Technische Universität München
Lehrstuhl für Integrierte Systeme

Rechenleistungsadaptive Algorithmen zur videobasierten Umgebungserkennung

Colin Claus Estermann

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing. habil. Gerhard Rigoll
Prüfer der Dissertation:	1.	apl. Prof. Dr.-Ing. habil. Walter Stechele
	2.	Univ.-Prof. Dr.-Ing. Eckehard Steinbach

Die Dissertation wurde am 12.05.2009 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 21.10.2009 angenommen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Der Computer hat als Hilfsmittel des Menschen einen unvergleichbaren Siegeszug hinter sich. Beginnend mit den raumfüllenden Anlagen der 50er Jahre, die nur für wenige Spezialisten nutzbar waren, bis hin zum heute omnipräsenten und von jedem verwendbaren PC. Parallel dazu entwickelten sich die Methoden der Mensch-Maschine-Kommunikation: Wo zu Beginn noch mit Lochkarten gearbeitet wurde, folgte bald rein textuelle Kommunikation und schließlich die heutige intuitive zeigerbasierte Ein- und graphisch dominierte Ausgabe.
Die momentan wohl ergonomischste Methode der Mensch-Maschine-Kommunikation stellt dabei die „Augmented Reality“ (AR, Erweiterte Realität) dar: Die Eingabe findet über ein automatisches Erkennen der Umgebung durch entsprechende Bildanalyse statt, die Ausgabe der jeweiligen Anwendung wird dem Benutzer direkt und intuitiv in dessen Sichtfeld eingeblendet. Voraussetzung dafür ist ein „See-through-device“, also ein Display, das in Echtzeit das aktuelle Bild einer damit verbundenen Videokamera wiedergibt, in das die Information eingeblendet werden soll [1].
Die eigentliche Herausforderung der AR besteht nicht im Einblenden virtueller Objekte selbst, sondern im Erkennen der Umgebung, um virtuelle Objekte oder Informationen paßgenau, perspektivisch korrekt und damit für den Anwender intuitiv in die Szene einzufügen. Entsprechend sollen folgende Grundfunktionalitäten betrachtet werden:

1. Tiefenschätzung
   Ermittlung der relativen Abstände zwischen Gerät und den Inhalten der aufgenommenen Szene, mit dem Ziel eines Tiefenprofils, das ein Zuordnen von Objekten zu Vorder- oder Hintergrund ermöglicht.

[1] Das oft fälschlich mit AR gleichgesetzte „Markerbased Tracking“, also das Einblenden von Objekten auf vordefinierte, künstliche Markierungen in der Szene stellt dabei lediglich eine vereinfachte Untergruppe dar (vgl. Abschnitt 1).

2. Tracking
   Mitverfolgen von Objekten, um virtuelle Texturen oder Objekte entsprechend perspektivisch korrekt in die Szene einzublenden.

Doch eben der grundsätzliche Echtzeitanspruch der AR setzt einerseits eine adäquate Rechenleistung voraus. Andererseits sind Informationen zur Umgebung des Benutzers genau dann besonders sinnvoll, wenn ihm diese unbekannt, er also mobil ist. Diese Kombination aus Rechenleistung und Mobilität wird dabei in der Regel durch eine Mobilisierung der Hardware erreicht (vgl. Abbildung 1.2 (c)), die zwar praktikabel, nicht jedoch praktisch ist und diese Systeme von einem Alltagsgebrauch ausschließt.
Mit der sich immer mehr durchsetzenden Kamera (vgl. Abbildung 1.2 (a)), den immer leistungsfähigeren Prozessoren (vgl. Abbildung 1.2 (b)) und seinem Display stellt das Mobilgerät, insbesondere -telefon, eine nicht nur ideale, sondern bereits weit verbreitete und akzeptierte Plattform für solche Anwendungen dar.
Der hier verfolgte Ansatz versucht entsprechend, diese bereits mobile, jedoch rechenleistungsschwache Hardware durch Anpassung der Verfahren für Methoden der AR zu nutzen. Um dabei einerseits stets die optimale Qualität zu gewährleisten, andererseits die knappe Ressource der Rechenleistung nie zu verschwenden, besteht das Ziel der Arbeit in einer skalierbaren Algorithmik, die sich selbst während der Programmlaufzeit an die (momentan) verfügbare Rechenleistung anpaßt und somit die Effizienz des Verfahrens, also den Zusammenhang aus erzielter Qualität und dazu nötigem Aufwand, optimiert. Hierbei sind ohne Einschränkung der Allgemeinheit zwei Grundszenarien denkbar:

1. Statische Zeitvoreinstellung
   Bei Prozeßbeginn wird dem Algorithmus ein fester Anteil der Systemleistung zugewiesen, die er konstant beibehält.

2. Dynamische Zeitanpassung
   Der Algorithmus optimiert sich während des Programmlaufs selbständig und paßt seine Laufzeit der aufgrund anderer Prozesse schwankenden Systemleistung an.

Ferner kann so auch ein problemloser Wechsel des Algorithmus zwischen unterschiedlichen Plattformen erzielt werden. Alle Anpassungen sollen daher ungleich dem Prinzip des sich auf Mikrobefehlsebene auswirkenden „selbstmodifizierenden Codes“[1] auf rein algorithmischer Ebene stattfinden und so eine optimale

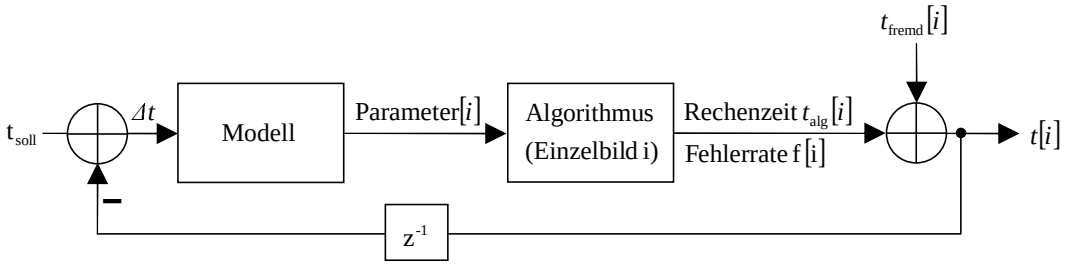


Abbildung 1.1.: Der Rechenzeitaufwand t_i für ein Einzelbild i setzt sich zusammen aus dem Rechenzeitaufwand $t_{alg}[i]$ des Algorithmus und dem dynamischen Rechenzeitaufwand $t_{fremd}[i]$ anderer Prozesse. Um stets Echtzeitfähigkeit (t_{soll}) gewährleisten zu können, soll der Algorithmus anhand geeigneter Parameter entsprechend eingestellt werden. Voraussetzung ist ein Modell, das den Zusammenhang zwischen diesen Parametern und dem Zeitverhalten des Algorithmus unter Berücksichtigung dessen Fehlerverhaltens beschreibt.

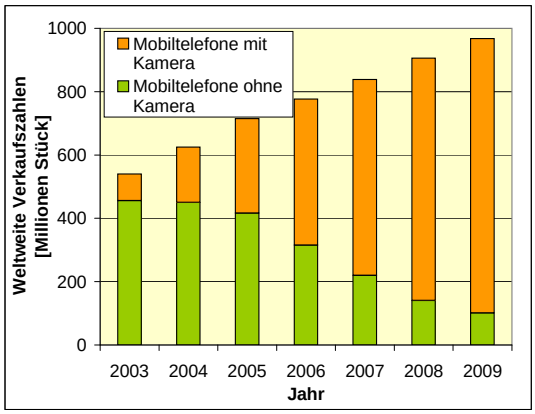
Systemportierbarkeit gewährleisten. Voraussetzungen für ein Skalieren der zugrundeliegenden Algorithmen und damit wissenschaftliche Beiträge dieser Arbeit sind (vgl. Abbildung 1.1):

1. Parametrisierung
   Für einen Algorithmus müssen geeignete Parameter eingeführt werden, anhand derer ein möglichst nahtloses Skalieren desselben hinsichtlich Qualität und Aufwand möglich ist.

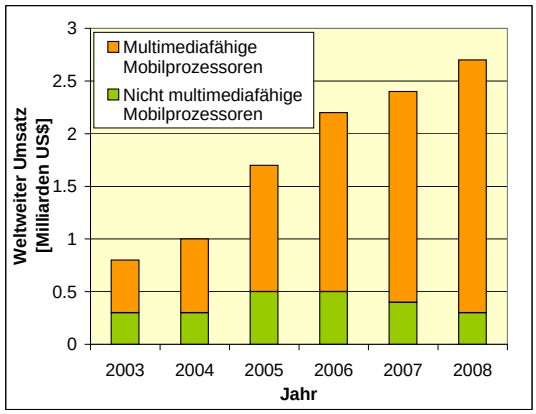
2. Modellierung
   Für diese Parameter muß ein Rahmensystem gefunden werden, das den Zusammenhang zwischen Qualität, Aufwand und den Skalierungsparametern beschreibt und so die Steuerung des Systems erlaubt.

In Kapitel 2 wird zunächst eine allgemeinen Einordnung des gegebenen Problems in die verschiedenen Klassen von Verfahren zur Umgebungserkennung gegeben. Abschnitt 3.1 befaßt sich nach Betrachtung der zugrundeliegenden Geometrie mit dem technischen Stand der Algorithmen zur Tiefenschätzung im Speziellen und deren jeweiliger Eignung zur mobilen Anwendung. Für das Verfahren, das sich hierbei als das geeignetste erweist, werden in Kapitel 3.6 entsprechende Skalierungsparameter eingeführt. In Kapitel 3.7 wird das zugrundeliegende Modell aufgestellt und über den Vergleich mit Testergebnissen verifiziert. Die endgültige

Herleitung der entsprechenden Skalierungsvorschrift für die Disparitätsschätzung findet dann in 3.7.7 statt. Die Übertragbarkeit der auf die Disparitätsschätzung angewandten Systematik auf andere Verfahren der AR wird dann in Abschnitt 4 anhand eines Trackingalgorithmus veranschaulicht. Abschnitt 5 zeigt schließlich die ressourcenabhängige Anpassung eines Algorithmus anhand der zugehörigen Skalierungsvorschrift.



(a)



(b)

(c)

Abbildung 1.2.: (a) Studie zu den weltweiten Verkaufszahlen von Mobiltelefonen mit Kamera (orange) im Vergleich zu Mobiltelefonen ohne Kamera (grün) [2]. (b) Studie der weltweiten Verkaufsumsätze multimediafähiger Prozessoren in Mobiltelefonen (orange) im Vergleich zu nicht multimediafähigen Prozessoren (grün) [3]. (c) Mobilisierte Hardware des „VAMPIRE“-Projekts der Universität Bielefeld.

2. Einführung

2.1. Dreidimensionale Umgebungserkennung

Nach [4] können Methoden zur Distanzmessung in drei Hauptgruppen aufgeteilt werden:

1. „Continuous Wave“ (CW) und „Time of Flight“ (TOF) -Methoden messen die Zeit, die die einem optischen Signal aufmodulierte Hüllkurve benötigt, um die zu messende Distanz zurückzulegen.

2. Interferometrie-Methoden messen ebenfalls die „Flug-“ bzw. Übertragungszeit eines Signals, wobei hier die Kohärenz der vom Objekt reflektierten Wellenfront mit einer Referenzwellenfront genutzt wird.

3. Triangulation bestimmt die Distanz eines Punktes anhand der Seitenwinkel des Dreiecks aus einer bekannten Basislinie und den Strahlen zwischen deren Enden und dem Punkt (vgl. Abbildung 2.2 (a)).

Der mit den ersten beiden Methoden verbundene hohe Hardwareaufwand ist nur schwer aufzubringen und disqualifiziert diese für eine mobile Anwendung. Wie Abschnitt 1 bereits zeigte, hat sich die Symbiose aus Mobilgerät (insbesondere -telefon) und Kamera bereits durchgesetzt und befindet sich weiterhin auf dem Vormarsch. Da also eine entsprechende Sensorik als vorhanden angenommen werden kann, bieten sich optische und damit triangulationsbasierte Verfahren zur Umgebungserkennung auf solchen Geräten geradezu an. Ein Überblick über diese Verfahren wird in Abbildung 2.1 gegeben.

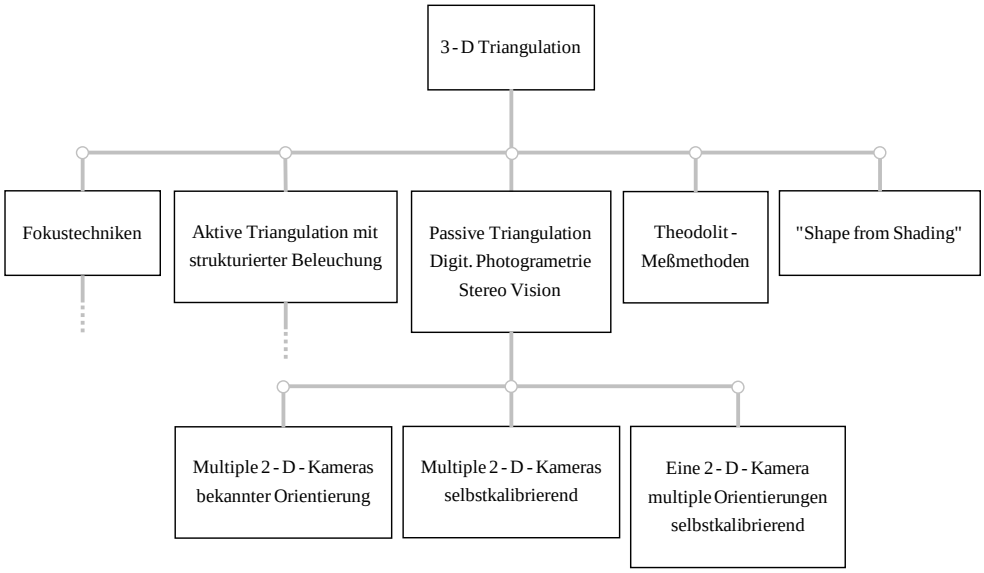


Abbildung 2.1.: Gliederung der Triangulationsverfahren nach [4].

2.2. Triangulation

2.2.1. Fokusbasierte Techniken

Bei „shape from blur“ wird die Abbildungsschärfe von Punkten erkannt und daraus auf deren Distanz geschlossen (Abbildung 2.2 (d)). Die in Mobilgeräten eingesetzten Kameras besitzen in der Regel einen festeingestellten Fokus, sodaß dieses Verfahren hier wenig Sinn macht. Darüberhinaus reagiert das System sensibel gegenüber anderen Unschärfequellen, wie etwa Bewegung, die sich gerade bei mobiler Anwendung nicht ausschließen läßt.

2.2.2. Aktive Triangulation mit strukturiertem Licht

Bei dieser Technik wird mithilfe eines Projektors an einem Ende einer Basislinie ein bekanntes Lichtmuster auf die Szene projeziert. Von der Verzerrung, die dieses Muster vom anderen Ende der Basislinie aus betrachtet beim Auftreffen auf die Szene erfährt, wird auf deren Struktur geschlossen (Abbildung 2.2 (b)). In abgeschlossenen Laborumgebungen ist dieses Verfahren sehr praktikabel, ein Einsatz im alltäglichen Bereich wird jedoch durch die optische Beeinflussung der

Szene selbst unmöglich. Ebenfalls die hohen Hardwareanforderungen sind für Mobilgeräte kaum realisierbar.

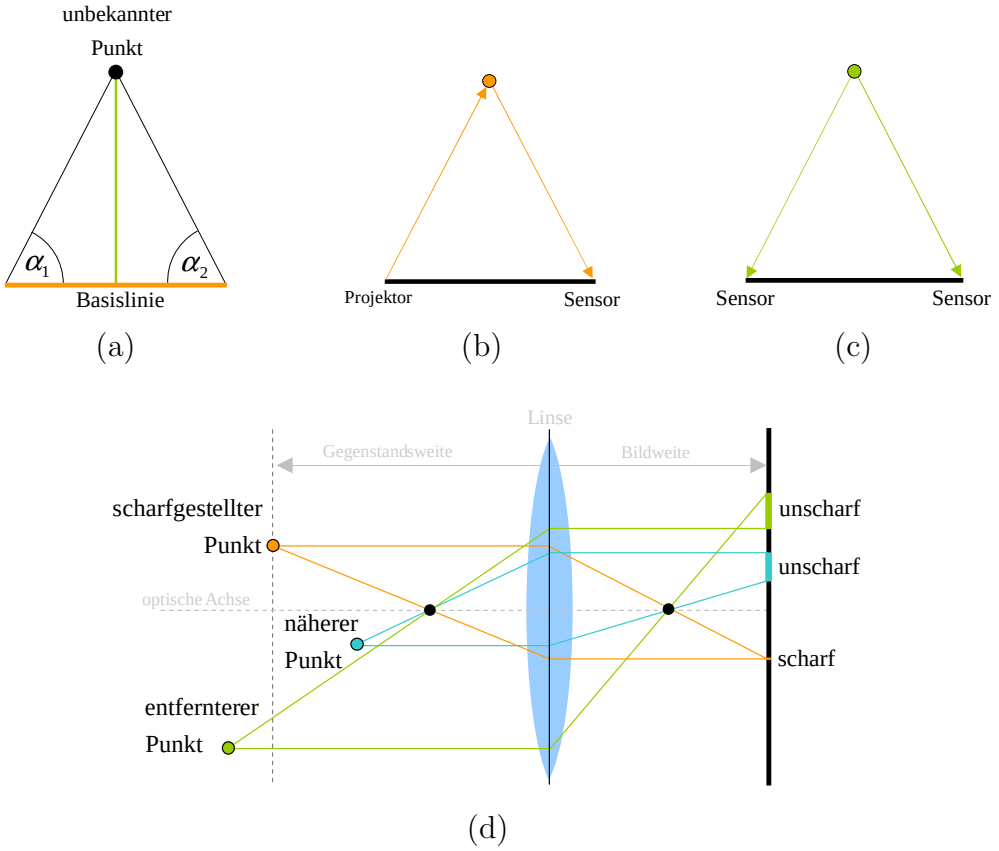


Abbildung 2.2.: (a) Grundprinzip der Triangulation: Durch Messen der Winkel α_1 und α_2 kann anhand einer bekannten Basislinie (orange) auf die Distanz (grün) des Meßpunktes geschlossen werden. (b) Aktive und (c) passive Triangulation. (d) Bei entsprechender Einstellung der Bildweite werden nur Punkte im Abstand der Gegenstandsweite scharf abgebildet. Durch Ermittlung der Schärfe eines Punktes kann damit auf dessen Distanz geschlossen werden („shape from blur“).

2.2.3. Theodolitbasierte Techniken

Diese im Vermessungswesen häufig eingesetzte Technik setzt die manuelle Bedienung besonderer Meßkameras (Theodoliten) voraus. Mit einem Fadenkreuz wird dann der zu bestimmende Punkt angepeilt und die Winkel α_1 und α_2 am Gerät direkt abgelesen. Der damit verbundene Aufwand und die hohen Anforderungen an den Benutzer schließen eine Anwendung aus, denn die Distanzschätzung soll nicht zum Selbstzweck, sondern als Grundlage anderer Anwendungen eingesetzt werden.

2.2.4. Shape from Shading

„Ein einfaches Modell der Bildformation ist das Lambert'sche Modell, bei dem der Grauwert eines Pixels im Bild von der Beleuchtungsrichtung und der Oberflächennormalen abhängt. Bei „structure from shading“ ist das Ziel, anhand eines Grauwertbildes die Beleuchtungsrichtung und Oberflächenstruktur für jedes Pixel im Bild zu bestimmen. Echte Bilder entsprechen jedoch nicht immer dem Lambert'schen Modell. Selbst unter der Annahme Lambert'scher Reflexion, einer bekannten Beleuchtungsrichtung und der tatsächlichen Beschreibbarkeit der Helligkeit als Funktion der Oberflächenstruktur, ist dieses Problem nicht einfach. Denn wenn die Oberflächenstruktur durch ihre Normale beschrieben wird, erhält man ein lineares Gleichungssystem mit drei Unbekannten und wenn sie durch ihren Gradienten beschrieben wird, erhält man eine nichtlineare Gleichung mit zwei Unbekannten. Entsprechend ist es schwer, eine eindeutige Lösung für das „Structure-from-shading“-Problem zu finden und zusätzliche Randbedingungen müssen eingeführt werden.“ [5]
Da diese Annahmen mit den hier angestrebten, realen Sequenzen nicht vereinbar und die Rechenzeiten von einer Echtzeitanwendung weit entfernt sind, ist diese Methode für die hier betrachteten Anwendungen nicht geeignet.

2.2.5. Passive Triangulation

Die Gruppe der passiven Triangulation umfaßt alle Methoden der digitalen Photogrammetrie (vgl. Abbildung 2.2 (c)), die entsprechend auf mechanischer Seite weiterhin nach Anzahl der verwendeten Kameras unterteilt werden kann. Un-

abhängig von Abbildung 2.2 können die Algorithmen der digitalen Photogrammetrie nach Art der Auswahl der zur Triangulation verwendeten Punkte eingeteilt werden ([4], S. 190):

1. Künstliche Zielpunkte
   Dabei werden vom Meßsystem gut erkennbare künstliche Markierungen („Marker“) in die Szene eingefügt. Dieses Vorgehen ist für die meisten Anwendungen, insbesondere im mobilen Fall, nicht durchführbar.

2. Merkmalsbasierte Zielpunkte
   Hierbei werden automatisch besondere Punkte in der Szene erkannt, die aufgrund der Eigenschaften ihrer lokalen Umgebung von verschiedenen Blickwinkeln gut wiedererkennbar sind. Für eine limitierte Anzahl von Punkten sind diese Verfahren sehr gut geeignet, vor allem, wenn es darum geht, die Orientierung der Kamera im Raum bzw. deren Bewegung zu erkennen. Der damit einhergehende hohe Rechenaufwand macht eine Echtzeitanwendung mit 25 Bildern pro Sekunde auf heutigen Mobilgeräten allerdings schwierig und diese Verfahren nur bedingt einsetzbar.

3. Texturbasiertes Zuordnen
   Um nicht nur einzelne Punkte, sondern ganze Oberflächen zu bestimmen und Gruppen aus vielen Tausend Punkten zu erzeugen, sind höhere örtliche Abtastraten nötig. Diese können durch die Verwendung von Texturen erreicht werden. Um Punkte in verschiedenen Ansichten zuordnen zu können, sollten diese Texturen dicht, hochfrequent und unperiodisch sein ([4], S.192). Da die mit diesen Verfahren verbundenen Anforderungen sowohl an die Szenen, als auch an die benötigte Rechenleistung dem hier betrachteten Fall gut vereinbar sind, sollen diese als Grundlage der folgenden Betrachtungen zur Realisierung der in Kapitel 1 vorgestellten Ziele dienen.

3. Disparitätsschätzung

3.1. „Motion“ oder „Stereo“?

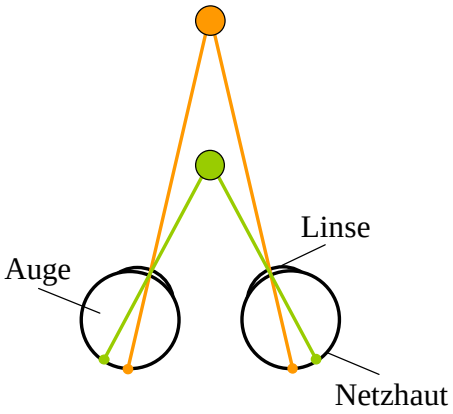


Abbildung 3.1.: Menschliches Sehsystem: Nahe Objekte (grün) verursachen immer eine größere Verschiebung auf der Netzhaut als entfernte (orange).

Wie Abbildung 2.1 zu entnehmen, können die passiven, photogrammetrischen Triangulationsverfahren in drei Gruppen unterteilt werden:

1. Multiple 2-D-Kameras bekannter Orientierung
   Das bezieht sich insbesondere auf Stereosysteme, die dem gleichen Prinzip unterliegen, wie das menschliche räumliche Sehen (Abbildung 3.1), aber auch mehr als zwei Kameras umfassen können.

2. Multiple 2-D-Kameras, selbstkalibrierend
   Befinden sich mehrere Kameras unbekannter Orientierung beliebig im Raum verteilt, gilt es nicht nur die Winkel α_1 und α_2 zu schätzen (vgl. Abbildung 2.2 (a)), sondern auch die Basislinie(n).

3. Eine 2-D-Kamera, multiple Orientierungen, selbstkalibrierend
   Systematisch ist diese Gruppe identisch mit der vorangegangenen. Der Unterschied besteht darin, daß nicht gleichzeitig aufgenommene Bilder verschiedener Kameras verwendet werden, sondern zeitlich aufeinanderfolgende Bilder einer einzelnen Kamera („shape from motion“, vgl. auch Optischer Fluß [6]). Daraus entsteht die Bedingung einer unbewegten Szene ([7] S. 110, [4] S. 190).

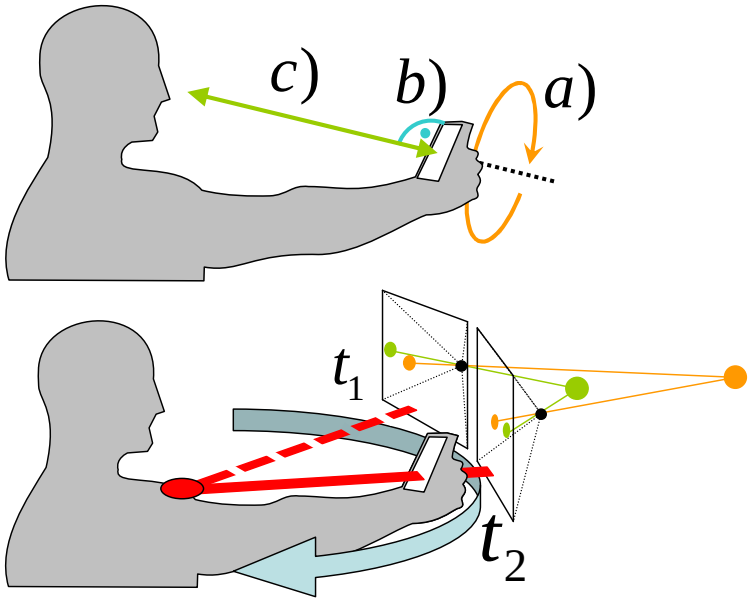
Da Mobilgeräte in der Regel über kein Stereosystem, sondern nur eine einzelne Kamera verfügen, scheinen die ersten beiden Verfahren zunächst auszuscheiden. Die Zielanwendung der betrachteten Verfahren besteht im Einblenden von Information für den Benutzer in dessen Ansicht einer Szene, sodaß entsprechend davon auszugehen ist, daß er stets das Display betrachtet. Er ist daher kontinuierlich bemüht, das Gerät

1. nicht um die optische Achse zu rotieren (Abbildung 3.2 (a)),
2. die Bildebene senkrecht zur Blickrichtung zu halten (Abbildung 3.2 (b)),
3. den Abstand zu seinen Augen konstant zu lassen (Abbildung 3.2 (c)).

Mit diesen Annahmen ergibt sich die in Abbildung 3.2 (d) beschriebene Geometrie eines Kameraschwenks, insbesondere bei inkrementeller Betrachtung, also bei einer kleinen Zeitdifferenz $t_2 - t_1$ zwischen den beiden Aufnahmen. Die optischen Achsen sind also stets als Normale einer Kugelschale um das Schultergelenk des Benutzers, mit Radius der Armlänge zu sehen.

Daraus folgen für die beiden Kamerapositionen in dieser Anordnung mit dem horizontalen und dem vertikalen Rotationswinkel zwei Freiheitsgrade. Selbstkalibrierende Methoden multipler Kameraorientierungen lassen alle sechs Freiheitsgrade eines starren Körpers im Raum zu. Stereosysteme hingegen besitzen aufgrund ihrer mechanisch zueinander starr installierten Kameras keinerlei Freiheitsgrade zwischen diesen.

Damit und mit der offensichtlichen Ähnlichkeit zur Stereogeometrie in Abbildung 3.1 wird deutlich, daß für diese Anordnung Algorithmen zur Stereokorrespondenz am nächsten liegen. Das setzt zusätzlich eine Schätzung der beiden Freiheitsgrade, und damit der Basislinie, anhand der Bilder voraus. Dieses zentrale Problem des „shape from motion“ wird dabei zu einer zweidimensionalen Bewegungsschätzung



(d)

Abbildung 3.2.: (a) Konstanter Winkel, (b) zur Blickrichtung orthogonales Display und (c) konstanter Abstand zu den Augen bei der typischen Bewegung eines Mobilgerätes. (d) Typische Bewegung eines Mobilgerätes bei Betrachten des Displays durch den Benutzer.

degradiert und, wie auch die detaillierten geometrischen Zusammenhänge, in Abschnitt 3.6 genauer betrachtet.

3.2. Stereokorrespondenzverfahren

3.2.1. Prinzip

Entsprechend der in Abschnitt 2 eingeführten Triangulation gilt es, die Winkel α_1 und α_2 zwischen Basislinie und den sich im zu bestimmenden Punkt schneidenden Lichtstrahlen zu ermitteln. Wie Abbildung 3.3 verdeutlicht, geht dieses Problem in die Identifikation der Abbildungen desselben Punktes in beiden Bildern über (Stereokorrespondenzproblem). Bei der klassischen Stereogeometrie sind dabei stets auch einheitliche Scanlinien in beiden Bildern gegeben, sodaß das Korrespondenzproblem nur eindimensional in Basislinienrichtung gelöst werden muß.

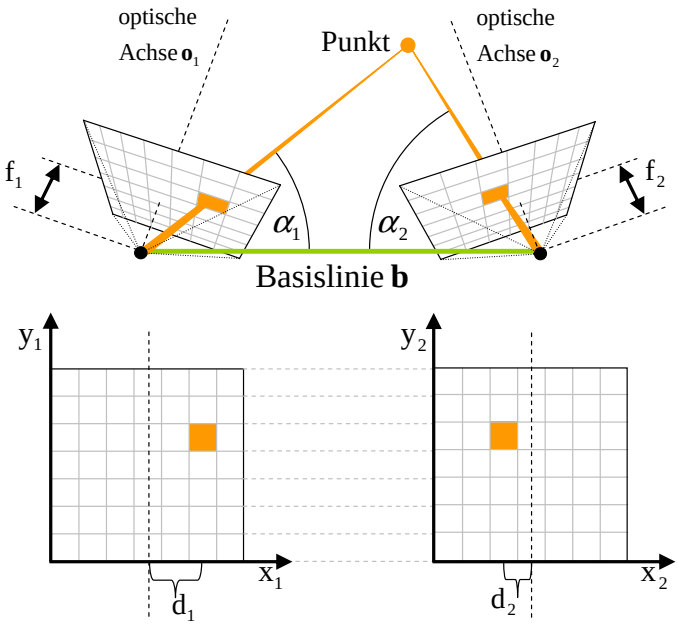


Abbildung 3.3.: Stereoskopische Triangulation.

Die zugrundeliegende Transformaiton beider Bilder in ein gemeinsames Koordinatensystem wird „image rectification“ genannt ([8] S. 325ff, [9] S. 302ff).
Die gesuchte Summe $d = d_1 + d_2$ in Abbildung 3.3, also die Verschiebung desselben Inhaltes zwischen beiden Bildern, heißt „Disparität“ ([8] S. 326, [10]). [11], [12] und [10] geben einen guten Überblick über die Vielfalt von Stereoalgorithmen und [13] bietet eine stets aktuelle Vergleichsplattform aktueller Algorithmen.

3.2.2. Block Matching

Aus der Videokodierung stammend, spiegelt das Block Matching die direkte Umsetzung der texturbasierten Korrespondenzidee wieder und stellt ein sowohl schnelles als auch zuverlässiges Werkzeug zum Wiederfinden von Bildinhalt dar [14, 15, 16, 17, 18]. Damit bietet es sich zur Anwendung auf das Stereokorrespondenzproblem geradezu an, weshalb es in diesem Zusammenhang bereits häufig verwendet wurde [19, 20, 21, 22, 23].
Als Ähnlichkeitsvergleich zweier quadratischer Bildfragmente (Blöcke) mit der ungeraden Seitenlänge k an den Bildpositionen (x_1, y_1) und (x_2, y_2) in den Bildern $f_1(x, y)$ und $f_2(x, y)$ wird mit $k' = \frac{k-1}{2}$ eines der folgenden Fehlermaße berechnet [10]:

- Normalized Cross-Correlation (NCC)

$$NCC_{k\times k}\left(f_1(x_1,y_1),f_2(x_2,y_2)\right) = \frac{\sum_{j=-k'}^{k'}\sum_{i=-k'}^{k'}\left(f_1(x_1+i,y_1+j)-\bar{f}_1(x_1+i,y_1+j)\right)\left(f_2(x_2+i,y_2+j)-\bar{f}_2\right)}{\sqrt{\sum_{j=-k'}^{k'}\sum_{i=-k'}^{k'}\left(f_1(x_1+i,y_1+j)-\bar{f}_1(x_1+i,y_1+j)\right)^2\left(f_2(x_2+i+i,y_2)-\bar{f}_2\right)^2}} \tag{3.1}$$

- Mittlerer Quadratischer Fehler (Mean Squared Error)

$$MSE_{k\times k}\left(f_1(x_1,y_1),f_2(x_2,y_2)\right) = \frac{1}{k^2}\sum_{j=-k'}^{k'}\sum_{i=-k'}^{k'}\left(f_1(x_1+i,y_1+j)-f_2(x_2+i,y_2+j)\right)^2. \tag{3.2}$$

- Quadratischer Fehler (Sum of Squared Differences)

$$SSD_{k\times k}\left(f_1(x_1,y_1),f_2(x_2,y_2)\right) = \sum_{j=-k'}^{k'}\sum_{i=-k'}^{k'}\left(f_1(x_1+i,y_1+j)-f_2(x_2+i,y_2+j)\right)^2. \tag{3.3}$$

- Normalisierte SSD

$$NSSD_{k\times k}\left(f_1(x_1,y_1),f_2(x_2,y_2)\right) = \sum_{j=-k'}^{k'}\sum_{i=-k'}^{k'}\left(\frac{f_1(x_1+i,y_1+j)-\bar{f}_1(x_1+i,y_1+j)}{\sqrt{\sum_{j=-k'}^{k'}\sum_{i=-k'}^{k'}\left(f_1(x_1+i,y_1+j)-\bar{f}_1(x_1+i,y_1+j)\right)^2}} - \frac{f_2(x_2+i,y_2+j)-\bar{f}_2(x_2+i,y_2+j)}{\sqrt{\sum_{j=-k'}^{k'}\sum_{i=-k'}^{k'}\left(f_2(x_2+i,y_2+j)-\bar{f}_2(x_2+i,y_2+j)\right)^2}}\right)^2 \tag{3.4}$$

- Absoluter Fehler (Sum of Absolute Difference)

$$SAD_{k\times k}\left(f_1(x_1,y_1), f_2(x_2,y_2)\right) = \sum_{j=-k'}^{k'}\sum_{i=-k'}^{k'} \left|f_1(x_1+i, y_1+j) - f_2(x_2+i, y_2+j)\right|. \quad (3.5)$$

- Rang

$$RANK\big(f_1(x_1,y_1), f_2(x_2,y_2)\big) = f'_{k\times k}\big(f_1(x_1,y_1)\big) - f'_{k\times k}\big(f_2(x_2,y_2)\big),$$
$$\text{mit } f'_{k\times k}\big(f(x,y)\big) = \sum_{j=-k'}^{k'}\sum_{i=-k'}^{k'} \xi\big(f(x,y), f(x+i,y+j)\big)$$
$$\text{und } \xi\big(f(x,y), f(x',y')\big) = \begin{cases} 1 & \text{, falls } f(x',y') < f(x,y); \\ 0 & \text{, sonst.} \end{cases} \quad (3.6)$$

- Zensus

$$CENSUS(f_1(x_1,y_1), f_2(x_2,y_2)) = \sum_{j=-k'}^{k'}\sum_{i=-k'}^{k'} \xi\big(f_1(x_1,y_1), f_1(x_1{+}i, y_1{+}j)\big) \text{ XNOR } \xi\big(f_2(x_2,y_2), f_2(x_2{+}i, y_2{+}j)\big),$$
$$\text{mit } a \text{ XNOR } b = \begin{cases} 1 & \text{, falls } a = b; \\ 0 & \text{, sonst.} \end{cases} \quad (3.7)$$

Rang und Zensus betrachten keine wirkliche Fehlerfunktion, sondern vielmehr die numerische Reihenfolge der Pixel in einem Block [24], was sie sehr unempfindlich gegenüber Rauschen macht. Der Zensus ist dabei sehr aufwendig zu berechnen und der Rang eher als Filteroperation vor Ausführung der SAD zu sehen [25]. Das statistische Standardverfahren für Ähnlichkeitsvergleiche ist die NCC. Für geschwindigkeitsorientierte Echtzeitanwendungen ist deren Berechnung jedoch zu komplex und wird meist durch die SAD ersetzt [10, 26, 27]. Da dieses Vergleichskriterium lediglich mit Additionen und Subtraktionen auskommt, ist dessen Verwendung für mobile Prozessoren besonders vorteilhaft, da deren Gleitkommaeinheit meist unverhältnismäßig langsam ist.

Das Referenzbild $f_r(x, y)$ ist dabei das Bild, für dessen Inhalt die Disparitäten berechnet werden sollen, das Suchbild $f_s(x, y)$ der entsprechende Stereopartner, dessen Bestimmung für den Fall einer Videosequenz in Abschnitt 3.6 behandelt wird. Um dessen Verschiebung d zu bestimmen, wird ein Block an der Stelle $\mathbf{x}_r = (x_r, y_r)$ im Referenzbild mit allen Blöcken innerhalb eines Suchbereichs $\mathbf{s}$ im Suchbild verglichen:

$$d(x_r, y_r) = |\mathbf{d}(x_r, y_r)| = \left| \arg\min_{\mathbf{s}} SAD_{k\times k}(f_r(x_r, y_r), f_s(x_r + s_x, y_r + s_y)) \right| . \quad (3.8)$$

Der Suchbereich ist dabei normalerweise quadratisch mit der Größe $w \times w$, sodaß gilt:

$$s_x, s_y \in \left[-\frac{w}{2}, \frac{w}{2} \right] . \quad (3.9)$$

Zu Kodierungszwecken wird das Referenzbild in disjunkte, aneinandergrenzende Blöcke unterteilt, sodaß sich für die Blockpositionen ergibt:

$$x_r, y_r \in \left[\frac{k}{2}, \frac{k}{2} + k, \frac{k}{2} + 2k, ... \right] . \quad (3.10)$$

Für die Stereokorrespondenz mit entsprechender Scanlinieneinheitlichkeit kann der Suchbereich auf einen Vektor in Basislinien- und damit auch x-Richtung reduziert werden:

$$d(x_r, y_r) = |\mathbf{d}(x_r, y_r)| = \left| \arg\min_{\mathbf{s}} SAD_{k\times k}(f_r(x_r, y_r), f_s(x_r + s_x, y_r)) \right| . \quad (3.11)$$

Außerdem wird für jedes Pixel im vollständigen Referenzbild ein Block als dessen Umgebung gesehen:

$$x_r, y_r \in \left[\frac{k}{2}, \frac{k}{2} + 1, \frac{k}{2} + 2, ... \right] . \quad (3.12)$$

3.2.2.1. Hierarchisierung

Häufig wird das Blockmatching hierarchisiert [28, 29]: Beginnend mit einem sehr großen k (oft bildfüllend), wird in einem Schritt (Hierarchieebene) i dabei ein herkömmliches Blockmatching durchgeführt. In der darauffolgenden Hierarchieebene $i+1$ wird jeder Block viergeteilt ($k_{i+1} = \frac{k_i}{2}$), ebenso wie die entsprechende Suchbereichsgröße ($w_{i+1} = \frac{w_i}{2}$). Außerdem wird das Zentrum der Suche für jeden

so entstandenen Subblock um den Ergebnisvektor $\mathbf{d}(x_r, y_r)_i$ des jeweilig übergeordneten Blocks verschoben:

$$\mathbf{d}_{i+1}(\mathbf{x}_r) = \arg\min_{\mathbf{s}} SAD_{\frac{k_i}{2}\times\frac{k_i}{2}}(f_r(\mathbf{x}_r), f_s(\mathbf{x}_r + \mathbf{d}_i(\mathbf{x}_r))). \tag{3.13}$$

Das Gesamtergebnis ergibt sich somit als Summe über die Ergebnisvektoren der einzelnen Hierarchieebenen.

3.2.2.2. Unterabtastung

Ein weit verbreitetes Mittel zur Beschleunigung blockbasierter Verfahren ist die Unterabtastung. Dabei wird das Bewertungskriterium nicht für alle Pixel im Block berechnet (vgl. Gleichung 3.5), sondern nur für eine eingeschränkte Menge, die in der Regel gleichmäßig über den Block verteilt und x- bzw. y-periodisch ist. Gleichung 3.5 lag noch zugrunde:

$$i, j \in \{0, \pm 1, \pm 2, \pm 3, \ldots\}. \tag{3.14}$$

Mit Unterabtastung mit der Periode s ändert sich diese Menge zu:

$$i, j \in \{0, \pm s, \pm 2s, \pm 3s, \ldots\}. \tag{3.15}$$

3.2.2.3. Suchstrategien

Um den hohen Aufwand einer vollständigen Abtastung des Suchbereichs zu minimieren, ohne das Ergebnis von vornherein einzuschränken (z.B. bei Unterabtastung), kommen häufig besondere Abtaststrategien zum Einsatz [14]. Dabei ist die logarithmische Suche die verbreitetste (vgl. Abbildung 3.4): Zunächst wird der Suchbereich mit der Abtastperiode l_0 unterabgetastet (vgl. Gleichung 3.9):

$$s_{x,0}, s_{y,0} \in \left\{-\frac{w}{2}, \ldots, -l_0, 0, l_0, \ldots, \frac{w}{2}\right\}. \tag{3.16}$$

Für alle nachfolgenden Suchschritte wird der Suchbereich dann modifiziert entsprechend:

$$\begin{aligned} s_{x,i+1} &\in \left\{d_{x,i} - l_i \;,\; d_{x,i} - \frac{l_i}{2} \;,\; d_{x,i} \;,\; d_{x,i} + \frac{l_i}{2} \;,\; d_{x,i} + l_i\right\} \\ s_{y,i+1} &\in \left\{d_{y,i} - l_i \;,\; d_{y,i} - \frac{l_i}{2} \;,\; d_{y,i} \;,\; d_{y,i} + \frac{l_i}{2} \;,\; d_{y,i} + l_i\right\}. \end{aligned} \tag{3.17}$$

Dieses Vorgehen wird iterativ wiederholt, bis die Abtastperiode „1“ entspricht (vgl. Abbildung 3.4).

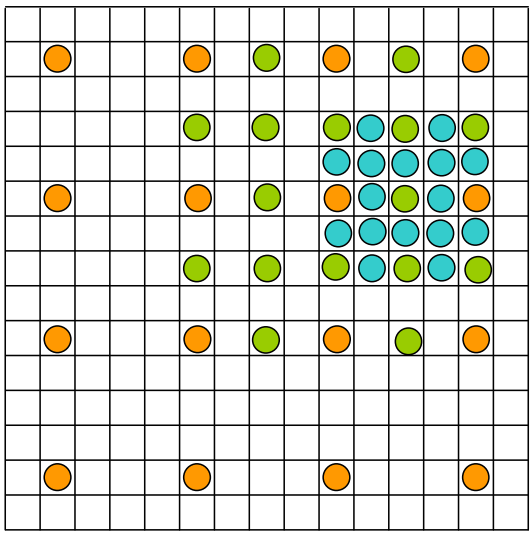


Abbildung 3.4.: Blockpositionen der ersten (orange), zweiten (grün) und dritten (violett) Stufe einer logarithmischen Suche („3 step search“). Ein markiertes Pixel ist dabei als Mittelpunkt eines Blocks im Suchbild zu betrachten.

3.2.3. Disparitätsraum

Die in Abschnitt 3.2.2 vorgestellte Vorgehensweise des Block Matchings ist extrem ineffizient, wenn für jedes Pixel ein Block verwendet wird (vgl. Gleichung 3.12), da sich benachbarte Blöcke dann überlappen und so viele redundante SAD-Berechnungen durchgeführt werden [10]. In [26] wurde das auch für die Echtzeit-Implementierung in [30] verwendete Konzept des Disparitätsraumes detailliert beschrieben, das als Standardkonzept der Stereokorrespondenz zu betrachten ist. Voraussetzung zu dessen Verwendung ist ein vektorförmiger Suchbereich, wie er bei der Disparitätsschätzung zweckmäßig ist. Dabei wird nach [12] die Berechnung in mehrere Schritte aufgeteilt:

1. Kostenberechnung (matching cost computation)
2. Kostenaggregation(cost (support) aggregation)
3. Disparitätsberechnung/Optimierung (disparity computation/optimization)
4. Disparitätsverfeinerung (disparity refinement)

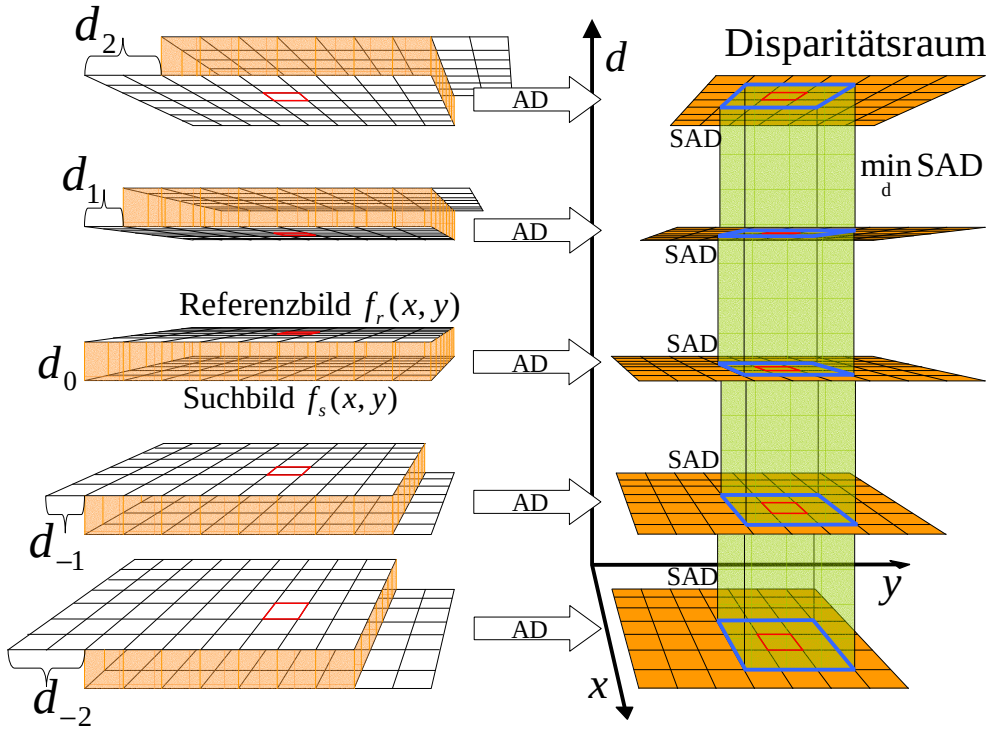


Abbildung 3.5.: Für alle Pixel der gegeneinander verschobenen Bilder werden die Absoluten Differenzen (AD) berechnet (orange). Die Ergebnisse spannen den Disparitätsraum auf, wobei jede Ebene einer Verschiebung entspricht. Über Aggregation wird in allen Ebenen die Summe der Absoluten Differenzen (SAD) berechnet (blau) und an der Stelle der jeweiligen Referenzbildkoordinaten über alle Ebenen hinweg minimiert (grün).

3.2.3.1. Kostenberechnung

Im ersten Schritt wird dabei der Disparitätsraum $D(x, y, d)$ aufgebaut, indem die beiden vollständigen Bilder gegeneinander verschoben und die Beträge der Pixeldifferenzen (Absolute Differences) berechnet werden:

$$AD(x, y, d) = AD(\mathbf{x}, d) = |f_r(\mathbf{x}) - f_s(\mathbf{x} + d\mathbf{s})|. \tag{3.18}$$

Voraussetzung dafür ist die Kenntnis der Basislinie, also der Positionen der beiden Kameras, wie sie etwa bei Stereosystemen bekannt und rein horizontal oder vertikal ist. Bei der Verwendung der Einzelbilder einer Videosequenz muß diese zunächst geschätzt werden (vgl. Abschnitt 3.6). Die Verschiebung d der beiden

Bilder erfolgt dann durch den Einheitsvektor $\mathbf{s}$ entlang des Basislinienvektors $\mathbf{b}$:

$$\mathbf{s} = \frac{\mathbf{b}}{|\mathbf{b}|}. \tag{3.19}$$

Jede d-Ebene im Disparitätsraum entspricht also einer bestimmten Verschiebung und enthält die entsprechenden Pixelfehler.

3.2.3.2. Kostenaggregation

Für einen quadratischen Block der ungeraden Seitenlänge k, der ein Pixel an der Stelle (x_r, y_r) in der Ebene d des Disparitätsraumes umgibt, kann dann im zweiten Schritt über Summation (Cost Aggregation) die SAD berechnet werden:

$$SAD^{agg}_{k\times k}(D(x_r, y_r, d)) = \sum_{i=-\frac{k-1}{2}}^{\frac{k-1}{2}} \sum_{j=-\frac{k-1}{2}}^{\frac{k-1}{2}} D(x_r + i, y_r + j, d). \tag{3.20}$$

Hier zeigt dieses Verfahren seinen enormen Geschwindigkeitsvorteil gegenüber dem Block Matching, der nicht nur in der minimierten Redundanz bei der Berechnung der Absoluten Differenzen, sondern auch in der hocheffizienten Implementierbarkeit des „Cost Aggregation"-Schrittes als „moving average filter" begründet ist [30, 26], was den benötigten Rechenaufwand von der Blockgröße unabhängig macht [12]:

Pro Zeile einer d-Ebene wird dabei zur Initialisierung über die ersten k Fehlerwerte die Summe gebildet und zwischengespeichert. Im nächsten Schritt wird der linke dieser Fehlerwerte von der Summe subtrahiert und der Fehlerwert rechts neben ihnen addiert und das Ergebnis wiederum zwischengespeichert (vgl. Abbildung 3.6 (a)). Dieses Vorgehen wird dann solange wiederholt, bis die ganze Zeile abgearbeitet ist. Aus den zwischengespeicherten Werten ergibt sich wiederum ein „halb" akkumuliertes Fehlerbild gleicher Höhe, jedoch mit der Breite $\frac{x}{k}$, wenn die ursprünglichen Dimensionen des Fehlerbildes $x \times y$ betrugen (vgl. Abbildung 3.6 (c)). Analog werden dann wiederum die Spalten dieses Zwischenbildes nach gleicher Vorschrift verarbeitet (vgl. Abbildung 3.6 (d)), sodaß sich schließlich ein SAD-Bild der Größe $\frac{x}{k} \times \frac{y}{k}$ ergibt (vgl. Abbildung 3.6 (e)). Die darauf folgende Minimierung findet dann in den so verarbeiteten d-Ebenen statt, mit der entstehenden Koordinatentransformation:

$$\mathbf{x}_r = (x_r, y_r) = (kx_{SAD}, ky_{SAD}) = k\mathbf{x}_{SAD}. \tag{3.21}$$

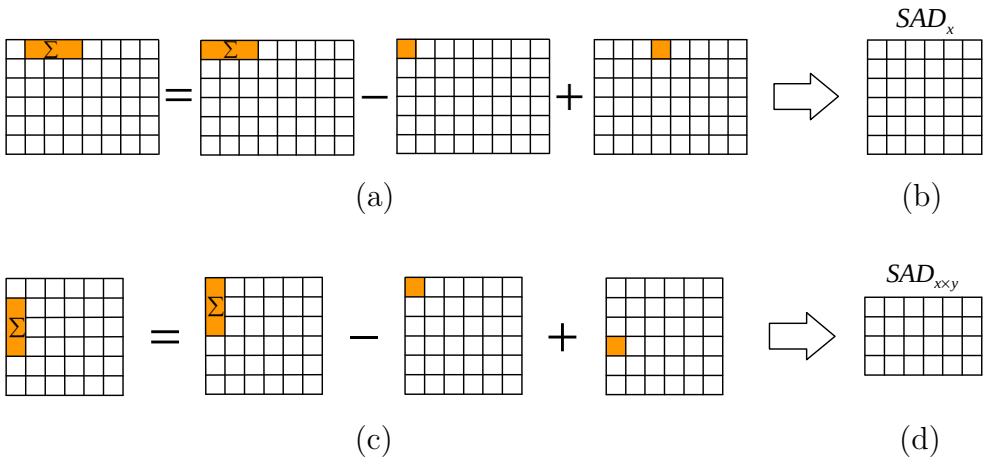


Abbildung 3.6.: Implementierung der Kostenaggregation als „ moving average“-Filter: (a,b) Zeilenweise Berechnung der neuen Summe aus der alten durch Subtraktion bzw. Addition der angrenzenden Pixel. (c,d) Analoges spaltenweises Vorgehen.

Die zweidimensionale Kostenaggregation kann dabei auch zu einer dreidimensionalen ausgedehnt werden [12]. Ein Block der Größe $k \times k$ wird dabei zu einem Quader der Größe $k \times k \times n$ und umfaßt auch die Absoluten Differenzen in darüber und darunterliegenden d-Ebenen:

$$SAD^{agg}_{k\times k\times n}(D(x_r, y_r, d)) = \sum_{h=-\frac{n}{2}}^{\frac{n}{2}} \sum_{i=-\frac{k}{2}}^{\frac{k}{2}} \sum_{j=-\frac{k}{2}}^{\frac{k}{2}} D(x_r + i, y_r + j, d + h). \qquad (3.22)$$

Andere Ansätze verwenden während der Fehlerminimierung mehrere Blöcke, um die Zugehörigkeit eines Pixels zu Vorder- oder Hintergrund zu präzisieren [25, 31] (vgl. dazu auch Abschnitt 3.2.3.5). Dasselbe Ziel verfolgen Verfahren, die ihre Blockgröße rekursiv an den Bildinhalt anpassen [32, 33]. Ein ausführlicher Vergleich verschiedener Algorithmen zur Kostenaggregation kann in [33] gefunden werden.
Algorithmen zur darauffolgenden Fehlerminimierung können nach [10] in lokale und globale Verfahren zur Stereokorrespondenz aufgeteilt werden.

3.2.3.3. Lokale Fehlerminimierung

Die lokale Fehlerminimierung stellt das einfachste und damit schnellste Verfahren im Disparitätsraum dar [33]. Der Disparitätsschätzwert $d(x_r, y_r)$ wird dabei über Minimierung des Fehlers in allen Tiefenebenen and der Stelle (x_r, y_r) berechnet:

$$d(x_r, y_r) = \arg\min_{d'} SAD^{agg}_{k\times k}(D(x_r, y_r, d')). \tag{3.23}$$

Da es sich um eine andere Implementierung desselben Prinzips handelt, sind die Ergebnisse der Fehlerminimierung im Disparitätsraum bei zweidimensionaler Kostenaggregation qualitativ identisch mit dem des Blockmatchings.

3.2.3.4. Globale Fehlerminimierung

Globale Verfahren betrachten nicht jede Bildkoordinate separat, sondern verwenden zusätzliche Zusammenhänge zwischen allen Pixeln des Bildes, wie sie aus der Stereogeometrie entstehen [11, 34]. Grundlegende Annahmen sind dabei die eindeutige Zuordenbarkeit eines Pixels zu genau einem Raumpunkt („uniqueness“) und die kontinuierliche Verteilung der Disparitäten im ganzen Bild bis auf die Objektränder („continuity“)[35, 36]. Dabei wird meist auf den Schritt der Kostenaggregation verzichtet und direkt im Disparitätsraum gearbeitet, wobei die Verfahren nach [12] wiederum in verschiedene Gruppen unterteilt werden können:

1. Cooperative
   Entsprechend der Stereotriangulation wird für einen Lichtstrahl (orange in Abbildung 3.7) zum Referenzbild der beim zu messenden Punkt (violett) kreuzende Lichtstrahl zum Suchbild gesucht. Das geschieht über den Vergleich des entsprechenden Pixels in der Scanlinie x_r mit den Pixeln der Scanlinie x_s, wie in den vorangegangenen Abschnitten beschrieben. Lokale Verfahren wiederholen diesen Vorgang für alle Pixel des Referenzbildes separat. Wird ein korrespondierendes Pixelpaar mit dem zugehörigen Punkt im Raum gefunden, folgt aus der Stereogeometrie jedoch, daß die Pixel im Suchbild, die dem selben Lichtstrahl entsprechen, nicht mehr anderweitig zugeordnet werden können (grün in Abbildung 3.7). „Cooperative“-Algorithmen versuchen, diesen sich global auswirkenden, lokalen Zusammenhang [12] zu erfüllen.

Aus den eingezeichneten Achsen in Abbildung 3.7 kann der Zusammenhang zwischen den Kreuzungspunkten und den Einträgen des Disparitätsraums $D(x_r, y, d)$ erkannt werden (Abbildung 3.7 (b)), wie er in Abschnitt 3.2.3 eingeführt wurde. Da bei der Ansicht in Abbildung 3.7 (a) die d-Achse verzerrt abgebildet wird (gleiche d-Werte sind grau dargestellt), findet oft die in diesem Zusammenhang intuitivere Darstellung $D(x_r, x_s, y)$ des Disparitätsraumes Verwendung [10] (Abbildung 3.7 (c)).

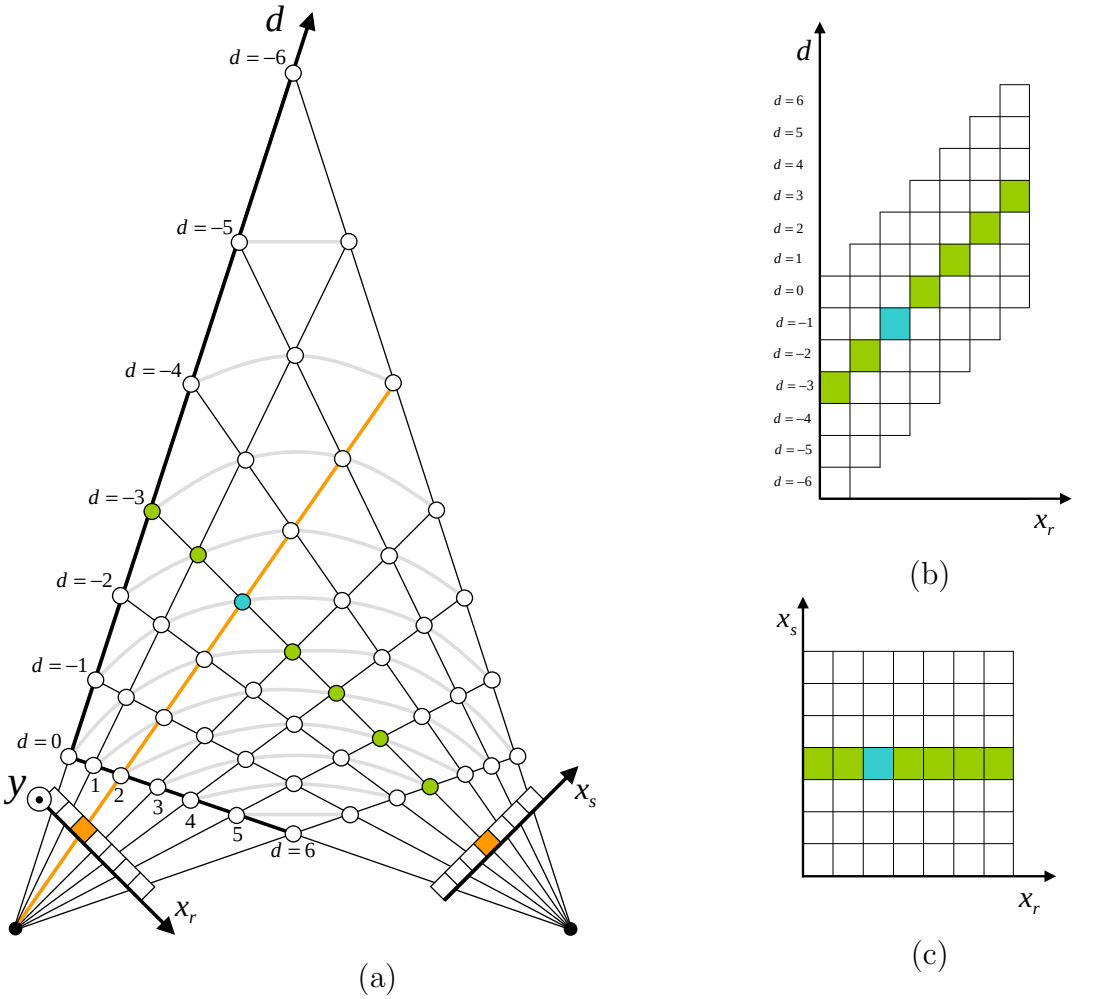
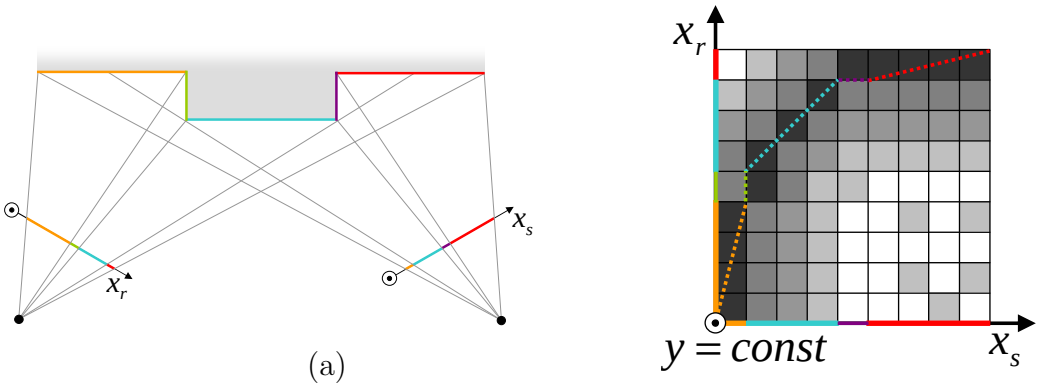


Abbildung 3.7.: Aus der Stereogeometrie entstehende Zusammenhänge zwischen einzelnen Pixeln, wie sie bei „Cooperative“-Algorithmen verwendet werden: Werte eines Lichtstrahls (grün), der einer bereits zugeordneten Korrespondenz (violett) angehört, können nicht mehr anderweitig zugeordnet werden.

2. Dynamic programming
   Die hierbei zugrundeliegende Annahme ist eine identische Reihenfolge entweder von Merkmalen (meist Kanten) oder der Pixel selbst in beiden Scanlinien („ordering constraint“)([8], S. 336), die sich aus der Stereogeometrie ergibt (vgl. Abbildung 3.8(a)). Als Fehlerkriterium wird dabei die Fehlersumme über den entsprechenden Weg durch den Disparitätsraum verwendet (Abbildung 3.8 (b)).



(a) (b)

Abbildung 3.8.: (a) Geometrischer Zusammenhang des „Ordering Constraint“: Die Reihenfolge „orange, blau, rot“ muß in den beiden Scanlinien x_r und x_s gleich sein. (b) Aus diesem globalen Zusammenhang wird die Fehlerminimierung beim „Dynamic Programming“ zur Suche nach einem optimalen Weg durch den Disparitätsraum.

3. Energieminimierung
   Hier wird nach einer globalen Disparitätsfunktion $d(x, y)$ gesucht, die eine ebenfalls globale Energie minimiert [34]:

$$E(d(x, y)) = E_{data}(d(x, y)) + \lambda E_{smooth}(d(x, y)). \tag{3.24}$$

   $E_{data}(d)$ gibt dabei wiederum die zu minimierenden Kosten der Disparitätsfunktion $d(x, y)$ für das Bildpaar an. Mit einem Disparitätsraum $D(x, y, d)$ der Größe $x_1 \times y_1$ ergibt sich so:

$$E_{data}(d(x, y)) = \sum_{y=0}^{y_1} \sum_{x=0}^{x_1} D(x, y, d(x, y)). \tag{3.25}$$

   $E_{smooth}(d(x, y))$ modelliert die Annahmen, die der Algorithmus bezüglich der Glattheit der Funktion macht, im einfachsten Fall durch die Differenzen

benachbarter Disparitäten:

$$E_{smooth}(d(x,y)) = \sum_{y=0}^{y_1} \sum_{x=0}^{x_1} \rho(d(x,y)-d(x+1,y))+\rho(d(x,y)-d(x,y+1)). \tag{3.26}$$

Dabei ist $\rho(d)$ wiederum eine monoton steigende Funktion. Prominenteste Vertreter dieser Gruppe sind „nonlinear diffusion“ [37], „simulated annealing“ [38], „maximum flow“ [39], „graph cuts“ [40] und „belief propagation“ [41].

3.2.3.5. Disparitätsverfeinerung

1. Subpixel-Verfeinerung
   An die bisher vorgestellten Schätzverfahren schließt oft ein Schritt zur Verfeinerung der errechneten Disparitätswerte an. Ziel ist häufig eine Subpixelgenaue Darstellung, wie sie vor allem bei Rendering-Verfahren sinnvoll ist [12], nicht jedoch im hier betrachteten Fall.

2. Interframe-Filterung
   Sowohl bei monokularen, als auch bei binokularen Videosequenzen überlappen aufeinanderfolgende Einzelbilder stark. Entsprechend existiert dann für jedes Pixel eine Vielzahl leicht unterschiedlicher Meßergebnisse, aus denen es den tatsächlichen Wert zu schätzen gilt. Mit der Auffassung als „hidden state“ geschieht diese „inter-frame“-Filterung im einfachsten Fall über Mittelwertbildung, häufig jedoch über aufwändige statistische Verfahren wie Kalman- [42] oder Partikelfilter [43, 44].

3. Intraframe-Filterung
   Ebenfalls eine anschließende „intra-frame“-Filterung wird oft praktiziert, wobei häufig ein Medianfilter verwendet wird [33], um die Annahme stetig verteilter Disparitäten zu realisieren.
   Andere Verfahren hingegen versuchten gezielt, bestimmte bekannte Fehlerquellen der Disparitätsschätzung zu kompensieren. Die wichtigste Fehlerquelle ist bei blockbasierten, lokalen Verfahren hier der „foreground fattening“-Effekt [12, 25]: In der Regel ist an Objekträndern die Objektkante die dominante Struktur im Block, verglichen mit dem angrenzenden, meist

wenig strukturierten Hintergrund. Sobald sich ein Block über dem Rand eines Objektes im Vordergrund befindet, bestimmt so dieses das Ergebnis, auch dann, wenn sich ein Großteil des Blocks auf dem Hintergrund befindet. Abbildung 3.9 (a) zeigt, wie Objekte durch diesen Effekt um die halbe Blockgröße verbreitert werden. Das in [25] vorgeschlagene Verfahren

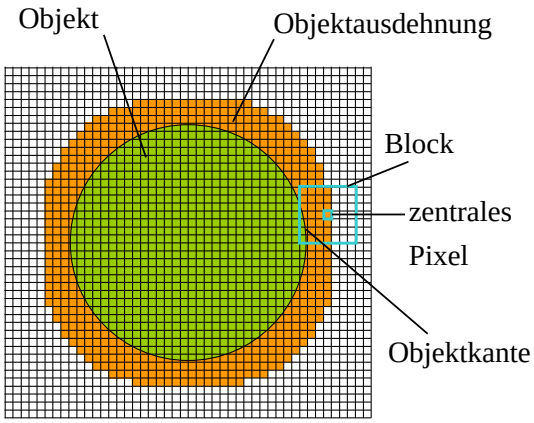


Abbildung 3.9.: Blöcke am Rand von Objekten werden von der dortigen Kantenstruktur dominiert. Dadurch erscheint das Objekt (grün) in der Disparitätskarte vergrößert (orange).

zur Kantenfilterung basiert auf der Detektion von Verdeckungen (vgl. Abbildung 3.10) anhand der SAD verschiedener Subblöcke. Wie sich in [25] jedoch zeigt, führt dieses nicht zur erwarteten Verbesserung, generiert vielmehr zusätzliche Fehler.

3.2.4. Segmentierungsbasiert

In den vergangenen Jahren rückten mit steigender Rechenleistung Verfahren der farbbasierten Bildsegmentierung immer mehr in den Vordergrund. Statt der willkürlichen Bildaufteilung in Blöcke wird das Referenzbild dabei zuerst segmentiert und die Fehlerminimierung dann für die so entstandenen Bildbereiche durchgeführt, entweder wieder analog zu den in 3.2.3.3 beschriebenen lokalen Verfahren durch Verschieben mit zugehöriger Fehlerberechnung, oder durch Ausnutzen globaler Zusammenhänge [45]. Die verwendeten Konzepte sind jedoch mit den bereits in 3.2.3.4 beschriebenen meist identisch, so kommen etwa „Cooperative“-Algorithmen [46] oder „Graph Cuts“ [47] zum Einsatz.

3.3. Zusatzverfahren

3.3.1. Multi-Block-Verfahren

Ähnlich dem bereits in 3.2.3.5 vorgestellten Verfahren zur auf die Disparitätsschätzung folgenden Kantenfilterung, verwenden manche Ansätze schon während der Fehlerminimierung mehrere Blöcke, um die Zugehörigkeit eines Pixels zu Vorder- oder Hintergrund zu präzisieren [25, 31]. In [25] werden so Anordnungen mehrerer Blöcke zur Bestimmung der Disparität bei den betrachteten Koordinaten vorgeschlagen. Hierbei werden für jedes zu berechnende Disparitätspixel mehrere Blöcke im Referenzbild mit den entsprechenden Blöcken im Suchbild verglichen. Dem betrachteten Pixel wird dann die Disparitätsverschiebung des Blocks mit der geringsten SAD zugewiesen. So kann dem „Foreground Fattening“-Effekt entgegengewirkt werden.

3.3.2. Zuverlässigkeitswerte

Fehlschätzungen („Mismatches“) und die daraus folgenden falschen Abstandsinformationen können auf der Tiefenberechnung aufbauende Anwendungen sehr negativ beeinflussen. Die Fehlerprädiktion eines Schätzwertes ist daher eine wertvolle Information, um dessen Zuverlässigkeit einschätzen zu können. Es ist daher praktikabel, Fehler schon während der Schätzung zu prädizieren und entsprechende unsichere Schätzergebnisse zu verwerfen. Eine Übersicht gängiger Verfahren kann in [48] gefunden werden:

- Links/Rechts-Konsistenz
  In der Regel wird das linke Stereobild als Referenzbild verwendet und das rechte als Suchbild, wobei die Disparitäten lediglich für ersteres berechnet werden. Beim „left/right consistency check“ werden ebenfalls für das rechte Bild als Referenz- und das linke als Suchbild die Disparitäten berechnet. Als unzuverlässig gelten die Pixel, für die keine identischen Ergebnisse erzielt werden [23, 49]. Der Hauptvorteil und damit wichtigste Verwendungsfall dieses Verfahrens ist das Ermitteln von Verdeckungen: Da Objekte im Vordergrund eine hohe und solche im Hintergrund eine niedrige Disparität erfahren, sind Bereiche des Hintergrundes an bestimmten Objekträndern

verdeckt (vgl. Abbildung 3.10).

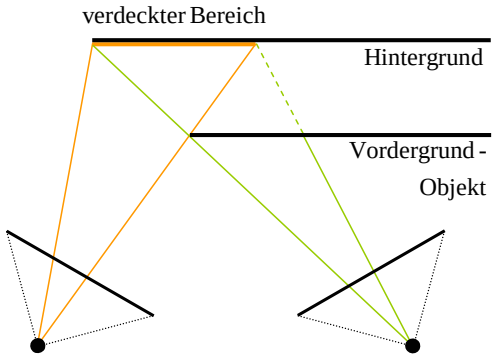


Abbildung 3.10.: Der orange Bereich des Hintergrundes ist für die linke sichtbar, für die rechte Kamera jedoch von einem Objekt im Vordergrund verdeckt.

- Entropieähnlich
  Dabei wird die informationstheoretische Informationsgehalt als Maß der Zuverlässigkeit verwendet. Das Zuverlässigkeitsmaß $C(t)$ berechnet sich für ein Bild $f_r(x, y)$ der Größe $x_1 \times y_1$ dann über die Entropie $H(x, y, t)$ zu [50, 51]:

$$C(t) = \frac{H(x,y,t)}{\min_d SAD_{k\times k}(x,y,t)},$$
$$mit\ H(x,y,t) = \sum_{x=0}^{G-1}\sum_{y=0}^{G-1} g(x,y)\log g(x,y). \tag{3.27}$$

  Dabei ist $g(x, y)$ die Komatrix:

$$g(i,j) \quad = \quad \sum_{p=0}^{x}\sum_{q=0}^{y} \quad = \quad \begin{cases} 1 & \text{, falls } f_r(p,q) = i \\ & \text{und} \\ & f_r(x+\delta x, y+\delta y) = j; \\ & \\ 0 & \text{, sonst.} \end{cases} \tag{3.28}$$

- Fehlermaßbasiert
  Da das Fehlermaß an sich schon als Zuverlässigkeitsmaß für die Ähnlichkeit

zweier Blöcke ist, bietet sich dieses ebenfalls als Zuverlässigkeitsmaß an. Im einfachsten Fall wird dabei ein Fehlerschwellwert verwendet, der nicht überschritten werden darf.
Für den „peak ratio“ wird nicht nur nach dem ersten globalen Minimum der Fehlerfunktion gesucht, sondern auch nach dem zweiten [52, 30, 25]. Als Zuverlässigkeitsmaß wird dann das Verhältnis beider verwendet, wobei sich ein Wert von 0.8 als zweckmäßig erwiesen hat [30], sodaß gilt:

$$Minimum_1 = \begin{cases} zuverlässig & \text{, falls } \frac{Minimum_2}{Minimum_1} \leq 0.8; \\ unzuverlässig & \text{, falls } \frac{Minimum_2}{Minimum_1} > 0.8. \end{cases} \tag{3.29}$$

- „Single View Stereo“
  Dieses in [48] vorgestellte Verfahren geht von einer Stereo-Videosequenz aus und ähnelt dem in [53]. Um die Zuverlässigkeit eines Blocks an der Stelle $(\mathbf{x}_r)$ aus dem aktuellen Referenzbild $f_{r,i}(\mathbf{x})$ für die Zuordnung im zugehörigen aktuellen Suchbild $f_{s,i}(\mathbf{x})$ zu bestimmen, wird dabei zunächst das vorangegangene Referenzbild $f_{r,i-1}(x, y)$ als Suchbild für eine zusätzliche, der eigentlichen Schätzung vorangehende Zuordnung verwendet:

$$d_{conf} = \arg\min_{\mathbf{s}} SAD_{k\times k}(f_{r,i}(\mathbf{x}_r), f_{r,i-1}(\mathbf{x}_r + \mathbf{s})). \tag{3.30}$$

  Da sich direkt aufeinanderfolgende Bilder kaum voneinander unterscheiden, wird davon ausgegangen, daß das Ergebnis dieser Schätzung gering ist und für die Zuverlässigkeit des Blocks ergibt sich:

$$Block(x_r, y_r) = \begin{cases} zuverlässig & \text{, falls } |d_{conf} - \Delta d| < 0; \\ unzuverlässig & \text{, sonst.} \end{cases} \tag{3.31}$$

  Weicht die errechnete Disparität um mehr als eine Schwelle Δd von 0 ab, wird der Block als unzuverlässig bewertet.

- Merkmalspunkte
  Das Gebiet der Zuverlässigkeitswerte stellt einen Berührungspunkt zwischen den in Abschnitt 2 definierten merkmalspunkt- und flächenbasierten Algorithmen dar. Denn ein Pixel im Referenzbild, dessen Zuordnung als zuverlässig eingestuft wurde, kann ebenso als Merkmalspunkt betrachtet werden. Normalerweise werden hier Verfahren, die zuerst nach außergewöhnlichen Punkten suchen und diese danach zuordnen als merkmalspunktbasiert

eingestuft, und solche, die zuerst eine Zuordnung ausführen und dann deren Zuverlässigkeit bewerten als flächenbasiert [54], auch wenn durch die Zuverlässigkeitsberechnung keine dichte Tiefenkarte mehr vorliegt.

3.3.3. Luminanzkorrektur

Bewegtbildsequenzen leiden häufig unter Schwankungen in der Gesamthelligkeit, einerseits aufgrund der Beleuchtungsverhältnisse, andererseits aufgrund der automatischen Kamerablende bzw. des Weißabgleiches. Da diese Schwankungen eine wichtige Fehlerquelle vieler optischer Verfahren sind, existiert eine Vielzahl von Ansätzen zu deren Kompensation. Dabei gilt es zunächst zu unterscheiden zwischen Verfahren, deren Ziel es ist, die Luminanzschwankungen explizit zu modellieren [55, 56] und solchen, die die der Berechnung zugrundeliegende Metrik kompensieren [57]. Das Standardverfahren zur Korrektur der Luminanzschwankungen für den Block-Matching-Algorithmus wurde in [58] und [59] vorgestellt: Dabei wird der Mittelwert $MW_{k\times k}(f(x,y))$ eines quadratischen Blocks der ungeraden Seitenlänge k an der Stelle (x,y) im Bild $f(x,y)$ als dessen Gesamtluminanz betrachtet:

$$MW_{k\times k}(f(x,y)) = \frac{1}{k^2} \sum_{j=-\frac{k-1}{2}}^{\frac{k-1}{2}} \sum_{i=-\frac{k-1}{2}}^{\frac{k-1}{2}} f(x+i, y+j). \tag{3.32}$$

Während der SAD-Berechnung wird dann jeder Block im Suchbild so skaliert, daß sein Mittelwert dem des Blocks im Referenzbild entspricht. Hierzu gibt es zwei Möglichkeiten:

- multiplikativ, wobei sich Gleichung 3.5 ändert zu:

$$SAD^*_{k\times k}\left(f_1(x_1,y_1), f_2(x_2,y_2)\right) = \sum_{i=-\frac{k}{2}}^{\frac{k}{2}} \sum_{j=-\frac{k}{2}}^{\frac{k}{2}} \left| f_1(x_1+i, y_1+j) - c f_2(x_2+i, x_2+j) \right| \tag{3.33}$$

  Der Korrekturfaktor c wird durch Division der beiden Blockmittelwerte berechnet:

$$c = \frac{MV_{k\times k}(f_1(x_1,y_1))}{MV_{k\times k}(f_2(x_2,y_2))}. \tag{3.34}$$

- additiv, durch Modifikation der Gleichung 3.5 zu:

$$SAD^{+}_{k\times k}\left(f_1(x_1,y_1),f_2(x_2,y_2)\right) = \sum_{i=-\frac{k}{2}}^{\frac{k}{2}}\sum_{j=-\frac{k}{2}}^{\frac{k}{2}}\Big|f_1(x_1+i,y_1+j)-(a+f_2(x_2+i,x_2+j))\Big| \quad (3.35)$$

  Den Korrektursummanden a erhält man durch Subtraktion beider Blockmittelwerte:

$$a = MV_{k\times k}(f_1(x_1,y_1)) - MV_{k\times k}(f_2(x_2,y_2)). \quad (3.36)$$

Durch dieses Vorgehen wird die Absolutluminanz eines Blocks als unzuverlässige Information ignoriert und die Fehlerberechnung auf einen reinen Strukturvergleich reduziert. Bei der Abwesenheit von Luminanzschwankungen wird so jedoch wertvolle Information verworfen. Entsprechend wird in [59] vorgeschlagen, zur Fehlervermeidung die Luminanzdifferenz $I_{k\times k}(f_1(x_1,y_1),f_2(x_2,y_2))$ als vorangehenden Indikator für Luminanzschwankungen einzusetzen:

$$I_{k\times k}(f_1(x_1,y_1),f_2(x_2,y_2)) = \sum_{i=-\frac{k}{2}}^{\frac{k}{2}}\sum_{j=-\frac{k}{2}}^{\frac{k}{2}} f_1(x_1+i,y_1+j) - f_2(x_2+i,x_2+j). \quad (3.37)$$

Zusätzlich zum unkompensierten Block Matching wird nur dann zusätzlich das kompensierte Block Matching angewandt, wenn die minimale Luminanzdifferenz zwischen dem Block im Referenzbild und allen Blöcken im Suchbereich eine bestimmte Schranke l nicht überschreitet:

$$\min_{d'} I_{k\times k}(f_r(\mathbf{x}),f_s(\mathbf{x}+d'\mathbf{s})) > l. \quad (3.38)$$

Das Ergebnis des kompensierten Block Matchings wird dann nur verwendet, wenn das Minimum der zugrundeliegenden Fehlerminimierung um mehr als eine Schwelle m kleiner ist als das des unkompensierten Block Matches:

$$\min_{d'} SAD_{k\times k}(f_r(\mathbf{x}_r),f_s(\mathbf{x}_r+d'\mathbf{s})) - \min_{d'} SAD^{comp}_{k\times k}(f_r(\mathbf{x}_r),f_s(\mathbf{x}_r+d'\mathbf{s})) > m. \quad (3.39)$$

Im Gegensatz zum aus der Videokodierung stammenden Block Matching existiert für Verfahren der Fehlerminimierung im Disparitätsraum kein eigener Ansatz zur Luminanzkorrektur, da diese von kalibrierten Stereokameras ohne die oben beschriebenen Effekte oder bereits kompensierten Bildern ausgehen.

3.4. Methoden der Darstellung und Bewertung

Gängiges Mittel zur Darstellung der berechneten Disparitäten ist die Disparitätskarte (engl.: disparity map). Dem Bereich im Referenzbild $f_r(x, y)$, für den eine Disparität bestimmt wurde, wird diese als Zahlenwert zugewiesen, sodaß ein Grauwertbild entsteht. Hohe Disparitäten und damit nahe Objekte resultieren in hellen Werten und umgekehrt, wobei zur besseren Sichtbarkeit oft innerhalb des 8-bit Darstellungsbereichs skaliert wird. Der Wert „0“ wird dabei in der Regel nicht als Disparität, sondern als „undefiniert“ bzw. „unzuverlässig“ (vgl. Abschnitt 3.3.2) verwendet. Die Bewertung eines Ergebnisses findet über den Vergleich mit einer per Hand oder hochgenauen Algorithmus erstellten „Ground Truth“ statt, also einer Disparitätskarte, deren Werte alle als korrekt agenommen werden. Mit der berechneten Tiefenkarte $d(x, y, f_s, f_r)$ und der „Ground Truth“ $d_0(x, y, f_s, f_r)$ ist folgende Bewertung eines Pixels üblich [12]:

$$f(x,y) = \begin{cases} 1 & \text{falls } |d(x,y,f_s,f_r) - d_0(x,y,f_s,f_r)| > 1; \\ 0 & \text{falls } |d(x,y,f_s,f_r) - d_0(x,y,f_s,f_r)| \leq 1. \end{cases} \tag{3.40}$$

Zur Gesamtbewertung eines Algorithmus wird dann für verschiedene Testszenen der Bildgröße $X \times Y$ der Relative Fehler berechnet:

$$F_{rel} = \frac{100\%}{XY} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} f(x,y). \tag{3.41}$$

Dabei wird häufig unter Verwendung entsprechender zusätzlicher „Ground Truth“ zwischen verdeckten Pixeln, Pixeln bei Disparitätsdiskontinuitäten und Pixeln untexturierter Bereiche unterschieden [12, 13].

3.5. Echtzeittauglichkeit

Eine Übersicht verschiedener Echtzeit-Stereosysteme wird in [10] gegeben. Zunächst fällt dabei die Vielzahl der Realisationen mit spezieller Hardware auf, die den hohen Rechenleistungsaufwand selbst der einfacheren Grundalgorithmen zeigt. Da Mobilgeräte über eine weitaus geringere Rechenleistung verfügen, besteht gerade hier ein hoher Bedarf für schnellere Algorithmen bzw. Methoden zur Beschleunigung existierender.
Außerdem sind Verfahren der lokalen Fehlerminimierung für Echtzeitanwendungen zu bevorzugen, insbesondere solche mit der SAD als Fehlerkriterium. Diese sind nicht nur schneller als globale Verfahren, sie liefern oftmals sogar bessere Ergebnisse [33]. Abbildung 3.11 zeigt den Zusammenhang zwischen Qualität und Zeitaufwand der schnellsten Algorithmen des umfangreichen Vergleichs in [12]. Beim schnellsten der Algorithmen handelt es sich um die in [25] vorgestellte lokale Fehlerminimierung mit dem „peak ratio“ als Zuverlässigkeitsmaß (vgl. Abschnitt 3.3.2) und soll die Grundlage für die folgenden Betrachtungen darstellen.
Zusammenfassend kann festgehalten werden, daß die steigende Rechenleistung einen wesentlichen Faktor beim Fortschreiten der Qualität der Stereo-Algorithmen darstellt. Dabei zeigt sich jedoch eine starke Sättigung bezüglich der erreichten Qualität, sodaß immer mehr zusätzliche Rechenleistung für immer kleinere Qualitätszuwächse nötig wird. Die Rechenzeit wird zwar in den meisten Veröffentlichungen mitberücksichtigt, spielt dabei jedoch stets eine der Qualität untergeordnete Rolle. Eine Ausnahme stellen hierbei Veröffentlichungen wie [26] dar, die sich mit einer effizienten Implementierung beschäftigen, also bei gleichbleibender Qualität die dafür nötige Rechenleistung minimieren, oft unter Ausnutzung besonderer plattformabhängiger Befehlssätze wie z.B. MMX.
Wo das jedoch nicht ausreicht, muß unweigerlich zugunsten der Rechenleistung die Qualität eingeschränkt werden. Mit einer systematischen Betrachtung verschiedener fehlerbehafteter Beschleunigungsverfahren und deren Wechselwirkung in Hinblick auf Zeit- und Fehlerverhalten soll diese Arbeit den Stand der Technik entsprechend sinnvoll ergänzen.

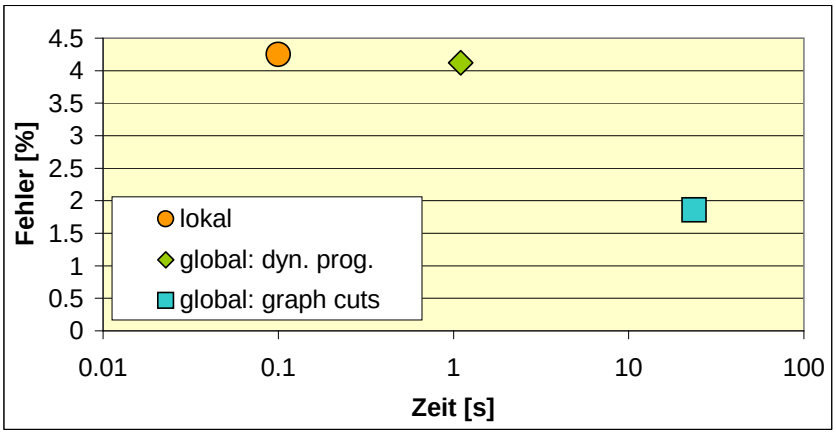


Abbildung 3.11.: Vergleich der effizientesten Algorithmen aus [12]. „ Lokal“ beschreibt das Verfahren aus [25], das hier dementsprechend als Ausgangspunkt der folgenden Optimierungen dient (vgl. Abschnitt 3.2.3.3). „Dynamic Programming“ und „Graph Cuts“ entsprechen den schnellsten globalen Verfahren (vgl. Abschnitt 3.2.3.4).

3.6. Parametrisierung

In diesem Abschnitt soll die in Kapitel 1 beschriebene Parametrisierung des Algorithmus hergeleitet werden, um diesen skalierbar zu machen. Dabei gilt es zwischen solchen Parametern zu unterscheiden, die einmalig geometrieabhängig eingestellt werden und während des gesamten Programmlaufs konstant bleiben (Abschnitt 3.6.1), und solchen, die tatsächlichen zur Skalierung des Systems während des Programmlaufs herangezogen werden (Abschnitt 3.6.2).

3.6.1. Geometrie und Konstanten

Nach Abschnitt A.1.2 ergibt sich für die Gesamtdisparität d in Abhängigkeit vom Kamerawinkel α, der Armlänge a und den Objekt-Kreiskoordinaten (ω, l) (vgl. Abbildung 3.12) mit Ursprung im Schwenkzentrum:

$$\begin{aligned} d(\alpha,\omega,l) = d_1(\alpha,\omega,l) + d_2(\alpha,\omega,l) = \\ f\tan\left(\arctan\left(\frac{1}{\frac{l}{a\sin\frac{\alpha}{2}+\omega}-\frac{1}{\tan\frac{\alpha}{2}+\omega}}\right)+\frac{\alpha}{2}+\omega\right) \\ + f\tan\left(\arctan\left(\frac{1}{\frac{l}{a\sin\frac{\alpha}{2}-\omega}-\frac{1}{\tan\frac{\alpha}{2}-\omega}}\right)+\frac{\alpha}{2}-\omega\right). \end{aligned} \quad (3.42)$$

Alternativ dazu kann d in Abhängigkeit von den kartesischen Objektkoordinaten (x, z) mit Ursprung im Mittelpunkt der durch die beiden Kamerabrennpunkte aufgespannten Basislinie dargestellt werden als (vgl. Abschnitt A.1.1):

$$\begin{aligned} d(x,z,\alpha) &= d_1(x,z,\alpha) + d_2(x,z,\alpha) \\ &= f\tan\left(\pi-\epsilon-\arctan\frac{z}{a\sin\frac{\alpha}{2}+x}\right) \\ &+ f\tan\left(\pi-\epsilon-\arctan\frac{z}{a\sin\frac{\alpha}{2}-x}\right). \end{aligned} \quad (3.43)$$

An dieser Stelle wird klar, daß keine absoluten Abstände berechnet werden können, da der Schwenkwinkel α zwischen den Kameraarmpositionen zu den beiden Aufnahmezeitpunkten nicht direkt meßbar und Gleichungen 3.42 und 3.43 somit unterbestimmt sind. Abbildung 3.13 zeigt das Disparitätsverhalten für alle Punkte der durch die beiden optischen Achsen aufgespannten Epipolarebene für das mit

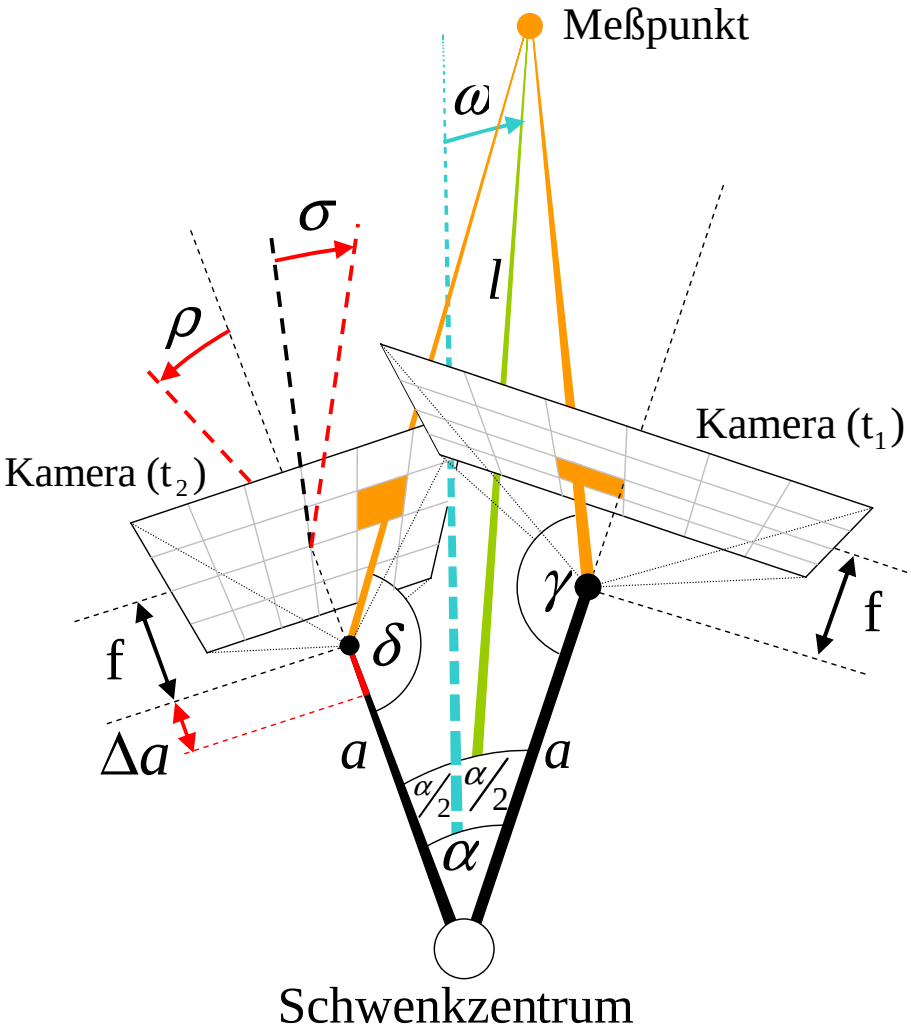


Abbildung 3.12.: Geometrie des Kameraschwenks mit den betrachteten Störgrößen (rot) σ, ρ und Δa.

Gleichung A.8 beschriebene klassische Stereosystem (Abbildung 3.13 (b)) mit der zugrundeliegenden Geometrie aus Abbildung 3.13 (a) und für den Kameraschwenk (Abbildung 3.13 (c)) mit den geometrischen Zusammenhängen nach Abbildung 3.13 (d). Hier zeigt sich beim Kameraschwenk nicht nur ein dem Stereosystem gleichwertiges Disparitätsverhalten, vielmehr treten auftrund der offenen Kameraanordnung kein Nulldurchgang und ausschließlich positive Disparitäten auf.

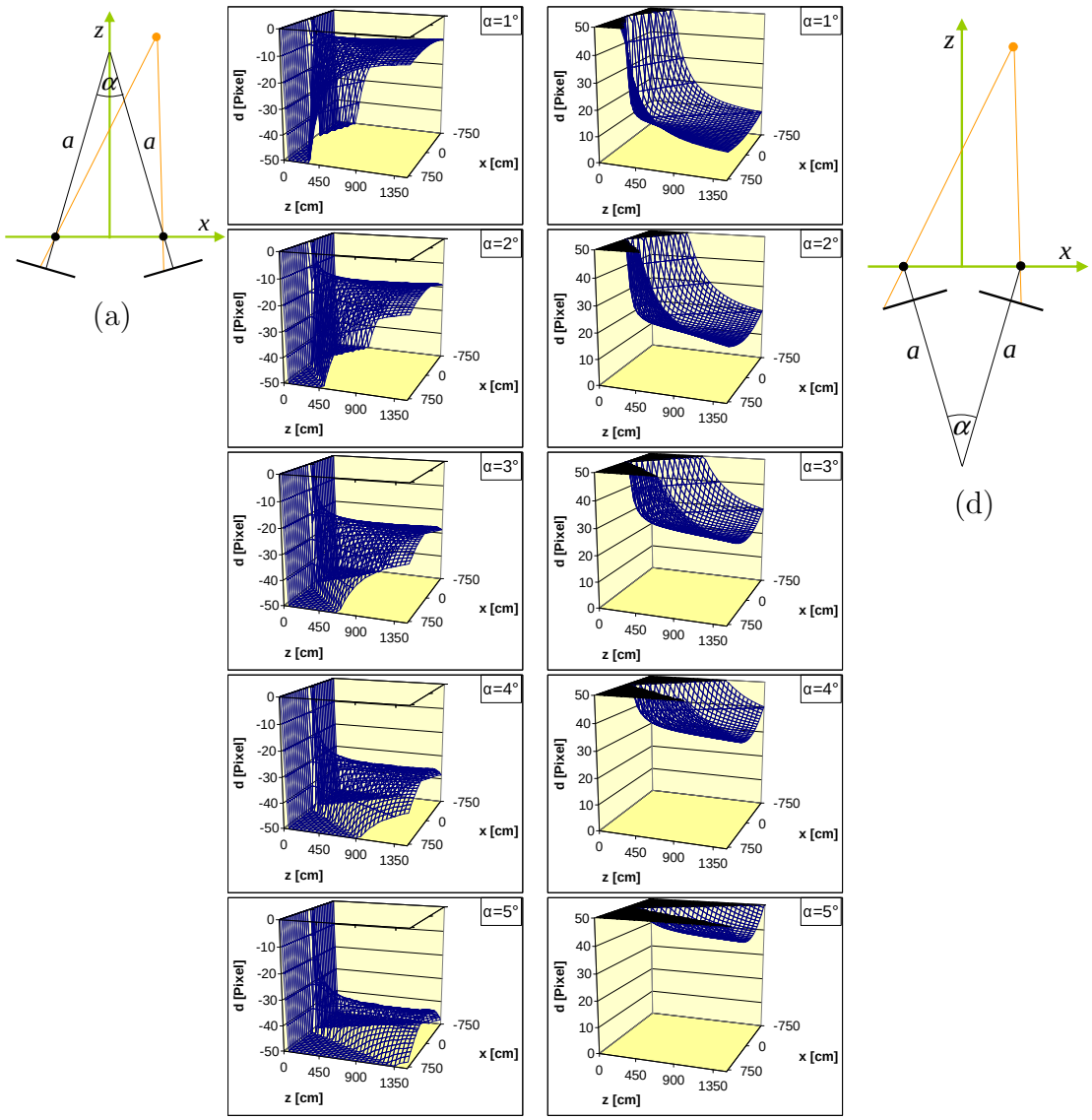


Abbildung 3.13.: (a) Kartesisches Koordinatensystem bei klassischen Stereokameras. (b) Disparitätsverteilung über die Epipolarebene beim klassischen Stereosystem. (c) Disparitätsverteilung über die Epipolarebene beim Kameraschwenk. (d) Kartesisches Koordinatensystem beim Kameraschwenk. Für beide Fälle wurde $a = 50cm$ verwendet.

3.6.1.1. Geometrische Fehler

In Abschnitt 3.2 wurden bereits die drei der Verwendung von Stereokorrespondenzverfahren zugrundeliegenden Annahmen vorgestellt, auf denen die Geometrie in Abbildung 3.12 basiert. Entsprechende, durch eine Abweichung von dieser Konstellation zu einem der beiden Aufnahmezeitpunkte entstehende Fehlergrößen sind:

1. Der Rotationswinkel σ um die Optische Achse (vgl. Abbildung 3.2 (a)),
2. Der Rotationswinkel ρ um eine Bildachse (vgl. Abbildung 3.2 (b)),
3. Die Armlängendifferenz $a(t_1) - a(t_2) = \Delta a$ (vgl. Abbildung 3.2 (c)).

Die fehlerbehaftete Disparität $d'(\alpha, \omega, l, \Delta a)$ ergibt sich aus Gleichung A.28 zu:

$$\begin{aligned} d'(\alpha, \omega, l, \Delta a) = d + \Delta d = \\ f \tan\left(\arctan\left(\frac{1}{\frac{l}{a \sin\frac{\alpha}{2}+\omega} - \frac{1}{\tan\frac{\alpha}{2}+\omega}} \right) + \frac{\alpha}{2} + \omega \right) \\ + f \tan\left(\arctan\left(\frac{1}{\frac{l}{(a-\Delta a) \sin\frac{\alpha}{2}-\omega} - \frac{1}{\tan\frac{\alpha}{2}-\omega}} \right) + \frac{\alpha}{2} - \omega \right). \end{aligned} \tag{3.44}$$

Die fehlerbehaftete Disparität $d'(\alpha, \omega, l, \sigma)$ ergibt sich mit der Herleitung aus A.46 zu:

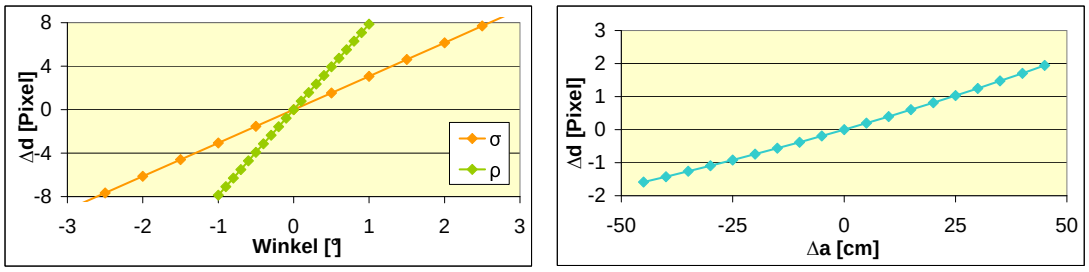
$$d'(\alpha, \omega, l, \sigma) = d_1 + d_2' = d_1 + \sqrt{y^2 + d_2^2} \cos\left(\sigma + \arctan\frac{y}{d_2} \right). \tag{3.45}$$

Die fehlerbehaftete Disparität $d'(\alpha, \omega, l, \rho)$ ergibt sich mit der Herleitung aus A.41 zu:

$$\begin{aligned} d'(\alpha, \omega, l, \rho) = \\ d_1 + f \tan\left(\arctan\left(\frac{1}{\frac{2f \sin\frac{\rho}{2} \sin\gamma}{l \sin\alpha_2 \sin\left(\gamma - \frac{\rho}{2} - \frac{\pi}{2}\right)} + \frac{1}{\tan\left(\gamma - \frac{\rho}{2} - \frac{\pi}{2}\right)}} \right) + \frac{\pi - \rho}{2} \right). \end{aligned} \tag{3.46}$$

Die entsprechenden absoluten Fehler $\Delta d = d'(\alpha, \omega, l, \Delta a) - d$, $\Delta d = d'(\alpha, \omega, l, \sigma) - d$ und $\Delta d = d'(\alpha, \omega, l, \rho) - d$ können Abbildung 3.14 entnommen werden.

Es zeigt sich bei einem Fehler von $0,04^{px}/_{cm}$ eine hohe Robustheit gegenüber Änderungen der Armlänge Δa.



(a) (b)

Abbildung 3.14.: Abhängigkeit der Fehlerdisparität Δd von (a) den Rotationswinkeln ρ um eine Bildachse und σ um die optische Achse und (b) einer Armlängenänderung Δa.

Rotationen σ um die Optische Achse erzeugen für Pixel am Bildrand einen maximalen Fehler von ca. $3^{px}/_{\circ}$.
Die höchste Sensitivität kann hingegen mit ca. $8^{px}/_{\circ}$ gegenüber Rotationen ρ um eine Bildachse orthogonal zur optischen Achse festgestellt werden.

3.6.1.2. Suchbereichsgröße

Eine Grundkonstante der Disparitätsschätzung ist die Suchbereichsgröße w (vgl. Gleichung 3.9), von der die Größe der maximalen Disparität abhängt, die gefunden werden kann. Eine zu niedrige Suchbereichsbreite schränkt somit das Ergebnis von vornherein ein, eine zu hohe Suchbereichsbreite erhöht einerseits den Aufwand, andererseits die Wahrscheinlichkeit für das Auftreten zusätzlicher Fehler. Entsprechend soll die Suchbereichsbreite nicht als Skalierungsparameter verwendet, sondern fest auf bestimmte Rahmenbedingungen eingestellt werden.
Da Meßpunkte im Vordergrund die größte Disparität erzeugen, bestimmt die Suchbereichsgröße die minimal zulässige Entfernung von Objekten. Der Schwenkwinkel α_s der verwendeten Stereopaare soll während der Einzelschätzungen durch eine globale Bewegungsschätzung möglichst konstant gehalten werden, um für zeitlich aufeinanderfolgende Schätzungen eine möglichst konstante Basislinie und damit in aufeinanderfolgenden Bildern einheitliche Disparitätswerte zu erlangen. Die Winkel α_f zwischen aufeinanderfolgenden Einzelbildern dürfen entsprechend kleiner als α_s sein, da α_s dann aus mehreren Winkeln zusammengesetzt werden kann, keinesfalls jedoch größer (vgl. Abbildung 3.15). Der Extremfall, der zugelassen werden soll ist entsprechend $\alpha_s = \alpha_f$, also die Verwendung direkt

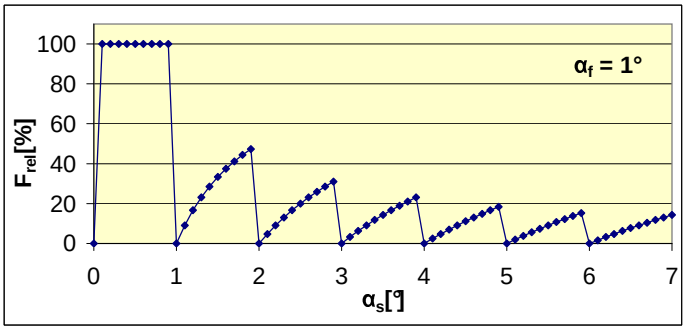


Abbildung 3.15.: Bei als fest angenommenen Winkeln $\alpha_f = 1°$ zwischen aufeinanderfolgenden Einzelbildern sinkt der maximale relative Fehler bei einem größeren Winkel α_s zwischen aktuellem Bild und Schätzpartner.

aufeinanderfolgender Einzelbilder. Durch die feste Bildfrequenz von 25fps hängt der maximale Winkel α_s zwischen zwei aufeinanderfolgenden Aufnahmen von der Schwenkgeschwindigkeit v ab. Abbildung 3.16 zeigt entsprechend den Zusammenhang zwischen Schwenkgeschwindigkeit und Disparität bei verschiedenen Meßpunktabständen. Der anwendungsspezifisch realistisch erscheinende „Worst Case“ einer Objektdistanz l von 1,5m bei einer Bewegungsgeschwindigkeit von $90^{cm}/_{s}$ resultiert danach in einer Disparität von 50 Pixeln. Um diese in allen Schwenkrichtungen erfassen zu können, wird im Folgenden ein Suchbereich der Größe 100×100 verwendet.

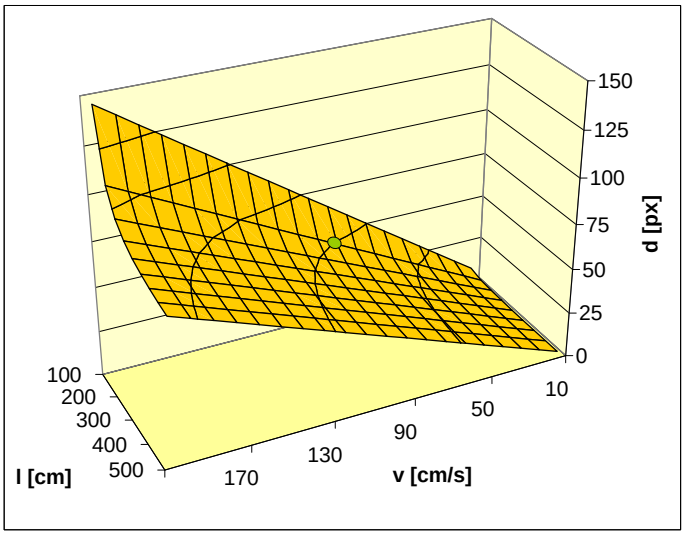


Abbildung 3.16.: Disparität d für einen Meßpunkt in Abhängigkeit von dessen Entfernung l zum Schwenkzentrum und der Schwenkgeschwindigkeit v bei $25\frac{Bildern}{Sekunde}$. Der grüne Punkt markiert dabei die verwendete Suchbereichsbreite von 50 Pixeln, bei der ein Meßpunkt im Abstand von $150cm$ bei Schwenkgeschwindigkeiten bis $90\frac{cm}{s}$ zugeordnet werden kann.

3.6.1.3. Suchbildbestimmung

Jedes aktuelle Kamerabild f_r wird gespeichert, sodaß sich die Menge Z aller vorhergehenden Bilder ergibt:

$$Z = \{f_{r-1}, f_{r-2}, f_{r-3}, ...\}. \tag{3.47}$$

Um für jedes aktuelle Bild f_r (Referenzbild) den geeignetsten Stereopartner f_s (Suchbild) so auszuwählen, daß auch die Schätzungen aufeinanderfolgender Zeitpunkte miteinander korrespondieren, gilt es diesen so zu wählen, daß der Kamerawinkel α zum aktuellen Bild und damit die Basislinie zwischen den beiden Stereobildern f_r und f_s stets näherungsweise konstant bleibt. Da dieser jedoch nicht direkt meßbar ist, muß über eine globale Bewegungsschätzung der globale Verschiebungsvektor $\mathbf{m}(f_1, f_2)$ bestimmt werden und als Auswahlkriterium dienen. Der Index s wird somit über den Abstand zu einem voreingestellten „Soll"-Verschiebungsvektor $\mathbf{b}$ bestimmt:

$$s = \arg\min_i |\mathbf{b} - \mathbf{m}(f_r, f_i)|. \tag{3.48}$$

Hierzu muß jedoch sichergestellt werden, daß sich die Schätzung von $\mathbf{b}$ für jedes Einzelbild auf denselben Bildinhalt bezieht. Ferner dient im Falle der Übernahme von Meßwerten aus vorangegangenen Disparitätskarten der globale Bildverschiebungsvektor $\mathbf{m}$ als Abbildungsvorschrift.
Bei den meisten Sequenzen kann von einem näherungsweise einheitlichen Hintergrund mit einzelnen Objekten im Vordergrund ausgegangen werden. Wird der Hintergrund als Bezugspunkt der Bildauswahl verwendet, führt dies bei Übernehmen von Werten aus älteren Disparitätskarten zu einem Verschmieren von Objekten im Vordergrund (vgl. Abbildung 3.17). Die Funktion $\mathbf{m}(f_1, f_2)$ soll also so gewählt werden, daß sie den größten Verschiebungsvektor zwischen den Bildern f_1 und f_2 bestimmt, womit sich theoretisch ein hierarchisches Block Matching (vgl. Abschnitt 3.2.2.1) anbietet. Da aufgrund deren Dominanz bei Berechnung der SAD beim Block Matching davon ausgegangen werden kann, daß sich das Ergebnis stets auf Objekte im Vordergrund bezieht, soll auf eine Hierarchisiernung verzichtet und für die Funktion $\mathbf{m}(f_1, f_2)$ lediglich ein einstufiges Block Matching verwendet werden. Der im Folgenden verwendete zentrierte Referenzblock besitzt die Größe $100 \times 80 Pixel$ und wird 6-fach unterabgetastet.

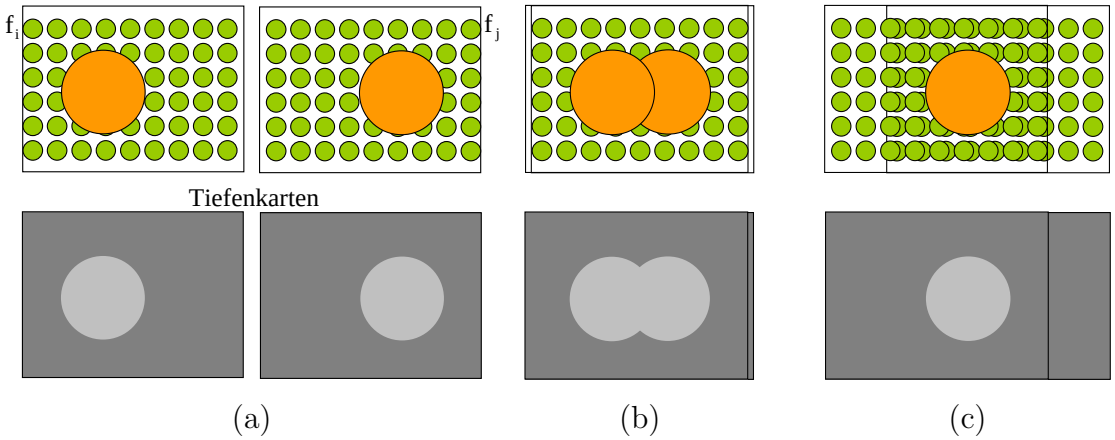


Abbildung 3.17.: (a) Beispielhafte Einzelbilder f_i und f_j mit Objekt im Vordergrund (orange) und Hintergrund (grün) einheitlicher Disparität. (b) Verwendung der Hintergrunddisparität als Gesamtbildverschiebung führt bei entsprechender Überlagerung zu einem „Verschmieren“ des Objektes im Vordergrund beim Zusammenführen der Disparitäten aus aufeinanderfolgenden Disparitätsbildern. (c) Bei Verwendung der Vordergrunddisparität als Gesamtbildverschiebung wird zwar der Hintergrund verschiedener Einzelbilder nicht exakt aufeinander abgebildet, bei einheitlicher Hintergrunddisparität bleibt dies in der Disparitätskarte jedoch ohne Effekt.

3.6.2. Skalierungsparameter

3.6.2.1. Grundalgorithmus

Wie bereits beschrieben, kommen mehrere Algorithmen zur Realisierung derselben Funktionalität in Frage, zwischen denen im Bedarfsfall gewechselt werden kann:

1. Block Matching (BM)
   „Straight-forward“-Implementierung der blockbasierten Stereokorrespondenz, wie in Abschnitt 3.2.2 beschrieben.

2. Disparitätsraum direkt (DRd)
   Wie in Abschnitt 3.2.3.2 beschrieben, wird dabei zunächst der Disparitätsraum aufgebaut mit anschließender lokaler Kostenaggregation und Fehlerminimierung.

3. Disparitätsraum moving average (DRma)
   Hierbei wird zunächst die Kostenaggregation für alle Ebenen des Disparitätsraums wie in 3.2.3.2 beschrieben als „Moving-Average-Filter“ durchgeführt. Die lokale Fehlerminimierung wird dann anschließend separat durchgeführt.

Für den zugehörigen Skalierungsparameter g ergibt sich somit die Menge G möglicher Zustände:

$$g \in G = \{BM, DRd, DRma\} . \tag{3.49}$$

3.6.2.2. Blockgröße

Große Blöcke beinhalten viele Pixel und besitzen aufgrund der daraus folgenden Vielzahl von Vergleichswerten zwar eine hohe Zuverlässigkeit bezüglich Fehlschätzungen, führen jedoch mit zunehmender Größe zu einer schlechteren Ortsauflösung. Kleine Blöcke hingegen führen zu einer höheren Ortsauflösung und Konturgenauigkeit, mit abnehmender Blockgöße steigt jedoch wiederum die Wahrscheinlichkeit für Fehlschätzungen. Für die Abhängigkeit $F(k)$ des Fehlers F von der Blockgröße bzw. -kantenlänge k (vgl. Gleichung 3.5) ist also eine nicht umkehrbare Funktion zu erwarten (vgl. Abbildung 3.18), sodaß ein eindeutiges Auflösen nach $k(F)$ nicht möglich ist. Da jedoch der Abschnitt der langsameren Rechenzeiten für den hier betrachteten Anwendungsfall uninteressant ist, sollen lediglich Werte des „schnelleren“ Abschnitts links des Fehlerminimums verwendet werden.
Da ein Block als lokale Umgebung seines zentralen Pixels betrachtet wird, sind ferner lediglich Blöcke ungerader Seitenlänge zu betrachten, sodaß sich die Menge möglicher Zustände für den Parameter k ergibt zu:

$$k \in K = \{1, 3, 5, \ldots\} . \tag{3.50}$$

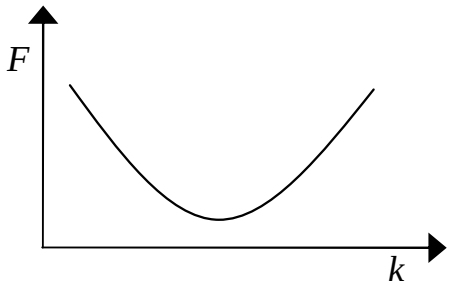


Abbildung 3.18.: Erwartetes Verhalten des Fehlers F in Abhängigkeit von der Blockgröße k.

3.6.2.3. Blockabstand

In der Videocodierung ist es üblich, das Referenzbild in ein Blockraster aufzuteilen und das Block Matching auf die so entstandenen disjunkten Blöcke anzuwenden (siehe Gleichung 3.10). Wird diese Vorgehensweise auf die Disparitätsschätzung übertragen, entsteht ein Disparitätsbild niedriger Auflösung, es werden jedoch wenige Block Matches benötigt. Bei der Disparitätsschätzung durch Kostenminimierung im Disparitätsraum wird für jedes Pixel im Referenzbild ein Block als dessen lokale Umgebung verwendet (siehe Gleichung 3.12), was in einer hohen Auflösung, jedoch auch einer Vielzahl an Blöcken resultiert. Als neuer Skalierungsparameter soll entsprechend der „Blockabstand“ b eingeführt werden, sodaß sich bei einer Blockgröße von k für die Blockpositionen (x_r, y_r) im Referenzbild ergibt:

$$x_r, y_r \in \left[\frac{k}{2}, \frac{k}{2} + b, \frac{k}{2} + 2b, ...\right]. \tag{3.51}$$

Die beiden oben beschriebenen Extremfälle entsprechen somit $b = k$ für disjunkte und $b = 1$ für pixelweise Blöcke und der entsprechende Skalierungsparameter ist:

$$b \in B = \{1, 2, 3, ..., k\}. \tag{3.52}$$

Abbildung 3.19 zeigt die entsprechenden Fehler, wobei hier zwei Fehlerquellen zu unterscheiden sind: Einerseits der Fehler durch die Unähnlichkeit der Objektränder aufgrund der niedrigen Ortsauflösung, andererseits der Fehler durch das „foreground fattening“ (vgl. Abbildung 3.9). Bei steigenden Werten für b sinkt der Einfluß der Unähnlichkeit, bis für $b = 1$ lediglich die Objektausdehnung als Fehlerquelle bestehen bleibt (vgl. Abbildung 3.19). In Hinblick auf eine anschließende Fehlerkompensation scheint es also sinnvoll, zunächst durch eine höhere Auflösung selektiv einen der beiden Fehler zu eliminieren.

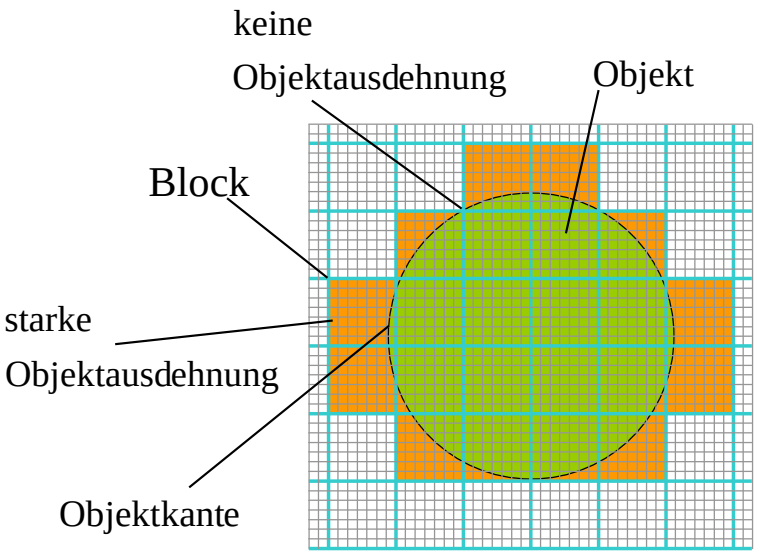


Abbildung 3.19.: Bei einer groben Disparitätsauflösung aufgrund großer Blockabstände nimmt die Ähnlichkeit der Konturen des Objektes (grün) im Referenzbild und dessen Abbildung (orange) in der Disparitätskarte ab. Entsprechend existiert keine einheitliche Objektausdehnung.

3.6.2.4. Abtastperiode

Da beim Block Matching die SAD in einem einzelnen Schritt berechnet wird, ist eine Blockunterabtastung hier sehr einfach und konsistent realisierbar (vgl. Abschnitt 3.2.2.2).
Bei der lokalen Fehlerminimierung im Disparitätsraum hingegen sind die Schritte der Kostenberechnung und -aggregation voneinander getrennt, um eine redundante Berechnung von Absoluten Differenzen zu vermeiden. Findet eine Unterabtastung bereits bei der Kostenberechnung statt, können im Schritt der Kostenaggregation lediglich abgetastete Pixel berücksichtigt werden.
Da bei der Disparitätsschätzung ein Block als lokale Umgebung seines zentralen Pixels betrachtet wird, ist es sinnvoll, für dieses immer einen Abtastwert zu erhalten. Entsprechend bieten sich zur Kostenaggregation nur solche Blöcke an, die um Pixel zentriert sind, für die eine Kostenberechnung stattgefunden hat. Damit stehen dann jedoch nicht mehr alle Blockabstände zur Verfügung, nur noch solche, die ein Vielfaches der Abtastrate sind. Dieses Vorgehen entspricht einer entsprechenden Skalierung der Einzelbilder. Für Gleichung 3.18 gilt dann mit der

Bildabtastperiode u:

$$x, y \in \{0, \pm u, \pm 2u, \pm 3u, \ldots\}. \tag{3.53}$$

Umgekehrt ergibt sich so die Menge U der möglichen Bildabtastperioden in Abhängigkeit vom Blockabstand b zu:

$$U = \left\{ u \in \mathbb{N}_0 | \frac{b}{u} \in \mathbb{N}_0 \right\}. \tag{3.54}$$

Die darauffolgende Kostenaggregation findet mit entsprechend modifizierter Blockgröße $k' \times k' = \frac{k}{u} \times \frac{k}{u}$ statt.

Sollen alle Blockabstände möglich sein, darf nach einer vollständigen Kostenberechnung erst bei der Kostenaggregation unterabgetastet werden, sodaß mit der Blockabtastperiode a für Gleichung 3.20 gilt:

$$i, j \in \{0, \pm a, \pm 2a, \pm 3a, \ldots\}. \tag{3.55}$$

Bei Implementierung als „Moving Average Filter“ sind auch in diesem Fall nur Blockabstände möglich, die Vielfache der Abtastrate sind (vgl. Abbildung 3.20), eine Unterabtastung während der Kostenaggregation bringt hier also keinerlei Vorteil gegenüber einer sofortigen Unterabtastung bei der Fehlerberechnung.

Darüberhinaus unterscheiden sich verschiedene Kombinationen von Blockgröße

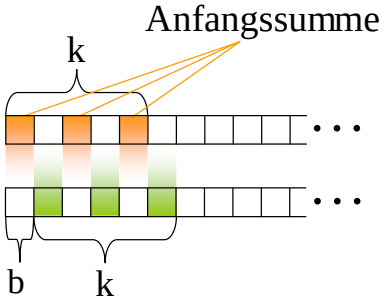


Abbildung 3.20.: Unterabtastung bei Implementierung der Kostenaggregation als Moving Average Filter (vgl. Abbildung 3.6): Wird der Block der Kantenlänge k um den Blockabstand b verschoben, stimmt die Abtastung der vorangegangenen Summe (orange) nicht mit der Abtastung überein, wie sie der aktuellen Summe entspräche (grün). Entsprechend sind nur Blockabstände möglich, die ein Vielfaches der Abtastrate sind.

und Abtastperiode nicht voneinander (vgl. Abbildung 3.21). Entsprechend soll an dieser Stelle die effektive Blockgröße K_{eff} eingeführt werden:

$$k_{eff} = 2 \left\lfloor \frac{\frac{k-1}{2}}{a} \right\rfloor a + 1. \tag{3.56}$$

Damit ergibt sich für den zugehörigen Skalierungsparameter a als Menge der möglichen Zustände die Menge ganzzahliger, positiver Teiler der jeweiligen Blockgröße k:

$$A = \left\{ a \in \mathbb{N}_0 | \frac{\frac{k-1}{2}}{a} \in \mathbb{N}_0 \right\}. \tag{3.57}$$

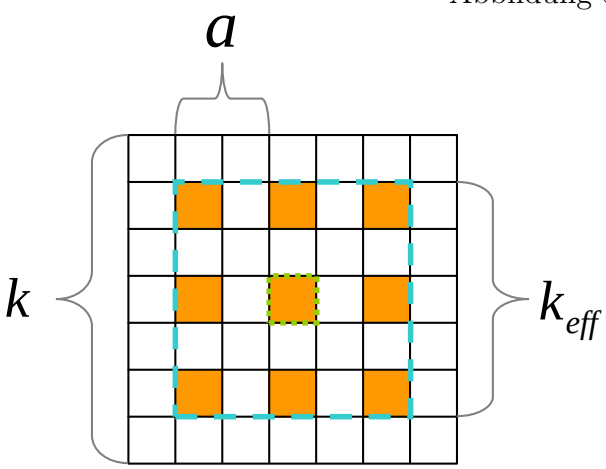


Abbildung 3.21.: Da bei der Disparitätsschätzung ein Block als Umgebung seines zentralen Pixels (grün) verwendet wird, macht es Sinn, eine Unterabtastung (orange) so zu platzieren, daß dieses abgetastet wird. Durch Unterabtastung mit der Abtastperiode a ergibt sich die neue effektive Blockgröße k_{eff} (blau).

3.6.2.5. Suchstrategie

In Abschnitt 3.2.2.3 wurden verschiedene Verfahren zur Optimierung der Suche bei der SAD-Minimierung angesprochen. Da der hier verwendete Suchbereich entlang eines Vektors verläuft (vgl. Abschnitt 3.6.1.3), bietet sich dafür eine Logarithmische Suche an. Der entsprechende Parameter ist dabei die Anzahl l der Suchschritte (l-Stufen-Suche):

$$l \in L = \{1, 2, 3, \ldots\}\,. \tag{3.58}$$

Die anfängliche Abtastperiode des Suchberichs ergibt sich damit zu $2^{(l-1)}$.

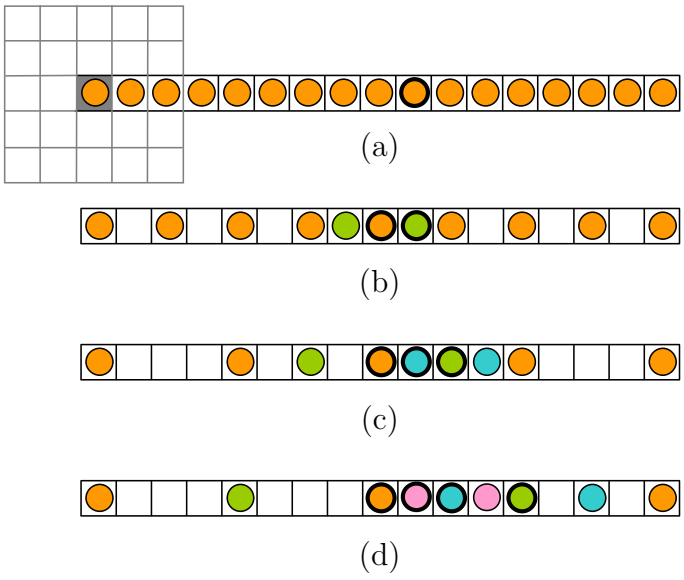


Abbildung 3.22.: Logarithmische Abtastung eines Suchvektors mit (a) 1- (b) 2- (c) 3- und (d) 4-Stufen-Suche. Die Abtastpositionen der ersten Stufe sind dabei orange dargestellt, die der zweiten grün, die der dritten blau und die der vierten rosa. Eine Abtastposition ist dabei das zentrale Pixel eines Blocks im Suchbild (grau in (a)). Das Minimum der jeweiligen Stufe ist fett umrandet.

3.6.2.6. Schätzbereich

Die Grundvoraussetzung für Disparitätsschätzung mit einer einzelnen Kamera ist Bewegung (vgl. Abschnitt 3.1), wobei sich aufeinanderfolgende Disparitätsbilder aufgrund der hohen Abtastrate von 25 Bildern pro Sekunde stark überlappen. Eine einfache Methode zur weiteren Reduktion der Anzahl der verwendeten Blöcke ist entsprechend eine Einschränkung des Schätzbereichs, also des Bereichs, für den die Disparität bestimmt wird. Das Zusammensetzen der einzelnen Disparitätskarten kann über den globalen Bildverschiebungsvektor $\mathbf{m}$ koordiniert werden (vgl. Abschnitt 3.6.1.3). Um auch bei kleinen Schätzbereichen stets eine möglichst vollständige Disparitätskarte zu gewährleisten, ist es zweckmäßig, neue Werte dort zu ergänzen, wo Fehlstellen entstehen, nämlich entlang eines Streifens orthogonal zur Bewegungsrichtung auf Seite der größeren Komponente des globalen Bildverschiebungsvektors. Außerdem soll der Einfachheit halber auf „updating"-Strategien, wie sie in Abschnitt 2 angesprochen wurden verzichtet und die aktuellsten Disparitätswerte immer direkt in die Disparitätskarte übertragen werden. Der zugehörige Skalierungsparameter s bezieht sich entsprechend auf die Breite des verwendeten Schätzbereiches.
Bei einer Verwendung der tatsächlichen Breite s_{px}, gemessen in Pixeln, als Skalierungsparameter, muß für eine Bestimmung der Anzahl der Schätzwerte zusätzlich der Blockabstand b berücksichtigt werden. Um hier schon im Vorfeld für eine Entkoppelung der Parameter zu sorgen, soll der Parameter s die Anzahl der durchgeführten Schätzungen entlang der größeren Suchvektorkomponente angeben, mit dem Zusammenhang für die effektive Schätzbereichsbreite in Pixeln:

$$s_{px} = sb. \tag{3.59}$$

Da sich jedoch die Einzelbildverschiebungen und somit auch die entsprechenden Maximalwerte s_{max} der verschiedenen Sequenzen voneinander unterscheiden, muß hier weiterhin eine einheitliche Darstellung von s geschaffen werden, um ein Zusammenführen der Testwerte unterschiedlicher Sequenzen zu ermöglichen. Dazu soll schließlich eine prozentuale Betrachtung der Anzahl von Schätzungen entlang der größeren Suchvektorkomponente, bezogen auf deren maximale Anzahl s_{max} herangezogen werden:

$$s_{\%} = 100\% \frac{s}{s_{max}} \in S_{\%} = \{0\%, \ldots, 100\%\}\,. \tag{3.60}$$

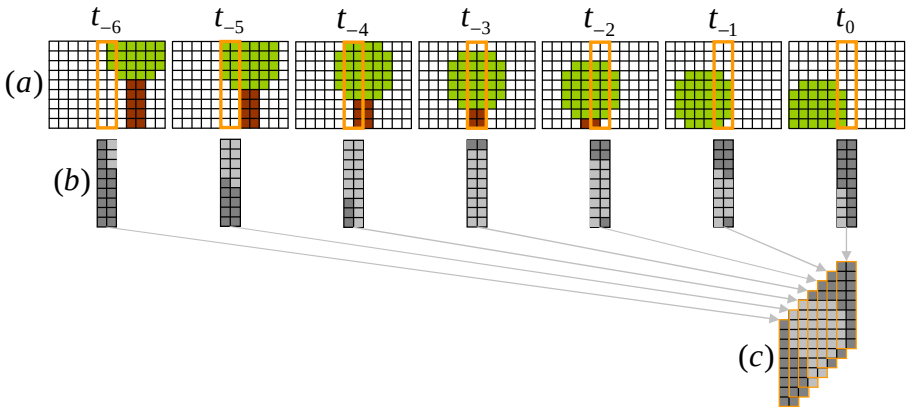


Abbildung 3.23.: (a) Für jedes Referenzbild zum aktuellen Zeitpunkt t_0 wird eine partielle Disparitätskarte berechnet (hier: s=2). (b) Aus der aktuellen Disparitätskarte und deren Vorgängern zu den Zeitpunkten $t_{-1}, t_{-2}, \ldots$ wird mithilfe des globalen Bildverschiebungsvektors die komplette Disparitätskarte zusammengesetzt (c).

3.6.2.7. Luminanzkorrektur

Wo in Abschnitt 3.3.3 das Standardverfahren zur Kompensation von Luminanzschwankungen beim Block Matching bereits motiviert und vorgestellt wurde, existiert kein entsprechender Ansatz für disparitätsraumbasierte Verfahren, da diese in der Regel auf „echte“, also gleichzeitig aufgenommene Stereobildpaare angewendet werden. Hinzu kommt die Tatsache, daß Mobilkameras in der Regel über eine automatische Blende und Weißabgleich verfügen, die von außen nicht beeinflußbar sind und zusätzlich gravierende Luminanzschwankungen erzeugen können.
Zur Herleitung eines neuen Ansatzes zur Luminanzkorrektur bei Fehlerminimierung im Disparitätsraum soll zunächst das Verhalten des luminanzkorrigierten Block Matchings analysiert und dann entsprechend umgesetzt werden [60]. Dazu werden für jedes Pixel die multiplikativen und additiven Luminanzkorrekturwerte c und a betrachtet, wie sie dem Ergebnis der jeweiligen Fehlerminimierung zugrundeliegen. Diese wurden nach den Gleichungen 3.34 bzw. 3.36 aus den Mittelwerten des Referenzblocks und des Suchblocks berechnet, der die SAD schließlich minimiert. Da es sich bei c um einen Faktor und bei a um einen Summanden handelt, entsprechen die beiden Werte $c = 1$ und $a = 0$ dabei dem unkorrigierten Fall.

Abbildung 3.24 zeigt die Histogramme über die entsprechenden Korrektursummanden und -faktoren für alle Pixel dreier Testsequenzen, wie sie später im Rahmen des Modellierungsprozesses verwendet und detaillierter erklärt werden.
Hier läßt sich zunächst eine Gauß'sche Verteilung erkennen, mit dem Maximum nahe den unkorrigierten Fällen.

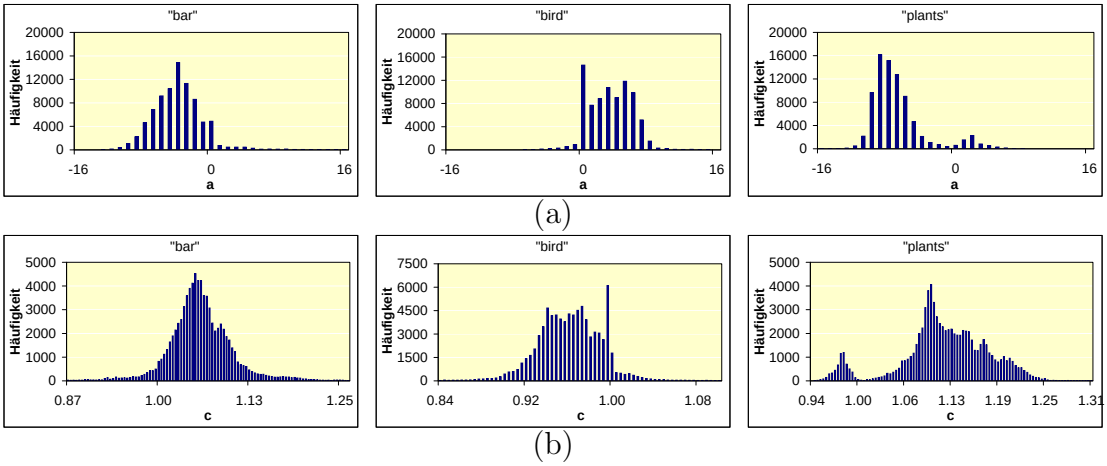


Abbildung 3.24.: Histogramme aller berechneten (a) Korrektursummanden a und (b) -faktoren c beim luminanzkorrigierten Block Matching aus Abschnitt 3.3.3 für die Testbildpaare „bar“, „bird“ und „plants“ (vgl. Abschnitt 3.7.2).

Anhand entsprechender „Ground Truth“ kann für jedes Pixel entschieden werden, ob im Vergleich zu einer nicht luminanzkorrigierten Berechnung der Disparitätswerte ein Fehler eliminiert wurde. Entsprechende Histogramme mit der Anzahl der eliminierten Fehler für verschiedene Korrekturwerte werden in Abbildung 3.25 gezeigt und erlauben eine Aussage über deren jeweiligen Erfolg. Auch hier zeigt sich eine Gauß-artige Verteilung um ein Maximum nahe dem unkorrigierten Fall.

Analog zu den eliminierten Fehlern kann festgestellt werden, für welche Pixel durch die Luminanzkorrektur zusätzlich Fehler generiert wurden, die bei einer nicht luminanzkorrigierten Berechnung der Disparitätswerte noch nicht vorhanden waren. Abbildung 3.26 zeigt wiederum die Histogramme der generierten Fehler für verschiedenen Korrekturwerte.

Abbildung 3.27 zeigt die Fehlerbilanzen, also die Differenz zwischen generierter und elimienierter Fehleranzahl für verschiedene Korrekturwerte. Eine positive Fehlerbilanz entsteht dabei, wenn durch einen Korrekturwert die Anzahl der ge-

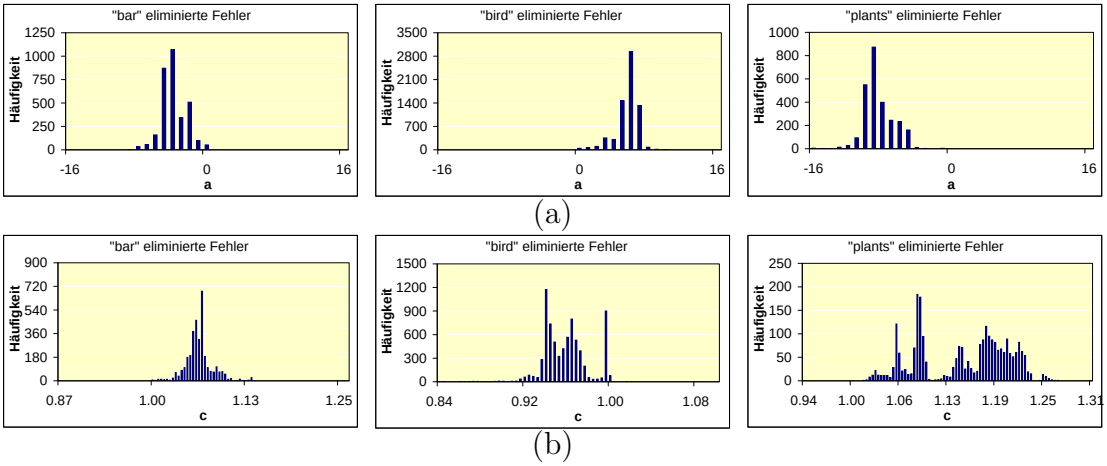


Abbildung 3.25.: Histogramme aller im Vergleich zum unkorrigierten Fall eliminierten Fehler bei mittelwertbasierter Berechnung eines (a) Korrektursummanden a und (b) Korrekturfaktoren c entsprechend dem luminanzkorrigierten Block Matching nach Abschnitt 3.3.3 für die Testbildpaare „bar“, „bird“ und „plants“ (vgl. Abschnitt 3.7.2).

nerierten Fehler größer ist als die der eliminierten. Vor allem bei der additiven Luminanzkorrektur (3.27 (a)) fällt auf, daß sich diese positiven Fehlerbilanzen (rot) an den Rändern der Verteilung der Korrekturwerte befinden. Dieses Verhalten ist tendenziell auch für die multiplikative Luminanzkorrektur (3.27 (b)) zu beobachten.

Es können also drei Beobachtungen festgehalten werden:

1. Die erfolgreichen Werte sind um $a = 0$ für die additive und $c = 1$ für die multiplikative Korrektur verteilt.

2. Die erfolgreichen Korrekturwerte sind gaußverteilt, Werte im Zentrum der Verteilung sind also erfolgreicher, als Werte am Rand.

3. Fehler treten ausschließlich am Rand der Verteilung auf.

Aus diesen Beobachtungen können drei Annahmen bezüglich der Korrekturwerte getroffen werden:

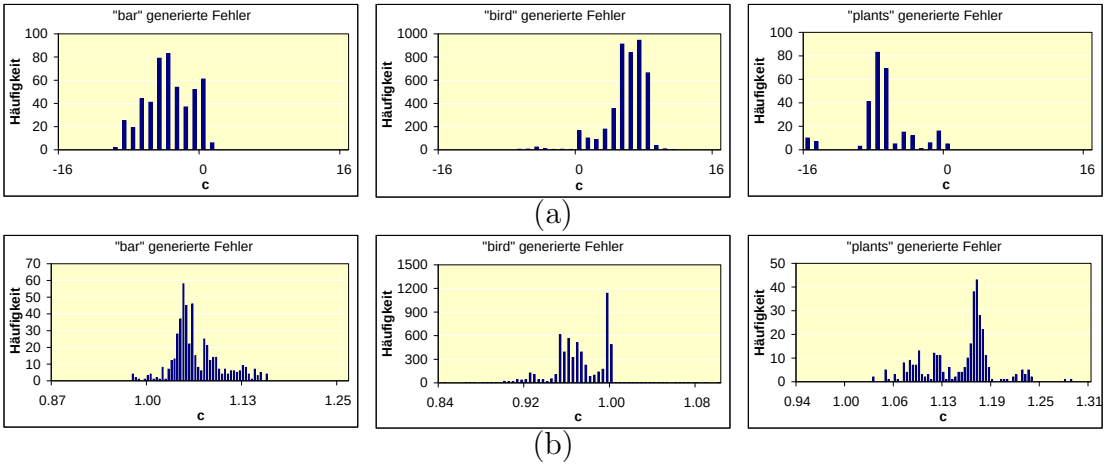


Abbildung 3.26.: Histogramme aller im Vergleich zum unkorrigierten Fall neu generierten Fehler bei mittelwertbasierter Berechnung eines (a) Korrektursummanden a und (b) Korrekturfaktoren c entsprechend dem luminanzkorrigierten Block Matching nach Abschnitt 3.3.3 für die Testbildpaare „bar“, „bird“ und „plants“ (vgl. Abschnitt 3.7.2).

1. Die Korrekturwerte sind dem unkorrigierten Fall $a = 0$ für die additive und $c = 1$ ähnlich.
2. Werte nahe $a = 0$ bzw. $c = 1$ sind zu bevorzugen.
3. Es gibt nur wenige verschiedene Korrekturwerte.

Der Ansatz zu einer skalierbaren Luminanzkorrektur im Disparitätsraum basiert auf einer vordefinierten Menge von Korrekturwerten, die anhand der getroffenen Annahmen gestaltet werden soll.
Da der kleinstmögliche von „0“ verschiedene Integer-Wert „1“ ist, sind für den additiven Fall die Annahmen 1 und 2 erfüllt mit der Menge der Korrektursummanden p:

$$p \in \{-p_{max}, \ldots, -1, 0, 1, \ldots, p_{max}\}. \tag{3.61}$$

Annahme 3 verlangt außerdem nach einem kleinen p_{max}, das hier als Skalierungsgröße variabel bleiben soll.
Für jeden Wert dieser eingeschränkten Menge von Korrektursummanden wird entsprechend ein eigener Disparitätsraum aufgestellt (vgl. Gleichung 3.18):

$$D_p^+(\mathbf{x}, d) = |f_r(\mathbf{x}) - (p + f_s(\mathbf{x} + d\mathbf{s}))|. \tag{3.62}$$

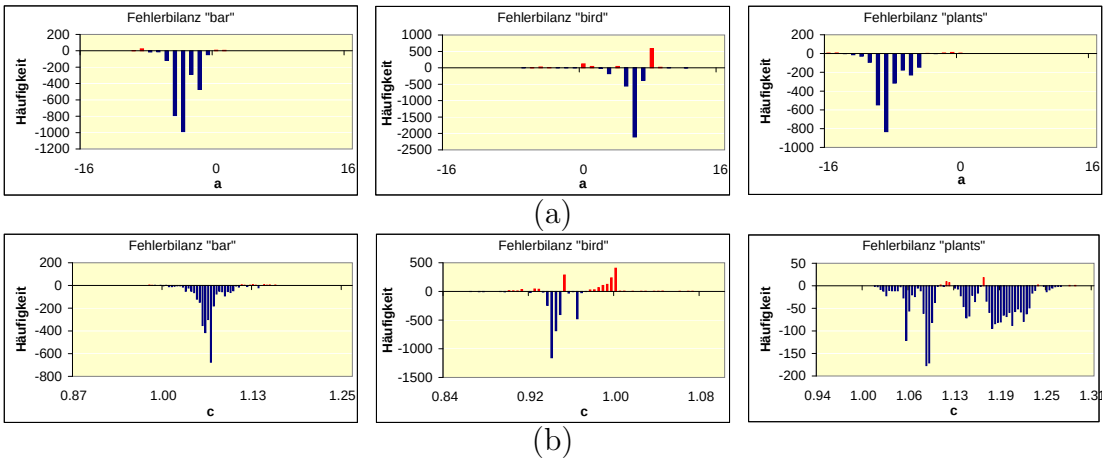


Abbildung 3.27.: Fehlerbilanzen (Differenz aus generierten und eliminierten Fehlern) für (a) additiv und (b) multiplikativ luminanzkorrigiertes Block Matching. Eine positive Fehlerbilanz (rot) für einen Korrekturwert bedeut, daß mehr Fehler generiert als verworfen wurden.

Die Fehlerminimierung aus Gleichung 3.23 wird dann über alle so entstandenen $2p_{max} + 1$ verschiedenen, vorskalierten Disparitätsräume ausgedehnt:

$$d(\mathbf{x}_r) = \arg\min_{d'} \Big(SAD_k^{agg}\big(D_p^+(\mathbf{x}_r, d')\big)\Big). \tag{3.63}$$

Der größtmögliche 8-Bit-Integer-Wert ist „255“, sodaß sich als kleinster Faktor, der bei einer Integerdivision einen Effekt zeigt „$\frac{1}{255}$“ ergibt. Bei einer multiplikativen Luminanzkorrektur ergibt sich die Menge der Korrekturwerte q somit zu:

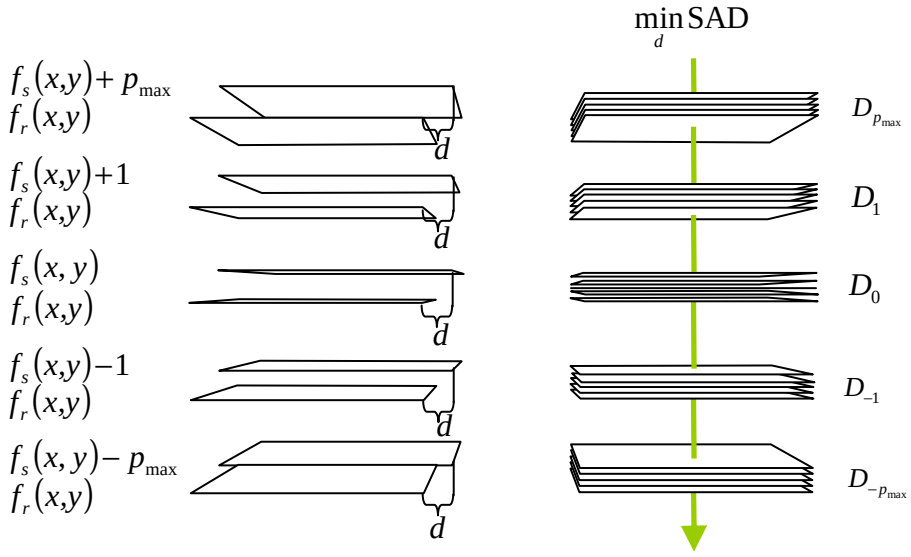
$$q \in \big\{ -q_{max}, \ldots, 1 - \frac{1}{255}, 1, 1 + \frac{1}{255}, \ldots, q_{max} \big\}. \tag{3.64}$$

Annahme 3 fordert hier wiederum ein kleines q_{max} und für die vorskalierten Disparitätsräume ergibt sich:

$$D_q^*(\mathbf{x}, d) = |f_r(\mathbf{x}) - (q f_s(\mathbf{x} + d\mathbf{s}))|. \tag{3.65}$$

Für eine einheitliche Darstellung mit der additiven Korrektur soll die Menge aus Gleichung 3.61 verwendet werden, sodaß sich Gleichung 3.65 ändert zu:

$$D_q^*(\mathbf{x}, d) = |f_r(\mathbf{x}) - ((1 + \frac{p}{255}) f_s(\mathbf{x} + d\mathbf{s}))|. \tag{3.66}$$



(a) (b)

Abbildung 3.28.: (a) Für jeden der vordefinierten Korrekturwerte $p \in P$ wird bei additiver Luminanzkorrektur ein eigener Disparitätsraum $D_p(x, y, d)$ gebildet. Dazu wird das mit dem jeweiligen Korrekturwert skalierte Suchbild $f_s(x, y)$ um ein variables d gegen das Referenzbild $f_r(x, y)$ verschoben (vgl. Abschnitt 3.2.3). (b) Die Kostenaggregation und Fehlerminimierung findet dann über alle entsprechenden Disparitätsräume statt.

Die Fehlerminimierung wird analog zur additiven Korrektur über die $2p_{max} + 1$ vorskalierten Disparitätsräume durchgeführt:

$$d(\mathbf{x}_r) = \arg\min_{d'} \left(SAD_k^{agg}\left(D_p^*(\mathbf{x}_r, d')\right)\right). \tag{3.67}$$

Mit steigenden Werten von p_{max} steigt zunächst der Aufwand. Solange die Werte klein bleiben ist ein stetiges Abnehmen der Fehler aufgrund des Korrektureffekts zu erwarten. Für größere Werte läßt sich ein Steigen der Fehlerrate aufgrund von Fehlschätzungen voraussagen, bis die Qualität des Ansatzes nach 3.3.3 erreicht ist.
Der vorgestellte Ansatz kann analog auf das Block Matching übertragen werden, sodaß sich Gleichung 3.8 bei additiver Luminanzkorrektur ändert zu:

$$d^+(x_r, y_r) = \left| \arg\min_{\mathbf{s}} SAD_{k\times k}(f_r(x_r, y_r), p + f_s(x_r + s_x, y_r + s_y)) \right|. \tag{3.68}$$

Analog folgt bei multiplikativer Korrektur:

$$d^*(x_r, y_r) = \left| \arg\min_{\mathbf{s}} SAD_{k\times k}(f_r(x_r, y_r), (1 + \frac{p}{255}) f_s(x_r + s_x, y_r + s_y)) \right| . \quad (3.69)$$

Der schließlich verwendete Skalierungsparameter c für die Luminanzkorrektur entspricht dem Wert p_{max}:

$$c \in C = \{0, 1, 2, \ldots\} . \quad (3.70)$$

3.6.2.8. Peak Ratio

Die Verwendung von Zuverlässigkeitswerten wird über die Tatsache motiviert, daß sich bei vielen Anwendungen ein nicht definierter Wert vorteilhafter auswirkt, als ein falscher Wert, der als richtig angenommen wird. Da jedoch nicht nur fehlerhafte Disparitätswerte als unzuverlässig eingestuft werden, sondern auch korrekt geschätzte, nimmt mit Verwendung des Peak Ratios (vgl. Abschnitt 3.3.2) auch deren Anzahl ab. Eine Bewertung des Verfahrens als Zuverlässigkeitsmaß kann also nicht nur anhand der Fehlerrate durchgeführt werden, da ein sehr selektiver Peak Ratio zwar sämtliche Fehler vermeiden, jedoch auch alle korrekten Werte verhindern würde. Da hier jedoch nicht das Verfahren an sich getestet, sondern ausschließlich der in der Literatur vorgeschlagene Wert $0,8$ eingestellt werden soll, der bereits ein optimales Verhältnis aus verworfenen Fehlern und verworfenen korrekten Schätzungen darstellt, soll bei Verwendung des Peak Ratios auf eine gesonderte Betrachtung unzuverlässiger Werte verzichtet werden. Aufgrund des Ziels einer vollständigen Disparitätskarte sollen vielmehr auch als unzuverlässig eingestufte Werte als Fehler aufgefaßt werden, was zunächst zu einer Erhöhung der Gesamtfehlerzahl führt. Diese wird jedoch wiederum durch den anschließenden Schritt der Fehlstellenergänzung (vgl. Abschnitt 3.6.2.9) relativiert. Als entsprechender Skalierungsparameter ergeben sich somit lediglich die Zustände „p=0“ für „ohne Peak Ratio“ und „p=1“ für „mit Peak Ratio“:

$$p \in P = \{0, 1\}. \tag{3.71}$$

3.6.2.9. Fehlstellenergänzung

Um die durch die Verwendung von Zuverlässigkeitsmaßen entstandenen undefinierten Stellen zu interpolieren, vergleicht dieses Filter die beiden nächstgelegenen Disparitäten links und rechts einer Lücke und setzt den kleineren der beiden Werte ein. Damit ergeben sich auch hier die beiden Zustände „f=0“ für „ohne Fehlstellenergänzung“ und „f=1“ für „mit Fehlstellenergänzung“:

$$f \in F = \{0, 1\}. \tag{3.72}$$

Eine Anwendung dieses Filters macht natürlich nur bei gleichzeitiger Anwendung eines Zuverlässigkeitsmaßes (Peak Ratio) Sinn.

3.6.2.10. Zusammenfassung der Parameter

Parameter	Bezeichnung		Menge möglicher Zustände	Grundzustand
Grundalgorithmus	g	$\in$	G={BM,DRd,DRma}	$g_0 = DRma$
Blockgröße	k	$\in$	K={1, 3, 5,. . .,k_{max}}	$k_0 = k_{max}$
Blockabstand	b	$\in$	B={1, 2, 3,. . .,k}	$b_0 = 1$
Bildabtastperiode	u	$\in$	$U = \left\{u \in \mathbb{N}_0 \middle\vert \frac{b}{u} \in \mathbb{N}_0\right\}$	$u_0 = 1$
Blockabtastperiode	a	$\in$	$A = \left\{a \in \mathbb{N}_0 \middle\vert \frac{b}{\frac{l-1}{2}} \in \mathbb{N}_0\right\}$	$a_0 = 1$
Schrittzahl bei logarithmischer Suche	l	$\in$	L={1, 2, 3, 4}	$l_0 = 1$
Schätzbereichsbreite	s	$\in$	S={0,. . .,100}	$s_0 = 100$
Maximaler Luminanzkorrekturwert	c	$\in$	C={0, 1, 2, 3,. . .}	$c_0 = 0$
Peak Ratio	p	$\in$	$P = \{0, 1\}$	$p_0 = 0$
Fehlstellenergänzung	f	$\in$	$F = \{0, 1\}$	$f_0 = 0$

Tabelle 3.1.: Übersicht über die verwendeten Skalierungsparameter: Der jeweils fettgedruckte Zustand entspricht dem Grundzustand, also dem Referenzpunkt, von dem aus einzelne Parameteränderungen betrachtet werden sollen.

3.7. Modellierung

3.7.1. Vorgehensweise

Ziel dieses Abschnitts ist die Herleitung einer Skalierungsvorschrift anhand der im vorangegangenen Abschnitt eingeführten Skalierungsparameter. Dazu soll über die Durchführung einer großen Anzahl von Tests die mittlere Fehlerhäufigkeit aller Kombinationen der möglichen Zustände der Skalierungsparameter über viele Sequenzen ermittelt und als Schätzwert für deren tatsächliche Fehlerwahrscheinlichkeit verwendet werden. Anders als bei vielen Veröffentlichungen, bei denen anhand einzelner weniger Testsequenzen bestimmte Fehlerquellen analysiert werden, ist hier also keine Bewertung der verwendeten Algorithmen das Ziel, diese hat bereits anderenorts ausführlich stattgefunden und war ausschlaggebend für deren Verwendung (vgl. Abschnitt 3.5). Zusammen mit deren ermittelten Rechenzeiten kann dann die Effizienz verschiedener Zustandskonstellationen verglichen und diese in ein sinnvolles Schema zur Skalierung des Gesamtsystems überführt werden.

3.7.2. Testsequenzen

Es kommen zwei Gruppen zu je 10 Testsequenzen zur Verwendung:

1. Die mobilen Testsequenzen aus Abbildung 3.29 wurden ausschließlich durch freihändiges Schwenken einer mobilen Kamera erzeugt, viele auch ohne die Intention einer späteren Verwendung als Testsequenz, sodaß von einem realistischen Aufnahmeverhalten ausgegangen werden kann. Ferner wurde keinerlei Vorverarbeitung durchgeführt und weder Bildrauschen, noch Linsenverzerrungen entfernt. Die „ground truths“ wurden dabei per Hand durch Errechnen der jeweiligen Disparitätsvektoren erstellt.

2. Ergänzend werden die in Abbildunge 3.30 gezeigten „Middlebury“-Testsequenzen [13] verwendet, sodaß einerseits ein Vergleich zwischen realen und inszenierten Testsequenzen gezogen werden kann. Andererseits spiegeln die Ergebnisse bezüglich dieser Sequenzen das unbeeinflußte Grundverhalten der Algorithmen wieder, weshalb diese ebenfalls in die Testmenge mit auf-

genommen wurden. Diese Sequenzen wurden mit einer kalibrierten Kamera erzeugt, sind praktisch rauschfrei und sämtliche Linsenverzerrungen wurden beseitigt. Die zugehörigen „ground thruths“ wurden mit hochpräzisen, aktiven Verfahren erstellt.

Zur Bestimmung der Rechenzeiten wurde einerseits mit einem Pentium M bei einer Taktfrequenz von 1,7GHz (PC), andererseits mit dem typischen Mobiltelefonprozessor ARM9 bei einer Taktfrequenz von 150MHz (Mobilgerät) gearbeitet.

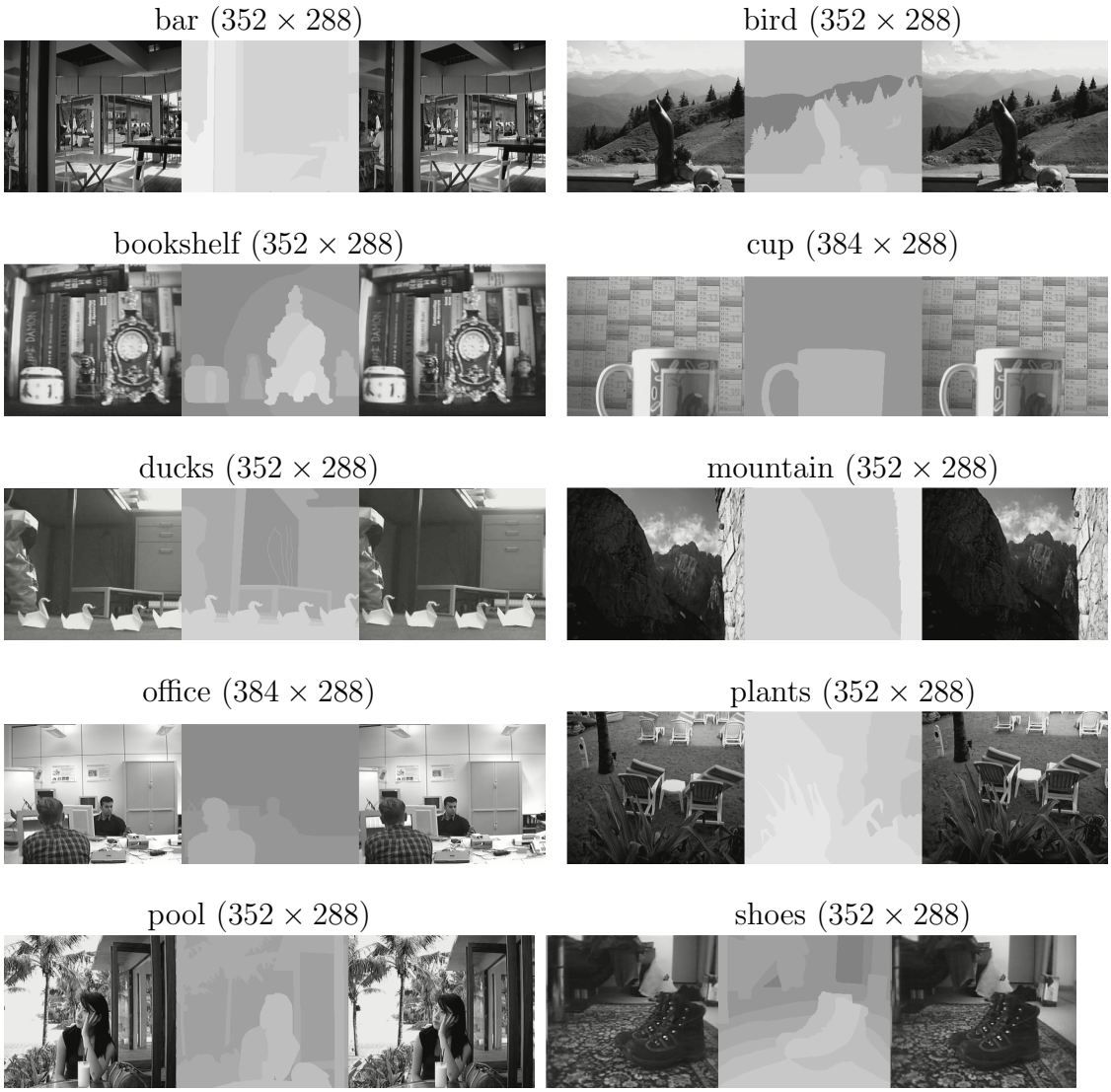
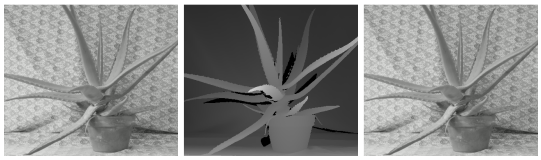


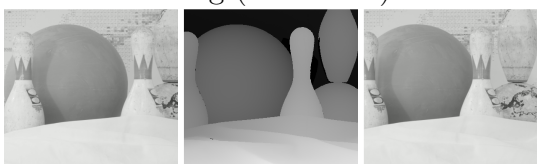
Abbildung 3.29.: Name, Größe [Pixel], Referenzbild, Ground Truth und Suchbild der verwendeten „Mobile“-Testsequenzen.

aloe (426 × 370)

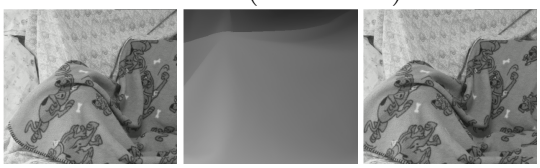


baby (412 × 370)

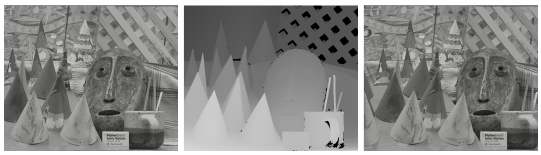
bowling (416 × 370)



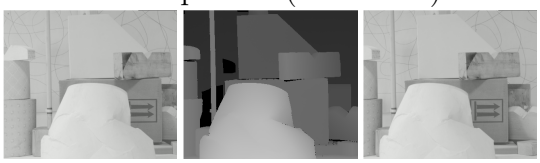
cloth (416 × 370)



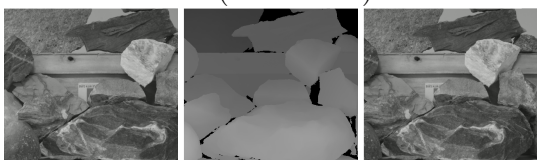
cones (450 × 370)



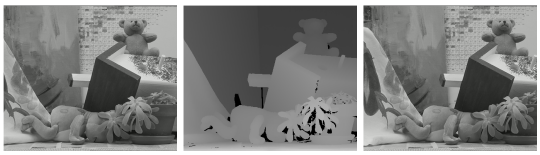
lampshade (432 × 370)



rocks (425 × 370)



teddy (450 × 370)

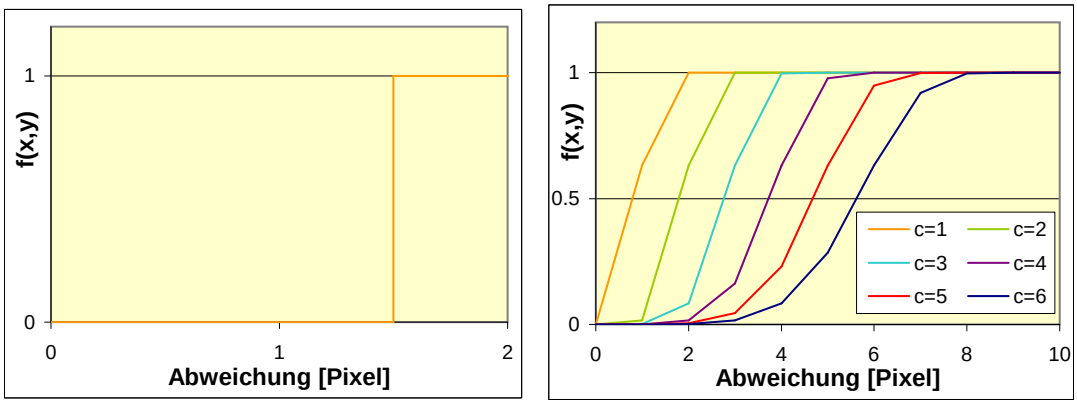


tsukuba (384 × 288)

venus (434 × 383)

Abbildung 3.30.: Name, Größe [Pixel], Referenzbild, Ground Truth und Suchbild der verwendeten „Middlebury“-Testsequenzen.

3.7.2.1. Fehlerkriterien



(a) (b)

Abbildung 3.31.: (a) „Hartes“ Fehlerkriterium nach Gleichung 3.73 bei $f_0 = 1$ und (b) „weiche“ Fehlerkriterien nach Gleichung 3.74.

Die Grundanforderungen an das System sollen bei der Berechnung der mittleren Fehlerhäufigkeiten über die Toleranz des einem Fehler zugrundeliegenden Fehlerkriteriums einfließen. Analog zu Gleichung 3.40 kann die Fehlerschwelle f_0 entsprechend variiert und für eine höhere Fehlertoleranz zu größeren Werten verschoben werden (vgl. Abbildung 3.31 (a)):

$$f(x,y) = \begin{cases} 1 & \text{, falls } |d(x,y,f_s,f_r) - d_0(x,y,f_s,f_r)| > f_0; \\ 0 & \text{, falls } |d(x,y,f_s,f_r) - d_0(x,y,f_s,f_r)| \leq f_0 \end{cases} \quad mit\ f_0 \in \{0,1,2,...\}. \tag{3.73}$$

Alternativ können hier auch „ weichere“ Fehlermaße verwendet werden (vgl. Abbildung 3.31 (b)):

$$f(x,y) = 1 - e^{-\left(\frac{d(x,y,f_s,f_r) - d_0(x,y,f_s,f_r)}{c}\right)^6}. \tag{3.74}$$

Die mittlere Fehlerhäufigkeit R_f als Schätzwert für die Fehlerwahrscheinlichkeit P_f ergibt sich dann über die gemittelte Fehlersumme aller überprüften Pixel der n Ergebnisbilder der Größe $x \times y$:

$$R_f = \frac{1}{nxy} \sum_n \sum_{x,y} f(x,y). \tag{3.75}$$

Im Folgenden wird zunächst ausschließlich das gängige Fehlerkriterium mit $f_0 = 1$ nach Gleichung 3.73 verwendet, bis in Abschnitt 3.7.4 schließlich der Einfluß des Fehlerkriteriums diskutiert wird.

3.7.3. Testergebnisse der Einzelparameter

3.7.3.1. Blockgröße

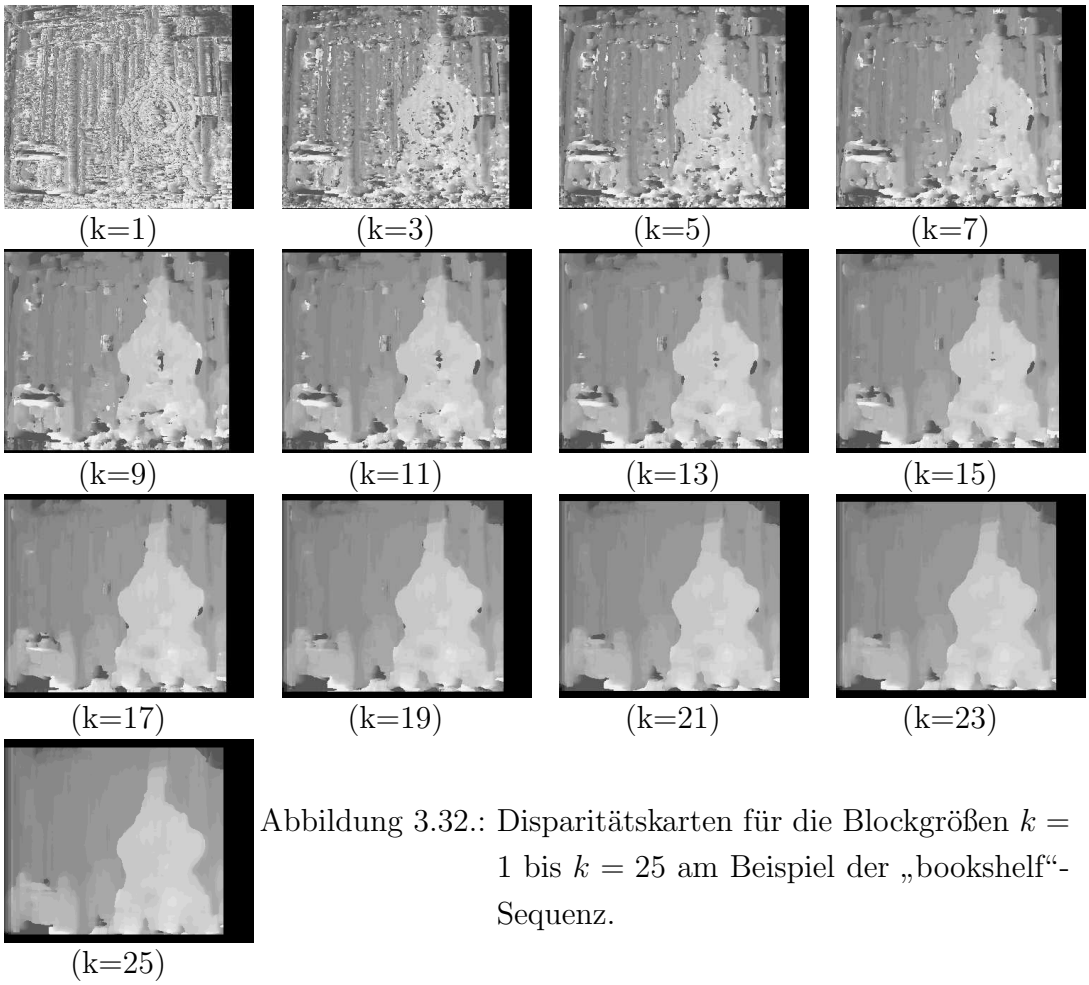


Abbildung 3.32.: Disparitätskarten für die Blockgrößen $k = 1$ bis $k = 25$ am Beispiel der „bookshelf“-Sequenz.

Abbildung 3.32 zeigt beispielhaft die berechneten Disparitätskarten der „bookshelf“-Sequenz für die verschiedenen Zustände der Blockgröße k. Die entsprechenden mittleren Fehlerhäufigkeiten über alle Testsequenzen mit den jeweiligen Rechenzeiten gehen aus Abbildung 3.33 hervor. Hier fällt, wie auch bei allen anderen Tests, zunächst auf, daß sich der Fehler der „Mobile“-Testsequenzen unter dem Fehler der „Middlebury“-Testsequenzen befindet. Da ein Großteil der Fehler stets in Gebieten der Disparitätsdiskontinuitäten, also an Objekträndern, auftritt, spielt die Komplexität der Objekte in einer Szene dabei eine entscheidende Rolle. Die Ursache deren kleineren Fehlers liegt also einerseits in der meist einfacheren Geometrie realistischer Sequenzen, verglichen mit den komplexen Objek-

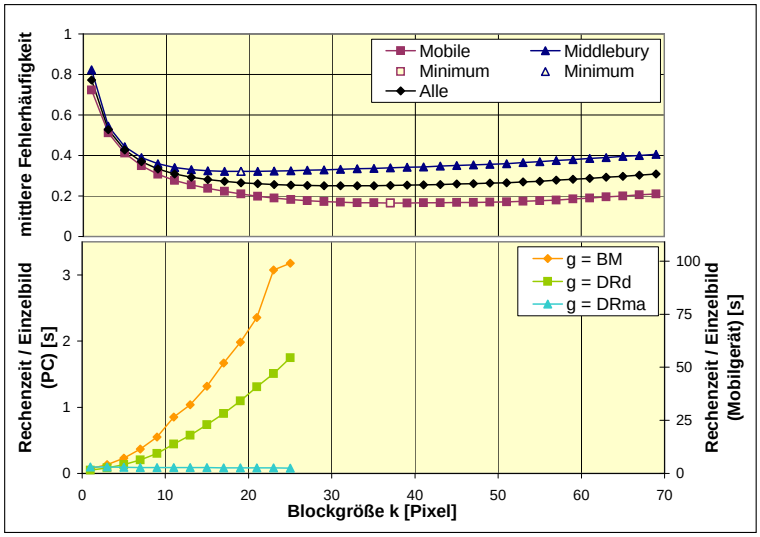


Abbildung 3.33.: Mittlere Fehlerhäufigkeit (oben) und Rechenzeit (unten) für die Testsequenzen aus Abschnitt 3.7.2 in Abhängigkeit von der Blockgröße k.

ten in den eigens zum Grund des Test einer Disparitätsschätzung hergestellten Testsequenzen. Andererseits sind die „Ground Truths“ der „Mobile“-Sequenzen aufgrund ihres Herstellungsverfahrens per Hand insbesondere bezüglich deren Tiefenauflösung nur sehr grob, was den verwendeten Algorithmen ebenfalls zugute kommt. Das sehr ähnliche Fehlerverhalten der Algorithmen bezüglich beider Testgruppen spricht dabei jedoch für die Qualität der „Mobile“-Sequenzen.
Wie bereits erwartet (vgl. Abbildung 3.18), ergibt sich bei variierter Blockgröße jeweils ein eindeutiges Minimum, das für die „Mobile“-Sequenzen bei der Blockkantenlänge $k = 39$ Pixel und für die „Middlebury-“ Sequenzen bei der Blockkantenlänge $k = 19$ Pixel liegt. Dieser Unterschied ist direkt auf die Abwesenheit von Störfaktoren wie Rauschen und Linsenverzeichnungen bei den „Middlebury-“ Sequenzen zurückzuführen. Wegen des für beide Fälle sehr geringen Fehlers aufgrund der Flachheit beider Kurven um deren Minima soll entsprechend im Folgenden eine Blockgröße von 25×25 Pixeln als optimal betrachtet und der Parameter k ausschließlich innerhalb $K = \{1, 3, 5, \ldots, 25\}$ variiert werden.
Bei der Rechenzeit zeigt sich das zu erwartende mit der Blockgröße fallende Verhalten für die beiden Verfahren mit blockbasierter Fehlerminimierung. Bei der Implementierung als Moving Average Filter kann ebenfalls das erwartete von der Blockgröße unabhängige Zeitverhalten beobachtet werden.

3.7.3.2. Blockabstand

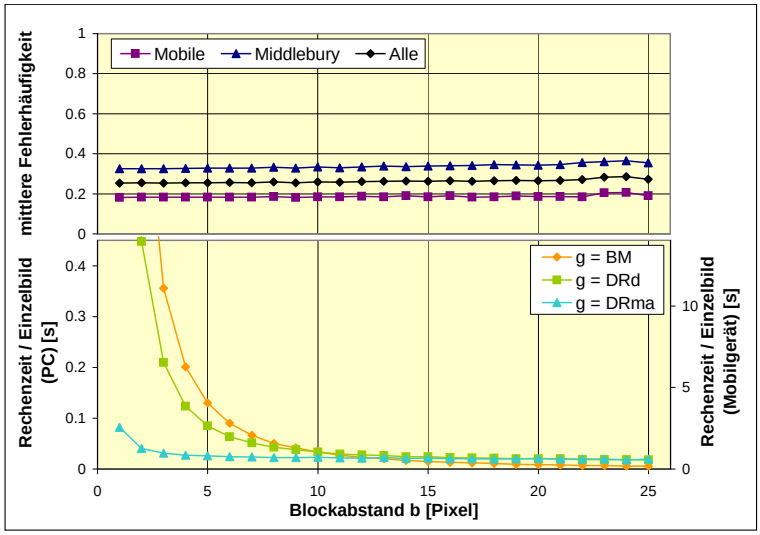


Abbildung 3.34.: Mittlere Fehlerhäufigkeit (oben) und Rechenzeit (unten) für die Testsequenzen aus Abschnitt 3.7.2 in Abhängigkeit vom Blockabstand b.

Abbildung 3.35 zeigt beispielhaft die berechneten Disparitätskarten der „bookshelf“-Sequenz für die verschiedenen Zustände des Blockabstandes b (s. Abschnitt 3.6.2.3). Die entsprechenden mittleren Fehlerhäufigkeiten über alle Testsequenzen mit den jeweiligen Rechenzeiten gehen aus Abbildung 3.34 hervor. Dabei fällt vor allem der flache Anstieg der mittleren Fehlerhäufigkeiten bei steigenden Blockabständen auf. Dieser wird von einem sehr stark abfallenden Zeitverhalten begleitet, das wiederum bei höheren Werten von b aufgrund dessen höherer Geschwindigkeit die Verwendung des Block Matchings rechtfertigt.

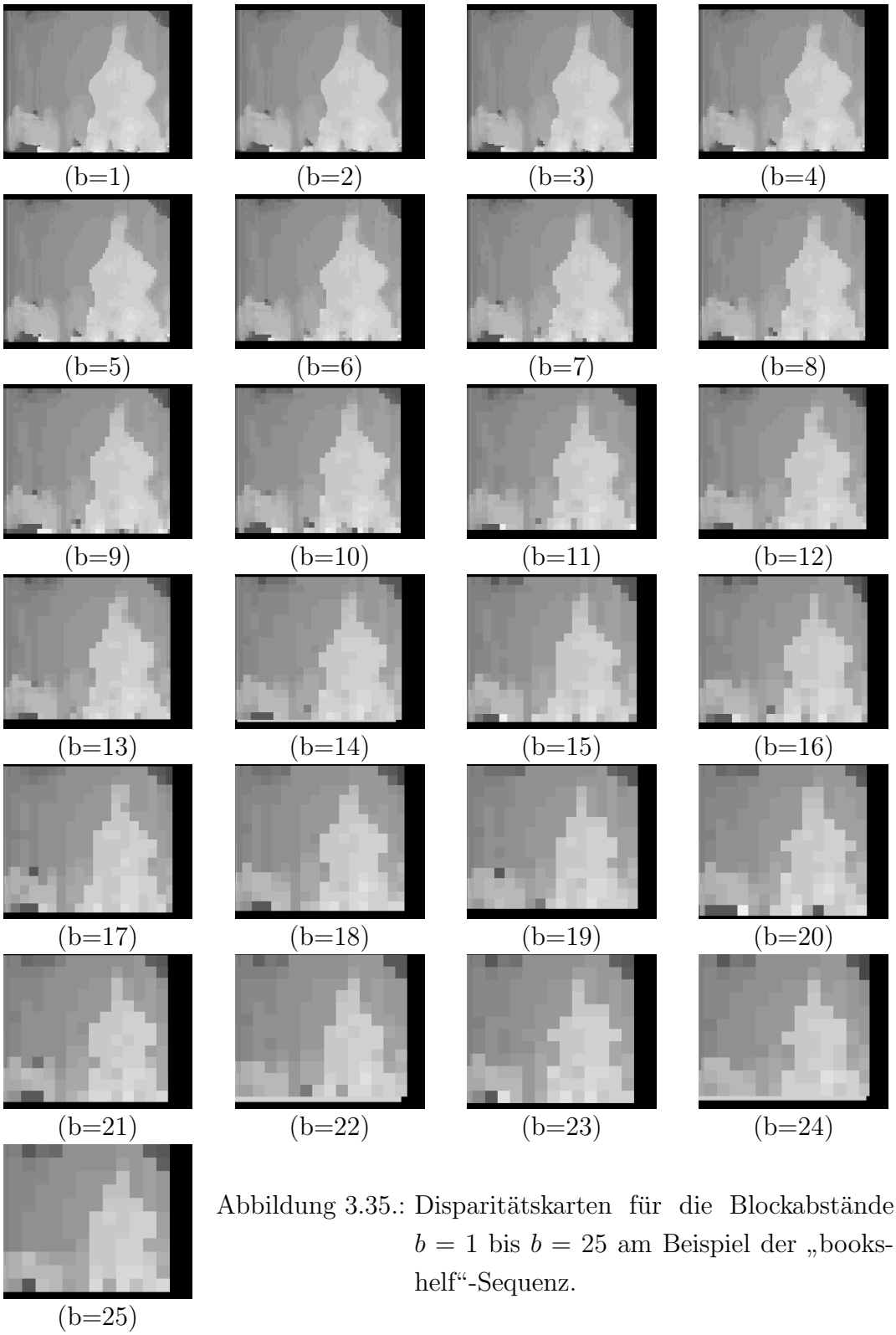


Abbildung 3.35.: Disparitätskarten für die Blockabstände $b = 1$ bis $b = 25$ am Beispiel der „bookshelf“-Sequenz.

3.7.3.3. Blockabtastperiode

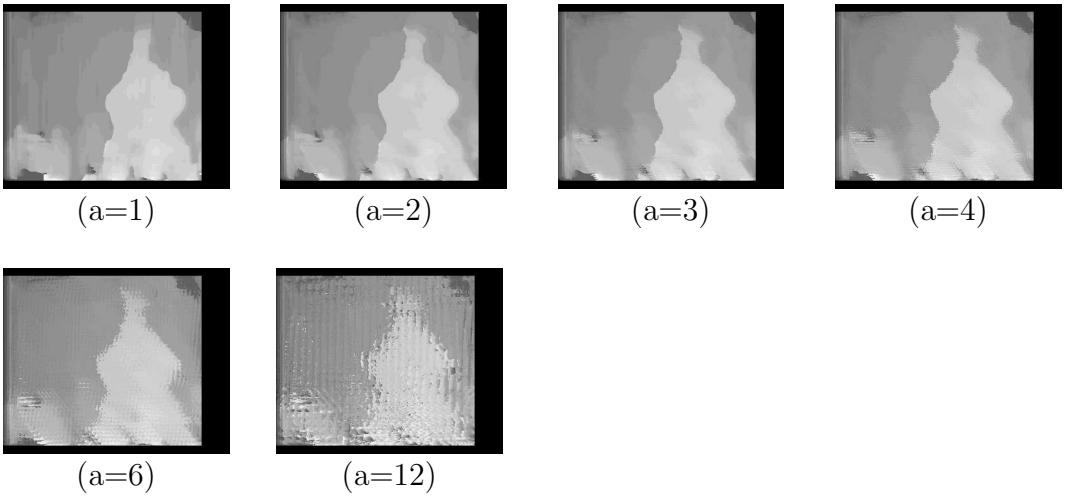


Abbildung 3.36.: Disparitätskarten der bei $k_{eff} = 25$ möglichen Blockabtastperioden am Beispiel der „bookshelf“-Sequenz.

Abbildung 3.36 zeigt beispielhaft die berechneten Disparitätskarten der „bookshelf“-Sequenz für die verschiedenen Zustände der Blockabtastperiode a (s. Abschnitt 3.6.2.4). Die entsprechenden mittleren Fehlerhäufigkeiten über alle Testsequenzen mit den jeweiligen Rechenzeiten gehen aus Abbildung 3.37 hervor. Bei einer effektiven Blockgröße von $k_{eff} = 25$ ergeben sich dabei lediglich die möglichen Zustände $a \in \{1, 2, 3, 4, 6, 12\}$ (vgl. Abschnitt 3.6.2.4). Da für die meisten Kombinationen von Blockabstand und effektiver Blockgröße, insbesondere für die Parameter in ihren Grundzuständen, keine Bildabtastperioden $u \neq 1$ möglich sind, soll hier von einer Darstellung abgesehen werden.

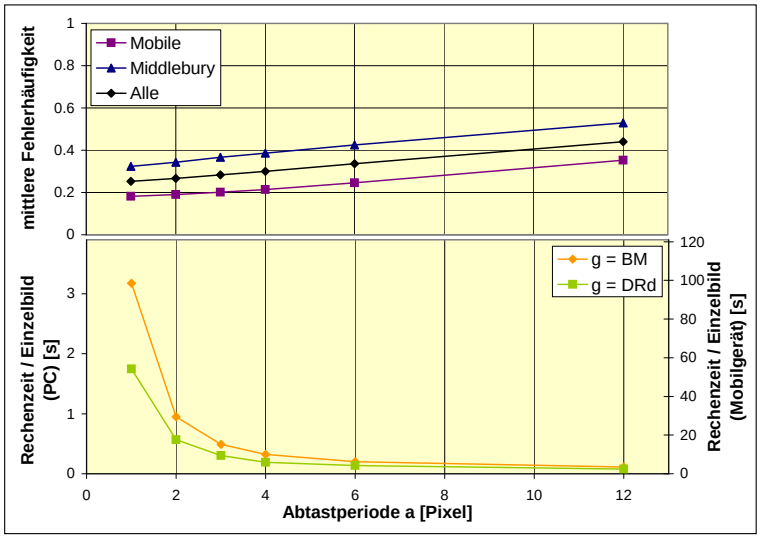
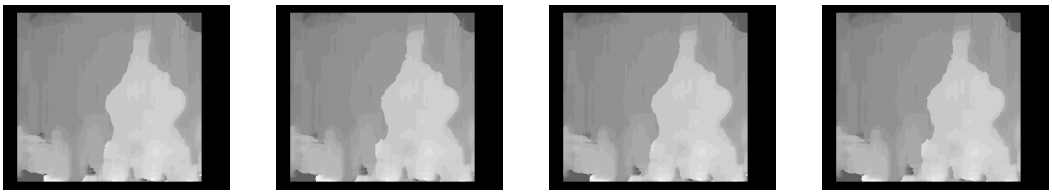


Abbildung 3.37.: Mittlere Fehlerhäufigkeit (oben) und Rechenzeit (unten) für die Testsequenzen aus Abschnitt 3.7.2 in Abhängigkeit von der Blockabtastperiode a.

3.7.3.4. Stufenanzahl der logarithmischen Suche



(l=1) (l=2) (l=3) (l=4)

Abbildung 3.38.: Disparitätskarten für $l = 1$ bis $l = 4$ logarithmische Suchschritte am Beispiel der „bookshelf“-Sequenz.

Abbildung 3.38 zeigt beispielhaft die berechneten Disparitätskarten der „bookshelf“-Sequenz für die verschiedenen Zustände der logarithmischen Stufenzahl l (s. Abschnitt 3.6.2.5). Die entsprechenden mittleren Fehlerhäufigkeiten über alle Testsequenzen mit den jeweiligen Rechenzeiten gehen aus Abbildung 3.39 hervor. Hierbei fällt vor allem beim Block Matching der hohe Geschwindigkeitszuwachs bei minimalen Qualitätseinbußen auf, selbst bei höheren Werten von l.

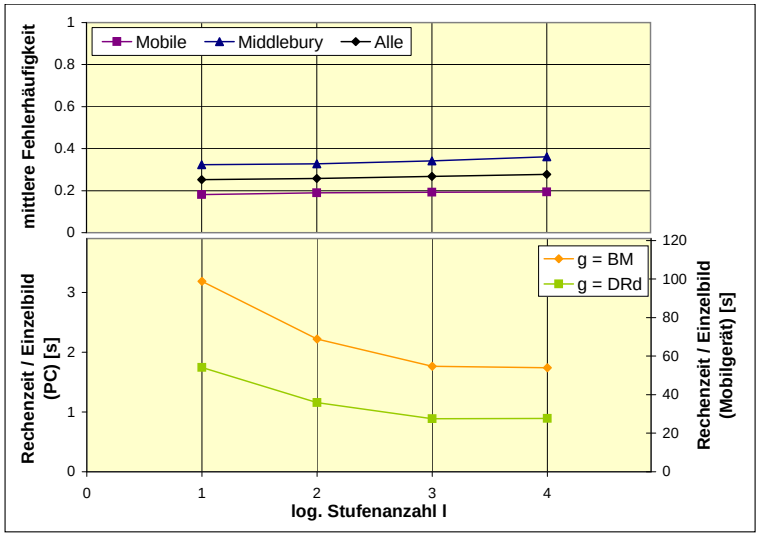


Abbildung 3.39.: Mittlere Fehlerhäufigkeit (oben) und Rechenzeit (unten) für die Testsequenzen aus Abschnitt 3.7.2 in Abhängigkeit von der Stufenanzahl l bei logarithmischer Abtastung des Suchvektors.

3.7.3.5. Peak Ratio

(p=0, f=0) (p=1, f=0) (p=1, f=1)

Abbildung 3.40.: Disparitätskarten mit und ohne Anwendung des Peak Ratio (p) und der Fehlstellenergänzung (f) am Beispiel der „bookshelf“-Sequenz.

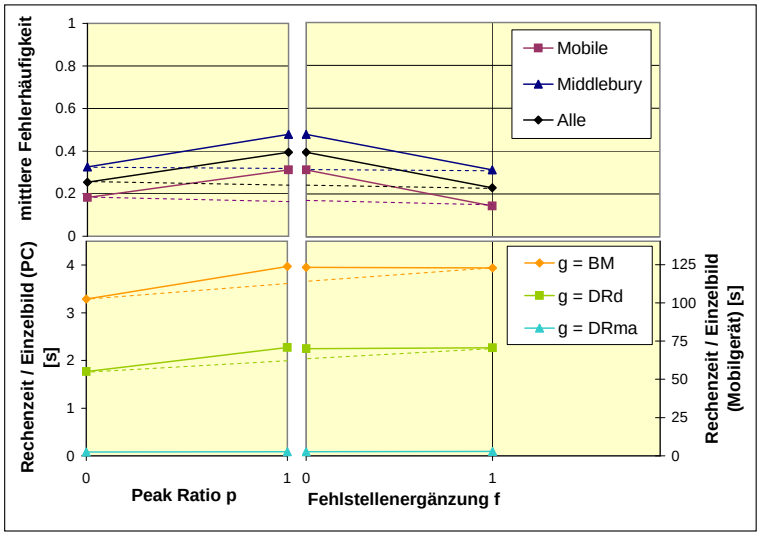
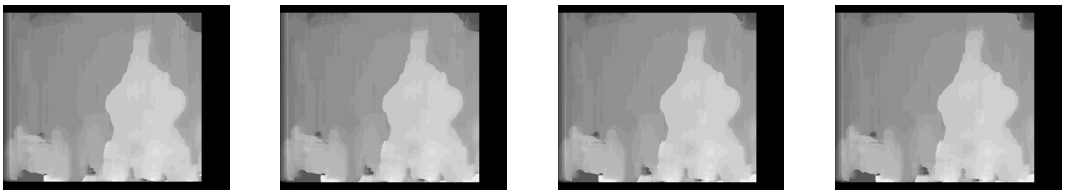


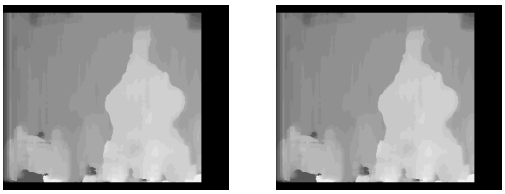
Abbildung 3.41.: Mittlere Fehlerhäufigkeit (oben) und Rechenzeit (unten) für die Testsequenzen aus Abschnitt 3.7.2 ohne ($p = 0$) und mit ($p = 1$) Verwendung des Peak Ratio als Zuverlässigkeitswert ohne ($f = 0$) und mit ($f = 1$) Fehlstellenergänzung.

Abbildung 3.40 zeigt beispielhaft die berechneten Disparitätskarten der „bookshelf“-Sequenz für die verschiedenen Zustände des Peak Ratios r (s. Abschnitt 3.6.2.8) mit und ohne zusätzlicher Fehlstellenergänzung f (s. Abschnitt 3.6.2.9). Hier zeigt sich, wie aufgrund der Bewertung unzuverlässiger Werte als Fehler die Fehlerrate bei Verwendung dieses Zuverlässigkeitskriteriums zunächst ansteigt. Erst ein Hinzukommen der Fehlstellenergänzung f bewirkt schließlich ein Abfallen unterhalb der Ursprungswerte, sodaß im Folgenden der Skalierungsparameter p ausschließlich mit nachfolgender Fehlstellenergänzung verwendet werden soll.

3.7.3.6. Luminanzkorrektur



(c=0) (c=1) (c=2) (c=3)



(c=4) (c=5)

Abbildung 3.42.: Disparitätskarten für die Luminanzkorrekturwerte $c = 0$ bis $c = 5$ am Beispiel der „bookshelf“-Sequenz.

Abbildung 3.42 zeigt beispielhaft die berechneten Disparitätskarten der „bookshelf“-Sequenz für die verschiedenen Zustände der Luminanzkorrektur c (s. Abschnitt 3.6.2.7). Die entsprechenden mittleren Fehlerhäufigkeiten über alle Testsequenzen mit den jeweiligen Rechenzeiten gehen aus Abbildung 3.43 hervor. Bei kleinen Luminanzkorrekturwerten zeigt sich der erwartete qualitätsverbessernde Effekt, der jedoch sehr gering ist im Vergleich zur benötigten Rechenleistung.

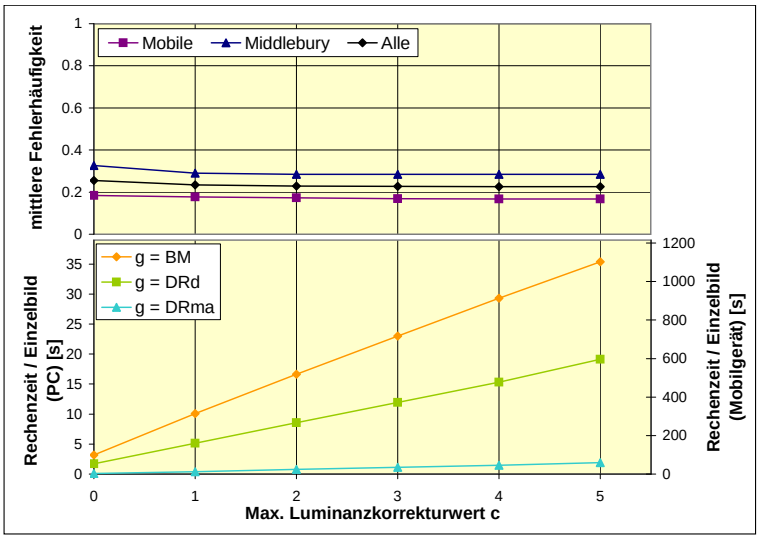


Abbildung 3.43.: Fehlerwahrscheinlichkeit (oben) und Rechenzeit (unten) für die Testsequenzen aus Abschnitt 3.7.2 in Abhängigkeit von der Luminanzkorrektur c.

3.7.3.7. Schätzbereichsbreite

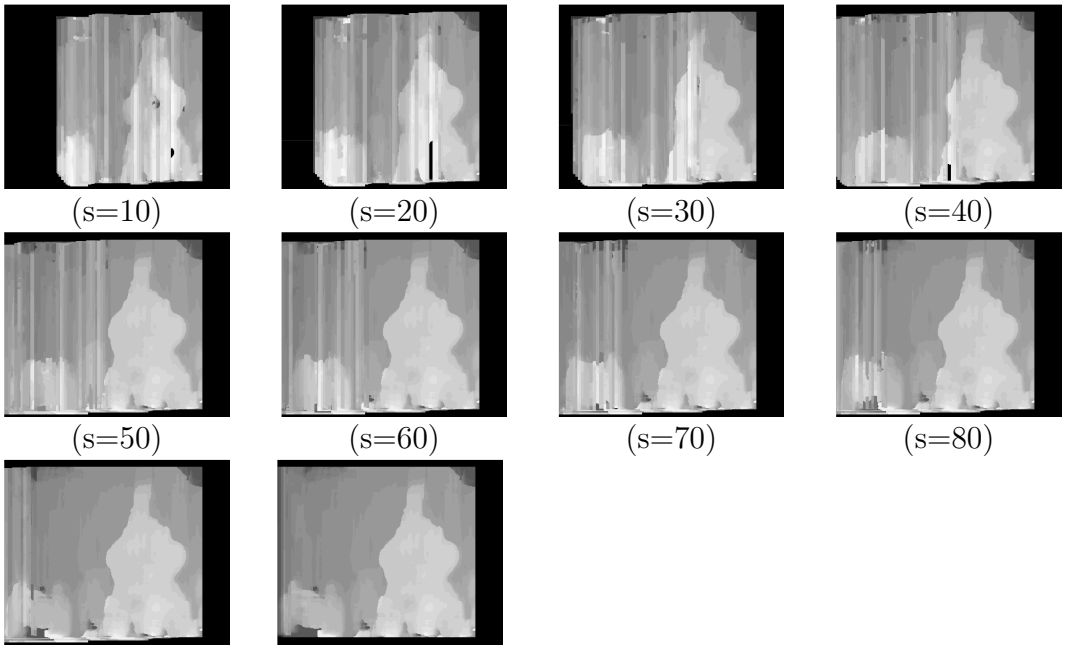


Abbildung 3.44.: Disparitätskarten für verschiedene prozentuale Schätzbereichsbreiten am Beispiel der „bookshelf“-Sequenz.

Abbildung 3.44 zeigt beispielhaft die berechneten Disparitätskarten der „bookshelf“-Sequenz für verschiedene Zustände der prozentualen Schätzbereichsbreite s (s. Abschnitt 3.6.2.6). Die entsprechenden mittleren Fehlerhäufigkeiten über alle Testsequenzen mit den jeweiligen Rechenzeiten gehen aus Abbildung 3.45 hervor. Dabei kann bei fallenden Werten von s ein Auseinanderdriften der beiden mittleren Fehlerhäufigkeiten für die „Middlebury“- und „Mobile“-Testsequenzen beobachtet werden: Da es sich bei den „Mobile“-Sequenzen um Videosequenzen aus mehreren Einzelbildern handelt, können bei partieller Berechnung eines Disparitätsbildes im fehlenden Teil Werte aus den vorangegangenen Bildern übernommen werden. Dies ist bei den jeweils nur aus einem Bildpaar bestehenden „Middlebury“-Sequenzen nicht möglich.

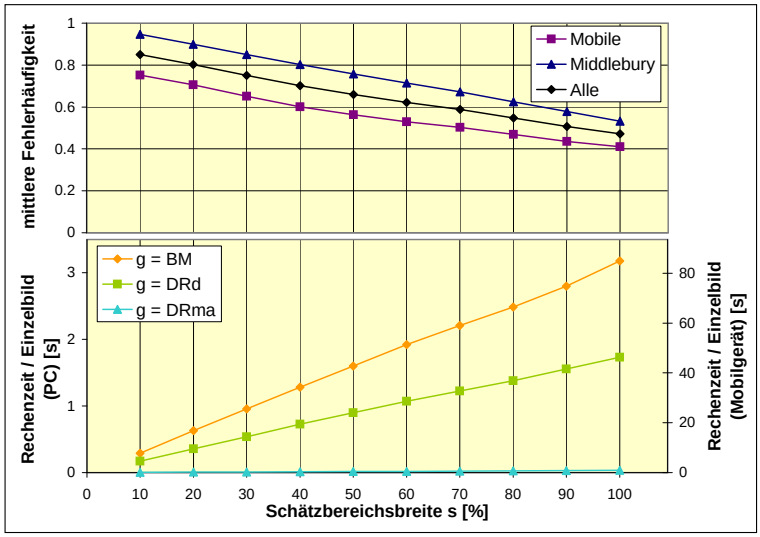


Abbildung 3.45.: Fehlerwahrscheinlichkeit (oben) und Rechenzeit (unten) für die Testsequenzen aus Abschnitt 3.7.2 in Abhängigkeit von der Schätzbereichsbreite s.

3.7.4. Einfluß des Fehlerkriteriums

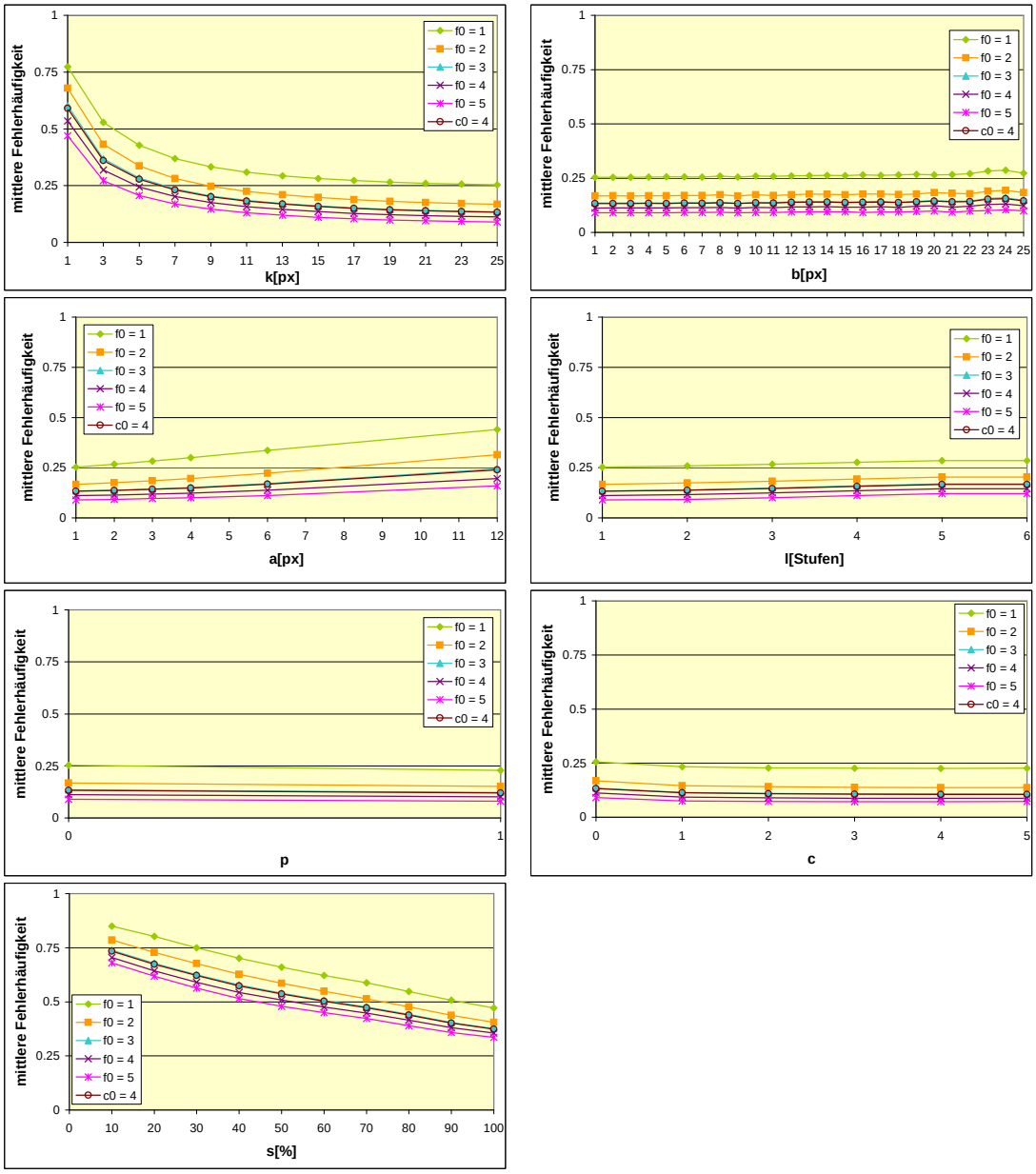


Abbildung 3.46.: Fehlerverhalten der verschiedenen Skalierungsparameter bei Verwendung unterschiedlicher Fehlerkriterien nach den Gleichungen 3.73 (f_0) und 3.74 (c_0).

Abbildung 3.46 zeigt den Einfluß der verschiedenen in den Abschnitt 3.7.2.1 vorgestellten Kriterien zur Bewertung eines berechneten Disparitätswertes als Fehler auf das Fehlerverhalten der verschiedenen Skalierungsparameter. Hierbei fällt

zunächts die näherungsweise Parallelverschiebung und damit die hohe Ähnlichkeit der Fehlerkuven für die verschiedenen Fehlerkriterien auf. Bei einer relativen Betrachtung der mittleren Fehlerhäufigkeiten eines einzelnen Skalierungsparameters sind somit kaum Veränderungen zu erwarten. Ein Vergleich aller Fehlerkurven zeigt dabei ein identisches Verhalten sämtlicher Skalierungsparameter bezüglich einer Änderung des Fehlerkriteriums. Da auch für das weiche Fehlerkriterium mit $c_0 = 4$ ausschließlich Ergebnisse erziehlt werden, die dem des harten Fehlerkriteriums $f_0 = 3$ gleichen, ist für einen Wechsel des Fehlerkriteriums keinerlei signifikante Änderung des Ergebnisses zu erwarten, weshalb auch im Weiteren darauf verzichtet und ausschließlich der Fall $f_0 = 1$ verwendet werden soll.

3.7.5. Fehlermodellierung

Da die Vielzahl möglicher Parameterkombinationen (insgesamt 11176800) ein vollständiges Testen erschwert, soll in diesem Abschnitt zunächst ein Modell für das Fehlerverhalten gefunden werden, sodaß aus einem reduzierten Satz von Messungen die restlichen Werte berechnet werden können. Hierdurch wird auch ein nachträgliches Hinzufügen zusätzlicher Skalierungsparameter stark erleichtert.
Die Kombination der Zustände der einzelnen Skalierungsparameter kann zu einem Zustandsvektor $\mathbf{z}$ zusammengefaßt werden:

$$\mathbf{z}_i = (g \in G, k \in K, b \in B, a \in A, l \in L, s \in S, c \in C, p \in P)^T \quad (3.76)$$

Als unskalierter Fall soll dabei die Kombination der Grundparameter (vgl. Tabelle 3.1) angesehen werden, sodaß sich für den zugehörigen Grundzustandsvektor $\mathbf{z}_0$ ergibt:

$$\begin{aligned} \mathbf{z}_0 &= (g = g_0, k = k_0, b = b_0, a = a_0, l = l_0, s = s_0, c = c_0, p = p_0)^T \\ &= (DRma, 25, 1, 1, 1, 100, 0, 0)^T \end{aligned} \quad (3.77)$$

Das Ereignis „Aufgrund Skalierungsparameter x ist ein Fehler aufgetreten“ soll im Folgenden beschrieben werden als:

$$Q(x). \quad (3.78)$$

Die Gesamtwahrscheinlichkeit P, daß bei einer bestimmten Parameterkombination $\mathbf{z}$ irgendein Fehler auftritt, entspricht somit einer „ODER“-Verknüpfung :

$$P\big(Q(\mathbf{z})\big) = P\Big(Q(g) \cup Q(k) \cup Q(b) \cup Q(a) \cup Q(l) \cup Q(p) \cup Q(c) \cup Q(s)\Big). \quad (3.79)$$

Bei statistischer Unabhängigkeit der einzelnen Skalierungsparameter kann Gleichung 3.79 mit der vereinfachten Schreibweise $P(Q(x)) =: P(x)$ umgeformt werden in:

$$P(\mathbf{z}) = P(g) \cup P(k) \cup P(b) \cup P(a) \cup P(l) \cup P(p) \cup P(c) \cup P(s). \quad (3.80)$$

Ein entsprechender Nachweis der statistischen Unabhängigkeit wird später in diesem Kapitel geführt.
Da die Kombination aller Parametergrundeinstellungen $\mathbf{z}_0$ als unskalierter Fall

angesehen wird, soll deren Fehlerwahrscheinlichkeit der des Grundalgorithmus entsprechen. Diese ist für alle dessen Parameterzustände identisch, da es sich bei den Grundalgorithmen um unterschiedliche Implementierungen derselben Funktionalität handelt, die sich ausschließlich hinsichtlich der benötigten Rechenzeit unterscheiden:

$$P\Big(\mathbf{z}_0\Big) = P\Big(g\Big) =: P_0. \tag{3.81}$$

Daraus ergibt sich mit Gleichung 3.80 für die Einzelparameter-Fehlerwahrscheinlichkeiten des Grundzustandes x_0 eines von g verschiedenen Parameters x:

$$P(x_0) = 0. \tag{3.82}$$

Für die möglichen Zustände $x_i \in X = \{x_0, x_1, x_2, \ldots\}$ eines Skalierungsparameters x kann weiterhin der Vektor eingeführt werden:

$$\mathbf{x} = (x_0, x_1, x_2, \ldots)^T. \tag{3.83}$$

Der entsprechende Vektor $\mathbf{p}(\mathbf{x})$ der zugehörigen Einzelparameter-Fehlerwahrscheinlichkeiten ergibt sich dann zu:

$$\mathbf{p}(\mathbf{x}) = \big(P(x_0), P(x_1), P(x_2), \ldots\big)^T. \tag{3.84}$$

In Abschnitt 3.7.3 wurden auf Grundlage der in Abschnitt 3.7.2 vorgestellten Testsequenzen die Verbundfehlerwahrscheinlichkeitsvektoren $\hat{\mathbf{p}}(\mathbf{x})$ der einzelnen Skalierungsparameter x anhand der gemessenen mittleren Fehlerhäufigkeiten geschätzt:

$$\hat{\mathbf{p}}(\mathbf{x}) = (P_0 \cup P(x_0), P_0 \cup P(x_1), P_0 \cup P(x_2), \ldots). \tag{3.85}$$

Die Grundfehlerwahrscheinlichkeit P_0 kann aus dem Element abgelesen werden, das ihrer Verknüpfung mit der Fehlerwahrscheinlichkeit des Grundzustandes x_0 entspricht, da nach Gleichung 3.82 gilt:

$$P_0 \cup P(x_0) = P_0 \cup 0 = P_0. \tag{3.86}$$

Somit kann der Vektor der Verbundfehlerwahrscheinlichkeiten $\hat{\mathbf{p}}(\mathbf{x})$ für jedes Element $\hat{p}_i$ in den Vektor der Einzelparameter-Fehlerwahrscheinlichkeiten $\mathbf{p}(\mathbf{x})$ umgerechnet werden:

$$\begin{aligned} \hat{p}_i(\mathbf{x}) &= P_0 \cup P(x_i) \\ &= P_0 + P(x_i) - P_0 P(x_i) \\ \Rightarrow P(x_i) &= \frac{\hat{p}_i(\mathbf{x}) - P_0}{1 - P_0} = p_i(\mathbf{x}). \end{aligned} \tag{3.87}$$

Die ausgehend von den in Abschnitt 3.7.3 gemessenen Gesamtfehlerwahrscheinlichkeitsvektoren berechneten Einzelparameter-Fehlerwahrscheinlichkeitsvektoren sind in Abbildung 3.47 dargestellt.
Skalierungsparameter können grundsätzlich in zwei Gruppen aufgeteilt werden: Einerseits in Parameter, die ausgehend vom Grundzustand die Fehlerwahrscheinlichkeit zu Gunsten einer höheren Geschwindigkeit erhöhen. Andererseits in solche, die auf kosten der Geschwindigkeit die Fehlerwahrscheinlichkeit senken (Parameter „c“ und „p“). Bei einem der zweiteren Gruppe angehörigen, fehlerkorrigierenden Skalierungsparameter x soll das Ereignis „Aufgrund Skalierungsparameter x wurde ein Fehler korrigiert“ im Folgenden beschrieben werden als:

$$R(x). \tag{3.88}$$

Für die Gesamtfehlerwahrscheinlichkeit ergibt sich daraus:

$$P\left(\mathbf{z}_i\right) = P\big(Q(g) \cap \overline{R(x)}\big). \tag{3.89}$$

Statistische Unabhängigkeit vorausgesetzt (s.u.), gilt mit Gleichung 3.81 somit:

$$P\left(\mathbf{z}_i\right) = P_0\Big(1 - P\big(R(x)\big)\Big). \tag{3.90}$$

Da es für die Systematik des weiteren Vorgehens jedoch zweckmäßig ist, alle Fehlerwahrscheinlichkeiten mit der gleichen Operation („ODER“) zu verknüpfen, soll für die entsprechende Fehlerwahrscheinlichkeit des Ereignisses $Q(x)$ („Aufgrund x ist ein Fehler aufgetreten“) gelten:

$$\begin{aligned} P_0 \cap P\big(\overline{R(x)}\big) &= P_0\Big(1 - P\big(R(x)\big)\Big) \\ = P_0 \cup P\left(Q(x)\right) &= P_0 + P\left(Q(x)\right) - P_0 P\left(Q(x)\right) \\ &\Rightarrow \; P\left(Q(x)\right) = \frac{P_0}{P_0 - 1} P\big(R(x)\big). \end{aligned} \tag{3.91}$$

Die für die Skalierungsparameter „p“ und „c“ in Abbildung 3.47 bezüglich einer „ODER“-Verknüpfung negativ erscheinenden Einzelparameter-Fehlerwahrscheinlichkeitswerte können entsprechend in positive „Fehlerkorrekturwahrscheinlichkeitswerte“ umgerechnet werden. [1]

[1] In Tabelle 3.1 bzw. Abschnitt 3.7.5 wurde, motiviert durch die zugrundeliegende Grundfunktionalität der Blockbasierten Stereokorrespondenz, der „Grundzustand“ aller Skalierungsparameterkombinationen definiert. Aus mathematischer Sicht ist dieser Grundzustand jedoch willkürlich, sodaß auch die Skalierungsparameterkombination mit der geringsten Fehlerwahrscheinlichkeit als Referenzpunkt gewählt werden könnte. In diesem Fall würden sich für alle anderen Skalierungsparameterzustände positive Einzelparameter-Fehlerwahrscheinlichkeiten ergeben („Fehler aufgrund Wegfallen des Korrekturverfahrens“).

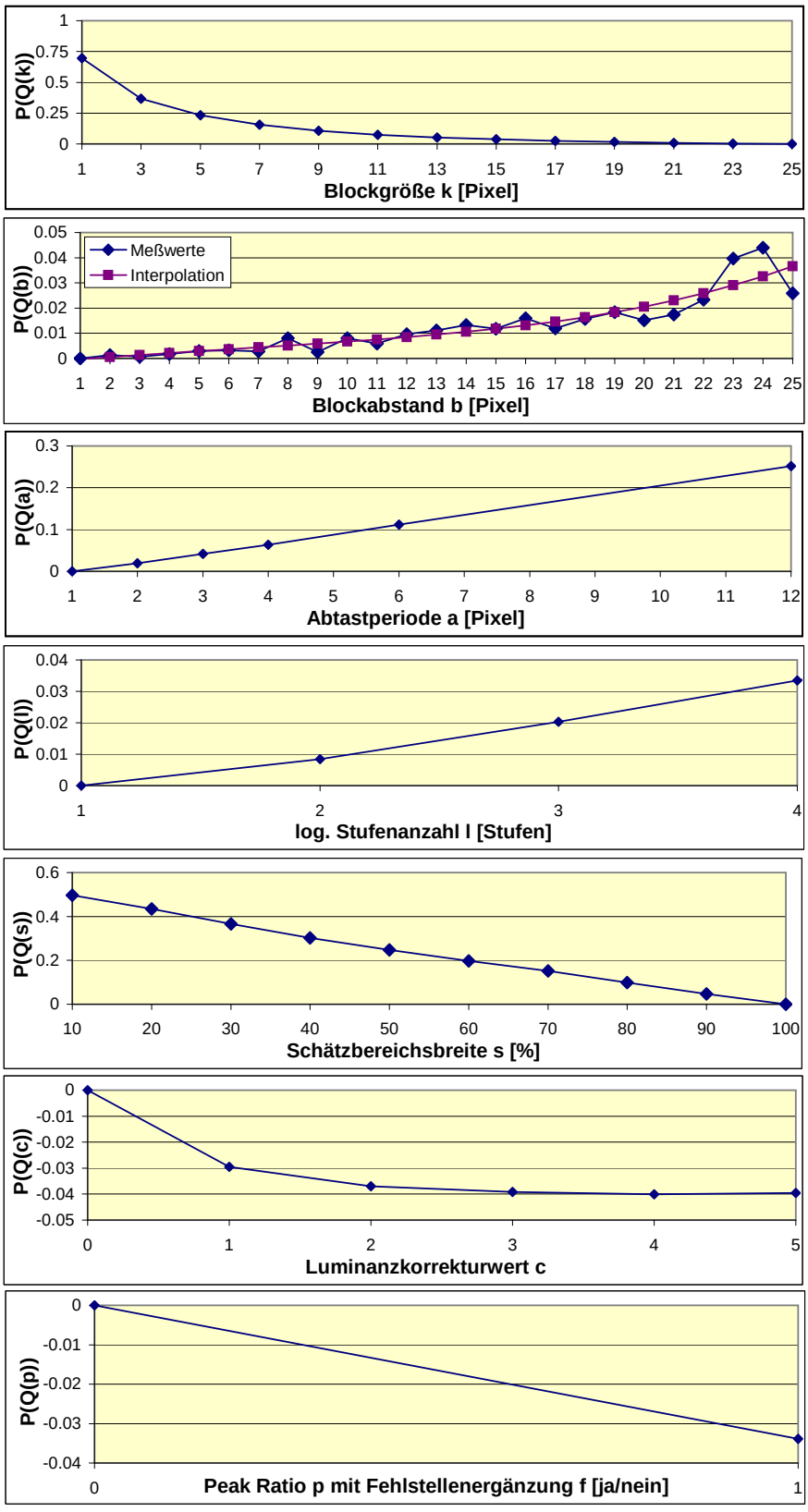


Abbildung 3.47.: Aus den Testergebnissen aus Abschnitt 3.7.3 über Gleichung 3.87 berechnete Einzelparameter-Fehlerwahrscheinlichkeitsvektoren $\hat{\mathbf{p}}$ der Skalierungsparameter.

In Abschnitt A.3.1 (Mitte) wurden dementsprechend für alle paarweisen Parameterkombinationen $x \in X = \{x_0, x_1, x_2, \ldots\}$ und $y \in Y = \{y_0, y_1, y_2, \ldots\}$ die entsprechenden Verbundfehlerwahrscheinlichkeiten berechnet, wie sie in Abbildung 3.48 beispielhaft für die Parameter $k \in K = \{1, 3, 5, \ldots, 25\}$ und $b \in B = \{1, 2, 3, \ldots, 25\}$ dargestellt sind:

$$\hat{\mathbf{P}}_{berechnet}(x, y) := P_0 \cup \mathbf{p}(\mathbf{x}) \cup \mathbf{p}^T(\mathbf{y}) \tag{3.92}$$

$$= P_0 \cup \begin{pmatrix} P(x_0) \\ P(x_1) \\ P(x_2) \\ \vdots \end{pmatrix} \cup \big(P(y_0), P(y_1), P(y_2), \ldots\big)$$

$$= \begin{bmatrix} P_0 \cup P(x_0) \cup P(y_0) & P_0 \cup P(x_1) \cup P(y_0) & \ldots \\ P_0 \cup P(x_0) \cup P(y_1) & P_0 \cup P(x_1) \cup P(y_1) & \ldots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

$$= \begin{bmatrix} P_0 + P(x_0) + P(y_0) & P_0 + P(x_1) + P(y_0) & \\ -P(x_0)P(y_0) - P_0\big(P(x_0) & -P(x_1)P(y_0) - P_0\big(P(x_1) & \ldots \\ +P(y_0) - P(x_0)P(y_0)\big) & +P(y_0) - P(x_1)P(y_0)\big) & \\ & & \\ P_0 + P(x_0) + P(y_1) & P_0 + P(x_1) + P(y_1) & \\ -P(x_0)P(y_1) - P_0\big(P(x_0) & -P(x_1)P(y_1) - P_0\big(P(x_1) & \ldots \\ +P(y_1) - P(x_0)P(y_1)\big) & +P(y_1) - P(x_1)P(y_1)\big) & \\ \vdots & \vdots & \ddots \end{bmatrix}$$

$$= \mathbf{1}\mathbf{p}^T(\mathbf{x}) + \mathbf{p}(\mathbf{y})\mathbf{1}^T - \mathbf{p}(\mathbf{x})\mathbf{p}^T(\mathbf{y}) + P_0$$

$$- \quad (\mathbf{1}\mathbf{p}^T(\mathbf{x}) + \mathbf{p}(\mathbf{y})\mathbf{1}^T - \mathbf{p}(\mathbf{x})\mathbf{p}^T(\mathbf{y}))P_0, \qquad \text{mit } \mathbf{1} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Dieselben Kombinationen wurden außerdem direkt gemessen (Abbildung 3.48 links, bzw. Abschnitt A.3.1 links):

$$\hat{\mathbf{P}}_{gemessen}(x, y) := \begin{bmatrix} P\big(Q(g) \cup Q(x_0) \cup Q(y_0)\big) & P\big(Q(g) \cup Q(x_1) \cup Q(y_0)\big) & \ldots \\ P\big(Q(g) \cup Q(x_0) \cup Q(y_1)\big) & P\big(Q(g) \cup Q(x_1) \cup Q(y_1)\big) & \ldots \\ \vdots & \vdots & \ddots \end{bmatrix}. \tag{3.93}$$

Aus beiden Verteilungen für das entsprechende Parameterpaar kann schließlich die Verteilung des relativen Fehlers berechnet werden (Abbildung 3.48 rechts, bzw. A.3.1 rechts):

$$\mathbf{F}_{rel} = 100\% \frac{\hat{\mathbf{P}}_{gemessen}(x,y) - \hat{\mathbf{P}}_{berechnet}(x,y)}{\hat{\mathbf{P}}_{gemessen}(x,y)}. \tag{3.94}$$

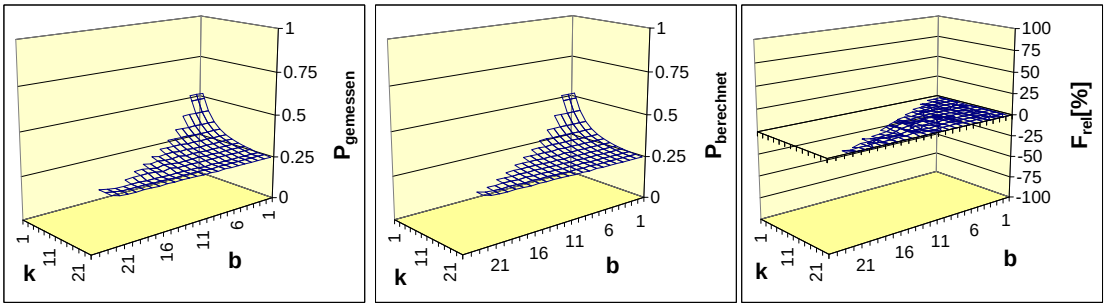


Abbildung 3.48.: (Links) Direkt gemessene mittlere Verbundfehlerhäufigkeiten $P_{gemessen} = P(Q(k) \cup Q(b))$ der Parameter k und b (vgl. Gleichung 3.79). (Mitte) Über Verknüpfung der für die Einzelparameter bestimmten mittleren Fehlerhäufigkeiten berechnete Verbundfehlerhäufigkeit $P_{berechnet} = P(Q(k)) \cup P(Q(b))$ (vgl. Abbildung 3.47 und Gleichung 3.80). (Rechts) Relativer Fehler F_{rel} beider Verteilungen nach Gleichung 3.94.
Vgl. dazu auch die übrigen paarweisen Parameterkombinationen in Abschnitt A.3.1.

Hier stellt sich zunächst die Frage nach einem akzeptierbaren relativen Fehler, sodaß von einer statistischen Unabhängigkeit ausgegangen werden kann, insbesondere da bereits zwischen tatsächlicher Fehlerwahrscheinlichkeit und der als deren Näherung verwendeten gemessenen mittleren Fehlerhäufigkeit ein gewisser nicht zu vernachlässigender Fehler anzunehmen ist. Die Herleitung der effektiven Blockgröße k_{eff} in Gleichung 3.21 zur Einschränkung der Zustandsmenge von a fand zur Entkoppelung beider Parameter statt, sodaß bei einer vollen Zustandsmenge $a' \in \{1, 2, 3, \ldots, 12\}$ von einer fehlenden statistischen Unabhängigkeit ausgegangen werden kann. Abbildung 3.49 zeigt entsprechend den Verlauf des relativen Fehlers bezüglich der Verknüpfung der beiden statistisch abhängigen Parameter a' und k. In Abbildung 3.48 und Abschnitt A.3.1 zeigt sich im Vergleich aus-

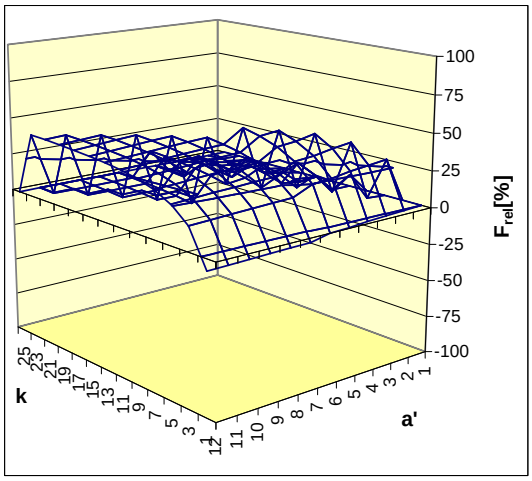


Abbildung 3.49.: Fehlerverteilung der beiden nicht statistisch unabhängigen Skalierungsparameter k und a'.

nahmslos eine gleichförmig niedrige Fehlerverteilung. Außerdem kann die eindeutige augenscheinliche Ähnlichkeit der gemessenen und berechneten Verteilungen als signifikantes Indiz für eine Korrektheit der Annahme einer statistischen Unabhängigkeit der Skalierungsparameter gewertet werden.

3.7.6. Zeitmodellierung

3.7.6.1. Einfluß des Grundalgorithmus

Neben einem Modell für den zu erwartenden Fehler einer bestimmten Parametereinstellung (vgl. Abschnitt 3.7.5) ist für die Bewertung deren Effizienz ein Modell des zu erwartenden Aufwands von zentraler Bedeutung. Die für die Disparitätsberechnung bei einer bestimmten Parameterkonstellation $\mathbf{z}_i$ (vgl. Gleichung 3.76) benötigte Rechenzeit soll dargestellt werden als:

$$T(\mathbf{z_i}). \tag{3.95}$$

Es gilt zu beachten, daß bei Betrachtung der Rechenzeit, ungleich dem Fehler, zwischen den drei Grundalgorithmen unterschieden werden muß. Da es sich bei diesen um verschiedene Implementierungen derselben Grundfunktionalität handelt, die auf die gleichen Skalierungsparameter jedoch völlig unterschiedlich reagieren (vgl. Abschnitt 3.7.3), müssen die Zeitvektoren für die Grundalgorithmen separat betrachtet werden. Die Rechenzeiten sollen daher grundsätzlich nach dem zugrundeliegenden Algorithmus unterschieden werden:

$$T\left(g,k,b,a,l,s,c,p\right) =: T^{g\in G}\left(k,b,a,l,s,c,p\right). \tag{3.96}$$

Analog zu Gleichung 3.81 soll ferner die Rechenzeit der Kombination aller Grundzustände der Parameter als Rechenzeit des Grundalgorithmus betrachtet werden:

$$\begin{aligned} T^{DRma}(\mathbf{z}_0) &= T_0^{DRma}, \\ T^{DRd}(\mathbf{z}_0) &= T_0^{DRd}, \\ T^{BM}(\mathbf{z}_0) &= T_0^{BM}. \end{aligned} \tag{3.97}$$

Ziel dieses Abschnitts ist es, analog zu Abschnitt 3.7.5 aus wenigen gemessenen Rechenzeiten die sehr große Menge der Rechenzeiten aller möglichen Parameterkombinationen zu berechnen. Wo man beim Fehler auf gemessene „a priori“-Erwartungswerte angewiesen ist, bietet sich beim Zeitmodell zusätzlich die Möglichkeit einer expliziten mathematischen Modellierung des verwendeten Algorithmus und der entsprechenden Skalierungsparameter. So gilt es zunächst, beide Ansätze zu vergleichen und den geeigneteren zu ermitteln.

3.7.6.2. Implizite Zeitmodellierung

Bei einer impliziten Zeitmodellierung soll von einer multiplikativen Verknüpfung der Rechenzeiten der Einzelparameter ausgegangen werden:

$$T^g(\mathbf{z}_i) = T_0^g T^g(k) T^g(b) T^g(a) T^g(l) T^g(s) T^g(c) T^g(p). \tag{3.98}$$

Für den Grundzeitfaktor $T^g(x_0)$ eines von g verschiedenen Parameters x in dessen Grundzustand x_0 ergibt sich nach Gleichung 3.97 somit analog zu Gleichung 3.82:

$$T^g(x_0) = 1. \tag{3.99}$$

Der Vektor der Zeitfaktoren für die möglichen Zustände $\mathbf{x}$ eines Skalierungsparameters x (vgl. Gleichung 3.83) kann analog zu Gleichung 3.84 aufgestellt werden:

$$\mathbf{t}^g(\mathbf{x}) = (T^g(x_0), T^g(x_1), T^g(x_2), \ldots)^T. \tag{3.100}$$

In Abschnitt 3.7.3 wurden auf Grundlage der in Abschnitt 3.7.2 vorgestellten Testsequenzen die absoluten Zeitvektoren $\hat{\mathbf{t}}_x$ der einzelnen Skalierungsparameter x gemessen.

$$\hat{\mathbf{t}}^g(\mathbf{x}) = (T_0^g T^g(x_0), T_0^g T^g(x_1), T_0^g T^g(x_2), \ldots)^T. \tag{3.101}$$

Die Grundrechenzeit T_0^g kann analog zu Gleichung 3.86 aus dem Element abgelesen werden, das ihrer Verknüpfung mit der Rechenzeit des Grundzustandes x_0 entspricht, da nach Gleichung 3.99 gilt:

$$T_0^g T^g(x_0) = T_0^g 1 = T_0^g. \tag{3.102}$$

Somit kann der Vektor der absoluten Rechenzeiten $\hat{\mathbf{t}}^g(\mathbf{x})$ für jedes Element $\hat{t}_i^g(\mathbf{x})$ in den Vektor $\mathbf{t}^g(\mathbf{x})$ der relativen Zeitfaktoren umgerechnet werden (vgl. Gleichung 3.87):

$$\begin{aligned} \hat{t}_i^g(\mathbf{x}) &= T_0^g T^g(x_i) \\ \Rightarrow T^g(x_i) &= \frac{\hat{t}^g(\mathbf{x})}{T_0^g} = t_i^g(\mathbf{x}). \end{aligned} \tag{3.103}$$

Die ausgehend von den in Abschnitt 3.7.3 gemessenen absoluten Rechenzeitvektoren berechneten relativen Rechenzeitvektoren sind in Abbildung 3.50 dargestellt.

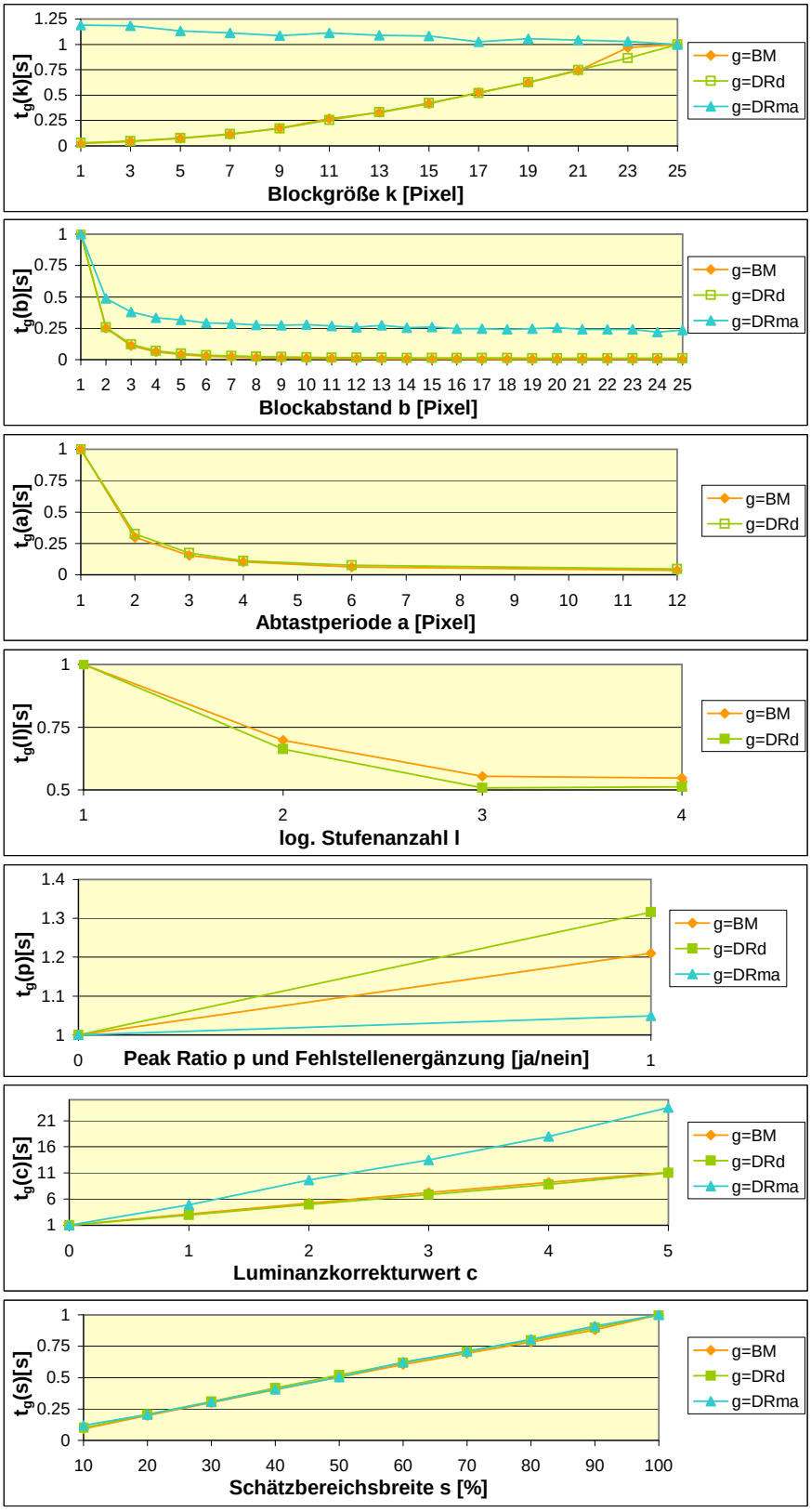


Abbildung 3.50.: Aus den Testergebnissen aus Abschnitt 3.7.3 über Gleichung 3.103 berechnete relative Zeitvektoren $\hat{\mathbf{t}}$ der Skalierungsparameter.

3.7.6.3. Explizite Zeitmodellierung

Bei expliziter Zeitmodellierung soll anhand einer die Struktur des jeweiligen Grundalgorithmus wiederspiegelnden Formel auf den zur Berechnung eines Disparitätsbildes nötigen Aufwand geschlossen werden. Bei Zugrundelegen einer willkürlichen systemspezifischen Grundzeiteinheit t_0 berechnet sich der Aufwand $E^g(\mathbf{z}_i)$ eines Grundalgorithmus g für eine Parameterkombination $\mathbf{z}_i$ als deren Faktor:

$$E^g(\mathbf{z}_i)t_0. \tag{3.104}$$

Die entsprechende Rechenzeit $T(\mathbf{z}_i)$ kann damit direkt unabhängig von t_0 aus einer als bekannt angenommenen Zeit $T(\mathbf{z}_j)$ formelmäßig berechnet werden zu:

$$\begin{aligned} \frac{T(\mathbf{z}_i)}{E^g(\mathbf{z}_i)\,t_0} &= \frac{T(\mathbf{z}_j)}{E^g(\mathbf{z}_j)\,t_0} \\ \Rightarrow T(\mathbf{z}_i) &= T(\mathbf{z}_j)\,\frac{E^g(\mathbf{z}_i)}{E^g(\mathbf{z}_j)} \end{aligned} \tag{3.105}$$

Ferner kann mit $T(\mathbf{z}_0)$ über Gleichung 3.105 jede beliebige Zustandskombination $\mathbf{z}_i$ direkt berechnet werden. Im folgenden sollen die Aufwandsformeln $E^g(\mathbf{z})$ der drei Grundalgorithmen hergeleitet werden:

- Block Matching (BM)

  Für eine mathematische Beschreibung des Gesamtaufwandes gilt es zunächst, die Größe des Bildbereiches zu bestimmen, für den die Disparitäten berechnet werden können. Abbildung 3.51 zeigt hierzu das Referenz- und Suchbild der Größe $x_f \times y_f$, die um die globale Bildverschiebung $d = (d_x, d_y)$ gegeneinander verschoben sein sollen (vgl. Abschnitt 3.2.3). Innerhalb des so entstehenden Überlappbereiches können aus dem Referenzbild weiterhin nur solche Blöcke verwendet werden, deren um den Suchvektor $\mathbf{v}$ verschobene Position sich noch im Suchbild befindet. Für die Breite u_x und Höhe u_y der effektiven Berechnungsfläche ergibt sich somit:

$$\begin{aligned} u_x &= (x_f - v_x - d_x), \\ u_y &= (y_f - v_y - d_y). \end{aligned} \tag{3.106}$$

  Zur Disparitätsschätzung wird der entsprechende Bereich im Referenzbild in Blöcke der Größe $k \times k$ aufgeteilt, die um den Blockabstand b versetzt positioniert sind (grün in Abbildung 3.52). Für die Anzahl der Referenzblöcke

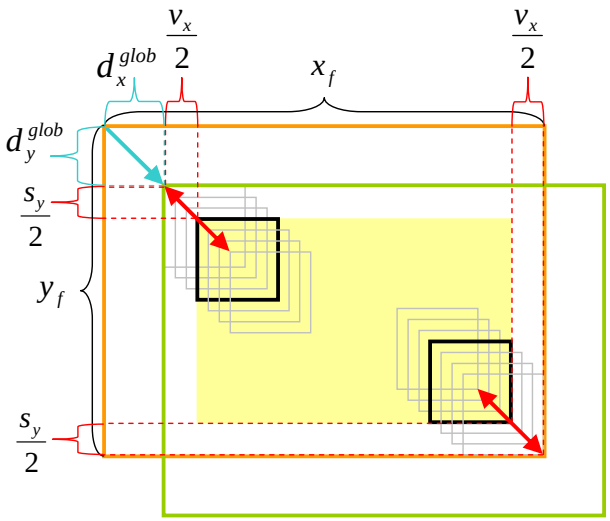


Abbildung 3.51.: Referenzbild f_r und Suchbild f_s sind um die globale Bildverschiebung d (blau) gegeneinander verschoben. Für alle Blockpositionen im Referenzbild müssen die innerhalb des Suchvektors (rot) verschobenen Blöcke im Suchbild liegen.

bei vollständiger Disparitätsberechnung ergibt sich somit:

$$\left(\left\lfloor\frac{u_x-k}{b}\right\rfloor+1\right)\left(\left\lfloor\frac{u_y-k}{b}\right\rfloor+1\right). \tag{3.107}$$

Bei partieller Disparitätsberechnung in einem um s eingeschränkten Schätzbereich ergibt sich:

$$n_{ref}^{BM}=s\left(\left\lfloor\frac{u_x-k}{b}\right\rfloor+1\right)\left(\left\lfloor\frac{u_y-k}{b}\right\rfloor+1\right). \tag{3.108}$$

Ein Referenzblock wird entlang des Suchvektors $\mathbf{v}$ mit Suchblöcken verglichen (blau in Abbildung 3.52). Bei einer Logarithmischen Suche mit l Stufen finden pro Block $\frac{|\mathbf{v}|}{2^{l-1}}+2(l-1)$ Vergleiche statt.
Eine Luminanzkorrektur erhönt die Anzahl verglichener Blöcke um den Faktor $1+2c$, sodaß sich deren Gesamtanzahl ergibt zu:

$$n_{vgl}^{BM}= s\left(\left\lfloor\frac{u_x-k}{b}\right\rfloor+1\right)\left(\left\lfloor\frac{u_y-k}{b}\right\rfloor+1\right)\left(\frac{|\mathbf{v}|}{2^{l-1}}+2(l-1)\right)(1+2c). \tag{3.109}$$

Die Anzahl der SAD-Berechnungen pro Block (orange in Abbildung 3.52) ergibt sich bei einem stets abgetasteten zentralen Pixel (vgl. Abschnitt

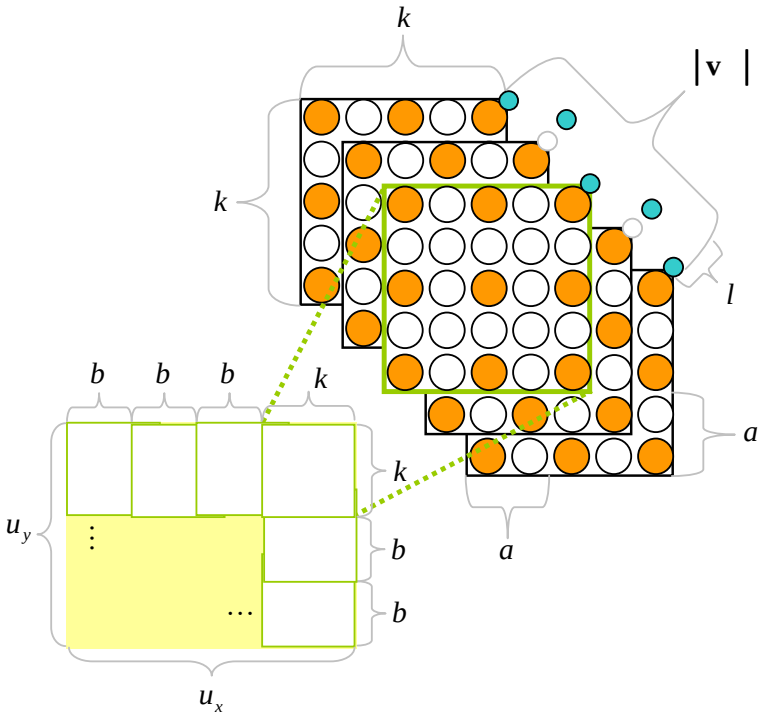


Abbildung 3.52.: Der Überlappbereich (gelb) wird in Blöcke (grün) der Kantenlänge k und mit Abstand b aufgeteilt. Eine Abtastung findet nur für Pixel mit Abstand a zum zentralen Pixel eines Blocks statt. Ein Referenzblock wird mit Blöcken im Suchbild entlang dem Suchbereichsvektor $\mathbf{v}$ verglichen. Hierfür werden nur einer logarithmischen Suche mit l Stufen entsprechende Blöcke verwendet (blau).

3.6.2.4) mit der Abtastperiode a zu $\left\lfloor \frac{k-1}{a} \right\rfloor^2$, sodaß sich die Gesamtzahl der Pixelvergleiche ergibt zu:

$$n_{sad}^{BM} = s \left(\left\lfloor \frac{u_x - k}{b} \right\rfloor + 1 \right) \left(\left\lfloor \frac{u_y - k}{b} \right\rfloor + 1 \right) \left(\frac{|\mathbf{v}|}{2^{l-1}} + 2(l-1) \right) (1 + 2c) \left\lfloor \frac{k-1}{a} \right\rfloor^2 . \tag{3.110}$$

Für eine konsistente Modellierung muß das geeignete Verhältinis der einzelnen Operationen zueinander berücksichtigt und somit die Gewichtungen

m_{ref}^{BM}, m_{vgl}^{BM} und m_{sad}^{BM} und der additive Offset m_0^{BM} bestimmt werden:

$$E^{BM}(\mathbf{z}, m_0^{BM}, m_{ref}^{BM}, m_{vgl}^{BM}, m_{sad}^{BM}) = m_0^{BM} + m_{ref}^{BM} n_{ref} + m_{vgl}^{BM} n_{vgl} + m_{sad}^{BM} n_{sad}. \quad (3.111)$$

- Disparitätsraum mit direkter Kostenaggregation (DRd)
  Im Falle der direkten Implementierung des Disparitätsraumes gilt es im ersten Schritt zunächst über Verschieben des Berechnungsbereiches um den Suchbereichsvektor $\mathbf{v}$ den bzw. die $1 + 2c$ zur Luminanzkorrektur nötigen Disparitätsräume aufzustellen. Mit Gleichung 3.106 und der prozentualen Schätzbereichsbreite s ergibt sich die Anzahl der dazu nötigen Absoluten Differenzen zu:

$$n_{ad} = \frac{u_x}{u}\frac{u_y}{u}|\mathbf{v}|s\,(1 + 2c)\,. \quad (3.112)$$

  Dabei gilt für die Bildabtastperiode u:

$$u = \begin{cases} a & \text{falls } b \mod a = 0; \\ 1 & \text{sonst.} \end{cases} \quad (3.113)$$

  Die Anzahl der Summen im darauffolgenden Kostenaggregations- und Minimierungsschritt entspricht der Anzahl der SAD's beim Block Matching. Das Verschieben eines Blocks um den entsprechenden Suchbereichsvektor korrespondiert dabei mit dem Verschieben eines Blocks durch die einzelnen Ebenen des Disparitätsraumes. Somit ergibt sich die gewichtete Anzahl der Einzelfunktionalitäten zu:

$$E^{DRd}(\mathbf{z}, m_0^{DRd}, m_{ad}^{DRd}, m_{ref}^{DRd}, m_{vgl}^{DRd}, m_{sad}^{DRd}) = m_0^{DRd} + m_{ad}^{DRd} n_{ad} + m_{ref}^{DRd} n_{ref} + m_{vgl}^{DRd} n_{vgl} + m_{sad}^{DRd} n_{sad}. \quad (3.114)$$

- Disparitätsraum mit Kostenaggregation als Moving Average Filter
  Bei Implementierung der Kostenaggregation als Moving Average Filter soll als Maß des Aufwandes die Anzahl γ der durchgeführten Additionen und Subtraktionen und die Anzahl σ der durchgeführten Speicherzugriffe betrachtet werden.
  Der Aufwand zum Aufstellen des Disparitätsraumes gleicht hierbei dem vorangegangenen Fall und damit Gleichung 3.112.

Für die Anfangssumme einer Kostenaggregationszeile der Länge u_x werden k Pixel ausgelesen und aufsummiert. Für alle übrigen $u_x - k$ Pixel der Zeile wird jeweils eine Leseoperation, eine Addition und eine Subtraktion durchgeführt (moving average), es ergibt sich für die entsprechenden Operationen pro Zeile:

$$k(\sigma + \gamma) + (u_x - k)(\sigma + 2\gamma). \tag{3.115}$$

Da nur Summen entsprechend dem Blockabstand Berücksichtigung finden sollen, ergeben sich pro Zeile $\frac{u_x - k}{b} + 1$ Werte, die in den Speicher geschrieben werden, sodaß sich bei einer Gesamtanzahl von u_y Zeilen ergibt:

$$u_y \left(k(\sigma + \gamma) + (u_x - k)(\sigma + 2\gamma) + \left(\frac{u_x - k}{b} + 1 \right) \sigma \right). \tag{3.116}$$

Für die Anfangssumme einer Kostenaggregationsspalte der Länge u_y werden k Pixel ausgelesen und aufsummiert. Für alle übrigen $u_y - k$ Pixel der Spalte wird wiederum jeweils eine Leseoperation, eine Addition und eine Subtraktion durchgeführt:

$$k(\sigma + \gamma) + (u_y - k)(\sigma + 2\gamma). \tag{3.117}$$

Zwischenergebnisse werden für jeden der $\frac{u_y - k}{b} + 1$ Blöcke pro Spalte geschrieben, sodaß sich bei einer Gesamtanzahl von $\frac{u_x - k}{b} + 1$ Spalten ergibt:

$$\left(\frac{u_x - k}{b} + 1 \right) \left(k(\sigma + \gamma) + (u_y - k)(\sigma + 2\gamma) + \left(\frac{u_y - k}{b} + 1 \right) \sigma \right). \tag{3.118}$$

Für jedes der $|\mathbf{v}|$ Differenzbilder der $1 + 2c$ Disparitätsräume ausgeführt, ergibt sich bei horizontaler Modifikation der Bildgröße um den Skalierungsparameter s für die Kostenaggregation als Moving Average Filter:

$$|\mathbf{v}| \, (1 + 2c) \left(u_y \left(k(\sigma + \gamma) + (su_x - k)(\sigma + 2\gamma) + \left(\frac{su_x - k}{b} + 1 \right) \sigma \right) + \left(\frac{su_x - k}{b} + 1 \right) \left(k(\sigma + \gamma) + (u_y - k)(\sigma + 2\gamma) + \left(\frac{u_y - k}{b} + 1 \right) \sigma \right) \right). \tag{3.119}$$

Der Aufwand zum Schreiben der Ergebnisdisparitäten für jedes Pixel des Berechnungsbereiches beläuft sich zusätzlich zu:

$$su_x u_y \sigma. \tag{3.120}$$

Bei Annahme gleicher Rechenzeiten für Speicherzugriffe und Additionen ergibt sich daraus mit $\sigma = \gamma$ für die Anzahl n_{op} der Operationen:

$$\begin{aligned} n_{op} &= |v|(1+2c)u_y \left(3su_x - k + \frac{su_x - k}{b} + 1\right) \\ &+ \left(\frac{su_x - k}{b} + 1\right)\left(3u_y - k + \frac{u_y - k}{b} + 1\right) \\ &+ su_x su_y. \end{aligned} \tag{3.121}$$

Mit einer der direkten Implementierung entsprechenden Anzahl von Absoluten Differenzen (Gleichung 3.112) beim Schritt der Kostenberechnung ergibt sich so:

$$E^{DRma}(\mathbf{z}, m_0^{DRma}, m_{ad}^{DRma}, m_{op}^{DRma}) = m_0^{DRma} + m_{ad}^{DRma} n_{ad} + m_{op}^{DRma} n_{op}. \tag{3.122}$$

Da der Aufwand des Peak Ratios p mathematisch nur schwer modellierbar ist, soll dieser ebenfalls durch einen später zu bestimmenden Gewichtungsfaktor beschrieben werden. Der zusätzliche Vergleichsaufwand n_p^g zur Ermittlung des zweiten Maximums des SAD-Verlaufs ist proportional zur Anzahl n_{vgl}^g der Blockvergleiche. Die mit einer Berechnung des Peak Ratios verbundene Fehlstellenergänzung bezieht sich auf das fertige Ergebnisbild und ist somit als zusätzlicher additiver Offset n_f^g zu realisieren. Damit verändern sich die Aufwände E^g der einzelnen Grundalgorithmen zu:

$$E^g + p(n_{vgl}^g n_p^g + n_f^g). \tag{3.123}$$

3.7.6.4. Vergleich der Zeitmodelle

In Abbildung 3.53 (links) wurden für alle paarweisen Parameterkombinationen $x \in X = \{x_0, x_1, x_2, \ldots\}$ und $y \in Y = \{y_0, y_1, y_2, \ldots\}$ die entsprechenden Rechenzeiten gemessen:

$$\hat{\mathbf{T}}(x,y) := \begin{bmatrix} T^g(x_0,y_0) & T^g(x_1,y_0) & \ldots \\ T^g(x_0,y_1) & T^g(x_1,y_1) & \ldots \\ \vdots & \vdots & \end{bmatrix}. \tag{3.124}$$

Der Fehler F_{abs} zwischen den gemessenen und mithilfe des Zeitmodells berechneten Rechenzeiten kann mit dem Vektor $\mathbf{m}^g = (m^g_{ref}, m^g_{sad}, m^g_{vgl}, \ldots, m^g_p, m^g_f)$ über die jeweiligen Gewichtungsfaktoren eines Algorithmenmodells für ein Parameterpaar x und y berechnet werden zu:

$$F_{abs}(x,y,\mathbf{m}^g) = \sum_{x\in X}\sum_{y\in Y}\left| T(x_i,y_j) - T^g_0 \frac{E^g(x_i,y_j,\mathbf{m})}{E^g(\mathbf{z}_0}, \mathbf{m})\right|. \tag{3.125}$$

Die Bestimmung der einzelnen Werte für die Gewichtungsfaktoren der expliziten Zeitmodellierung (vgl. Abschnitt 3.7.6.3) findet über Minimierung der absoluten Fehlersumme über alle Rechenzeiten paarweise verknüpfter Skalierungsparameter statt:

$$\mathbf{m}^g = \arg\min_{\mathbf{m}'^g} \sum_{i=\{k,b,a,l,s,p,c\}} \sum_{j=\{k,b,a,l,s,p,c\}} F_{abs}(i,j,\mathbf{m}'^g). \tag{3.126}$$

So ergeben sich die in Tabelle 3.2 zusammengefaßten Parameter, die der Berechnung der in Abbildung 3.53 (rechts) dargestellten Zeitverteilungen zu Grunde liegen. Zum Vergleich der beiden Modellierungsmethoden zeigt Abschnitt A.3.2

g	m^g_0	m^g_{ref}	m^g_{vgl}	m^g_{sad}	m^g_{ad}	m^g_{op}	m^g_p	m^g_f
BM	1358391	198	5	6			0	239
DRd	0	0	161	10	73		135013638	1
DRma	0				1225	4	0	2

Tabelle 3.2.: Über Fehleroptimierung ermittelte Gewichtungsfaktoren für die Einzelelemente der formelmäßigen Modellierung des Aufwandes der Grundalgorithmen.

(Mitte) die über implizite Modellierung berechneten Rechenzeiten aller paarweisen Verknüpfungen von Skalierungsparametern $x \in X = \{x_0, x_1, x_2, \ldots\}$ und $y \in Y = \{y_1, y_2, y_3, \ldots\}$, wie sie beispielhaft in Abbildung 3.53 (Mitte) für die Skalierungsparameter $k \in K = \{1, 3, 5, \ldots, 25\}$ und $b \in B = \{1, 2, 3, \ldots, 25\}$ dargestellt sind.

$$\begin{aligned}
\hat{\mathbf{T}}^g(x,y) &:= T_0^g \mathbf{t}^g(x) \mathbf{t}^{g\mathrm{T}}(y) \\
&= T_0^g \begin{pmatrix} T^g(x_0) \\ T^g(x_1) \\ T^g(x_2) \\ \vdots \end{pmatrix} \left(T^g(y_0), T^g(y_1), T^g(y_2), \ldots\right) \qquad (3.127) \\
&= \begin{bmatrix} T_0^g T^g(x_0) T^g(y_0) & T_0^g T^g(x_1) T^g(y_0) & \ldots \\ T_0^g T^g(x_0) T^g(y_1) & T_0^g T^g(x_1) T^g(y_1) & \ldots \\ \vdots & \vdots & \end{bmatrix}. \qquad (3.128)
\end{aligned}$$

Die nach Gleichung 3.126 berechnete absolute Gesamtfehlersumme der impliziten Modellierung liegt mit 194,87833s deutlich unter der absoluten Gesamtfehlersumme von 584.769771s der expliziten Modellierung. Auch ein optischer Vergleich der gemessenen und berechneten Verteilungen in Abbildung 3.53 und Abschnitt A.3.2 zeigt eindeutig eine größere Ähnlichkeit der implizit berechneten Werte zu den gemessenen, insbesondere bei Fehlerminimierung im Disparitätsraum und Kostenaggregation als Moving Average Filter. Die Ursache ist hier in der ungenauen Modellierbarkeit des Moving Average Filters anhand von Einzeloperationen zu suchen, die der Modellierung über gewichtete Operationsgruppen, wie sie bei den beiden anderen Implementierungen möglich ist, offensichtlich unterliegt. Entsprechend soll zur folgenden Berechnung der Skalierungsvorschrift eine implizite Zeitmodellierung verwendet werden.

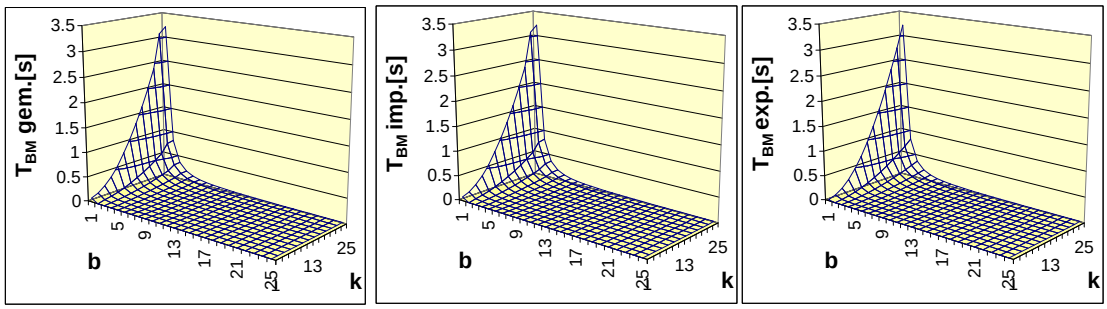


Abbildung 3.53.: (Links) Verteilungen der direkt gemessenen Verbundrechenzeiten $T(k, b)$ der Parameter k und b. (Mitte) Aus der Verknüpfung $T_0^g T(k)T(b)$ der für die Einzelparameter gemessenen Rechenzeiten (vgl. Abbildung 3.50) berechnete Verbundrechenzeiten. (Rechts) Über formelmäßige Modellierung explizit berechnete Rechenzeiten. Vgl. dazu auch Abschnitt A.3.2.

3.7.6.5. Komponentenabhängige Verhaltensanalyse

Insbesondere bei „Embedded Systems“ ist eine starke Abhängigkeit der Rechenleistung des Gesamtsystems von bestimmten Einzelkomponenten gegeben. Entsprechend soll hier, auch im Hinblick auf einen Wechsel zwischen unterschiedlichen Systemen, eine Übersicht über die Anzahl der Speicherzugriffe σ und Ganzzahloperationen (Additionen/Subtraktionen) γ in Abhängigkeit von den einzelnen Skalierungsparametern gegeben werden. Dabei muß analog zu Abschnitt 3.7.6.3 zwischen den einzelnen Grundalgorithmen unterschieden werden:

- Block Matching
  Für jede SAD-Berechnung muß ein Pixelwert aus dem Referenzbild und einer aus dem Suchbild ausgelesen werden. Diese werden subtrahiert und das Ergebnis aufaddiert. Schließlich wird in jedes Pixel im Berechnungsbereich ein Disparitätswert geschrieben. Für den Aufwand A_{BM} ergibt sich mit den Gleichungen 3.110 und 3.108 somit:

$$A_{BM} = n_{sad}(2\sigma + 2\gamma) + \left(n_{ref} - 1)b^2 + k^2\right)\sigma. \tag{3.129}$$

- Fehlerminimierung im Disparitätsraum (direkt)
  Die Anzahl der Pixel im Disparitätsraum ergibt sich mit Gleichung 3.112, wobei für jedes einmal aus dem Referenzbild gelesen, einmal aus dem Suchbild gelesen, die Differenz berechnet und in den Disparitätsraum geschrieben wird. Bei der Kostenaggregation wird dann jedes Vergleichspixel aus dem Disparitätsraum gelesen und aufsummiert. Schließlich wird wiederum in jedes Pixel im Berechnungsbereich ein Disparitätswert geschrieben. Somit gilt für den Aufwand:

$$A_{DRd} = n_{ad}(\gamma + 3\sigma) + n_{sad}(\gamma + \sigma) + \left(n_{ref} - 1)b^2 + k^2\right)\sigma. \tag{3.130}$$

- Fehlerminimierung im Disparitätsraum (Moving Average Filter)
  Der Aufwand zum Aufstellen des Disparitätsraumes gleicht hierbei dem vorangegangenen Fall.
  Eine Aufschlüsselung der Einzeloperationen hat ferner bereits in Gleichung

3.119 stattgefunden, sodaß für den Aufwand gilt:

$$
\begin{aligned}
&A_{DRma} = n_{ad}(\gamma + 3\sigma) \\
&+ |\mathbf{v}|\,(1+2c)\left(u_y\left(k(\sigma+\gamma) + (su_x - k)(\sigma + 2\gamma) + \left(\frac{su_x - k}{b} + 1\right)\sigma\right)\right. \\
&\left. + \left(\frac{su_x - k}{b} + 1\right)\left(k(\sigma+\gamma) + (u_y - k)(\sigma + 2\gamma) + \left(\frac{u_y - k}{b} + 1\right)\sigma\right)\right) \\
&\qquad + \left(n_{ref} - 1)b^2 + k^2\right)\sigma. \quad (3.131)
\end{aligned}
$$

Abbildung 3.54 zeigt entsprechend das prozentuale Verhalten der Operationen des Speichers und der Ganzzahleinheit bezüglich des Grundzustandes. Dabei kann, mit einziger Ausnahme des Blockabstandes b bei Fehlerminimierung im Disparitätsraum und Kostenaggregation als Moving Average Filter, ein identisches Skalierungsverhalten beider Operationsgruppen beobachtet werden. Wird der Algorithmus auf unterschiedlichen Systemen mit einem unterschiedlichen Speicher- und Prozessorverhalten ausgeführt, kann trotzdem von einem identischen relativen Skalierungsverhalten des Gesamtsystems ausgegangen werden, sodaß eine optimale Portierbarkeit gewährleistet ist.

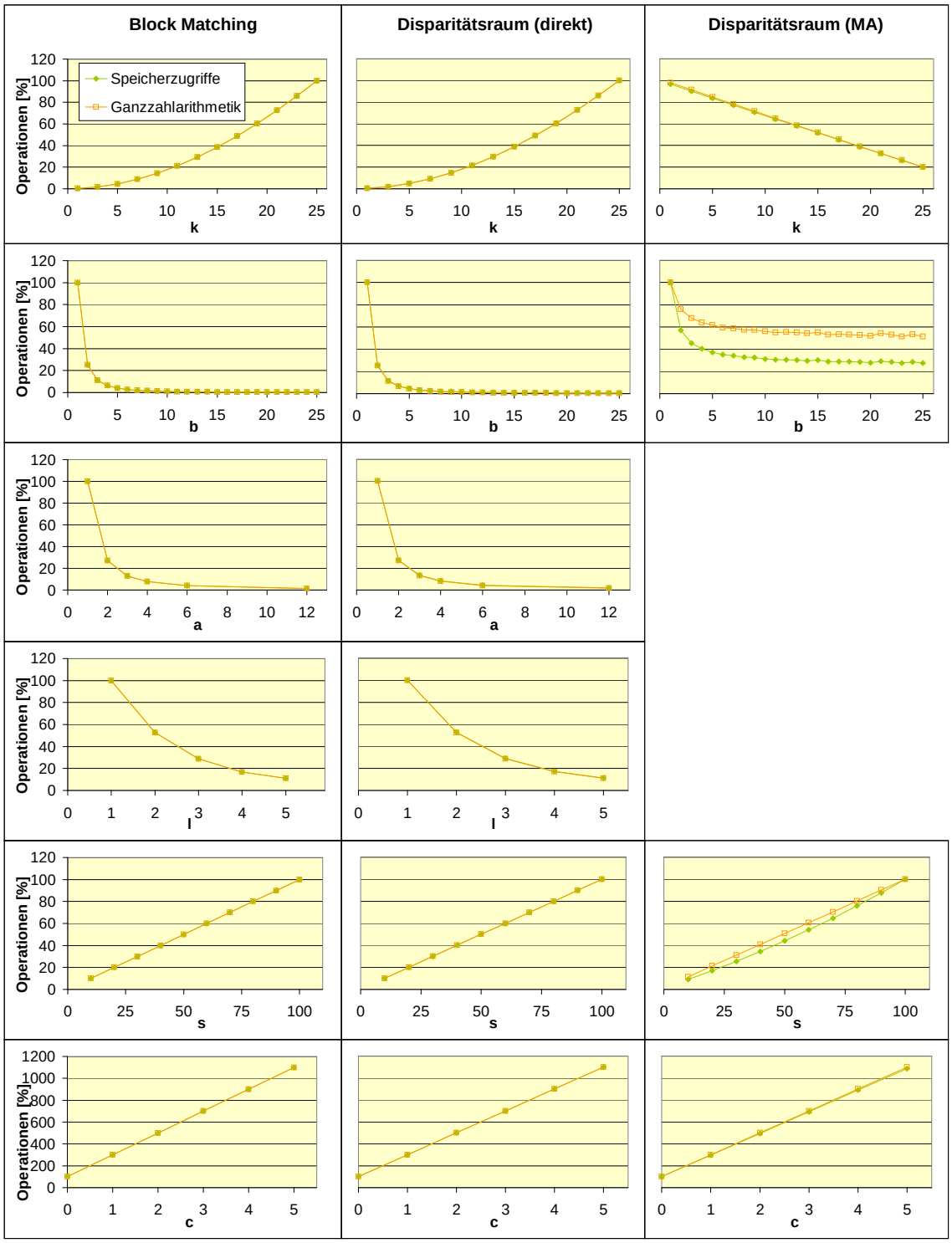


Abbildung 3.54.: Prozentuales Verhalten der Speicherzugriff- und Ganzzahlarithmetikoperationen bezogen auf den Grundzustand, in Abhängigkeit der einzelnen Skalierungsparameter.

3.7.7. Skalierungsvorschrift

3.7.7.1. Herleitung

Im vorangegangenen Abschnitt wurde die Bestimmung sowohl des Fehler-, als auch des Zeitverhaltens beliebiger Parameterkombinationen aus den jeweiligen Meßwerten der Einzelparameter erörtert und verifiziert. Um die in Abschnitt 1 beschriebene Skalierbarkeit zu erlangen und während des Programmlaufs vom Zeitverhalten des aktuellen Zustands auf das aller anderen schließen zu können, wird eine Vorschrift benötigt, die jedem sinnvollen Parameterzustand dessen Fehlerwahrscheinlichkeit und Rechenzeit zuordnet. Ferner gilt es, ineffiziente Parameterkombinationen zu eliminieren, also solche, deren Qualität (Fehlerwahrscheinlichkeit) mit weniger Aufwand (Rechenzeit) erreicht werden kann. Eine vollständige Berechnung aller Zustände findet über die entsprechenden Einzelparameter-Fehlerwahrscheinlichkeits- (vgl. Abbildung 3.47) bzw. relativen Zeitvektoren (vgl. Abbildung 3.50) statt.
Mit dem Tensorprodukt können alle Kombinationen der Elemente der relativen Zeitvektoren $\mathbf{t}^g(\mathbf{x})$ berechnet werden:

$$\mathbf{w}^g = T_0^g\Big(\Big(\big(\mathbf{t}^g(\mathbf{k}) \otimes \mathbf{t}^g(\mathbf{b})\big) \otimes \mathbf{t}^g(\mathbf{a})\Big) \otimes \mathbf{t}^g(\mathbf{l})\Big) \otimes \ldots \tag{3.132}$$

$$\text{z.B.:} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \otimes \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_0 y_0 \\ x_0 y_1 \\ x_1 y_0 \\ x_1 y_1 \end{pmatrix}. \tag{3.133}$$

Mit der Tensorsumme soll für die Verknüpfung zweier Vektoren wegen $A \cup B = A + B - AB$ gelten:

$$\tilde{\mathbf{r}}(\mathbf{x}, \mathbf{y}) := (\mathbf{x} \oplus \mathbf{y}) \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \mathbf{x} \otimes \mathbf{y}, \tag{3.134}$$

$$\text{z.B.: } \tilde{\mathbf{r}}\left(\begin{pmatrix} x_0 \\ x_1 \end{pmatrix}, \begin{pmatrix} y_0 \\ y_1 \end{pmatrix}\right) = \begin{pmatrix} x_0 + y_0 - x_0 y_0 \\ x_0 + y_1 - x_0 y_1 \\ x_1 + y_0 - x_1 y_0 \\ x_1 + y_1 - x_1 y_1 \end{pmatrix}. \tag{3.135}$$

Damit können alle entsprechenden Verbundfehlerwahrscheinlichkeiten aus den Vektoren $\mathbf{p}(\mathbf{x})$ der Einzelparameter-Fehlerwahrscheinlichkeiten berechnet wer-

den:

$$\mathbf{n} = P_0\tilde{\mathbf{r}}\Big(\tilde{\mathbf{r}}\Big(\tilde{\mathbf{r}}\big(\mathbf{p}(\mathbf{k}),\mathbf{p}(\mathbf{b})\big),\mathbf{p}(\mathbf{a})\Big)\mathbf{p}(\mathbf{l})\Big)\ldots \tag{3.136}$$

Für die Verknüpfung eines Vektors $\mathbf{x}$ und der $m \times n$-Matrix $\mathbf{Y}$ zu einer kombinatorischen Matrix soll gelten:

$$\tilde{\mathbf{V}}(\mathbf{x},\mathbf{Y}) = [\mathbf{x}\otimes\mathbf{1}_m\big|\mathbf{1}_{\dim(\mathbf{x})}\otimes\mathbf{Y}],$$

$$\text{z.B.: } \tilde{\mathbf{V}}\left(\begin{pmatrix} x_0 \\ x_1 \end{pmatrix}, \begin{bmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{bmatrix}\right) =$$

$$\left[\begin{pmatrix} x_0 \\ x_1 \end{pmatrix}\otimes\begin{pmatrix} 1 \\ 1 \end{pmatrix}\middle|\begin{pmatrix} 1 \\ 1 \end{pmatrix}\otimes\begin{pmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{pmatrix}\right] = \begin{bmatrix} x_0 & y_{00} & y_{01} \\ x_0 & y_{10} & y_{11} \\ x_1 & y_{00} & y_{01} \\ x_1 & y_{10} & y_{11} \end{bmatrix}. \tag{3.137}$$

Für die Matrix aller Parameterkombinationen ergibt sich somit aus den Parametervektoren $\mathbf{x}$:

$$\mathbf{N} = \tilde{\mathbf{V}}\Big(\mathbf{g},\tilde{\mathbf{V}}\big(\mathbf{k},\tilde{\mathbf{V}}(\mathbf{b},\ldots)\big)\Big). \tag{3.138}$$

Entsprechend kann die Matrix aufgestellt werden:

$$\mathbf{M} = \begin{bmatrix} \mathbf{N} & \begin{matrix}\mathbf{n} & \mathbf{w}^{BM}\\ \mathbf{n} & \mathbf{w}^{DRd}\\ \mathbf{n} & \mathbf{w}^{DRma}\end{matrix} \end{bmatrix} = \begin{bmatrix} \mathbf{z}_i^T & P(Q(\mathbf{z}_i)) & T(\mathbf{z}_i) \\ \mathbf{z}_{i+1}^T & P(Q(\mathbf{z}_{i+1})) & T(\mathbf{z}_{i+1}) \\ \vdots & \vdots & \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{m}_0 \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \end{bmatrix} \tag{3.139}$$

Eine Zeile $\mathbf{m}_i$ dieser Matrix entspricht einer bestimmten Parameterzustandskombination mit der zugehörigen Fehlerwahrscheinlichkeit und Rechenzeit. Zunächst müssen die Zeilen in der Praxis nicht realisierbarer Kombinationen eliminiert werden. Darauffolgend werden alle Zeilen entsprechend ihrer Rechenzeiten aufsteigend sortiert, sodaß gilt:

$$\forall_{T(\mathbf{z}_i)} : \big(T(\mathbf{z}_{i-j}) < T(\mathbf{z}_i)|0 \le j < i\big). \tag{3.140}$$

Schließlich werden alle Zeilen eliminiert, deren Fehlerwahrscheinlichkeit durch eine schnellere Parameterkonstellation erreicht oder unterboten wird (Pareto-Optimierung), sodaß eine Zeile $\mathbf{m}_i$ eliminiert wird, wenn gilt:

$$\exists_{P(Q(\mathbf{z}_j))} : \big(P(Q(\mathbf{z}_j)) \le P(Q(\mathbf{z}_i))|0 \le j < i\big). \tag{3.141}$$

Somit gilt zusätzlich für alle Zeilen $\mathbf{m}_i$:

$$\forall_{P(Q(\mathbf{v}_i))} : \big(P(Q(\mathbf{z}_{i-j})) < P(Q(\mathbf{z}_i))|j < i\big). \tag{3.142}$$

Von den ursprünglich 11176800 Zustandskombinationen bleiben damit noch die 129 in Tabelle 3.3 dargestellten Kombinationen übrig, deren Fehlerwahrscheinlichkeit in Abhängigkeit von der Rechenzeit in Abbildung 3.55 abgebildet sind.

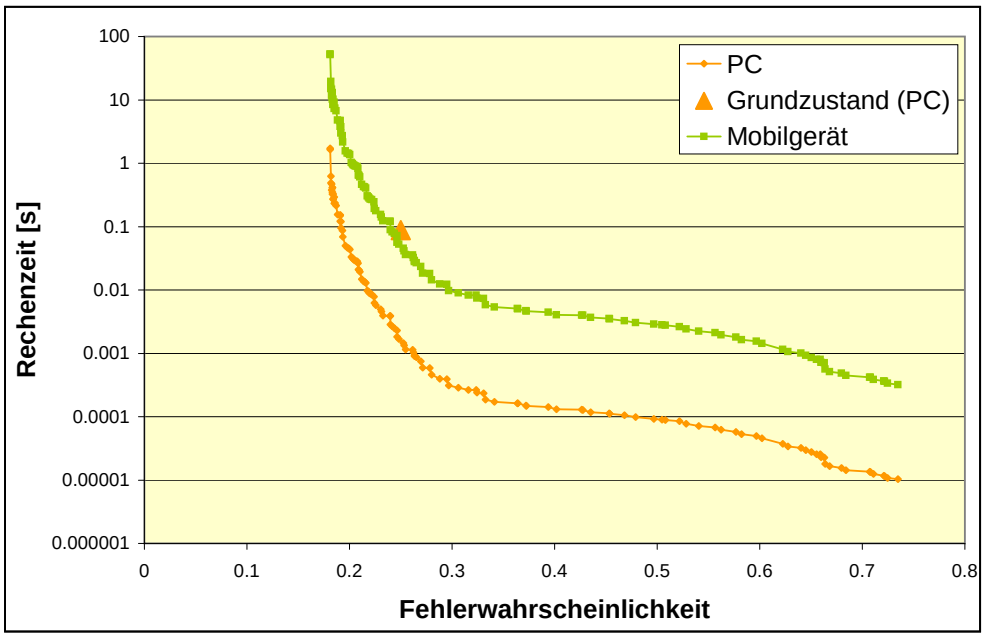


Abbildung 3.55.: Fehlerwahrscheinlichkeit in Abhängigkeit von der dazu nötigen Rechenzeit für die ermittelte Skalierungsvorschrift.

3.7.7.2. Bewertung

Aus Tabelle 3.3 gehen die 129 Zustände der Skalierungsvorschrift hervor, die dem Pareto-Optimum der Menge aller Zustandskombinationen entsprechen.
Bezüglich des Grundalgorithmus gilt es zunächst festzuhalten, daß bis zum Zustand Nummer 110 ausschließlich das Block Matching zum Einsatz kommt. Erst bei den hohen Fehlerwahrscheinlichkeiten wird die Fehlerminimierung im Disparitätsraum mit Implementierung der Kostenaggregation als Moving Average Filter (DRma) effizienter. Die Fehlerminimierung im Disparitätsraum bei direkter Implementierung der Kostenaggregation (DRd) ist nie am effizientesten und tritt in der Liste nicht auf.
Die Werte der Blockgröße sind nach einem leichten Schwanken in der ersten Hälfte der Skalierungsvorschrift konstant auf deren Maximum.
Der Blockabstand b bleibt mit Werten nahe seinem Maximum bis Zustand Nummer 110 sehr hoch und nimmt dann gegen Ende hin stark ab.
Sowohl Blockabtastperiode a, wie auch die Anzahl logarithmischer Suchschritte l ändert sich stetig vom jeweiligen Maximum bei hohen Fehlerwahrscheinlichkeiten bis hin zum Minimum von 1 bei den niedrigen Fehlerwahrscheinlichkeiten.
Die Bildabtastung u der Fehlerminimierung im Disparitätsraum mit Implementierung der Kostenaggregation als Moving Average Filter bleibt konstant bei 1 und damit ohne Effekt, da der entsprechende Grundalgorithmus nur bei wenigen Zuständen geringen Fehlers auftritt.
Die Suchbereichsbreite s steigt beginnend mit ihrem Minimum von 10% bis hin zu 100% stetig an, bis sie ab Zustand Nummer 45 dort verharrt.
Der Wert der Luminanzkorrektur c bleibt zunächst bei 0 und steigt dann erwartungsgemäß beginnend ab Zustand Nummer 74 stetig an.
Der Peak Ratio fluktuiert bis zum Zustand Nummer 45, ab dem er dann konstant immer verwendet wird.
Zusammenfassend kann festgehalten werden, daß sich alle Parameter in einem realistischen Rahmen bewegen, was für die Korrektheit des Modells spricht. Die einzige Überraschung liegt in der hohen Effizienz des Block Matchings, die hauptsächlich in dessen Verhalten bezüglich hoher Blockabstände begründet liegt.

#	g	k	b	a	u	l	s	c	p	Fehler	Zeit[s] (PC)	Zeit[s] (Mobilgerät)
1	BM	23	23	11	1	4	10	0	0	0.734777379	0.00001031	0.000321392
2	BM	21	21	10	1	4	10	0	0	0.724675231	0.00001078	0.000335973
3	BM	19	19	9	1	4	10	0	0	0.721622771	0.00001159	0.000361263
4	BM	21	21	10	1	3	10	0	0	0.720940004	0.00001171	0.000365125
5	BM	21	21	10	1	4	10	0	1	0.71100651	0.00001242	0.000387098
6	BM	19	19	9	1	4	10	0	1	0.707802508	0.00001335	0.000416237
7	BM	21	21	10	1	3	10	0	1	0.707085845	0.00001350	0.000420686
8	BM	25	25	6	1	4	10	0	0	0.683935268	0.00001435	0.000447386
9	BM	25	25	6	1	3	10	0	0	0.679647337	0.00001560	0.000486206
10	BM	25	25	6	1	4	10	0	1	0.668243978	0.00001654	0.000515465
11	BM	25	25	6	1	3	10	0	1	0.66374317	0.00001797	0.000560191
12	BM	25	22	6	1	3	10	0	1	0.66289263	0.00002271	0.000707918
13	BM	25	25	6	1	2	10	0	1	0.659648086	0.00002305	0.000718457
14	BM	21	20	5	1	4	10	0	1	0.659197406	0.00002552	0.000795459
15	BM	21	21	5	1	3	10	0	1	0.655350804	0.00002564	0.000799142
16	BM	25	25	4	1	4	10	0	1	0.650409963	0.00002762	0.000860749
17	BM	25	25	6	1	4	20	0	0	0.645096247	0.00002961	0.000922824
18	BM	25	25	6	1	3	20	0	0	0.640281401	0.00003218	0.001002897
19	BM	25	25	6	1	4	20	0	1	0.627476763	0.00003411	0.001063249
20	BM	25	25	6	1	3	20	0	1	0.622422881	0.00003707	0.001155507
21	BM	25	25	6	1	4	30	0	0	0.60206811	0.00004586	0.001429414
22	BM	25	25	6	1	3	30	0	0	0.596669518	0.00004984	0.001553443
23	BM	25	25	6	1	4	30	0	1	0.582312459	0.00005284	0.001646927
24	BM	25	25	6	1	3	30	0	1	0.57664585	0.00005742	0.00178983
25	BM	25	25	6	1	4	40	0	0	0.56241768	0.00006234	0.001943028
26	BM	25	25	6	1	3	40	0	0	0.556481166	0.00006775	0.002111624
27	BM	25	25	6	1	4	40	0	1	0.540693552	0.00007182	0.002238697
28	BM	25	25	6	1	4	50	0	0	0.52808161	0.00007779	0.002424735
29	BM	25	25	6	1	3	50	0	0	0.521679271	0.00008454	0.002635128
30	BM	21	21	10	1	4	80	0	0	0.507863303	0.00008839	0.00275493
31	BM	25	25	6	1	4	50	0	1	0.50465284	0.00008963	0.002793706
32	BM	25	25	6	1	4	60	0	0	0.496752187	0.00009204	0.002868778
33	BM	21	21	10	1	4	90	0	0	0.478987244	0.00009809	0.00305741
34	BM	25	25	6	1	4	70	0	0	0.46829388	0.00010496	0.003271445
35	BM	21	21	10	1	4	100	0	0	0.45349043	0.00011175	0.003483175
36	BM	21	21	10	1	4	90	0	1	0.453121144	0.00011302	0.003522655
37	BM	25	25	6	1	4	80	0	0	0.435041555	0.00011770	0.003668504
38	BM	25	25	6	1	3	80	0	0	0.427376977	0.00012791	0.003986818
39	BM	21	21	10	1	4	100	0	1	0.42635852	0.00012875	0.004013209
40	BM	25	25	6	1	4	90	0	0	0.401892688	0.00013062	0.004071291
41	BM	25	25	6	1	3	90	0	0	0.393778392	0.00014195	0.004424555
42	BM	25	25	6	1	4	100	0	0	0.372623097	0.00014881	0.004638246
43	BM	25	25	6	1	4	90	0	1	0.372199167	0.00015049	0.004690817
44	BM	25	25	6	1	3	100	0	0	0.364111711	0.00016172	0.005040705
45	BM	25	25	6	1	3	90	0	1	0.36368203	0.00016355	0.005097838
46	BM	25	25	6	1	4	100	0	1	0.341476463	0.00017145	0.005344046
47	BM	25	25	6	1	3	100	0	1	0.332542522	0.00018633	0.005807747
48	BM	25	22	6	1	3	100	0	1	0.330854232	0.00023547	0.007339296
49	BM	25	25	6	1	2	100	0	1	0.324413932	0.00023897	0.007448553
50	BM	21	20	5	1	4	100	0	1	0.323519349	0.00026458	0.008246871
51	BM	21	21	5	1	3	100	0	1	0.315883983	0.00026581	0.008285054
52	BM	25	25	4	1	4	100	0	1	0.306076597	0.00028630	0.008923761
53	BM	25	25	4	1	3	100	0	1	0.2966624	0.00031114	0.009698073
54	BM	25	22	4	1	3	100	0	1	0.294883353	0.00039319	0.012255533
55	BM	25	25	4	1	2	100	0	1	0.288096846	0.00039904	0.012437975
56	BM	25	25	3	1	3	100	0	1	0.28011563	0.00046161	0.014388022
57	BM	25	22	3	1	3	100	0	1	0.278294729	0.00058334	0.018182259
58	BM	25	25	3	1	2	100	0	1	0.271348563	0.00059202	0.018452931
59	BM	25	22	3	1	2	100	0	1	0.269505486	0.00074814	0.023319117
60	BM	25	21	3	1	2	100	0	1	0.265043074	0.00086312	0.026902769
61	BM	25	25	2	1	3	100	0	1	0.2635038	0.00089840	0.028002573
62	BM	25	20	3	1	2	100	0	1	0.263386272	0.00093369	0.029102367
63	BM	21	20	2	1	3	100	0	1	0.262367703	0.00103856	0.032371282
64	BM	25	22	2	1	3	100	0	1	0.261640881	0.00113532	0.035387077
65	BM	25	25	2	1	2	100	0	1	0.254534426	0.00115222	0.035913869
66	BM	21	20	2	1	2	100	0	1	0.253384493	0.00133198	0.041516827
67	BM	25	22	2	1	2	100	0	1	0.252648819	0.00145607	0.045384646
68	BM	25	21	2	1	2	100	0	1	0.248083435	0.00167983	0.0523593
69	BM	25	20	2	1	2	100	0	1	0.2463884	0.00181718	0.056640249
70	BM	25	22	2	1	1	100	0	1	0.246296142	0.00229643	0.071578231

#	g	k	b	a	u	l	s	c	p	Fehler	Zeit[s] (PC)	Zeit[s] (Mobilgerät)
71	BM	25	17	2	1	2	100	0	1	0.243930668	0.00250390	0.078044994
72	BM	25	21	2	1	1	100	0	1	0.241691951	0.00264934	0.082578281
73	BM	25	20	2	1	1	100	0	1	0.239982508	0.00286595	0.089329965
74	BM	25	22	2	1	3	100	1	1	0.239806001	0.00387210	0.120690989
75	BM	25	25	1	1	2	100	0	1	0.239572396	0.00389717	0.12147228
76	BM	25	25	2	1	2	100	1	1	0.232489393	0.00392975	0.122487663
77	BM	21	20	2	1	2	100	1	1	0.231305454	0.00454283	0.141597084
78	BM	25	22	2	1	2	100	1	1	0.230548025	0.00496605	0.154788646
79	BM	25	21	2	1	2	100	1	1	0.225847632	0.00572923	0.178576366
80	BM	25	20	2	1	2	100	1	1	0.224102472	0.00619766	0.193176949
81	BM	25	22	2	1	1	100	1	1	0.224007486	0.00783219	0.244124358
82	BM	25	17	2	1	2	100	1	1	0.221572059	0.00853979	0.266179866
83	BM	25	21	2	1	2	100	2	1	0.220253998	0.00869953	0.271158927
84	BM	25	21	2	1	1	100	1	1	0.219267138	0.00903583	0.281641074
85	BM	25	20	2	1	2	100	2	1	0.218496228	0.00941082	0.293329154
86	BM	25	20	2	1	1	100	1	1	0.217507144	0.00977461	0.304668332
87	BM	25	17	2	1	2	100	2	1	0.215947531	0.01296723	0.404180288
88	BM	25	17	2	1	1	100	1	1	0.214955221	0.01346850	0.41980462
89	BM	25	21	2	1	1	100	2	1	0.213625956	0.01372044	0.427657329
90	BM	25	20	2	1	1	100	2	1	0.211853245	0.01484224	0.462623023
91	BM	25	21	1	1	2	100	1	1	0.210309837	0.01937810	0.604002822
92	BM	25	17	2	1	1	100	2	1	0.209282884	0.02045122	0.63745149
93	BM	25	20	1	1	2	100	1	1	0.208529651	0.02096247	0.6533867
94	BM	25	22	1	1	1	100	1	1	0.208432758	0.02649099	0.825707255
95	BM	25	17	2	1	1	100	3	1	0.207648236	0.02831182	0.882461203
96	BM	25	17	1	1	2	100	1	1	0.20594845	0.02888433	0.900306093
97	BM	25	21	1	1	2	100	2	1	0.204603935	0.02942463	0.917146883
98	BM	25	21	1	1	1	100	1	1	0.203597268	0.03056209	0.952600882
99	BM	25	20	1	1	2	100	2	1	0.202810886	0.03183042	0.992133735
100	BM	25	20	1	1	1	100	1	1	0.201801949	0.03306088	1.030486489
101	BM	25	17	1	1	2	100	2	1	0.200211035	0.04385936	1.367067996
102	BM	25	17	1	1	1	100	1	1	0.199198808	0.04555482	1.419914522
103	BM	25	21	1	1	1	100	2	1	0.197842864	0.04640695	1.446474914
104	BM	25	20	1	1	1	100	2	1	0.196034573	0.05020123	1.564740158
105	BM	25	17	1	1	1	100	2	1	0.193412623	0.06917263	2.156066381
106	BM	25	15	1	1	1	100	2	1	0.193238027	0.08668468	2.701905971
107	BM	25	17	1	1	1	100	3	1	0.191745166	0.09575969	2.984768194
108	BM	25	15	1	1	1	100	3	1	0.191570209	0.12000265	3.740405711
109	BM	25	17	1	1	1	100	4	1	0.191021716	0.12150430	3.787211049
110	DRma	25	4	1	1	1	100	1	1	0.190943421	0.14908011	4.646731362
111	BM	25	15	1	1	1	100	4	1	0.190846602	0.15226488	4.745998656
112	BM	25	11	1	1	1	100	2	1	0.188459948	0.15465481	4.820491367
113	BM	25	11	1	1	1	100	3	1	0.186782252	0.21409766	6.673286796
114	DRma	25	4	1	1	1	100	2	1	0.185097587	0.23298019	7.261843252
115	BM	25	9	1	1	1	100	2	1	0.185752969	0.23704320	7.388484693
116	DRma	25	3	1	1	1	100	2	1	0.184212643	0.27020826	8.42221821
117	DRma	25	9	1	1	1	100	3	1	0.184069678	0.27209593	8.481055664
118	DRma	25	4	1	1	1	100	2	1	0.185097587	0.29402333	9.164518589
119	DRma	25	4	1	1	1	100	3	1	0.18341294	0.32617227	10.16658055
120	BM	25	9	1	1	1	100	3	1	0.184069678	0.32815270	10.22830944
121	DRma	25	9	1	1	1	100	4	1	0.183339358	0.34983762	10.90421443
122	DRma	25	3	1	1	1	100	3	1	0.182526167	0.37829156	11.79110549
123	DRma	25	4	1	1	1	100	3	1	0.18341294	0.41223603	12.84913281
124	BM	25	9	1	1	1	100	4	1	0.183339358	0.41637523	12.97814905
125	DRma	25	3	1	1	1	100	3	1	0.182526167	0.46960450	14.63727129
126	DRma	25	3	1	1	1	100	4	1	0.181794466	0.48637487	15.15999278
127	DRma	25	3	1	1	1	100	4	1	0.181794466	0.62449707	19.46517329
128	DRma	25	1	1	1	1	100	4	1	0.18120914	1.64878852	51.39168125
129	DRma	25	1	1	1	1	100	4	1	0.18120914	1.71208136	53.3644783

Tabelle 3.3.: Auflistung der Einzelzustände der Skalierungsvorschrift mit zugehöriger Fehlerwahrscheinlichkeit und Rechenzeit.

3.7.7.3. Hinzufügen zusätzlicher Skalierungsparameter

An dieser Stelle soll, nicht zuletzt in Hinblick auf eine etwaige Entwurfsautomatisierung, das nachträgliche Hinzufügen zusätzlicher Skalierungsparameter betrachtet werden. Hierzu sind folgende Schritte nötig:

1. Implementierung
   Zunächst muß die Funktionalität des neuen Parameters in den bisherigen Programmablauf aufgenommen werden.

2. Test der Einzelzustände
   Darauffolgend findet eine große Anzahl von Tests bezüglich der einzelnen Zustände des Parameters statt, wobei sich alle anderen Parameter im Grundzustand $\mathbf{z}_0$ befinden.

3. Zustandseinordnung
   Dann muß unter den möglichen Zuständen des neuen Parameters derjenige ermittelt werden, der der bisherigen Grundzustandskombination $\mathbf{z}_0$ entspricht. Tritt dieser Parameter noch in keinerlei Hinsicht im verwendeten Algorithmus in Erscheinung, so ist sein Grundzustand 0 (z.b. Parameter c), handelt es sich um eine bereits verwendete Größe, muß unter Umständen eine geeignete Ermittlung des Fehlerminimums zur Einschränkung ihrer Zustandsmenge durchgeführt werden (vgl. Parameter k).

4. Normierung
   Zur Eingliederung in das Gesamtsystem muß die Normierung des Fehlers entsprechend Gleichung 3.87 und des Zeitverhaltens entsprechend Gleichung 3.105 auf den Grundzustand stattfinden, sodaß sich der Vektor der Einzelparameter-Fehlerwahrscheinlichkeiten und der relative Zeitvektor ergeben.

5. Test statistischer Unabhängigkeit
   Schließlich muß paarweise mit allen bereits verwendeten Skalierungsparametern die statistische Unabhängigkeit des Parameters bezüglich seines Vek-

tors der Einzelparameter-Fehlerwahrscheinlichkeiten nachgewiesen werden und das multiplikative Verhalten der zugehörigen Rechenzeiten.

6. Aufstellen der neuen Skalierungsvorschrift
   Wurden die im vorherigen Schritt durchgeführten Tests bestanden, kann unter Eingliederung des neuen Skalierungsparameters nach dem Vorgehen in Abschnitt 3.7.7.1 eine neue Skalierungsvorschrift aufgestellt werden.
   Schlägt der Test auf statistische Unabhängigkeit fehl, muß diese durch entsprechende Definition des Parameters und Einschränkung dessen Zustandsmenge geschaffen werden (vgl. Gleichung 3.57, Parameter k und a).
   Ist dies nicht möglich, müssen die Fehlerwahrscheinlichkeiten für die Einzelparameter in Kombination mit jedem möglichen Zustand des neuen Parameters einzeln gemessen und das Vorgehen in Abschnitt 3.7.7.1 mit der entsprechend höheren Anzahl von Einzelparameter-Fehlervektoren durchgeführt werden. Ein analoges Vorgehen wird im Falle eines Nichtbestehens des Tests auf multiplikatives Zeitverhalten nötig.

Alle der Implementierung folgenden Schritte sind dabei theoretisch automatisierbar.

4. Mitverfolgen planarer Objekte

Eine weitere elementare Grundfunktionalität im Rahmen der Augmented Reality ist das Einfügen Virtueller Objekte in reale Szenen. Dabei spielen virtuelle planare Flächen, also Texturen (z.B. auch Schriften), eine grundlegende und damit sehr wichtige Rolle. Für das Einblenden solcher Texturen sind entsprechend planare Flächen in der Szene attraktiv, sodaß sich der folgende Abschnitt mit dem Mitverfolgen solcher Flächen und der Bestimmung deren Position zur Kamera beschäftigt, wie er in der Literatur ebenfalls häufig betrachtet wird[61],[56],[62], häufig auch als Hyperebene eines dreidimensionalen Objektes [63]. Dies soll insbesondere als ein weiterer Anwendungsfall der gleichen Prinzipien zur Skalierbarkeit des Grundalgorithmus dienen, wie sie im vorangegangenen Abschnitt anhand der Disparitätsschätzung dargestellt wurden.

4.1. Prinzip

Bei Verwendung der homogenen Bildkoordinaten $\mathbf{x} = (x_1, x_2, x_3)^T$ wird die perspektivische Abbildung der Pixel einer planaren Fläche durch die Homographie $\mathbf{H}$ beschrieben [9]:

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \mathbf{x}' = \mathbf{H}\mathbf{x} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}. \tag{4.1}$$

Die Matrix $\mathbf{H}$ besitzt acht Freiheitsgrade, einen für jedes Matrixelement abzüglich einem für eine beliebige Skalierung der Matrix. Es kann gezeigt werden, daß vier bekannte Punktkorrespondenzen aus zwei Bildern ausreichen, um die der perspektivischen Transformation zugrundeliegende Homographiematrix zu bestimmen [9]. Als Näherung der Homographie kann die affine Transformation verwendet werden, die mit dem Verschiebungsvektor $\mathbf{s}$ in homogenen Bildkoordinaten

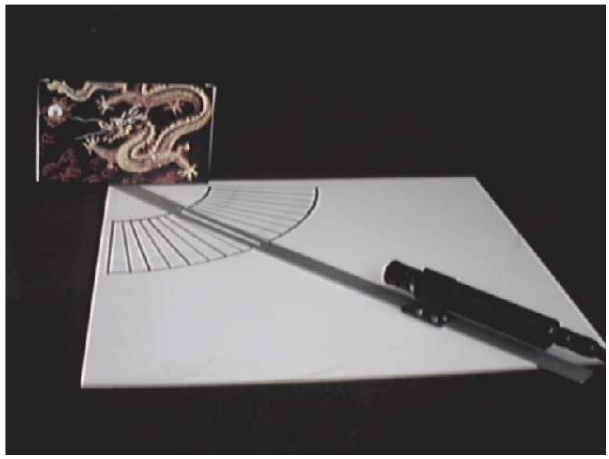
dargestellt werden kann als:

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \mathbf{x}' = \begin{bmatrix} \mathbf{A} & \mathbf{s} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} a_1 & a_2 & s_x \\ a_3 & a_4 & s_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}. \tag{4.2}$$

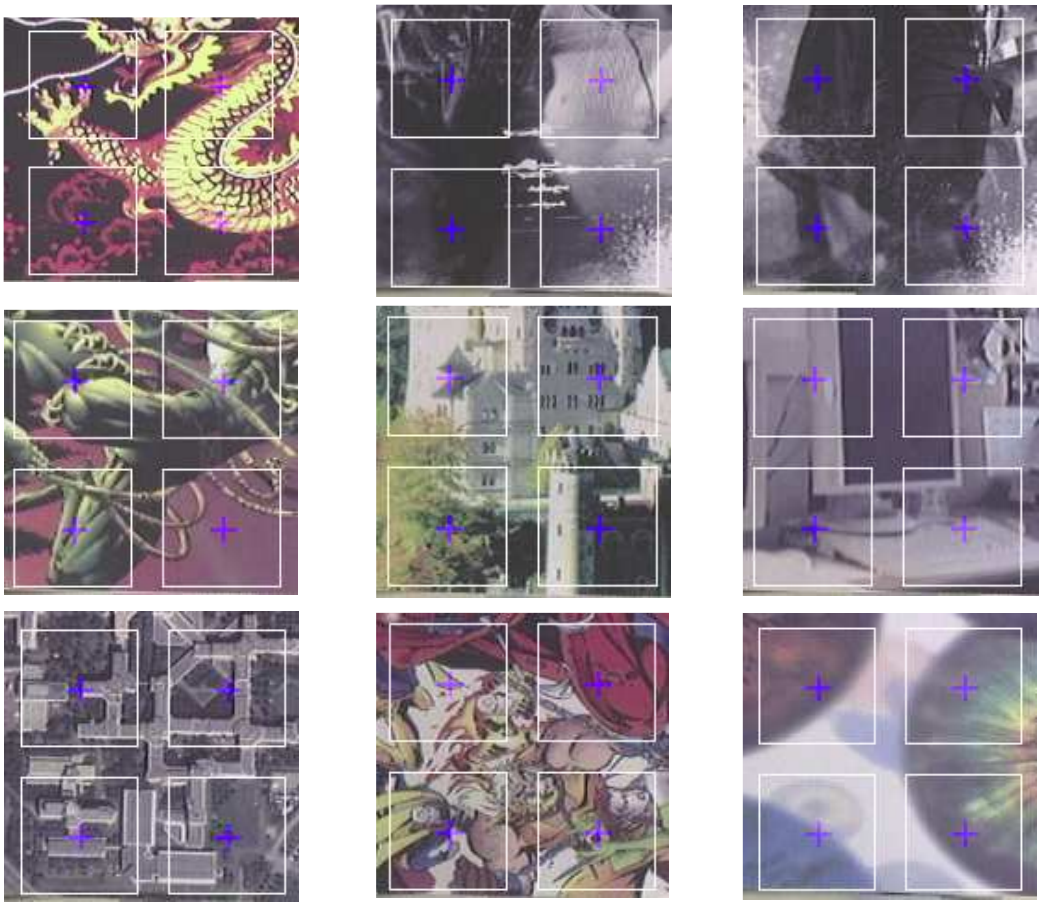
Die affine Transformation besitzt sechs Freiheitsgrade und kann aus drei Punktkorrespondenzen zweier Bilder berechnet werden.
Algorithmen zum Mitverfolgen planarer Flächen haben eine Berechnung entsprechender Transformationsforschrift zum Ziel, da diese zur perspektivischen Anpassung auf die einzublendenden Texturen angewandt werden muß. Wie auch die Disparitätsschätzung (vgl. Abschnitt 3), unterliegen diese Verfahren den in Abschnitt 2 angesprochenen Grundprinzipien von Algorithmen zur Videoanalyse. Da hier, anders als bei einer Disparitätsschätzung, keine dichte Fläche im Bild mit Berechnungswerten abgedeckt werden muß, sondern prinzipiell lediglich vier Punktkorrespondenzen zu einer vollständigen Berechnung ausreichen, dominieren beim Tracking punktbasierte Algorithmen [64]. Ob es sich um das Mitverfolgen planarer oder dreidimensionaler Objekte handelt, spielt dabei erst beim rekonstruierten Bewegungsmodell eine Rolle. Entsprechend den in Abschnitt 1 beschriebenen, aufgrund der Mobilität hohen Zeitanforderungen an das System, soll auch hier ein einfacher, blockbasierter Algorithmus verwendet werden. Die hohe Skalierbarkeit dieser Gruppe von Algorithmen hat sich bereits im vorangegangenen Abschnitt gezeigt, sodaß dieser Abschnitt als Nachweis für die Übertragbarkeit der dort angewandten Prinzipien auf andere Anwendungen gesehen werden soll.

4.2. Testsequenzen

Als Testsequenzen dienen unter Verwendung der in Abbildung 4.1 (a) gezeigten Apparatur aufgenommene Videos mit einem von 0° (Bildebene parallel zur Objektebene) bis zu 90° (Bildebene senkrecht zur Objektebene) steigendem Aufnahmewinkel bezüglich der in Abbildung 4.1 (b) dargestellten Testbilder. Die zeitlichen Angaben beziehen sich dabei wiederum auf einen Pentium M mit einer Taktfrequenz von 1,7 GHz (PC) und den typischen Mobiltelefonprozessor ARM9 mit einer Taktfrequenz von 150 MHz (Mobilgerät).



(a)



(b)

Abbildung 4.1.: (a)Apparat zur Aufnahme von Testsequenzen unter variablem Betrachtungswinkel der Testbilder. (b) Testbilder mit eingetragenen Referenzblöcken (weiß).

4.3. Fehlerkriterium

Um die Bewertung der Qualität eines bestimmten Algorithmus zu ermöglichen, muß zunächst ein geeignetes Fehlermaß eingeführt werden. Ein Algorithmus zum Mitverfolgen von Punkten ist je genauer, desto geringer die Abweichung der berechneten Punktpositionen von den tatsächlichen Punktpositionen ist. So sollen als „Ground Truth“ die per Hand bestimmten oder über einen präzisen Algorithmus berechneten Koordinaten der betrachteten Punkte in allen Einzelbildern einer Testsequenz verwendet werden. Der Fehler eines bestimmten Algorithmus bezüglich dieser Sequenz soll sich dann aus der Summe der Euklid'schen Distanzen zwischen berechneten Punkten (x_i, y_i) und deren „Ground Truth“ (x'_i, y'_i) über alle Punkte i und Einzelbilder f einer Sequenz ergeben:

$$\sum_f \sum_i \sqrt{(x_i - x'_i)^2 + (y_i - y'_i)^2}. \tag{4.3}$$

4.4. Verwendeter Algorithmus

Im Folgenden soll lediglich ein Mitverfolgen planarer Objekte betrachtet werden. Es werden entsprechend eine anfängliche Parallelität der mitzuverfolgenden Fläche zur Bildebene und die Bekanntheit zum Mitverfolgen geeigneter Punkte vorausgesetzt.
Als effizientes Mittel zur Bestimmung einer Punktkorrespondenz hat sich das Block Matching herausgestellt. Dabei wird um einen mitzuverfolgenden Punkt $\mathbf{x}_r$ im Ursprungsbild $f_r(x, y)$ ein quadratisches Fenster ungerader Seitenlänge k betrachtet, das dessen Umgebung darstellt. Diese Bildregion wird innerhalb eines Suchbereiches der Größe $s_0 \times s_0$ um entsprechende Stelle im aktuellen Vergleichsbild $f_s(x, y)$ anhand eines Bewertungskriteriums (SAD, vgl. Abschnitt 3.2.2) verglichen, sodaß sich die neuen Punktkoordinaten $\mathbf{x}_s$ ergeben:

$$\mathbf{x}_s = \mathbf{x}_{s-1} + \arg\min_{\mathbf{s}} \sum_{i=-\frac{k-1}{2}}^{\frac{k-1}{2}} \sum_{j=-\frac{k-1}{2}}^{\frac{k-1}{2}} |f_r(x_{s-1}, y_{s-1}) - f_s(x_{s-1} + s_x, y_{s-1} + s_y)|$$
$$mit\ s_x, s_y \in \{-\frac{s_0}{2}, -\frac{s_0}{2} + 1, \ldots, -1, 0, 1, \ldots, \frac{s_0}{2} - 1, \frac{s_0}{2}\}. \tag{4.4}$$

Das Bild $f_{s-1}(x, y)$ ist dabei das vorhergehende Bild, für das die Koordinaten der mitzuverfolgenden Punkte bestimmt wurden.

Es sind folgende Annahmen zu treffen:

1. Die Ähnlichkeit eines Blocks im Ursprungsbild zum korrespondierenden Block im aktuellen Bild nimmt mit zunehmender perspektivischer Verzerrung stark ab.

2. Die Unterschiede der Verzerrungen, die ein Block in aufeinanderfolgenden Einzelbildern erfährt, sind sehr gering.

Um eine höhere Ähnlichkeit zwischen den Blöcken im Ursprungsbild und den Blöcken im Vergleichsbild soll die für das vorangegangene Einzelbild $f_{s-1}(x, y)$ berechnete Transformationsmatrix $\mathbf{H}_{s-1}$ als Schätzung der zu berechnenden Transformationsmatrix $\mathbf{H}_s$ des aktuellen Bildes $f_s(x, y)$ verwendet werden. Hierzu werden die Blöcke im Vergleichsbild entsprechend modifiziert, sodaß sich Gleichung 4.4 ändert in:

$$\mathbf{x}_s = \mathbf{x}_{s-1} + \arg\min_{\mathbf{s}} \sum_{i=-\frac{k-1}{2}}^{\frac{k-1}{2}} \sum_{j=-\frac{k-1}{2}}^{\frac{k-1}{2}} \Bigg| f_r(x_r, y_r) - f_s\left(x_{s-1} + \frac{h_1 i + h_2 j + h_3}{h_7 i + h_8 j + h_9} + s_x, y_{s-1} + \frac{h_4 i + h_5 j + h_6}{h_7 i + h_8 j + h_9} + s_y\right) \Bigg|. \quad (4.5)$$

Als Näherung der Homographie soll ebenfalls eine affine Transformation untersucht werden:

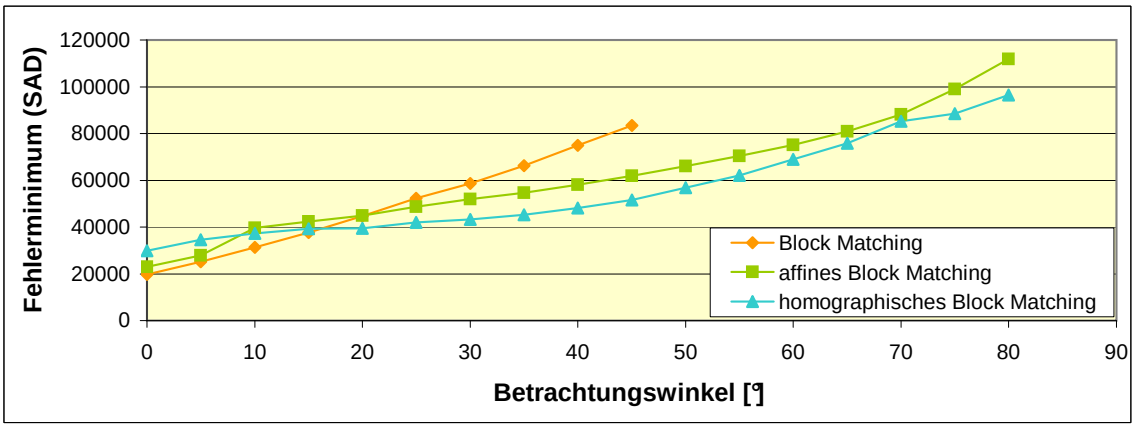
$$\mathbf{x}_s = \mathbf{x}_{s-1} + \arg\min_{\mathbf{s}} \sum_{i=-\frac{k-1}{2}}^{\frac{k-1}{2}} \sum_{j=-\frac{k-1}{2}}^{\frac{k-1}{2}} \Big| f_r(x_r + i, y_r + j) - f_s\left(x_{s-1} + (a_1 i + a_2 j) + s_x, y_{s-1} + (a_3 i + a_4 j) + s_y\right) \Big|. \quad (4.6)$$

Als ein Maß für die Ähnlichkeit eines einzelnen Blocks mit dem tatsächlichen Bildinhalt soll zunächst das jeweilige Minimum der SAD beim Blockvergleich betrachtet werden. Abbildung 4.2 (a) zeigt dabei den entsprechenden Mittelwert über alle Testsequenzen in Abhängigkeit vom Betrachtungswinkel. Wie zu erwarten, zeigt sich hier bei geringen Betrachtungswinkeln für alle Algorithmen eine hohe Ähnlichkeit, die bei steigenden Winkeln abnimmt. Interessant ist hierbei, daß sich für einen Betrachtungswinkel von 0° die Ähnlichkeit genau umgekehrt zur Komplexität der Algorithmen verält, der kleinste Fehler ergibt sich also für das starre Block Matching und der größte für die exakte Modellierung einer perspektivischen Verzerrung durch Berechnung der Homographie. Zurückzuführen

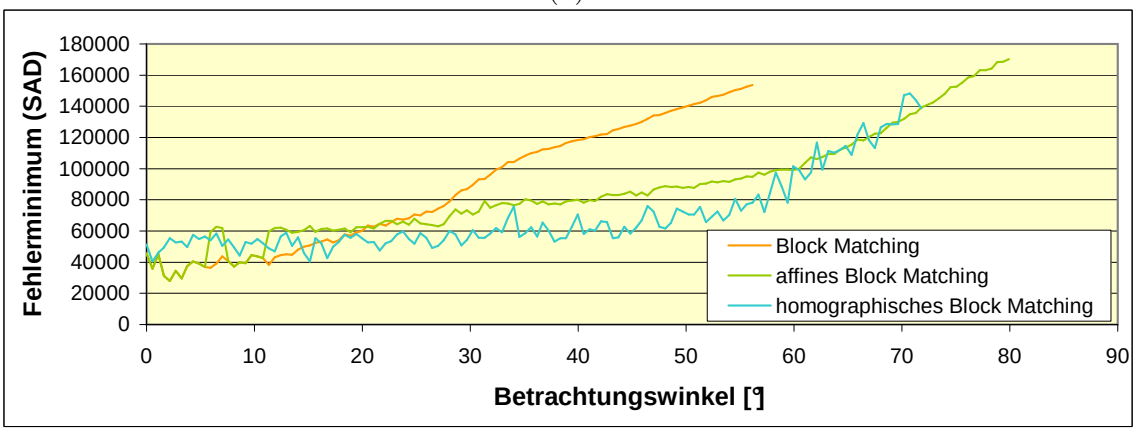
ist dieser Effekt auf die unterschiedliche Anzahl von Freiheitsgraden, die beim starren Block Matching die Möglichkeiten, Fehler zu machen stark einschränkt. Erst bei größeren Betrachtungswinkeln kann die Modellierung der perspektivischen Verzerrung ihre Wirkung entfalten, und zwar wie zu erwarten mit einer höheren Ähnlichkeit des homographischen Matchings. Das starre Block Matching verliert hier ab einem Winkel von 45° die mitzuverfolgenden Punkte und kann somit für höhere Betrachtungswinkel nicht verwendet werden.
Abbildung 4.2 (b) zeigt denselben Verlauf der Blockvergleichs-Fehlerminima für eine Einzelsequenz. Hier fällt ein sehr starkes Schwanken der SAD bei Berechnung der Homographie auf, das in der vorangegangenen Abbildung herausgemittelt wurde. Dieses Schwanken spiegelt sich bei Betrachtung des berechneten Ergebnisvideos in einem extrem unruhigen Verlauf der berechneten Punktpositionen wieder und läßt sich wie auch die Anfangspositionen auf die hohe Anzahl von Freiheitsgraden bei der Homographieberechnung zurückführen. Insbesondere beim Einblenden einer virtuellen Textur anhand der berechneten Werte wirk dieser Effekt mehr als störend und schließt somit eine Verwendung der Homographie zur Anpassung der verwendeten Blockformen aus.

4.5. Skalierungsparameter

Da es sich dabei ebenfalls um einen blockbasierten Algorithmus handelt, können viele Skalierungsparameter von der Disparitätsschätzung übernommen werden (vgl. Abschnitt 3.6). Als Skalierungsparameter sollen entsprechend die Blockgröße k, die Blockabtastperiode a und die Anzahl der logarithmischen Suchschritte l verwendet werden. Abbildung 4.3 zeigt die entsprechenden Zeit- und Fehlerverläufe der Einzelparameter und Abbildung 4.4 die der Parameterkombinationen.



(a)



(b)

Abbildung 4.2.: Verlauf des Fehlerminimums beim Blockvergleich (gemittelt über alle Blöcke pro Einzelbild) in Abhängigkeit vom Betrachtungswinkel (a) gemittelt über alle Testsequenzen und (b) für eine Einzelsequenz.

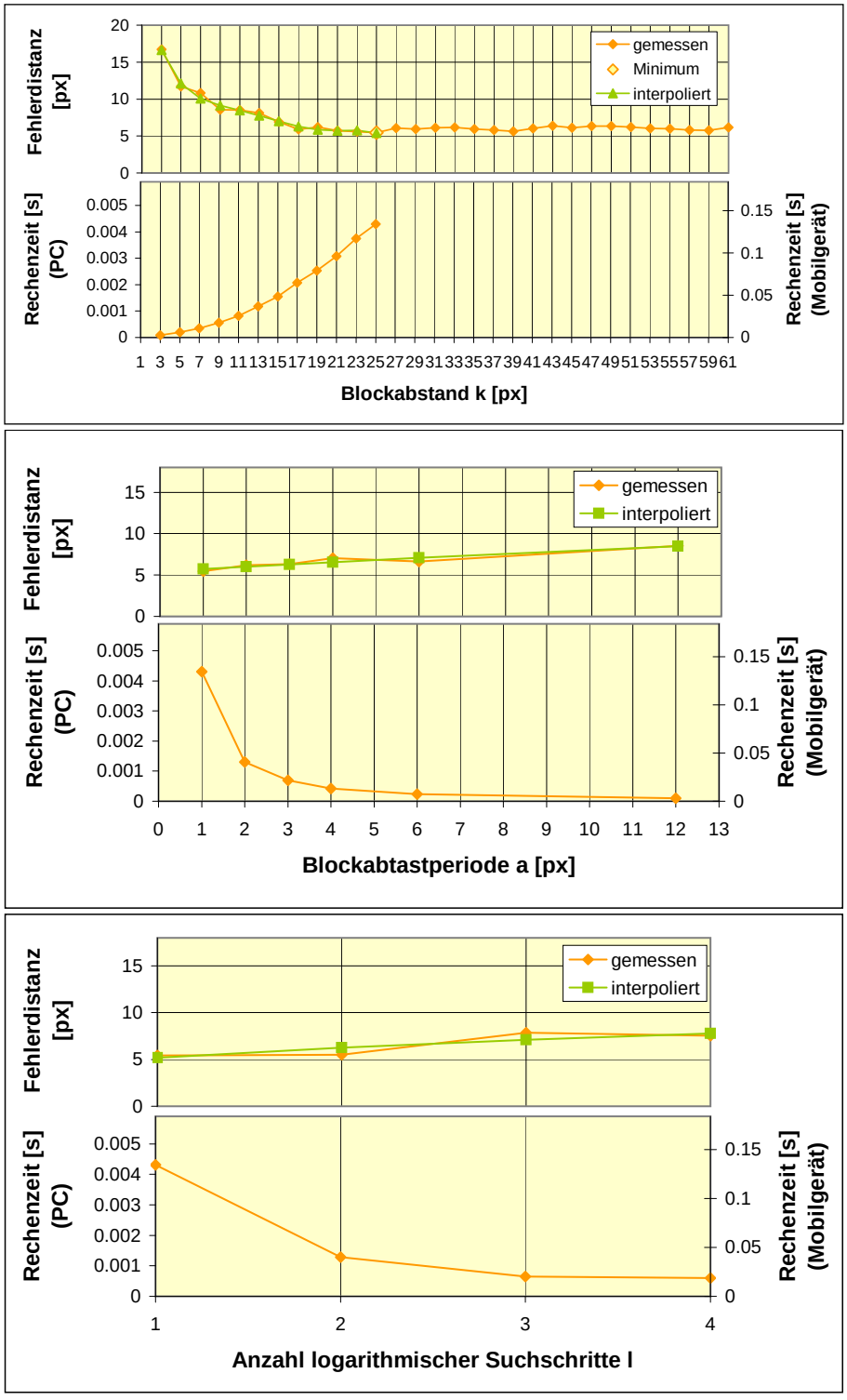


Abbildung 4.3.: Zeit- und Fehlerverhalten der Einzelparameter.

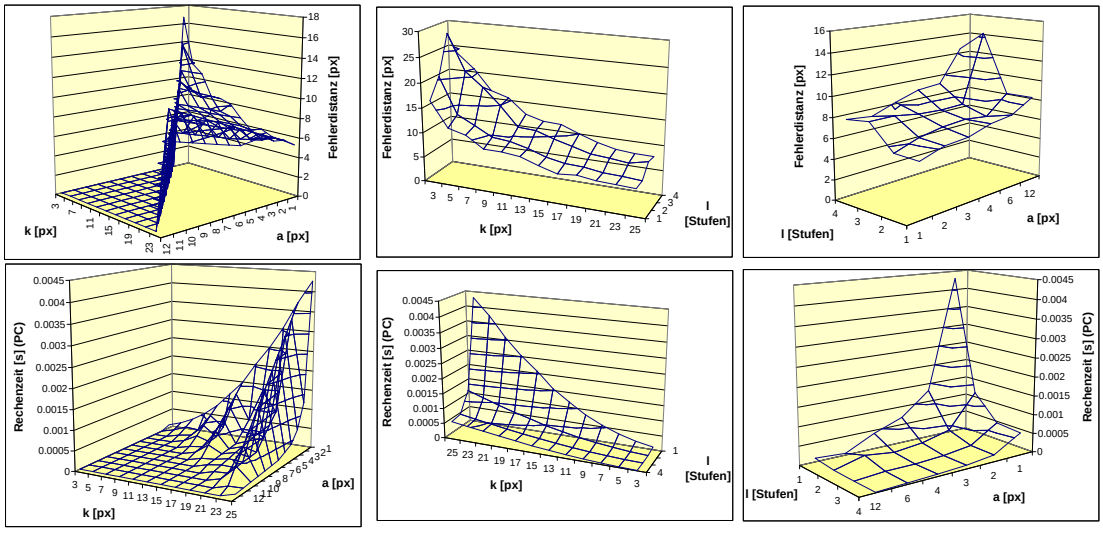


Abbildung 4.4.: Fehler-(oben) und Zeitverhalten (unten) der paarweise verknüpften Skalierungsparameter.

4.6. Fehlermodellierung

Aufgrund der geringeren Anzahl von Skalierungsparametern kann hier im Gegensatz zur Disparitätsschätzung auf eine zusätzliche Modellierung des Zeitverhaltens verzichtet und alle kombinatorischen Parameterkonstellationen getestet werden. Da aufgrund der geringen Anzahl von Testsequenzen die Ergebniskurven bezüglich des Fehlers nicht die gewünschte Glattheit aufweisen, soll hier jedoch auf die Fehlerverläufe der Einzelparameter zusätzlich eine polynomiale Interpolation angewandt und die endgültige Skalierungsvorschrift mit den so geglätteten Werten berechnet werden. Entsprechend gilt es zunächtst, die geeignete Verknüpfung der Fehler in Abhängigkeit von den einzelnen Skalierungsparameter zu bestimmen. Da es sich hier, ungleich der Disparitätsschätzung, nicht um Fehlerwahrscheinlichkeiten handelt, sondern um mittlere Fehlerdistanzen, kommt hier keine statistische Verknüpfung zum Tragen. Vielmehr zeigt sich anhand des geringen relativen Fehlers bei einer Berechnung aller Parameterverknüpfungen, wie er in Abbildung 4.4 dargestellt ist, daß eine multiplikative Verknüpfung der einzelnen Werte angewandt werden kann.

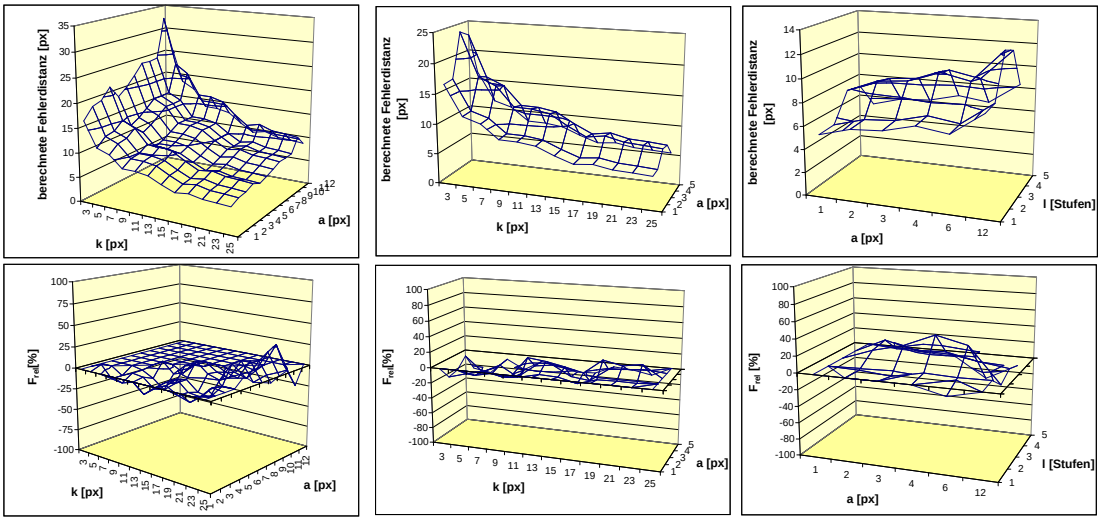


Abbildung 4.5.: Aus den Fehlerverläufen der Einzelparameter multiplikativ berechnete Fehler für alle paarweisen Parameterverknüpfungen (oben) und entsprechende relative Fehler bezüglich den gemessenen Werten aus Abbildung 4.4(unten).

4.7. Herleitung der Skalierungsvorschrift

Entsprechend Abschnitt 3.7.7 werden die Ergebnisse aller Zustandskombinationen der Parameter berechnet und zeitlich geordnet. Ineffiziente Zustandskombinationen, also solche, deren Fehlerdistanz durch andere Zustandskombinationenen bei geringerer Rechenzeit erreicht oder unterboten wird, werden anschließend eliminiert (Pareto-Optimierung), sodaß von ursprünglich 170 Zustandskombinationen die 26 effizientesten übrig bleiben. Diese sind in Tabelle 4.1 aufgelistet und in Abbildung 4.6 als Zeit-/Fehlerdiagramm dargestellt.

#	k	a	l	Fehlerdistanz [px]	Rechenzeit [s] (PC)	Rechenzeit [s] (Mobilgerät)
1	9	4	4	15.72476255	0.00000746	0.000232582
2	9	4	3	14.3858606	0.00000818	0.000254863
3	11	5	3	13.90055455	0.00000940	0.000292885
4	13	6	3	13.23627964	0.00000960	0.000299107
5	15	7	3	12.33813939	0.00001142	0.000356096
6	19	9	4	11.9939325	0.00001354	0.000422162
7	17	8	3	11.46422357	0.00001373	0.000428063
8	19	9	3	10.97269612	0.00001484	0.000462605
9	15	7	2	10.8427187	0.00002259	0.000704
10	17	8	2	10.07472418	0.00002715	0.000846278
11	19	9	2	9.642771381	0.00002934	0.000914566
12	25	6	3	9.183472978	0.00003498	0.001090218
13	17	4	2	8.716947381	0.00006066	0.001890622
14	25	4	3	8.503514614	0.00006359	0.001982215
15	25	6	2	8.070407626	0.00006915	0.002155355
16	19	9	1	8.004158274	0.00009802	0.003055177
17	19	3	2	7.752851625	0.00012283	0.003828545
18	25	4	2	7.47286232	0.00012573	0.003918827
19	17	4	1	7.235661176	0.00020263	0.006315763
20	25	3	2	7.162428634	0.00020865	0.006503386
21	25	6	1	6.698989058	0.00023100	0.007200122
22	19	3	1	6.435395907	0.00041032	0.012789539
23	25	4	1	6.202985678	0.00042000	0.013091131
24	25	3	1	5.945304534	0.00069700	0.021725043
25	25	2	1	5.681170425	0.00130300	0.040613675
26	25	1	1	5.410583345	0.00430600	0.13421526

Tabelle 4.1.: Effizienteste Zustanskombinationen der Parameter zur Skalierung des planaren Trackings.

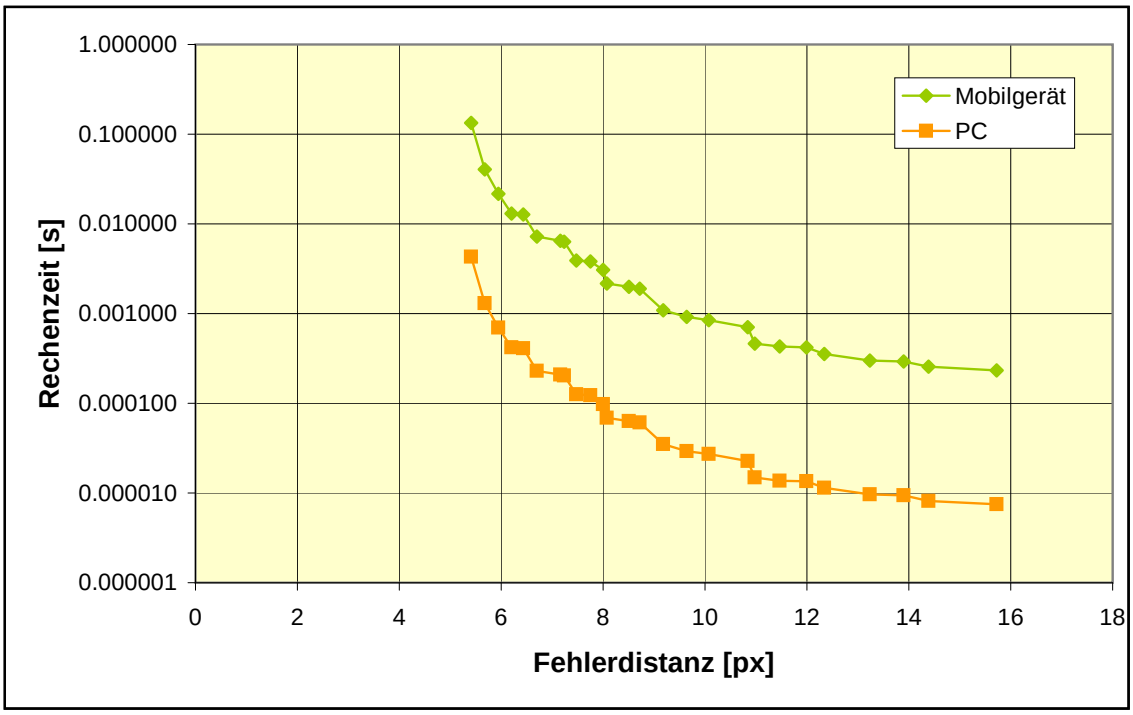


Abbildung 4.6.: Zusammenhang zwischen Aufwand (Rechenzeit) und Fehler für die effizienten Kombinationen der betrachteten Skalierungsparameter.

5. Programmsteuerung

Dieser Abschnitt soll aufzeigen, wie basierend auf einer Skalierungsvorschrift (vgl. Abbildungen 3.55 und 4.6 bzw. Tabellen 3.3 und 4.1) der entsprechende Algorithmus effizient, also unter stets optimaler Nutzung der zur Verfügung stehenden Ressourcen eingesetzt werden kann. Hierzu soll der in Abbildung 5.1 dargestellte Regelkreis betrachtet werden.

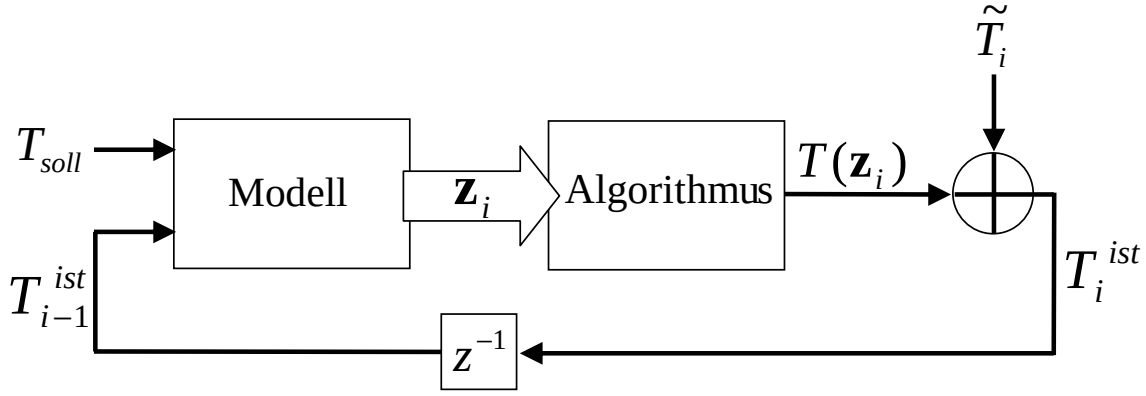


Abbildung 5.1.: Darstellung des Gesamtsystems als Regelkreis mit dem Algorithmus und dem entsprechenden Modell. z^{-1} ist dabei das „z-Verzögerungsglied“ der z-Transformation.

Bei einer einzustellenden Einzelbildrate f_{soll} ergibt sich die entsprechende Rechenzeit pro Einzelbild zu $T_{soll} = \frac{1}{f_{soll}}$. Voraussetzung einer Aufwandsanpassung ist die Zeitmessung zu Beginn und Ende der Berechnung jedes Disparitätsbildes. So kann einerseits der Rechenzeitbedarf $T(\mathbf{z}_i)$ des Algorithmus im momentanen Parameterzustand, andererseits über die Endzeit der vorangegangenen und Anfangszeit der aktuellen Berechnung der aktuelle Rechenzeitbedarf $\tilde{T}_i$ der Fremdalgorithmen bestimmt werden. Die „soll“-Rechenzeit des Algorithmus ergibt sich

analog aus der „soll“-Rechenzeit eines Einzelbildes T_{soll} zu:

$$T_{soll}^{alg} = T_{soll} - \tilde{T}_{i-1}. \tag{5.1}$$

Eine wichtige Eigenschaft der Skalierungsvorschrift ist die Tatsache, daß für eine höhere Rechenleistung stets eine höhere Qualität erreicht wird. Wird eine bestimmte Rechenzeit eingestellt, ist dabei eine optimale Qualität stets gewährleistet.

Über Vergleich des im vorangegangenen Einzelbild verwendeten Zustandsvektors $\mathbf{z}_{i-1}$ mit denen der Liste wird aus dieser die Referenzrechenzeit $\check{T}(\mathbf{z}_{i-1})$ ausgelesen. Mit der Rechenzeit $T(\mathbf{z}_{i-1})$ des Algorithmus im Vorgängerbild kann der entsprechende Soll-Zeitwert $\check{T}_{soll}$ der Skalierungsvorschrift bestimmt werden:

$$\check{T}_{soll} = T_{soll}^{alg} \frac{\check{T}(\mathbf{z}_{i-1})}{T(\mathbf{z}_{i-1})}. \tag{5.2}$$

Es gilt also in der Liste der Skalierungsvorschrift den Zeitwert zu finden, der $\check{T}_{soll}$ am nächsten kommt:

$$\begin{aligned} \mathbf{z}_i &= \arg\min_{\mathbf{z}'} \left| \check{T}_{soll} - \check{T}(\mathbf{z}') \right| \\ &= \arg\min_{\mathbf{z}'} \left| T_{soll} \frac{\check{T}(\mathbf{z}_{i-1})}{T(\mathbf{z}_{i-1})} - \check{T}(\mathbf{z}') \right| \ , \ mit\ \check{T}(\mathbf{z}') \leq \check{T}_{soll}. \end{aligned} \tag{5.3}$$

Im Einzelbild i wird dann entsprechend der Algorithmus unter Verwendung der Parametereinstellungen $\mathbf{z}_i$ durchgeführt.

Abbildung 5.2 (links) zeigt das Zeitverhalten einer statischen Disparitätsschätzung mit dem festen Parameterzustand 110 (vgl. Tabelle 3.3). Während der Laufzeit eines höherprioren Fremdprozesses kann dabei der deutlich höhere Rechenzeitaufwand der Disparitätsschätzung mit der entsprechend starken Verletzung der Echtzeitbedingungen festgestellt werden.

Abbildung 5.2 (rechts) hingegen zeigt dieselbe Funktionalität bei Einsatz der Programmsteuerung. Das Zeitverhalten zeigt sich dabei auf Kosten einer leicht ansteigenden Fehlerwahrscheinlichkeit vom höherprioren Fremdprozess unbeeinflußt.

Die zeitlichen Grundschwankungen folgen in beiden Fällen aus zusätzlichen Hintergrundprozessen (z.B. Betriebssystem) und werden im Falle der Programmsteuerung durch deren Kompensationsverhalten leicht verstärkt.

Hier gilt es zu beachten, daß die Anwendung selbst zwar keinen „harten“ Echtzeitanforderungen unterliegt, da die Disparitätsschätzung für jedes Einzelbild separat durchgeführt wird. Darauf aufbauende Funktionalitäten jedoch können sehr

wohl „harten“ Echtzeitanforderungen unterliegen, sodaß diese ausschließlich mit einer rechenleistungsadaptiven Disparitätsschätzung realisierbar sind.

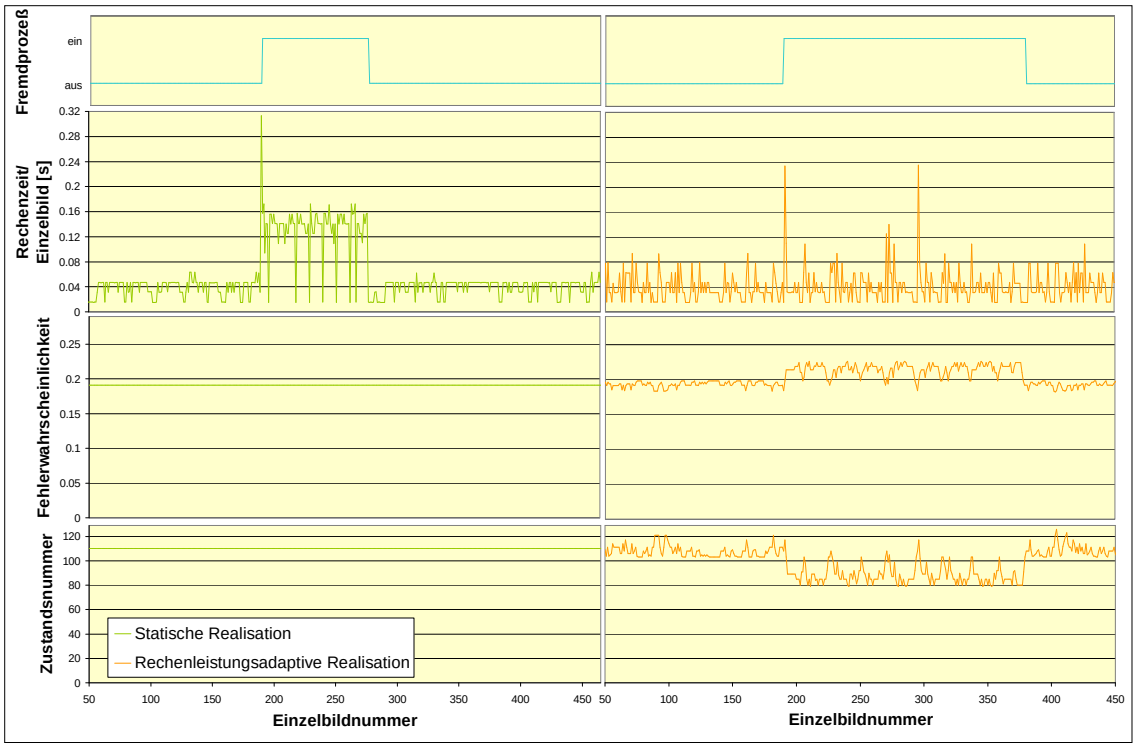


Abbildung 5.2.: Zeitverhalten und Fehlerwahrscheinlichkeit einer statisch (links) und rechenleistungsadaptiv (rechts) realisierten Disparitätsschätzung bei gleichzeitigem Fremdprozeß.

6. Zusammenfassung und Ausblick

Ein grundlegendes Problem der Augmented Reality ist der gleichzeitig hohe Bedarf an Mobilität und Rechenleistung, die jedoch insbesondere aufgrund der nötigen Echtzeitfähigkeit nur schwer miteinander vereinbar sind. Der Hauptgrund hierfür liegt in der Art der verwendeten Algorithmen, die üblicherweise einen festen Aufwand besitzen und nicht anwendbar sind, wenn dieser durch ein System nicht abgedeckt wird. Entsprechend wurde in dieser Arbeit ein Weg aufgezeigt, wie für grundlegende Funktionalitäten der Videoanalyse stets eine der verfügbaren Rechenleistung adäquate Programmlaufzeit zur Einhaltung der Echtzeitbedingung erreicht werden kann.
Für eine Disparitätsschätzung wurden ausgehend von geeigneten Algorithmen Parameter eingeführt, die deren systematische und vorhersehbare Skalierung während des Programmlaufs ermöglichen. Dabei wurden einerseits bereits existierende Beschleunigungs- sowie Qualitätsvervesserungsmethoden angewandt, andererseits neue eingeführt, wie etwa der Blockabstand oder die Schätzbereichsgröße. Ferner wurde ein neues, skalierbares Verfahren zur Korrektur von Luminanzschwankungen im Disparitätsraum hergeleitet und ebenso als Skalierungsparameter eingesetzt.
Über statistische Aussagen wurde entsprechend ein Modell für das Fehlerverhalten hergeleitet, auch mit Hinblick auf eine Auskoppelung oder ein nachträgliches Hinzufügen einzelner Skalierungsparameter. Bezüglich des Zeitverhaltens wurden zwei alternative Modellierungsverfahren hergeleitet und verglichen, einerseits eine implizite Modellierung durch Berechnung aus den Zeitmessungen der Einzelparameter, andererseits eine explizite strukturelle Modellierung der zugrundeliegenden Algorithmen. Beide ermöglichen relative Aussagen über das Zeitverhalten bezüglich einer gemessenen Referenzzeit, wobei sich die implizite Modellierung als die exaktere herausgestellt hat.
Die Robustheit des Verfahrens gegenüber einer Portierung auf andere Systeme, und damit die Allgemeingültigkeit der aufgestellten Modelle, wurde anhand der

Analyse des Ressourcenbedarfs bezüglich der kritischen Komponenten „Prozessor“ und „Speicher“ nachgewiesen.
Aus allen im Rahmen der Skalierungsparameter möglichen Zuständen mit den zugehörigen Fehlerwahrscheinlichkeiten und Rechenzeiten wurden ineffiziente Kombinationen eliminiert und für die verbleibenden, effizienten Kombinationen wurde als Skalierungsvorschrift ein Zeit-Fehler-Diagramm mit den zugehörigen Einstellungen der Skalierungsparameter aufgestellt.
Zur Veranschaulichung der Verallgemeinerbarkeit des vorgestellten Ansatzes, insbesondere bei blockbasierten Algorithmen zur Videoanalyse, wurde dasselbe Vorgehen analog auf ein Mitverfolgen planarer Objekte übertragen und auch diesbezüglich eine Skalierungsvorschrift aufgestellt.
Schließlich wurde aufgezeigt, wie die entsprechenden Funktionalitäten mithilfe der zugehörigen Skalierungsvorschriften effizient realisiert werden können, sodaß bei einer vorgegebenen Rechenleistung stets das optimale Ergebnis erzielt wird. Das Erreichen des angestrebten Verhaltens konnte schließlich anhand Meßwerten der resultierenden Echtzeitanwendung nachgewiesen werden.
Zukünftige Arbeiten könnten sich hier auf eine Automatisierung der vorgestellten Entwurfsmethodik beziehen, wie sie in Abschnitt 3.7.7.3 angesprochen wurde. Ferner bietet die Strategie zur Auswahl des neuen Zustandes aus der Skalierungsvorschrift in Hinblick auf das angestrebte Zeitverhalten Spielraum für weitere Betrachtungen.

A. Appendix

A.1. Herleitung der Gesamtdisparität

A.1.1. Kartesische Koordinaten

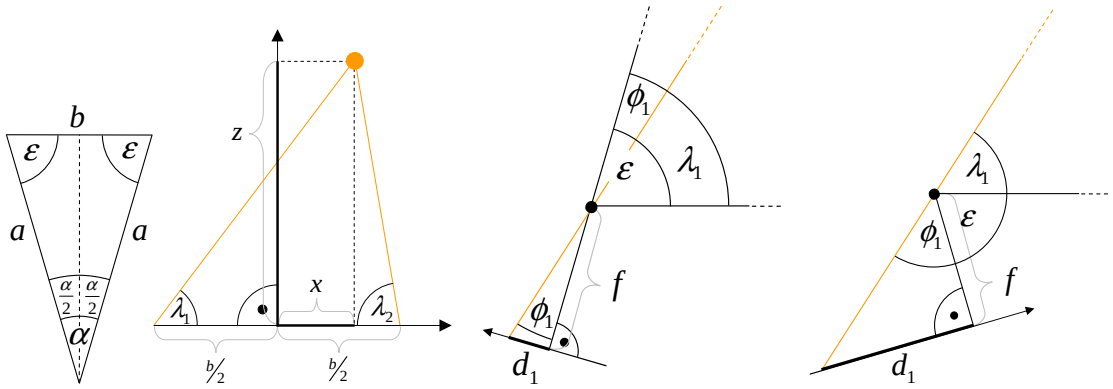


Abbildung A.1.: (a) Gleichschenkeliges Dreieck zwischen den Kameraarmpositionen bzw. optischen Achsen der beiden Kameras und der Basislinie b. (b) Dreieck zwischen Basislinie und den Objektstrahlen (orange) beider Kameras. (c) Geometrie einer Kamera im Stereosystem. (d) Geometrie einer Kamera beim Kameraschwenk.

Abbildung A.1 (a) zeigt das gleichschenkelige Dreieck zwischen den beiden Positionen des Kameraarms a beim Kameraschwenk und der Basislinie b. Diese Anordnung kann analog auf das klassische Stereosystem übertragen werden, wenn sich die optischen Achsen der beiden Kameras im Abstand a unter dem Winkel α schneiden. Die Basislinie b soll dabei durch die beiden Kamerabrennpunkte

verlaufen und ergibt sich zu:

$$b = 2a \sin \frac{\alpha}{2}. \tag{A.1}$$

Für die beiden Seitenwinkel ϵ ergibt sich:

$$\epsilon = \frac{\pi - \alpha}{2}. \tag{A.2}$$

Aus Abbildung A.1 (b) ergibt sich für die Seitenwinkel λ im Dreieck zwischen Objektstrahlen und Basislinie:

$$\lambda_1 = \arctan \frac{z}{\frac{b}{2} + x}, \tag{A.3}$$

$$\lambda_2 = \arctan \frac{z}{\frac{b}{2} - x}. \tag{A.4}$$

Für den Winkel ϕ_i zwischen optischer Achse und Objektstrahl ergibt sich für das echte Stereosystem nach Abbildung A.1 (c):

$$\phi_i = \epsilon - \lambda_i. \tag{A.5}$$

Beim Kameraschwenk berechnet sich der Winkel ϕ_i entsprechend Abbildung A.1 (c) zu:

$$\phi_i = \pi - \epsilon - \lambda_i. \tag{A.6}$$

Die Teildisparität d_i ergibt sich für beide Fälle wiederum analog mit der Brennweite f (vgl. Abbildungen A.1 (c) und (d)):

$$d_i = f \tan \phi_i. \tag{A.7}$$

Die Gesamtdisparität $d_{stereo}(x, z, \alpha)$ ergibt sich für das Stereosystem nach den Gleichungen A.1,A.2,A.3,A.4,A.5 und A.7 somit zu:

$$\begin{aligned} d_{stereo}(x, z, \alpha) &= d_{stereo,1} + d_{stereo,2} \\ &= f \tan \left(\epsilon - \arctan \frac{z}{a \sin \frac{\alpha}{2} + x} \right) \\ &+ f \tan \left(\epsilon - \arctan \frac{z}{a \sin \frac{\alpha}{2} - x} \right). \end{aligned} \tag{A.8}$$

Die Gesamtdisparität $d(x, z, \alpha)$ für den Kameraschwenk ergibt sich analog nach den Gleichungen A.1,A.2,A.3,A.4,A.6 und A.7 zu:

$$\begin{aligned} d_{schwenk}(x, z, \alpha) &= d_{schwenk,1} + d_{schwenk,2} \\ &= f \tan \left(\pi - \epsilon - \arctan \frac{z}{a \sin \frac{\alpha}{2} + x} \right) \\ &+ f \tan \left(\pi - \epsilon - \arctan \frac{z}{a \sin \frac{\alpha}{2} - x} \right). \end{aligned} \tag{A.9}$$

A.1.2. Kreiskoordinaten

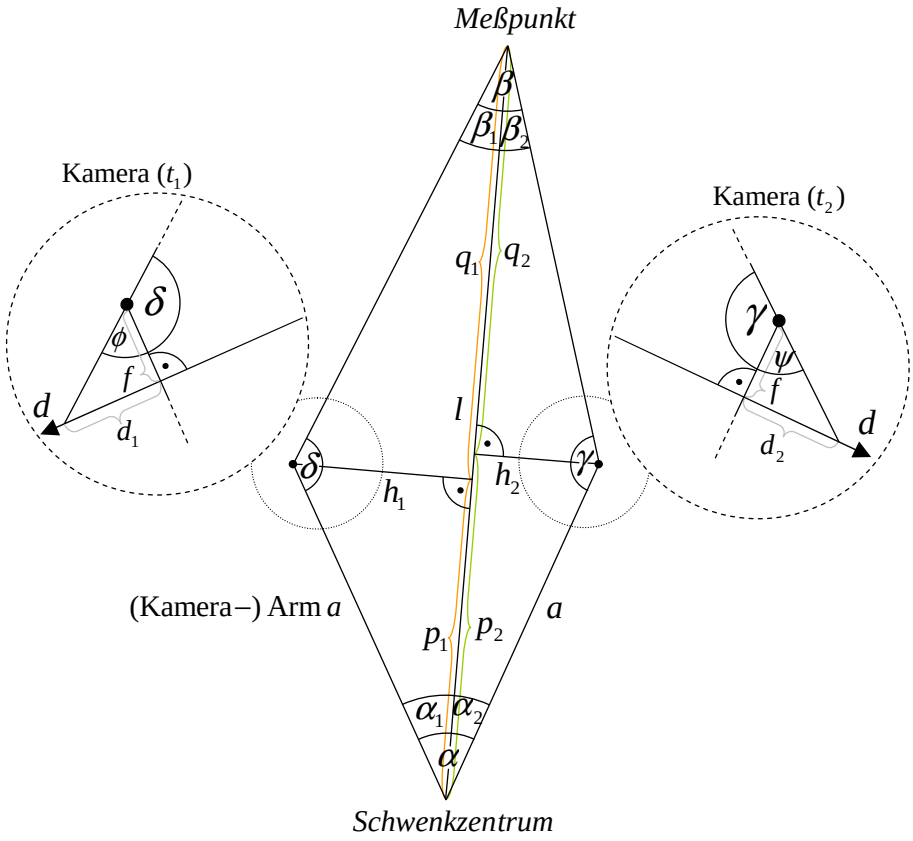


Abbildung A.2.: Geometrie der typischen Bewegung eines Mobilgerätes.

Abbildung A.2 zeigt die Geometrie des in 3.1 eingeführten typischen Bewegungsverhaltens eines Mobibilgerätnutzers. Daraus ergeben sich folgende Zusammenhänge:

$$\beta_1 = \pi - \delta - \alpha_1, \quad \text{(A.10)} \qquad \beta_2 = \pi - \gamma - \alpha_2, \quad \text{(A.15)}$$

$$\frac{h_1}{p_1} = \tan\alpha_1, \quad \text{(A.11)} \qquad \frac{h_2}{p_2} = \tan\alpha_2, \quad \text{(A.16)}$$

$$\frac{h_1}{q_1} = \tan\beta_1, \quad \text{(A.12)} \qquad \frac{h_2}{q_2} = \tan\beta_2, \quad \text{(A.17)}$$

$$\frac{h_1}{a} = \sin\alpha_1, \quad \text{(A.13)} \qquad \frac{h_2}{a} = \sin\alpha_2, \quad \text{(A.18)}$$

$$\frac{d_1}{f} = \tan\phi = \tan(\pi - \delta), \quad \text{(A.14)} \qquad \frac{d_2}{f} = \tan\psi = \tan(\pi - \gamma). \quad \text{(A.19)}$$

Aus den Gleichungen A.11 und A.12 ergibt sich:

$$l = p_1 + q_1 = \frac{h_1}{\tan \alpha_1} + \frac{h_1}{\tan \beta_1} \tag{A.20}$$

und mit Gleichung A.13:

$$l = a \sin \alpha_1 \left(\frac{1}{\tan \alpha_1} + \frac{1}{\tan \beta_1} \right). \tag{A.21}$$

Setzt man Gleichung A.10 ein, erhält man schließlich:

$$l = a \sin \alpha_1 \left(\frac{1}{\tan \alpha_1} + \frac{1}{\tan (\pi - \delta - \alpha_1)} \right). \tag{A.22}$$

Gleichung A.14 nach δ_1 aufgelöst und in A.22 eingesetzt ergibt:

$$l = a \sin \alpha_1 \left(\frac{1}{\tan \alpha_1} + \frac{1}{\tan \left(\arctan \left(\frac{d_1}{f} \right) - \alpha_1 \right)} \right). \tag{A.23}$$

Nach d_1 aufgelöst ergibt sich für die Teildisparität somit:

$$d_1 = f \tan \left(\arctan \left(\frac{1}{\frac{l}{a \sin \alpha_1} - \frac{1}{\tan \alpha_1}} \right) + \alpha_1 \right). \tag{A.24}$$

Für die Disparität der „rechten“Kamera ergibt sich analog:

$$d_2 = f \tan \left(\arctan \left(\frac{1}{\frac{l}{a \sin \alpha_2} - \frac{1}{\tan \alpha_2}} \right) + \alpha_2 \right). \tag{A.25}$$

Verwendet man die Gerade zwischen Benutzer (bzw. Schwenkzentrum) und einem Objekt, für das $\alpha_1 = \alpha_2 = \frac{\alpha}{2}$ gilt als Zentralstrahl des Systems, kann der Zentralwinkel ω als Winkel zwischen Zentralstrahl und der Geraden zwischen einem Objekt und dem Schwenkzentrum eingeführt werden:

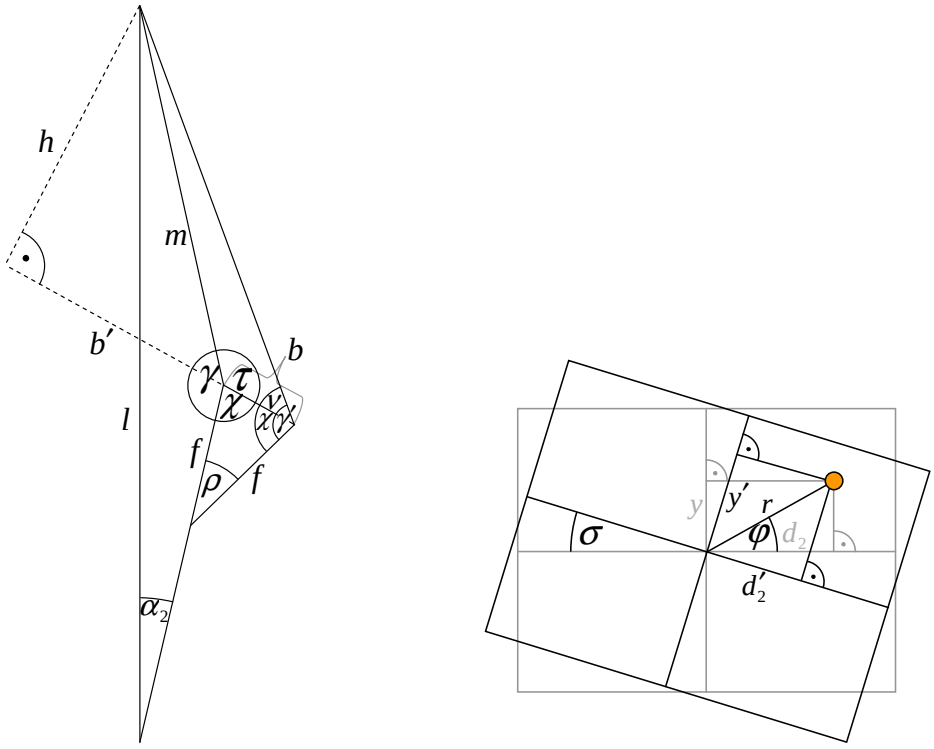
$$\alpha_1 = \frac{\alpha}{2} + \omega \tag{A.26}$$

$$\alpha_2 = \frac{\alpha}{2} - \omega \tag{A.27}$$

Für die Gesamtdisparität $d(l, \omega, \alpha)$ eines Meßpunktes mit Abstand l und Zentralwinkel ω bei einem Schwenkwinkel α ergibt sich somit:

$$\begin{aligned} d = d_1 + d_2 = \\ f \tan \left(\arctan \left(\frac{1}{\frac{l}{a \sin \frac{\alpha}{2} + \omega} - \frac{1}{\tan \frac{\alpha}{2} + \omega}} \right) + \frac{\alpha}{2} + \omega \right) \\ + f \tan \left(\arctan \left(\frac{1}{\frac{l}{a \sin \frac{\alpha}{2} - \omega} - \frac{1}{\tan \frac{\alpha}{2} - \omega}} \right) + \frac{\alpha}{2} - \omega \right). \end{aligned} \tag{A.28}$$

A.2. Herleitung der fehlerbehafteten Gesamtdisparität



(a) (b)

Abbildung A.3.: (a) Der Fehlerwinkel ρ bei Rotation um eine zur Bildebene parallele Achse durch das Bildzentrum führt zu einem fehlerhaften Winkel γ'. (b) Der Fehlerwinkel σ bei Rotation um die Optische Achse führt zu einer fehlerhaften Teildisparität d_2'.

A.2.1. Rotation um eine Bildachse

Aus Abbildung A.3 (a) ergeben sich folgende Zusammenhänge:

$$\frac{m}{\sin\alpha_2} = \frac{l}{\sin\gamma} \quad \text{(A.29)} \qquad \tau = 2\pi - \gamma - \chi \quad \text{(A.33)}$$

$$\frac{b/2}{f} = \sin\frac{\rho}{2} \quad \text{(A.30)} \qquad \frac{h}{b'} = \tan(\pi - \tau) \quad \text{(A.34)}$$

$$\chi = \frac{\pi - \rho}{2} \quad \text{(A.31)} \qquad \frac{h}{m} = \sin(\pi - \tau) \quad \text{(A.35)}$$

$$\nu = \gamma' - \chi \quad \text{(A.32)} \qquad \frac{h}{b + b'} = \tan\nu \quad \text{(A.36)}$$

Aus den Gleichungen A.31 und A.33 ergibt sich:

$$\tau = \frac{3\pi}{2} - \gamma + \frac{\rho}{2} \tag{A.37}$$

Gleichungen A.30, A.31, A.32, A.34 und A.35 in Gleichung A.36 eingesetzt ergeben:

$$\begin{aligned} \tan\left(\gamma' - \frac{\pi - \rho}{2}\right) &= \frac{m\sin(\pi - \tau)}{2f\sin\frac{\rho}{2} + \frac{m\sin(\pi-\tau)}{\tan(\pi-\tau)}} \\ &= \frac{1}{\frac{2f\sin\frac{\rho}{2}}{m\sin(\pi-\tau)} + \frac{1}{\tan(\pi-\tau)}} \end{aligned} \tag{A.38}$$

Mit A.29 und A.37 ergibt sich damit für den fehlerbehafteten Winkel $\gamma' = \gamma + \Delta\gamma$:

$$\gamma' = \arctan\left(\frac{1}{\frac{2f\sin\frac{\rho}{2}\sin\gamma}{l\sin\alpha_2\sin\left(\gamma - \frac{\rho}{2} - \frac{\pi}{2}\right)} + \frac{1}{\tan\left(\gamma - \frac{\rho}{2} - \frac{\pi}{2}\right)}}\right) + \frac{\pi - \rho}{2} \tag{A.39}$$

Dabei gilt analog zu Gleichung A.22 für den nicht fehlerbehafteten Winkel γ:

$$\gamma = \pi - \alpha_2 - \arctan\left(\frac{1}{\frac{l}{a\sin\alpha_2} - \frac{1}{\tan\alpha_2}}\right). \tag{A.40}$$

Mit Gleichung A.19 ergibt sich für die fehlerbehaftete Gesamtdisparität $d' = d + \Delta d$ schließlich:

$$\begin{aligned} d' &= d_1 + d_2' = d_1 + f\tan(\pi - \gamma') \\ &= d_1 + f\tan\left(\arctan\left(\frac{1}{\frac{2f\sin\frac{\rho}{2}\sin\gamma}{l\sin\alpha_2\sin\left(\gamma - \frac{\rho}{2} - \frac{\pi}{2}\right)} + \frac{1}{\tan\left(\gamma - \frac{\rho}{2} - \frac{\pi}{2}\right)}}\right) + \frac{\pi - \rho}{2}\right) \end{aligned} \tag{A.41}$$

A.2.2. Rotation um die optische Achse

Eine Rotation um die optische Achse und damit um den Bildmittelpunkt wird in Abbildung A.3 (b) dargestellt und es gilt:

$$\frac{d_2'}{r} = cos(\phi + \sigma) \qquad \text{(A.42)}$$

$$\frac{y}{d_2} = \tan\phi \qquad \text{(A.43)}$$

$$r^2 = y^2 + r^2 \qquad \text{(A.44)}$$

Aus den Gleichungen A.43 und A.44 in A.42 eingesetzt ergibt sich für die fehlerbehaftete Teildisparität d_2':

$$d_2' = \sqrt{y^2 + d_2^2}\cos\left(\sigma + \arctan\frac{y}{d_2}\right). \qquad \text{(A.45)}$$

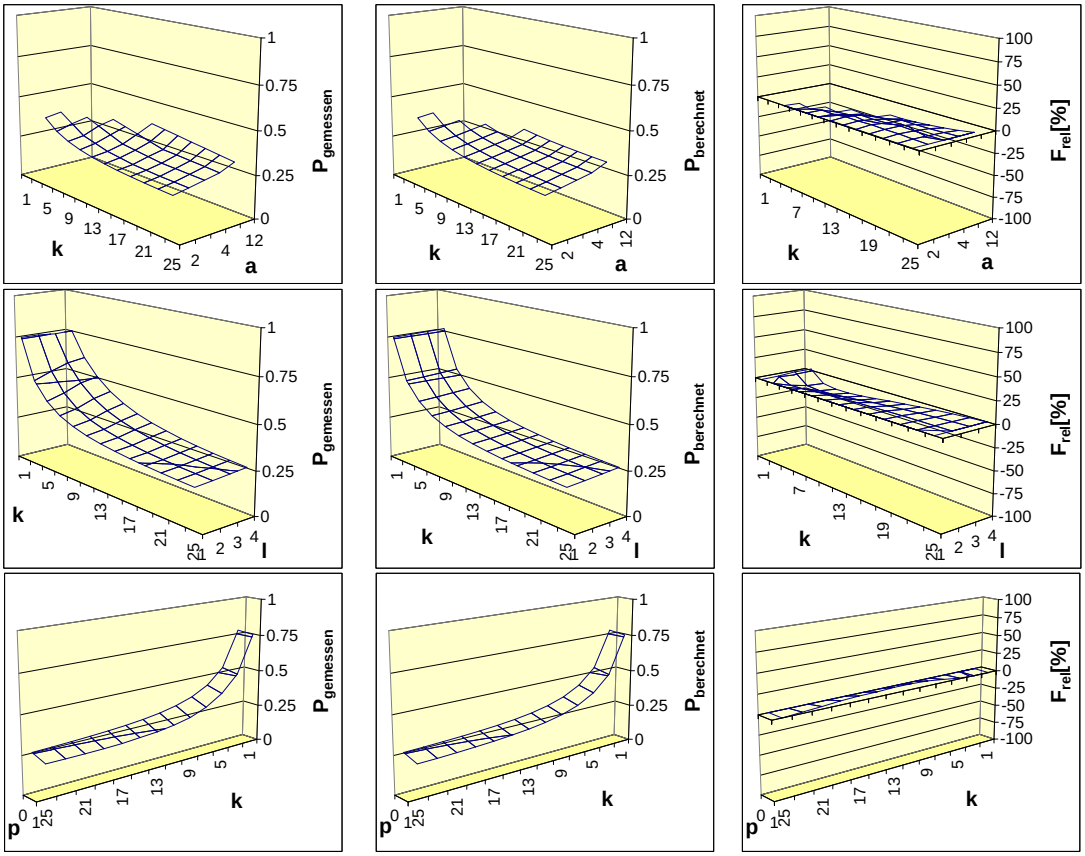
Entsprechend berechnet sich die fehlerbehaftete Gesamtdisparität $d' = d + \Delta d$ wiederum zu:

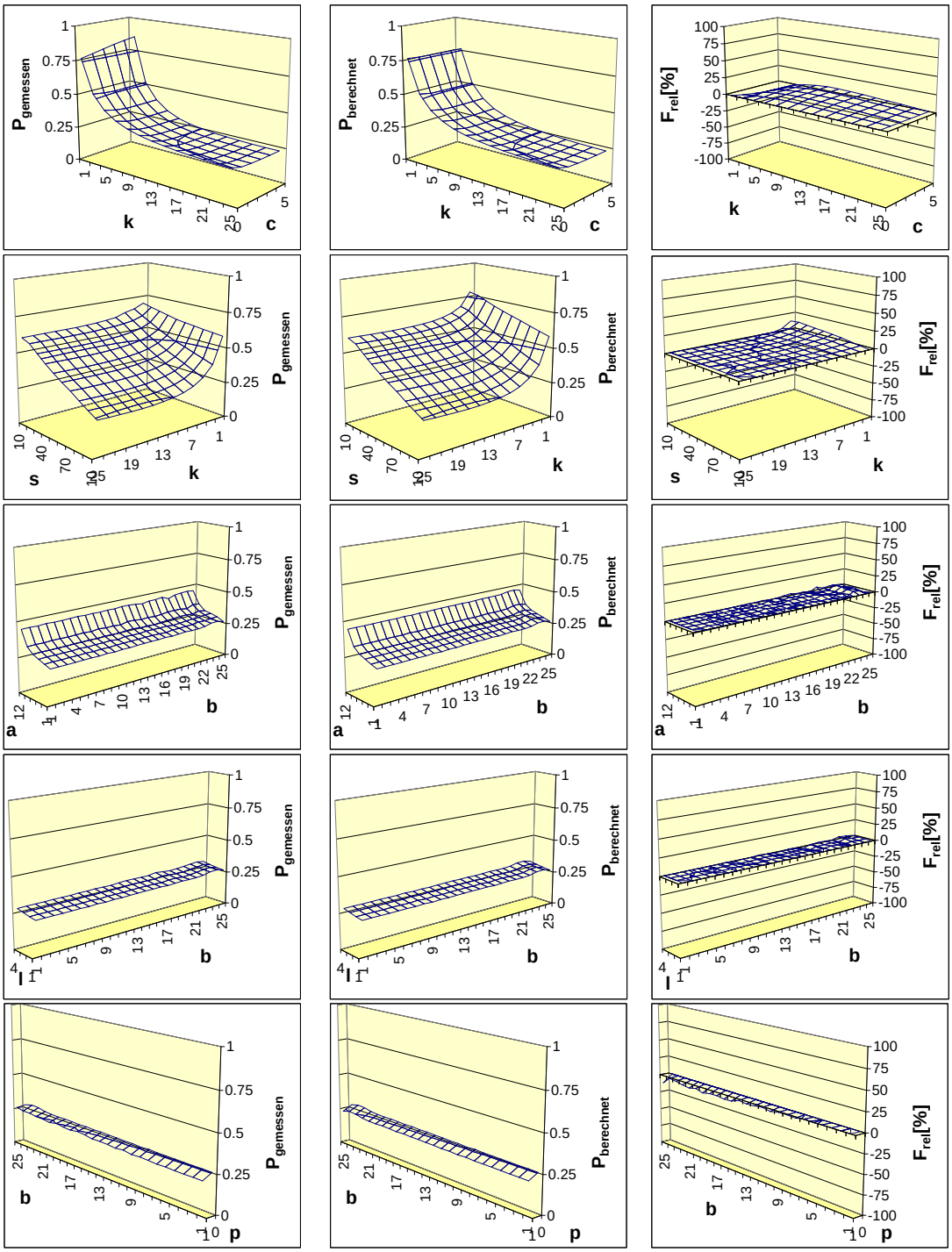
$$d' = d_1 + d_2' = d_1 + \sqrt{y^2 + d_2^2}\cos\left(\sigma + \arctan\frac{y}{d_2}\right). \qquad \text{(A.46)}$$

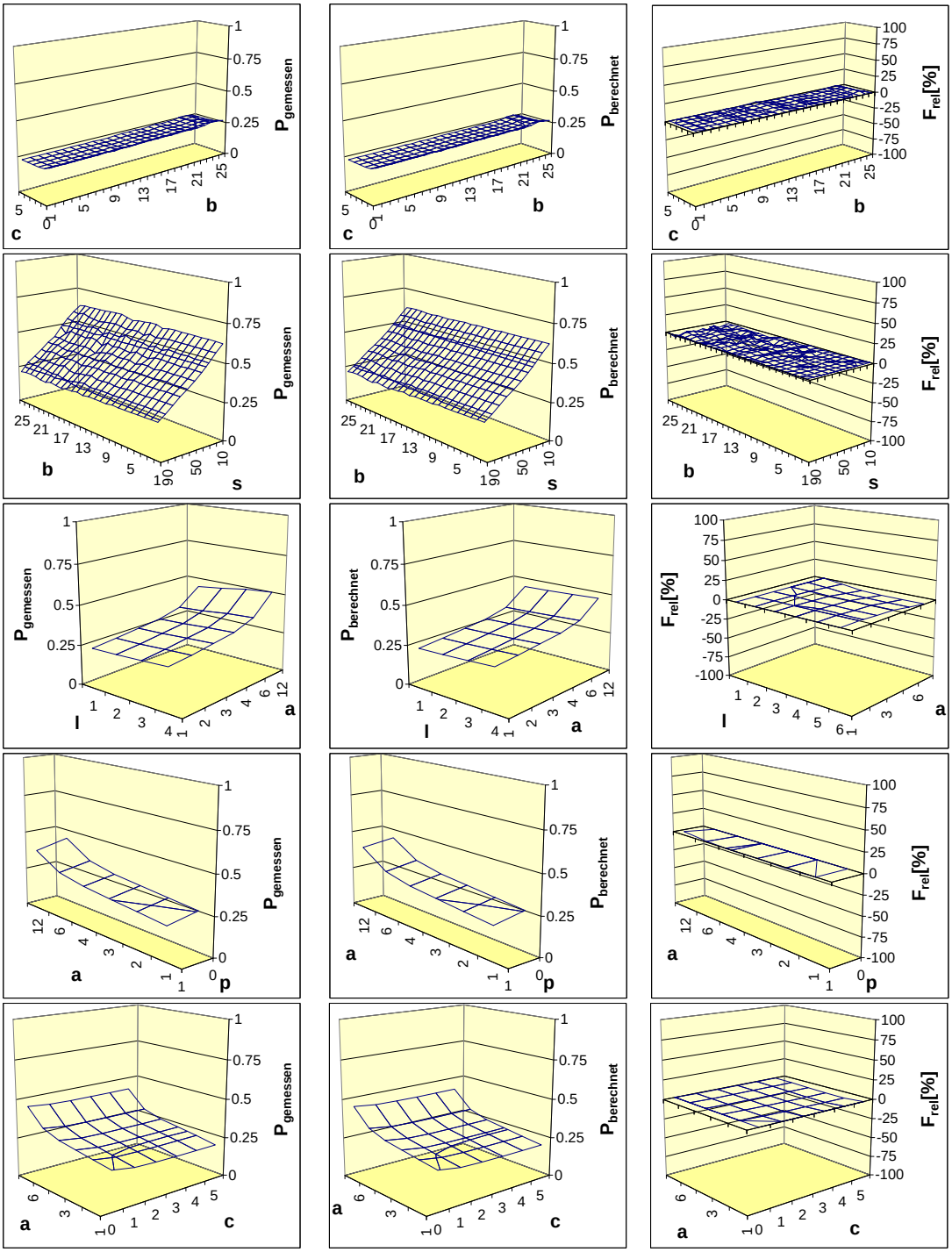
A.3. Diagramme paarweise verknüpfter Skalierungsparameter

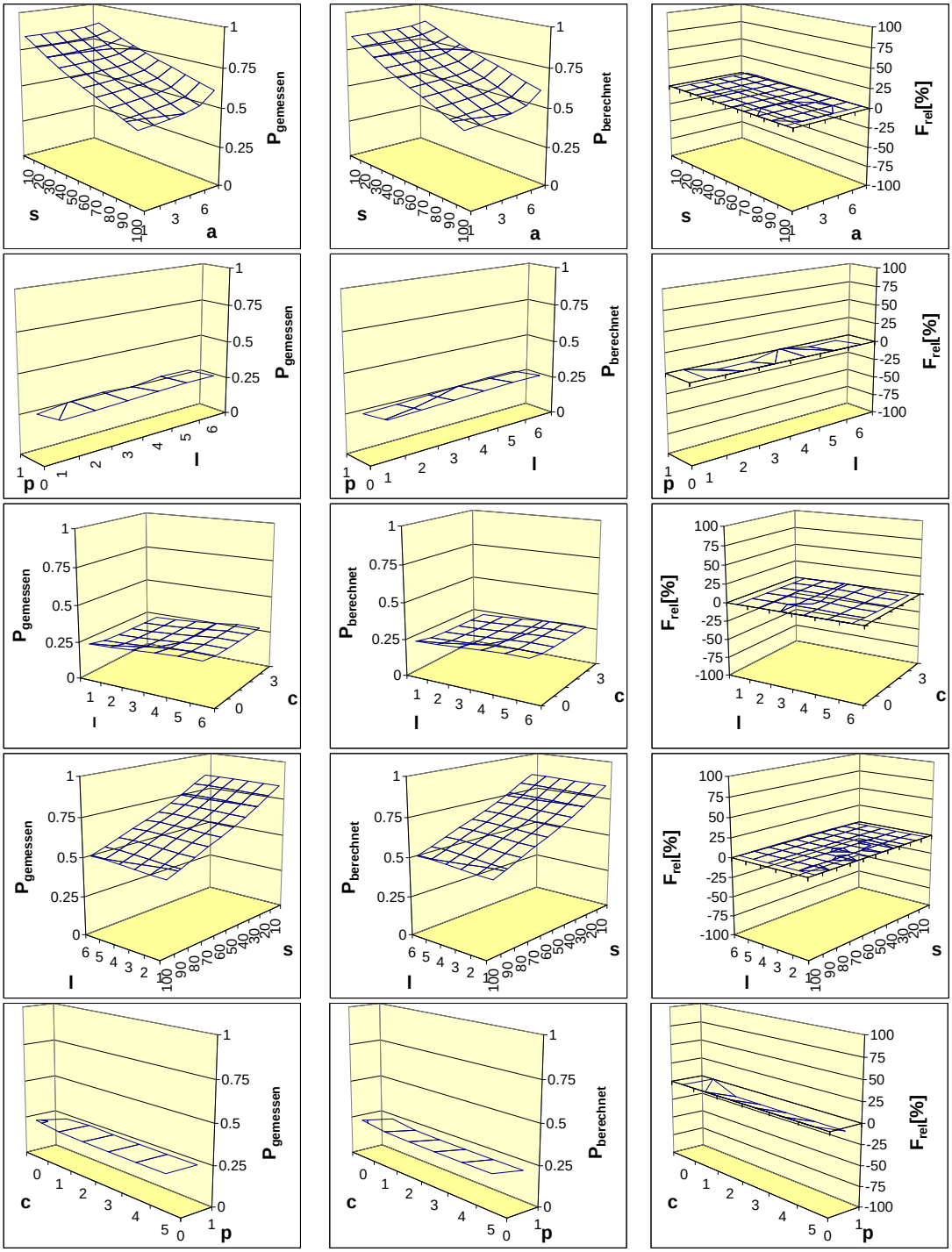
A.3.1. Fehlerwahrscheinlichkeiten

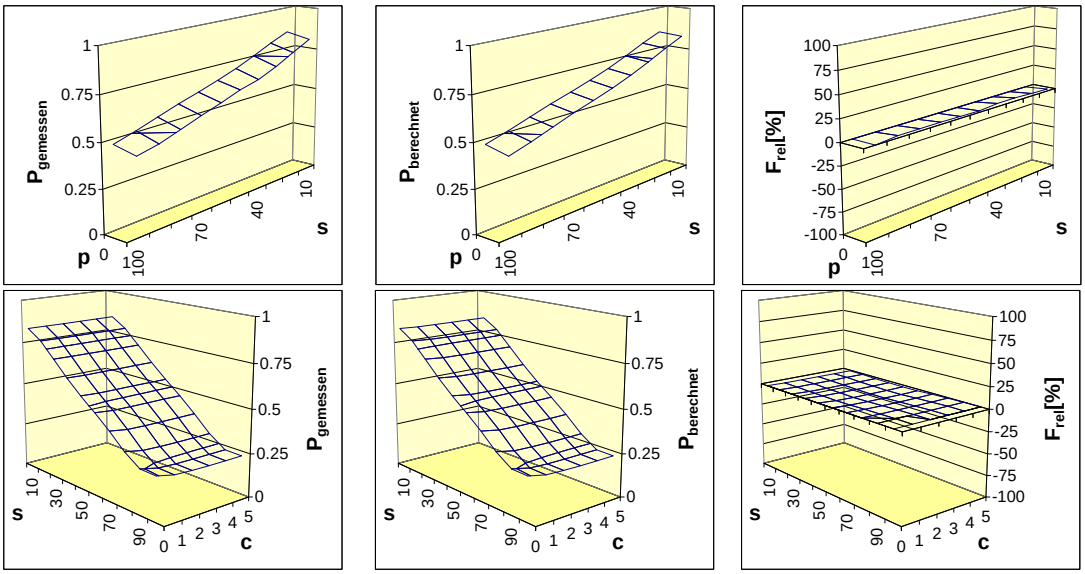
Das jeweils linke der folgenden Diagramme zeigt die direkt gemessene mittlere Fehlerhäufigkeit $P_{gemessen} = P\big(Q(g) \cup Q(x) \cup Q(y)\big)$ zweier Parameter x und y (vgl. Gleichung 3.79). Das jeweils mittlere Diagramm zeigt die über Verknüpfung der für die Einzelparameter bestimmten mittleren Fehlerhäufigkeiten berechnete Verbundfehlerhäufigkeit $P_{berechnet} = P_0 \cup P\big(Q(x)\big) \cup P\big(Q(y)\big)$ (vgl. Abbildung 3.47 und Gleichung 3.80). Das jeweils rechte Diagramm zeigt den relativer Fehler F_{rel} beider Verteilungen nach Gleichung 3.94.




P gemessen
P berechnet
F rel [%]
k
c
s
a
b
l
p

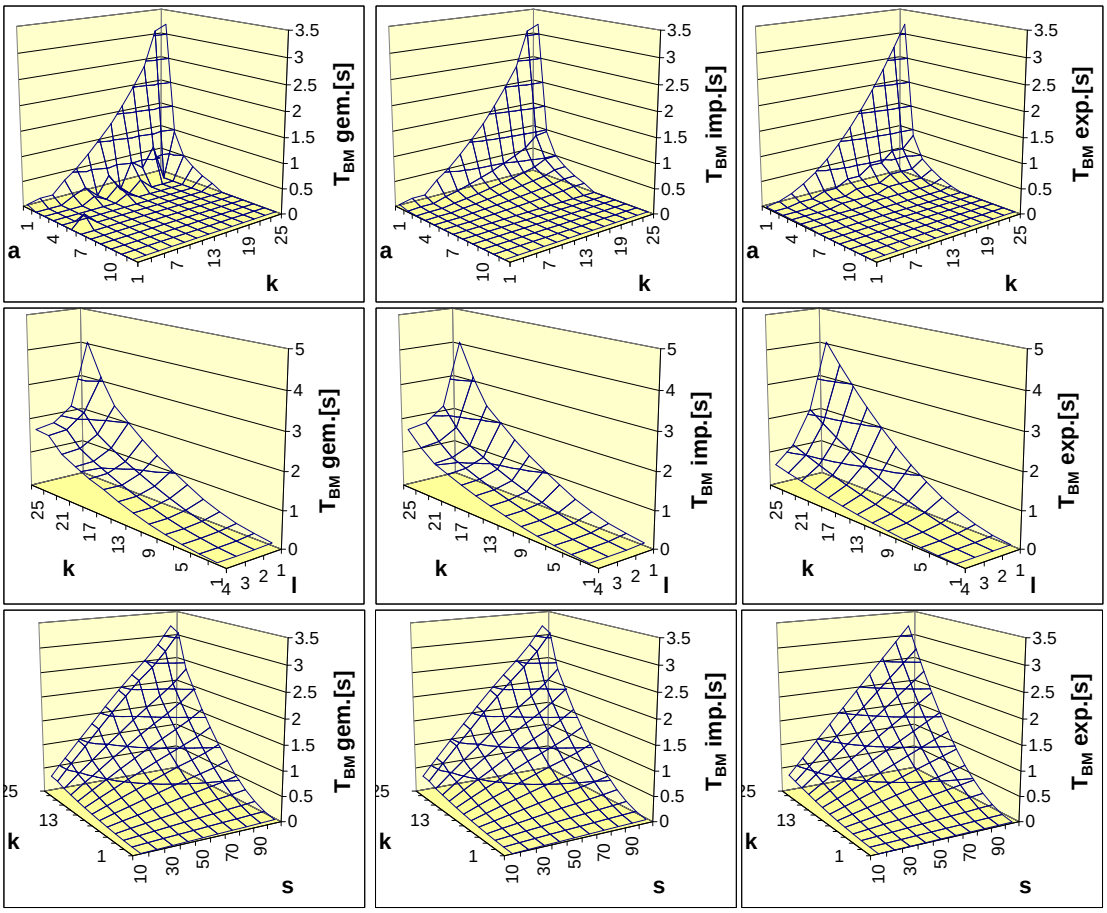

P gemessen
P berechnet
F rel[%]
b
c
s
l
a
p

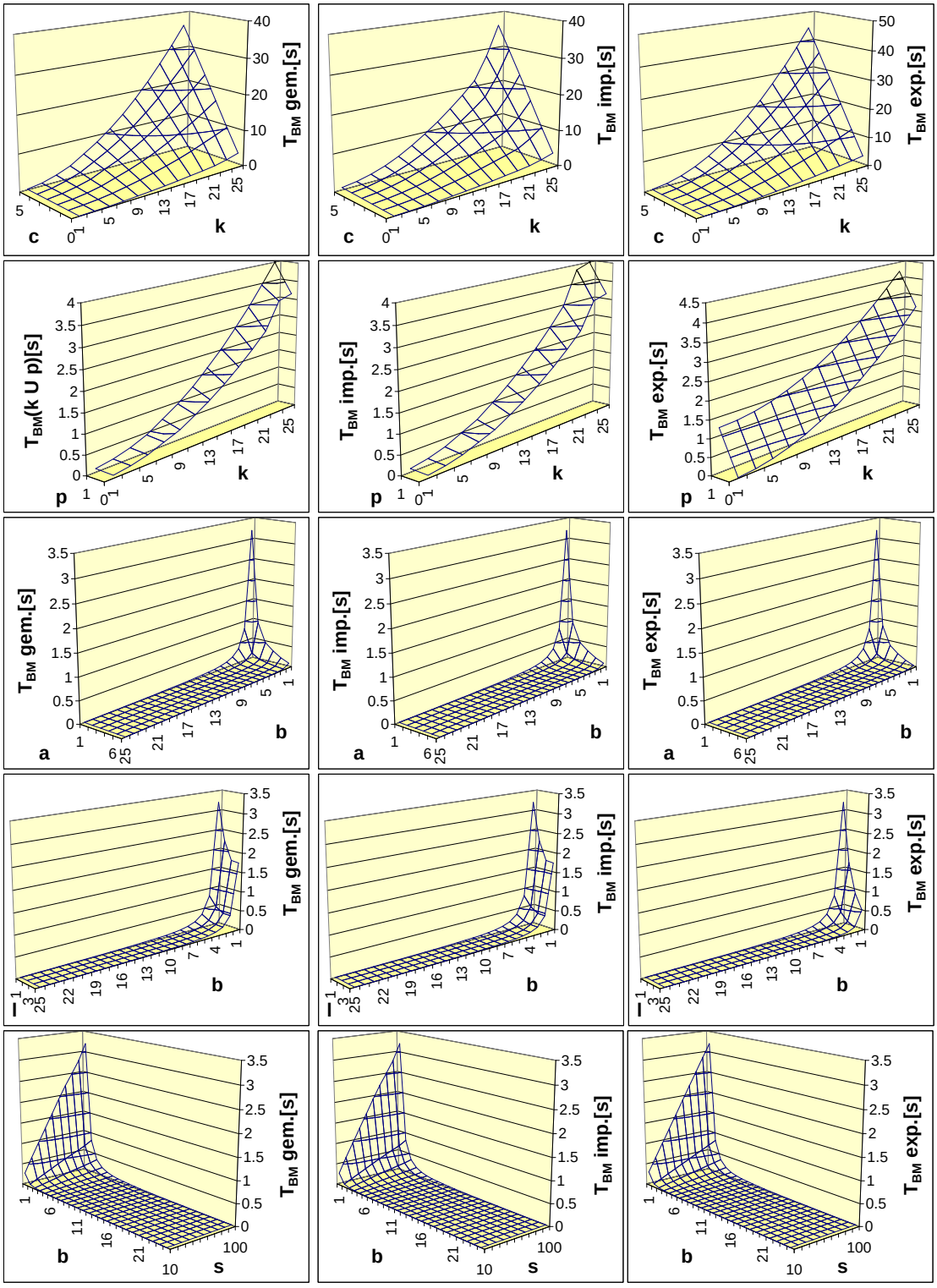

P gemessen
P berechnet
F rel [%]
s
a
l
p
c

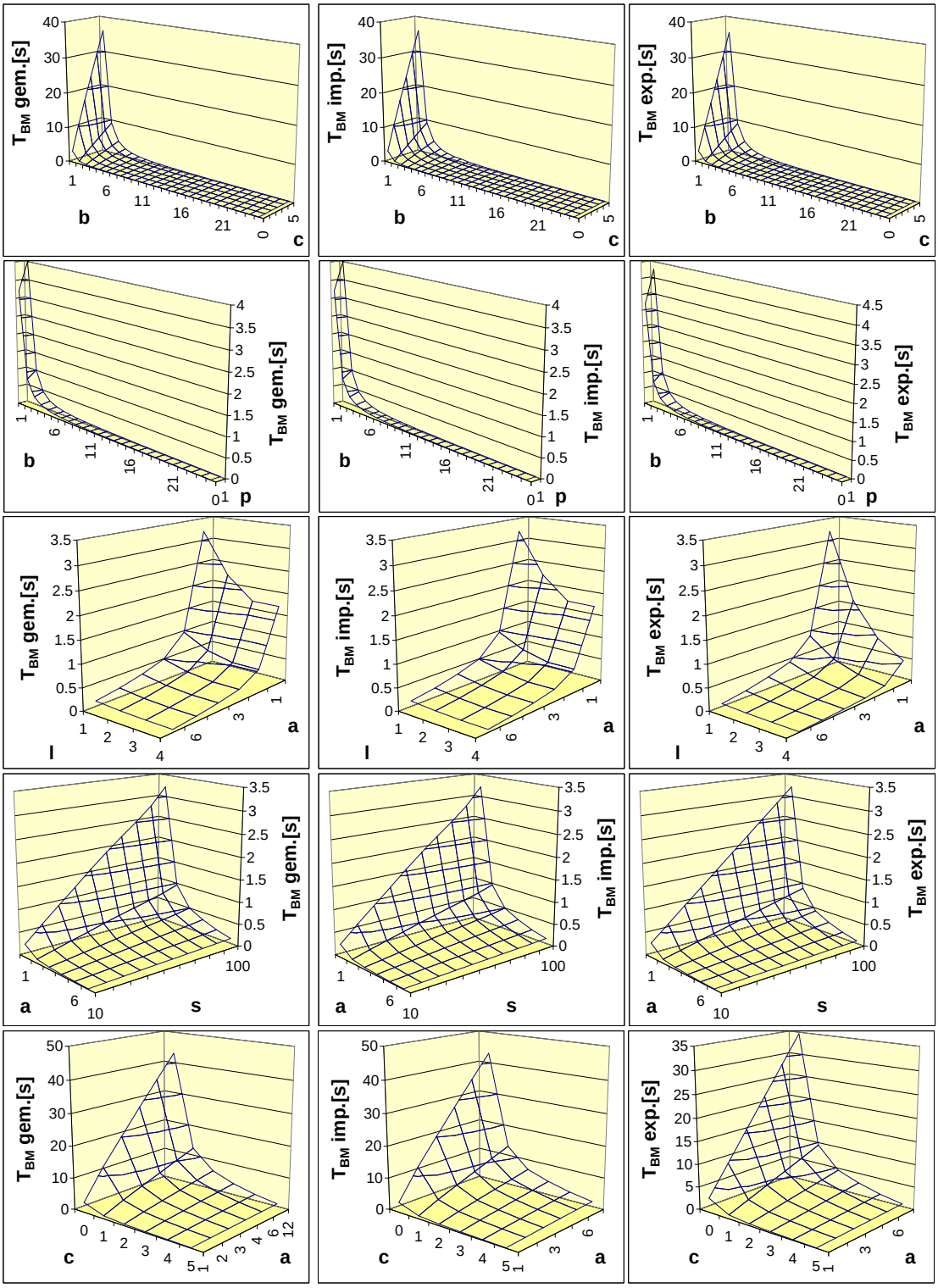

P gemessen
P berechnet
F rel [%]
p
s
c

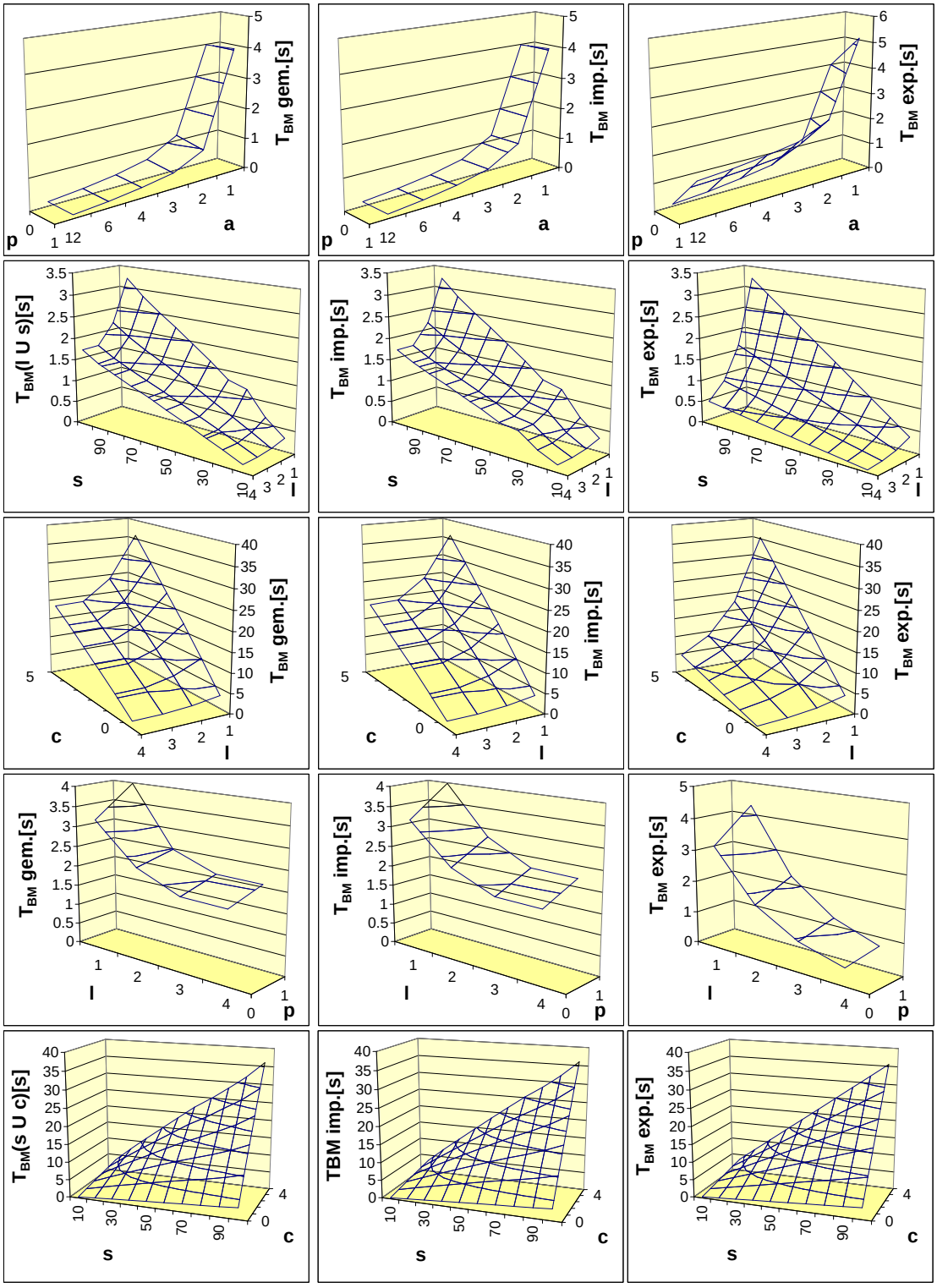
A.3.2. Rechenzeiten

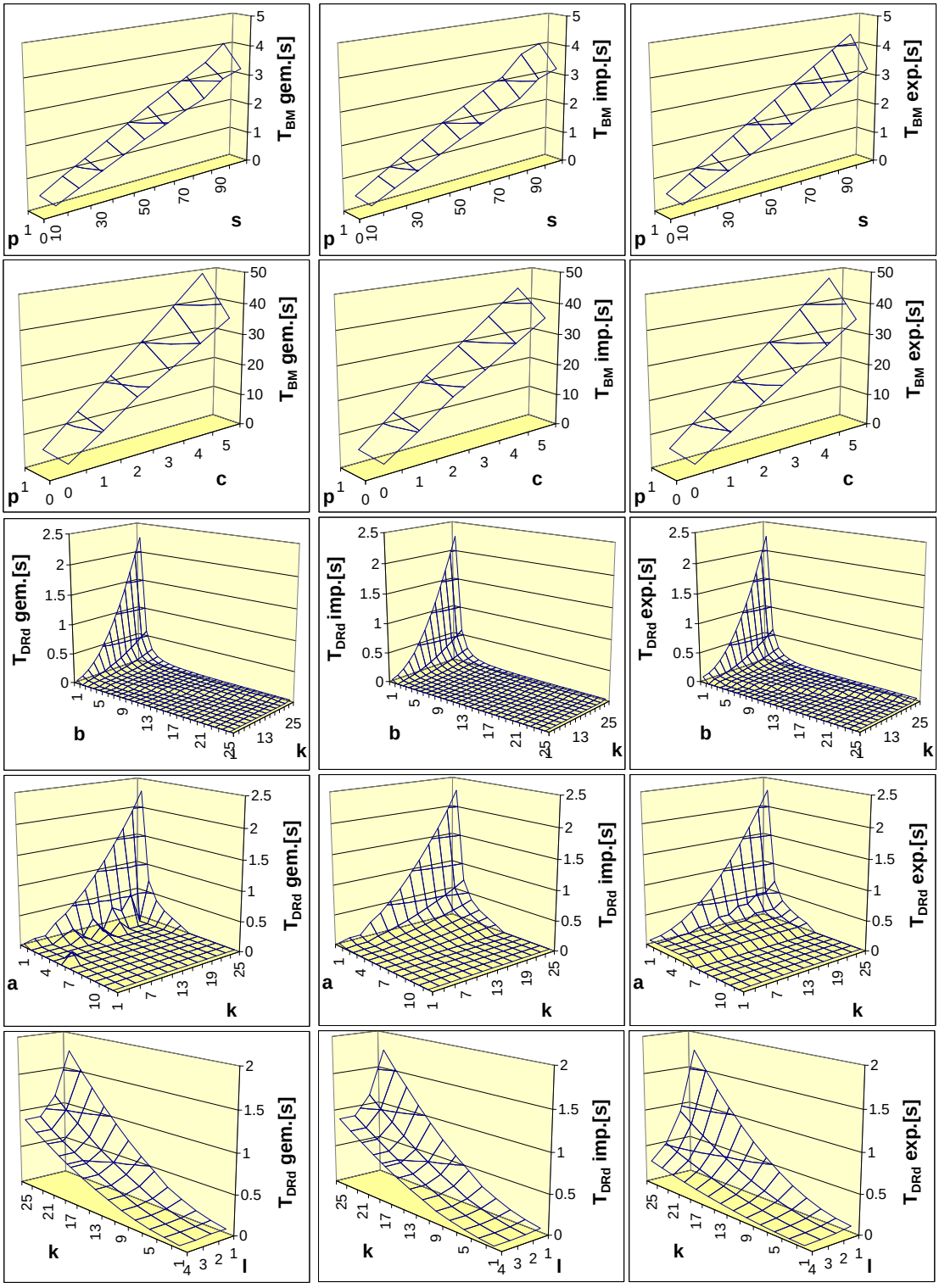
Das jeweils linke der folgenden Diagramme zeigt die Verteilung der direkt gemessenen Verbundrechenzeit $T^g(x, y)$ zweier Parameter x und y. Das jeweils mittlere Diagramm zeigt die aus der Verknüpfung $T_0^g T^g(x) T^g(y)$ der für die Einzelparameter gemessenen Rechenzeiten (vgl. Abbildung 3.50) berechnete Verbundrechenzeiten. Das jeweils rechte Diagramm zeigt die über formelmäßige Modellierung explizit berechneten Rechenzeiten.

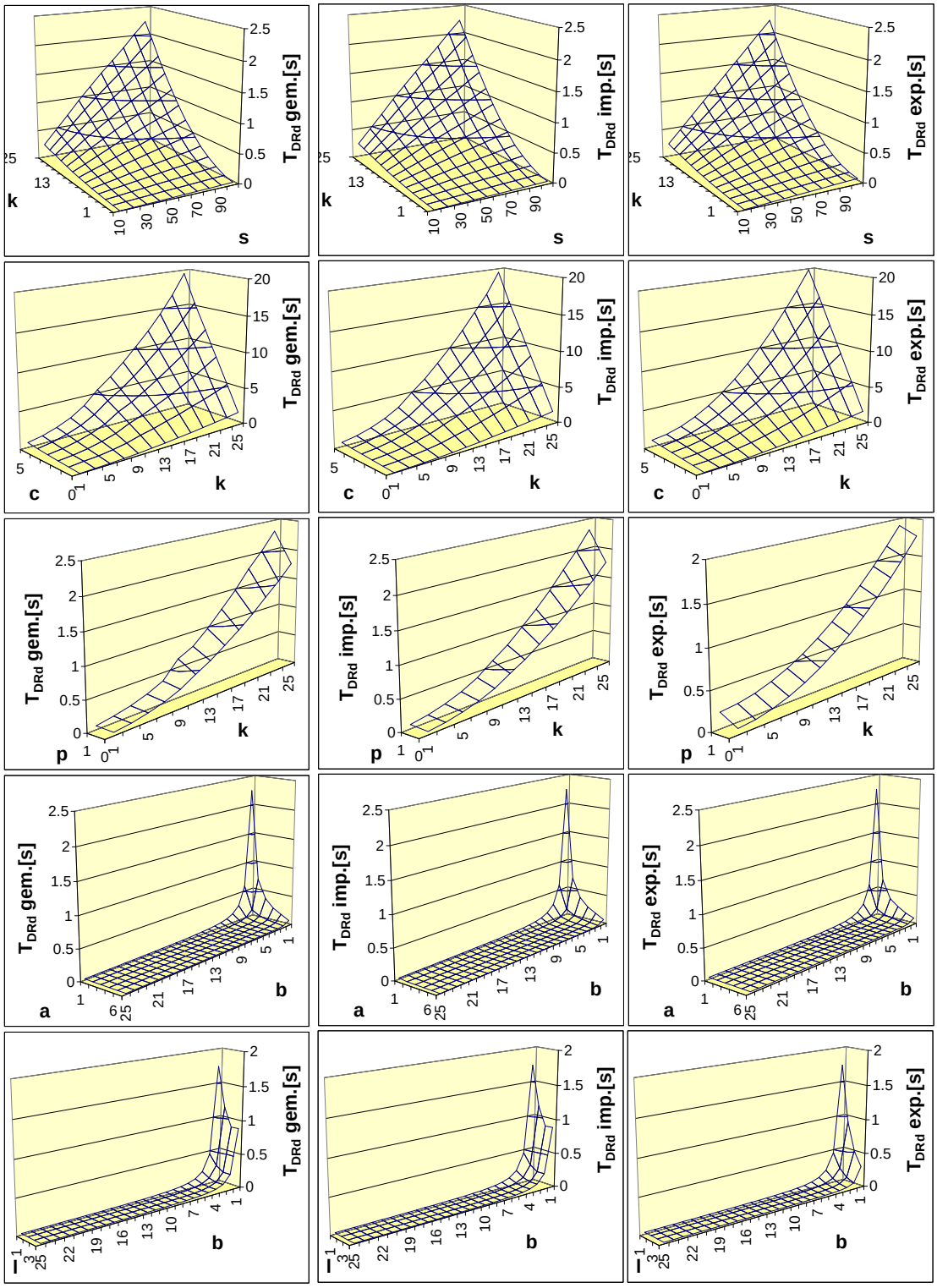


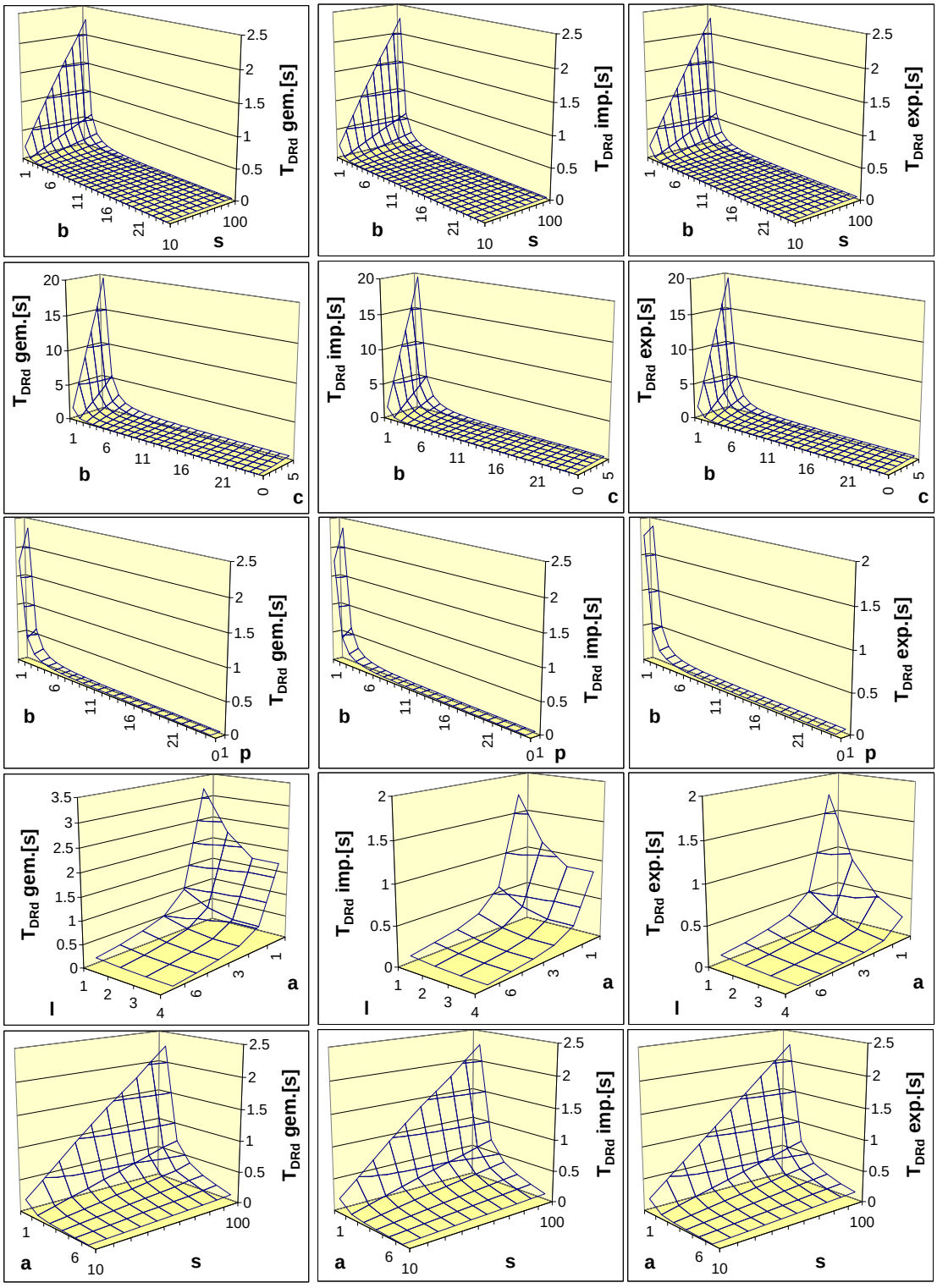

T_BM gem.[s]
T_BM imp.[s]
T_BM exp.[s]
T_BM(k U p)[s]
c
k
p
a
b
s

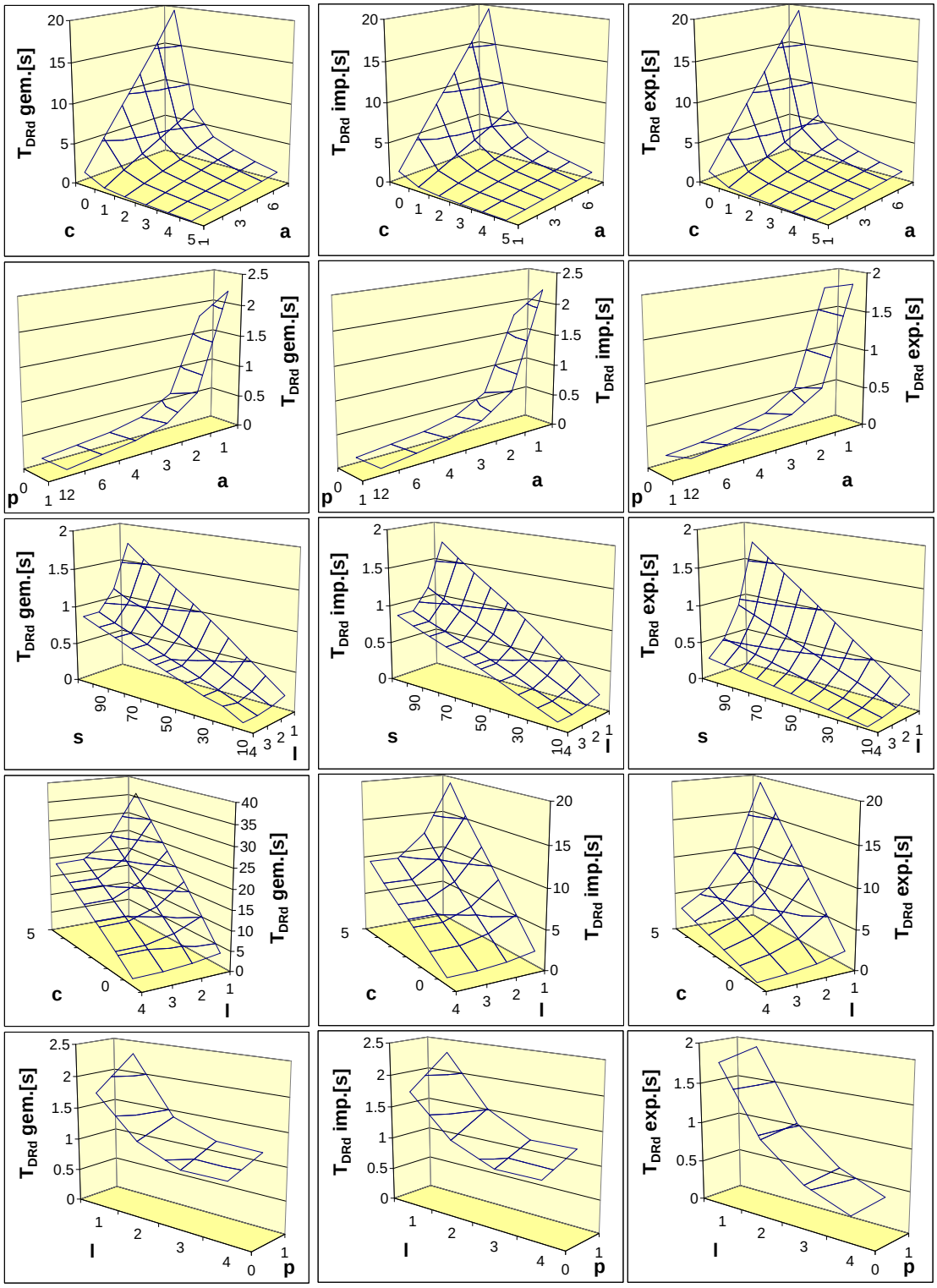

T_BM gem.[s]
T_BM imp.[s]
T_BM exp.[s]
b
c
p
l
a
s

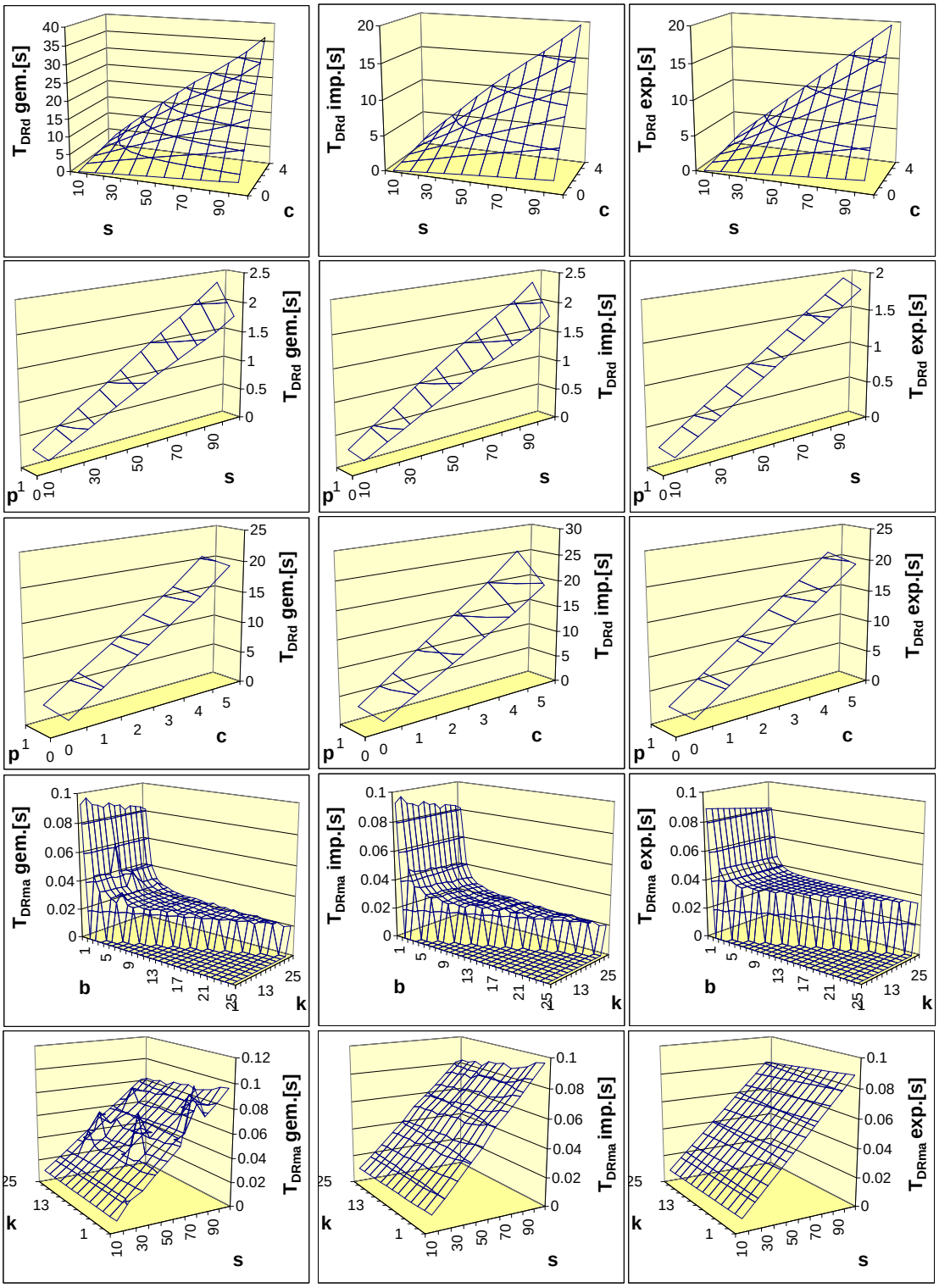

p
a
T_{BM} gem.[s]
T_{BM} imp.[s]
T_{BM} exp.[s]
T_{BM}(l U s)[s]
s
l
c
p
TBM imp.[s]
T_{BM}(s U c)[s]

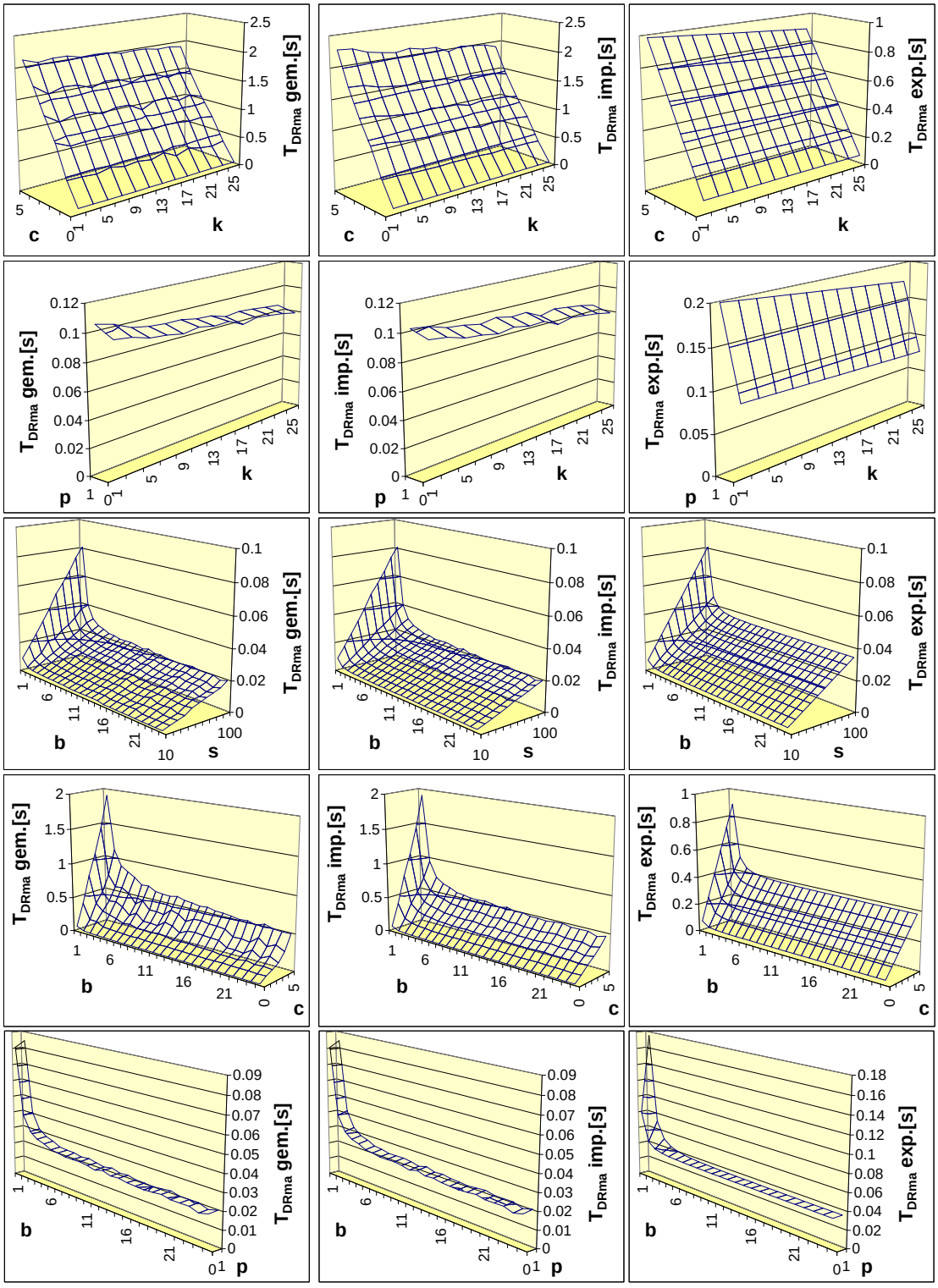

T_BM gem.[s]
T_BM imp.[s]
T_BM exp.[s]
T_DRd gem.[s]
T_DRd imp.[s]
T_DRd exp.[s]
p
s
c
b
a
k
l

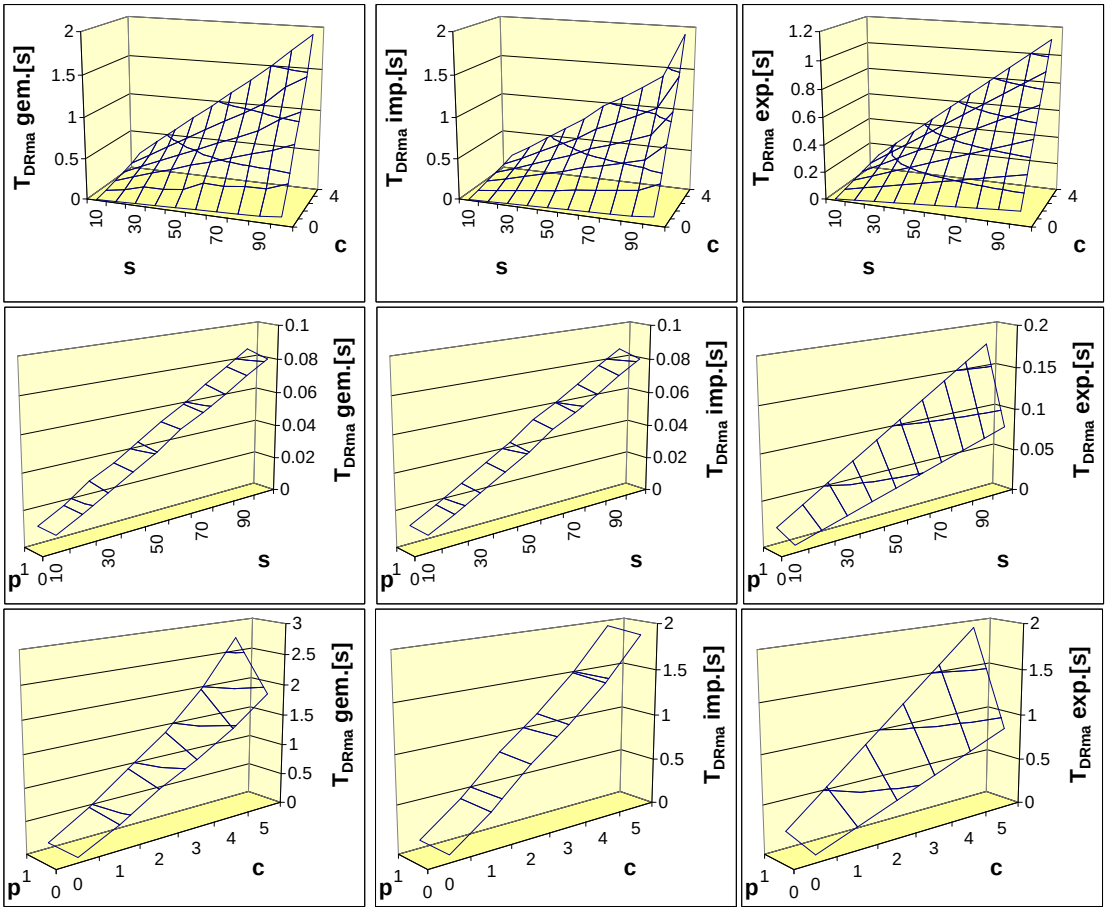

T_DRd gem.[s]
T_DRd imp.[s]
T_DRd exp.[s]
k
s
c
p
a
b


T_DRd gem.[s]
T_DRd imp.[s]
T_DRd exp.[s]
b
s
c
p
l
a


T_DRd gem.[s]
T_DRd imp.[s]
T_DRd exp.[s]
c
a
p
s
l


T_DRd gem.[s]
T_DRd imp.[s]
T_DRd exp.[s]
T_DRma gem.[s]
T_DRma imp.[s]
T_DRma exp.[s]
s
c
p
b
k


T_DRma gem.[s]
T_DRma imp.[s]
T_DRma exp.[s]
c
k
p
b
s


T_DRma gem.[s]
T_DRma imp.[s]
T_DRma exp.[s]
s
c
p

Literaturverzeichnis

[1] Hongxu Cai, Zhong Shao, and Alexander Vaynberg. Certified self-modifying code. In *Proceedings of the Conference on Programming Language Design and Implementation (PLDI)*, pages 66–77, 2007.

[2] Infotrends. Mobile imaging study. Studie, 2004.

[3] Research in China. Global market scale of multimedia application processors. Studie, 2005.

[4] Bernd Jähne and Horst Haußecker, editors. *Computer vision and applications: a guide for students and practitioners.* Academic Press, Inc., Orlando, FL, USA, 2000.

[5] Ruo Zhang, Ping-Sing Tsai, J. Cryer, and M. Shah. Shape from shading: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 21(8):690–706, August 1999.

[6] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–466, 1995.

[7] Yi Ma, Stefano Soatto, Jana Kosecka, and Shankar S. Sastry. *An Invitation to 3-D Vision.* Springer, November 2003.

[8] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach.* Prentice Hall, 2002.

[9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, second edition, 2004.

[10] Myron Z. Brown, Darius Burschka, and Gregory D. Hager. Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(8):993–1008, 2003.

[11] U. Dhond and J. Aggarwal. Structure from stereo - a review. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1489–1510, Nov/Dec 1989.

[12] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision (IJCV)*, 47(1-3):7–42, April 2002.

[13] Middlebury stereo vision page. http://vision.middlebury.edu/stereo/.

[14] Ch. Hufnagl and A. Uhl. Fast block-matching algorithms for high-resolution video compression. In *Proceedings of the International Picture Coding Symposium (PCS)*, pages 295–298, 1999.

[15] Eckehard Steinbach, Thomas Wiegand, and Bernd Girod. Using multiple global motion models for improved block-based video coding. In *Proceedings of the International Conference on Image Processing (ICIP)*, volume 2, pages 56–60, 1999.

[16] M. K. Steliaros, R. A. Packwood, and G. R. Martin. Adaptive block matching motion compensation for low bit-rate video coding. In *Proceedings of the Conference on Advanced Digital Video Compression Engineering (ADVICE)*, pages 81–88, 1996.

[17] Tuukka Toivonen and Janne Heikkilä. Fast full search block motion estimation for h.264/avc with multilevel successive elimination algorithm. In *Proceedings of the International Conference on Image Processing (ICIP)*, volume 3, pages 1485– 1488, 2004.

[18] Yong-Sheng Chen; Yi-Ping Hung; Chiou-Shann Fuh. A fast block matching algorithm based on the winner-update strategy. *IEEE Transactions on Image Processing*, 10(8):1212 – 1222, 2001.

[19] P. Nillius and J.-O. Eklundh. Fast block matching with normalized cross-correlation. Technical Report ISRN KTH/NA/P–02/11–SE, KTH, 2002.

[20] Jean-Philippe Tarel and Jean-Marc Vézien. A generic approach for planar patches stereo reconstruction. In *Proceedings of the Scandinavian Conference on Image Analysis*, volume 2, pages 1061–1070, 1995.

[21] A. Koschan. Dense stereo correspondence using polychromatic block matching. In *Proceedings of the International Conference on Computer Analysis*

of Images and Patterns (ICAIP), pages 538–542, 1993.

[22] A. Koschan. Chromatic block matching for dense stereo correspondence. In *Proceedings of the International Conference on Image Analysis and Processing (ICIAP)*, pages 641–648, 1993.

[23] K. Konolige. Small vision system: Hardware and implementation. In *Proceedings of the International Symposium on Robotics Research*, pages 111–116, 1997.

[24] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *Proceedings of the European Conference on Computer Vision*, pages 150–158, 1994.

[25] H. Hirschmüller, P. Innocent, and J. Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *International Journal of Computer Vision (IJCV)*, 47(1-3):229–246, 2002.

[26] Karsten Mühlmann, Dennis Maier, Jürgen Hesser, and Reinhard Männer. Calculating dense disparity maps from color stereo images, an efficient implementation. *International Journal of Computer Vision (IJCV)*, 47:79–88, 2002.

[27] Takeo Kanade, Atsushi Yoshida, Kazuo Oda, Hiroshi Kano, and Masaya Tanaka. A stereo machine for video-rate dense depth mapping and its new applications. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 196–202, 1996.

[28] Sriram Sethuraman, M. W. Siegel, and Angel G. Jordan. A multiresolution framework for stereoscopic image sequence compression. In *Proceedings of the International Conference on Image Processing (ICIP)*, pages 361–365, 1994.

[29] M. Accame, F. De Natale, and D. Giusto. Hierarchical block matching for disparity estimation in stereo sequences. In *Proceedings of the International Conference on Image Processing (ICIP)*, volume 2, page 2374, 1995.

[30] O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy. Real time correlation based stereo: Algorithm, implementations and applications. Technical

report, Institut National de Recherche en Informatique et en Automatique INRIA, 1993.

[31] A. Fusiello, V. Roberto, and E. Trucco. Efficient stereo with multiple windowing. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 858–863, 1997.

[32] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *Proceedings of the Conference on Pattern Analysis and Machine Intelligence (PAMI)*, 16(9):920–932, September 1994.

[33] Minglun Gong, Ruigang Yang, Liang Wang, and Mingwei Gong. A performance study on different cost aggregation approaches used in real-time stereo matching. *International Journal of Computer Vision (IJCV)*, 75(2):283–296, 2007.

[34] Olga Veksler. *Efficient graph-based energy minimization methods in computer vision.* PhD thesis, Cornell University, Ithaca, NY, USA, 1999.

[35] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194:283–287, 1976.

[36] C. Zitnick and T. Kanade. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(7):675–684, Juli 2000.

[37] Daniel Scharstein and Richard Szeliski. Stereo matching with non-linear diffusion. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, page 343, Washington, DC, USA, 1996. IEEE Computer Society.

[38] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[39] Sebastien Roy and Ingemar J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 492–502, 1998.

[40] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 377–384, 1999.

[41] Jian Sun, Nanning Zheng, and Heungyeung Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(7):787–800, July 2003.

[42] Larry Matthies, Takeo Kanade, and Richard Szeliski. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision (IJCV)*, 3(3):209–238, September 1989.

[43] J. Carpenter, P. Clifford, and P. Fernhead. An improved particle filter for non-linear problems. Technical report, Department of Statistics, University of Oxford, 1997.

[44] Peng Chang. *Robust Tracking and Structure from Motion with Sampling Method*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, February 2003.

[45] H. Tao and H. Sawhney. Global matching criterion and color segmentation based stereo. In *Proceedings of the Workshop on Applications of Computer Vision (WACV)*, pages 246–253, 2000.

[46] Ye Zhang and Chandra Kambhamettu. Stereo matching with segmentation-based cooperation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 2, pages 556–571, 2002.

[47] Michael Bleyer and Margrit Gelautz. Graph-cut-based stereo matching using image segmentation with symmetrical treatment of occlusions. *Image Communication*, 22(2):127–143, 2007.

[48] G. Egnal, M. Mintz, and R. Wildes. A stereo confidence metric using single view imagery. In *Proceedings of Vision Interface (VI)*, pages 162–170, 2002.

[49] A. Luo and H. Burkhardt. An intensity-based cooperative bidirectional stereo matching with simultaneous detection of discontinuities and occlusions. *International Journal of Computer Vision (IJCV)*, 15(3):171–188, July 1995.

[50] Y. Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision (IJCV)*, 3(1):73–102, 1989.

[51] P. Willinski and K. Overveld. Depth from motion using confidence based block matching. In *Proceedings of the Image and Multidimensional Digital Signal Processing Workshop (IMDSP)*, pages 159–162, 1998.

[52] D. Scharstein. Stereo vision for view synthesis. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 852–858, June 1996.

[53] Y. Leclerc, Q. Luong, and P. Fua. Self-consistency: a novel approachtocharacterizing the accuracy and reliability of point correspondence algorithms. In *Proceedings of the Image Understanding Workshop*, pages 793–807, 1998.

[54] Daniel Scharstein. A gradient-based evidence measure for image matching. Technical report, Cornell University, 1994.

[55] N. Labit. Motion and illumination variation estimation using a hierarchy of models: Application to image sequence coding. Technical report, INRIA, July 1993.

[56] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 20(10):1025–1039, 1998.

[57] Joaquin Lopez, Jae Hoon Kim, Antonio Ortega, and George Chen. Block-based illumination compensation and search techniques for multiview video coding. In *Proceedings of the Picture Coding Symposium*, 2004.

[58] K. Kamikura, H. Watanabe, H. Jozawa, H. Kotera, and S. Ichinose. Global brightness-variation compensation for video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(8):988–1000, December 1998.

[59] M. Gilge. Motion estimation by scene adaptive block matching (sabm) and illumination correction. In *Proceedings of the Conference on Image Processing Algorithms and Techniques*, pages 355–366, 1990.

[60] Colin Estermann, Walter Stechele, Robert Kutka, and Andreas Hutter. Luminance correction in stereo correspondence based structure from motion. In *Proceedings of the International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, pages 179–182, 2008.

[61] F. Jurie and M. Dhome. A simple and efficient template matching algorithm. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 544–549, 2001.

[62] P. Heckbert. Fundamentals of texture mapping and image warping. Master's

thesis, University of California, Berkeley, 1989.

[63] C. Gräßl, T. Zinßer, and H. Niemann. Illumination insensitive template matching with hyperplanes. In B. Michaelis and G. Krell, editors, *Proceedings of the DAGM Symposium on Pattern Recognition*, volume 2781, pages 273–280, Berlin, Heidelberg, New York, 2003. Springer-Verlag.

[64] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.