objekt gehören. (Dies wird im nächsten Abschnitt genauer beschrieben.) Die überwiegende Anzahl der Ausreißer unter den Flussvektoren, die nicht über das Hin-und Rückfluss-Filtern (Abschnitt 3.3.2) entfernt wurden, wurden beim dynamischen *thresholding* den IDM-Vektoren zugeordnet. Dies rührt daher, dass die Ausreißer-Flüsse, wie die IDM-Vektoren, nicht zu der geschätzten Egomotion passen und somit meist auch schlechtere Qualitätswerte besitzen. Möchte man einen durchschnittlichen Clustervektor in diesen von Ausreißern durchsetzten Clustern berechnen, so wird dieser durch die Ausreißer verfälscht, und vergleicht man diese durchschnittlichen Vektoren benachbarter Cluster, die zum selben IDM-Objekt gehören, so unterscheiden sich diese oft so stark, dass man

sie nicht zusammenfassen würde. Aus diesem Grund ist es günstig, die Cluster vorher von Ausreißern zu befreien. Ein weiterer Grund, aus dem man Ausreißer unter den IDM-Vektoren entfernen möchte, wäre, wenn man anhand der IDM-Vektoren die Translationsrichtung des IDM-Objekts schätzen wollte. Das Filtern der einzelnen Cluster wird mit dem RANSAC-Algorithmus durchgeführt(Abschnitt2.6). Es wird die Länge und der Winkel mit der x-Achse eines zufällig gewählten Vektors aus dem Cluster bestimmt, und mit den Längen und Winkel mit der x-Achse der andern Clustervektoren verglichen. Liegt der Längenunterschied unter 3 Pixel und der Winkelunterschied unter 8°, so wird der Vektor dem *consensus set* hinzugefügt [3] . Die Iterationsanzahl i beträgt:

$$i = \begin{cases} \text{Clustergöße}/2 \quad falls \quad \text{Clustergöße}/2 \quad \leq \quad 25 \\ 25 \quad falls \quad \text{Clustergöße}/2 \quad > \quad 25 \end{cases} \tag{3.1}$$

Aus dem größten *consensus set* nach i Iterationen wird der durchschnittliche Clustervektor berechnet. Abbildung 3.7a zeigt ein Beispiel. Jedes Cluster ist mit einem weißen Rechteck umrandet [4]. Aus dem Cluster ganz links wurden fünf Ausreißer (eingezeichnet in grün) heraus gefiltert.

3.3.9 Eventuelle Fusionierung von Clustern bei zu feiner Unterteilung

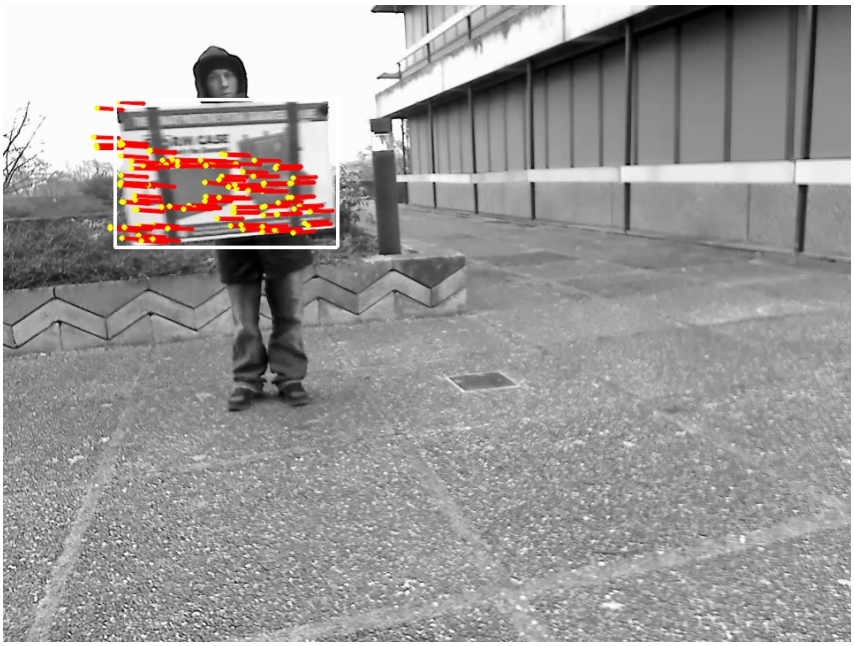
Falls die IDM-Objekte in zu feine Cluster untergliedert wurden, ist es mit diesem Modul möglich, nah beieinanderliegende Cluster deren Vektoren ähnlich sind, zu fusionieren. Die Cluster werden paarweise bezüglich ihres Abstands geprüft. Der festgelegte Höchstabstand für eine Fusionierung wurde auf einen Abstand von 15 Pixeln festgesetzt. Ist das der Fall, werden die durchschnittlichen Clustervektoren (siehe 3.3.8) verglichen. Wenn sich die Länge ihrer Durchschnittsvektoren nicht um mehr als 3 Pixel, die Winkel mit der x-Achse nicht um mehr als 5° unterscheiden[5], so werden die beiden Cluster fusioniert. Die Cluster werden so lange paarweise geprüft, bis sich keine Änderungen mehr ergeben. Abbildung 3.7 zeigt ein Beispiel. Im oberen Bild sind drei Cluster zu erkennen. Im unteren Bild wurden die drei Cluster zu einem fusioniert.

[3] Verschiedene Testläufe wiesen auf Längenunterschied unter 3 Pixel und der Winkelunterschied unter 8° als gute Kriterien hin

[4] Die die Cluster umschreibenden Rechtecke überlappen zwar, die Cluster aber nicht. Die Rechtecke dienen der Visualisierung. Die umschreibenden Rechtecke wurden etwas größer eingezeichnet, als das Cluster eigentlich ist, damit alle Punkte umschlossen und nicht verdeckt werden.

[5] Diese Parameter lieferten in verschiedenen Testsituationen die besten Ergebnisse.

(a) Vor dem Verschmelzen der Cluster



(b) Nach dem Verschmelzen der Cluster

Abbildung 3.7: Oben die gefilterten, noch nicht fusionierten Cluster. Die grün eingezeichneten Vektoren wurden als Ausreißer im Cluster ganz links identifiziert und entfernt. Unten der resultierende Cluster nach der Fusion.

3.4 Aufnahme der Testsequenzen

Zur Aufnahme der Testsequenzen wurde eine Logitech Webcam QuickCam 9000 Pro mit 2 Megapixeln verwendet. Diese besitzt einen horizontalen Bild-

winkel von 41° und einen vertikalen Bildwinkel von 33°. Die zeitliche Auflösung für Videoaufnahmen beträgt 30 Bilder/sec. Um eine rein translatorische Bewegung zu ermöglichen und Vibrationen bei der Aufnahme weitestgehend zu vermeiden, wurde die Kamera fest auf einem eigens dafür gebauten Holzgerüst montiert, das mit Schienen versehen war (siehe Abbildung 3.8). Da durch Rotation entstehende Flussvektoren keinen Aufschluss über die räumliche Struktur der aufgenommenen Szene liefern (siehe 2.1.5), war eine rein translatorische Bewegung wünschenswert. Mit dieser Anordnung war es sogar bei Außenaufnahmen auf unebenem Grund möglich, scharfe und unverwackelte Aufnahmen zu erhalten. Als IDM-Objekte wurden mit verschiedenfarbiger Pappe beklebte Kartons verwendet. Es wurde darauf geachtet, dass zahlreiche scharf getrennte Helligkeitsbereiche vorhanden waren, um zuverlässige Flussberechnungen zu ermöglichen. Die IDM-Kartons wurden von freundlichen Menschen gleichmäßig im Blickfeld der Kamera bewegt. Die Testsequenzen wurden zur Verwendung im implementierten Klassifikator in Einzelbilder umgewandelt.

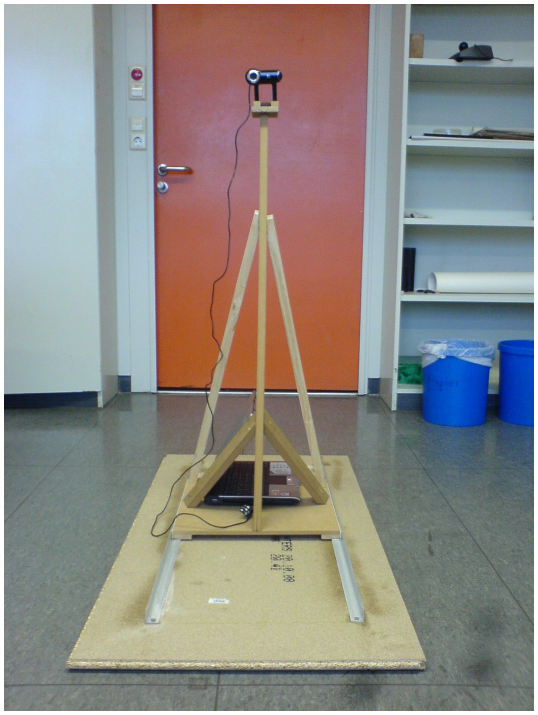


Abbildung 3.8: Die zur Aufnahme der Testsequenzen verwendete Konstruktion. Die Kamera ist fest montiert. Auf dem auf Schienen laufenden Holzgestell ist Platz für den zur Aufnahme benötigten Laptop.

3.5 Ermittlung der *ground truth*

Zu den Testsequenzen existieren keine *ground truth*-Daten. Daher habe ich den tatsächlichen Sachverhalt aus den von den verschiedenen Algorithmen gelieferten Daten rekonstruiert und so eine Approximation der *ground truth* erhalten. Wie die die Ergebnisse in Abschnitt 4.3.1 zeigen, sind die approximierten *ground truth*-Daten gut genug, um konsistente Ergebnisse zu liefern.

Benötigt wurden:

- Die Anzahl derjenigen EM-Vektoren, die fälschlicherweise als IDM klassifiziert wurde (EM→IDM).
- Die Anzahl derjenigen IDM-Vektoren, die fälschlicherweise als EM klassifiziert wurde (IDM→EM).
- Die Anzahl derjenigen IDM-Vektoren, die richtigerweise als IDM klassifiziert wurde (IDM→IDM).
- Die Anzahl derjenigen EM-Vektoren, die richtigerweise als EM klassifiziert wurde (EM→EM).

Ermittlung von EM→IDM

Die Anzahl der EM-Vektoren, die fälschlich der IDM zugeordnet wurden, kann man unter als IDM klassifizierten Vektoren abzählen.
In Abbildung 3.9 sind die als IDM klassifizierten Vektoren in gelb-rot eingezeichnet und mit einem weißen Kasten umrandet.
Man kann folgendes erkennen: das Cluster in der Mitte besteht aus echten IDM-Vektoren, aber die beiden anderen, oben in der Mitte und ganz links außen, gehören nicht zur IDM. Das heisst die Anzahl der EM→IDM beträgt in diesem Fall drei.

Ermittlung von IDM→EM

Die Anzahl der IDM-Vektoren, die fälschlicherweise der EM zugerechnet wurden, kann man unter den als EM klassifizierten Vektoren abzählen. Abbildung 3.10 zeigt alle der EM zugeschlagenen Vektoren. Die „falschen“ IDM-Vektoren sind die in der oberen Hälfte der Kiste.

Ermittlung von IDM→IDM

Von der Anzahl der vom jeweiligen Verfahren als IDM klassifizierten Vektoren muss man die Anzahl der fälschlich als IDM klassifizierten Vektoren (EM→IDM) abziehen.

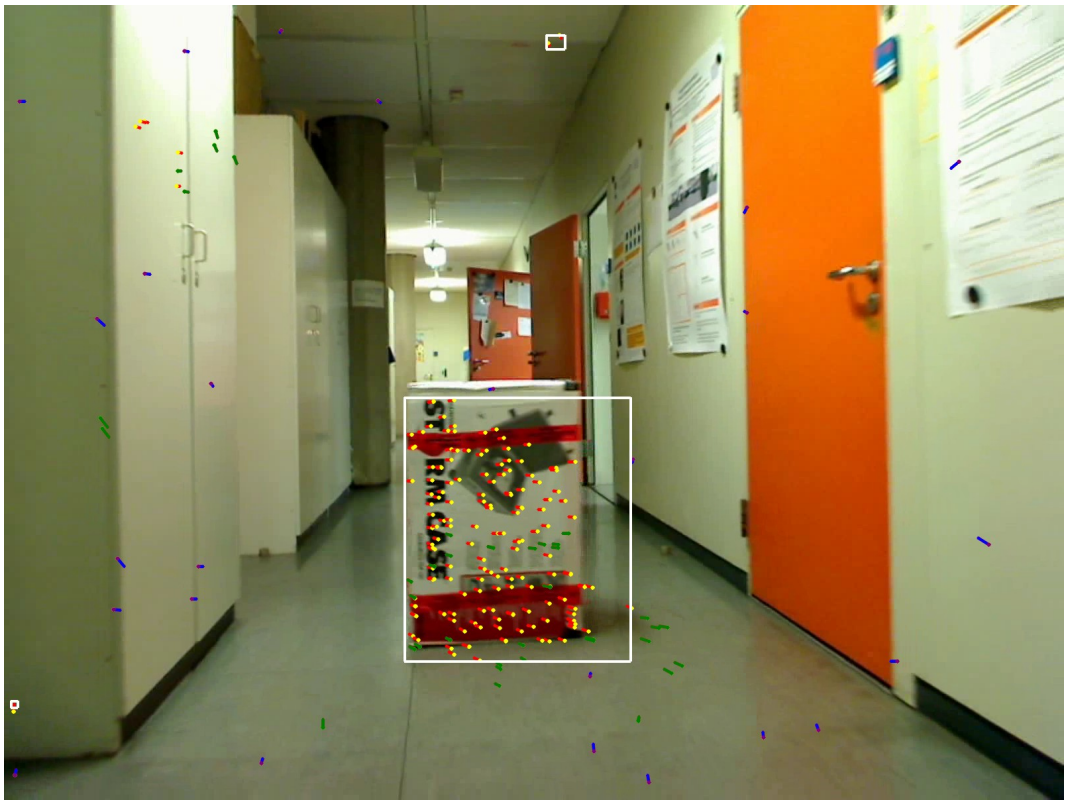


Abbildung 3.9: Alle initial als IDM klassifizierten Flussvektoren. Die letztlich als IDM klassifizierten Vektoren sind in rot-gelb eingezeichnet und mit einem weißen Kasten umrandet. Als Ausreißer identifizierte Vektoren sind in blau und grün eingetragen.

Ermittlung von EM→EM

Von der Anzahl der vom jeweiligen Verfahren als EM klassifizierten Vektoren muss man die Anzahl der fälschlich als IDM klassifizierten Vektoren (IDM→EM) abziehen.

Ermittlung der Anzahl „echter“ Ausreißer

Unter dem vom Verfahren festgestellten Ausreißern können auch Vektoren sein, die fälschlich gefiltert wurden. Diese müssen gezählt und abgezogen werden. Ein Beispiel in Abbildung 3.9. Die als Ausreißer klassifizierten Vektoren sind in blau und grün eingezeichnet. Jedoch wurden im Bereich des IDM-Objekts (der Kiste) Vektoren entfernt, die zur IDM gehören. Diese müssen gezählt und abgezogen werden.

Abbildung 3.10: Die als EM klassifizierten Vektoren.

Kapitel 4

Ergebnisse und Diskussion

4.1 Wahl des optimalen *thresholds*

Zur Bewertung der drei dynamischen *thresholding*-Verfahren musste der ideale *threshold* zur Trennung von EM und IDM anhand den Reprojektionsfehlern ermittelt werden. Die im Algorithmus automatisch erstellten Qualitätshistogramme besitzen, wie in Abschnitt 3.3.4 beschrieben, jeweils 100 *bins*. Um den idealen *threshold* für ein Bildpaar zu bestimmen, wurde für jeden möglichen *threshold* die Trennung in EM und IDM betrachtet, um denjenigen *threshold* als ideal zu identifizieren, der nach einer visuellen Inspektion die beste Trennung der beiden Klassen liefert. Ein typisches Beispiel ist in den Abbildungen 4.1a bis 4.1e gegeben. Die als IDM klassifizierten Vektoren sind immer auf der linken, die als EM klassifizierten Vektoren auf der rechten Seite dargestellt. Abbildung 4.1a zeigt die Trennung zwischen IDM und EM wenn der *threshold* an der Untergrenze von *bin* 1 liegt, Abbildung 4.1b zeigt die Trennung zwischen IDM und EM wenn der *threshold* an der Untergrenze von *bin* 16 liegt, usw. Die ganze Serie der Bilder umfasst also 100 Bilder, eines für jeden möglichen *threshold* entsprechend den 100 *bins* des jeweiligen Histogramms. Der *threshold* liegt dabei jeweils an der Untergrenze des jeweilgen *bins*. Die dargestellten mittleren drei Bilder 4.1b bis 4.1d sind die zur Festlegung des idealen *thresholds* relevantesten, der ideale *threshold* wurde hier an der Untergrenze von *bin* 17 festgelegt. Bild *thr17* ist der visuell bestimmte bestmögliche Kompromiss zwischen den Zielen, möglichst viele IDM-Vektoren als IDM zu klassifizieren und andererseits möglichst wenige EM-Vektoren fälschlicherweise als IDM-Vektoren zu deklarieren. Vergleicht man die beiden Klassifikationen in Bild 4.1b und Bild 4.1c so stellt man fest, dass links in Bild 4.1b zwar noch ein Vektor weniger der IDM zugeordnet wird als in Bild 4.1c, jedoch werden beim vergleich der beiden Bilder rechts gleich vier IDM-Vektoren zusätzlich fälschlicherweise der EM zugerechnet, als wenn man den *threshold* an der Untergrenze von *bin* 16 wählt. Vergleicht man die Bilder 4.1c und 4.1d, so kann man erken-

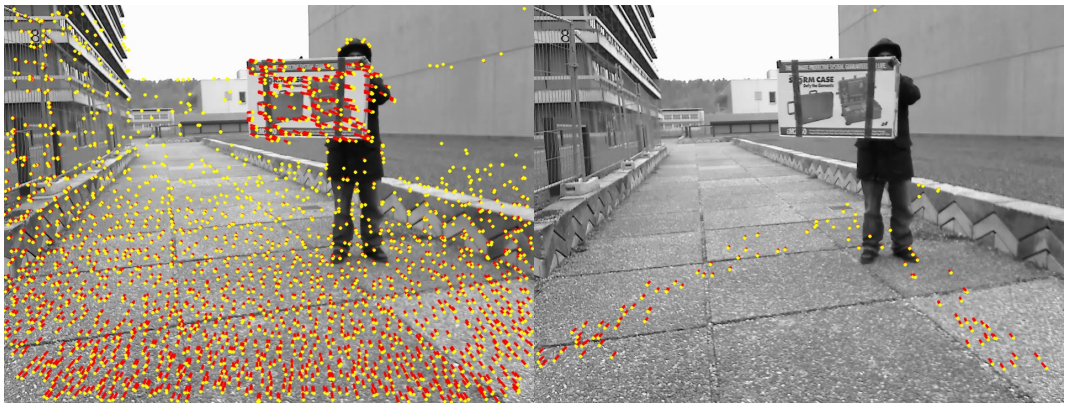


Abbildung 4.1a: Der *threshold* liegt an der Untergrenze von *bin*1

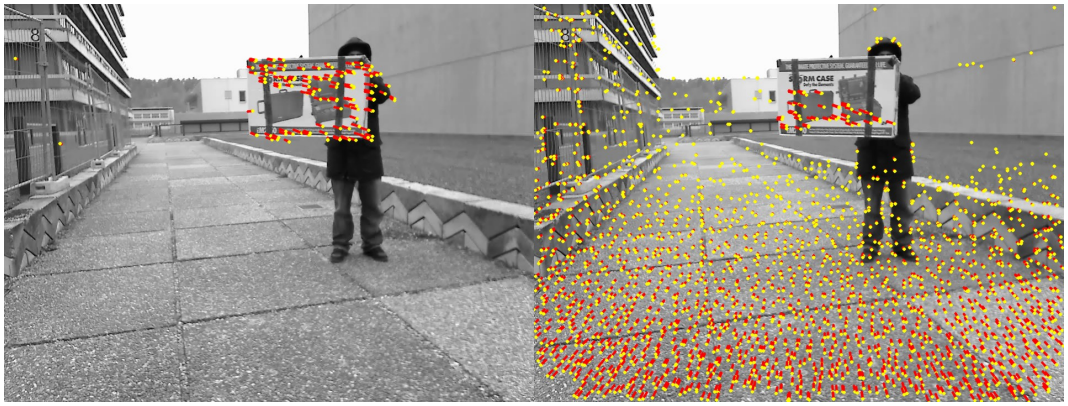


Abbildung 4.1b: Der *threshold* liegt an der Untergrenze von *bin*16

nen, dass in Bild 4.1d zwar drei IDM-Vektoren mehr als in Bild 4.1c richtig klassifiziert werden (betrachte die linken Hälften der Abbildungen), dafür werden aber auch vier EM-Vektoren mehr falsch klassifiziert (rechte Bildhälften). Diese Abwägungen lagen der Festlegung des idealen *thresholds* an der Untergrenze von *bin* 17 zugrunde.

Bei einige Bildpaaren war die Lage des optimalen *thresholds* weniger eindeutig als im beschriebenen Beispiel. In diesem Fall wurde ein Intervall definiert, in dem der ideale *threshold* liegen muss. Ein Beispiel hierzu befindet sich im Anhang A.1.

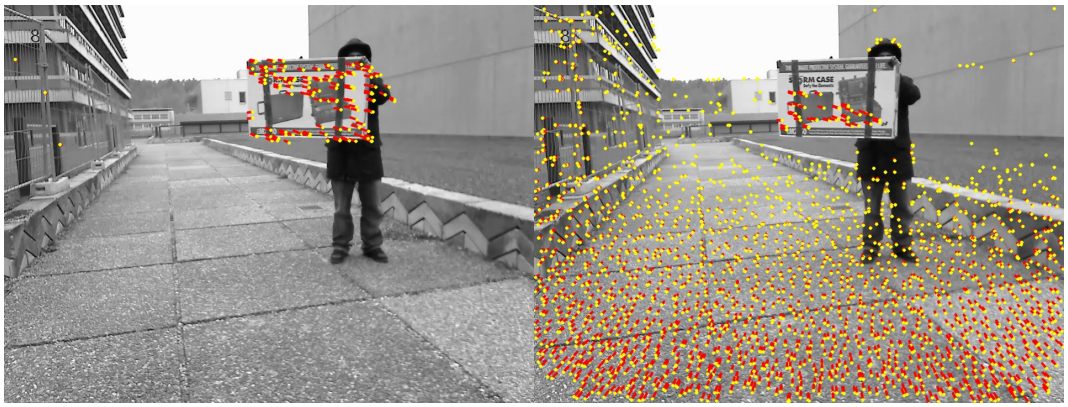


Abbildung 4.1c: Der *threshold* liegt an der Untergrenze von $bin17$

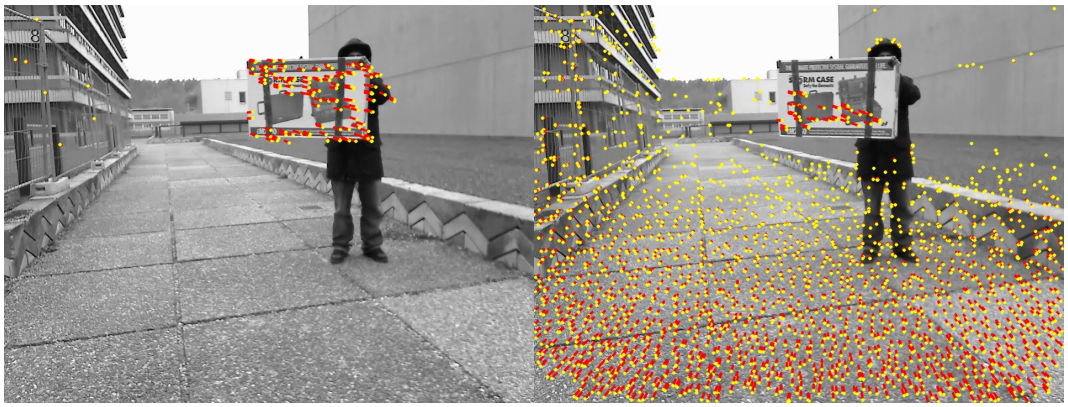


Abbildung 4.1d: Der *threshold* liegt an der Untergrenze von $bin18$

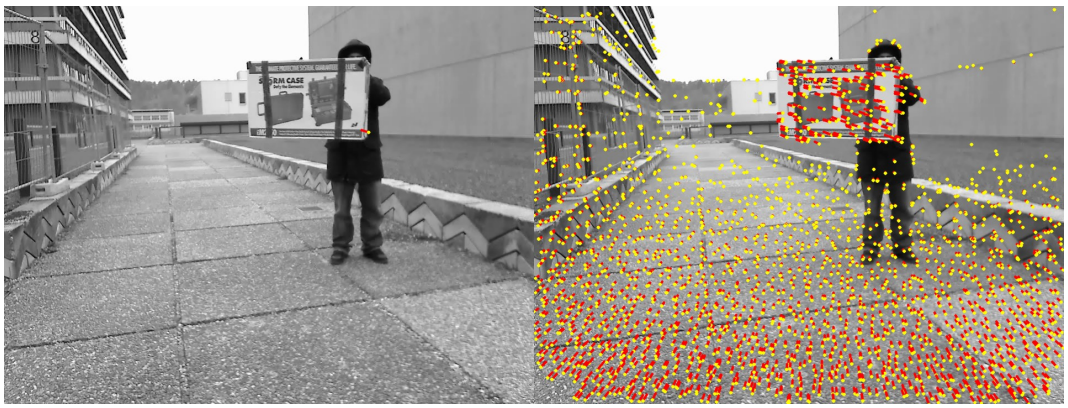


Abbildung 4.1e: Der *threshold* liegt an der Untergrenze von $bin100$

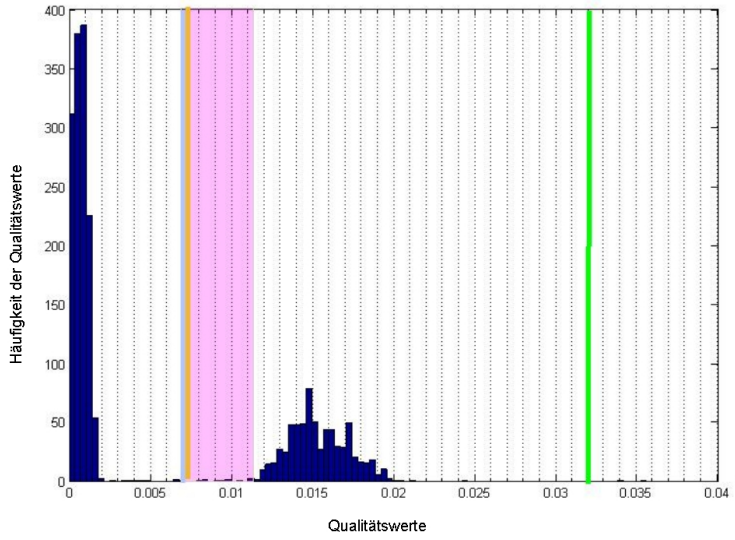
4.2 Bewertung der Lage der dynamisch gelegten *thresholds* im Vergleich zum idealen *threshold*

Grundlage für die Bewertung der Lage der berechneten *thresholds* im Vergleich zum idealen *threshold* bilden 19 Bildpaare, 10 Innen- und 9 Außenaufnahmen. Die Bewertung wird getrennt für Innen- und Außenaufnahmen durchgeführt. Bei Außenaufnahmen ist die Flussdetektion häufig schwieriger. Blätter und Äste bewegen sich im Wind, dadurch verändert sich der Schattenwurf. Die Beleuchtungseigenschaften sind nicht stabil, was die Flussdetektion erschwert und zu einem häufigeren Auftreten von Ausreißern führen kann. Daher bestand die Möglichkeit, dass sich die Leistungen der Verfahren bei Innen- und bei Außenaufnahmen stark unterscheiden.

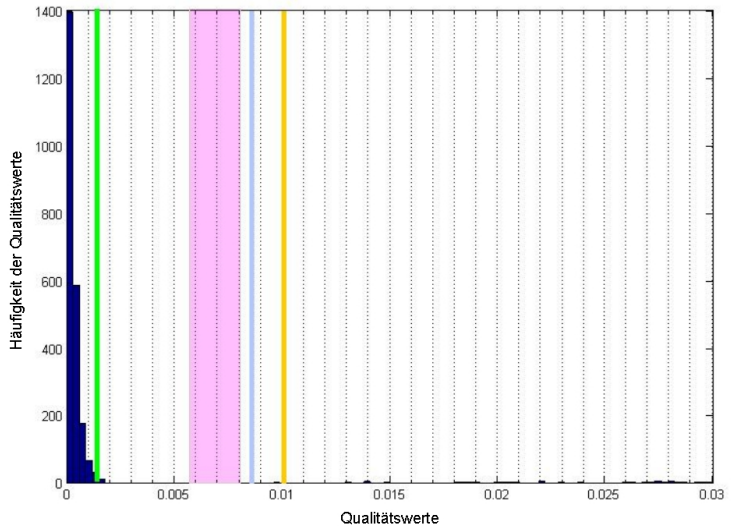
Für jedes der 19 Testbildpaare wurde manuell der optimale *threshold* bestimmt (siehe Abschnitt 4.1). Dieser wurde mit den von den drei Verfahren ermittelten *thresholds* verglichen.

Die Abbildungen 4.2a, 4.2b und 4.6b zeigen einige typische Beispiele für die Histogramme der Testbildpaare. In magenta ist der optimale *threshold* eingezeichnet. In den ersten beiden Abbildungen ist ein Intervall für den optimalen *threshold* definiert. In der dritten Abbildung gibt es nur einen möglichen Wert für den idealen *threshold*. In grün ist der Kittler-*threshold* eingezeichnet, in hellblau der Brink-*threshold* und in orange der Yuan-*threshold*. Betrachtet man ein Histogramm, so bekommt man oft eine ungefähre Vorstellung davon, wo der optimale *threshold* liegen müsste. In Abbildung 4.2a sind beispielsweise die beiden Peaks für die EM und die IDM klar voneinander zu unterscheiden. Für eine genauere Einschränkung der Lage des optimalen *thresholds* muss man jedoch die Klassifizierung der Vektoren an den Bildern zur Rate ziehen (siehe 4.1).

In Abbildung 4.3 sind die Abweichungen der automatisch bestimmten *thresholds* vom idealen *threshold* graphisch dargestellt. Auf der x-Achse sind die 10 beziehungsweise 9 Testfälle aufgetragen, auf der y-Achse der Qualitätswert. In magenta ist wieder der optimale *threshold*, in grün der Kittler-, in hellblau der Brink- und in orange der Yuan-*threshold* eingezeichnet. In Fällen in denen der optimale *threshold* einen fest definierten Wert annehmen muss ist er als Punkt eingezeichnet, in Fällen in denen ein Intervall für den idealen *threshold* definiert ist, kennzeichnet ein Strich den Bereich. Dieses Diagramm soll einen Eindruck von der Unterschiedlichkeit der Lage der idealen *thresholds* vermitteln, sowie der Lage der automatisch bestimmten relativ zum idealen. Man kann erkennen, dass das Kittler-Verfahren dazu tendiert, den *threshold* etwas tiefer als den idealen Wert zu legen. Das Yuan-Verfahren dagegen legt den *threshold* zumeist höher als den idealen *threshold*. Beim Brink-Verfahren ist



(a)

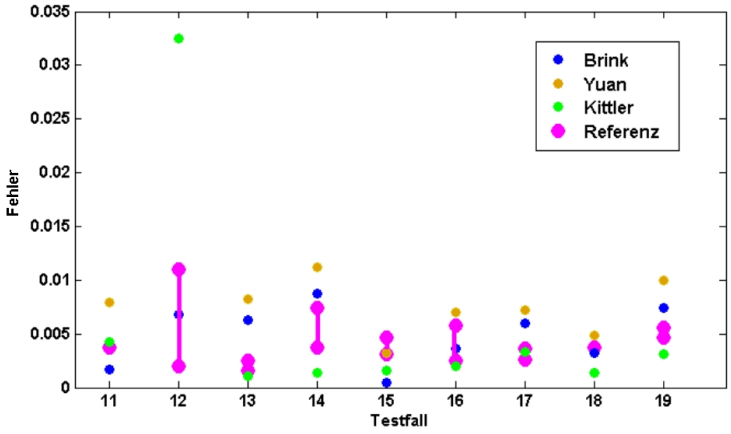


(b)

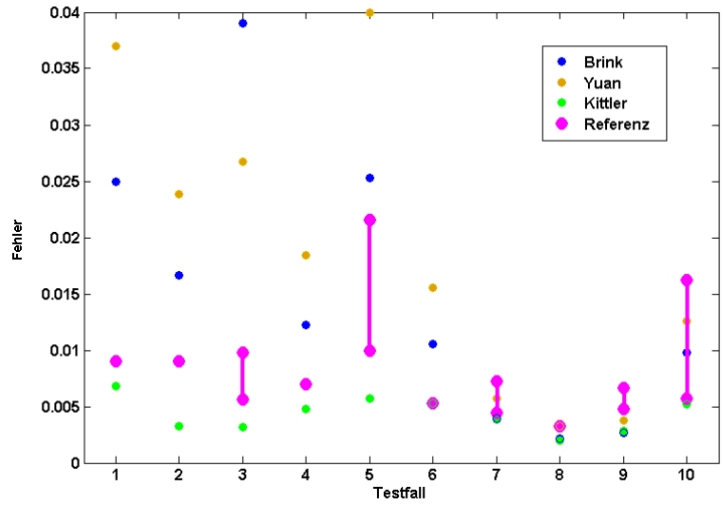
Abbildung 4.2: Zwei typische Histogramme. blau = Brink-Pendock-*threshold*, magenta = idealer *threshold*, grün = Kittler-*threshold*, gelb = Yuan-*threshold*

keine solche Tendenz zu erkennen.

Die Abbildung 4.4 bietet eine andere Darstellung der Abweichungen der automatisch bestimmten *thresholds* vom idealen *threshold*. Auf der x-Achse sind wieder die 10 beziehungsweise 9 Testfälle aufgetragen, auf der y-Achse die Abweichung vom Idealen *threshold*. Der ideale *threshold* ist als Nullinie dargestellt. Die fehlerhafte Abweichung der automatisch bestimmten *thresholds* sind als Balken in y-Richtung aufgetragen. Man gewinnt hier einen Eindruck der absoluten Abweichungen der automatischen *thresholds* vom idealen. Bei



(a) Außenaufnahmen

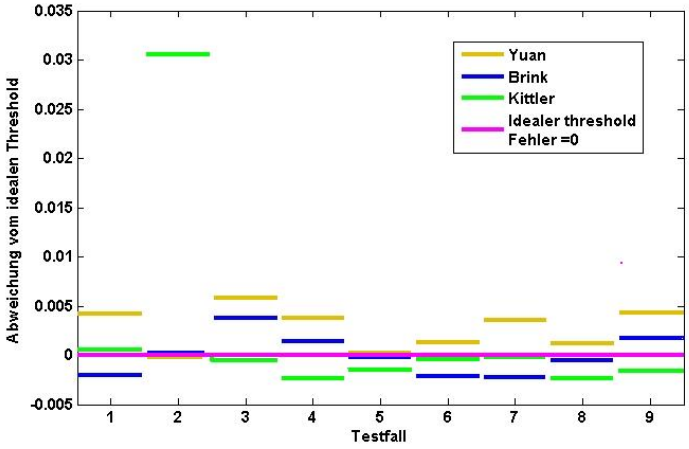


(b) Innenaufnahmen

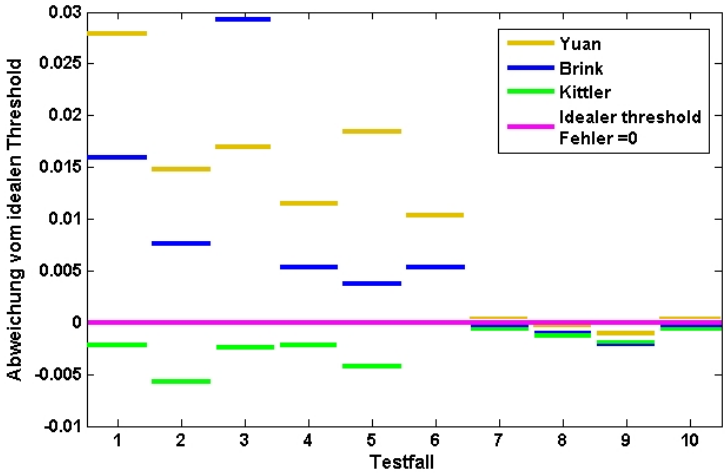
Abbildung 4.3: Lage des idealen *thresholds*, sowie Lage der automatisch bestimmten *thresholds* im Vergleich zum idealen

den Außenaufnahmen liegen die automatisch gelegten *thresholds* (bis auf einen Ausreißer beim Kittler-Verfahren) alle recht nah am idealen *threshold*. Bei den Innenaufnahmen ist die Streuung größer. Insgesamt liegt das Kittler-Verfahren am dichtesten am idealen *threshold*.

Die Abbildungen 4.5a und 4.5b stellen den Median der Fehler der drei Verfahren grafisch dar. Man sieht, dass das Kittler-Verfahren die kleinste Abweichung vom idealen *threshold* besitzt. Die Befürchtung, dass bei Innen- und Außenaufnahmen die Güte der Ergebnisse sehr verschieden sein könnte, hat sich hier nicht bestätigt. Im Gegenteil, der Median der Fehler ist bei allen drei Verfahren bei den Außenaufnahmen sogar kleiner. Eine denkbare Begründung ist: Die Flussberechnungen auf den Innenaufnahmen waren allgemein schlech-



(a) Aussenaufnahmen

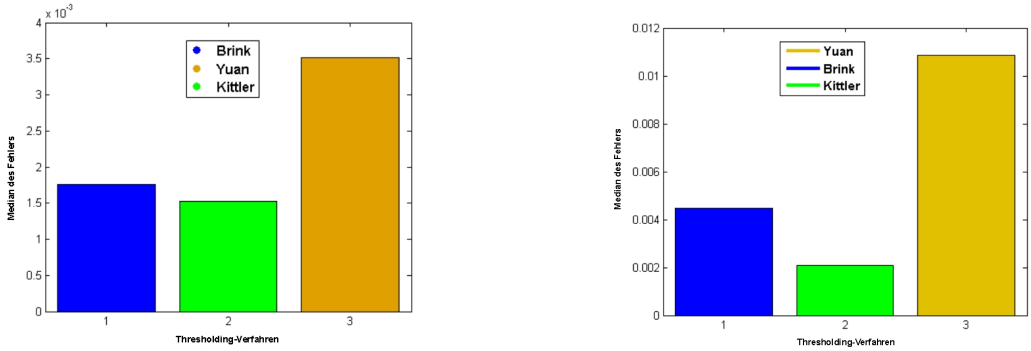


(b) Innenaufnahmen

Abbildung 4.4: Abweichung der automatisch bestimmten von den idealen *thresholds*

ter als bei den Außenaufnahmen. Dies hat dann unweigerlich eine erschwerte Positionierung eines guten *thresholds* zur Folge. Die Innenaufnahmen wurden zum größten Teil auf den Fluren des Instituts aufgenommen, einige auch im Robotik-Labor. Trotzdem die Wände mit Postern ausgestattet sind, existierten wohl zu wenige über die Bilder verfolgbare Helligkeitsunterschiede für eine gute Flussberechnung.

Eine abschließende Bewertung der Lage der automatischen *thresholds* im Vergleich zum idealen wird im Rahmen von Abschnitt 4.6 gegeben.



(a) Aussenaufnahmen (b) Innenaufnahmen

Abbildung 4.5: Mediane der Fehler gegenüber dem idealen *threshold*

4.3 Bewertung der Gesamtverfahren

Möchte man die Güte der drei Verfahren im Hinblick darauf beurteilen, wie sehr sie die Weiterberarbeitung in späteren Prozessierungsschritten zu verbessern vermögen, so bietet sich eine zielorientierte Evaluierung an (siehe auch Abschnitt 3.2). So wird in diesem Abschnitt die Gesamtleistung der implementierten Klassifikationsalgorithmen untersucht. Für jeden Algorithmus wird die Leistung als Klassifikator beurteilt, indem die Kenngrößen der Klassifizierungsleistung berechnet und diskutiert werden. Weiterhin wird für jedes Verfahren das Vermögen, Ausreißer zu erkennen beurteilt.

Die Bewertung wird wieder für Innen- und Außenaufnahmen separat durchgeführt. Grundlage für die Bewertung bilden jeweils fünf typische Ergebnisse für jedes der drei Verfahren. Um die Verfahren zu bewerten benötigt man eine Gegenüberstellung des vom Klassifikator gelieferten Ergebnis' und des tatsächlichen Sachverhalts (das heisst der *ground truth*).
Für die verwendeten Testsequenzen lagen eigentlich keine *ground truth*-Daten vor. Ich habe daher die *ground truth* manuell ermittelt. Dies wird in Abschnitt 3.5 beschrieben.
Die verwendeten *ground truth*-Daten lieferten bei der Berechnung der Kenngrößen in allen Testfällen konsistente Ergebnisse und gestatten verlässliche Rückschlüsse auf die Charakteristika, auf die Stärken und Schwächen der drei implementierten Verfahren.

Sowohl für die Ermittlung der Kenngrößen, als auch für die Beurteilung des Vermögens der Verfahren, Ausreißer zu identifizieren, wurden pro Verfahren und Aufnahmesituation fünf typische Ergebnisse ausgewertet. Die *ground truth* wurde für jedes der 30 bewerteten Ergebnisse jeweils nach dem in Abschnitt 3.5 beschriebenen Verfahren ermittelt.

Obwohl in diesem Abschnitt der gesamte Klassifikator beurteilt wird, also *thresholding* inklusive Nachbearbeitung, werden die Verfahren trotzdem als

Kittler-Verfahren etc. bezeichnet, da das *thresholding* den Unterschied in den Klassifikationsalgorithmen ausmacht. Die Vor- und Nachbearbeitung ist für alle drei Verfahren gleich.

Im ersten Abschnitt werden die Kenngrößen der drei Verfahren angegeben, im zweiten die Fähigkeit Ausreißer zu filtern behandelt.

4.3.1 Bewertung der Verfahren als Klassifikatoren

Um die Kenngrößen zu ermitteln benötigt man eine Gegenüberstellung des vom Klassifikator gelieferten Ergebnis und des tatsächlichen Sachverhalts.
Diese Gegenüberstellung wird für gewöhnlich in einer Wahrheitsmatrix dargestellt. Eine solche Wahrheitsmatrix zur Beurteilung der hier implementierten Klassifikatoren ist in Tabelle 4.1 dargestellt. Sie zeigt alle vier möglichen Kombinationen von Klassifikation und tatsächlichem Sachverhalt.

	Vektor ist IDM	Vektor ist EM
Test klassifiziert als IDM	richtig positiv	falsch positiv
Test klassifiziert als EM	falsch negativ	richtig negativ

Tabelle 4.1: Wahrheitsmatrix für Beurteilung von Klassifikatoren

Damit die Zuordnung zwischen Sachverhalt und Klassifikator-Ergebnis intuitiver ist, habe ich die vier Kombinationsmöglichkeiten fortan folgendermaßen benannt:

- richtig positiv = IDM-Vektor als IDM klassifiziert = IDM→IDM
- falsch positiv = EM-Vektor als IDM klassifiziert = EM→IDM
- richtig negativ = EM-Vektor als EM klassifiziert = EM→EM
- falsch negativ = IDM-Vektor als EM klassifiziert = IDM→EM

Aus den Werten der Wahrheitsmatrix lassen sich dann die Kenngrößen der Klassifikatoren berechnen.
Die sechs Wahrheitsmatrizen für die unterschiedlichen Verfahren und Aufnahmesituationen werden hier nicht extra angegeben. Sie wurden so gewonnen, dass für jedes der verwendeten 30 Ergebnisse die Wahrheitsmatrix-Einträge ermittelt wurden. Pro Aufnahmesituation und Klassifikationsalgorithmus wurden diese Einträge dann gemittelt, um eine Wahrheitsmatrix pro Verfahren und Aufnahmesituation zu erhalten.
Stattdessen werden gleich die aussagekräftigeren Kenngrößen angegeben.

Für jede der Kenngrößen wird die Berechnungsformel angegeben, ihre Bedeutung erläutert und angegeben, welches Verfahren das bezüglich dieser Kenngröße bedeutendste Ergebnis lieferte. Eine Gesamtbewertung der drei Verfahren, bezüglich aller Kenngrößen, zusammen mit der im nächsten Abschnitt vorgenommen Beurteilung der Fähigkeit, Ausreißer zu identifizieren, sowie der Lage der dynamischen *thresholds* im Vergleich zum idealen wird im Abschnitt 4.6 gegeben.
Wichtig ist noch zu bemerken, dass die Kenngrößen nicht unabhängig voneinander sind. Die Kenngrößenpaare Sensitivität und Falsch-Negativ-Rate, sowie Spezifität und Falsch-Positiv-Rate addieren sich jeweils zu eins. Es wurden jeweils beide Größen angegeben, da so die Vorteile eines Verfahrens direkt abgelesen werden können.

	Brink	Kittler	Yuan
Sensitivität	0,85896	0,91279	0,58616
Spezifität	0,95811	0,99539	0,99971
Korrektklassifikationsrate	0,95146	0,99006	0,97281
Falschklassifikationsrate	0,04854	0,00994	0,02719
Relevanz	0,59589	0,93175	0,99282
Segreganz	0,98953	0,99400	0,97201
Falsch-Positiv-Rate	0,04189	0,00461	0,00029
Falsch-Negativ-Rate	0,14104	0,08721	0,41384

Tabelle 4.2: Kenngrößen für Leistung bei Außenaufnahmen

	Brink	Kittler	Yuan
Sensitivität	0,88856	0,92515	0,81481
Spezifität	0,99931	0,99559	1,00000
Korrektklassifikationsrate	0,99129	0,99052	0,98740
Falschklassifikationsrate	0,00871	0,00948	0,01260
Relevanz	0,99020	0,94207	1,00000
Segreganz	0,99136	0,99420	0,98666
Falsch-Positiv-Rate	0,00069	0,00441	0,00000
Falsch-Negativ-Rate	0,11143	0,07485	0,18519

Tabelle 4.3: Kenngrößen für Leistung bei Innenaufnahmen

Sensitivität

$$\text{Sensitivität} = \frac{\text{IDM}\rightarrow\text{IDM}}{\text{IDM}\rightarrow\text{IDM} + \text{IDM}\rightarrow\text{EM}} \tag{4.1}$$

Die Sensitivität, auch Richtig-Positiv-Rate genannt, gibt an die Wahrscheinlichkeit an, dass eine vorhandene IDM auch erkannt wird. Eine Sensitivität von 1.0 bedeutet, dass der Test alle vorhandenen IDM-Vektoren als solche erkennt. Sowohl bei den Innen- als auch bei den Außenaufnahmen schnitt der Kittler-Klassifikator hier am besten ab. Der am wenigsten sensitive in beiden Fällen war der von Yuan.

Falsch-Negativ-Rate

$$\text{Falsch-Negativ-Rate} = \frac{\text{IDM}\rightarrow\text{EM}}{\text{IDM}\rightarrow\text{EM} + \text{IDM}\rightarrow\text{IDM}} \tag{4.2}$$

Die Falsch-Negativ-Rate gibt die Wahrscheinlichkeit an, dass eine vorhandene IDM nicht erkannt wird.
Der Algorithmus mit der kleinsten Falsch-Negativ-Rate in beiden Aufnahmesituationen ist der Kittler-Klassifikator, der mit der größten in beiden Fällen der nach Yuan.

Spezifität

$$\text{Spezifität} = \frac{\text{EM}\rightarrow\text{EM}}{\text{EM}\rightarrow\text{EM} + \text{EM}\rightarrow\text{IDM}} \tag{4.3}$$

Die Spezifität, auch Richtig-Negativ-Rate genannt, gibt die Wahrscheinlichkeit an, dass ein EM-Vektor als solcher klassifiziert wird, es also keinen Fehlalarm durch ein vermeintliches IDM-Objekt gibt. Der spezifischste unter den getesteten Klassifikatoren ist in beiden Testsituationen der nach Yuan. Der Algorithmus mit der kleinsten Spezifität unterscheidet sich für die beiden Testsituationen. Bei den Außenaufnahmen war der Algorithmus nach Brink der am wenigsten spezifische, bei den Innenaufnahmen der von Kittler [1].

Falsch-Positiv-Rate

$$\text{Falsch-Positiv-Rate} = \frac{\text{EM}\rightarrow\text{IDM}}{\text{EM}\rightarrow\text{IDM} + \text{EM}\rightarrow\text{EM}} \tag{4.4}$$

Die Falsch-Positiv-Rate ist die Wahrscheinlichkeit für Fehlalarm, das heisst, dass EM-Vektoren als IDM klassifiziert werden.
Der Algorithmus mit der kleinsten Falsch-Positiv-Rate ist der Yuan. Der Algorithmus mit der größten Falsch-Positiv unterscheidet sich für die beiden

[1] Diese Inkonkruenz wird in Abschnitt 4.4 erläutert.

Testsituationen. Bei den Außenaufnahmen war der Algorithmus nach Brink der mit der größten Gefahr für Fehlalarm, bei den Innenaufnahmen der von Kittler [2].

Relevanz

$$\text{Relevanz} = \frac{\text{IDM}\rightarrow\text{IDM}}{\text{IDM}\rightarrow\text{IDM} + \text{EM}\rightarrow\text{IDM}} \tag{4.5}$$

Die Relevanz, auch Positiver Vorhersagewert oder kurz PPV (engl.: *Positive Predictive Value*) genannt, gibt die Wahrscheinlichkeit an, dass ein als IDM klassifizierter Vektor auch wirklich zu einem IDM-Objekt gehört.
Der Algorithmus mit der relevantesten Klassifikation ist in beiden Testsituationen der von Yuan. Der Algorithmus mit dem kleinsten Relevanz-Wert unterscheidet sich für die beiden Testsituationen. Bei den Außenaufnahmen war der Algorithmus nach Brink der mit der geringsten Relevanz, bei den Innenaufnahmen der von Kittler [3].

Segreganz

$$\text{Segreganz} = \frac{\text{EM}\rightarrow\text{EM}}{\text{EM}\rightarrow\text{EM} + \text{IDM}\rightarrow\text{EM}} \tag{4.6}$$

Die Segreganz, auch Negativer Vorhersagewert oder kurz NPV (engl.: *Negative Predictive Value*) genannt, gibt die Wahrscheinlichkeit an, dass ein als EM klassifizierter Vektor auch wirklich zur EM gehört.
Die beste Leistung bezüglich Segreganz bot in beiden Testsituationen die Kittler-Methode. Den schlechtesten Segreganz-Wert die Yuan-Methode.

Vorläufige Bewertung

Die ermittelten Kenngrößen weisen bei allen drei Verfahren auf gute Klassifikationsleistungen hin. Die extrem hohen (0.99...) oder niederen (0.01...) Werte in den Kenngrößen kommen dadurch zustande, dass die Anzahl der zu klassifizierenden Vektoren in einem Bild grundsätzlich recht hoch ist, während nur bei einer vergleichsweise kleinen Menge die Zuordnung zu einer der beiden Klassen nicht eindeutig ist. In den hier verwendeten Testbildpaaren waren je nach Güte der Flussberechnung zwischen 750 und 2400 Vektoren zu klassifizieren. Die IDM umfasste je nach Bildsituation und Flussgüte zwischen 15 und 150 Vektoren. Ausreißer waren - ebenso nach Flussgüte - unterschiedlich häufig. Die meisten der Vektoren sind durch alle drei Verfahren eindeutig einer der beiden Klassen EM und IDM zuordenbar. Nur bei einer im Gegensatz

[2] Diese Inkonkruenz wird in Abschnitt 4.4 erläutert.
[3] Diese Inkonkruenz wird in Abschnitt 4.4 erläutert.

zur Gesamtzahl der Vektoren kleinen Menge ist die Zuordnung heikel. Nur bei durchschnittlich 18 der durchschnittlich 1530 Vektoren pro Bild waren sich die drei Verfahren nicht einig. Nichtsdestotrotz ist die Zuordnung dieser relativ kleinen Menge relevant. Daher machen auf den ersten Blick geringe Unterschiede im zweiten Nachkommabereich trotzdem einen großen Unterschied in der Klassifikationsleistung aus.

Kittler Die Kittler-Methode besitzt die größte Sensitivität der drei Verfahren. Das heißt, die Wahrscheinlichkeit, dass eine vorhandene IDM erkannt wird ist bei ihr am größten. Dagegen besitzt das Verfahren die kleinste Spezifität der drei Verfahren, das heisst die Gefahr für einen Fehlalarm ist am größten. Die Methode besitzt außerdem die größte Segreganz der drei Verfahren, also ist die Wahrscheinlichkeit, dass ein durch dieses Verfahren als EM klassifizierter Vektor auch wirklich EM ist, groß.

Yuan Das Verfahren nach Yuan besitzt die kleinste Falsch-Positiv-Rate der drei Verfahren. Das heisst die Wahrscheinlichkeit für Fehlalarm ist bei diesem Verfahren am kleinsten. Auch hinsichtlich der Relevanz ist das Yuan-Verfahren am besten: wenn die Yuan-Methode einen Vektor als IDM deklariert, so handelt es sich mit großer Wahrscheinlichkeit wirklich um einen IDM-Vektoren. Dies geht aber auf Kosten der Sensitivität. Diese ist von allen drei Verfahren am kleinsten, das heißt, die Gefahr, eine vorhandene IDM nicht zu erkennen, ist mit diesem Verfahren am größten.

Brink Bei dem Verfahren von Brink liegen die meisten Kenngrößen, bis auf Spezifität, deren Komplementärgröße Falsch-Positiv-Rate, sowie der Relevanz bei den Aussenaufnahmen, zwischen denen der beiden anderen Verfahren. Man könnte also auf den ersten Blick meinen, dass dieses Verfahren dann wahrscheinlich einen guten Kompromiss bietet. Das Verfahren besitzt aber einen gravierenden Nachteil, der mit der Art und Weise zusammenhängt, nach welchen Kriterien das Brink-Pendock-Verfahren den *threshold* ermittelt. Dieser wird in Abschnitt 4.4 diskutiert.

4.3.2 Eignung der Verfahren zum Filtern von Ausreißern

Der durchschnittliche Prozentsatz erkannter Ausreißer wurde für jedes Verfahren folgendermaßen ermittelt: Für jedes der 30 Ergebnisse wurde die Anzahl echter Ausreißer ermittelt (siehe Abschnitt 3.5). Für jedes der 10 Testbildpaare wurde die maximale Anzahl von Ausreißern unter den Leistungen der drei Verfahren an diesem Bildpaar festgestellt. Diese wurde als Anzahl vorhandener Ausreißer unter den Flüssen dieses Bildpaars betrachtet. Die Leistung jedes

Verfahrens an diesem Testbildpaar wurde dann relativ zu dieser maximalen Ausreißeranzahl angegeben.

Ein Beispiel: das Brink-Verfahren stellt unter den Flussvektoren eines Bildpaars 12 Ausreißer fest, das Kittler-Verfahren 3 und das Yuan-Verfahren 6. Die maximale Anzahl festgestellter Ausreißer beträgt 12 durch das Brink-Verfahren. Das Brink-Verfahren erkennt also 100% der Ausreißer, das Kittler-Verfahren 25% und das Yuan-Verfahren 50%.

Diese für jedes der Testbildpaare ermittelten Prozentwerte wurden dann über die beiden Bildklassen, Innen- und Außenaufnahmen, jeweils gemittelt und so ein durchschnittlicher Prozentsatz erkannter Ausreißer ermittelt. Tabelle 4.4 zeigt die durchschnittliche Prozentzahl an Ausreißern, welche die Verfahren erkannten. Das Kittler-Verfahren ist das Verfahren, welches die meisten Ausreißer zu identifizieren vermag.

Durchschnittlicher Prozentsatz erkannter Ausreißer	Brink	Kittler	Yuan
Aussenaufnahmen	60%	85%	24%
Innenaufnahmen	59%	99%	37%

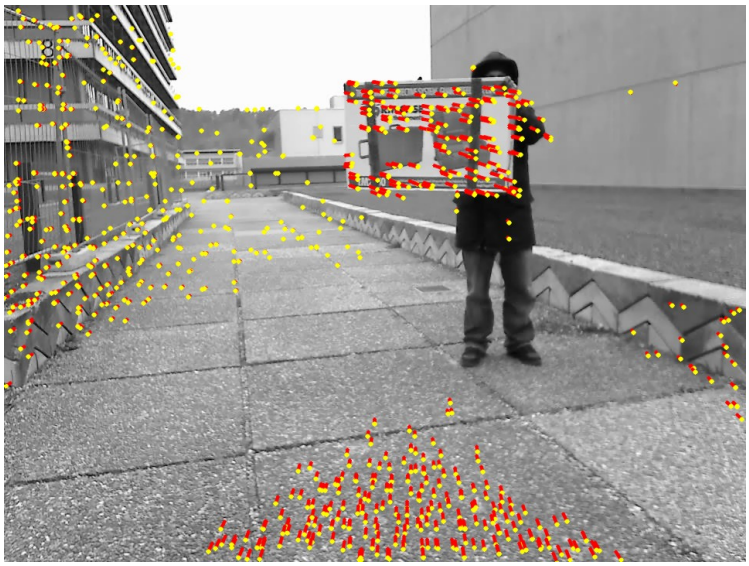
Tabelle 4.4: Durchschnittliche Anzahl erkannter Ausreißer (auf Ganzzahl gerundet)

4.4 Erklärung der Inkongruenz

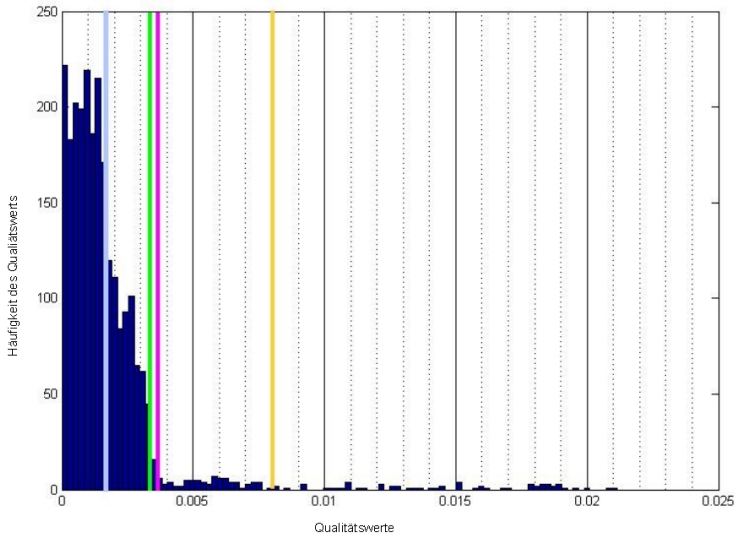
Bei fast allen durchgeführten Bewertungen nehmen die Kittler- und die Yuan-Methode die beiden Extrempositionen ein, währen der Brink-Algorithmus in der Mitte liegt. Ausnahmen von dieser Anordnung der Verfahren bilden lediglich die Spezifität, ihre Komplementärgröße die Falsch-Positiv-Rate, sowie die Relevanz bei den Außenaufnamen. Bei diesen nimmt die Yuan-Methode die eine Extremposition und die Brink-Methode die andere Extremposition ein.

Diese Inkongruenz gegenüber allen anderen Ergebnissen lässt jedoch erklären. Das Brink-Verfahren liefert in einer nicht zu vernachlässigenden Anzahl von Fällen eine unbrauchbare Trennung von EM und IDM. Zu viele EM-Vektoren werden der IDM zugeschlagen. Abbildung 4.6a zeigt ein Beispiel einer solchen Fehlklassifikation, Abbildung 4.6b das zugehörige Reprojektionsfehler-Histogramm der Flussvektoren und den vom Brink-Verfahren gelegten *threshold* in hellblau. Eine solche Positionierung des *thresholds* durch das Brink-Verfahren kommt durch die dem Verfahren zugrundeliegenden Gütebewertung zustande.

Als Maß für die Güte des *thresholds* wird die Kreuzentropie zwischen den beiden Klassifikationen verwendet, die als Maß für den Informationsgewinn interpretiert wird. In Fällen, in denen die Qualitätswert-Peaks für EM und IDM



(a) Ein Schwachpunkt des Brink-Verfahrens: der *threshold* liegt zu niedrig. In der Folge werden zu viele EM-Vektoren zur IDM klassifiziert.



(b) Das Histogramm der Qualitätswerte. blau = Brink-Pendock-*threshold*, magenta = idealer *threshold*, grün = Kittler-*threshold*, gelb = Yuan-*threshold*

Abbildung 4.6: Schwachpunkt der Brink-Methode

in den Histogrammen sehr nah beieinander liegen, wie in Abbildung 4.6b, neigt das Brink-Verfahren dazu, den *threshold* mitten in den Peak zu legen. Da eine solche Klassifikation in nicht zu vernachlässigender Anzahl auftritt, wurde eines dieser Ergebnisse (Abbildung 4.6a) zu den Testbildern für die Außenaufnahmen hinzugenommen. Dies führt beim Brink-Verfahren bei den Außen-

aufnahmen zur geringsten Relevanz, zur kleinsten Spezifität und zur größten Falsch-Positiv-Rate, während das Verfahren in allen anderen Fällen zwischen den beiden anderen Verfahren platziert ist. Obwohl das Brink-Verfahren durch die häufige mittlere Platzierung als ein guter Kompromiß zwischen den beiden anderen Verfahren scheint, liefert es doch durch diese Eigenschaft in manchen Fällen unbrauchbare Ergebnisse und ist so wenig geeignet für die zuverlässige Trennung von EM und IDM.

4.5 Übergeordnete Bewertungskriterien

Welche der, nun von verschiedenen Seiten betrachteten und bewerteten, Eigenschaften der Klassifikationsalgorithmen man nun als kritisch und ausschlaggebend betrachtet, hängt hauptsächlich von der Art der Anwendung, der Art und Funktion des übergeordneten Algorithmus ab, in den man den Klassifikator einbetten möchte.
Ich möchte hier zwei mögliche Anwendungsszenarien beschreiben.

Das erste ist der von Yuan, Recktenwald und Mallot entwickelte Algorithmus zur 3D-Navigation von UAVs [Yuan et al., 2008]. Mit diesem Verfahren soll die Flugrichtungsbestimmmung von Fluginsekten nachgeahmt werden. Von einigen Fluginsekten ist bekannt, dass sie so fliegen, dass der optischen Fluss auf beiden Augen ausgeglichen wird. So bleiben sie möglichst in der Mitte zwischen zwei möglichen Hindernissen rechts und links [Srinivasan et al., 1996]. Bei fahrenden Robotern wurde so ein Korridor-Zentrierungsverhalten vielfach erfolgreich implementiert, in komplexeren Umgebungen funktionieren diese in Korridoren erfolgreichen Verfahren jedoch häufig nicht, da sie keine frontale Hinderniserkennung und -vermeidung berücksichtigen. Bei Flugrobotern ergeben sich noch weitere Probleme, da zusätzlich die Flughöhe kontrolliert werden muss um Kollisionen mit Decke oder Boden zu vermeiden. Yuan et. al. entwickelten einen *fuzzy*-Algorithmus zur Navigation von Flugrobotern, der sich in kritischer Nähe befindliche Hindernisse detektiert und eine hindernissvermeidende Flugrichtung bestimmt. Es gibt fünf *fuzzy* Flugrichtungen: *left*, *forward left*, *forward*, *forward right* und *right*. In Abhängigkeit der Sensorinformationen wird jeder möglichen Bewegungsrichtung eines von folgenden drei möglichen *fuzzy*-Prädikaten zugeordnet: FA (*favorable*), AC (*acceptable*) oder NA (*not acceptable*).

Nimmt man eine solche Anwendung an, so muss man die Objektgrenzen des IDM-Objekts nicht genau kennen. Es genügt, die Richtung zu kennen, in der sich dieses Objekt befindet, um dann in eine andere Richtung abzudrehen. Für diese Anwendung des Klassifikationsalgorithmus benötigt man auch kein Clustering, da es ja eigentlich egal ist, wie viele Objekte sich im Blickfeld befinden, und es ja nur darum geht, den Weg so zu wählen, dass man den Hindernissen ausweicht

In einem anderen denkbaren Anwendungsfall, könnte man die Grenzen eines IDM-Objekts genau detektieren, und eventuell sogar dessen Bewegungsparameter ermitteln wollen, so dass man nicht versehentlich direkt auf Kollisionskurs mit dem Objekt geht, sondern ihm frühzeitig ausweichen kann. In solch einem Szenario wäre es erstens wichtig, zu wissen wie viele IDM-Objekte es im Blickfeld gibt, und welche Vektoren zusammen ein Objekt bilden. Zum anderen wäre es wichtig, möglichst alle zum IDM gehörigen Vektoren zu identifizieren, da, wenn man die Bewegungsparameter des IDM schätzen wollte, man eine

Mindestanzahl von Vektoren benötigt, damit eine Schätzung überhaupt möglich ist.

4.6 Gesamtbewertung

Die Trennung der Tests in Innen- und Außenaufnahmen wäre nicht nötig gewesen, die Ergebnisse sind konsistent für Innen- und Außenaufnahmen. Hier wird nun eine überblickende Gesamtbewertung der drei Verfahren bezüglich aller durchgeführten Tests gegeben.

4.6.1 Kittler

Beim Kittler-Verfahren ist der Fehlermedian der Abweichung des automatisch gelegten *thresholds* gegenüber dem idealen am kleinsten (Abschnitt 4.2). Er besitzt die größte Sensitivität der drei Verfahren (Abschnitt 4.1), das heißt dass die überwiegende Mehrzahl der IDM-Vektoren richtig erkannt wird. Die Falsch-Positiv-Rate, die Wahrscheinlichkeit für Fehlalarm, ist am größten (Abschnitt 4.4). Er besitzt die geringste Relevanz (Abschnitt 4.5), dafür die größte Segreganz (Abschnitt 4.6). Die Fähigkeit, Ausreißer unter den Flussvektoren zu detektieren ist beim Kittler-Verfahren am größten (Abschnitt 4.3.2). Mit der Kittler-Methode kann man den größten Anteil der drei Verfahren an vorhandenen IDM-Vektoren richtig zuordnen. Dies geht jedoch auf Kosten davon, dass EM-Vektoren mit schlechterer Qualität, beziehungsweise Ausreißer (der Übergang ist fließend) ebenfalls der IDM zugeordnet werden. Dies kann zur Detektion vermeintlicher IDM-Objekte führen. Verwendet man die Kittler-Methode, so muss man Wege finden, diese vermeintlichen IDM-Objekte zuverlässig als falsch zu identifizieren. Dieses Verhalten ist auf der anderen Seite wieder von Vorteil, da so Ausreißer unter den Flussvektoren identifiziert werden können. Das Kittler-Verfahren ist besonders vorteilhaft, wenn man möglichst alle der IDM gehörigen Vektoren identifizieren möchte, anstatt nur festzustellen, dass IDM vorhanden ist. Weißt das Objekt genügend Gradienten für eine gute Flussbestimmung auf, und ist eine gute Egomotionschätzung anhand der EM-Vektoren möglich, so können anhand dieser Methode sogar die Grenzen des IDM-Objekts ziemlich genau detektiert werden.

4.6.2 Brink

Das Brink-Verfahren liegt in den Leistungsbewertungen zumeist in der Mitte zwischen dem Kittler- und dem Yuan-Verfahren. Es ist jedoch durch relativ häufig vorkommende Fehlklassifikationen nicht für die Trennung von EM und IDM anhand der Qualitätswerte geeignet (Abschnitt 4.4).

4.6.3 Yuan

Die Yuan-Methode legt den *threshold* meist höher als den idealen *threshold*. Der Median der Abweichung des automatisch gelegten *thresholds* gegenüber dem idealen am größten (Abschnitt 4.2). Sie besitzt die größte Falsch-Negativ-Rate (Abschnitt 4.2), das heißt, die Wahrscheinlichkeit, dass IDM-Vektoren fälschlich der EM zugeordnet werden, ist am größten. Sie besitzt die größte Relevanz(Abschnitt 4.5), das heißt die Wahrscheinlichkeit, dass ein als IDM klassifizierter Vektor wirklich IDM ist, ist mit dieser Methode am größten (Abschnitt 4.3.2). Die Möglichkeit, Ausreißer zu filtern, ist mit dieser Methode am geringsten ausgeprägt. Mit der Yuan-Methode wird meist ein signifikanter Teil der eigentlich zur IDM gehörenden Vektoren der EM zugeschlagen. IDM-Objekte werden jedoch stets erkannt, nur werden nicht alle zugehörigen Vektoren detektiert. Ein von der Yuan-Methode detektiertes Objekt ist mit der größten Wahrscheinlichkeit der drei Methoden auch wirklich vorhanden. Dafür ist die Möglichkeit, die Yuan-Methode zur Entfernung von Ausreißern zu verwenden, am geringsten ausgeprägt. Die Yuan-Methode stellt eine gute Wahl für Anwendungen wie den in Abschnitt 4.5 beschriebenen *fuzzy*-Navigations-Algorithmus dar. Das Anliegen in diesem Fall ist es, IDM-Objekte zuverlässig zu erkennen, die genauen Objektgrenzen, beziehungsweise die richtige Klassifikation wirklich aller zum IDM-Objekt gehörender Flussvektoren spielt eine eher untergeordnete Rolle.

Kapitel 5

Schlussfolgerung und Ausblick

Die durchgeführten Versuche zur Trennung von EM und IDM mit dynamischem *thresholding* waren erfolgreich und zeigen, dass das dynamische *thresholding* ein gute Wahl zu dieser Klassifikation ist. Von den drei getesteten Verfahren erwies sich das Brink-Verfahren als unbrauchbar, das Kittler- und das Yuan-Verfahren jedoch sind zur Detektion von IDM unter Berücksichtigung der Eigenbewegung der Kamera gut geeignet. Sie unterscheiden sich jedoch in einigen Charakteristika, die sie für unterschiedliche Anwendungsschwerpunkte geeignet macht.

Im folgenden werden kurz einige Verbesserungsvorschläge diskutiert. Man könnte das dynamische Clustern, also den *G-means*-Algorithmus, mit einem anderen statistischen Testverfahren kombinieren, um die Anzahl der Cluster dynamisch anzupassen. Der verwendete Anderson-Darling-Test auf Normalverteilung legt Gewicht auf die Randbereiche der Verteilung. Eventuell wäre ein Test auf Normalverteilung, der die mittleren Quantile höher gewichtet, wie beispielsweise der Kolmogorow-Smirnow-Test, geeigneter und könnte eine Auftrennung der IDM-Objekte in zu viele Einzelcluster entgegenwirken.

Da die Detektion von IDM-Objekten abhängig ist von der Güte der Flussberechnung, sowie der Egomotion-Schätzung, wäre es von Vorteil ein Maß für diese beiden zu haben, um vor der Berechnung schon Abschätzen zu können, ob sich eine Klassifikation überhaupt lohnt, oder ob die Ergebnisse ohnehin fehlerhaft sein werden.

Ein nächster Schritt könnte es sein, den Fluss und die Objekte über mehrere Bilder hinweg zu betrachten. Vermeintliche Objekte, die eigentlich aus Ausreißern bestehen, könnten so detektiert werden, da sie nicht über mehrere Bilder hinweg auftauchen. Eventuell könnte man um die Position der IDM-Objekte vorherzusagen bzw. ihre Schätzung zu stabilisieren einen Kalman-Filter verwenden.

Literaturverzeichnis

[Adiv, 1989] Adiv, G. (1989). Inherent ambiguities in recovering 3-d motion and structure from a noisy flow field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):477–489.

[Al-Daoud and Roberts, 1996] Al-Daoud, M. B. and Roberts, S. A. (1996). New methods for the initialisation of clusters. *Pattern Recogn. Lett.*, 17(5):451–455.

[Anderson and Darling, 1952] Anderson, T. W. and Darling, D. A. (1952). Asymptotic theory of certain goodness of fit criteria based on stochastic processes. *Annals of Mathematical Statistics*, 23:193–212.

[Baker et al., 2007] Baker, S., Scharstein, D., Lewis, J. P., Roth, S., Black, M. J., and Szeliski, R. (2007). A database and evaluation methodology for optical flow. In *ICCV*, pages 1–8. IEEE.

[Belkasim et al., 1991] Belkasim, S. O., Shridhar, M., and Ahmadi, M. (1991). Pattern recognition with moment invariants: a comparative study and new results. *Pattern Recogn.*, 24(12):1117–1138.

[Blazek, 2009] Blazek, S. (2009). Ein biologisch motivierter ansatz zur selbstlokalisation eines agenten im raum mittels rundumsicht-bilder. Master's thesis, Universität Tübingen.

[Bouguet, 2002] Bouguet, J. Y. (2002). Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm. `Jean-YvesBouguet`.

[Brink and Pendock, 1996] Brink, A. and Pendock, N. (1996). Minimum cross-entropy threshold selection. *Pattern Recognition*, 29:179–188.

[Burgoyne et al., 2007] Burgoyne, J. A., Pugin, L., Eustace, G., and Fujinaga, I. (2007). A comparative survey of image binarisation algorithms for optical recognition on degraded musical sources.

[Carlo Tomasi and Takeo Kanade, 1991] Carlo Tomasi and Takeo Kanade (1991). Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University.

[christophe Zufferey and Floreano, 2006] christophe Zufferey, J. and Floreano, D. (2006). Fly-inspired visual steering of an ultralight indoor aircraft. *IEEE Transactions on Robotics*, pages 137–146.

[Coombs and Roberts, 1993] Coombs, D. and Roberts, K. (1993). Centering behavior using peripheral vision. In *In CVPR*, pages 440–445. IEEE, IEEE Press.

[Duller, 2008] Duller, C. (2008). *Einführung in die nichtparametrische Statistik mit SAS und R.* Physica-Verlag, auflage: 1 (september 2008) edition.

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

[Gennert and Negahdaripour, 1987] Gennert, M. A. and Negahdaripour, S. (1987). Relaxing the brightness constancy assumption in computing optical flow. Technical report, Cambridge, MA, USA.

[Hamerly and Elkan, 2003] Hamerly, G. and Elkan, C. (2003). Learning the k in k-means. In *in Proc. 17th NIPS*, page 2003.

[Hartigan, 1975] Hartigan, J. A. (1975). *Clustering Algorithms (Probability & Mathematical Statistics).* John Wiley & Sons Inc.

[Hartley and Zisserman, 2004] Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition.

[He et al., 2004] He, J., Lan, M., lim Tan, C., yuan Sung, S., and boon Low, H. (2004). Initialization of cluster refinement algorithms: A review and comparative study. In *in Proceedings of International Joint Conference on Neural Networks (IJCNN*, pages 297–302.

[He et al., 2003] He, J., Tan, A.-H., Tan, C.-L., and Sung, S.-Y. (2003). *On Quantitative Evaluation of Clustering Systems.* Kluwer Academic Publishers.

[Horn and Schunck, 1981] Horn, B. and Schunck, B. (1981). Determing optical flow. *Artificial Intelligence*, 17:185–203.

[Horn, 1986] Horn, B. K. P. (1986). *Robot Vision (MIT Electrical Engineering and Computer Science).* The MIT Press, mit press ed edition.

[Kanatani, 1991a] Kanatani, K. (1991a). Computational projective geometry. *CVGIP: Image Underst.*, 54(3):333–348.

[Kanatani, 1991b] Kanatani, K. (1991b). Computational projective geometry. *CVGIP: Image Underst.*, 54(3):333–348.

[Kanatani, 1993] Kanatani, K. (1993). 3-d interpretation of optical flow by renormalization. *Int. J. Comput. Vision*, 11(3):267–282.

[Katsavounidis et al., 1994] Katsavounidis, I., Jay, C., and Zhang, Z. (1994). A new initialization technique for generalized lloyd iteration. *Signal Processing Letters, IEEE*, 1(10):144–146.

[Kearney et al., 1987] Kearney, J. K., Thompson, W. B., and Boley, D. L. (1987). Optical flow estimation: an error analysis of gradient-based methods with local optimization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(2):229–244.

[Kittler and Illingworth, 1986] Kittler, J. and Illingworth, J. (1986). Minimum error thresholding. *Pattern Recogn.*, 19(1):41–47.

[Klappstein et al., 2009] Klappstein, J., Vaudrey, T., Rabe, C., Wedel, A., and Klette, R. (2009). Moving object segmentation using optical flow and depth information. In *PSIVT*, pages 611–623.

[Kullback, 1959] Kullback, S. (1959). *Information theory and statistics*. John Wiley and Sons., New York.

[Lucas and Kanade, 1981] Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision (darpa). In *Proceedings of the 1981 DARPA Image Understanding Workshop*, pages 121–130.

[Mallot and Allen, 2000] Mallot, H. and Allen, J. (2000). *Computational Vision: Information Processing in Perception and Visual Behavior*. MIT Press.

[Naderi, 2007] Naderi, M. (2007). Implementierung eines echtzeitverfahrens zur erstellung von bildmosaiken aus endoskopischen videosequenzen. Master's thesis, University of Applied Sciences Cologne.

[Negahdaripour, 1995] Negahdaripour, S. (1995). Revised interpretation of optical flow for dynamic scene analysis. In *SCV95*, pages 473–478.

[Negahdaripour, 1998] Negahdaripour, S. (1998). Revised definition of optical flow: Integration of radiometric and geometric cues for dynamic scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(9):961–979.

[Negahdaripour and Lanjing, 1995] Negahdaripour, S. and Lanjing, J. (1995). Direct recovery of motion and range from images of scenes with time-varying illumination. In *SCV95*, pages 467–472.

[Negahdaripour and Yu, 1993] Negahdaripour, S. and Yu, C. (1993). A generalized brightness change model for computing optical flow. In *ICCV93*, pages 2–11.

[Niblack, 1985] Niblack, W. (1985). *An introduction to digital image processing*. Strandberg Publishing Company, Birkeroed, Denmark.

[Pena et al., 1999] Pena, J., Lozano, J., and Larranaga, P. (1999). An empirical comparison of four initialization methods for the k-means algorithm. 20(10):1027–1040.

[Recktenwald, 2009] Recktenwald, F. (2009). *Dissertation in Bearbeitung*. PhD thesis, Lehrstuhl Kognitive Neurowissenschaften, Universität Tübingen.

[Redmond and Heneghan, 2007] Redmond, S. J. and Heneghan, C. (2007). A method for initialising the k-means clustering algorithm using kd-trees. *Pattern Recogn. Lett.*, 28(8):965–973.

[Rüger, 2002] Rüger, B. (2002). *Test- und Schätztheorie, Bd. 2, Statistische Tests*. Oldenbourg-Verlag.

[Sezgin and Sankur, 2004] Sezgin, M. and Sankur, B. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–168.

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600.

[Srinivasan et al., 1996] Srinivasan, M. V., Zhang, S. W., Lehrer, M., and Collett, T. S. (1996). Honeybee navigation en route to the goal: Visual flight control and odometry. *Journal of Experimental Biology*, 199:237–244.

[Stephens, 1974] Stephens, M. A. (1974). Edf statistics for goodness of fit and some comparisons. *Journal of the American Statistical Association*, 69(347):730–737.

[Stephens, 1976] Stephens, M. A. (1976). Asymptotic results for goodness-of-fit statistics with unknown parameters. *Ann. Statist.*, 4(2):357–369.

[Tian et al., 1996] Tian, T. Y., Tomasi, C., and Heeger, D. J. (1996). Comparison of approaches to egomotion computation. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:315.

[Trier and Jain, 1995] Trier, O. D. and Jain, A. K. (1995). Goal-directed evaluation of binarization methods. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(12):1191–1201.

[Tsai, 1986] Tsai, R. (1986). An efficient and accurate camera calibration technique for 3d machine vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, FL*, pages 364–374.

[Wedel et al., 2008] Wedel, A., Pock, T., Braun, J., Franke, U., and Cremers, D. (2008). Duality tv-l1 flow with fundamental matrix prior. In *Image Vision and Computing*, Auckland, New Zealand.

[Yuan, 2004] Yuan, C. (2004). Simultaneous tracking of multiple objects for augmented reality applications.

[Yuan et al., 2008] Yuan, C., Recktenwald, F., and Mallot, H. A. (2008). Autonomous 3d navigation of unmanned aerial vehicles based on optical flow. *Spatial Cognition 2008 Workshop on Biologically Motivated Models of Spatial Behaviour: Insights from Animals*, pages 1–4.

Anhang A

Anhang

A.1 Weiteres Beispiel zur Wahl des optimalen *thresholds*

Ein Beispiel für ein Bildpaar, bei dem die Lage des optimalen *thresholds* nicht eindeutig war:
Abgebildet sind hier wieder die relevantesten der 100 *threshold*-Varianten. Abbildung A.1a und A.1b begründen die Wahl zur Untergrenze des idealen *thresholds* an der Untergrenze von *bin* 5. Legt man den *threshold* tiefer, an die Untergrenze von *bin* 4, so wird eine ganze Gruppe von EM-Vektoren fälschlicherweise der IDM zugeschlagen. Daher erscheint *bin* 5 die Untergrenze zu sein.
Die Wahl der Obergrenze verdeutlichen die Abbildungen A.1c, A.1d, A.1e und A.1f. Die Obergrenze für den idealen *threshold* liegt an der Untergrenze von *bin* 22. Legt man den *threshold* an die Untergrenze von *bin* 21, so ändert sich nichts an der Zuordnung der Vektoren zu den beiden Klassen gar nichts (und auch wenn man den *threshold* an die Untergrenze von *bin* 20 oder 19 legt, ändert sich an der Zuordnung nichts). Legt man den *threshold* jedoch an die Untergrenze von *bin* 23, so erhöht sich die Anzahl, der IDM-Vektoren, die fälschlicherweise der EM zugerechnet werden (wenn man die rechten Hälften von Abbildung A.1d und A.1e vergleicht, so kann man sehen, dass in Abbildung A.1e im Vergleich zu Abbildung A.1d Vektoren hinzugekommen sind). Erhöht man den *threshold* weiter, so kommen bei jeder Änderung in der rechts Hälfte der Bilder zunehmend falsch klassifizierte IDM-Vektoren hinzu A.1f. Aus diesem Grund, wurde die Obergrenze für den idealen *threshold* für dieses Bildpaar an der Untergrenze von *bin*22 festgelegt.

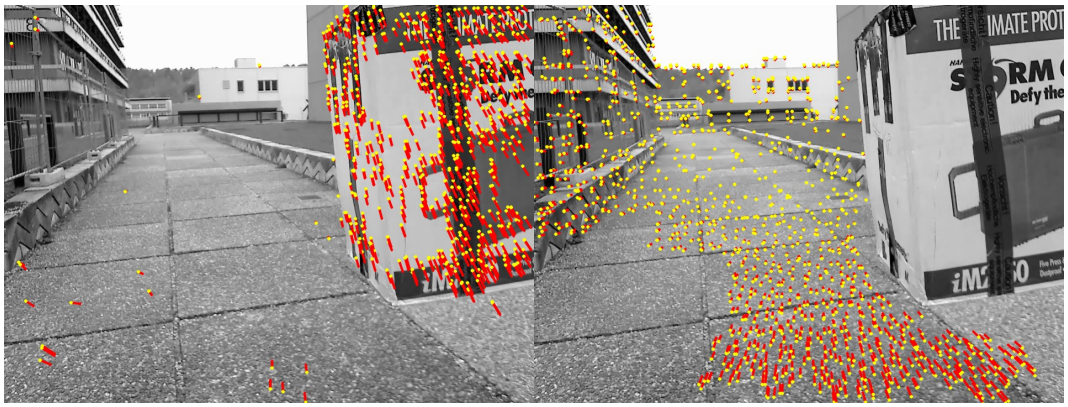


Abbildung A.1a: Der *threshold* liegt an der Untergrenze von $bin4$

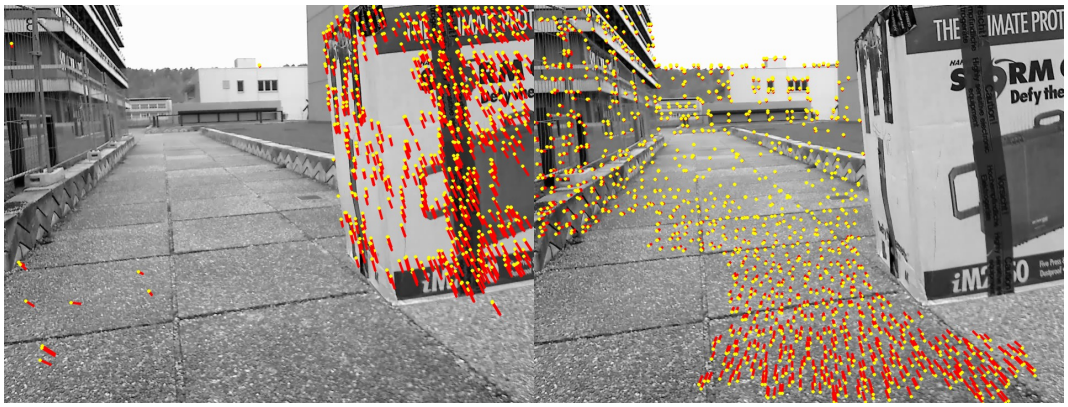


Abbildung A.1b: Der *threshold* liegt an der Untergrenze von $bin5$

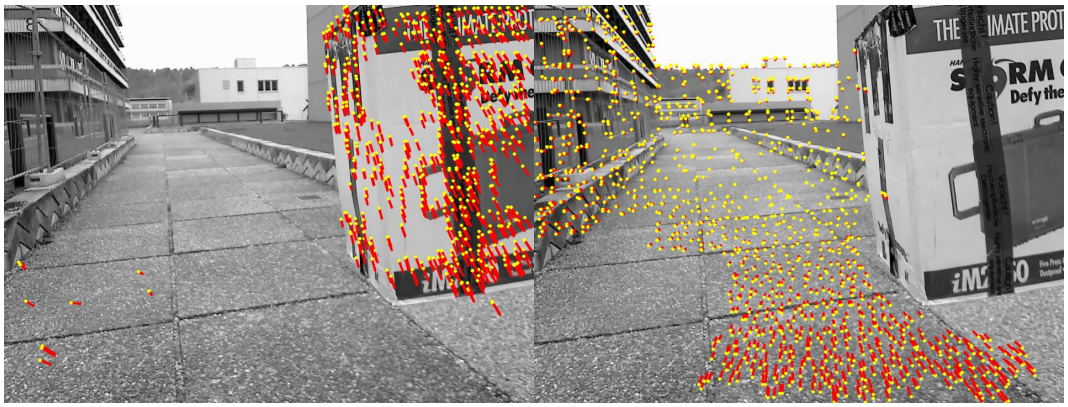


Abbildung A.1c: Der *threshold* liegt an der Untergrenze von $bin21$

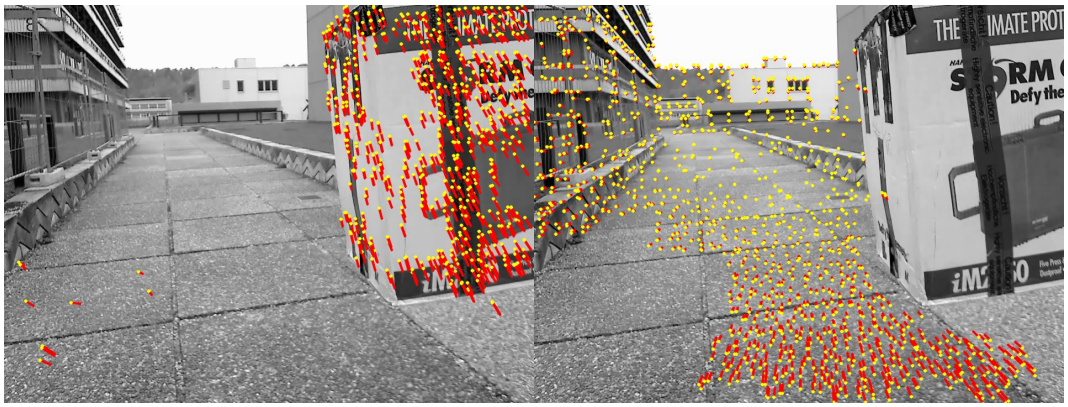


Abbildung A.1d: Der *threshold* liegt an der Untergrenze von $bin22$

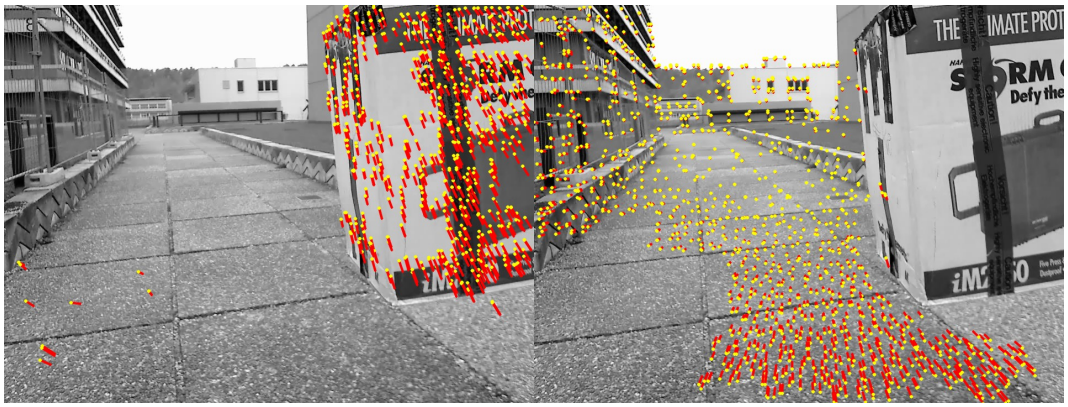


Abbildung A.1e: Der *threshold* liegt an der Untergrenze von $bin23$

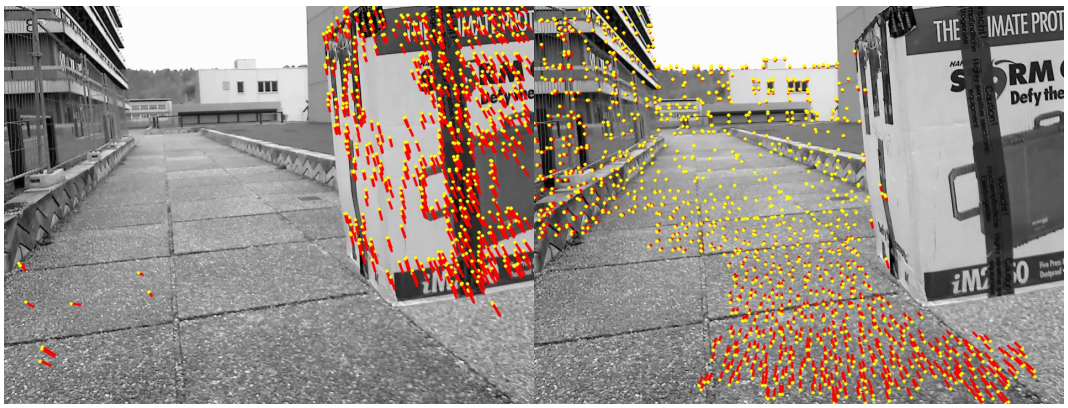


Abbildung A.1f: Der *threshold* liegt an der Untergrenze von $bin24$