in: J. Hektor und R. Grebe (Hrsg.): Parallele Datenverarbeitung mit dem Transputer
TAT'93, Aachen, September 1993, Reihe Informatik aktuell,
Springer-Verlag, Berlin, 1994, S. 238-249.

Parallelverarbeitende dreidimensionale Oberflächenrekonstruktion

V. David Sánchez A.
Deutsche Forschungsanstalt für Luft- und Raumfahrt
Institut für Robotik und Systemdynamik
D-82230 Weßling

Zusammenfassung

Das Problem der dreidimensionalen Oberflächenrekonstruktion wird behandelt. Zunächst wird das Problem formuliert und ein Lösungsansatz mittels eines Variationsproblems wird angegeben. Anschließend werden iterative Lösungsverfahren linearer Gleichungssysteme auf Parallelrechnern beschrieben, da sie zur numerischen Lösung erforderlich sind. Die Ergebnisse eines Fallbeispieles werden dann vorgestellt. Dazu wird ein äquivalentes Energiemodell der Aufgabenstellung aufgebaut, das dazugehörige kontinuierliche Variationsproblem wird aufgestellt und anschließend wird es mit Hilfe der Finite-Elemente-Methode diskretisiert. Eine Lösung des enstandenen linearen Gleichungssystems wird mit Hilfe einer Parallelimplementierung des relaxierten Gauß-Seidel-Verfahrens (SOR) bestimmt.

1 Einleitung

Im allgemeinen gibt es für dreidimensionale Körper zwei Repräsentationsformen: Oberflächen- und Volumen-Repräsentation. Beim Problem der Oberflächenrekonstruktion wird die erste Repräsentationsform zugrundegelegt. Die Oberflächenrepräsentation kann explizit, implizit oder parametrisch sein [4]. Die explizite Repräsentation ist der Graph einer Funkton zweier Veränderlichen. Die implizite Repräsentation wird mit Hilfe folgender Funktion angegeben: $f : R^3 \rightarrow R$, $f(x,y,z) =$ Konstante. (x,y,z) sind die Koordinaten der Punkte P auf der Oberfläche. Die parametrische Repräsentation wird folgendermaßen angegeben: $P = (x(s,t), y(s,t), z(s,t))$, s,t sind die Parameter. In diesem Beitrag werden lediglich Methoden zur Rekonstruktion der expliziten Form einer Oberfläche behandelt.

Allgemeine Grundlagen über Oberflächenrekonstruktionsverfahren können z.B. in [3,20,4,24] nachgelesen werden. Parallelalgorithmen für das SOR-Verfahren werden in [13,14] diskutiert. In [19] wurden zur Oberflächenrekonstruktion hierarchische Basisfunktionen und die Methode der konjugierten Gradienten eingesetzt. Über eine Parallelimplementation wurde in [5] berichtet.

Zwei Einsatzgebiete dieser Rekonstruktionsverfahren in der Robotik sind die Objekterkennung im Rahmen von Manipulationsaufgaben und die Geländemodellierung im Rahmen von Navigationsaufgaben eines Autonomen Mobilen Robotersystems. Im Abschnitt 2 wird das Problem formuliert und ein Lösungsansatz mittels eines Variationsproblems wird angegeben. Abschnitt 3 beschreibt iterative Lösungsverfahren linearer Gleichungssysteme auf Parallelrechnern und Abschnitt 4 stellt die Ergebnisse eines Fallbeispieles vor.

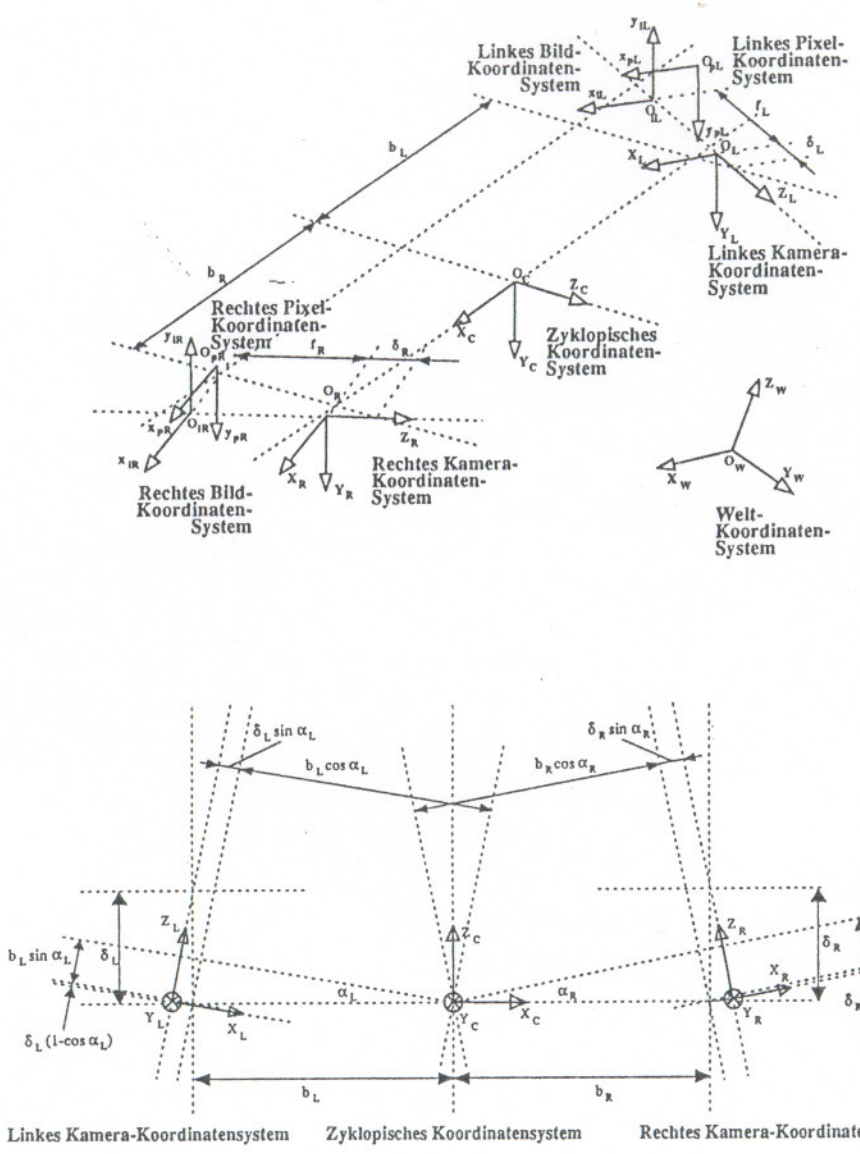

Abbildung 1: Stereokameramodell

2 Dreidimensionale Oberflächenrekonstruktion

2.1 Problemformulierung

Gegeben ist eine Menge $\{P_i = (x_i, y_i, z_i), i = 1, \cdots n\} \subset R^3$ spärlich verteilter Meßdaten, $(x_i, y_i) \in \Omega \subset R^2$, Ω ist eine Referenzfläche. Solch eine Menge liegt z.B. vor, wenn Stereobildverarbeitungsverfahren, wie die in [16] beschriebene Verfahren, auf einem Stereobildpaar angewendet worden sind und mit Hilfe der Parameter eines kalibrierten Stereokamerasystems[1] aus den Pixelwerten einzelner zugeordneten Punkte P_i, $(x_{iL}, y_{iL}), (x_{iR}, y_{iR})$ jeweils für linke und rechte Kamera, die $X-$, $Y-$ und $Z-$Koordinaten der Punkte P_i zurückgewonnen worden sind. Das zyklopische Koordinatensystem (vgl. Abbildung 1) kann als Referenzkoordinatensystem in diesem Fall angenommen werden.

Gesucht wird eine Funktion $z = v(x, y)$, $\forall (x, y) \in \Omega$, die die vorgegebenen Entfernungsdaten möglichst optimal und kontinuierlich approximiert unter Berücksichtigung der vorgegebenen Tiefen- und Orientierungsunstetigkeiten. In der numerischen Lösung liegen meistens lediglich diskrete Werte für $(x_i, y_i) \in \Omega$ vor. Deshalb läßt sich auch sagen, daß die Menge $\{P_i = (x_i, y_i, z_i), \forall (x_i, y_i) \in \Omega, z_i = v(x_i, y_i)\}$ dicht verteilter Entfernungswerte gebildet wird. Es handelt sich also um die Approximation einer Funktion zweier Veränderlichen, die bekanntlich ein Inversproblem darstellt. Abbildung 2 zeigt ein Beispiel einer idealen Rekonstruktion.

2.2 Lösungsansatz als Variationsproblem

Die Lösung des oben gestellten Approximationsproblems kann folgendermaßen angesetzt werden:

$$\text{Suche } u : \mathcal{E}(u) = \inf_{v \in \mathcal{H}} \mathcal{E}(v) \tag{1}$$

$$\mathcal{E}(v) = \mathcal{S}(v) + \mathcal{P}(v) \tag{2}$$

$\mathcal{H}$ ist der Raum der möglichen Lösungen. $\mathcal{S}$ ist ein Regularisierungs-Funktional. $\mathcal{P}$ ist ein Straf-Funktional, das die Abweichung zwischen Meßdaten und der Funktion $v(x, y)$ mißt.

Um das ursprünglich gestellte Inversproblem zu regularisieren, werden für Rekonstruktionsprobleme stellvertretend für eine Problemklasse im Bereich des Rechnersehens Funktionale eingeführt, die die Stetigkeitsanforderungen in das zu minimierende Funktional $\mathcal{E}(v)$ integrieren. Für Funktionen $v : R^2 \to R$, Elemente des Sobolevraumes $\mathcal{H}^m(R^2)$, wurden auf dem Gebiet der multivariaten Spline-Approximation folgende Funktionale[2] vorgeschlagen:

$$|v|_m^2 = \int\int_{R^2} \sum_{i=0}^{m} \binom{m}{i} \left(\frac{\partial^m v}{\partial x^i \partial y^{m-i}}\right)^2 dx dy \tag{3}$$

Um Unstetigkeiten miterfassen zu können wurden Funktionale mit kontrollierter Stetigkeit eingeführt [20,12]. Im Abschnitt 4 werden explizit für einen Fallbeispiel ausführliche Ausdrücke angegeben. Ebenfalls wird dort für die numerische Behandlung die Überführung des ursprünglichen Lösungsansatzes auf die Lösung eines linearen Gleichungssystems beschrieben. Zunächst wird deshalb eine Auswahl iterativer Methoden zur Lösung linearer Gleichungssysteme auf Parallelrechnern zusammenfaßend dargestellt.

[1] Das kann mit Hilfe eines dazugehörigen Stereokamerasystems und herkömmlicher Kalibrierverfahren erzielt werden.

[2] Solche Funktionale stellen das Quadrat von Seminormen dar. Die Angabe hier erfolgt für den zweidimensionalen Fall.

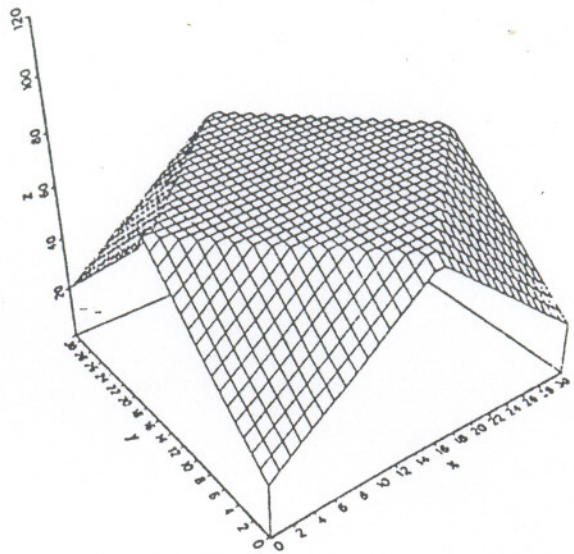

Abbildung 2: Ideale Oberflächenrekonstruktion

3 Iterative Lösung linearer Gleichungssysteme auf Parallelrechnern

Für $A \in R^{m \times n}, \vec{x} \in R^n, \vec{b} \in R^m$, wird das folgende lineare Gleichungssystem definiert:

$$A \cdot \vec{x} = \vec{b} \tag{4}$$

Zwei Klassen von Lösungsverfahren für lineare Gleichungssysteme sind:

- Direkte Verfahren: liefern abgesehen von Rundungsfehlern die exakte Lösung. Dazu gehören u.a. der Gaußsche Algorithmus, das Gauß-Jordan-Verfahren, das Cholesky-Verfahren, Verfahren für Systeme mit Bandmatrizen und die Methode des Pivotisierens. Bei schlecht konditionierten Systemen kann die mit einem direkten Verfahren ermittelte Näherungslösung iterativ verbessert werden.
- Iterative Verfahren: schrittweise verbessern eine vorgegebene Lösungsnäherung (Startvektor). Dazu gehören u.a. das Gesamtschrittverfahren, das Einzelschrittverfahren, die Relaxationsverfahren und die Methode der konjugierten Gradienten.

Weitere Klassen von Lösungsverfahren für lineare Gleichungssysteme sind die Mehrgitterverfahren und die Gebietszerlegungsmethoden. Nachfolgend werden die iterativen Lösungsverfahren zusammengefaßt und ihre Parallelisierungsmöglichkeiten werden besprochen.

3.1 Iterative Lösungsverfahren

3.1.1 Gesamtschritt-, Einzelschritt- und Relaxationsverfahren

Das lineare Gleichungssystem $A \cdot \vec{x} = \vec{b}, A \in R^{n \times n}$, regulär, $\vec{x}, \vec{b} \in R^n$ läßt sich in seine Fixpunktform überführen: $\vec{x} = T \cdot \vec{x} + \vec{c}$. Auf dieser Basis und bei vorgegebener Ausgangsnäherung $\vec{x}^0$ läßt sich der Iterationsschritt dieser Verfahren folgendermaßen ausdrücken:

Table 1: Iterationsmatrix und Konstantenvektor einzelner iterativer Verfahren

Verfahren →	JOR (Jacobi für $\omega = 1$)	SOR (Gauß-Seidel für $\omega = 1$)
Iterationsmatrix T	$(1-\omega)\cdot I + \omega \cdot D^{-1}\cdot(L+U)$	$(D-\omega\cdot L)^{-1}[(1-\omega)\cdot D + \omega\cdot U]$
Konstantenvektor $\vec{c}$	$\omega \cdot D^{-1}\cdot \vec{b}$	$\omega\cdot(D-\omega\cdot L)^{-1}\cdot\vec{b}$

$$\vec{x}_{k+1} = T\cdot\vec{x}_k + \vec{c}, \quad k \in N^0 \tag{5}$$

Tabelle 1 faßt die Ausdrücke der Iterationsmatrix $T \in R^{n\times n}$ und des Konstantenvektors $\vec{c} \in R^n$ einzelner iterativer Verfahren zusammen. Dabei wurde folgende Zerlegung zugrundegelegt: $A = D - L - U$, $D, L, U \in R^{n\times n}, D = \mathrm{Diag}(a_{ii})$, regulär, L ist eine strikt untere Dreieckmatrix, U ist eine strikt obere Dreieckmatrix. Im einzelnen handelt es sich um folgende Verfahren:

1. Das JOR-Verfahren (engl.:*Jacobi-Over-Relaxation*).
2. Für $\omega = 1$ erhält man beim JOR-Verfahren das Jacobi-Verfahren (auch J-Verfahren oder Gesamtschrittverfahren genannt).
3. Das SOR-Verfahren (engl.:*Successive-Over-Relaxation*).
4. Für $\omega = 1$ erhält man beim SOR-Verfahren das Gauß-Seidel-Verfahren (auch Einzelschrittverfahren genannt).

Bei den relaxierten Versionen der Verfahren (JOR, SOR), auch Relaxationsverfahren genannt, stellt ω der Relaxationsparameter dar, der zur Konvergenzbeschleunigung eingeführt wird. Ein Maß für die Konvergenzgeschwindigkeit (auch Konvergenzrate genannt) des Iterationsverfahrens nach Gl.(5) ist der Spektralradius der Iterationsmatrix: $\rho(T) = \max\{|\lambda|/\lambda$ ist Eigenwert von $T\}$.

Die Konvergenz des Iterationsverfahrens nach Gl.(5) (d.h. der erzeugten Folge $\{\vec{x}_k\}$) gegen einen eindeutigen Vektor $\vec{x}^*$, d.h. $\lim_{k\to\infty}\vec{x}_k = \vec{x}^*$, $\vec{x}^* = T\cdot\vec{x}^* + \vec{c}$, ist äquivalent zu: $\rho(T) < 1$. Ein Iterationsverfahren konvergiert umso schneller, je kleiner sein Spektralradius $\rho(T)$ ausfällt. Weitere Maße für die Konvergenzgeschwindigkeit eines Iterationsverfahrens sind die mittlere Konvergenzgeschwindigkeit: $R_n(T) = -\frac{1}{n}\cdot\log_{10}||T^n||$ und die asymptotische Konvergenzgeschwindigkeit $R(T) = \lim_{n\to\infty} R_n(T) = -\log_{10}\rho(T)$.

3.1.2 Die Methode der konjugierten Gradienten

Für $A \in R^{n\times n}$, symmetrisch (d.h. $A = A^T$), positiv definit (d.h. $\vec{x}^T\cdot A\cdot\vec{x} > 0, \forall\vec{x}\in R^n$), ist die Lösung des linearen Gleichungssystems $A\cdot\vec{x} = \vec{b}$ das Minimum der quadratischen Funktion $F(\vec{x}) = \frac{1}{2}\cdot\vec{x}^T\cdot A\cdot\vec{x} - \vec{x}^T\cdot\vec{b}$. Diese Aussage läßt sich mit Hilfe der Ausdrücke des Gradientes: $G(\vec{x}) = \nabla F = A\cdot\vec{x} - \vec{b} = 0$ und der Hessematrix $H(\vec{x}) = A$, positiv definit, beweisen.

Zwei Vektoren $\vec{0} \neq \vec{p}_i, \vec{p}_j \in R^n$ heißen konjugiert oder A-orthogonal für $A \in R^n$, positiv definit, wenn $(\vec{p}_i)^T\cdot A\ \vec{p}_j = 0$. Bei vorgegebenem Startvektor $\vec{x}_0$ werden $\vec{p}_0 = \vec{r}_0 = \vec{b} - A\cdot\vec{x}_0$ initialisiert. Eine verbreitete Version der Methode der konjugierten Gradienten basiert auf folgender Iteration ($k \in N^0$):

$$\alpha_k = \frac{(\vec{r}_k)^T\cdot\vec{r}_k}{(\vec{p}_k)^T\cdot A\cdot\vec{p}_k} \tag{6}$$

$$\vec{x}_{k+1} = \vec{x}_k + \alpha_k\cdot\vec{p}_k \tag{7}$$

$$\vec{r}_{k+1} = \vec{r}_k - \alpha_k\cdot A\cdot\vec{p}_k \tag{8}$$

$$\vec{p}_{k+1} = \vec{r}_{k+1} + \frac{(\vec{r}_{k+1})^T\cdot\vec{r}_{k+1}}{(\vec{r}_k)^T\cdot\vec{r}_k}\cdot\vec{p}_k \tag{9}$$

Theoretisch liefert die Methode der konjugierten Gradienten die Lösung eines linearen Gleichungssystems mit $\vec{x} \in R^n$ in höchstens n Schritten. Die Richtungsvektoren $\vec{p}_k$ sind paarweise konjugiert, die Residuenvektoren $\vec{r}_k$ bilden ein Orthogonalsystem und $\vec{x}_{k+1}$ minimiert $F(\vec{x}_k + \alpha_k\cdot\vec{p}_k)$ lokal bzgl. α_k ausgehend von $\vec{x}_k$ in Richtung $\vec{p}_k$.

Eine Verbesserung der Konvergenzeigenschaften der Methode der konjugierten Gradienten wird durch Vorkonditionierung, d.h. durch Reduktion der Konditionszahl $\kappa(A)$, erreicht, indem $A\cdot\vec{x} = \vec{b}$ in folgende äquivalente Form überführt wird:

$$A'\cdot\vec{x}' = \vec{b}' \tag{10}$$

$$A' = C^{-1}\cdot A\cdot(C^{-1})^T,\ \vec{x}' = C^T\cdot\vec{x},\ \vec{b}' = C^{-1}\cdot\vec{b} \tag{11}$$

$$\kappa(A') < \kappa(A), \quad \text{bei geeigneter Wahl von } C \tag{12}$$

Die Vorkonditionierungsmatrix $M = C\cdot C^T$ ist auch symmetrisch, positiv definit. Mit Hilfe folgender Norm: $||E(\vec{x})||_A = (\vec{x} - \vec{x}^*)^T\cdot A\cdot(\vec{x} - \vec{x}^*)$, mit $\vec{x}^*$ Lösung von $A\cdot\vec{x} = \vec{b}$ wird eine Konvergenzabschätzung der Methode der konjugierten Gradienten durch folgende Ausdrücke angegeben:

$$\frac{||E(\vec{x}_k)||_A}{||E(\vec{x}_0)||_A} \leq 2\cdot\left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^k \tag{13}$$

$$\frac{||E(\vec{x}_k)||_A}{||E(\vec{x}_0)||_A} \leq \varepsilon \ \Rightarrow\ k \leq \frac{1}{2}\cdot\sqrt{\kappa(A)}\cdot\ln(\frac{2}{\varepsilon}) + 1 \tag{14}$$

Die Methode der konjugierten Gradienten gehört zu den Krylow-Unterraum-Methoden. Eine Darstellung über den aktuellen Stand kann [7] entnommen werden. Der Einfluß von Rundungsfehlern auf die Konvergenzeigenschaften der Methode der konjugierten Gradienten wird z.B. in [15] behandelt.

Für eine weiterführende Darstellung der Methoden können z.B. [22,23,21,9,18] nachgeschlagen werden. Einige numerische Algorithmen auf Transputersystemen sind in [1] enthalten.

3.2 Parallelisierung

3.2.1 Klassische Iterationsverfahren

Zu den klassischen iterativen Methoden gehören folgende Verfahren: das Jacobi Verfahren, das Gauß-Seidel-Verfahren, das relaxierte Jacobi Verfahren (JOR), das relaxierte Gauß-Seidel-Verfahren (SOR) und die Richardson Methode. Alle können auf Parallelrechnern implementiert werden, wobei Gauß-Seidel-Verfahren i.a. als nicht ideal geeignet für Parallelisierung gelten, insbesondere für vollbesetzte Matrix A. Die nachfolgenden Betrachtungen konzentrieren sich auf den Fall der vollbesetzten Matrizen A bei JOR- und SOR-Verfahren. Diese Verfahren werden folgendermaßen dargestellt:

$$D_\omega\cdot\vec{x}_{k+1} = B_\omega\cdot\vec{x}_k + \vec{b} \tag{15}$$

Das JOR-Verfahren läßt sich aus Tabelle 1 nach Umformung in der Form der Gl.(15) ausdrücken mit $B_\omega = (L+U) + \frac{1-\omega}{\omega}\cdot D$ ($B_\omega = (b_{ij})_{1\leq i,j\leq n}, i \neq j : b_{ij} = -a_{ij}, b_{ii} = \frac{1-\omega}{\omega}\cdot a_{ii}$) und $D_\omega = \frac{D}{\omega}$. Die auszuführenden Operationen sind deshalb: Matrix-Vektor-Multiplikation $B_\omega\cdot\vec{x}_k$, Vektor-Addition $B_\omega\cdot\vec{x}_k + \vec{b}$ und Vektor-Multiplikation $\alpha_i\cdot(B_\omega\cdot\vec{x}_k + \vec{b})_i$, mit $\alpha_i = \frac{\omega}{a_{ii}}, i = 1,\cdots,n$ ($A = (a_{ij})_{1\leq i,j\leq n}$). Die Parallelisierung der zwei letzten Operationen ist offensichtlich. Parallelisierungsmethoden für Matrix-Vektor-Multiplikation können z.B. [2,8] entnommen werden.

Für das Jacobi-Verfahren (für $\omega = 1$) sind die Diagonalelemente von B_ω Nullen, was zu einer gesonderten Ausführung der Matrix-Vektor-Multiplikation (Aufspaltung) führt. Nach jeder Iteration k wird jedem Prozessor die notwendigen Komponenten der Iterationsergebnisse mitgeteilt.

Das SOR-Verfahren läßt sich aus Tabelle 1 nach Umformung in der Form der Gl.(15) ausdrücken mit $B_\omega = U + \frac{1-\omega}{\omega} \cdot D$ und $D_\omega = \frac{D}{\omega} - L$. Nachfolgend wird die Parallelisierung der $ji-$Form des SOR-Algorithmus besprochen, die effizienter als die $ij-$Form sein dürfte und die zeilenweise Abspeicherung der Matrix B_ω auf den Prozessoren voraussetzt [8]. Folgende Zwischenwerte $t_i^{j,k}$ für Iterationsschritt k und Zeile $0 \le j \le n, 1 \le i \le n$ werden definiert[3]:

$$t_i^{j,k} := \begin{cases} b_i + \sum_{l=i}^{j} b_{il} \cdot x_l^{k+1} & 1 \le i \le j \\ b_i + \sum_{l=i}^{n} b_{il} \cdot x_l^{k} + \sum_{l=1}^{j} b_{il} \cdot x_l^{k+1} & j+1 \le i \le n \end{cases} \quad (16)$$

Die Bestimmung der $t_i^{j,k}$ schließt Teilberechnungen zweier aufeinanderfolgenden Iterationen ein. Für die Bestimmung von x_i^{k+2} und x_i^{k+1} werden jeweils die Werte von $t_i^{j,k}$ für $1 \le i \le j$ und $t_i^{j,k}$ für $j+1 \le i \le n$ verwendet. Die Parallelisierung beruht darauf, daß jeder Prozessor für seine zugeordneten Zeilen j folgende Operationen durchführt: $x_j^{k+1} := \alpha_j \cdot t_j^{j-1,k}$, anschließend wird x_j den anderen Prozessoren mitgeteilt, $t_j^{j,k} := b_j$ und für seine zugeordneten Zeilen i werden die $t_i^{j,k}$ folgendermaßen bestimmt:

$$t_i^{j,k} := \begin{cases} t_i^{j-1,k} + b_{ij} \cdot x_j^{k+1}, & i \ne j \\ b_i + b_{ij} \cdot x_j^{k+1}, & i = j \end{cases} \quad (17)$$

3.2.2 Methode der konjugierten Gradienten mit Vorkonditionierung

Vier zeitraubende Operationen sind: der Skalarprodukt, die Vektoraktualisierung, die Matrix-Vektor-Multiplikation und die Vorkonditionierung. Für die ersten drei Operationen ist die Parallelisierung entweder offensichtlich oder bekannt. Gängige Vorkonditionierungsvarianten können nicht direkt parallelisiert werden. In bestimmten Fällen kann das Umordnen der Teiloperationen bzw. der Unbekannten helfen. Die Parallelisierung der Vorkonditionierung für die Methode der konjugierten Gradienten hat zur Entwicklung neuer Vorkonditionierungsvarianten geführt. Oft muß man sich zwischen erhöhter Parallelität und numerischer Stabilität entscheiden.

Bei vorgegebenem Startvektor $\vec{x}_0$ lautet die Initialisierung: $\vec{r}_0 = \vec{b} - A \cdot \vec{x}_0, \vec{p}_{-1} = \vec{0}, \beta_{-1} = 0, \vec{\omega}_0$ Lösung von:[4] $K \cdot \vec{\omega}_0 = \vec{r}_0, \rho_0 = \vec{r}_0^T \cdot \vec{\omega}_0$ für eine Variante der Methode der konjugierten Gradienten mit Vorkonditionierung (vgl. Abschnitt 3.1.2 und Algorithmus 24 in [6]), die auf folgender Iteration ($k \in N^0$) basiert:

$$\vec{p}_k = \vec{\omega}_k + \beta_{k-1} \cdot \vec{p}_{k-1} \quad (18)$$

$$\vec{q}_k = A \cdot \vec{p}_k, \quad \alpha_k = \frac{\rho_k}{\vec{p}_k^T \cdot \vec{q}_k} \quad (19)$$

$$\vec{x}_{k+1} = \vec{x}_k + \alpha_k \cdot \vec{p}_k, \quad \vec{r}_{k+1} = \vec{r}_k - \alpha_k \cdot \vec{q}_k \quad (20)$$

$$\vec{\omega}_{k+1} \quad \text{Lösung von:} \quad K \cdot \vec{\omega}_{k+1} = \vec{r}_{k+1} \quad (21)$$

$$\rho_{k+1} = \vec{r}_{k+1}^T \cdot \vec{\omega}_{k+1}, \quad \beta_k = \frac{\rho_{k+1}}{\rho_k} \quad (22)$$

Die effektivste Parallelisierung erfolgt bei der Berechnung von $\vec{q}_k$ und der Bestimmung von $\vec{\omega}_{k+1}$. Der Algorithmus hat ein Verhältnis Gleitpunktoperationen/Speicherzugriffe von 10/7 und zwei Synchronisierungspunkte an beiden Skalarprodukt-Berechnungen. Verbesserungen des Algorithmus befaßen sich u.a. mit der Erhöhung des o.g. Verhältnises bzw. der Reduktion der Synchronisierungspunkte. Die Parallelisierung weiterer Varianten der Methode der konjugierten Gradienten und der Krylow-Unterraum-Methoden kann man [6] entnehmen.

[3] b_i sind Elemente des Vektors $\vec{b}$, nicht zu verwechseln mit b_{ij} Elementen der Matrix B_ω.

[4] K ist eine Approximation von A, sodaß $K \cdot \vec{y} = \vec{d}$ einfacher zu lösen ist als: $A \cdot \vec{x} = \vec{b}$.

4 Ergebnisse eines Fallbeispieles

4.1 Variationsproblem

Das Energie-Funktional $\mathcal{E}(v) = \mathcal{S}(v) + \mathcal{P}(v)$ wird zugrundegelegt. Das Stabilisierungs-Funktional mit kontrollierter Stetigkeit $\mathcal{S}_{\rho\tau}$ und das Straf-Funktional $\mathcal{P}$ werden folgendermaßen definiert:

$$\mathcal{S}_{\rho\tau}(v) = \frac{1}{2} \int_\Omega \int \rho(x,y)\{\tau(x,y)(v_{xx}^2 + 2v_{xy}^2 + v_{yy}^2) + [1-\tau(x,y)](v_x^2 + v_y^2)\} dx dy \quad (23)$$

$$\mathcal{P}(v) = \frac{1}{2}\{\sum_{i \in D} \alpha_{di}[v(x_i,y_i) - d_{(x_i,y_i)})]^2 + \sum_{i \in P} \alpha_{pi}[v_x(x_i,y_i) - p_{(x_i,y_i)})]^2 + \sum_{i \in Q} \alpha_{qi}[v_y(x_i,y_i) - q_{(x_i,y_i)})]^2\} \quad (24)$$

$\Omega \subset R^2$ ist die Referenzfläche, $v(x,y)$ ist die Approximationsfunktion, $\rho(x,y), \tau(x,y)$ sind die Stetigkeitskontrollfunktionen, $\rho(x,y), \tau(x,y) \in [0,1]$, die explizit Tiefen- und Orientierungs-Unstetigkeiten repräsentieren, d sind die lokalen Tiefenwerte, p, q sind die Orientierungswerte (Komponenten der Oberflächennormale). Es gilt, das Funktional $\mathcal{E}(v)$ zu minimieren.

4.2 FEM-Diskretisierung

Das kontinuierliche Variationsproblem wird zuerst in ein diskretes Variationsproblem mit Hife der Finite-Elemente-Methode (FEM) überführt. Die Referenzfläche Ω wird mit Hilfe eines regulären, kartesischen Gitters mit Knotenabstand h (Seitenlänge) unterteilt. Folgende Finite Differenzen werden eingesetzt:

$$v_{xx}^h = \frac{1}{h^2}(v_{i+1,j}^h - 2v_{i,j}^h + v_{i-1,j}^h), \; v_{yy}^h = \frac{1}{h^2}(v_{i,j+1}^h - 2v_{i,j}^h + v_{i,j-1}^h),$$
$$v_{xy}^h = \frac{1}{h^2}(v_{i+1,j+1}^h - v_{i,j+1}^h - v_{i+1,j}^h + v_{i,j}^h),$$
$$v_x^h = \frac{1}{h}(v_{i+1,j}^h - v_{i,j}^h), \; v_y^h = \frac{1}{h}(v_{i,j+1}^h - v_{i,j}^h) \quad (25)$$

4.3 Parallelimplementation

Eine Lösung folgenden Problems wird gesucht: Finde $u \in \mathcal{H}$, sodaß $\mathcal{E}(u) = \inf_{v \in \mathcal{H}} \mathcal{E}(v)$, $\mathcal{H}$ ist ein linearer Raum der möglichen Lösungen. Eine notwendige Bedingung für das Minimum im diskreten Fall und das das daraus entstandene lineare Gleichungssystem lauten:

$$\nabla \mathcal{E}_{\rho\tau}^h(u^h) = \nabla \mathcal{S}_{\rho\tau}^h(u^h) + \nabla \mathcal{P}^h(u^h) = 0 \quad (26)$$

$$A \cdot \vec{u} = \vec{b} \quad (27)$$

In A sind die Koeffizienten der $u_{i,j}^h$ enthalten und in $\vec{b}$ sind $\alpha_{d_{i,j}}^h d_{i,j}^h, \frac{\alpha_{p_{i,j}}^h}{2h} p_{i,j}^h, \frac{\alpha_{q_{i,j}}^h}{2h} q_{i,j}^h$ enthalten. Die Lösung wird numerisch bestimmt. Die Berechnung erfolgt lokal für jeden Gitterknoten; insgesamt entsteht ein Netzwerk rechnender Moleküle. Ein Beispiel für ein inneres Knotenmolekül, wo keine Orientierungsunstetigkeit vorhanden ist und es keine Tiefenunstetigkeiten an angrenzenden Knoten gibt, wird in Abbildung 3 gezeigt.

Das relaxierte Gauß-Seidel-Verfahren (SOR) wurde als Gleichungslöser implementiert. Ebenfalls wurden verschiedene Initialisierungsroutinen sowie eine erste Version eines Mehrgitterverfahrens implementiert. Folgendes Iterationsschema wurde herangezogen:

$$u_{i,j}^{h(k)} = u_{i,j}^{h(k-1)} - \omega \frac{\xi_{i,j}}{a_1}, \; \omega_{opt} = \frac{2}{1 + \sqrt{1-\rho^2}}, \; \rho_{max} = \cos\frac{\Pi}{N} \quad (28)$$

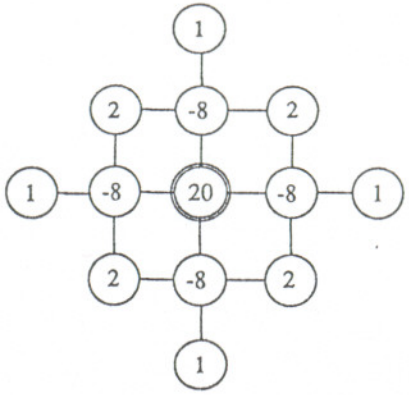

Abbildung 3: Beispiel für inneres Knotenmolekül

Abbildung 4: Prozessortopologie

mit k der Iterationsschritt, ω die Schrittweite, $a_{1_{i,j}}$ ist der Koeffizient des ersten Molekülatoms, $\xi_{i,j}$ wird mit der Residuengleichung des k-ten Iterationssschrittes bestimmt, ρ ist der Spektralradius (zur Konvergenzbetrachtungen einzusetzen), N die Anzahl der Gitterknoten.

Zu Testzwecken wurde ein Transputersystem für Rechnersehen [17] verwendet, bei dem für das Fallbeispiel als Topologie die lineare Verkettung von bis zu acht Transputern (vgl. Abbildung 4) gewählt wurde. Abbildung 5 zeigt die Knotenpartitionierung an der Grenze zwischen zwei benachbarten Transputern. Abbildung 6 zeigt die Rekonstruktionsergebnisse eines Polyeders nach 40 Iterationen auf einem 32 × 32-Gitter. Der Speedup und die Effizienz der Parallelimplementierung wird in Abbildung 7 dargestellt.

5 Ausblick

Interessant wäre die Untersuchung der Parallelisierung weiterer Verfahren. So z.B. wird in [10, 11] das Problem der Oberflächenrekonstruktion in zwei Phasen aufgeteilt. Zuerst werden die Topologie der unbekannten Oberfläche und eine grobe Schätzung ihrer Geometrie bestimmt. Dann wird mit Hilfe einer Methode zur 'Mesh Optimization'[5]der Ausgleich der n vorgegebenen Meßpunkte $\vec{x}_i \in R^3$ verbessert und die Anzahl m der Eckpunkte $\vec{v}_j \in V$ von (K, V) wird reduziert. Im Energiemodell wird die zu minimierende Funktion folgendermaßen ausgelegt:

[5]Bei einem 'Mesh' (K, V) stellt K die Topologie dar und $V = \{\vec{v}_i \in R^3\}$ stellt die geometrische Realisierung dar (Menge der Eckpunkte).

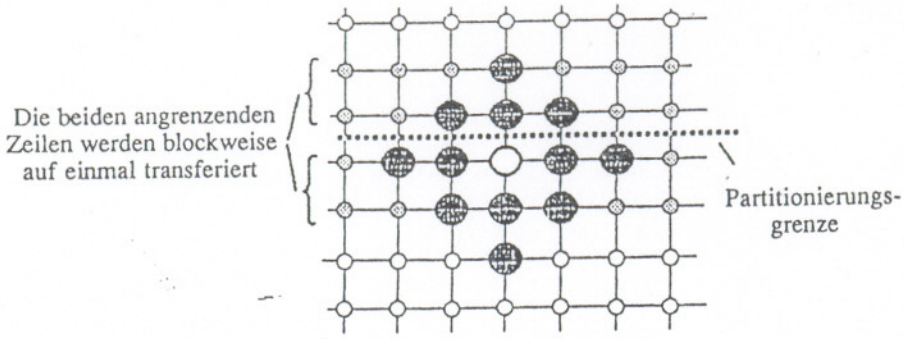

Abbildung 5: Knotenpartitionierung

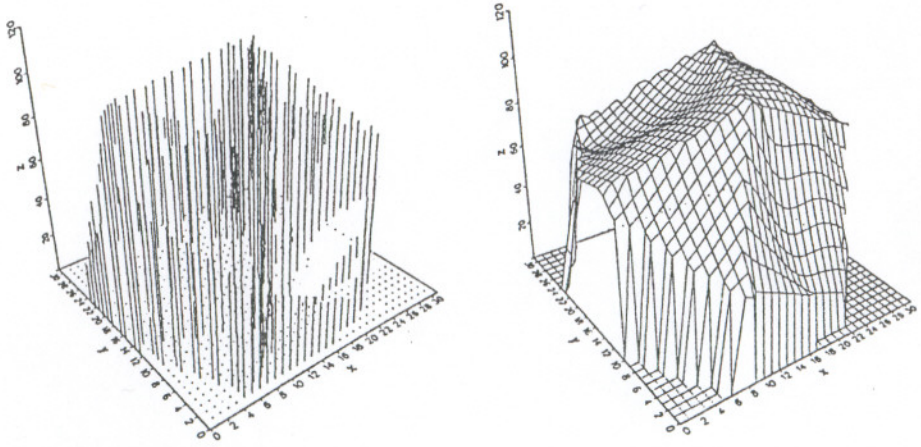

Abbildung 6: Rekonstruktionsbeispiel

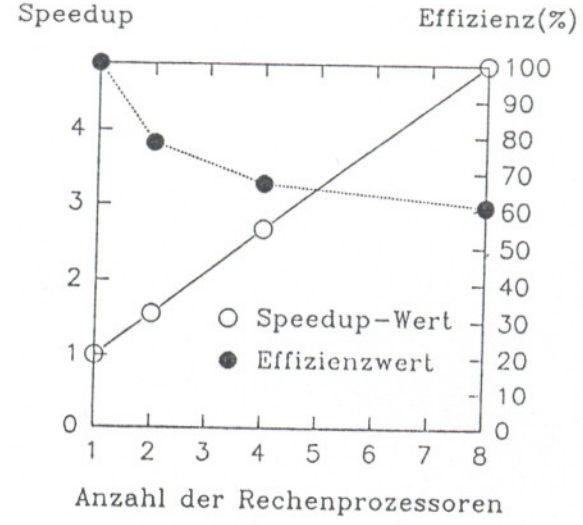

Abbildung 7: Speedup und Effizienz der Parallelimplementierung

$$E(K,V) = E_{dist}(K,V) + E_{rep}(K,V) + E_{reg}(K,V) \quad (29)$$

$$E_{dist}(K,V) = \sum_{i=1}^{n} d^2(\vec{x}_i, \phi_V(|K|)) \quad (30)$$

$$E_{rep}(K,V) = c_{rep} \cdot m \quad (31)$$

$$E_{reg}(K,V) = \sum_{\{j,k\} \in K} \kappa \cdot ||\vec{v}_j - \vec{v}_k|| \quad (32)$$

$E_{dist}(K,V)$ mißt die Distanz der Meßpunkte zur Oberfläche. In diesem Kontext ist $E_{rep}(K,V)$ neu und ein Straf-Term, proportional zur Anzahl der Eckpunkte. Der Parameter c_{rep} erlaubt die Einstellung einer gröberen Repräsentation mit niedrigerem Datenausgleich oder einer feineren Repräsentation mit größerem Datenausgleich. $E_{reg}(K,V)$ ist nicht wie üblich ein Straf-Term, der z.B. Stetigkeit kontrolliert, sondern ist ein Regularisierungsterm der darunterliegenden Optimierung. Der Einsatz weiterer parallelisierten Optimierungstechniquen in diesem Problemkreis bleibt ebenfalls äußerst ansprechend.

Literatur

[1] G. Bader, R. Rannacher, and G. Wittum, *Numerische Algorithmen auf Transputersystemen*, Teubner, Stuttgart, 1993.

[2] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel Distributed Computation*, Prentice-Hall, Englewood Cliffs, 1989.

[3] A. Blake and A. Zisserman, *Visual Reconstruction*, The MIT Press, Cambridge, Massachusetts, 1987.

[4] R.M. Bolle and B.C. Vemuri, *On three-dimensional surface reconstruction methods*, IEEE Trans. PAMI 13 (1991) 1, 1-13.

[5] D.J. Choi and J.R. Kender, *Solving the depth interpolation problem on a parallel architecture with a multigrid approach*, Proc. IEEE Conf. Computer Vision and Pattern Recognition, June 1988, 189-194.

[6] J.W. Demmel, M.T. Heath, and H.A. van der Vorst, *Parallel numerical linear algebra*, in: Acta Numerica 1993, Cambridge University Press 1993, 111-197.

[7] R.W. Freund, G.H. Golub, and N.M. Nachtigal, *Iterative solution of linear systems*, in: Acta Numerica 1992, Cambridge University Press 1992, 57-100.

[8] A. Frommer, *Lösung linearer Gleichungsysteme auf Parallelrechnern*, Vieweg, Braunschweig, 1990.

[9] G.H. Golub and C.F. van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 1989.

[10] H. Hoppe, T. DeRose, T. Duchamp, J.McDonald, and W. Stuetzle, *Surface reconstruction from unorganized points*, Computer Graphics (SIGRAPH'92 Proceedings) 26 (1992) 71-78.

[11] H. Hoppe, T. DeRose, T. Duchamp, J.McDonald, and W. Stuetzle, *Mesh Optimization*, Computer Graphics (SIGRAPH'93 Proceedings) 27 (1993) 19-26.

[12] R. Korzer und V.D. Sánchez A., *Untersuchung und Implementierung von Algorithmen für echtzeitfähige 3D-Rekonstruktion*, DLR-IB-515-90-10, Juni 1990.

[13] N.M. Missirlis, *Scheduling parallel iterative methods on multiprocessor systems*, Parallel Computing 5 (1987) 295-302.

[14] W. Niethammer, *The SOR method on parallel computers*, Numerische Mathematik (1989) 247-254.

[15] Y. Notay, *On the convergence rate of conjugate gradients in presence of rounding errors*, Numerische Mathematik 65 (1993) 3, 301-317.

[16] J. Rothammer und V.D. Sánchez A., *Untersuchung und Implementierung von Algorithmen für Echtzeit-Stereobildverarbeitung*, DLR-IB-515-89-32, Mai 1989.

[17] V.D. Sánchez A., *Eine parallel-verteilte Architektur für Rechnersehen und Telerobotik*, in: M. Baumann und R. Grebe (Hrsg.): Parallele Datenverarbeitung mit dem Transputer, Proc. TAT'92, Aachen, September 1992, Reihe Informatik aktuell, Springer-Verlag, Berlin, 1993, 295-303.

[18] H.R. Schwarz, *Numerische Mathematik*, Teubner, Berlin, 1993.

[19] R. Szeliski, *Fast surface interpolation using hierarchical basis functions*, IEEE Trans. PAMI 12 (1990) 6, 513-528.

[20] D. Terzopoulos, *The computation of visible-surface representations*, IEEE Trans. PAMI 10 (1988) 4, 417-438.

[21] W. Törnig und P. Spellucci, *Numerische Mathematik für Ingenieure und Physiker*, Band 1: Numerische Methoden der Algebra, Springer-Verlag, Berlin, 1988.

[22] R. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, 1962.

[23] D. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

[24] Y.-J. Zheng, *Digital photogrammetric inversion: theory and application to surface reconstruction*, Photogrammetric Engineering & Remote Sensing 59 (1993) 4, 489-498.