Seminararbeit
zum Thema

„Stereo Without Epipolar Lines”
Eine „maximum flow“ Formulierung des “stereo correspondence” Problems mit zwei oder mehr Kameras

Sascha Finke, Februar 2003
Seminar „3D-Rekonstruktion“, WS 2002/03
Lehrstuhl für Praktische Informatik IV, Universität Mannheim
Seminarleiter: Dirk Farin

Inhalt

0. Übersicht

Das vorliegende Paper beschreibt einen neuen Algorithmus zur Lösung des „stereo correspondence“ Problem, indem es in ein „maximum flow“ Problem umgewandelt wird. Durch diese Umwandlung werden Epipolarlinien nicht mehr benötigt und es erlaubt zudem die Nutzung von mehr als zwei willkürlich positionierten Kameras. Der global effiziente „maximum flow“ Algorithmus erzeugt in einem Schritt einen so genannten „minimum cut“, der die Tiefenkarte des fotografierten Objektes darstellt. Dieser effiziente Ansatz zur Stereo-Analyse arbeitet dabei genauer und erzeugt eine bessere Tiefenkarte als die traditionellen Stereo-Algorithmen. Weiche Übergänge innerhalb der Tiefenkarte sind dabei nicht nur entlang der Epipolarlinien, sondern auch vertikal dazu vorhanden, während das globale Optimum der Tiefendarstellung garantiert ist. Das globale Optimum entspricht in diesem Zusammenhang einem Ergebnis, bei dem die vorgegebene Kostenfunktion minimiert ist. Ergebnisse zeigen, dass dieses Verfahren sowohl bessere Tiefeneinschätzungen vornimmt, als auch „Tiefen-Sprünge“ bei den Objekten besser verarbeitet. Die „worst case“ Laufzeit ist $O(n^{1,5} d^{1,5} \log(nd))$, während die „average case“ Laufzeit $O(n^{1,2} d^{1,3})$ beträgt (n: Anzahl der Pixel der einzelnen Bilder, d: Tiefenauflösung).

1. Einführung

Es ist bekannt, dass tiefenabhängige Unterschiede in Stereobildern immer entlang Linien auftreten, die anhängig von der Kamerabewegung sind, den so genannten Epipolarlinien. Genau diese Linien reduzieren das „stereo correspondence“ Problem auf nur noch eine Dimension und die so genannte „ordering“ Bedingung erlaubt die Verwendung der dynamische Programmierung (Baker, 1981; Ohta and Kanade, 1985; Cox et al., 1996; Faugeras, 1993). Dabei ist klar, dass diese Reduzierung auf nur eine Dimension eine zu starke Vereinfachung des Problems darstellt, da bei einer unabhängigen Betrachtung der einzelnen Zeilen vertikal dazu Artefakte entstehen können. Dieses fällt besonders bei Objekträndern auf, die vertikal zu den Epiolarlinien liegen.

In diesem Paper wird das Problem zweidimensional betrachtet, Epiolar Linien werden nicht mehr benötigt, und die so genannte „ordering“ Bedingung wird durch die „local coherence“ Beschränkung ersetzt. Um eine zweidimensionale Optimierung vorzunehmen, wird das „stereo correspondence“ Problem nun in ein „maximum flow“ Problem innerhalb eines Graphen umgewandelt und gezeigt, dass der „minimum cut“ als Tiefendarstellung interpretiert werden kann. Während die „worst case“ Laufzeit signifikant größer ist als beim Dynamischen Programmieren, bleibt die „average case“ Laufzeit ungefähr gleich. Wir zeigen außerdem, dass die zweidimensionale Optimierung sogar mehr als zwei Kameras, sowie willkürlich gewählte Rekonstruktionskörper zulässt.

Es hat schon frühere Ansätze zur abhängigen Berechnung von aufeinander folgenden Epipolarlinien durch dynamisches Programmieren gegeben. In Ohta and Kanade (1985) wurde zuerst dynamisches Programmieren zur Tiefenberechnung entlang der Epipolarlinien verwendet, um dann die Ergebnisse iterativ durch Verwendung von vertikalen Referenzlinien zu verbessern. In Cox et al. (1996) wurde ein probabilistischer Ansatz verwendet, um die Qualität des Tiefenbildes zu verbessern. Zum einen wurden Berechnungen durch Ergebnisse der vorhergehenden Linien beeinflusst, was aber zu einer vertikalen Asymmetrie führte. Als zweite Möglichkeit wurde vorgeschlagen, das Ergebnis einer Linie iterativ durch die Lösungen ihrer Nachbarlinien zu verbessern. Obwohl dieser lokale Ansatz nicht global optimal ist, war es ein effizienter Weg, vertikal zu den Epipolarlinien weiche Übergänge zu

erzwingen. In Bellhumeur (1996) wurde ein Bayescher Ansatz beschrieben. Das resultierende Optimierungsproblem kann durch dynamische Programmierung entlang der Epipolarlinien effizient gelöst werden, resultiert aber im gleichen Problem wie Ohta and Kanade, 1985; Cox et al., 1996), nämlich der Unabhängigkeit der einzelnen Lösungen. Er schlägt eine heuristische Methode, genannt „iterated stochastic dynamic programming“ vor, die Ergebnisse von bereits berechneten Epipolarlinien benutzt, um Ergebnisse von anderen, zufällig gewählten Linien iterativ zu verbessern. Dieser Ansatz ist nicht global optimal und zeichnet Übergänge so weich, dass „Tiefen-Sprünge“ zum Teil nicht mehr erkannt werden können.

Das Konzept des „Maximum flow“ trat zuerst bei Greig et al. (1989) im Zusammenhang mit binären Markov Zufallsfeldern auf, bei dem jeder Pixel eines digitalen Bildes eines von zwei Tiefen zugewiesen bekam. Die „maximum flow“ Formulierung für mehr als zwei Tiefen und einer konvexen Kostenfunktion wurde von Roy und Cox (1998) im Kontext von „stereo correspondence“ vorgestellt. Ishikawa und Geiger (1998) präsentierten ein ähnliches Verfahren wie Roy und Cox (1998), fügten es aber in den Kontext der Markov Zufallsfelder ein. Boykov et al. (1998) präsentierten ebenfalls eine Markov Zufallsfeld Formulierung mit einer nicht linearen Kostenfunktion.

Einige *n*-Kamera-Algorithmen wurden vorgestellt (siehe Faugeras, 1993; Cox, 1994; Kang et al., 1994; Kanade et al, 1996). In Cox (1994) wurde ein Kamerapaar als Referenz- bzw. Basispaar genutzt, während weitere Kameras Zusatzinformationen zur besseren Tiefendarstellung boten. Zur Lösung des „matching“ Problems wurde dann dynamisches Programmieren wie bei Cox et al. (1996) verwendet. Bei Kang et al. (1994) und in Kanade et al. (1996) wurde ein mehrfach-Kamera-Echtzeit-Stereosystem vorgestellt. Sie benutzten eine Kamera als Referenz, während jede der restlichen Kameras Informationen für jeweils eine Tiefe des Referenzbildes boten. Die Tiefe wurde dabei für jeden Pixel unabhängig voneinander berechnet, was natürlich weiche Übergänge zwischen Pixel unmöglich machte. Stattdessen wurde ein Tiefpassfilter verwendet und statt einzelne Pixel zu betrachten, wurde ein Sichtfenster verwendet. Obwohl dieses weiche Übergänge hervorrief, kam es zum Teil zu Verwischungen und Tiefeninformationen gingen verloren.

In Abschnitt 2 wird das so genannte „stereo framework“ mit mehreren Bildern aus willkürlich gewählten Ansichten beschrieben. Zudem wird eine sehr einfache „stereo matching“ Kostenfunktion vorgestellt, die mehr als zwei Bilder unterstützt. In Abschnitt 3 wird das Stereoproblem unter Benutzung der „local coherence“ Bedingung von der eindimensionalen Variante auf die zweidimensionale erweitert. In Abschnitt 4 wird das „stereo matching“ Problem als „maximum flow“ Problem formuliert. Details zu den Algorithmen sowie deren Laufzeiten werden in Abschnitt 4.3 vorgestellt. In Abschnitt 5 werden verschiedene Beispiele aufgezeigt und diskutiert.

2. Das „Stereo Framework“

Dieser Abschnitt beschreibt das allgemeine „stereo framework“, bestehend aus zwei Teilen. Zum einen wird ein Berech in der realen Welt ausgewählt, in dem das „stereo matching“ stattfinden soll. Jedes Objekt, dessen Oberfläche rekonstruiert werden soll, muss innerhalb dieses Raumes liegen. Als Zweites wird jeder Punkt innerhalb des „matching“ Raumes auf einen bestimmten Punkt bzw. dessen Intensitätswert auf die einzelnen Bilder projiziert. Diese Information wird im weiteren Verlauf dazu genutzt, die „matching“ Kosten zu berechnen. Der Algorithmus von Roy (1999) bildet nun ein Tiefenbild, das den Raum in zwei Hälften teilt, und nicht willkürlich innerhalb des Raumes liegt.

2.1 Der „stereo matching“ Raum

Der 3D-Raum, der jede mögliche Tiefendarstellung enthält, ist der so genannte „matching space“ und wurde schon bei Yang and Yuille (1995) und Marr und Poggio (1979) benutzt. Dieser Raum wird in diskrete Abschnitte unterteilt und vom Stereoalgorithmus nach der optimalen Tiefendarstellung durchsucht. Die Vorder- und Rückseite dieses Raumes müssen disjunkt sein. Gemäß Definition trennt die Tiefendarstellung die Vorder- und Rückseite des „matching spaces“ und kann deshalb als Funktion der Vorder- bzw. Rückseite definiert werden.

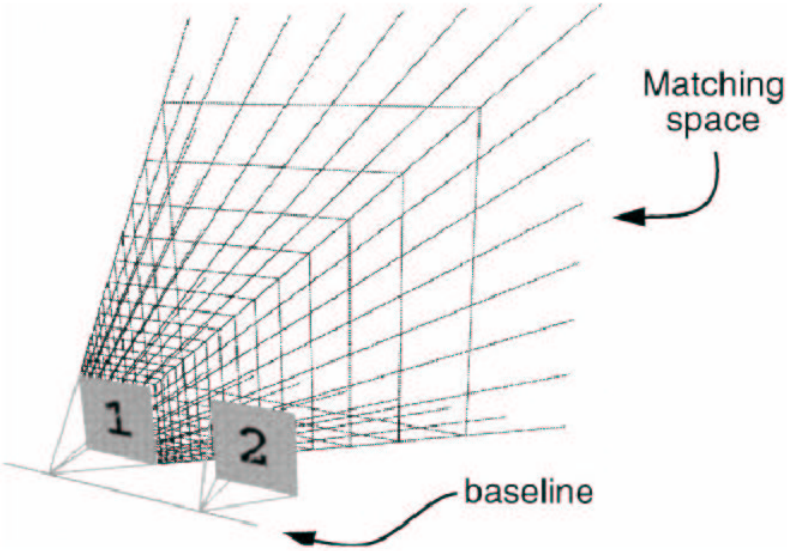

Abb. 1: Standard „stereo framework“. Der „stereo matching“ Körper ist das Blickfeld der ersten Kamera.

Beim Standard-Stereo ist der „matching space“ pyramidenförmig, passend zu dem Sichtfeld einer Kamera. Die von der Kamera aus gesehen nächste bzw. am weitesten entfernte Fläche sind die Vorder- und Rückseite der Sichtpyramide. Offensichtlich bietet jede gültige Oberfläche, die Vorder- und Rückseite trennt, exakt einen Tiefenwert für jeden Pixel des Bildes. Es gibt in diesem Zusammenhang keine weiteren Restriktionen bzgl. des „matching space“, um die Lösung des Problems durch den Stereoalgorithmus nicht zu gefährden. Es werden nur einheitliche Skalen, entweder „disparity“ oder Tiefe innerhalb eines „matching space“, zugelassen (dargestellt in Abb.2).

Der „matching space“ ist als 3D-Körper definiert (als Pyramide oder auch als Würfel) der von den Achsen a,b,d aufgespannt wird. Der Bereich in dem ein Punkt innerhalb des Würfels liegt, ist demnach

$$0 \le a' \le a'_{size}$$
$$0 \le b' \le b'_{size}$$
$$0 \le d' \le d'_{size}$$

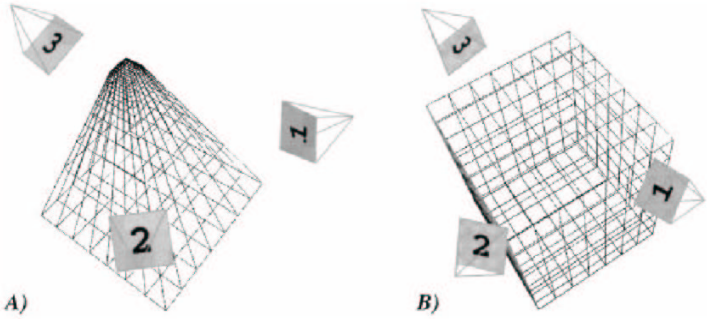

Abb.2: Drei Kameras in willkürlich gewählten Positionen und Ausrichtungen im Raum, um zwei Arten von „matiching spaces“ herum ausgerichtet (A) mit einheitlichen „disparity“ Schritten (B) mit einheitlichen Tiefenschritten

wobei *a’*, *b’* die Pixelkoordinate innerhalb eines Bildes und *d’* die Tiefe bzw. „disparity“ darstellt. a'_{size}, b'_{size} und d'_{size} ist die aufgespannte Größe des Körpers.

Ein Punkt (*a’,b’,d’*) wird in der realen Welt als ein homogener Punkt p_w dargestellt, der definiert ist als

$$p_w = Q\begin{bmatrix} a' \\ b' \\ d' \\ 1 \end{bmatrix}$$

Q ist hier eine 4x4 Matrix, die es erlaubt, die Gestalt und die Lage des „matching space“ in der realen Welt zu ändern.

Der „matching space“ ist identisch zum Sichtkegel einer Kamera, wenn Q definiert wird als

$$Q = W^{-1}\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & 1 \\ & & 1 & 0 \end{bmatrix} \times \begin{bmatrix} \frac{x_{size}}{a'_{size}-1} & & & 0 \\ & \frac{y_{size}}{b'_{size}-1} & & 0 \\ & & \frac{d_{\max}-d_{\min}}{d'_{size}-1} & d_{\min} \\ 0 & 0 & 0 & d' \end{bmatrix} \quad (1)$$

wobei x_{size} und y_{size} die Bildgröße definiert, d_{min} und d_{max} die erlaubten „disparity“ Intervalle sind und *W* die Sichttransformationsmatrix der Kamera ist. Man beachte, dass *d’* in der 4. Zeile steht und dadurch „disparity“ statt Tiefe repräsentiert. Will man eine Tiefenquantisierung, so muss die letzte Zeile von *Q* [0, 0, 0, 1] sein

$$Q = \begin{bmatrix} \frac{x_{size}}{a'_{size} - 1} & & & a_{\min} \\ & \frac{y_{size}}{b'_{size} - 1} & & b_{\min} \\ & & \frac{d_{\max} - d_{\min}}{d'_{size} - 1} & d_{\min} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Man beachte, dass der „matching space“ in diesem Fall unabhängig von der Kamerageometrie definiert ist.

2.2 Pixelintensität

In diesem Abschnitt wird ein allgemeines „framework“ dargestellt, das Stereoprobleme bzgl. mehrerer Bilder und willkürlich gewählter Kamerapositionen löst. Jede Kamera *i* bietet eine Matrix W_i die jeden Punkt in realen Weltkoordinaten in einen Punkt der Bildfläche des Kamerabildes transferiert. Diese Matrix ist von Parametern der Kamera abhängig, die man in externe und interne Parameter unterteilen kann. Interne Parameter sind z.B. die Brennweite oder Linsenart, während externe Parameter z.B. die Position oder Ausrichtung der Kamera innerhalb der realen Welt sind. Für die Stereobetrachtung ist es nicht nötig, alle Parameter zu kennen („strong calibration“ nicht notwendig). Matrizen, die die Beziehungen zwischen den einzelnen Kameras darstellen, können auch durch das Finden von einigen korrespondierenden Punktpaaren berechnet werden, ohne das eine volle Kalibrierung benötigt wird. Im Kontrast zu der „strong calibration“ haben die berechneten Tiefenwerte bei diesen schwach kalibrierten Kameras keine feste Beziehung zu den reellen Tiefen in der Szene.

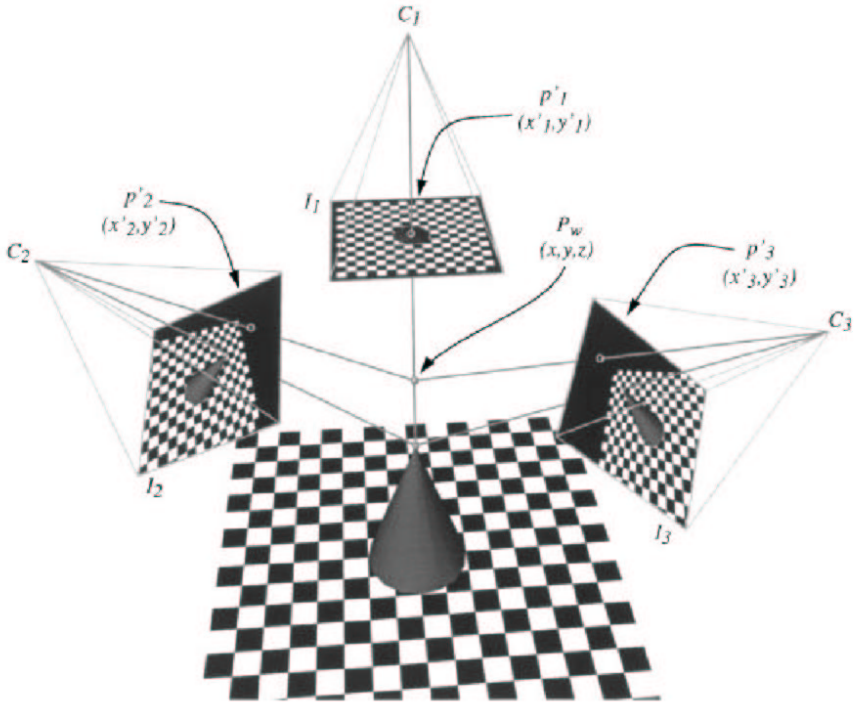

Abb. 3: Mehrfachkamera-Aufbau. Jeder Punkt in der reellen Welt kann auf jeweils einen Punkt in den Kamerabildern zurückprojiziert werden

Eine Menge von n Kameras $C_1,\ldots,C_n$ macht n Bilder $I_1,\ldots,I_n$ von einer Szene (siehe Abb.3 mit n=3). Ein Würfel stellt den „matching“ Körper dar (in Abb.3 nicht zu sehen), in dem die Tiefendarstellung berechnet wird. Innerhalb des Körpers wird ein Punkt (a',b',d') in den homogenen Punkt p_i der Kamera i durch folgende Relation überführt.

$$\begin{aligned} \boldsymbol{p}_i &= \boldsymbol{J}\,\boldsymbol{W}\,\boldsymbol{p}_w \\ &= \boldsymbol{J}\,\boldsymbol{W}_i\,\boldsymbol{Q}\,[a'\ b'\ d'\ 1) \end{aligned}$$

wobei W_i eine 4x4 Matrix ist, die die Kamerageometrie beschreibt. Q ist aus Gleichung (1), und J ist die einfache 3x4 Projektionsmatrix

$$J = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Für einen transformierten und projizierten Punkt p_i wird die dazugehörige Bildkoordinate p'_i durch folgende Relation berechnet

$$\boldsymbol{p'_i = H\,(p_i)}$$

wobei H die homogene Funktion

$$H\left(\begin{bmatrix} x \\ y \\ h \end{bmatrix}\right) = \begin{bmatrix} x/h \\ y/h \end{bmatrix}$$

ist.

Der Pixelintensitätsvektor $v_{(a',b',d')}$ wird in Abhängigkeit von den Würfelpunkten (a',b',d') definiert als

$$v_{(a',b',d')} = \{I_i\,(H(JW_iQ[a' \quad b' \quad d' \quad 1]^T)), \forall i \in [1,\ldots,n]\}$$

wobei $I_i\,([x' \ y']^T)$ die Intensität des Pixels $[x' \ y']^T$ in Bild i ist. Dieser Vektor enthält also alle Pixelintensitäten aus allen Kamerapositionen für einen Punkt (a',b',d') des Würfels.

2.2 „Matching“ Kosten

Um „stereo matching“ durchführen zu können, wird eine entsprechende Kostenfunktion benötigt. Im Idealfall ist sie minimal für einen Treffer und maximal für keinen Treffer. Dabei ist es nicht trivial, eine passende Funktion zu finden, geschweige denn eine Funktion, die in Polynomialzeit global minimiert werden kann. Bis jetzt war dynamisches Programmieren ein effizienter Weg um Kostenfunktionen zu minimieren und weiche Übergänge, die schon immer sehr wichtig für das Stereo „matching“ waren, zu ermöglichen. Nebeneffekt dieser Methode war, dass die Kostenfunktion geschwächt werden musste, damit weiche Übergänge entstehen. In Roy (1999) ist diese Einschränkung beim „maximum flow“ Ansatz nicht mehr vorhanden, trotzdem bleibt die „maximum flow“ Minimierung effizient. Im Folgenden wird die Kostenfunktion beschrieben.

Wenn man davon ausgeht, dass die einzelnen Pixel auf den Oberflächen aus cooljedem Blickwinkel die gleiche Intensität haben, dann sind die Pixelintensitäten (Komponenten von $v_{(a',b',d')}$) identisch, wenn (a',b',d') auf der Oberfläche ein und desselben Objektes liegt und damit ein gültiger Treffer ist. Dann kann man die Kostenfunktion cost(a',b',d') als die Varianz der Pixelintensitäten von $v_{(a',b',d')}$ darstellen

$$\text{cost}(a', b', d') = \frac{1}{n}\sum_{i=1}^{n}\left(v_{(a',b',d')i} - \overline{v_{(a',b'd')}}\right)^2 \quad (3)$$

wobei $\overline{v_{(a',b'd')}}$ der Mittelwert der Komponenten von $v_{(a',b',d')}$ ist.

3. Berechnung einer kompletten Tiefenkarte

Typischerweise wird „stereo matching“ entlang Epipolarlinien durchgeführt, um einen effizienten Algorithmus, wie z.B. das dynamische Programmieren, benutzen zu können. Eine Weiterführung dessen wäre, die Berechnung in einem Schritt auf das gesamte Bild auszuweiten (siehe Abb.4), indem man alle Paare von Epipolarlinien gleichzeitig testet. Der „matching“ Körper wird durch zwei Achsen (a,b), die die Bildpixel repräsentieren, und einer Tiefenachse d aufgespannt. Die Tiefenkarte stellt sämtliche berechnete Tiefeninformationen des verwendeten Bildes dar. Der Vorteil dieser Konstruktion ist, dass die „local coherence“ Eigenschaft ausgenutzt werden kann. Diese besagt, dass Tiefeninformationen in der näheren Pixelumgebung nicht nur entlang von Epipolarlinien, sondern auch über mehrere Linien hinweg, sehr ähnlich sind.

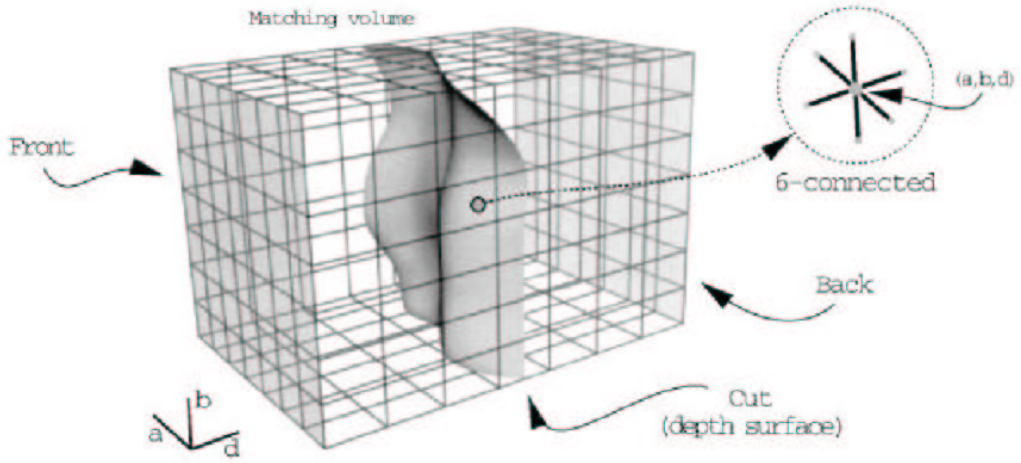

Abb.4: „Matching“ des gesamten Bildes. „Front“ und „Back“ entsprechen dem nahesten und entferntesten Tiefenschritt. Die „Depth surface“ teilt den Körper und trennt „Front“ und „Back“ voneinander

Dynamische Programmieren kann in diesem Zusammenhang nicht mehr benutzt werden, da die zweidimensionale Betrachtung nicht die gleichen Methoden wie die eindimensionale Betrachtung entlang der Epipolarlinien zulässt.

Viele Methoden für eine globale Tiefenberechnung wurden vorgestellt (Cox, 1994; Ohta und Kanade, 1985; Belhumeur, 1996). Typischerweise arbeiteten diese Algorithmen nach der Methode, eine Lösung iterativ zu verbessern, wobei sie vorhergehende Lösungen von Nachbarlinien benutzen. Dieses kann in der Praxis manchmal gute Ergebnisse liefern, jedoch sind diese Algorithmen nicht unbedingt effizient und optimal, da sie nicht das globale Minimum ihrer Kostenfunktion finden können.

4. „Stereo matching“ als „maximum flow“ Problem

Um das Problem zu lösen, wird es in Roy (1999) als ein Flussproblem in einem Graph interpretiert (siehe Abb.5). Dafür wird zum Aufbau in Abb.4 eine Quelle und eine Senke hinzugefügt. Es handelt sich dabei um ein Flussnetzwerk, bei dem jede Kante eine feste Flusskapazität hat und jeder Knoten als Verbindungsstück betrachtet wird. Die Stärke des Flusses, der in einer dieser Knoten hineinfließt, ist gleich der Stärke des Flusses, der wieder aus ihm herausfließt. Dieses nennt man die „flow conservation“ Eigenschaft. Das zu lösende „maximum flow“ Problem beschäftigt sich nun mit der Frage, wie groß der maximale Fluss ist, der von der Quelle zur Senke fließen kann, ohne dass die Kapazität der einzelnen Kanten überschritten wird (Cormen et al. (1990). Bezugnehmend zum „max-flow min-cut theorem“ (Cormen et al. 1990) bilden diejenigen Kanten, die durch den maximalen Fluss gesättigt sind, einen minimalen Schnitt durch den Graphen. Indem die Quelle bzw. Senke mit der Vorder- bzw. Rückseite verbunden wird (Abb.5), bildet ein Schnitt zwischen diesen eine effiziente Tiefenkarte. Zudem repräsentiert ein minimaler Schnitt nun die gesuchte Tiefenkarte mit minimalen Kosten.

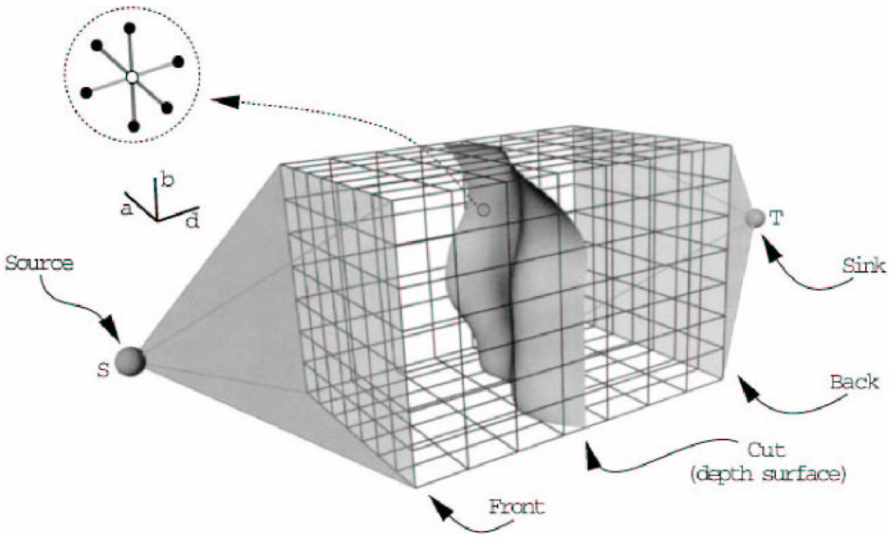

Abb.5: „Matching“ als ein „maximum flow“ Problem

Man bezeichne den Graph aus Abb. 5 als G = (*V*,*E*). Die Knoten V werden definiert als

$$V = V^* \cup \{s,t\}$$

wobei *s* die Quelle, *t* die Senke und *V** die 3D Maschen

$$V^* = \{(a',b',d') : 0 \le a' < a'_{size}, 0 \le b' < b'_{size}, 0 \le d' < d'_{size}\}$$

sind, bei denen (a'_{size}, b'_{size}) die Basisbildgröße und d'_{size} die Tiefen- bzw. „disparity“-Auflösung ist.

Die Maschen sind intern sechsfach verbunden. Es gibt zwei disjunkte Gruppen von Knoten (V_{front}, V_{back}), die die Vorder- und Rückseite des Graphen darstellen, so dass die Quelle mit jedem Knoten von V_{front} und die Senke mit jedem Knoten von V_{back} verbunden ist. In Roy (1999) sind sie definiert als

$$V_{front} = \{(a', b', 0)\}$$
$$V_{back} = \{(a', b', d'_{size})\}$$

Die Kanten des Graphen werden definiert als

$$E = E_{label} \cup E_{penalty} \cup E_{in} \cup E_{out}$$

mit

$$E_{in} = \{(s, u) : u \in V_{front}\}$$
$$E_{out} = \{(u, t) : u \in V_{back}\}$$
$$E_{label} = \{(u, v) \in V^* \times V^* : u - v = (0,0,\pm 1)\}$$
$$E_{penalty} = \{(u, v) \in V^* \times V^* : u_{d'} = v_{d'} \wedge (v_{a'}, v_{b'}) \in N(u_{a'}, u_{b'})\}$$

wobei ($u_{a'}$,$u_{b'}$,$u_{d'}$) und ($v_{a'}$,$v_{b'}$,$v_{d'}$) die *a'*, *b'* und *d'* Komponenten der Knoten *u* und *v* und *N*(*a'*,*b'*) die Nachbarn der Pixel (*a'*,*b'*) sind. In Roy (1999) wird die Vierfachnachbarschaft definiert als

$$N(a', b') = \{(a' \pm 1, b'), (a', b' \pm 1)\}$$

Bezugnehmend zur Abb.5 bilden die Kanten zwischen der Quelle und der Vorderseite die Gruppe E_{in}, während E_{out} die Kanten zwischen der Rückseite und der Senke sind. Die Kanten E_{label} drücken die Pixel-„matching"-kosten aus, und beinhalten alle Kanten parallel zur d-Achse. Die Kanten $E_{penalty}$ wirken sich auf die Weichheit des Bildes aus und beinhalten alle Kanten auf der (*a*,*b*) Ebene. Wenn der minimale Schnitt berechnet ist, werden die Kanten in $E_{penalty}$ verworfen, während diejenigen in E_{label} die gesuchten Tiefen präsentiert. Die unterschiedliche Rolle zwischen „label" und „penalty" wird in Abschnitt 4.1 und 4.2 genauer beschrieben.

In Roy (1999) wird die Kantenkapazität im Graphen nach einem einfachen System berechnet. Die Verbindungen zur Quelle bzw. Senke haben unendliche Kapazitäten. Jeder Knoten (*a'*,*b'*,*d'*) im Graphen repräsentiert einen potenziellen Treffer, der einem Pixel (*a*,*b*) einen Tiefenwert zuordnet, so dass wir Gleichung (3) zur Berechnung der Kosten hinzuziehen können. Als Kosten der Knoten werden direkt die Kapazitäten der „label" Kanten übernommen, die zu diesen Knoten gehören. Für die weichen Übergänge bekommen die „penality" Kanten eine konstante Kapazität. Die Kantenkapazität *c*(*u*,*v*) von Knoten *u* nach *v* ist definiert als

$$c(u,v) = \begin{cases} 0 & \text{wenn } (u,v) \notin E \\ \infty & \text{wenn } (u,v) \in E_{in} \text{ oder } (u,v) \in E_{out} \\ \text{cost}(u) & \text{wenn } (u,v) \in E_{label} \text{ und } u_{d'} < v_{d'} \\ \text{cost}(v) & \text{wenn } (u,v) \in E_{label} \text{ und } u_{d'} > v_{d'} \\ K & \text{wenn } (u,v) \in E_{penalty} \end{cases} \quad (4)$$

wobei K der „smoothness" Faktor ist.

Der minimale Schnitt C_{min}, der durch die Berechnung des „maximum flow" über *G* entsteht, beinhaltet eine Gruppe von Kanten mit minimaler Kapazität, die die Quelle und die Senke trennt.

C_{min} wird definiert als

$$\arg\min_{C}\left(\sum_{(u,v)\in C} c(u,v)\right) = \arg\min_{C}\left(\sum_{(u,v)\in C_{label}} c(u,v) + \sum_{(u,v)\in C_{penalty}} c(u,v)\right)$$

wobei $C_{label} = C \cap E_{label}$ und $C_{penalty} = C \cap E_{penalty}$ ist. In Roy (1999) wird die „labelling" Funktion $L(a',b')$ als die kleinste d' Komponente der Knoten einer Kante in C_{label} definiert. So eine Kante hat die Form

$$((a',b',L(a',b')),(a',b',L(a',b')+1))$$

oder

$$((a',b',L(a',b')+1),(a',b',L(a',b'))).$$

Man beachte, das beide Formen dieselbe Kapazität haben: $cost(a',b',L_{(a',b')})$. Wenn man die Kantenkapazitäten bezüglich Gleichung (4) ändert, bekommt man

$$\min_{C}\left(\sum_{(u,v)\in C_{label}} \text{cost}(u) + \sum_{(u,v)\in C_{penalty}} K\right) = \min_{C}\left(\sum_{((a',b',L(a',b')),v)\in C_{label}} \text{cost}(a',b',L(a',b')) + \sum_{((a',b',L(a',b')),v)\in C_{penalty}} K\right) \quad (5)$$

$$= \sum_{\forall(a',b')} \text{cost}(a',b',L^*(a',b')) + \frac{K}{2} \sum_{\forall(a',b'),\forall(i',j')\in N(a',b')} \left|L^*(a',b') - L^*(i',j')\right|$$

wobei $L^*(a',b')$ die „labelling" Funktion ist, die zum minimalen Schnitt C_{min} gehört. Diese Transformation ist möglich, weil der minimale Schnitt eine Eigenschaft hat, die besagt, dass für jedes (a',b') genau eine Tiefe $L(a',b')$ existiert, so dass die Kante $((a',b',L(a',b')), (a',b',L(a',b')+1))$ zu C_{label} gehört. Diese Eigenschaft wird in Abschnitt 4.2 diskutiert.

In Roy (1999) kann man sehen, dass die Anzahl der „penalty" Kanten im minimalen Schnitt direkt von den Tiefenunterschieden zwischen benachbarten Pixel abhängt. Dieses wird intuitiv dadurch erklärt, dass in dem Fall, dass zwei benachbarte Pixel unterschiedliche Tiefen haben, ein Loch zwischen den beiden „label" Kanten entsteht. Dieses wird „geschlossen", indem dieser Schnitt mit einer Anzahl von „penalty" Kanten überbrückt wird, deren Menge vom Tiefenunterschied zwischen den Pixel abhängt.

Zusammenfassend kann gesagt werden, das die Kostenfunktion von Gleichung (5) das Finden einer Tiefendarstellung, die einen „pixelmatching"- und einen „smoothness"- Term global minimiert, beinhaltet. Diese „smoothness" Kosten werden durch den Faktor K ausgedrückt.

4.1 Weiche Übergänge durch Kantenkapazitäten

Durch die Unterteilung von E in zwei Kantengruppen, werden die „penalty" Kanten als Regler für „smoothness" in der Tiefendarstellung verwendet (zweiter Teil der Gleichung (5)). Wie in Abb.6 gezeigt wird, sind die „penalty" Kanten diejenigen, die nicht an der Tiefenachse d ausgerichtet sind. Wie in der Gleichung (4) gezeigt, definieren die Trefferkosten die Kapazität der „label" Kanten, während „penalty" Kanten den konstanten Wert K zugeordnet bekommen, die intuitiv auch Verdeckungskosten genannt werden können. In Abb.6 sind die dunkleren Kanten zwischen den schwarzen Punkten die „penalty" Kanten, während die helleren Kanten die „label" Kanten sind. Höhere Verdeckungskosten (z.B. durch größeres K) steigern die „smoothness" der berechneten Tiefendarstellung, wobei geringere Kosten das Berechnen von Tiefensprüngen erleichtern.

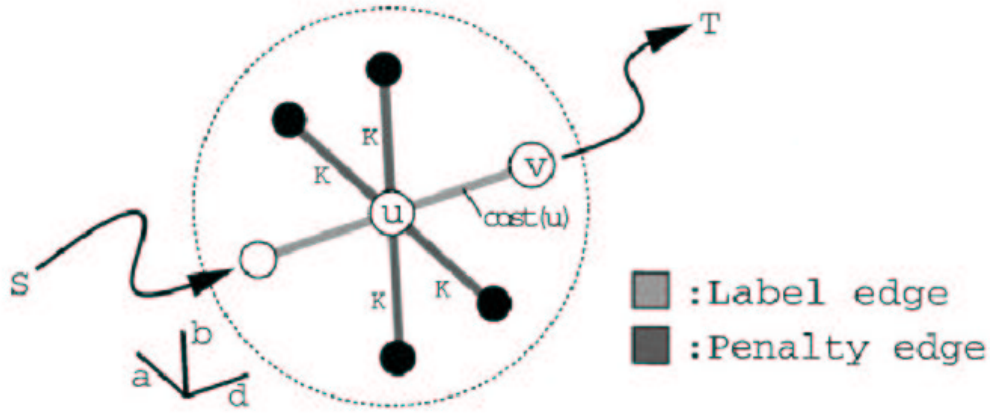

Abb.6: „Smoothness“ durch Kantenkapazitäten

Der Effekt des Parameters *K* wird in Abb.7 mit einem einfachen 2D Beispiel und einfacher Kostenfunktion illustriert. Der minimale Schnitt dieses Graphen wird für drei verschiedene *K* (0,1, ∞) berechnet und in Abb.7 als dicke Linien dargestellt. Man beachte, dass die „label“ Kanten, die die Lösung des Problems darstellen, horizontal sind. Diese extremen Werte für *K* haben intuitiv klare Auswirkungen. Für *K*=0 wird jeder Zeile des Graphen unabhängig eine Tiefe zugewiesen, was hier die Tiefenunterschiede maximiert. Ist *K*=∞, dann ist die daraus resultierende Tiefendarstellung flach (maximal „weich“) und es wird nur ein einziger Tiefenwert für das gesamte Bild berechnet. Für *K*=1 ist eine Balance zwischen den benötigten Trefferkosten und „smoothness“ erreicht.

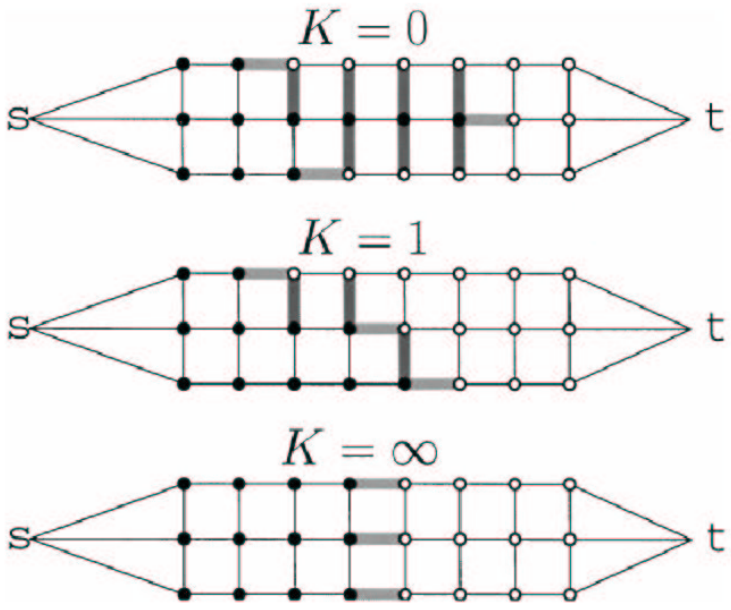

Abb.7: Beispiele für verschiedene *K*. *K*=0, maximale Tiefenunterschiede; *K*=1, durchschnittliche „smoothness“; *K*=∞, unendliche „smoothness“.

4.2 Vom Schnitt zur Tiefendarstellung

Das „maximum flow-minimum cut“ Theorem besagt, dass wenn der „maximum flow“ gefunden wurde, ein „minimum cut“ die Quelle von Senke auf solche Weise trennt, das die Summe der Kantenkapazitäten von C_{min} minimiert ist. Dieser Schnitt ist zudem global optimal für die in Roy (1999) gewählte Kostenfunktion. Um die Kostenfunktion aus Gleichung (5) abzuleiten, wird eine wichtige Bedingung der Graphformulierung verwendet und zwar dass der minimale Schnitt garantiert exakt einen Tiefenwert für jeden Pixelwert liefert. Diese Bedingung kann auf verschiedene Arten verifiziert werden.

Man kann davon ausgehen, dass eine Kapazitätsfunktion mit einer konstanten Kapazität K für alle „penalty“ Kanten (wie in Gleichung (4)) diese Bedingung erfüllt.

Wie bei Boykov et al. (1998) beschrieben, kann zur Gleichung (3) eine große Konstante hinzugefügt werden. Die Kapazitätsfunktion (Gleichung (4)) wird dann zu

$$c(u,v) = \begin{cases} \ldots \\ \text{cost}(u) + B \text{ wenn } (u,v) \in E_{label} \wedge u_{d'} < v_{d'} \\ \text{cost}(v) + B \text{ wenn } (u,v) \in E_{label} \wedge u_{d'} > v_{d'} \end{cases}$$

wobei der Wert B größer als die Summer alle „penalty“ Kanten des Graphen ist. Dieses fügt zur Funktion in Gleichung (5) eine Konstante hinzu, wobei sichergestellt ist, dass das Minimum der modifizierten Funktion gleich bleibt.

Ishakawa und Geiger (1998) schlagen nun vor, denjenigen „label“ Kanten, die von der Senke zur Quelle führen, unendliche Kapazitäten zuzuteilen. Die Kapazitätsfunktion ist nun die modifizierte Gleichung (4)

$$c(u,v) = \begin{cases} \ldots \\ \text{cost}(u) \text{ wenn } (u,v) \in E_{label} \wedge u_{d'} < v_{d'} \\ B \text{ wenn } (u,v) \in E_{label} \wedge u_{d'} > v_{d'} \end{cases}$$

wobei B unendlich ist (Ishikawa und Geiger, 1998). Das bedeutet, es ist hinreichend und praktisch sinnvoll, B als einen Wert, der größer als die Summe aller „penalty“ Kanten des Graphes ist, zu definieren.

Man beachte, dass die beiden vorhergehenden Lösungen (Boykov et al., 1998; Ishikawa und Geiger, 1998) keine Angabe zu den Kapazitäten der „penalty“ Kanten machen. Dieses hält die Wahl dieser Kapazitäten flexibel und es könnte in Zukunft bewiesen werden, ob das sinnvoll ist.

Die komplette Tiefenansicht kann nun sehr einfach aus dem minimalen Schnitt von G konstruiert werden. Für jeden Punkt (a',b') ist die Tiefe $L^*(a',b')$, wenn die Kante $(a',b',L^*(a',b')) – (a',b',L^*(a',b')+1)$ zu C_{min} gehört, wie in Gleichung (5) beschrieben.

4.3 Lösen des „maximum flow“ Problems

Es gibt eine Anzahl an Algorithmen, die das Problem lösen (Cormen et al., 1990; Goldberg und Rao, 1997). In Roy (1999) wird ein sehr bekannter Algorithmus implementiert, der

„preflow-push lift-to-front“ Algorithmus (Cormen et al., 1990). Momentan ist der wohl schnellste Algorithmus von Goldberg und Rao (1997), der am besten mit Graphen arbeitet, die für das „stereo matching“ entworfen wurden.

Die Anzahl der Knoten v in gleich der Anzahl der Bildpixel multipliziert mit der Tiefenauflösung. Bei einem Bild mit der Größe $s = ab$ Pixel und einer Tiefenauflösung d gilt $v = sd$. Da der Graph eine dreidimensionale Menge aus Maschen und jeder Knoten sechsfach verbunden ist, ist die Anzahl der Kanten $e \approx sd$.

Dieses impliziert, dass der benutzte „preflow-push“ Algorithmus mit einer Laufzeit von

$$O(ve\log(v^2/e))$$

für unser Problem die Laufzeit

$$O(s^2 d^2 \log(sd))$$

hat.

Der momentan schnellste Algorithmus von Goldberg und Rao (1997) läuft in

$$O(e^{\frac{3}{2}} \log(v^2/e)\log(U))$$

wobei U die größte Kantenkapazität ist. Dieses ist im Beispiel von Roy (1999) eine Konstante, da die Pixel hier endliche Werte besitzen. Die Laufzeit beträgt dann

$$O(s^{1,5} d^{1,5} \log(sd)\log(U)) .$$

Roy (1999) benutzt nicht diesen Algorithmus, da der Geschwindigkeitsvorteil nur im „worst case“ auftritt und nicht im „average case“. Es würde laut Roy (1999) gegenüber dem „preflow-push“ Algorithmus keine signifikanten Verbesserungen im „average case“ ergeben.

Der Ansatz von Cox et al. (1996), der dynamisches Programmieren verwendet, benötigt eine Laufzeit von $O(sd)$, was sehr viel besser als der „maximum flow“ Algorithmus erscheint. Dabei ist jedoch zu beachten, dass die „average case“ Laufzeit durch z.B. die Topologie des Graphen, die Position der Quelle und der Senke und die Struktur der Kantenkapazitäten sehr stark verbessert werden kann. In Abb.8 wird die Geschwindigkeit als eine Funktion der Bildgröße s (in Pixel) und Tiefenauflösung d definiert. Die mittlere Laufzeit beträgt $O(s^{1,2}d^{1,3})$, was fast linear zur Bildgröße s (in Pixel) und vergleichbar mit dem Ansatz, bei dem dynamisches Programmieren verwendet wird, ist. Die typische Laufzeit für ein 256x256 Pixel Bild liegt bei einem Pentium 166 irgendwo zwischen 1 und 30 Minuten, abhängig von der Tiefenauflösung. Dieses ist sehr wahrscheinlich langsamer als Cox et al. (1996), da deren Algorithmus auf Geschwindigkeit optimiert ist. Der Algorithmus von Roy (1999) ist noch nicht optimiert. Für die Zukunft werden Verbesserungen erwartet.

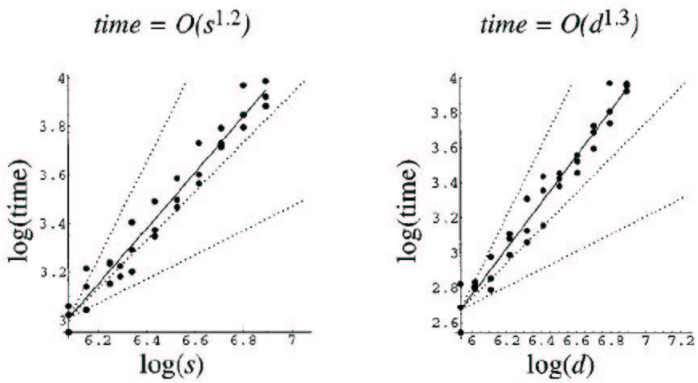

Abb.8: (links) Geschwindigkeit als eine Funktion von *s* für fixierte Tiefenauflösung. (rechts) Geschwindigkeit als eine Funktion der Tiefenauflösung *d* für ein fixiertes *s*. Die drei gepunkteten Linien zeigen die „performance level" von $O(\sqrt{s})$, $O(s)$ und $O(s^2)$.

5. Experimente und Ergebnisse

In diesem Abschnitt werden Ergebnisse von binokularen und multiokularen „maximum flow" Algorithmen vorgestellt und mit zwei anderen Algorithmen, die auf dynamischem Programmieren basieren, verglichen. Diese Vergleiche sind manchmal schwierig, da viele Stereo Algorithmen nicht mehr als zwei Kameras verwenden können.

Der erste zu vergleichende Algorithmus ist der „standard stereo", der Zeile für Zeile die Ergebnisse mit dynamischem Programmieren, *n* Kameras und variabler Tiefenauflösung berechnet. Er unterscheidet sich vom „maximum flow" nur in der Art, wie er zum Schluss die Tiefenkarte berechnet. Ansonsten sind sie identisch, die Ergebnisse benutzen die gleiche Tiefenskala und sind nicht angeglichen. Mit einem Angleich meint Roy (1999) eine ergebnisunabhängige, nicht lineare Transformation, die zur Kontrastverbesserung angewandt wird. Die bekannteste Methode dazu ist „histogram equalization". Leider ist auch dadurch ein fairer Vergleich der Ergebnisse sehr schwierig, wenn er überhaupt möglich ist.

Der zweite Algorithmus ist MLMH+V, der auf einer effizienten dynamische Programmierung von Cox et al. (1996) (binocular) und Cox et al. (1994) (multiokular) basiert. Er unterscheidet sich vom vorhergehenden Algorithmus in der Tatsache, dass er die Ergebnisse der Linien iterativ verbessert, um auch senkrecht zu den Epipolarlinien weiche Übergänge zu erreichen. Es muss dabei beachtet werden, dass die Ergebnisse dieses Algorithmus eine andere Grauskala verwendet und deswegen angepasst werden mussten.

Random Dot Stereogram

Um die Symmetrie in der Tiefenkarte vom „maximum flow" Algorithmus zu demonstrieren, wurde in Roy (1999) das „random-dot" Stereogramm als Bild (siehe Abb.9) mit Verschiebungen von 0,4 und 8 Pixel verwendet. Die berechnete Tiefenkarte (siehe Abb.10) unterscheidet sich vor allem in Bereichen mit Tiefensprüngen. „Maximum flow" berechnet dabei glatte Kanten, während „standard stereo" Artefakte erzeugt, was daran liegt, dass die Ergebnisse alle horizontal berechnet und vertikal keine Informationen weitergegeben werden.

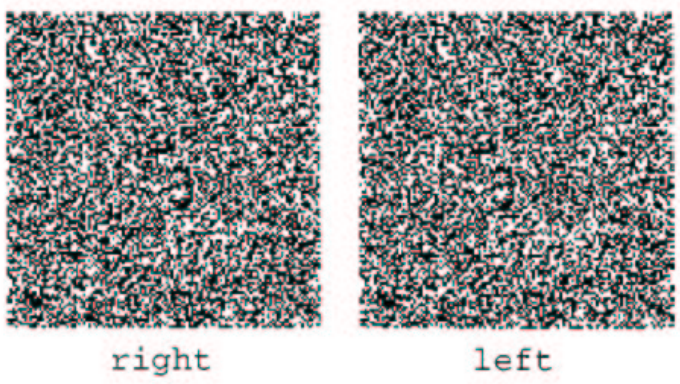

Abb.9: „Random dot" Stereogramm

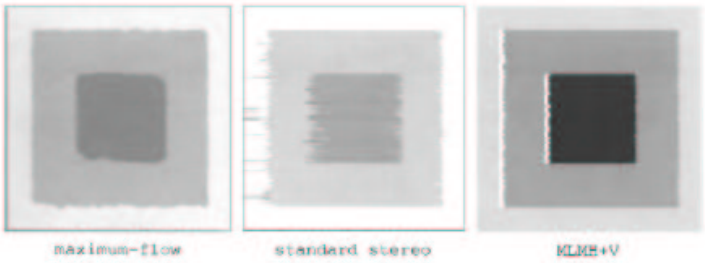

Abb.10: Tiefenkarte für das “random dot“ Stereogramm

Granite

Abb.11 zeigt den Kamera- und Szenenaufbau für eine Sequenz von fünf Ansichten einer weich texturierten Oberfläche. Die einzelnen Kamerabilder sieht man in Abb.12.

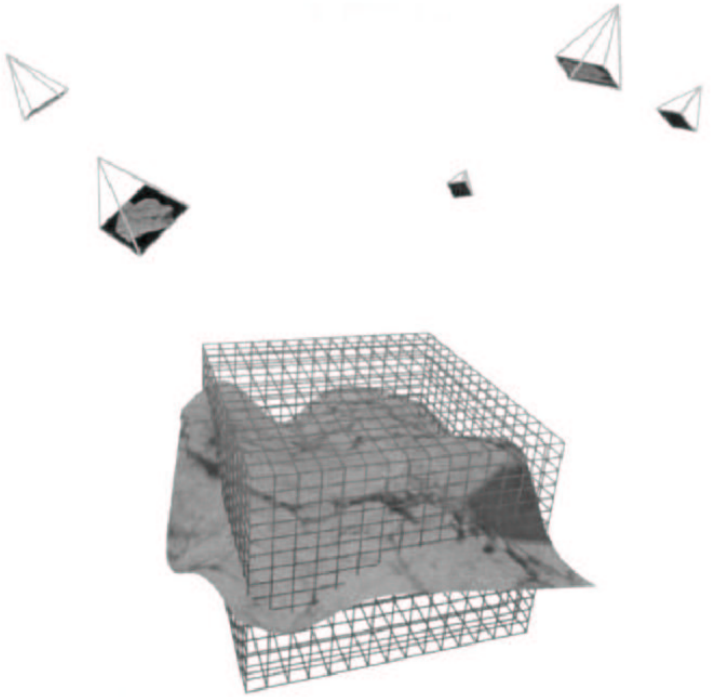

Abb.11: Die „granite“ Szene und der Kameraaufbau. Bei dem Gitter handelt es sich um den „matching“ Körper

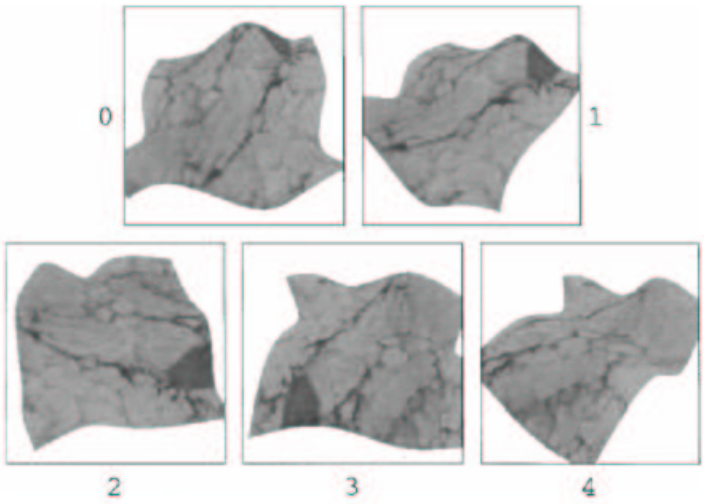

Abb.12: Die Kamerabilder (256x256 Pixel)

Die Ergebnisse für eine unterschiedliche Anzahl von Bildern und verschiedenen *K* werden in Abb.13 und 14 gezeigt. Bei *K*=0 handelt es sich um eine direkte Suche, wodurch eine sehr stark rauschende Tiefenkarte berechnet wird. In Abb.14 wird die Genauigkeit der Tiefenkarte als Funktion über K für zwei bis fünf Kameras abgebildet. Es ist dabei nicht überraschend, dass mit mehr Kameras auch die Genauigkeit größer wird. Auch bei der Wahl von *K* ist es auf jeden Fall besser ein kleines *K* zu wählen als *K*=0. Zudem sinkt die Genauigkeit mit großer Zunahme von *K* nur gering. Das bedeutet, dass der „maximum flow" Algorithmus unanfällig gegenüber einer schlechten Wahl von *K* ist.

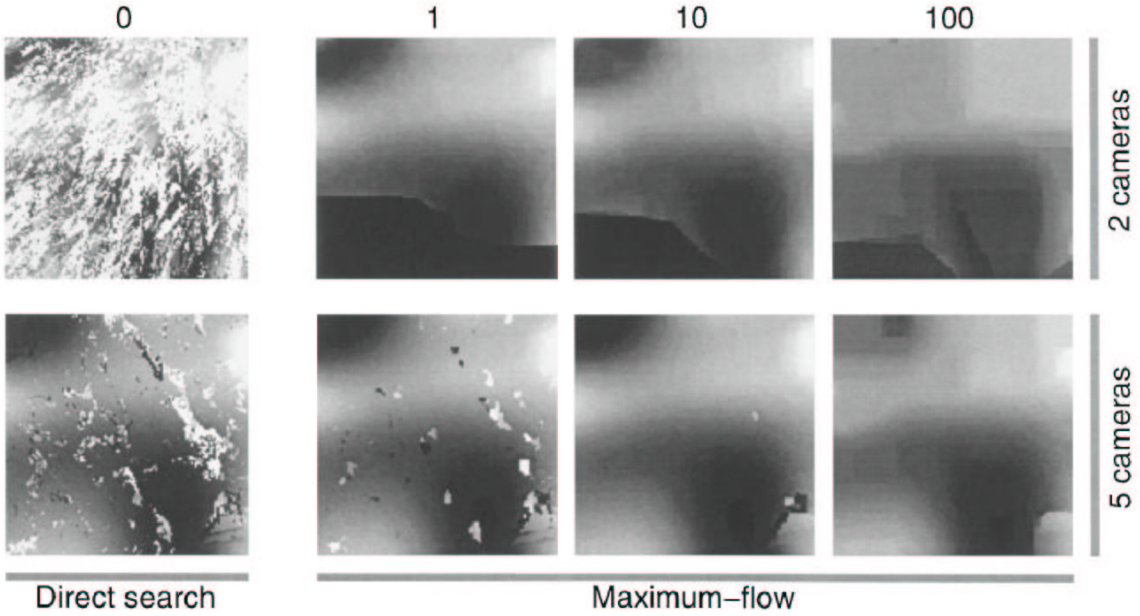

Abb.13: „Granite" Ergebnisse. K = 0 bis 100, zwei und fünf Kameras

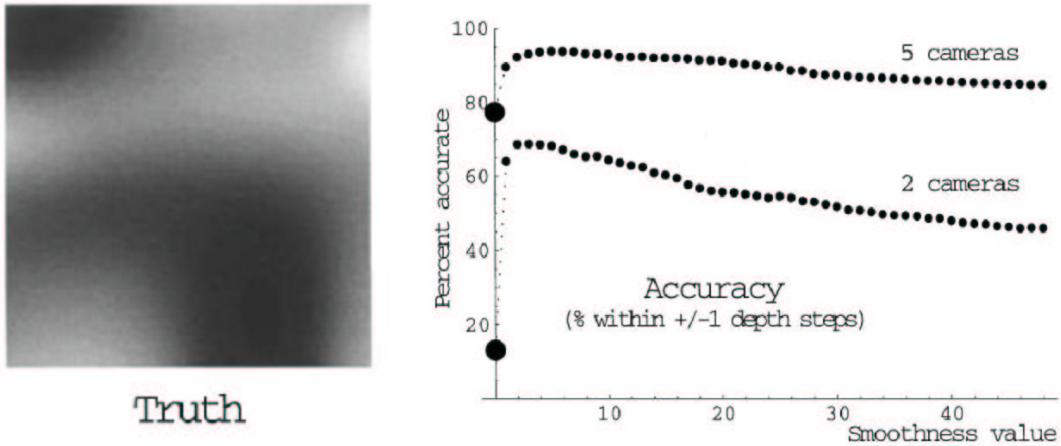

Abb.14: Die „granite" Ergebnisse

Shrub

Abb.15 zeigt ein Bilderpaar der „shrub" Bildsequenz (von T. Kanade und T. Nakahara vom CMU). Die Ergebnisse in Abb.16 zeigen, wie es „maximum flow" schafft, scharfe und präzise Tiefensprünge zu zeichnen, während „standard stereo" und MLMH+V viele vertikale Artefakte erzeugen. Es gibt bzgl. Der Tiefenauflösung zwei Stufen (32 und 128 Schritte) mit unterschiedlichem „smoothness" Faktor *K*. Es ist hervorzuheben, dass „maximum flow" sogar bei einem großen *K* keine horizontalen Verbindungen zwischen der Lücke in dem Strauch produziert. Das Ergebnis von mehr als zwei Kameras sieht man in Abb.17. Alle Bilder dieser

Sequenz besitzen die gleiche horizontale Basislinie. Obwohl die Anzahl der Kameras unterschiedlich ist (4 und 7), ist die größte Spanne zwischen zwei Kameras dieselbe und demnach ist auch die Stärke der Verschiebung gleich. Bei MLMH+V wurde das Ergebnis wieder etwas normalisiert, bei den anderen beiden Algorithmen nicht.

Abb.15: Das „shrub“ Stereopaar

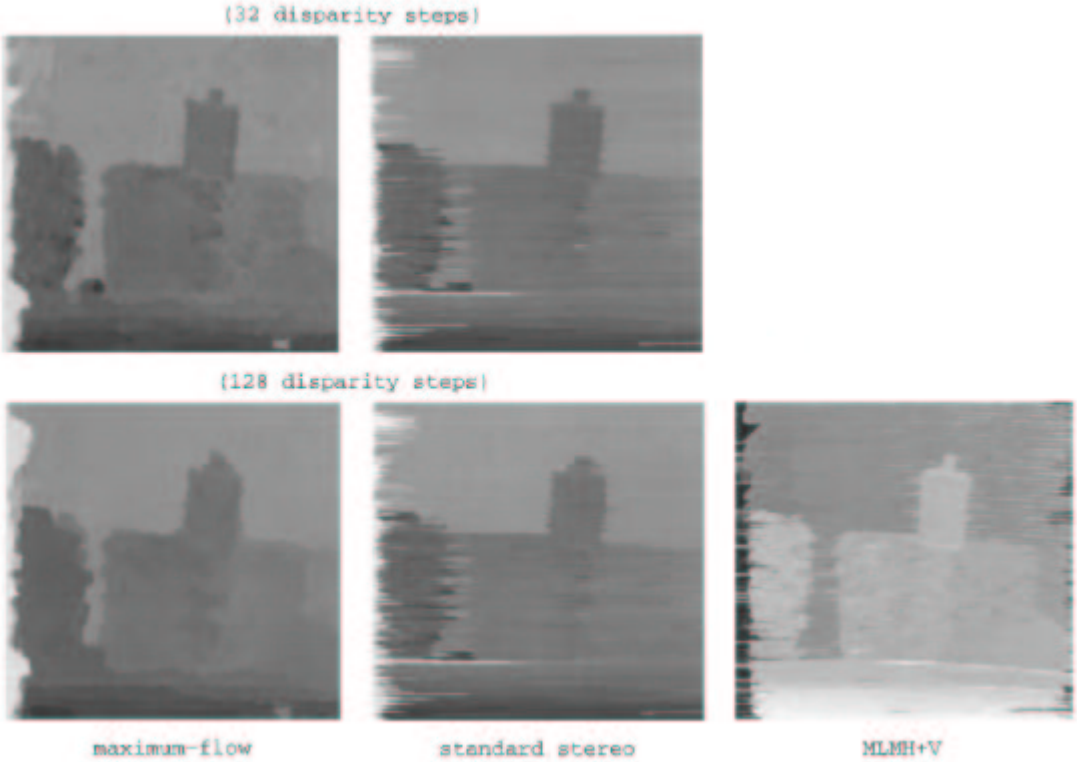

Abb.16: Tiefenkarten für „shrub“ mit zwei verschiedenen Tiefenauflösungen. (links) Die Ergebnisse vom „maximum flow“ Algorithmus. (mitte) Ergebnisse von „standard stereo“. (rechts) Ergebnisse von „MLMH+V

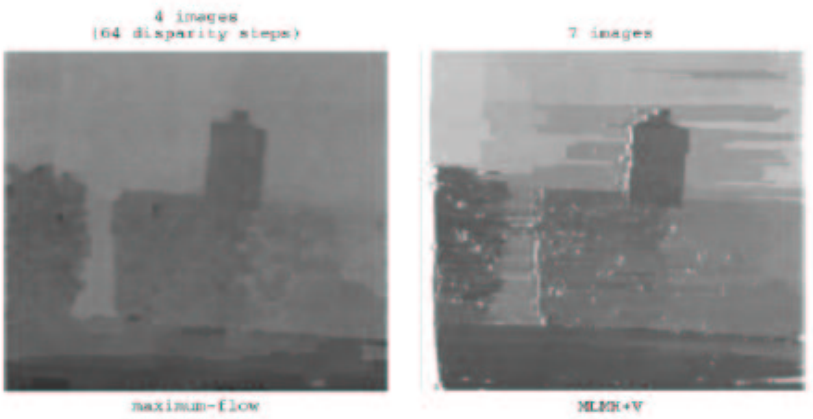

Abb.17: Tiefenkarten für die „shrub“ Sequenz mit vier bzw. sieben Bildern. Weiße Punkte im rechten Bild zeigen erkannte Verdeckungen

Pentagon

Das Stereopaar „pentagon“ wird in Abb.18 gezeigt. Die Ergebnisse sind in Abb.19 zu sehen. Dieses Beispiel ist eine Herausforderung, da die Kamerabewegung nicht vollständig horinzontal ist und eine kleine Drehung beinhaltet, die eine Verletzung der Epipolarbedingungen bedeutet. MLMH+V löst das Problem ganz gut, da es sowohl negative als auch positive Verschiebungen zulässt. Daher wird auch der Highway oben links im Bild sehr gut dargestellt, während andere Algorithmen dort Probleme haben. Der „maximum flow“ Algorithmus zeichnet ein sehr symmetrisches Ergebnis, mit wenig horizontalen Artefakten.

Abb.18: Das „pentagon“ Stereopaar

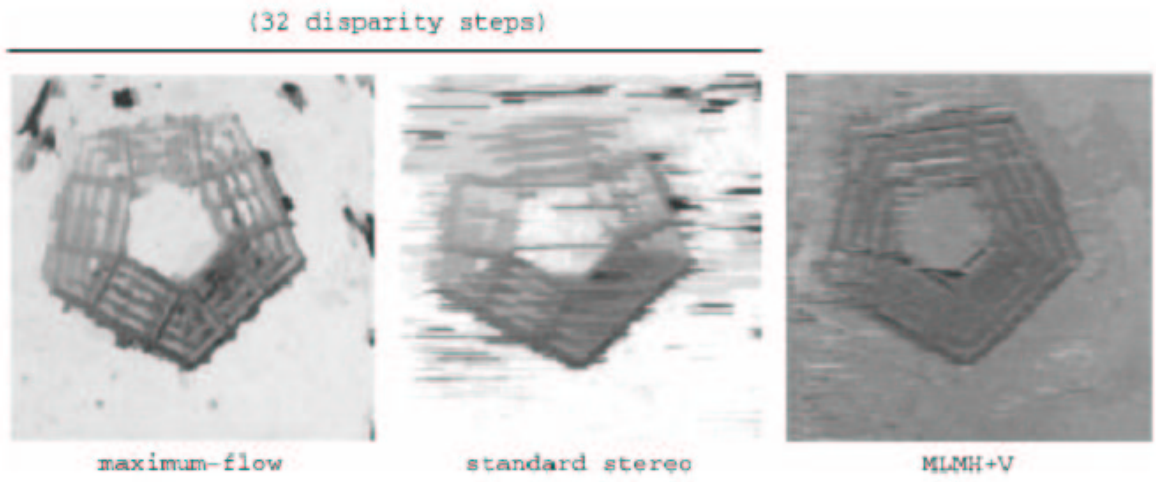

Abb.19: Tiefenbilder für das „pentagon“ Beispiel

„Park meter“

Die Bildersequenz „park meter“ (siehe Abb.20) wurde mit unterschiedlichen Anzahlen von Bildern berechnet. Die Ergebnisse des binokularen Falles werden in Abb.21 gezeigt. Darin ist zu erkennen, dass einige vertikale Objekte dem „standard stereo“ und dem MLMH+V Algorithmus Probleme bereiten. Beim „maximum flow“ sind wieder keine horizontalen Artefakte zu erkennen. Bei der Variante mit vier Bildern (siehe Abb.22) ist das Ergebnis um einiges besser als in Abb.21. Es gab hier jedoch keine Ergebnisse für den MLMH+V Algorithmus.

Abb.20: Das „park meter“ Beispiel

Abb.21: Tiefenbilder für das „park meter“ Beispiel. Es wurden zwei Bilder verwendet.

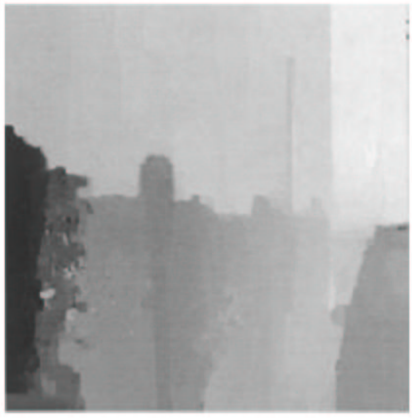

Abb.22: Tiefenbild für das „park meter“ Beispiel. Hier wurden vier Bilder verwendet. Der „matching“ Körper hat die Größe 256x240x64.

„Roof“

Die Bildsequenz „roof“ wird in Abb.23 gezeigt. Sie beinhaltet 13 Bilder mit sowohl vertikalen als auch horizontalen Verschiebungen. Die Egebnisse für „maximum flow“ und MLMH+V werden in Abb.24 gezeigt. Man kann sehen, dass das Tiefenbild, das durch „maximum flow“ berechnet wurde, sehr detailliert ist, die Struktur des Daches wurde dabei sehr gut rekonstruiert. Abb.25 zeigt eine 3D Rekonstruktion der „roof“ Sequenz, basierend auf der Tiefenkarte, die vom „maximum flow“ Algorithmus berechnet wurde. Es zeigt sich, dass auch kleine Details sehr gut rekonstruiert werden können.

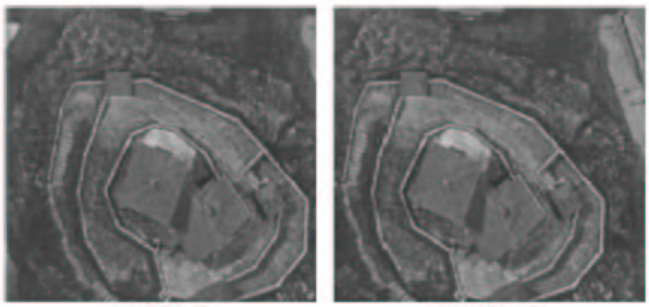

Abb.23: Zwei Bilder der „roof“ Sequenz.

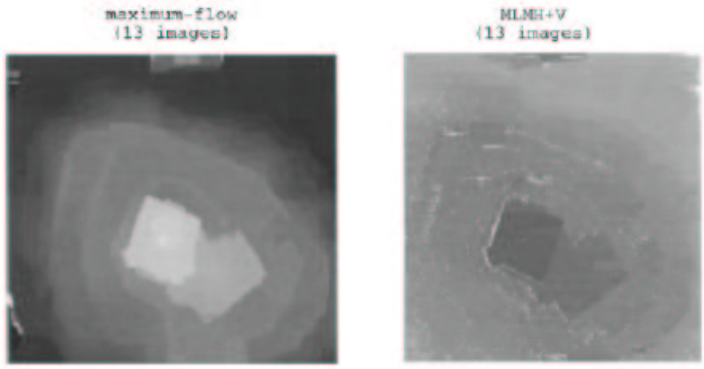

Abb.24: Tiefenbilder der „roof“ Sequenz. Es wurden 13 Bilder verwendet. Weiße Punkte auf dem rechten Bild zeigen Verdeckungen. Der „matching“ Körper misst 256x240x64.

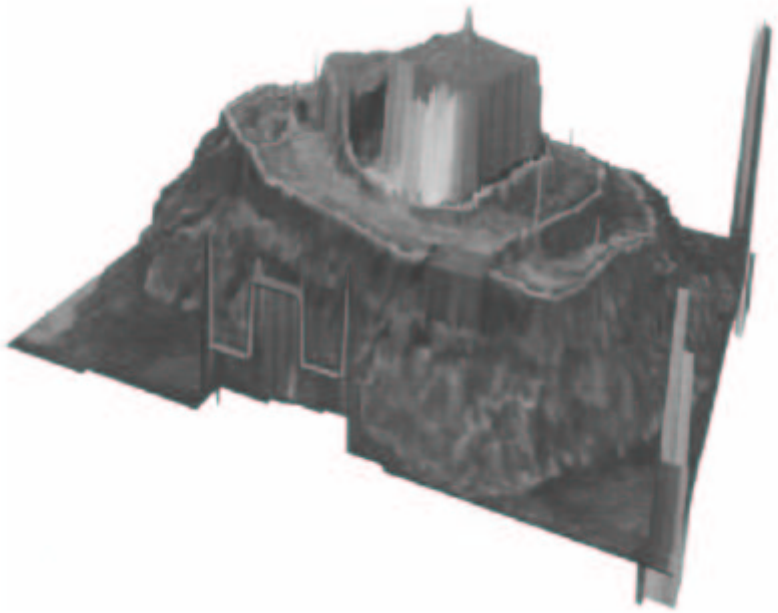

Abb.25: Rekonstruierte 3D-Oberfläche basierend auf der „roof“ Sequenz und berechnet mit „maximum flow“
.

„Castle“
Die „castle“ Sequenz vom CMU wird in Abb.26 präsentiert. Sie besteht aus elf Bildern mit einer Kombination aus horizontalen, vertikalen und Vorwärtsbewegungen. Die elf Bilder wurden verwendet, um das Tiefenbild (rechts zu sehen) aus dem Ursprungsbild (links) zu berechnen. Das Bild ist sehr detailliert, es sind jedoch ein paar Fehler vorhanden.

Es ist dabei wichtig zu sagen, dass diese Sequenz eine wirkliche Herausforderung ist, da die Tiefenamplitude, die Differenz zwischen dem nahesten und dem entferntesten Objekt, nur 2,7 Pixel beträgt. Da 96 Tiefenstufen verwendet wurden, bedeutet das, das die Präzision in die Tiefe mit 0,03 Pixel sehr hoch ist.

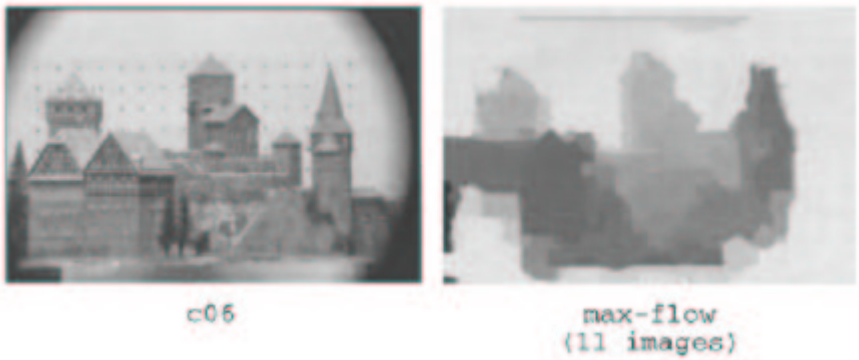

Abb.26: Das „castle“ Beispiel. Links ist eins von elf Bildern. Rechts ist das sich ergebende Tiefenbild, berechnet mit dem „maximum flow“ Algorithmus.

5.1 Grad der „Weichheit“

In diesem Abschnitt werden die Auswirkungen des „Weichheitsgrades“, repräsentiert durch den Parameter *K* aus Abschnitt 4.1, auf die Qualität der Tiefenbilder gezeigt. Abb.27 führt dieses für vier Werte von *K* (0,1,10,100) vor. Bei *K*=100 ist die Kapazität der „smoothness“

Kanten null, von daher wird jedem Pixel unabhängig voneinander eine Tiefe zugeordnet. Dieses Verfahren nennt man dann „direct search“ mit einem einzelnen Pixel als Fenster.

Wie erwartet, zeichnet ein geringer Wert für *K* größere Tiefenunterschiede und demnach auch schärfere Objektkanten auf Kosten weicherer Übergänge zwischen den unterschiedlichen Tiefen. Es kann beobachtet werden, dass große Tiefensprünge auch bei einer Vergrößerung von *K* gut erkennbar bleiben. Dies ist so, da die weichen Übergänge nicht nur entlang der Epipolarlinien, sondern in alle Richtungen berechnet werden. Dieses Ergebnis unterscheidet sich stark von vielen anderen Methoden, bei denen ein hoher „smoothness“ Wert dazu führt, dass viele Tiefeninformationen verloren gehen und vieles verschwommen ist.

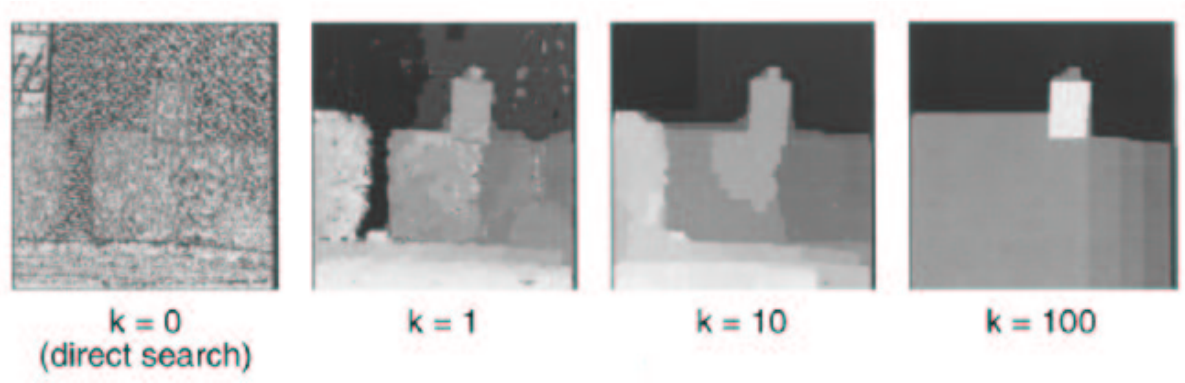

Abb.27: Tiefenbilder basierend auf der „shrub“ Sequenz für vier Werte von K berechnet. K=0: Keine weichen Übergänge, K=1, K=10, K=100, es wird immer mehr „smoothness“ hineingerechnet, was zu einer Degradation der Tiefenkarte führt.

6. Zusammenfassung

In Roy (1999) wurde ein neuer Algorithmus zur Berechnung des „stereo correspondence“ Problems präsentiert, der darauf basiert, den maximalen Fluss in einem Graphen zu finden. Er kann die optimale Tiefenkarte in einem Schritt berechnen und dabei die sonst üblichen Inkonsistenzen über Epipolarlinien hinweg vermeiden. Die „ordering“ Bedingung, die noch für das dynamische Programmieren benötigt wurde, wird durch die allgemeinere „local coherence“ Bedingung ersetzt, die über alle Linien hinweg und nicht nur entlang einer Linie gilt. Es werden mehr als zwei Kameras mit willkürlich gewählten Positionen unterstützt und die Berechnung kann aus der Sicht einer virtuellen Kamera erfolgen. Die Tiefensprünge werden für jeden gewünschten „smoothness“ Grad gut erkannt und es werden sogar ungewollte Sprünge durch die „local coherence“ Bedingung eliminiert. Zur Berechnung der Kosten reicht für gute Ergebnisse schon eine einfache Kostenfunktion, die im Gegensatz zu den meisten, sehr komplexen Funktionen, global effizient minimiert werden kann.

Es gibt noch viele Verbesserungsmöglichkeiten dieses Ansatzes. Es könnten mehrere Auflösungen oder lokale „smoothness“ Variationen innerhalb des Graphen eingebettet sein, um die Qualität des Tiefenbildes zu verbessern. Daneben könnte zudem noch die Geschwindigkeit des „maximum flow“ Algorithmus gesteigert werden.

7. Literatur

Baker, H.H. 1981. Depth from Edge and Intensity Based Stereo. Ph.D. Thesis. University of Illinois at Urbana-Champaign

Belhumeur, P.N. 1996. A Bayesian approach to binocular stereopsis. Int. J. Computer Visions, 19(3):237-260.

Boykov, Y., Veksler, O., and Zabih, R.1998. Markov random fields with efficient approximations. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition, Santa Barbara, CA.

Cormen, T.H., Leiserson, C.E. and Rives, R.L. 1990. Introduction to Algorithms. McGraw-Hill: New York.

Cox, I.J. 1994. A maximum likelihood N-camera stereo algorithm. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition, 733-739.

Cox, I.J., Hingorani, S., Maggs, B.M. and RAO, S.B. 1996. A maximum likelihood stereo algorithm. Computer Visions and Image Understanding, 63(3):542-567.

Faugeras, O. 1993. Three-Dimensional Computer Vision. MIT Press: Cambridge

Goldberg, A.V. and Rao, S.B. 1997. Length functions for flow computations. Technical Report 97-055, NEC Research Institute, Princeton, NJ:

Greig; D.M., Porteous, B.T., and Seheult, A.H. 1989. Exact maximum a posteriori estimation for binary images. J.R. Statist. Soc., 51(2):271-279.

Ishikawa, H. and Geiger, D. 1998. Segmentation by grouping junctions. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition, Santa Barbara, CA.

Kanade, T., Yoshida, A., Oda, K., Kano, H., and Tanaka, M. 1996. A Stereo machine for video-rate dense mapping and its new applications. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition, San Francisco.

Kang, S.B., Webb, J.A., Zitnick, C.L., and Kanade, T. 1994. An active multibaseline stereo system with real-time image acquisition. Technical Report CMU-CS-94-167, School of Computer Science, Carnegie Mellon University.

Marr, D. and Poggio, T. 1979. A theory of human stereopsis. Proceedings of the Royal Society, B 204:301-328.

Ohta, Y. and Kanade, T. 1985. Stereo by intra- and inter-scanline using dynamic programming. IEEE Trans. Pattern Analysis and Machine Intelligence, 7(2):139-154.

Roy, S. and Cox, I.J. 1998. A maximum-flow formulation of the n-camera stereo correspondence problem. In Proc. Int. Conference on Computer Vision, Bombay, India, 492-499

Yang, Y. and Yuille, A.L. 1995. Multilevel enhancement and detection of stereo disparity surfaces. Artificial Intelligence, 78:121-145.