Endbericht

der
Projektgruppe 489
Graphaeology.NET

vorgelegt von

Chachaj, Adam
Frenzel, David
Garstka, Jens
Griehsel, Lars
Hasebein, Anke
Hüntler, Timo
Petke, Matthias
Rabe, Garvin
Ruppelt, Mark
Schowe, Benjamin
Vossebein, Dennis
Zimmermann, Christian

27. April 2007

Betreuer : Dipl.-Inf. Claus-Peter Alberts
Dipl.-Inf. Martin Wawro

Universität Dortmund
Fachbereich Informatik
Lehrstuhl für grafische Systeme
D-44221 Dortmund

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

In vielen Bereichen der Forschung und Wissenschaft ist eine 3D-Darstellung von Gegenständen, Werkstücken, Artefakten etc. nützlich. Beispielsweise werden in der Medizin Hüftgelenke vor einer Operation computergestützt rekonstruiert. In der Archäologie werden historisch wertvolle Artefakte digitalisiert und virtuell zugänglich gemacht.

Um eine solche 3D-Darstellung zu erreichen, scannt man die Gegenstände aus unterschiedlichen Richtungen und erhält so pro Scan ein 2,5-dimensionales Höhenmodell. Um aus diesen einzelnen Abtastungen ein vollständiges Modell zu erhalten, müssen die Scans miteinander registriert werden, d. h. die Koordinatensysteme der einzelnen Teile müssen zueinander ausgerichtet werden, wie in Abbildung 1.1 zu sehen ist.

Will man die 3D-Daten über ein Netzwerk zur Verfügung stellen, muss man beachten, dass Scans und komplette Modelle mehrere hundert Megabyte groß sein können. Daher ist es wünschenswert, die Daten abschnittsweise so zu übertragen, dass der Betrachter jederzeit einen groben Überblick über das Modell erhält.

1.1 Graphaeology.NET

Die Projektgruppe Graphaeology.NET hat ein System entwickelt, um solche Teilscans zu einem kompletten Modell zusammenzusetzen und in einer Client-Server-Umgebung zur Verfügung zu stellen. Abbildung 1.2 zeigt einen schematischen Arbeitsablauf und das Zusammenspiel der einzelnen Module Registrierung, Reduktion und Client-Server. Zuerst werden die Teilscans miteinander registriert, um ein vollständiges Modell zu erhalten. Ist dies erfolgreich geschehen, wird das Modell mithilfe des Reduzierers auf verschiedene Detailstufen gebracht um es dann in der Client-Server Umgebung zur Verfügung zu stellen. Die einzelnen Module werden im Folgenden kurz vorgestellt.

1.1.1 Registrierung

Die 3D-Modelle werden in der Vorverarbeitung aus einer Anzahl von Teilscans, die Ausschnitte des Artefakts abdecken, zusammengesetzt. Diese Teilscans können beispielsweise von einem taktilen Scanner stammen. Um das Modell zusammenzusetzen, werden verschiedene Registrierungsverfahren eingesetzt, die eine möglichst vollautomatische Abwicklung der Registrierung ermöglichen sollen. Dabei wird jeder Scan gegen alle anderen Scans getestet.

Anfangs werden die Teilscans mittels eines Grobregistrierungsverfahrens, von denen eine Reihe Unterschiedlicher zur Auswahl stehen, registriert. Es wurden deshalb mehrere unterschiedliche Verfahren implementiert, da es noch kein Verfahren gibt, das stets zu guten Ergebnissen führt. Jedes Verfahren gibt jeweils nach der erfolgten Registrierung von zwei Scans eine Bewertung aus, die angibt wie gut diese

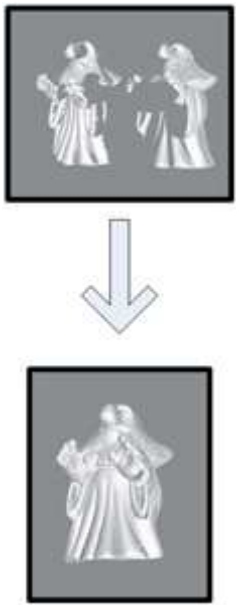

Abbildung 1.1: Schema eines Registrierungsvorgangs bei dem zwei Teilscans zu einem zusammengesetzt werden.

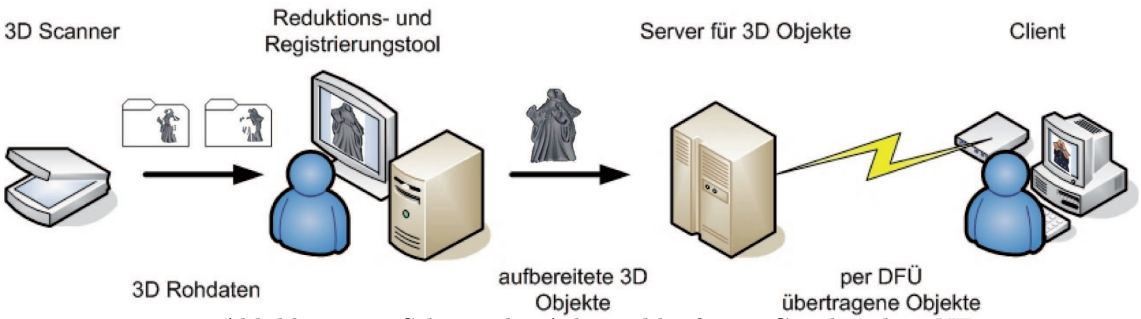

Abbildung 1.2: Schema des Arbeitsablaufs von Graphaeology.NET

beiden Scans zusammenpassen. Das Ergebnis dieser Registrierung wird in einem so genannten Modellgraf eingetragen, dessen Knoten die Scans repräsentieren, während die Kanten mit der Bewertung als Kantengewicht die Registrierungen darstellen.

Anschließend wird das Ergebnis der Grobregistrierung mittels eines Feinregistrierungsverfahrens präzisiert. Hierzu wird ein ICP-Verfahren eingesetzt. Es berechnet zueinander nächste Punkte zweier Scans und minimiert dann deren Entfernung. Dabei konvergiert es gegen ein lokales Optimum, so dass es auf eine erfolgreiche Grobregistrierung angewiesen ist.

Nach der Ausführung einer Feinregistrierung wird mittels einer Globalregistrierung im Grafen nun ein Spannbaum aufgebaut, so dass die Güte der verwendeten Kanten möglichst hoch ist. Mittels dieses Spannbaums, der alle zu registrierenden Knoten verknüpft, können dann die Scans, deren Anordnung nun bekannt ist, zu einem kompletten Modell zusammengesetzt werden. Dabei werden gleichzeitig die bei der vorhergehenden Registrierung aufgetretenen Restfehler über den Spannbaum verteilt, so dass an keiner Stelle durch große Abweichungen benachbarter Scans unschöne Nahtstellen auftreten. Die Globalregistrierung gibt eine Punktwolke aus, die durch ein vom Lehrstuhl bereitgestelltes Programmmodul trianguliert werden soll.

Nach erfolgreicher Registrierung erhält man schließlich ein detailliertes 3D-Modell des gescannten Artefaktes in Form eines Dreiecksnetzes. Als Dateiformat wird das Object-File-Format OFF (siehe Kapitel

B.2) verwendet, ein betriebssystemunabhängiges, ASCII-basiertes Dateiformat, um 3D-Objekte zu speichern.

1.1.2 Reduktion

Im Anschluss an die Registrierung wird das berechnete Modell auf mehrere modellabhängige Detailstufen (*levels of detail*) reduziert, da die Datenmenge wie erwähnt sehr groß sein kann. So kann der Server dem Client erst grob aufgelöste Scans liefern und nach und nach, je nach Anforderung, höher aufgelöste Bereiche hinterherschicken. Der Reduzierer wandelt die Daten anschließend in ein streambares Datenformat.

1.1.3 Client-Server

Der Server stellt die fertigen Modelle in dem Format, das der Reduzierer liefert, bereit, so dass der Client verschiedene Modelle und deren verschiedene Detailstufen anfordern kann. Dabei wird über eine serverseitige Benutzerverwaltung sichergestellt, dass die Modelle nur authorisierten Benutzern zur Verfügung gestellt werden. Im Client kann nun das Modell betrachtet werden. Dabei kann das Modell rotiert, verschoben und skaliert sowie die Beleuchtung beeinflusst werden. Außerdem können zum Modell hinterlegte, zusätzliche Informationen als HTML-Seite abgerufen werden.

1.2 Struktur des Berichts

Der vorliegende Endbericht dokumentiert die Arbeit der Projektgruppe Graphaeology.NET. Im Folgenden Kapitel werden die notwendigen mathematischen Grundlagen und Konzepte dargestellt und erläutert. Anschließend werden in den Kapiteln 3, 4 und 5 die einzelnen Module Registrierung, Reduktion und Client/Server und deren Bestandteile vorgestellt und erläutert. Dabei wird auf die jeweilige allgemeine Vorgehensweise eingegangen, nicht jedoch auf Implementierungsdetails.
Das darauf folgende Kapitel beschäftigt sich mit der tatsächlichen Implementierung. Dort werden die Klassendiagramme der Module vorgestellt und erläutert. Anschließend wird das System evaluiert und die einzelnen implementierten Verfahren werden kritisch betrachtet, bevor dann in den Anhängen das Handbuch zur Benutzung des Systems, die verwendeten Dateiformate und die benötigten externen Bibliotheken behandelt werden.

Kapitel 2

Grundlagen

In diesem Kapitel werden einige theoretische Grundlagen vermittelt. Die beschriebenen Verfahren werden innerhalb des Projektes an verschiedenen Stellen genutzt. So wird die Parametrisierung und die Krümmungsberechnung für verschiedene Registrierungsverfahren benötigt. Die Quaternionen hingegen kommen im Wesentlichen bei der Feinregistrierung aber auch bei der Rotation von Objekten im Zusammenhang mit der Visualisierung vor. Die Erläuterungen sollen lediglich als Grundlagen verstanden werden. Daher wird in den Kapiteln auf die entsprechenden Quellen verwiesen.

2.1 Parametrisierung

Bei der Parametrisierung handelt es sich allgemein um eine Abbildung von $\mathbb{R}^3$ nach $\mathbb{R}^2$. In dieser Projektgruppe werden Dreiecksnetze im $\mathbb{R}^3$ genutzt. Da einige Registrierungsverfahren wie *Mutual Information* (siehe Kapitel 3.1.6) oder *NDT* (siehe Kapitel 3.1.10) jedoch auf zweidimensionalen Netzen oder Bitmaps arbeiten, wird eine Parametrisierung benötigt, die ein Dreiecksnetz auf einen zweidimensionalen Parameterbereich abbildet.

Das hier eingesetzte Verfahren basiert auf dem Verfahren von Floater [7]. Die dort präsentierte Parametrisierung arbeitet mit offenen Objekten[1]. Dabei werden zuerst die Randknoten eines Objektes auf den Rand eines konvexen Polygons abgebildet. Anschließend wird für jeden inneren Knoten seine Position im parametrisierten Netz durch eine Gleichung formuliert. Mit dieser Gleichung wird seine Position als Kombination der Positionen seiner Netznachbarn bestimmt. Es wird direkt deutlich, dass hierbei ein sehr großes Gleichungssystem entsteht, welches letztlich eine über der Anzahl der Knoten quadratische Menge an Koeffizienten enthält. Durch die Vorgabe der Randknoten ist dieses lineare Gleichungssystem lösbar. Stehen die Koordinaten der Knoten fest, ist die Parametrisierung abgeschlossen. In diesem Kapitel sollen die Grundlagen des Verfahrens von Floater erläutert werden. Für nähere Details des theoretischen Hintergrundes sei auf [16] verwiesen.

Im Folgenden sei eine Fläche S als Dreiecksnetz gegeben, welche durch $S_T = T_1, ..., T_M$ definiert ist. Hierbei sind die T_i die einzelnen Dreiecke des Netzes. Das Dreiecksnetz besteht aus den Knoten der Menge V, welche in die Menge der Randknoten V_B und die Menge der innen liegenden Knoten V_I unterteilt werden kann. Da die Fläche S_T nach $\mathbb{R}^2$ abgebildet werden soll, genügt es eine Abbildung ψ für die Knoten zu finden. Dies kann wie folgt geschrieben werden:

$$\psi(\vec{v}) \in \mathbb{R}^2, \quad \forall \vec{v} \in V \subseteq \mathbb{R}^3$$

Bei dem Verfahren von Floater handelt es sich um eine Abbildungen basierend auf Konvexkombinationen. Wichtig ist hierbei, dass ein Dreiecksnetz auf ein konvexes Polygon abgebildet wird. Dieses Verfahren funktioniert demnach nur für einen konvexen Parameterbereich. Im Vorfeld wird ein konvexes Polygon ausgewählt, in der von der PG implementierten Form ist es ein Quadrat. Nun müssen die Randpunkte

[1] Als offenes Objekt wird ein Dreiecksnetz verstanden, welches Kanten mit nur einem einzigen inzidenten Dreieck enthält.

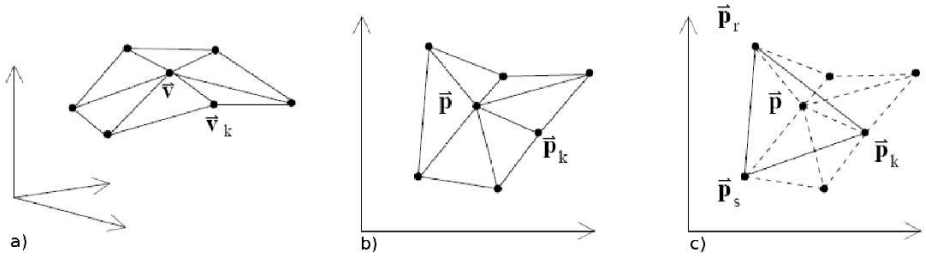

Abbildung 2.1: Berechnung lokaler baryzentrischer Koordinaten, entnommen aus [6]

des Dreiecksnetzes, $\vec{v} \in V_B$, auf den Rand des Polygons abgebildet werden. Wichtig ist die Reihenfolge der Punkte beizubehalten. Weiterhin sollte der Abstand der Randpunkte auf dem Rand des konvexen Polygons proportional zu den Abständen im Dreicksnetz gewählt werden, dies hilft die Verzerrungen am Rand gering zu halten. Sind die Randpunkte nun auf dem Rand des Polygons verteilt, kann man die Position der inneren Knoten $\psi(\vec{v}), \vec{v} \in V_I$, bestimmen.

Die Idee ist, die Position jedes Knotens als konvexe Kombination seiner Nachbarn zu bilden. Dabei wird zu jedem inneren Knoten $\vec{v} \in V_I$ die Menge seiner Nachbarn $N_{\vec{v}} = \{\vec{w} \in V : (\vec{w}, \vec{v}) \in E\}$ bestimmt, das sind diejenigen Knoten, die mit $\vec{v}$ zusammen eine gemeinsame Kante haben. Jeder Nachbarknoten geht mit einem Gewicht λ in die Kombination ein. Die Abbildung von $\vec{v}$ kann man wie folgt schreiben:

$$\psi(\vec{v}) = \sum_{\vec{w} \in N_{\vec{v}}} \lambda_{\vec{v}\vec{w}} \psi(\vec{w}), \quad \vec{v} \in V_I \tag{2.1}$$

Weiterhin muss für die Gewichte gelten, dass sie die Partition der Eins erfüllen und positiv sind:

$$\sum_{\vec{w} \in N_{\vec{v}}} \lambda_{\vec{v}\vec{w}} = 1, \quad \lambda_{\vec{v}\vec{w}} > 0, \quad \forall \vec{v}, \vec{w} \in V \tag{2.2}$$

Diese kombinatorischen Formeln bilden ein Gleichungssystem. Es ist lösbar, da bereits die Randpunkte festgelegt sind. Die Lösbarkeit kann man ebenfalls an der Form der Matrix ersehen, wenn das Gleichungssystem in Matrixschreibweise aufgestellt wird. Nun müssen noch die Gewichte entsprechend der Form des Netzes gewählt werden.

Die hier benutzte Wahl der Gewichte bezeichnet Floater als „Form erhaltende Parametrisierung“ (*Shape Preserving Parameterization*). Die Gewichte werden folgendermaßen gewählt: Das Prinzip ist, erst einmal eine lokale Parametrisierung mittels baryzentrischer Koordinaten vorzunehmen, und dann daraus die entsprechenden Knotengewichte zu errechnen.

Als erster Schritt wird ein beliebiger innerer Knoten $\vec{v} \in V_I$ mit seinen Nachbarn $N_{\vec{v}}$ gewählt. Dieser Knoten $\vec{v}$ wird als Punkt $\vec{p}$ in die $\mathbb{R}^2$-Ebene abgebildet. Dabei wählt man in der Regel $\vec{p} = \vec{0}$. Seine Nachbarn werden ebenfalls abgebildet. Abbildung 2.1 zeigt diesen Vorgang. Dabei müssen folgende Bedingungen erfüllt werden:

$$|\vec{p_j} - \vec{p}| = |\vec{v_j} - \vec{v}|, \forall \vec{v_j} \in N_{\vec{v}}$$

$$ang(\vec{v_k}, \vec{v}, \vec{v_j}) = c \cdot ang(\vec{p_k}, \vec{p}, \vec{p_j})$$

Dies bedeutet, dass die Abstände erhalten bleiben sollen, zwei Knoten sollen im 3D-Netz den gleichen Abstand haben wie in der 2D-Darstellung. Ferner sollen Winkel proportional abgebildet werden. Diese Formeln sind für die entsprechenden korrespondierenden Dreiecke formuliert, c ist dabei eine Konstante. Sie wird eingeführt um sicherzustellen, dass die Dreiecke um den Punkt $\vec{p}$ diesen vollständig umfassen, sich ihre Winkel am Punkt $\vec{p}$ in der Abbildung exakt zu der Winkelsumme von 2π aufsummieren. Abbildung 2.1 stellt diesen Vorgang dar. Sie zeigt in 2.1(a) den Knoten $\vec{v}$ mit seinen Nachbarn im Raum und in 2.1(b) die abgebildeten Punkte $\vec{p}$ im $\mathbb{R}^2$.

Im Folgenden wird der Punkt $\vec{p}$ mittels baryzentrischer Koordinaten relativ zu jedem möglichen Dreieck dargestellt. Hierfür wird ein Punkt $\vec{p_k}$ genommen und diejenige Kante gesucht, mit der sich ein Dreieck bilden lässt, in dem $\vec{p}$ enthalten ist. Abbildung 2.1(c) zeigt dieses Prinzip. Hier befindet sich zwischen

$\vec{p_r}$ und $\vec{p_s}$ diejenige Kante, welche mit $\vec{p_k}$ ein Dreieck bildet, welches $\vec{p}$ enthält. Der Punkt $\vec{p}$ kann nun mittels baryzentrischer Koordinaten $(\tau_k^k, \tau_r^k, \tau_s^k)$ relativ zu den Eckpunkten wie folgt angegeben werden:

$$\vec{p} = \tau_k^k \vec{p_k} + \tau_r^k \vec{p_r} + \tau_s^k \vec{p_s}$$

Man kann diese Formel auch bezogen auf alle Nachbarn von $\vec{p}$ schreiben:

$$\vec{p} = \sum_{j=1}^{d} \tau_j^k p_j,\ mit\ \tau_j^k = 0,\ \forall j \neq k, r, s,\ d = |N_{\vec{v}}|$$

Die baryzentrischen Koordinaten werden nun für alle möglichen Dreiecke berechnet. Dies geschieht, indem man $\vec{p_k}$ gegen den Uhrzeigersinn wandern lässt und dann immer die entsprechenden Koordinaten berechnet.

Sind nun für alle möglichen Dreiecke diese Koordinaten berechnet, so kann man aus dieser lokalen Parametrisierung eine globale Parametrisierung berechnen. Dafür werden die Gewichte wie folgt berechnet:

$$\lambda_{\vec{v}\vec{v_j}} = \frac{1}{d} \sum_{k=1}^{d} \tau_j^k$$

Die globalen Gewichte berechnen sich als Mittel der baryzentrischen Koordinaten, sie werden durch die Anzahl der Nachbarn gemittelt. Somit ergibt sich auch hier die Partition der Eins, und alle Gewichte sind positiv, da baryzentrische Koordinaten positiv sind, wenn der Punkt im Dreieck liegt.

Die konkrete Form des Gleichungssystems für die globale Parametrisierung ergibt sich nun analog zur *uniformen Parametrisierung*, bei der $\lambda_{\vec{v}\vec{v_j}}$ allerdings konstant auf $\frac{1}{d}$ gesetzt wird, als

$$\psi(\vec{v}) = \sum_{j=1}^{d} \lambda_{\vec{v}\vec{v_j}} \vec{v_j},\ \vec{v} \in V_I$$

Durch diesen Ablauf ist der Aussenrand parametrisiert und es stehen die Gewichte fest, mit deren Hilfe sich innere Knoten als Linearkombination ihrer Nachbarn abbilden lassen. Das daraus resultierende Gleichungssystem wird in diesem Projekt mit der UMFPACK-Bibliothek gelöst. Die dabei entstehenden Matrizen liegen in einer kompakten Form für spärlich besetzte Matrizen vor.

Es gibt zwei bekannte Probleme bei der Anwendung der Parametrisierung. Das erste ist ein Sonderfall bei der Berechnung der baryzentrischen Koordinaten, wobei diese noch nachträglich skaliert werden müssen. Dies ist immer dann der Fall, wenn der betrachtete Knoten auf einer Kante liegt, welche sich durch Verbinden zweier Nachbarknoten bilden lässt. Definiert man den betrachteten Knoten als im Dreieck liegend, wenn er auf der Kante liegt, so lassen sich zwei Dreiecke bilden, welche genau diese Kante gemein haben. Damit hat man ein Dreieck mehr als man haben dürfte und die Gewichte summieren sich zu einem Wert größer eins auf. Wenn man den Knoten als nicht im Dreieck liegend definiert, so wie in der PG geschehen, so fehlt ein Dreieck und die aufaddierten Gewichte sind kleiner. In diesem Fall werden die Gewichte so skaliert, dass sie eins ergeben. Als Beispiel kann es bei taktilen Abtastungen durch die Rasterung der Abtastpunkte zu diesem Sonderfall kommen.

Der zweite Fall ist eine Einschränkung bei den Eingabenetzen. Diese müssen für dieses Verfahren frei von Löchern sein, da das Gleichungssystem sonst nicht lösbar ist, und die Randkanten der Löcher sonst als Aussenrand des Netzes interpretiert werden könnten. Im Programm hilft eine Deaktivierung des Filters für lange Kanten, um keine extra Löcher zu erzeugen. Wenn das Netz allerdings schon Löcher enthält, ist eine Nutzung der Parametrisierung nicht möglich.

2.2 Krümmungsberechnung

Die hier vorgestellte Methode zur Krümmungsberechnung basiert auf den Ausführungen von Garimella und Swartz [13]. Das Anlegen von Quadriken[2], um eine Krümmung zu bestimmen, basiert auf der Idee,

[2] Als Quadrik bezeichnet man in der linearen Algebra spezielle Polynomfunktionen zweiten Grades.

dass die Oberfläche eines Objektes über eine quadratische Gleichung abgeschätzt werden kann. Hierzu wird die Quadrik über die Nachbarn des Punktes $\vec{p}$, dessen Krümmung bestimmt werden soll, gelegt. Der Punkt $\vec{p}$ stellt gleichzeitig auch den Ursprung des lokalen Koordinatensystems über x',y' und z', in dem sich die Quadrik befindet, dar. Die Ausrichtung des lokalen Koordinatensystems erfolgt über die Normale von $\vec{p}$, wobei diese die z'-Achse darstellt. Die Krümmung der Quadrik bei $\vec{p}$ wird benutzt, um die Krümmung der diskreten Oberfläche abzuschätzen. Eine einfache Quadrik ist über $z' = ax'^2 + bx'y' + cy'^2$ gegeben.

Die folgenden sechs Schritte dienen zur Bestimmung der Krümmung:

1. Bestimmen der Normalen $\vec{n}$ von $\vec{p}$.

2. Erzeugen eines lokalen Koordinatensystem (x', y', z') mit $\vec{p}$ als Ursprung. Die Normale von $\vec{p}$ stellt hierbei die z'-Achse dar. Um die x'-Achse zu fixieren, wird diese über die Projektion der x-Achse auf die tangentiale Ebene, die über $\vec{n}$ definiert wird, gelegt. Dies resultiert in einer Rotationsmatrix $\boldsymbol{R} = [r_1, r_2, r_3]^T$, um vom globalen ins lokale Koordinatensystem zu transformieren:

$$\vec{r_3} = \vec{n} \tag{2.3}$$
$$\vec{r_1} = \frac{(\boldsymbol{I} - \vec{n}\vec{n}^T)\boldsymbol{i}}{|(\boldsymbol{I} - \vec{n}\vec{n}^T)\boldsymbol{i}|} \tag{2.4}$$
$$\vec{r_2} = \vec{r_3} \times \vec{r_1} \tag{2.5}$$

   Hierbei ist $\boldsymbol{I}$ die Identitätsmatrix und $\boldsymbol{i}$ die globale x-Achse $[1, 0, 0]^T$.

   Ein problematischer Sonderfall, der auftreten kann, ist wenn der Normalenvektor $\vec{n}$ der globalen x-Achse entspricht. In dem Fall würde die Berechnung von $\vec{r_1}$ einen Nullvektor ergeben. Um dies zu verhindern wird statt der globalen x-Achse die globale y-Achse für die Projektion benutzt.

3. Es wird die Menge der Punkte bestimmt, auf die eine Quadrik gelegt werden soll. Die einfachste Möglichkeit ist es die Nachbarn von $\vec{p}$ zu nehmen.

4. Mit Hilfe der Rotationsmatrix $\boldsymbol{R}$ werden diese Punkte in das lokale Koordinantensystem transformiert. Es gilt folgende Beziehung: $\vec{x'} = \boldsymbol{R}(\vec{x} - \vec{p})$.

5. Um die Koeffizienten der Quadrik zu bestimmen, muss folgendes Gleichungssystem gelöst werden:

$$\begin{bmatrix} x_1^2 & x_1y_1 & y_1^2 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_ny_n & y_n^2 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} \tag{2.6}$$

   Bemerkung: Die Lösung eines Gleichungssystems der Form $\boldsymbol{A}\vec{x} = \vec{b}$ ergibt sich als $\vec{x} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\vec{b}$.

6. Die Schätzungen der Hauptkrümmungen k_1 und k_2, der Gaußkrümmung K und der mittleren Krümmung H (alle in $\vec{p}$) ergeben sich als:

$$k_1 = a + c + \sqrt{(a-c)^2 + b^2} \tag{2.7}$$
$$k_2 = a + c - \sqrt{(a-c)^2 + b^2} \tag{2.8}$$
$$K = 4ac - b^2 \tag{2.9}$$
$$H = a + c \tag{2.10}$$

Die Schätzung der Krümmung über eine derart einfache Quadrik hat den Nachteil, dass das Ergebnis enorm von der Abschätzung der Oberflächennormalen abhängig ist. Um diese Empfindlichkeit zu reduzieren erweitert man die Quadrik um lineare Komponenten. Die erweiterte Quadrik ergibt sich dann als

$z' = ax'^2 + bx'y' + cy'^2 + dx' + ey'$. Die Koeffizienten dieser Quadrik können benutzt werden, um eine neue Abschätzung für die Normale zu erhalten. Diese ergibt sich als:

$$\vec{n} = \frac{1}{(d^2 + e^2 + 1)^{1/2}}[-d, -e, 1]^T \tag{2.11}$$

Die neue Abschätzung der Normalen erlaubt es bei $\vec{p}$ ein neues lokales Koordinatensystem zu berechnen und eine neue Quadrik anzulegen. Dieser Vorgang kann wiederholt werden, bis die Normalen gegen einen bestimmten Vektor konvergieren. Die Gaußkrümmung K und die mittlere Krümmung H ergeben sich aus den Koeffizienten der resultierenden Quadrik:

$$K = \frac{4ac - b^2}{(1 + d^2 + e^2)^2} \tag{2.12}$$

$$H = \frac{a + c + ae^2 + cd^2 - bde}{(1 + d^2 + e^2)^{3/2}} \tag{2.13}$$

2.3 Quaternionen

Quaternionen sind eine vierdimensionale Divisionsalgebra über dem Körper der reellen Zahlen mit einer nicht kommutativen Multiplikation [37]. Als vierdimensionale reelle Algebra sind die Quaternionen ein vierdimensionaler reeller Vektorraum. Daher ist jedes Quaternion durch vier reelle Komponenten x_0, x_1, x_2, x_3 eindeutig bestimmt. Als Basiselemente dieses Vektorraums werden vier Elemente mit der Länge 1 gewählt, die senkrecht aufeinander stehen. Sie werden mit $1, i, j, k$ bezeichnet. Die Linearkombination der vier Komponenten mit den vier Basiselementen ist das Quaternion:

$$x_0 + x_1 i + x_2 j + x_3 k = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{2.14}$$

Einsatz in der Computergrafik

Arthur Cayley entdeckte, dass sich mit Quaternionen Drehungen im Raum beschreiben lassen. Genutzt wird dies heutzutage im Bereich der interaktiven Computergrafik, insbesondere bei Computerspielen, sowie bei der Steuerung und Regelung von Satelliten. Bei der Verwendung von Quaternionen an Stelle von Drehmatrizen werden etwas weniger Rechenoperationen benötigt. Insbesondere, wenn viele Drehungen miteinander kombiniert (multipliziert) werden, steigt die Verarbeitungsgeschwindigkeit.

Drehung mit Quaternionen

Um mit Quaternionen eine Drehung zu berechnen, benötigt man die Rotationsachse $a = (x, y, z)^T$ und einen Winkel ϕ. Mit diesen Informationen kann man ein Quaternion definieren:

$$q^\circ = \begin{pmatrix} w_q \\ x_q \\ y_q \\ z_q \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{\phi}{2}\right) \\ x \cdot \sin\left(\frac{\phi}{2}\right) \\ y \cdot \sin\left(\frac{\phi}{2}\right) \\ z \cdot \sin\left(\frac{\phi}{2}\right) \end{pmatrix} \tag{2.15}$$

Dabei ist zu beachten, dass das Quaternion danach normalisiert werden muss, da sonst neben der Rotation auch eine Skalierung erfolgt.

Eine Rotation eines Vektors $\vec{v}$ erfolgt durch die Multiplikation des Quaternions mit dem Vektor und des konjugiert komplexen Quaternions:

$$\vec{v}' = q^\circ \cdot \vec{v} \cdot \overline{q^\circ} \tag{2.16}$$

$$\vec{v}' = \begin{pmatrix} w_{q^\circ} \\ x_{q^\circ} \\ y_{q^\circ} \\ z_{q^\circ} \end{pmatrix} \cdot \begin{pmatrix} 0 \\ x_{\vec{v}} \\ y_{\vec{v}} \\ z_{\vec{v}} \end{pmatrix} \cdot \begin{pmatrix} w_{q^\circ} \\ -x_{q^\circ} \\ -y_{q^\circ} \\ -z_{q^\circ} \end{pmatrix} \tag{2.17}$$

Ausmultipliziert ergibt sich:

$$\vec{v}' = \begin{pmatrix} w_{q^\circ}^2 + x_{q^\circ}^2 - y_{q^\circ}^2 - z_{q^\circ}^2 & 2(x_{q^\circ} y_{q^\circ} - w_{q^\circ} z_{q^\circ}) & 2(x_{q^\circ} z_{q^\circ} + w_{q^\circ} y_{q^\circ}) \\ 2(w_{q^\circ} z_{q^\circ} + x_{q^\circ} y_{q^\circ}) & w_{q^\circ}^2 - x_{q^\circ}^2 + y_{q^\circ}^2 - z_{q^\circ}^2 & 2(y_{q^\circ} z_{q^\circ} - w_{q^\circ} x_{q^\circ}) \\ 2(x_{q^\circ} z_{q^\circ} - w_{q^\circ} y_{q^\circ}) & 2(w_{q^\circ} x_{q^\circ} + y_{q^\circ} z_{q^\circ}) & w_{q^\circ}^2 - x_{q^\circ}^2 - y_{q^\circ}^2 + z_{q^\circ}^2 \end{pmatrix} \cdot \begin{pmatrix} x_{\vec{v}} \\ y_{\vec{v}} \\ z_{\vec{v}} \end{pmatrix} \tag{2.18}$$

Hinweis: Eine Rotation ist auf diese Art maximal bis zu einem Winkel von 180° möglich. Eine Alternative dazu sind Fermionen. Fermionen gelangen erst nach einer Rotation von 720° wieder in die Ausgangslage.

Kapitel 3

Registrierung

Der umfangreichste Teil der Arbeit dieser Projektgruppe lag in der Umsetzung mehrerer, möglichst unterschiedlicher Registrierungsverfahren. Die Registrierung mehrerer Scans zu einem Objekt gliedert sich in drei Teilaufgaben: Als erstes werden die einzelnen Scans grob zueinander ausgerichtet, so dass sie sich näherungsweise überlappen. Anschließend wird die Feinregistrierung ausgeführt, die die bereits grobregistrierten Scans nun entsprechend ihrer Überlappung genau aufeinander abbildet. Zuletzt wird durch die Globalregistrierung ein komplettes Modell aus allen bereits feinregistrierten Scan-Paarungen mit Hilfe des Modellgrafen erzeugt. Es werden jeweils die Paarungen mit den besten Bewertungen verwendet bis alle Abtastungen berücksichtigt wurden. Das komplette Modell muss in einem letzten Schritt noch als vollständiges Objekt zusammengesetzt werden.

Im Rahmen der Projektgruppe wurden verschiedene Verfahren zur Lösung der drei Teilaufgaben umgesetzt. In diesem Kapitel wird nun der theoretische Hintergrund der einzelnen Verfahren genauer betrachtet. Die Verfahren gliedern sich in die Bereiche Grob-, Fein- und Globalregistrierung.

3.1 Grobregistrierung

Ein erfolgreicher Grobregistrierungsvorgang zweier Teilscans bestimmt eine Transformation derart, dass der zweite Scan nach Anwendung dieser Transformation am ersten Scan ausgerichtet ist. Die Qualität der Ausrichtung wird dann von einer Bewertungsfunktion geprüft, um entscheiden zu können, ob die Registrierung erfolgreich war, d. h. ob die Transformation eine zufriedenstellende Ausrichtung erreicht hat. Kleinere Abweichungen bzw. Verschiebungen der Teilnetze zueinander sind hierbar tolerabel, da diese anschließend per Feinregistrierung ausgeglichen werden.

Um die Transformation zu bestimmen, gibt es verschiedenste Ansätze, wie beispielsweise auf Eigenschaften der Oberfläche zurückgreifende oder auf RANSAC[1] basierende Verfahren.

Leider existiert bis heute kein Grobregistrierungsverfahren, das immer zum Erfolg führt, je nach Anwendung und Einstellung der Parameter kann ein Verfahren mal mehr und mal weniger gute Ergebnisse liefern. Aufgrund dessen wird eine größere Auswahl an Verfahren zur Verfügung gestellt, die im Folgenden genauer beschrieben werden.

3.1.1 Manuelle Grobregistrierung

Bei der manuellen Grobregistrierung zwischen zwei Scans ist man bei der Bestimmung einer Transformation komplett auf die Eingabe bzw. die Informationen des Benutzers angewiesen. Es gibt recht viele verschiedene visuelle Möglichkeiten, wie man dem Benutzer diese Eingabe ermöglichen kann, z. B. über Schalter für Translationen und Rotationen, oder über eine Drei-Punkte-Registrierung.

[1] Ein mathematisches Schätzverfahren, das zur Anpassung mathematischer Figuren verwendet wird.

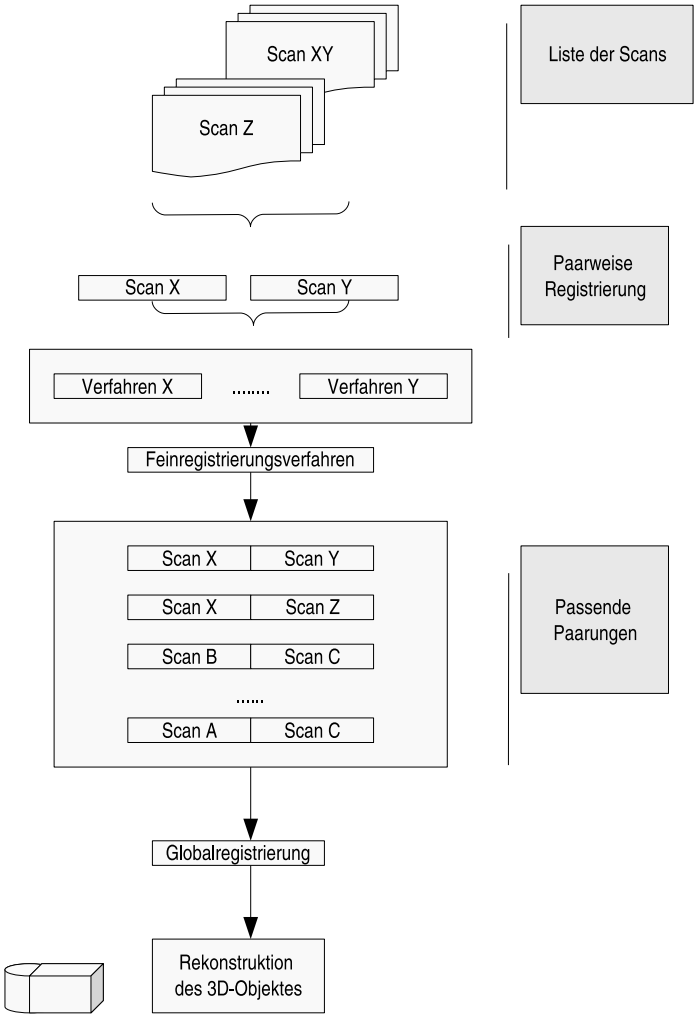

Abbildung 3.1: Grobregistrierungsverfahren: Grafische Darstellung des Vorganges der automatischen Registrierung von 3D-Scans. Zuerst werden die gegebenen Scans mit Hilfe eines Grobregistrierungsverfahrens zu passenden Paarungen gruppiert. Danach erfolgt bei den zueinanderpassenden Paarungen eine Feinregistrierung. Abschließend wird eine Globalregistrierung ausgeführt, die das 3D-Objekt zusammensetzt.

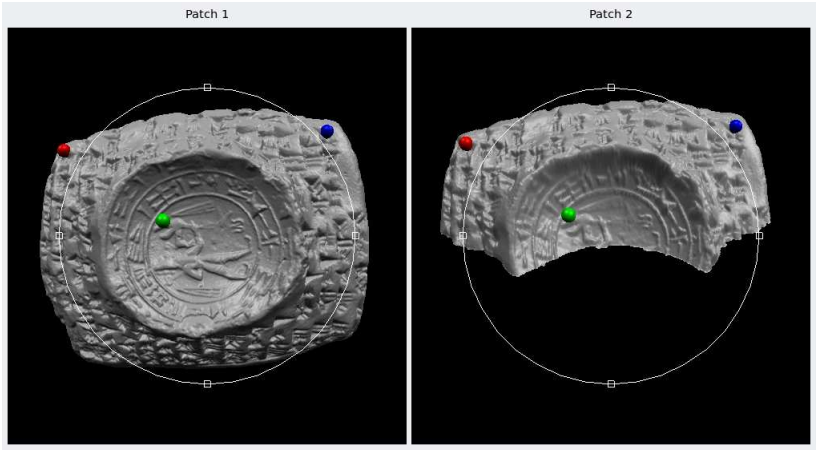

Abbildung 3.2: Manuelle Grobregistrierung: Die korrespondierenden Punkte werden als farbige Kugeln dargestellt

In dem Registrierungswerkzeug wird eine Drei-Punkte-Registrierung bevorzugt. Hierbei markiert der Benutzer in den beiden Scans drei korrespondierende Punkte (siehe Abbildung 3.2), die dann aufeinander transformiert werden. Das Ergebnis bzw. die Qualität dieser Methode ist maßgeblich von der Genauigkeit der Eingabe des Benutzers abhängig. Ungenaue Eingaben können zu beliebig schlechten Transformtionen führen.

3.1.2 Spin Tables, Winkelbach et al.

Winkelbach et al. [38] benutzen einen dem RANSAC[2]-Verfahren recht ähnlichen Ansatz. Jedoch werden zum Bestimmen der sechs Freiheitsgrade einer Transformation hier nicht drei sondern nur zwei Punkte benötigt. Es werden nämlich die Normalen auf allen Punkten berechnet, welche Freiheitsgrade festlegen.

Vorgehen

Szene und Modell sind zwei Scans, welche registriert werden sollen. Zuerst werden für alle Punkte in Szene S und Modell M deren Normalenvektoren berechnet. Dies geschieht in Zeit $O(|S|+|M|)^3$. Dann wählt man zufällig zwei Punkte $\vec{p}_i \in S$ und $\vec{p}_j \in M$. Durch die zwei Punkte sind nun fünf der sechs Freiheitsgrade festgelegt. Die relative Lage der Punkte bestimmt die drei translatorischen Freiheitsgrade und durch paralleles Ausrichten der Normalen beider Punkte werden noch zwei rotatorische Freiheitsgrade festgelegt. Einzig offen ist dann noch die Rotation um diese Normalen. Diese lässt sich durch Wahl zweier weiterer korrespondierender Punkte festlegen. Man wählt $\vec{q}_i \in S$ zufällig und auch einen korrespondierenden Punkt $\vec{q}_j \in M$. Dies geschieht mit Hilfe von *Spin-Tables*, welche im Folgenden genauer beschrieben werden. Danach kann die Transformation T_c berechnet werden. Anschließend bestimmt man, wie gut Szene und Modell durch T_c miteinander registriert sind. Dieser Vorgang wird solange wiederholt, bis die Qualität von T_c einen Schwellwert überschreitet oder ein „Zeitlimit“ erreicht wurde.

Bei der Wahl von $\vec{p}_i, \vec{p}_j$ bzw. $\vec{q}_i, \vec{q}_j$ ist noch eine Optimierung möglich. Man kann von vornherein nur Paare $\vec{p}_i, \vec{p}_j$ bzw. $\vec{q}_i, \vec{q}_j$ in Betracht ziehen, bei denen gewisse Oberflächeneigenschaften gleich sind, wie zum Beispiel die mittlere Oberflächenkrümmung.

[2]RANSAC heißt Random Sample Consensus. Dieses mathematische Verfahren wird zur Anpassung geometrischer Figuren verwendet und findet Anwendung in der Bildanalyse und in der automatisierten Kartografie. Der Algorithmus wurde 1981 erstmals von Fischler und Bolles veröffentlicht.

[3]Die O-Notation wird in der Mathematik und in der Informatik verwendet, um das asymptotische Verhalten von Funktionen und Folgen zu beschreiben. In der Informatik wird sie insbesondere in der Komplexitätstheorie verwendet, um verschiedene Probleme und Algorithmen danach zu vergleichen, wie „schwierig“ oder aufwendig sie zu berechnen sind.

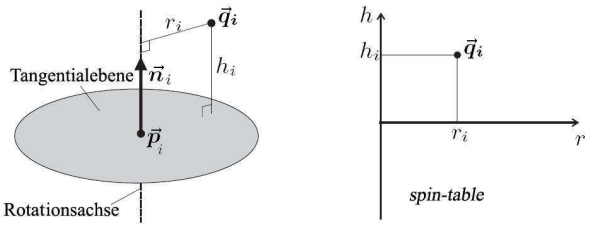

Abbildung 3.3: Spin-Tables: Koordinaten im Spin-Table

Spin-Tables

Den zu $\vec{q_i}$ korrespondierenden Punkt $\vec{q_j}$ findet man, indem man die Position von $\vec{q_i}$ relativ zu $\vec{p_i}$ und von $\vec{q_j}$ relativ zu $\vec{p_j}$ betrachtet; $\vec{q_i}$ hat den Abstand h_i von der Ebene, die durch $\vec{p_i}$ und den zugehörigen Normalenvektor $\vec{n}_i$ gegeben ist, und den Abstand r_i von der Achse $\vec{n}_i$ durch $\vec{p_i}$, siehe Abbildung 3.3. Also werden sukzessive für alle Punkte ihre Koordinaten (r, h) bezüglich $\vec{p_i}$ bzw. $\vec{p_j}$ berechnet, bis ein Paar mit gleichem (r, h) gefunden wird. Dies geschieht folgendermaßen:

1. Lösche Spin-Tables ST_S und ST_M.
2. Wiederhole k-mal (k ist ein Grenzwert und dient als Zeitlimit)
3. Wähle zufällig $\vec{q_i} \in S$
4. Berechne Spin-Table-Koordinaten (r_i, h_i) bzgl. $\vec{p_i}$
5. Füge $\vec{q_i}$ in seine Spin-Table ein: $ST_S(r_i, h_i) := \vec{q_i}$
6. $\vec{q_j} := ST_M(r_i, h_i)$
7. Wenn ein $\vec{q_j}$ existiert, dann gehe zu 10.; das neue Paar ist $(\vec{q_i}, \vec{q_j})$
8. Vertausche die Rollen von S,M und ST_S,ST_M und gehe zu 2
9. Falls die k Wiederholungen kein Ergbnis lieferten, beende erfolglos
10. Berechne T_c mit Hilfe von $(\vec{p_i}, \vec{p_j}), (\vec{n}_i, \vec{n}_j)$ und $(\vec{q_i}, \vec{q_j})$.

Güte der Transformation

Die Güte einer möglichen Transformation T_c lässt sich durch die relative Größe der Überschneidung Ω von transformierter Szene und Modell ausdrücken. Man nehme an, dass die Oberflächen sich berühren, wenn der Abstand zweier Oberflächenpunkte kleiner als ε ist. Genauso kann Ω als die Wahrscheinlichkeit gedeutet werden, mit der ein zufälliger Punkt aus $T_c * S$ mit M in Berührung kommt. Daher kann Ω mit einer effizienten *Monte-Carlo-Strategie*[4] vorhergesagt werden. Seien $\vec{x}_1, \ldots, \vec{x}_n$ zufällige, unabhängige Punkte mit $\vec{x}_i \in T_c * S$. Dann ist

$$\Omega = \lim_{n \to \infty} \frac{\sum_{i=1}^{n} contact_M(\vec{x}_i)}{n} \tag{3.1}$$

mit

[4]Monte-Carlo-Algorithmen sind randomisierte Algorithmen, die mit einer nach oben beschränkten Wahrscheinlichkeit ein falsches Ergebnis liefern dürfen. Dafür sind sie im Vergleich zu deterministischen Algorithmen häufig effizienter. Ihr Nachteil besteht darin, dass das berechnete Ergebnis falsch sein kann.

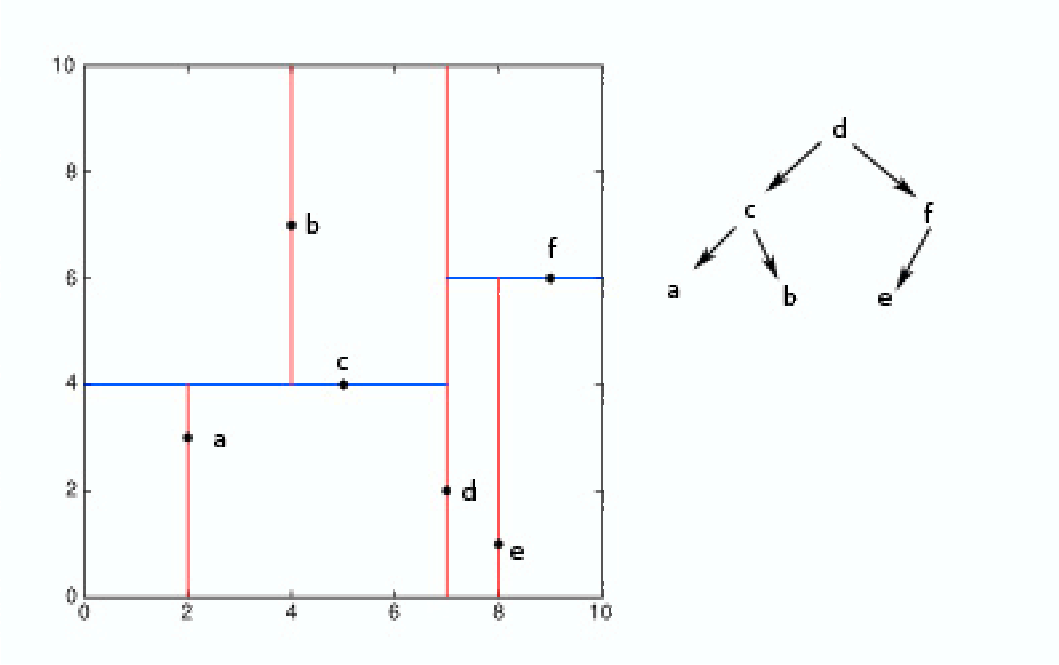

Abbildung 3.4: Spin-Tables: Beim k-d-Baum sind die Unterteilungsebenen an den Achsen ausgerichtet.

$$contact_M(\vec{x}) = \begin{cases} 1, & \text{falls } dist_M(\vec{x}) < \varepsilon \\ 0, & \text{sonst} \end{cases} \tag{3.2}$$

und

$$dist_M(\vec{x}) = \min_{\vec{y} \in M} |\vec{x} - \vec{y}| \tag{3.3}$$

In der Implementierung von Winkelbach wird ein k-d-Baum [36] (vgl. Kapitel C.2) zur binären Raumaufteilung verwendet (siehe Abbildung 3.4), um in logarithmischer Zeit zu einem Punkt aus der Szene den nächsten korrespondierenden Punkt im Modell zu finden. Es steht keine unendliche Anzahl an Punkten zur Verfügung. Deshalb kann für Ω nur eine Näherung angegeben werden. Diese Approximation ergibt sich gemäß [38] als:

$$\Omega \approx \frac{\sum_{i=1}^{n} contact_M(\vec{x}_i)}{n} \pm \frac{1.96}{2\sqrt{n}} \tag{3.4}$$

Wenn die obere Schranke dieses Intervalls schlechter als die Güte der letzten besten Transformation ist, wird die Berechnung der Approximation abgebrochen und mit der Bestimmung der nächsten Transformation angefangen. Um sicherzustellen, dass sich die Fragmente nicht gegenseitig durchdringen, wird die Funktion $contact_M(\vec{x})$ noch etwas erweitert:

$$contact_M(\vec{x}) = \begin{cases} 1, & \text{falls } dist_M(\vec{x}) < \varepsilon \\ -4, & \text{falls } dist_M(\vec{x}) \geq \varepsilon \quad \wedge \quad (\vec{x} - \vec{y}) \cdot \vec{n_y} < 0 \\ 0, & \text{sonst} \end{cases} \tag{3.5}$$

mit $\vec{y} \in M$ stellt den Punkt mit der kleinsten Distanz zu $\vec{x}$ dar, $\vec{n_y}$ ist die Oberflächennormale von $\vec{y}$.

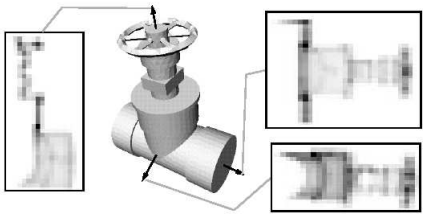

Abbildung 3.5: Spin-Images: Die Spin-Image Darstellung. Die Grauwerte sind proportional zur Häufigkeit der Einträge, entn. aus [20]

3.1.3 Spin-Images, Andrew Edie Johnson, Martial Hebert

Hebert und Johnson [20] haben mit Spin-Images ein Verfahren entwickelt, das es ermöglicht, komplexe Szenen zu registrieren. Die Scans liegen als Dreiecksnetze vor. Die Punktkorrespondenzen werden durch Spin-Images ermittelt und mit den Korrespondenzen eine Transformation berechnet, die die Teilnetze übereinander legt. Anschließend wird zur Feinregistrierung ein ICP-Algorithmus verwendet.

Die Spin-Images sind invariant gegenüber Rotation und Translation. Zur Konstruktion eines Spin-Images, wird zuerst eine sog. Spin-Map entwickelt. Hierzu wird in einem triangulierten Distanzbild ein Referenzknoten gewählt. Alle anderen Knoten innerhalb einer Umgebung werden mathematisch zu diesem Basispunkt in Beziehung gesetzt. Aus den neu berechneten Koordinaten der Spin-Map entwickeln Hebert und Johnson eine Art Histogramm, das Spin-Image, in dem die neuen 2D-Koordinaten in ein 2D-Array eingetragen werden. Um Korrespondenzen zu ermitteln, werden diese Abbildungen miteinander verglichen, Korrelationen ermittelt und die gesuchten Transformationen berechnet.

Spin-Map und Spin-Images

Hebert und Johnson verwenden Dreiecksnetze als Grundlage für die Generierung von Spin-Maps und Spin-Images. Der Hauptkern für die Konstruktion eines Spin-Images ist ein dreidimensionaler Punkt $\vec{p}$, herausgegriffen aus dem gegebenen Dreicksnetz, der durch seine Oberflächenposition (x, y, z) und eine Normale $\vec{n}$ gekennzeichnet ist. Dieser Punkt wird als Bezugspunkt für die Berechnung einer Häufigkeitsverteilung verwendet. Für diese Konstruktion wird eine Tangentialebene P bzgl. der Nachbarknoten im Netz erzeugt und dazu eine Normale $\vec{n}$ berechnet, die durch den Basispunkt geht. Für einen 3D-Punkt $\vec{x}$ auf dem Netz wird die neue 2D-Koordinate mit Hilfe der Spin-Map Funktion (3.6) ermittelt.

$$S(\vec{x}) \rightarrow (\alpha, \beta) = \left(\sqrt{|\vec{x} - \vec{p}|^2 - (\vec{n}\,(\vec{x} - \vec{p}))^2}, \vec{n}(\vec{x} - \vec{p}) \right) \tag{3.6}$$

Für ein genaues Verständnis der Formel (3.6) ist es hilfreich, sich Abbildung 3.8 anzusehen. Der Wert α beschreibt den orthogonalen Abstand zwischen $\vec{x}$ und L, wobei L eine Gerade durch $\vec{p}$ in Richtung der Normalen $\vec{n}$ darstellt; β ist der kürzeste Abstand von $\vec{x}$ zur Ausgleichsebene P. Mit dieser Spin-Map Funktion erhält man eine Abbildung, die rotations- und translationsinvariant ist. Die mathematische Funktion (3.6) erfasst nur die geometrischen Eigenschaften der Oberfläche. Ein Spin-Image beschreibt die Oberflächencharakteristik einer bestimmten Nachbarschaft an Knoten um einen gewählten Basispunkt mit seiner Normalen $\vec{n}$. Ein Spin-Image einer Basis wird erzeugt, indem für alle Punkte in einem Umkreis auf der Oberfläche die Koordinaten (α,β) berechnet und in ein quantisiertes Histogramm eingetragen werden. Die Abzisse beschreibt die α-, die Ordinate die β-Koordinate. Der Grauwert einer Zelle zeigt die Häufigkeit des Auftretens von Punkten an, die im Abstand von (α,β) bzgl. der Basis liegen. Bei der Berechnung der Koordinaten erhält man keine glatten Werte, die exakt die Mitte einer Zelle des Histogramms treffen. Die ermittelten Werte werden entsprechend bilinear auf die Nachbarzellen verteilt und in das Histogramm eingetragen. Da die berechneten Koordinaten einen Wert in der Histogramm-Zelle treffen können, der mehr am Rand der Zelle liegt, werden somit die drei näherliegenden Nachbarzellen ebenfalls inkrementiert, um das Histogramm zu verbessern und Rundungsfehler abzumildern.

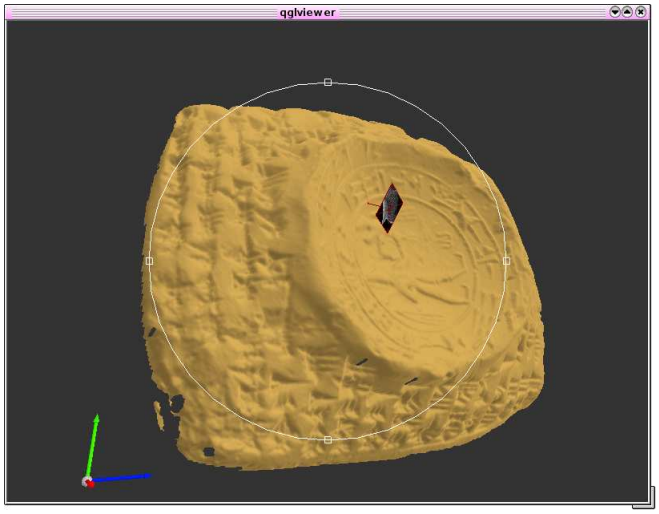

Abbildung 3.6: Spin-Images: Visualisierung des Spin-Images am Beispiel des Siegelabdruckes. Der Basispunkt ist der Schnittpunkt der Oberfläche mit der roten Geraden.

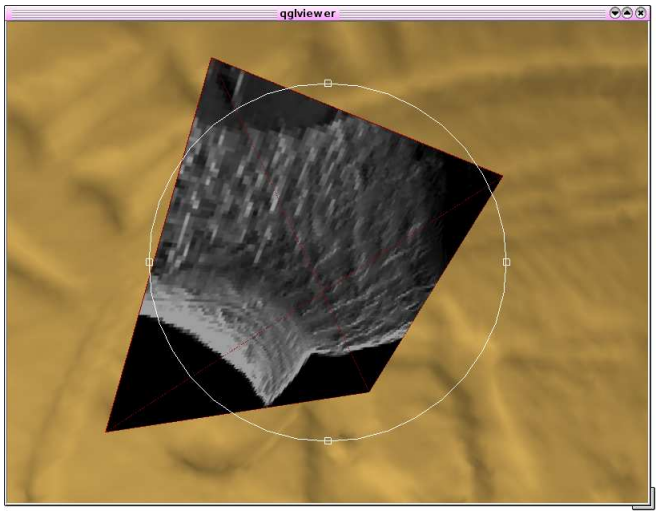

Abbildung 3.7: Spin-Images: Die Nahaufnahme zeigt das Spin-Images sehr deutlich. Im Gegensatz zu Hebert und Johnson ist die Größe eines Spin-Images hier auf 100×100 Bins gesetzt. Die Oberflächenstruktur des Siegelabdruckes ist durch dieses Grauwerthistogramm noch gut zu erkennen.

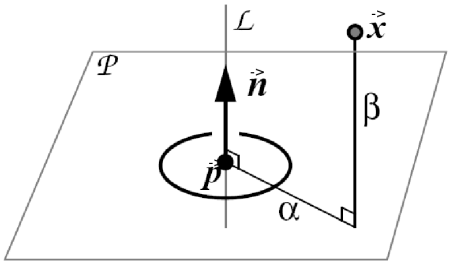

Abbildung 3.8: Spin-Images: Die Bildung der Spin-Map Koordinaten des Punktes $\vec{x}$ bezüglich der Basis $(\vec{p}, \vec{n})$, entn. aus [20]

Die Eigenschaft, die die Spin-Images zur Registrierung haben, ist die, dass es für jeden gewählten Basispunkt eine spezielle Spin-Map gibt. Das zugehörige Spin-Image beschreibt die Oberfläche genau, denn in ihm ist die Form durch die neu berechneten 2D-Koordinaten enthalten, und diese Darstellung ist ebenfalls invariant bzgl. Rotation und Translation. Vergleiche zwischen Spin-Images ermöglichen es, herauszufinden, inwieweit gegebene Distanzbilder gemeinsame Oberflächenstrukturen besitzen. Nach Johnson und Hebert werden die Spin-Images durch 2D-Arrays mit Fließkommazahlen repräsentiert, die durch Bildkorrelation verglichen werden können. Beim Vergleich der Spin-Images muss allerdings noch die Auflösung berücksichtigt werden. Für zwei korrespondierende Punkte zweier Distanzbilder, die ein Objekt aus unterschiedlichen Blickwinkeln betrachten, muss die Auflösung zugehöriger Spin-Images identisch sein. Oftmals sind Auflösungen von Distanzbildern, die nur von einem Scanner geliefert werden, gleich. Falls die Punktdichte unterschiedlich sein sollte, wird ein Netz-Simplifizierungs-Algorithmus benötigt, um Punkte zu entfernen.

Punktkorrespondenzen

Die Scanaufnahmen desselben Objektes aus unterschiedlichen Blickwinkeln bewirken, dass sich deren Spin-Images sehr ähneln, jedoch nicht identisch sind. Ein Spin-Image beschreibt dabei die Oberflächencharakteristik. Der lineare normalisierte Korrelationskoeffizient ist dazu geeignet, Korrespondenzen nach Wertigkeit aufzulisten. Die Korrelation gibt die Möglichkeit, die Punktkorrespondenzen zu ordnen, und dadurch falsche Korrespondenzen auszuschließen und zu verwerfen. Bei der Korrelation zweier Spin-Images werden nur die Zellen verglichen, in denen beide Bilder Daten besitzen. Dadurch findet ein Vergleich zweier Spin-Images nur in überlappenden Bereichen statt. Beim Vergleich zweier Bilder verwenden die Autoren eine Ähnlichkeitsmessung, dessen Ergebnis wiederum in ein Histogramm eingetragen wird. Für diesen Vergleich zwischen zwei Scans P und Q wird der normalisierte lineare Korrelationskoeffizient R berechnet. Ein hoher Wert zeigt eine hohe Korrelation und einen großen Überlappungsbereich an.

$$C(P,Q) = \left(\tanh^{-1}(R(P,Q))\right)^2 - \lambda\left(\frac{1}{N-3}\right) \tag{3.7}$$

Die Verwendung des $\tanh^{-1}$ ist nur ein statistisches Hilfsmittel, um die berechneten Werte von R besser aufzubereiten. Dabei lautet die Varianz $1/(N-3)$, wobei N die Anzahl der überlappenden Zellen darstellt. Die Gewichtung λ wird meistens auf 3 gesetzt.

Globalregistrierung

Durch die Funktion $C(P,Q)$ (vgl. Formel (3.7)) ist es möglich für Korrespondenzen Ränge aufzustellen. Für die Konstruktion eines Histogramms, das die Ähnlichkeiten zwischen zwei gescannten Bildern auf-

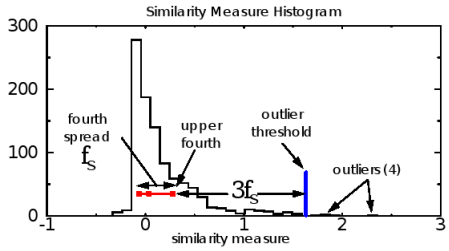

Abbildung 3.9: Spin-Images: Histogramm und resultierende Ergebnisse der Ähnlichkeitsfunktion, entnommen aus [20]. Die Ausreißergrenze liegt bei $4f_S$.

zeigt, gehen Hebert und Johnson wie folgt vor: Zunächt berechnet man für das erste triangulierte Netz für alle Knoten deren Spin-Images. Aus dem zweiten Netz wird ein zufälliger Punkt ausgewählt und ebenfalls sein Spin-Image B erzeugt. Danach folgt ein Vergleich zwischen Spin-Image B und allen anderen vorher berechneten Spin-Images vom ersten Netz (vgl. Abbildung 3.9). Die Korrelation wird für jedes Spin-Image-Paar berechnet und in ein Histogramm eingetragen. Zuletzt werden ca. 1/10 der Punkte aus dem zweiten Netz zufällig ausgewählt und für die Histogramm-Konstruktion verwendet.

Um anschließend gute Korrespondenzen zu ermitteln werden die Ausreißer im Histogramm ermittelt. Dazu gibt es ein statistisches Standardverfahren, bei dem man eine Ausreißergrenze auf der Abzisse im Diagramm festlegt. Die Grenze entsteht, indem man zuerst die Breite auf der Abzisse bestimmt, in der sich drei Viertel aller Werte im Histogramm befinden, und danach die Ausreißergrenze auf das Vierfache dieses Wertes im Histogramm festlegt (vergleiche Abbildung 3.9). Die Bestimmung von Ausreißern ist wichtig, um charakteristische Merkmale in den Spin-Images zu finden. Wenn keine Ausreißer existieren, besitzt das Spin-Image Ähnlichkeiten zu allen Spin-Images aus Netz A. Falls viele Ausreißer zu finden sind gibt es wiederum viele Spin-Images des Netzes B, die Ähnlichkeiten zu einem Spin-Image von A aufweisen.

Gruppierung von Korrespondenzen

Für die Berechnung einer geeigneten Transformation zweier Distanzbilder S und M, ist eine Gruppierung der Korrespondenzen hilfreich, um geometrische Unterschiede bzw. Gemeinsamkeiten zu entschlüsseln. Zum einen ist es wichtig Korrespondenzpaare herauszufiltern, die in Bereichen vorkommen, wo keine Überlappung herrscht. Zum anderen besitzen Korrespondenzpaare zwischen zwei Basispunkten eine geometrische Übereinstimmung, wenn ihre berechneten Spin-Map-Koordinaten innerhalb eines festgelegten Grenzbereiches liegen.

Diese geometrische Konsistenz kann man mit Hilfe der Formel (3.8) überprüfen. Durch Distanzbestimmung wird festgelegt, ob es sich um korrekte korrespondierende Punktpaare $[\vec{s}, \vec{m}]$ handelt oder nicht, und ob man sie behalten oder verwerfen muss. Dazu sind zwei Korrespondenzpaare $[\vec{s_1}, \vec{m_1}]$ und $[\vec{s_2}, \vec{m_2}]$ gegeben. Durch Einsetzen in die Spin-Map-Funktion (3.6) und anschließender Distanzbestimmung mit (3.8), kann man überprüfen, ob die Korrespondenzpaare beide innerhalb eines Distanz-Schwellwertes D liegen oder nicht.

$$|S_{m_1}(\vec{m_2}) - S_{s_1}(\vec{s_2})| < D \text{ und } |S_{m2}(\vec{m_1}) - S_{s2}(\vec{s_1})| < D \tag{3.8}$$

Nachdem die Korrespondenzpaare festgelegt sind, wird eine Transformation berechnet. Um Rechenzeit zu sparen, werden die Korrespondenzen nach Ähnlichkeiten und geometrischer Konsistenz gruppiert und in disjunkte Mengen eingeteilt. Die Teilmenge, die hohe Ähnlichkeitswerte aufweist, wird zur Transformationsberechnung verwendet.

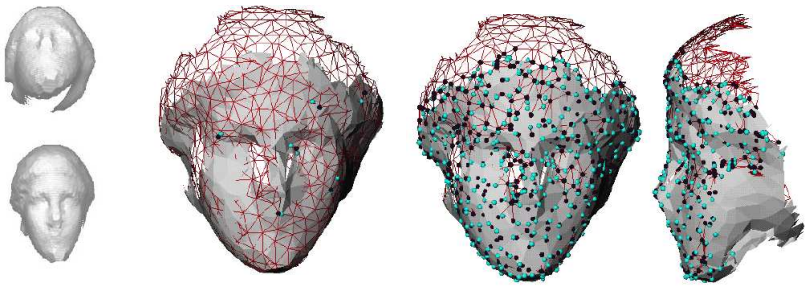

Abbildung 3.10: Spin-Images: Die ICP Variante zur Validierung gefundener Korrespondenzen, links das Modell und die beiden Ansichten, die gescannt wurden, rechts die überlagerten Dreiecksnetze. Die dunklen Punkte kennzeichnen die gefundenen Korrespondenzen, von ihnen werden umliegende, durch eine Kante verbundene, Punkte auf Korrespondenz getestet. Werden weitere Korrespondenzen gefunden, erkennt man sie grafisch durch die dunkle Darstellung, entnommen aus [20].

Verifikation

Hebert und Johnson verwenden zur Verifikation, um Transformationen zu validieren, eine Variante des ICP-Algorithmus' von Besl und McKay [32]. Als Modell werden im Folgendem die bereits übereinander gelegten Netze bezeichnet und als Szene das neue Netz, das man entsprechend verschmelzen möchte. Der ICP-Algorithmus wird hier mit der vorher ermittelten Transformation der Punktkorrespondenzen initiiert. Mit der gefundenen Transformation wird der Satz an Punktkorrespondenzen transformiert, so dass sich die Scans entsprechend überlappen. Weitere Korrespondenzen werden gefunden, indem man von den bestehenden Korrespondenzen ausgeht und mittels des Bereichswachstumsverfahrens den nächsten Punkt (aus der Szene) heraussucht. Dann wird überprüft, ob der Abstand zwischen diesem Szene-Punkt und einem nächsten Punkt aus dem Modell möglichst minimal ist. Dazu wird eine bestimmter Schwellwert vorher festgelegt (z. B. zwei mal Auflösung des Netzes). Falls die Bedingung erfüllt ist, wird auch diese Punktpaarung in eine Korrespondenzliste aufgenommen und weiter mit dem nächsten Punkt mittels des Bereichswachstumsverfahrens, bis keine Korrespondenzen mehr gefunden werden. Für die Verifikation betrachtet man die Anzahl der gefundenen Korrespondenzen. Die Anzahl der gefundenen Paare ist ein Indiz dafür, ob zwei Ansichten sich gut oder schlecht überlappen.

Ergebnis

Das Verfahren nach Johnson et al. liefert eine gute Technik, um Überlappungsbereiche festzustellen. Dies wird ebenfalls im Registrierungsverfahren nach Lucchese et al. ausgenutzt (vgl. Kapitel 3.1.5), indem die überlappenden Bereiche zuerst durch das Spin-Image Verfahren ermittelt werden und anschließend dem Grobregistrierungsverfahren im Frequenzbereich als Grundlage dienen.

3.1.4 Range image registration driven by a hierarchy of surface differential features, P. Krsek

Das Verfahren von Krsek et al. [22] benutzt zum Erzeugen einer Transformation rotationsinvariante Eigenschaften der Oberflächen. In Szene und Modell werden markante Punkte gesucht, welche in beiden Mengen ähnliche Merkmale haben. Mit Hilfe dieser werden mögliche starre Transformationen erzeugt und bewertet.

Vorgehen

Als Merkmal, an dem sich die Registrierung orientiert, nimmt dieses Verfahren die Oberflächenkrümmung. Daher wird zuerst für alle Punkte in Szene S und Modell M deren mittlere Oberflächenkrümmung be-

Abbildung 3.11: Verfahren nach Krsek: Höhenlinien auf den Entfernungsdaten

rechnet [22]. Die einzelnen Krümmungswerte der Punkte ziehen sich wie Höhenlinien durch die Entfernungsdaten der Scans, siehe Abbildung 3.1.4. Hieraus werden die Kurven extrahiert, welche die Punkte mit einer Krümmung von 0 miteinander verbinden. Die Kurven können offen oder geschlossen sein. Aus diesen Kurven wiederum extrahiert man die Strecken (mit Anfangs- und Endpunkt), welche die kürzeste Verbindung zu den benachbarten Kurven darstellen. Diese Strecken werden der Länge nach sortiert in einem Array gespeichert, sowohl für die Szene als auch für das Modell. Anschließend werden jeweils zwei Linienzüge miteinander verglichen. Um eine mögliche Transformation T_c zu erhalten, werden Linienpaare aufeinander verschoben und rotiert.

Zur Überprüfung von T_c werden alle anderen Strecken transformiert und deren Abstand zum Modell gemessen. Ist das Ergebnis ausreichend wird der ICRP-Algorithmus (*Iterative-Closest-Reciprocal-Point*) zuerst auf die Strecken und zum Schluss noch einmal auf die gesamte Punktmenge angewendet, welche sich dann in einer günstigen Lage befindet.

3.1.5 A Frequency Domain Technique for Range Data Registration, Lucchese et al.

Dieses Kapitel handelt von der Lösung des Problems eine geeignete Transformation zu finden, indem die Registrierung im Frequenzbereich behandelt wird. Die Überführung der ggf. texturierten Scandaten in den Frequenzbereich, ermöglicht es, die gesuchten Größen Rotationsmatrix und Translationsvektor getrennt zu berechnen. Lucchese et al. [24] verwenden hierfür eine Technik, in der sie zuerst die Rotation ermitteln. Dabei wird die Rotationsachsenberechnung von der Berechnung des Rotationswinkels entkoppelt. Anschließend wird die Translation geschätzt, was mit einer Standard-Phasenkorrelationstechnik durchgeführt wird. Die Fourier-Registrierung liefert robuste Abschätzungen und eine gute Vorberechnung für den Iterative-Closest-Point-Algoritmus (ICP, siehe Kapitel 3.2). Dieses Verfahren ist ebenfalls für texturierte Objekte geeignet. Mit Lucchese et al. werden aus einem Problem mit sechs gesuchten Parametern der Rototranslation, zwei Probleme, indem Rotation $\mathbf{R}$ und Translation $\vec{t}$ getrennt werden.

Diskrete Fouriertransformation

Eine der wichtigsten Bildtransformationen ist neben der Wavelet- und der Kosinustransformation die Fouriertransformation. Diese mathematische Methode zerlegt ein Signal in eine Summe von Sinusschwin-

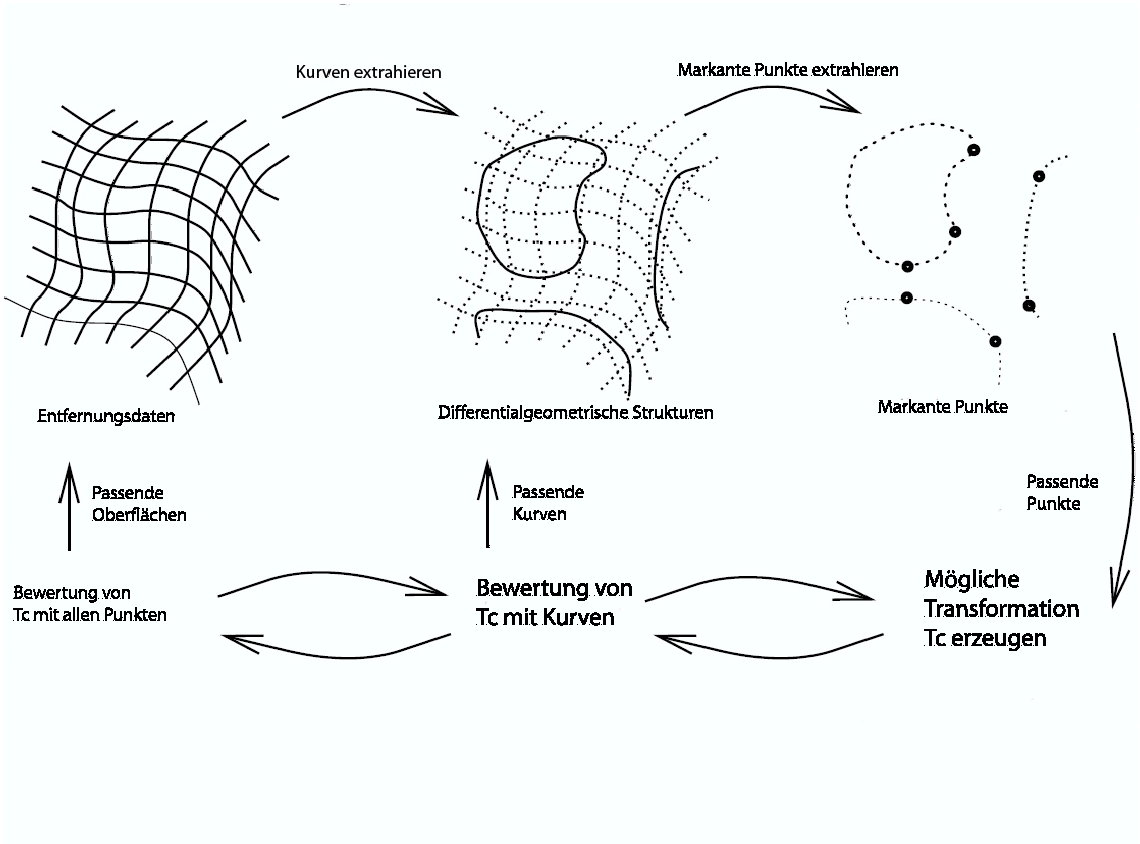

Abbildung 3.12: Verfahren nach Krsek: Schematisches Vorgehen

gungen unterschiedlicher Phase und Amplitude. Erhält man aus einem kontinuierlichen Signal durch Abtastung ein diskretes Signal $s(n)$, so wird das Spektrum $S(k)$ eines zeitdiskreten Signals $s(n)$ definiert als:

$$S(k) = \sum_{n=0}^{M-1} s(n) e^{-j2\pi nk/M} \qquad k = 0, ..., M-1 \tag{3.9}$$

und man erhält ein frequenzdiskretes Spektrum. Dabei wurde berücksichtigt, dass das zeitdiskrete Signal auf M diskrete Werte beschränkt ist.

Die inverse Fouriertransformation lässt sich ebenfalls berechnen:

$$s(n) = \frac{1}{M} \sum_{k=0}^{M-1} S(k) e^{-j2\pi nk/M} \qquad n = 0, ..., M-1 \tag{3.10}$$

Die diskrete Fouriertransformation erlaubt es vor allem schnelle Algorithmen zur Berechnung der Spektren zu entwickeln.

Vorüberlegungen im Fourierbereich

Die Autoren Lucchese et al. benötigen für ihre Berechnung die gemeinsamen Überlappungen der Scanbereiche. Hat man Datensätze vorliegen, die Teilüberlappungen aufweisen, kann man manuell oder durch in dieser Ausarbeitung vorgestellte Techniken (z. B. Spin-Images, siehe Kapitel 3.1.3) diese Überlappungen

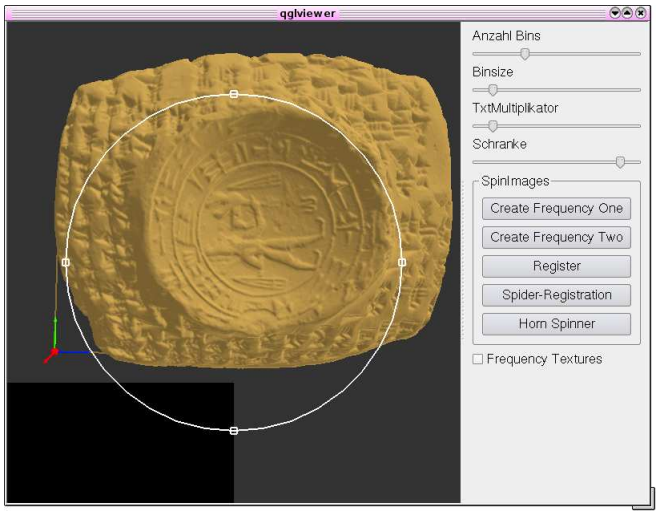

Abbildung 3.13: DFT: Darstellung des Siegelabdruckes.

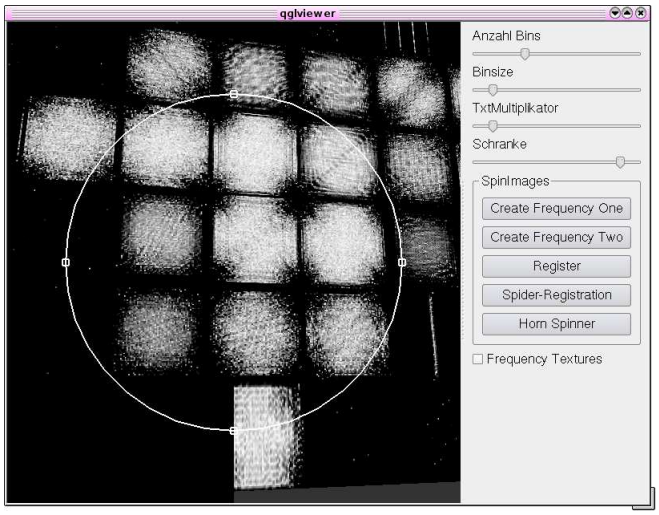

Abbildung 3.14: DFT: Abschnittsweise Anwendung der diskreten Fouriertransformation auf den Siegelabdruck und visuelle Darstellung des Ergebnisses.

bestimmen. Diese Bereiche seien im Folgenden mit $s_1(\vec{x})$ und $s_2(\vec{x})$ bezeichnet. Damit sich diese Scanbilder durch geeignete Wahl von $\boldsymbol{R}$ und $\vec{t}$ überdecken, muss ein Datensatz so transformiert werden, dass folgende Gleichung erfüllt ist:

$$s_2(\vec{x}) = s_1(\boldsymbol{R}^{-1}\vec{x} - \vec{t}) \tag{3.11}$$

Durch Rotation $\boldsymbol{R}$ und Translation $\vec{t}$ werden alle Punkte des ersten Datensatzes $s_1(\vec{x})$ zuerst rotiert und dann um $\vec{t}$ verschoben, so dass beide Bereiche zur Deckung gebracht werden. Um mathematische Schwierigkeiten mit einzelnen Abtastspitzen zu vermeiden, werden die Abtastpunkte durch eine kontinuierliche Funktion ersetzt. Hierbei wird eine Tiefpassfilterung mittels einer Gaußfunktion durchgeführt. Die Spitzen werden durch die Gaußkurve geglättet. Mathematisch erfolgt dieses durch eine Faltung.

$$l_n(\vec{x}) = \int s_n(\vec{x} - \vec{\xi})h(\vec{\xi})d\xi, \quad n = 1, 2 \tag{3.12}$$

Bei der Überführung in den Frequenzbereich unter Ausnutzung des Verschiebungssatzes, erkennt man, dass durch das Betragsspektrum von $L_n(\vec{k})$ $(n = 1, 2)$ die Rotation $\boldsymbol{R}$ berechnet werden kann, ohne gleichzeitig $\vec{t}$ ermitteln zu müssen. Mit dieser mathematischen Umformung fällt der Phasenterm, in dem $\vec{t}$ enthalten ist, in der weiteren Betrachtung vorerst weg. Zunächst lassen sich die beiden Gleichungen (3.11) und (3.12) im Frequenzbereich zusammenfassen zu:

$$L_2(\vec{k}) = L_1(\boldsymbol{R}^{-1}\vec{k})e^{-j2\pi\vec{k}^T\boldsymbol{R}\vec{t}} \tag{3.13}$$

Der Unterschied in den Betragsspektren der beiden Funktionen, die sich nur durch ihre Position und Lage im Raum unterscheiden, zeigt sich nur in dem Term der Rotation $\boldsymbol{R}$.

$$|L_2(\vec{k})| = |L_1(\boldsymbol{R}^{-1}\vec{k})| \tag{3.14}$$

Um die Rotation $\boldsymbol{R}$ ermitteln zu können, wird eine Differenzfunktion $\Delta(\vec{k})$ eingeführt. Durch Subtraktion der Betragsspektren und vorherige Normierung durch den Gleichanteil wird die Differenz $\Delta(\vec{k})$ $(\vec{k} = [k_x, k_y, k_z]^T)$ ermittelt. Besonders für texturierte Daten ist eine Normierung unerlässlich, da so Lichtintensitäten ausgeglichen werden können.

$$\Delta(\vec{k}) = \left| \frac{\left|L_1(\vec{k})\right|}{L_1(\vec{0})} - \frac{\left|L_2(\vec{k})\right|}{L_2(\vec{0})} \right| \tag{3.15}$$

Durch Einsetzen von Gleichung (3.14), in der der notwendige Rotationsparameter $\boldsymbol{R}$ für die Überlappung ersichtlich wird, erhält man folgende Formel:

$$\Delta(\vec{k}) = \left| \frac{|L_1(\vec{k})|}{L_1(\vec{0})} - \frac{\left|L_1(\boldsymbol{R}^{-1}\vec{k})\right|}{L_1(\vec{0})} \right| \tag{3.16}$$

Die Differenz $\Delta(\vec{k})$ ist ein Hilfsmittel, mit dem man die Rotationsachse ω berechnen bzw. abschätzen kann.

Rotationsachsenberechnung

Die Rotationen werden in Eulerdarstellung mathematisch durch eine schiefsymmetrische Drehmatrix $\boldsymbol{\Omega}$ und einen Winkel φ dargestellt. Diese mathematische Formel ist für die Betrachtung im Fourierbereich nützlich, denn daraus wird deutlich, dass es für die Bestimmung der Rotation hilfreich ist zuerst die Richtung der Rotationsachse $\vec{\omega}$ zu ermitteln. Durch diese Darstellung kann man $\vec{\omega}$ herauslösen und gleichzeitig die Richtung der einfachen Rotation (in der Euler-Darstellung) um eine Achse durch den Vektor $\vec{\omega} = [w_x, w_y, w_z]^T$ mit $|\vec{\omega}| = 1$ bestimmen. Die Umformung der Rotation $\boldsymbol{R}$ in die Euler-Darstellung hat folgende Gestalt:

$$\boldsymbol{R} = e^{\boldsymbol{\Omega}\varphi} \quad \text{mit} \quad \boldsymbol{\Omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{3.17}$$

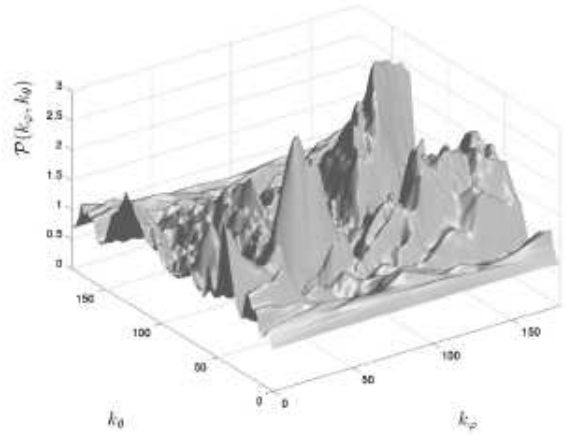

Abbildung 3.15: Grobregistrierung im Frequenzbereich: Die Grafik zeigt die Darstellung der Funktion $P(k_\varphi, k_\theta)$. Für eine Minimumsuche zeigt das zugehörige Höhenliniendiagramm (siehe Abbildung 3.16) allerdings eine anschaulichere Grafik. Entnommen aus [24]

Dabei fließen die Komponenten von $\vec{\omega}$ in die schiefsymmetrische Drehmatrix $\boldsymbol{\Omega}$ ein. Wenn $\Delta(\vec{k})$ (vgl. Gleichung (3.16)) gleich null ist, gilt $\boldsymbol{R}^{-1}\vec{k} = \vec{k}$ bzw. $\boldsymbol{R}\vec{k} = \vec{k}$, d. h. $\vec{k}$ ist Eigenvektor der Matrix $\boldsymbol{R}$. Ferner kann gezeigt werden, dass $\boldsymbol{R}$ nur einen reellen Eigenwert hat und der zu ihm gehörende Eigenvektor ein Vielfaches von $\vec{\omega}$ ist.

Für die weitere Berechnung von $\vec{\omega}$ überführen Lucchese et al. die kartesischen Koordinaten (k_x, k_y, k_z) in ein neues Koordinatensystem, in Polarkoordinaten. Mit diesem Standardverfahren erhält man den Vektor $\vec{k}$ aufgesplittet in Betrag und Winkeldarstellung. Dieses Vorgehen ist für die Abschätzung der Richtung der Rotationsachse $\vec{\omega}$ geeignet. Die Richtung von $\vec{\omega}$ ergibt sich, indem man $\Delta(\vec{k}) = 0$ bestimmt, für eine Funktion $\Delta(\vec{k})_{polar}$, die $\Delta(\vec{k})$ in Polarkoordinaten darstellt. Die Länge des Lösungsvektors spielt keine Rolle, nur seine Richtung. Dies ist die Richtung der Rotationsachse.

$$\text{Betrag } k = (k_x^2 + k_y^2 + k_z^2)^{1/2}, \quad k \geq 0 \tag{3.18}$$

$$\text{Winkel } k_\varphi = \arctan(\frac{k_y}{k_x}), \quad 0 \leq k\varphi < 2\pi \tag{3.19}$$

$$\text{Winkel } k_\theta = \arccos\left(\frac{k_x}{\sqrt{(k_x^2 + k_y^2 + k_z^2)}}\right), \quad 0 \leq k_\theta < \pi/2 \tag{3.20}$$

Ziel ist es $\Delta(\vec{k})_{polar} = 0$ einfach berechnen zu können. Daher wird der Suchraum um eine Dimension verkleinert und eine Radialprojektion durchgeführt. Dafür wird das Integral über $\Delta(k, k_\varphi, k_\theta)$ über dem Betrag gebildet.

$$P(k_\varphi, k_\theta) = \int_0^\infty \Delta(k, k_\varphi, k_\theta)dk \tag{3.21}$$

Wegen $\Delta(\vec{k})_{polar} = 0 \Longleftrightarrow P(k\varphi, k_\theta) = 0$ und da $P(k\varphi, k_\theta) \geq 0 \; \forall k_\varphi, k_\theta$ kann man die Richtung von ω mit

$$(\hat{\varphi}, \hat{\theta}) = \arg\min P(k_\varphi, k_\theta) \tag{3.22}$$

ermitteln. $(\hat{\varphi}, \hat{\theta})$ muss nur noch aus dem polaren in das kartesische Koordinatensystem überführt werden. Aufgrund von Rechnungsungenauigkeiten wird keine Nullstellensuche durchgeführt, sondern das Argument des Minimums verwendet. Durch die ermittelten Winkel $\hat{k}_\varphi$ und $\hat{k}_\theta$ ist die Richtung von ω gegeben.

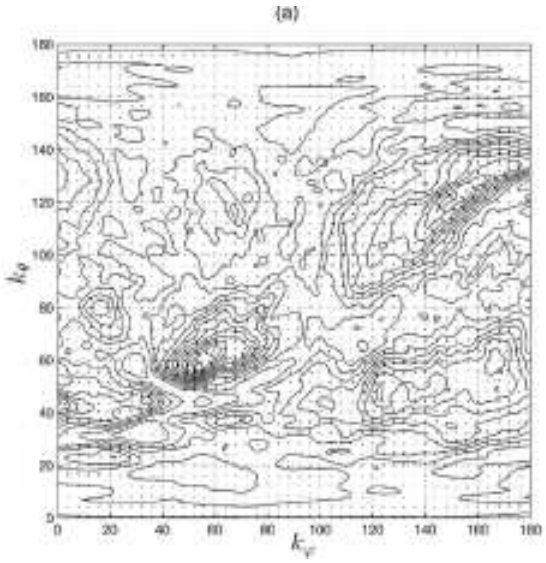

Abbildung 3.16: Grobregistrierung im Frequenzbereich: Das Höhenliniendiagramm der Funktion $P(k_\varphi, k_\theta)$, auf der Abzisse ist k_φ aufgetragen, auf der Ordinate k_θ, entn. aus [24]

Rotationswinkelberechnung

Mit der Abschätzung der Rotationsachse $\vec{\omega}$ erhält man eine weitere Grundlage, um den Rotationswinkel zu bestimmen. Dazu ist die Überführung in ein neues kartesisches Koordinatensystem über $\vec{u}$ notwendig, dessen $\vec{\omega}$-Achse in Richtung von $\vec{\omega}$ zeigt.

$$\vec{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = C^T \begin{bmatrix} k_x \\ k_y \\ k_z \end{bmatrix} \tag{3.23}$$

Durch Einsetzen von der Umformungsmatrix $\vec{C}$ in die Gleichung

$$|L_2(\vec{k})| = |L_1(\boldsymbol{R}^{-1}\vec{k})| \tag{3.24}$$

und anschließender Umformung der Rotationsmatrix erhält man:

$$|\hat{L}_2(\vec{u})| = |\hat{L}_1(C^{-1}\boldsymbol{R}^{-1}C\vec{u}| = |\hat{L}_1(\boldsymbol{R}_\omega^{-1}(\phi)\vec{u})| \tag{3.25}$$

Zur Vereinfachung des Problems (durch Dimensionsreduktion) wird entlang der $\vec{\omega}$-Achse projiziert:

$$p_n(u,v) = \int \left|\hat{L}_n(u,v,w)\right| dw, \quad n = 1,2 \tag{3.26}$$

Eine weitergehende 1D-Radialprojektion hat folgende Gestalt:

$$f_n(\mu) = \int_0^\infty \hat{p}_n(r,\mu)dr, \quad n = 1,2 \tag{3.27}$$

$\hat{p}_n(r,\mu)$ ist dabei $p_n(u,v)$ in Polarkoordinaten. Wenn man beide Scanbilder in Beziehung setzt, erkennt man, dass die beiden berechnete Projektionen um eine Phase verschoben sind. Um diese Verschiebung herauszulösen kann eine Standard-Methode aus der Statistik angewendet werden: Korrelation bzw. Phasenkorrelation. Möchte man die Ähnlichkeit zweier Signale bestimmen, so kann dazu im Ortsbereich ein Korrelationswert berechnet werden. Die Korrelation in der Signalverarbeitung wird als Maßstab betrachtet, wie ähnlich zwei Signale sind. Im Fourierbereich erhält man daraus das Kreuzleistungsdichtespektrum. Wird das Kreuzleistungsdichtespektrum mit dem Amplitudenbetrag beider Funktionen normiert, so erhält man hier eine Exponentialfunktion.

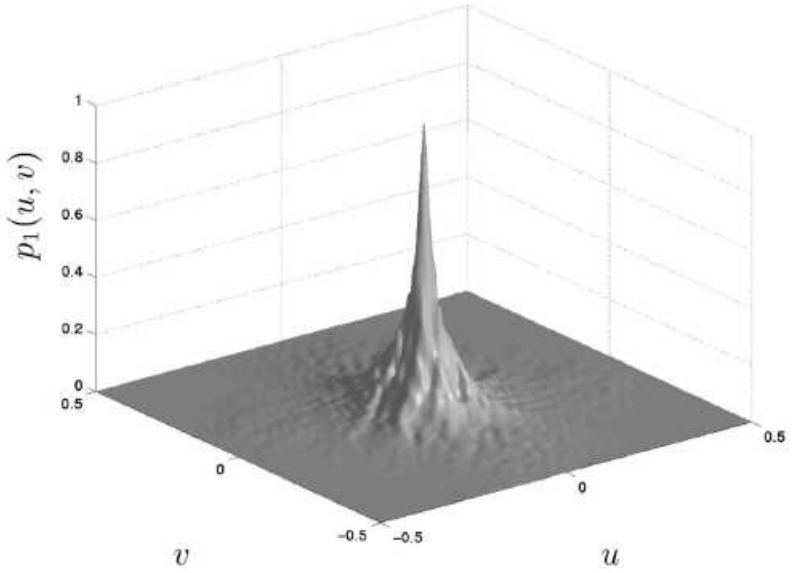

Abbildung 3.17: Grobregistrierung im Frequenzbereich: Die Grafik zeigt die 3D Darstellung der Axialprojektion am Beispiel von $p_1(u, v)$, entn. aus [24]

$$Q(k_\mu) = \frac{F_1^*(k)F_2(k)}{|F_1(k)F_2(k)|} = e^{-j2\pi k_\mu \phi} \tag{3.28}$$

Der Term $F_1^*(k)$ bezeichnet die Konjugiertkomplexe von $F_1(k)$. Die Rücktransfomation zur Kreuzkorrelationsfunktion ergibt einen Delta-Impuls der Höhe eins, der um die gesuchte Phase ϕ verschoben ist: $\delta(\mu - \varphi) = q(\mu)$. Allerdings resultiert das gewonnene Ergebnis in der Praxis nicht immer in einem Spitzenwert, sondern es finden sich mehrere Spitzen in der Gleichung wieder.

Berechnung der Transformation $\vec{t}$

Für den Rotationswinkel erhält man zwei mögliche Kandidaten und damit zwei mögliche Rotationsmatrizen. Ferner kann gezeigt werden, dass eine der beiden keine Lösung darstellt. Zunächst definiert man ein neues Signal:

$$d(\vec{x}) = l_2(\hat{\boldsymbol{R}}_2\vec{x}) = l_1(\vec{x} - \vec{t}) \tag{3.29}$$

Ähnlich zum vorangehenden Schritt, kann man auch bzgl. der Lösung von $\vec{t}$ verfahren. Man bildet das normalisierte Kreuzleistungsdichtespektrum im Frequenzbereich. Die Rücktransformation zum Ergebnis der Kreuzkorrelationsfunktion ergibt wiederum einen Delta-Impuls der Höhe eins, verschoben um den gesuchten Vektor $\vec{t}$.

$$\Phi(\vec{k}) = \frac{L_1^*(\vec{k})D_1(\vec{k})}{|L_1(\vec{k})D_1(\vec{k})|} = e^{-j2\pi\vec{k}^T\vec{t}} \tag{3.30}$$

Um Berechnungen im Dreidimensionalen zu vermeiden, wird die Berechnung auf drei eindimensionale Funktionen gesplittet.

$$\varsigma_{l1}^x = \int_{\Re^2} l_1(\vec{x})dydz, \varsigma_{l1}^y = \int_{\Re^2} l_1(\vec{x})dxdz, \varsigma_{l1}^z = \int_{\Re^2} l_1(\vec{x})dxdy \tag{3.31}$$

Ebenfalls ist die Verschiebung in der Gleichung $\varsigma_{d1}^n(n) = \varsigma_{l1}^n(n - t_n)$ $n = x, y, z$ ersichtlich (ς_{d1}^n wird dabei analog zu ς_{l1}^n definiert). Mit dem letzten Schritt hat man alle sechs notwendigen Parameter für die Transformation gefunden.

Ergebnis

Lucchese et al. konstruieren in ihrem Artikel [24] einen neuen Ansatz für die Lösung des Problems der Grobregistrierung. Im Gegensatz zu anderen Verfahren setzen sie allerdings auch schon vorraus, dass überlappende Bereiche bereits bekannt sind. Die Autoren weisen ebenfalls darauf hin, wie wichtig die genaue Bestimmung der Rotationsachse für die weiteren Berechnungsschritte ist. Der Ansatz der Autoren klingt sehr vielversprechend, und es wird sich in der praktischen Implementierung zeigen, wie effizient er wirklich ist.

3.1.6 Mutual Information Registration, Josien P.W. Pluim et al.

Das Verfahren orientiert sich an der Zusammenfassung über „Mutual Information Registration“ von Josien P. W. Pluim, J. B. Antoine Maintz und Max A. Viergever. Dabei vergleichen die Autoren verschiedene Lösungsansätze für eine Registrierung via wechselseitiger Informationen miteinander. Es wurden zwei verschiedene Varianten umgesetzt.

Ansatz

Das Grobregistrierungsverfahren „Mutual Information Registration“ (vgl. [27]) verwendet wechselseitige Informationen um eine mögliche Transformation zwischen zwei Scans zu errechnen. Dabei funktioniert die eigentliche Berechnung mit parametrisierten Scans (vgl. Kapitel 2.1). Die Parametrisierung eines Scans liefert ein Graustufenbild, wobei die Grauwerte die jeweilige Krümmung im ursprünglichen Scan visualisieren. Sind beide Bilder parametrisiert, startet man mit den Graustufenbildern A und B die Registrierung. Ist der Wert der wechselseitigen Informationen maximiert, sollten die Bilder registriert sein. Danach findet mit den Werten der 2D-Bildtransformation eine Rückrechnung in eine 3D-Transformation statt. Dabei genügt es pro Scan jeweils 3 Punkte zu finden.

Für die wechselseitigen Informationen gibt es zwei Varianten, zwischen denen der Anwender wählen kann. Zum einen steht die klassische Definition für wechselseitige Informationen zur Verfügung, es gilt:

$$I(A,B) = H(A) + H(B) - H(A,B) \tag{3.32}$$

Sind die Ergebnisse nicht zufriedenstellend, kann die normalisierte Variante benutzt werden:

$$I(A,B) = \frac{H(A) + H(B)}{H(A,B)} \tag{3.33}$$

Bei beiden Formeln ist $H(x)$ der Entropiewert eines Bildes. Es gilt:

$$H(x) = \sum_i \; p_i \; log \frac{1}{p_i} \tag{3.34}$$

Dabei ist i der i-te Grauwert des Bildes x. Die Wahrscheinlichkeit eines Grauwertes ist:

$$p_i = \frac{Anzahl\ der\ Pixel\ mit\ Grauwert\ i}{Gesamtzahl\ der\ Pixel} \tag{3.35}$$

Der Entropiewert ist ein Maß für den Informationsgehalt des Bildes. Er wird errechnet über die Histogrammverteilung der Grauwerte. Man setzt dabei die Häufigkeit eines Grauwerts in Bezug zur Anzahl der Bildpunkte und erhält so die Wahrscheinlichkeit für diesen Grauwert.

Der gemeinsame Entropiewert zweier Bilder A und B wird definiert als

$$H(A,B) = \sum_{i,j} \; p_{i,j} \; log \frac{1}{p_{i,j}} \tag{3.36}$$

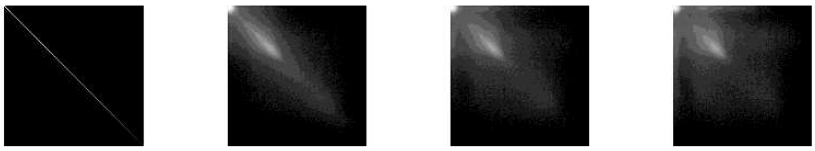

Abbildung 3.18: Grobregistrierung mit wechselseitigen Informationen: Verbundhistogramm eines Bildes mit sich selbst, ganz links: Bilder registriert, links: ein Bild rotiert um 2°, rechts: rotiert um 5°, ganz rechts: rotiert um 10°

Dabei sind i Grauwerte aus Bild A und j Grauwerte aus Bild B. Nun werden alle Grauwerte aus A mit allen Grauwerten aus B kombiniert und Grauwertpaare der Form (i, j) gebildet. Dazu betrachtet man einen Bildpunkt in Bild A und den analogen Bildpunkt in Bild B und erhöht die Häufigkeit des zugehörigen Grauwertpaars (i, j) um eins. Es werden die Häufigkeiten aller möglichen Grauwertpaare im Bezug zur Gesamtzahl der Bildpunkte als Wahrscheinlichkeiten aufgefasst. Informationstheoretisch drückt dieser Wert die Informationen, die das eine Bild über das andere Bild enthält, aus. Die Grauwertpaare lassen sich in einem gemeinsamen Histogramm einzeichnen. Ein Grauwertpaar (A_i, B_j), mit A_i: Grauwert i aus Bild A, B_j: Grauwert j aus Bild B im Verbundhistogramm an Position (i, j) eingezeichnet wird. Der Helligkeitswert an dieser Stelle wird durch die Häufigkeit in Relation zur Gesamtzahl an Pixeln bestimmt. Das Verbundhistogramm von einem Bild mit sich selbst zeigt sehr gut welche Qualität eine Transformation hat. (vgl. 3.18).

Vorgehensweise

Im Folgenden wird der Ablauf der Registrierung dargestellt: Zunächst werden die zu registrierenden Scans parametrisiert. Dabei gilt es zu beachten, dass die Parametrisierung die Größenverhältnisse der Scans zueinander erhält, so dass sich die möglichen Freiheitsgrade auf starre Transformationen beschränken. Das Verfahren testet nun zufällig erzeugte Translationen und zufällig erzeugte Rotationswinkel auf den jeweiligen resultierenden Wert für die gemeinsame Entropie der zwei Bilder. Wichtig ist dabei, dass nur der jeweilige Überlappungsbereich in die Betrachtung mit eingeht. Die getestete Transformation wird mit dem zugehörigen Entropiewert gespeichert und eine neue Transformation erzeugt. Die Anzahl an Wiederholungen für diesen Vorgang können vom Benutzer als Parameter festgelegt werden.

Ist die geforderte Anzahl an Transformationen erzeugt worden, arbeitet das Verfahren mit den besten zehn Werten weiter. Dabei werden jeweils drei Punkte pro Bild aus dem Überlappungsbereich benutzt. Die Parametrisierung bietet die Möglichkeit aus den Bildpunkten des zweidimensionalen Bildes den zugehörigen Knoten im dreidimensionalen ursprünglichen Scan zu errechnen. Hat man pro Scan drei Knoten erhalten, kann man im dreidimensionalen Raum eine Transformation errechnen. Die entsprechende Transformationsmatrix wird als Ergebnis der Registrierung ausgegeben. Der Registrierungsvorgang ist abgeschlossen.

Ergebnis

Das Verfahren funktioniert nur mit Abtastungen, die sich parametrisieren lassen. In der vorliegenden Version lassen sich nur Scans ohne „Löcher“ oder ähnliche Unregelmäßigkeiten parametrisieren. Daher resultiert eine Einschränkung bezüglich der Eingabedaten. Da das Registrierungsverfahren versucht zwei zweidimensionale Bilder zu registrieren, gelingt eine Registrierung nur, wenn ein entsprechend großer Überlappungsbereich existiert. Weitere Schwierigkeiten ergeben sich durch ungünstige Winkel der Abtastungen zueinander. Auch hier misslingt eine Registrierung. Da die Graustufenbilder die Krümmung in den Abtastungen widerspiegeln, sind glatte Objekte ungünstiger als solche mit unregelmäßiger Oberfläche. Diese ergeben in den Graustufenbildern eher vergleichbare Merkmale.

3.1.7 Feature based matching of triangular meshes, Heinrich Müller et al.

Dieses Registrierungsverfahren ist stark an das von Fröhlich und Müller [12] präsentierte Verfahren angelehnt. Das Verfahren basiert im Wesentlichen darauf, dass zueinander passende Scans an mehreren Stellen sehr ähnliche Krümmungen aufweisen und dadurch eine Transformation bestimmt werden kann.

Vorgehensweise

Das Verfahren erhält zwei miteinander zu registrierende Scans als Eingabe und versucht eine Transformationsmatrix zu bestimmen, die den zweiten Scan so transformiert, dass er an den ersten Scan passt.

Die zur Funktionsfähigkeit des Verfahrens nötige mittlere Krümmung H wird über den Mittelwert aus den Skalarprodukten, die sich aus der Punktnormalen des betrachtenden Vektors und je einer Punktnormalen der umliegenden Punkte ergeben, ermittelt.

Die eigentliche Bestimmung der Transformation geschieht in drei Stufen. Zuerst werden Punktepaare ermittelt, von denen der eine Punkt aus dem ersten der beiden Scans stammt und der andere Punkt aus dem zweiten. Die Krümmung der beiden Scans an der Stelle dieser Punkte ist ähnlich innerhalb einer Toleranzgrenze, und zwar sowohl an der Stelle selbst, als auch in der nahen Umgebung. Vorher wird allerdings der Bereich der zu betrachtenden Punkte zur Einsparung von Rechenzeit eingeschränkt, indem nur Punkte ausgewählt werden, deren Krümmung über einem bestimmten Schwellwert liegen.

Wurden zueinander passende Punktepaare gefunden, werden passende Punktepaare zu gleichlangen Strecken verbunden, so dass sich zwei zueinander passende Punktepaare ergeben, von denen jeweils die zwei im gleichen Scan liegenden Punkte gleich weit voneinander entfernt sind.

Im dritten Schritt verbindet man die zuvor erstellten Strecken derart, dass sich zwei deckungsähnliche Dreiecke ergeben. Die Eigenschaft dieser Dreiecke ist, dass je ein Eckpunkt des ersten Dreiecks eine zu einem Eckpunkt des zweiten Dreiecks ähnliche Krümmung aufweist. Diese sechs Punkte ermöglichen es eine Transformation zu berechnen, die den zweiten Scan auf den ersten transformiert.

Auf diese Weise wird je nach Toleranz für Krümmungsähnlichkeit und Entfernungsähnlichkeit eine größere oder kleinere Menge an möglichen Transformationen ermittelt. Diese Transformationen werden der Reihe nach getestet und das Ergebnis wie in Kapitel 3.1.11 beschrieben bewertet. Die Transformation mit der besten Bewertung wird dann endgültig angewandt.

Ergebnis

Abbildung 3.19 zeigt das Ergebnis eines Registrierungsvorgangs mit dem hier beschriebenen Verfahren. Die in der Abbildung erreichte Güte der Transformation reicht aus, da die Feinregistrierung den restlichen Versatz korrigieren kann.

Das Verfahren funktioniert auf Scans mit rauen, ungeglätteten Oberflächen nicht oder nur unzureichend, da durch die raue Oberfläche eine unüberschaubar große Anzahl krümmungsähnlicher Punkte existiert. Das lässt sich auch durch Verschärfung der Kriterien nicht umgehen, da mit einiger Wahrscheinlichkeit auch die tatsächlich zueinander gehörenden Punkte nicht mehr gefunden werden.

3.1.8 Correlation of Distances to Neighbors, T. Hüntler, D. Frenzel

Correlation of Distances to Neighbors, kurz CDN, ist ein von Timo Hüntler und David Frenzel entwickeltes Grobregistrierungsverfahren. Dieses Verfahren arbeitet ausschließlich auf Rasterscans mit gleicher Auflösung. Diese Einschränkung beruht im Wesentlichen auf der Arbeitsweise des Verfahrens. Ähnliche Punkte zueinander passender Dreiecksnetze weisen ähnliche Distanzen zu ihren Nachbarpunkten auf. Aufgrund dieser Tatsache bestimmt das CDN-Grobregistrierungsverfahren ähnliche bis identische Punkte zweier Dreiecksnetze und bestimmt daraus eine Transformation.

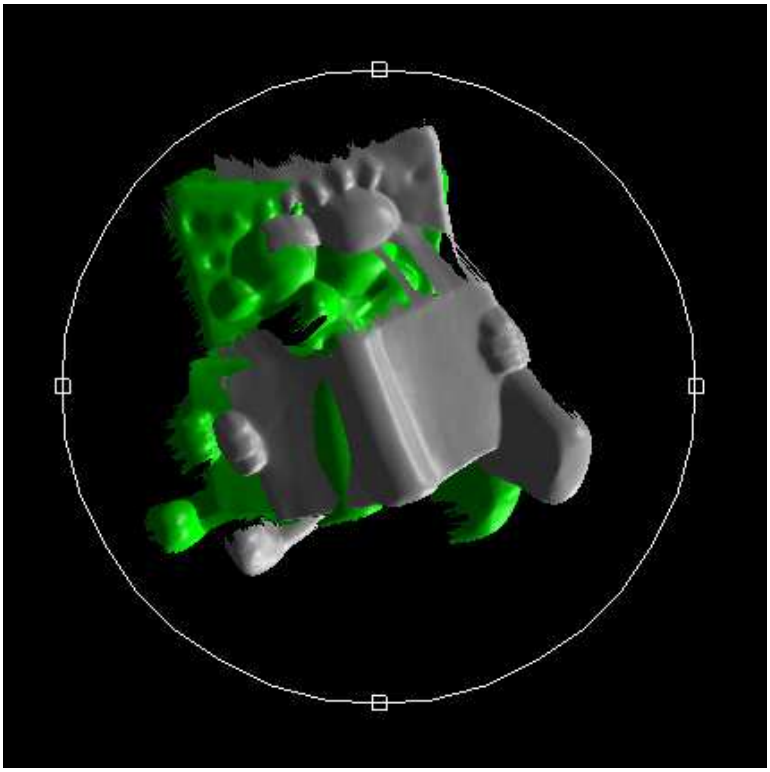

Abbildung 3.19: Feature based matching: Das Ergebnis einer Registrierung nach dem Verfahren von Müller et al..

Vorgehensweise

Das Verfahren nutzt als Eingabe zwei zu registrierende Dreiecksnetze. Um diese Dreiecksnetze miteinander zu registrieren wird eine Transformationsmatrix für das zweite Netz, anhand überlappender Bereiche so bestimmt, dass die zwei gegebenen Dreiecksnetze zusammengesetzt werden können. Dabei wird das zweite Netz durch die Matrix transformiert. Die Matrix wird in drei Schritten aufgebaut.

In einem Vorverarbeitungsschritt werden Punkte aus beiden Netzen mit ähnlicher Krümmung, gemäß einer oberen und unteren Toleranzgrenze (α und β), gewählt. Die mittlere Krümmung H wird über den Mittelwert aus den Skalarprodukten, die sich aus der Punktnormalen des betrachtenden Vektors und je einer Punktnormalen der umliegenden Punkte ergeben, ermittelt. Dann wird der euklidsche Abstand von jedem ausgewählten Punkt zu jedem seiner Nachbarn der Stufe eins und zwei berechnet und jeweils in einer dynamischen Liste aufsteigend sortiert gespeichert. Das Resultat ist je eine Liste aus Punkten mit ihren Distanzen der Stufen eins und zwei für das Dreiecksnetz. Die eigentliche Punktepaarberechnung wird durchgeführt, indem jeder berechnete Punkt aus dem ersten Dreiecksnetz mit allen berechneten Punkten aus dem zweiten Netz verglichen wird. In dem Vergleich wird die Anzahl der Nachbarn miteinbezogen:

- Wenn die Anzahl der Nachbarn der Stufe eins oder zwei abweicht wird der jeweilige Punkt verworfen.
- Ist die Anzahl der Nachbarn gleich, werden die Distanzen innerhalb einer Toleranz verglichen.
- Falls die Toleranz überschritten wird, wird der betrachtete Punkt verworfen.
- Andernfalls wird der Punkt mit einer Bewertung versehen.

Nachdem alle Punkte aus dem zweiten Netz mit dem Punkt aus dem ersten Netz verglichen wurden, wird der Punkt mit der besten Bewertung aus dem zweiten Netz dem Punkt aus dem ersten Netz zugeordnet.

Auf diese Weise werden zueinander passende Punktepaare gefunden. Im zweiten Schritt werden in beiden Teilscans Streckenlängen zwischen Punkten, die Teil je eines Punktepaares sind, berechnet. Es wird also

für zwei Punktepaare die Distanz der im ersten Teilscan liegenden Punkte und die Distanz der im zweiten Teilscan liegenden Punkte kalkuliert. Stimmen die Distanzen überein, so sind die beiden Punktepaare als zueinander passend zu betrachten.

Aus drei zueinander passenden Punktepaaren kann man dann zwei geometrisch ähnliche Dreiecke konstruieren. Dazu müssen zwei zueinander passende Punktepaare mit einem dritten Punktepaar kombiniert werden und überprüft werden, ob die Dreiecksgeometrie übereinstimmt. Es reicht aus zu überprüfen, ob die Distanzen zwischen den Dreieckspunkten aus dem ersten Teilscan mit den Distanzen zwischen den Dreieckspunkten aus dem zweiten Teilscan übereinstimmen. Ist das nicht der Fall, wird diese drei-Punktepaare-Kombination verworfen. Stimmt die Geometrie überein, erhält man sechs Punkte, je drei pro Teilscan. Diese sechs Punkte bilden die zwei Dreiecke, eines pro Teilscan.

Aus den sechs Punkten kann man dann die Transformation berechnen, die den zweiten Teilscan auf den ersten transformiert.

Es entsteht bei dieser Vorgehenweise eine Menge an möglichen Transformationen. Diese Menge an Transformationen wird der Reihe nach getestet und bewertet. Für genauere Informationen über die Bewertung wird auf das Kapitel 3.1.11 verwiesen. Die Transformation mit der besten Bewertung ist dann die Transformationsmatrix und wird auf das zweite Dreiecksnetz angewandt.

Ergebnis

Die Bearbeitungszeit des beschriebenen Verfahrens ist stark abhängig von den gewählten Toleranzen und somit von der Menge der zu vergleichenden Punkte. Die Anzahl der zu vergleichenden Punkte hat jedoch kaum Einfluss auf die Güte der Transformation. Wenn zu viele gleiche Punkte in einem Dreiecksnetz vorhanden sind ist die Transformation eher zufällig. Dennoch werden Transformationen gefunden, die mit nachträglicher Feinregistrierung gute Ergebnisse liefern. Das Verfahren arbeitet insbesondere gut auf geglätteten Dreiecksnetzen, da durch den Vorverarbeitungsschritt die Punkte anhand ihrer Krümmung selektiert werden.

3.1.9 Registering Two Overlapping Range Images, Gerhard Roth

In diesem Abschnitt wird das Registrierungsverfahren aus *Registering Two Overlapping Range Images* von G. Roth [31] vorgestellt, das automatisch zwei Tiefenbilder registiert, die sich in einem signifikanten Teil überlappen. Die Merkmale zur Auffindung von möglichen Übereinstimmungen werden aus den Intensitätsbildern extrahiert. Danach wird auf den zugehörigen 3D-Punkten der Tiefenbilder eine Delaunay-Triangulierung durchgeführt. Alle möglichen Dreiecke der zwei Tiefenbilder werden gegeneinander getestet. Um diesen Aufwand zu reduzieren, wird die Menge der Dreiecke mit Hilfe verschiedener Kompatibilitätstests noch weiter verkleinert.

Merkmalsextraktion

Die Besonderheit des Verfahrens von Roth ist, dass die Merkmale aus den Intensitätsbildern über das Helligkeitsgefälle bestimmt werden. Dies passiert mit dem SUSAN-Eckendetektor (siehe [34]), indem die Unterschiede der Helligkeitswerte eines jeden Pixels innerhalb einer kreisförmigen Maske zu der des Maskenmittelpunktes untersucht werden. Daraus kann eine Region abgeleitet werden, die gleiche oder ähnliche Helligkeitswerte wie der Mittelpunkt hat. Diese Region wird USAN (*Univalue Segment Assimilating Nucleus*) genannt. Hiermit können Ecken identifiziert werden, die Geradenschnittpunkte oder Endpunkte sind. Die Anzahl der erkannten Merkmalspunkte ist unter anderem von der Textur des Objektes im Bild abhängig (siehe Abbildung 3.20).

Jedem Merkmalspunkt aus dem Graustufenbild wird nun der entsprechende Punkt aus dem Tiefenbild zugeordnet. Drei solcher Punkte bilden ein 3D-Dreieck. Das Matchen eines korrespondierenden Dreieckspaares würde genügen um die Transformation zwischen zwei Tiefenbildern zu bestimmen.

Bei m Merkmalspunkten gibt es $\binom{m}{3}$ mögliche Dreiecke. Wenn man davon ausgeht, dass m Merkmalpunkte aus dem ersten Bild und n aus dem zweiten Bild ermittelt werden, ergeben sich $\binom{m}{3}\binom{n}{3}$ mögliche

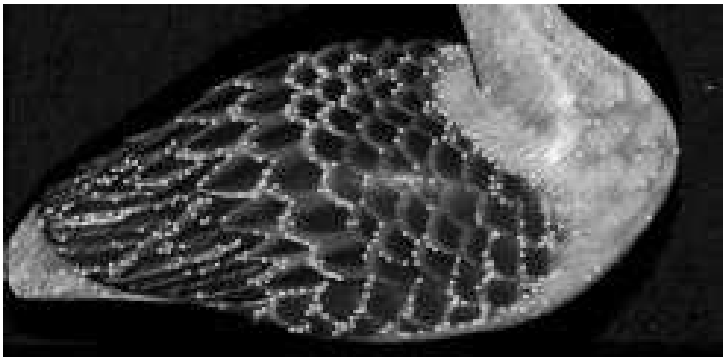

Abbildung 3.20: Roth: Erkannte Merkmalspunkte auf der Oberfläche eines Entenkörpers

Vergleiche zwischen den Dreiecken. Um die Menge der Dreiecke einzugrenzen, wird auf den Merkmalpunkten eine Delaunay-Triangulierung durchgeführt (siehe Abbildung 3.21). Ein Vorteil ist, dass die Delaunay-Triangulierung eindeutig und invariant gegenüber Translation und Rotation ist.

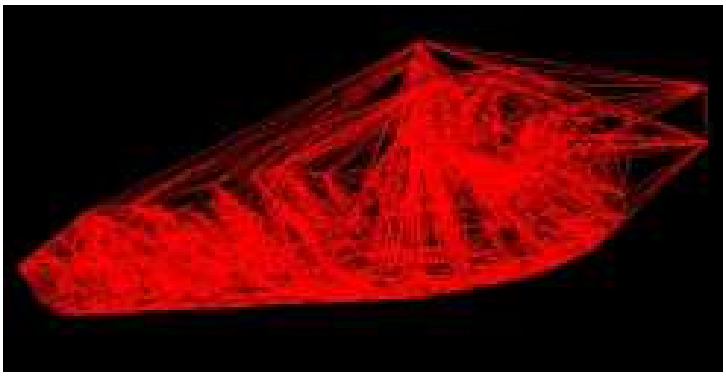

Abbildung 3.21: Roth: Delaunay-Triangulierung der Merkmalspunkte

Obwohl die Menge der möglichen Paar-Kandidaten schon stark eingeschränkt wurde, wäre der Vergleich der restlichen Dreiecke immer noch sehr rechenintensiv und somit nicht praktikabel. Um die Anzahl der Vergleiche noch weiter einzuschränken macht man sich eine Bedingung der starren Transformation zunutze. Diese besagt, dass sich die Kantenlängen eines Dreiecks nicht ändern, wenn es einer starren Transformation unterzogen wird. Daraus folgt, dass man nur noch Dreiecke mit gleichen Kantenlängen als Paar-Kandidaten betrachten muss. Die Kantenlängen eines Dreiecks werden der Länge nach sortiert, dabei wird jede Kante mit k Bits quantisiert. Jedes Dreieck wird somit durch einen $3k$-Bitstring repräsentiert. Die Dreiecke werden abhängig von ihrem Bitstring in Klassen eingeteilt. Dabei werden Dreiecke mit ähnlichen Bitstrings der gleichen Klasse zugeordnet. Die Suche nach Paar-Kandidaten aus zwei Bildern beschränkt sich auf die entsprechenden Klassen.

Anschließend werden weitere Tests auf den Knoten (Merkmalspunkten) der Dreiecke durchgeführt um die Suche nach korrespondierenden Dreiecken noch weiter einzuschränken. Jeder Dreiecksknoten stellt ebenfalls einen 3D-Punkt in dem Tiefenbild dar. In einer bestimmten Umgebung um diesen Punkt wird eine Ebene durch die Nachbarpunkte gespannt und der entsprechende Normalenvektor bestimmt. Die Winkeldifferenz zwischen den Normalenvektoren von je zwei korrespondierenden Dreiecksknoten ist invariant bzgl. einer starren Transformation. Daher werden für alle Paar-Kandidaten (Dreiecke) die Winkeldifferenzen dieser Normalenvektoren verglichen. Sollte sich eine starke Winkelabweichung ergeben, so können die entsprechenden Dreiecke verworfen werden.

Der nächste Test basiert auf der Beobachtung, dass die Distanzen eines Merkmalspunktes zu allen anderen Punkten in einem sich überlappenden Gebiet (siehe Abbildung 3.22) für jedes Tiefenbild ähnlich sind. Für jeden Merkmalspunkt in einem Tiefenbild werden deshalb die Distanzen zu allen anderen Punkten

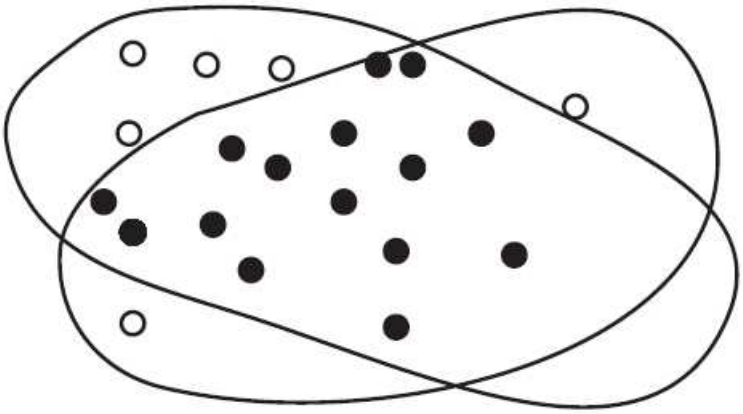

Abbildung 3.22: Roth: Sich überlappende Gebiete zweier Tiefenbilder

berechnet. Diese Distanzen werden absteigend in einem Vektor gespeichert (*vector of sorted feature distances*). Für korrespondierende Merkmale sind diese Vektoren sehr ähnlich. Der Grad der Ähnlichkeit hängt davon ab, wie stark sich die Tiefenbilder überlappen. Um die Kompatibilität zweier Merkmalspunkte zu bestimmen, werden deren Distanzvektoren verglichen. Je mehr Distanzen zwischen den Merkmalen übereinstimmen, desto wahrscheinlicher ist es, dass die betreffenden Merkmale korrespondieren. Paar-Kandidaten, die sich nicht „ähnlich“ sind, werden verworfen. All diese Tests führen zu signifikanten Einsparungen, wie man Abbildung 3.23 entnehmen kann.

	Type of object			
Compatibility test	Duck	Vase	Exca	Boat
Triangle edge length	118.2	64.0	321.3	273.7
Vertex face normal	15.4	31.5	63.2	94.0
Vertex inter-feature	25.8	50.4	66.6	64.3

Abbildung 3.23: Roth: Vergleich der Effizienz von verschiedenen Kompatibilitätstests: Verhältnis von möglichen Paaren zu akzeptierten Paaren. Je höher der angegebene Wert ist, desto effizienter ist der Test.

Registrierung

Für die übrig gebliebenen Paar-Kandidaten (Dreiecke der Delaunay-Triangulierung) werden für die Tiefenbilder die Transformationen bestimmt. Wenn eine Transformation korrekt ist, müssten die korrespondierenden Merkmale relativ genau anliegen. Deswegen wird für jede Transformation der Distanzfehler zwischen den Merkmalspunkten bestimmt. Dies wird effektiv mit einem Voxelgitter gemacht. Die Voxelgitter der transformierten Tiefenbilder werden übereinander gelegt und die Anzahl der aufeinanderliegenden Merkmale bestimmt. Je größer die Anzahl ist, desto besser fällt die Bewertung der Transformation aus. Die Transformation mit der besten Bewertung wird ausgewählt.

3.1.10 The Normal Distributions Transform, Peter Biber, Wolfgang Straßer

Das Grobregistrierungsverfahren von Biber und Straßer [4] wurde entwickelt, um die Orientierung eines mobilen Roboters in einem Raum zu ermöglichen. Es ist darauf ausgelegt das Bild eines zweidimensionalen Laserscans in einem vorhandenen Bild zu lokalisieren. Dafür verwendet es eine Normalverteilungs-Transformation, kurz NDT.

Um das Verfahren auch auf dreidimensionale Scans anwenden zu können, werden im Verlauf des Kapitels zwei Varianten bzw. Erweiterungen des 2D-Verfahrens beschrieben, die dies ermöglichen.

Das Verfahren in 2D

NDT konvertiert die ursprüngliche Punktwolke des ersten Bildes in eine andere Repräsentation. Dabei werden die Punkte über ihre Verteilung charakterisiert. Um dies zu erreichen, wird die Fläche des ersten Bildes in ein regelmäßiges Gitter eingeteilt. Die Verteilung der Punkte in jeder Zelle des Gitters wird als Normalverteilung betrachtet. Für jede Zelle werden das arithmetische Mittel $\vec{q}$ und die Kovarianzmatrix Σ ermittelt:

$$\vec{q} = \frac{1}{n}\sum_{k=1}^{n} \vec{x_k} \tag{3.37}$$

$$\Sigma = \frac{1}{n}\sum_{k=1}^{n} (\vec{x_k} - \vec{q})(\vec{x_k} - \vec{q})^T \tag{3.38}$$

Die Wahrscheinlichkeit eines Punktes an der Stelle $\vec{x}$ in der Zelle i wird über die Normalverteilung $N(q_i, \Sigma_i)$ beschrieben:

$$p(\vec{x}) = C \cdot \exp\left(-\frac{1}{2}(\vec{x} - \vec{q_i})^T \Sigma_i^{-1} (\vec{x} - \vec{q_i})\right) \tag{3.39}$$

Für die Zwecke der NTD wird C gleich 1 gesetzt.

Um zwei Scans (Punktwolken) zu registrieren wird für den ersten, wie schon beschrieben, die NDT bestimmt. Auf dem zweiten Scan wird eine starre Transformation in der Ebene durchgeführt.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{3.40}$$

Die drei Parameter dieser Transformation sind $\vec{t} = (t_x, t_y, \phi)$. Sie müssen entsprechend geschätzt werden um die folgende Bewertungsfunktion zu maximieren:

$$score(\vec{t}) = \sum_{j=1}^{J} \exp\left(-\frac{1}{2}(\vec{x_j}' - \vec{q_{i_j}})^T \Sigma_{i_j}^{-1} (\vec{x_j}' - \vec{q_{i_j}})\right) \tag{3.41}$$

$\vec{x_j}', 1 \leq j \leq J$, stellt die transformierte Menge von Punkten aus dem zweiten Scan dar; i_j ist der Index der Zelle, in dem sich Punkt $\vec{x_j}'$ befindet.

Da Optimierungsalgorithmen Funktionen im Allgemeinen minimieren, benutzt man die Negation der Bewertungsfunktion $f = -score(\vec{t})$. Die Transformationsparameter werden jeweils mit null initialisiert. Folgende Schritte werden als Newton-Verfahren iteriert bis ein bestimmtes Konvergenzkriterium erreicht wird:

1. Transformiere jeden Punkt $\vec{x}_j$ des zweiten Scans gemäß Gleichung (3.40) mit Hilfe der Transformationsparameter $\vec{t}$ zu $\vec{x}_j'$.
2. Bestimme den Zellenindex i_j, in dem sich jeder Punkt $\vec{x}_j'$ befindet, und ermittle die Parameter q_{i_j}, Σ_{i_j} der Normalverteilung der entsprechenden Zelle.
3. Berechne den Wert von $score(\vec{t})$ gemäß der Bewertungsfunktion (3.41).
4. Bestimme den neuen Parameter $\vec{t}$, indem die Funktion $f = -score(\vec{t})$ optimiert wird. Folgende Gleichung wird dabei gelöst:

$$H\Delta\vec{t} = -\vec{g} \tag{3.42}$$

Die Funktion $\vec{g}$ ist hierbei der transponierte Gradient von f mit den Einträgen:

$$g_i = \frac{\partial f}{\partial p_i} \tag{3.43}$$

die Matrix H ist die Hesse-Matrix von f mit den Einträgen:

$$H_{ij} = \frac{\partial f}{\partial p_i \partial p_j} \tag{3.44}$$

Die Lösung dieses linearen Gleichungssystems ist $\Delta\vec{t}$, was zur aktuellen Schätzung hinzuaddiert wird:

$$\vec{t} \leftarrow \vec{t} + \Delta\vec{t} \tag{3.45}$$

Anpassung an 3D

Das NDT-Verfahren arbeitet ursprünglich auf 2D-Scans. Um das Verfahren auch auf dreidimensionale Objekte anwenden zu können, gibt es zwei Möglichkeiten:

Parametrisierung

Die dreidimensionalen Objekte können auf eine Fläche parametrisiert werden (siehe Kapitel 2.1). So erhält man ein 2D-Netz, auf das nun das Original-NDT-Verfahren angewendet werden kann. Nach Abschluss des Verfahrens muss die Transformation des 3D-Netzes aus der 2D-Transformation Δt bestimmt werden.

Erweiterung auf 3D

Die andere Möglichkeit besteht darin, dass das NDT-Verfahren direkt auf 3D erweitert wird. Das hat den Vorteil, dass keine Verzerrungen durch eine Parametrisierung auftreten. Beim 2D-Verfahren werden zwei Parameter für die Translation und einer für die Rotation benötigt. Um das NDT-Verfahren auf 3D zu adaptieren müssen jedoch fünf Parameter berücksichtigt werden, so dass die drei Translations- und zwei Rotationsfreiheitsgrade beschrieben werden können.

Ergebnis

Das NDT-Verfahren liefert keine zufriedenstellenden Ergebnisse, da die Lage der 3D-Objekte unbekannt ist und somit das Newtonverfahren nicht funktioniert (siehe Kapitel 7.1).

3.1.11 Bewertung einer Registrierung

Damit automatisiert ein Gütemaß einer Registrierung zweier Scans bestimmt werden kann, wird ein Algorithmus benötigt, der die registrierten Scans bewertet. Dieser Algorithmus wird einerseits dafür genutzt, um bereits während einzelner Groberegistrierungsverfahren (z. B. Spin-Tables in Kapitel 3.1.2 oder Krsek in Kapitel 3.1.4) erfolglose Registrierungen direkt auszuschließen. Andererseits können die Scan-Paare nach der Registrierung in drei Klassen eingeteilt werden: Paare mit erfolgreicher Registrierung, Paare mit fehlgeschlagener Registierung und Paare, für die die automatische Registrierung kein eindeutiges Ergebnis liefert.

Der Bewertungs-Algorithmus soll einen Wert liefern, der die Qualität der Registrierung widergibt. Um so kleiner dieser Wert ist, um so besser ist die Registrierung. Da diese klare Zuordnung schwierig zu erreichen ist, werden drei verschiedene Bewertungsvarianten vorgestellt, die unterschiedliche Qualitätsmerkmale berücksichtigen.

Alle Verfahren verwenden nur eine zufällige Anzahl an Knoten aus den Scans, da eine Berechnung auf den kompletten Scans zu lange dauern würde. Damit eine weitere Berechnung überhaupt durchgeführt wird, müssen sich die Scans zu einem bestimmten Teil überdecken. Um dieses sicherzustellen, wird dieses

Kriterium in einem Vortest überprüft. Eine ausreichende Überdeckung bedeutet dabei, dass ein Abstandskriterium zu dem gewählten Knoten des zweiten Scans eingehalten wird. Dieses sagt aus, dass für einen Knoten aus Scan 1 ein Knoten im zweiten Scan existiert und der Abstand zwischen ihnen unterhalb eines Schwellwerts liegt.

Rating 1

Das erste Verfahren nutzt als Qualitätskriterium den Abstand der Knoten aus beiden Scans zueinander. Es sucht für alle zufällig ausgewählten Knoten aus dem einen Scan den nächsten Nachbarn im anderen. Die einzelnen Abstände werden nach folgender Formel aufsummiert:

$$result = 1 + \frac{1}{n} \sum_{i=1}^{n} dist(i)$$

$dist(i)$ ist dabei der Abstand des i-ten Knotens aus dem ersten Scan zu seinem nächsten Nachbarn in Scan zwei, n ist dabei die Größe der Knotenauswahl. Das Ergebnis $result$ ist eins, wenn sich die Scans genau überdecken.

Original

Dieses Verfahren entspricht der Bewertung, die bei der Grobregistrierung mit Spin-Tables [38] im Kapitel 3.1.2 beschrieben wird.

ABN – Angle Between Normals

Diese Variante betrachtet die Winkel zwischen benachbarten Punkten. Es sucht für jeden der zufällig ausgewählten Punkte aus dem ersten Scan den nächsten Nachbarn im zweiten Scan. Falls der Abstand zwischen den Punkten einen bestimmten Schwellwert nicht überschreitet, werden die Normalenvektoren der beiden Punkte miteinander verglichen und das Skalarprodukt der Vektoren aufsummiert:

$$result = m \left(\sum_{i=1}^{m} \vec{N}(i) \cdot \vec{N}'(i) \right)^{-1}$$

$\vec{N}(i)$ ist der Normalenvektor des i-ten Knotens aus dem ersten Scan, bei dem der Abstands-Schwellwert eingehalten wird. $\vec{N}'(i)$ ist der Normalenvektor des nächsten Nachbarns von Knoten i in Scan 2. m stellt die Anzahl der Knoten dar, die das Kriterium einhalten.

3.2 Feinregistrierung

3.2.1 Allgemeines zur Feinregistrierung

Die Aufgabe einer Feinregistrierung ist es, den Restfehler, den ein Grobregistrierungsvorgang bei der Registrierung zweier Scans übriglässt, zu korrigieren und damit die Scans möglichst perfekt zur Deckung zu bringen. Die Feinregistrierung führt nur dann zum Erfolg, wenn die vorherige Grobregistrierung ein ausreichend gutes Ergebnis geliefert hat, da der verwendete Feinregistrierungs-Algorithmus gegen ein lokales Optimum konvergiert und somit bei einer schlechten Grobregistrierung den Fehler unter Umständen nicht korrigieren kann.

ICP

Die Wissenschaftler Paul J. Besl and Neil D. McKay [3] stellten 1992 ein unabhängiges Verfahren vor, um die Registrierung von Oberflächen zu verfeinern. Voraussetzung für das ICP ist allerdings eine Vorberechnung der zu registrierenden Objekte. Für das damals vorgestellte Iterative-Closest-Point-Verfahren (ICP) wurden bisher bereits mehrere Varianten entwickelt. Der Algorithmus kann problemlos auf geometrische Daten angewendet werden, die in folgender Form vorliegen:

- Punktwolke
- Polygone
- Parametrische Kurve
- Impliziten Kurven
- Triangulierter Datensatz
- Implizite Oberflächen
- Parametrische Oberflächen

Klassischer ICP Algorithmus

Der klassische ICP-Algorithmus verläuft in folgenden Schritten: Die Punktwolken beider zu registrierenden Oberflächen sind gegeben. Eine Initialisierung hat bereits stattgefunden, die die beiden Oberflächen grob zueinander bewegt. Die Feinregistrierung liefert dann die komplette Transformation. Jeder Feinregistrierungsalgorithmus verläuft nach Rusinkiewicz und Levoy folgende Schritte:

1. Punktauswahl vornehmen aus einem oder beiden Scans
2. Punkte eines Scans transformieren, so dass beide Scans zueinander registriert werden
3. Korrespondierende Punktpaare gewichten
4. Punkte verwerfen, die nicht zueinander korrespondieren
5. Eine Fehlermetrik als Kriterium für die Punktpaare wählen
6. Minimierung der Fehlermetrik

ICP Probleme und Vorteile

Das ICP funktioniert bei bestimmten Oberflächen nicht exakt. Zwei glatte Blattseiten werden zwar zueinandergezogen, aber nicht unbedingt auch richtig gedreht. Manchmal dauert die Konvergenz des ICP lange. Ausreißer können das ICP verfälschen und die Berechnung der nächsten Punkte stellt sich oft als zeitintensiv dar.

Vorteile bietet ICP, indem es alle sechs Freiheitsgrade berechnen kann. Es ist weitgehend unabhängig von der Darstellung der Oberfläche und es benötigt keine größeren Vorberechnungen der 3D-Daten. Zusätzlich ist es rauschunempfindlich und verbessert schlechte Grobregistrierungen im Allgemeinen.

3.2.2 Verfahren

Nachdem durch die Grobregistrierung bereits eine Abschätzung einer Rotation $\mathbf{R}$ und einer Translation $\vec{t}$ ermittelt wurde, mit denen die zwei Netze an den bereits überlappenden Bereichen in Deckung gebracht werden können, wird im nächsten Schritt das Feinregistrierungsverfahren mit den groben Werten gestartet und somit als Anhaltspunkt für eine Berechnung eines genaueren Ergebnisses verwendet. Dem Verfahren liegen zwei zu vergleichende Netze, P und Q, vor, die durch die Parameter $\mathbf{R}$ und $\vec{t}$ bereits grob angenähert wurden. Von dem Verfahren gibt es unterschiedliche Implementierungen.

Feinregistrierungsablauf

Die Unterteilung des ICP-Algorithmus' wird nach Rusinkiewicz et al. [32] in vier Schritten vorgenommen, die solange iteriert werden, bis eine ausreichende Qualität erreicht wird. Im Folgenden wird die von der PG verwendete Implementierung kurz skizziert.

1. Die Wahl der Vergleichspunkte aus Netz P; dies geschieht durch eine zufällige Auswahl an Punkten. Die Anzahl beläuft sich auf ca. 10 Prozent. Dabei werden für jeden Schritt der Iteration neue Punkte ausgewählt.
2. Zu diesen Zufallspunkten werden korrespondierende Punkte aus Netz Q herausgesucht. Das geschieht unter Zuhilfenahme eines k-d-Baumes, aus dem jeweils der Knoten mit dem geringsten Abstand ausgelesen werden kann.
3. Es wird von einer konstanten Gewichtung der Korrespondenzen ausgegangen.
4. Für die genaue Berechnung der starren Transformation ist es nötig als Fehlerwert eine Summe über die quadratischen Abstände der Korrespondenzpunkte zu berechnen. Als eine effektive Methode hat sich für die anschließende Minimierung der Algorithmus von Horn [18] gezeigt. Um einen möglichst kleinen Fehlerwert zu berechnen, der die Distanz zwischen Korrespondenzpunkten festhält, verwendet Horn die Quaternionenalgebra. Die Verwendung von Quaternionen stellt eine einfache und ebenso schlanke Rechnung dar.

Die einzelnen Schritte werden so oft wiederholt, bis sich die berechnete Rotationsmatrix der Einheitsmatrix annähert. Da die Translation von der Rotationsmatrix abhängt, ist dieses Kriterium ausreichend, um die Einheitsmatrix als Abbruchkriterium zu wählen. In diesem Fall stagniert der Algorithmus, und die beiden Scans werden nicht mehr weiter bewegt. Für eine „grobe“ Feinregistrierung reicht auch ein einmaliger Durchgang der Schritte aus, um akzeptable Ergebnisse zu erzielen.

Die Quaternionenmethode nach Horn

Das Hauptziel in der Feinregistrierung besteht darin, eine Rotation $\mathbf{R}$ und eine Translation $\vec{t}$ zu finden, welche die quadratische Summe aller Abstände korrespondierender Punkte minimiert:

$$e = \sum_{i=1}^{n} \left| \vec{p}_{i_{transformiert}} - \vec{q}_i \right|^2 \tag{3.46}$$

Wenn man berücksichtigt, dass sich ein Punkt $\vec{p}$ durch Rotation $\mathbf{R}$ und Translation $\vec{t}$ entsprechend verschoben hat, kann man Gleichung (3.46) umschreiben zu:

$$e = \sum_{i=1}^{n} \left| (\mathbf{R}\vec{p}_i - \vec{t}) - \vec{q}_i \right|^2 \tag{3.47}$$

Für die weitere Berechnung ist es notwendig zuerst eine Vereinfachung vorzunehmen, die es erlaubt $\mathbf{R}$ und $\vec{t}$ getrennt zu betrachten. Eine einfache Rotation $\mathbf{R}$ eines Punktes $\vec{p}$ kann dann z. B. durch Verwendung von Quaternionen r° folgendermaßen ausgedrückt werden:

$$p^\circ{}_{rotiert} = r^\circ p^\circ \overline{r^\circ} \tag{3.48}$$

Die Entwicklung der Fehlerfunktion e geschieht somit in folgenden Schritten:

1. Man bildet für beide Netze das arithmetische Mittel aller Knoten (n=Anzahl gewählter Knoten).

$$\vec{z}_p = \frac{1}{n} \sum_{i=1}^{n} \vec{p}_i \tag{3.49}$$

$$\vec{z}_q = \frac{1}{n}\sum_{i=1}^{n}\vec{q}_i \tag{3.50}$$

Mit den Mittelpunkten werden nun alle Punkte auf neue Koordinaten berechnet:

$$\hat{\vec{p}} = \vec{p} - \vec{z}_p \tag{3.51}$$

$$\hat{\vec{q}} = \vec{q} - \vec{z}_q \tag{3.52}$$

Damit erhält man eine Gleichung, in der $\vec{t}$ jetzt nur von $\mathbf{R}$ und dem berechneten Mittelpunkt $\vec{z}_p$ und $\vec{z}_q$ abhängig ist.

$$\vec{t} = \vec{z}_q - \mathbf{R}\vec{z}_p \tag{3.53}$$

Der Vektor $\vec{t}$ setzt sich aus dem arithmetischen Mittel aller gewählten Punkte aus Netz P zusammen, abzüglich dem arithmetischen Mittel aller Korrespondenzpunkte zu Netz Q, das durch die ermittelte Matrix $\mathbf{R}$ rotiert wurde. Da $\vec{t}$ von $\mathbf{R}$ abhängig geworden ist, konzentriert sich die Suche zuerst auf das Finden einer geeigneten Rotation $\mathbf{R}$.

2. Man betrachte folgende Matrix $\mathbf{M}$, die sich zusammensetzt aus der Summe aller Tensorprodukte korrespondierender Punktkoordinaten. Man erhält eine 3×3-Matrix

$$M = \sum \vec{q} \cdot \vec{p}^T \tag{3.54}$$

Die Elemente der Matrix $\mathbf{M}$ lassen sich durch eine einfache Berechnungsformel darstellen:

$$\mathbf{M} = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix} \tag{3.55}$$

Dabei bezeichnet z. B. das Matrizenelement S_{xy} die Gesamtsumme des Produktes aller korrespondierenden x-Koordinaten aus Netz P und aller y-Koordinaten aus Netz Q:

$$S_{xy} = \sum_{i=1}^{n} x_{qi} y_{pi} \tag{3.56}$$

mit $\vec{q_i} = (x_{qi}, y_{qi})^T$. Diese Umformung erweist sich später als nützlich.

Während einer Rotation wird die Länge eines Vektors nicht verändert. Ebenfalls verändert sich auch nicht der Winkel zwischen den rotierenden Vektoren. Diese Eigenschaft ist bei der Verwendung von Quaternionen wichtig. Im Folgenden bezeichnet r° ein Quaternion. Nach Horn reicht es, die beste Rotation in folgender Gleichung zu suchen:

$$e = \sum_{i=1}^{n} \left| \mathbf{R}\hat{\vec{p}}_i - \hat{\vec{q}}_i \right|^2 \tag{3.57}$$

Dabei bezeichnen $\hat{\vec{p}}_i$ und $\hat{\vec{q}}_i$, die um die berechneten zugehörigen Mittelpunkte verschobenen Punkte der Netze P und Q.

Setzt man die Ergebnisse in Gleichung (3.47) ein, so ergibt sich nach mehreren Umformungen:

$$e = \sum_{i=1}^{n} r^{\circ T} \left(\sum_{i=1}^{n} \left| \hat{\vec{p}}_i \right|^2 \mathbf{I} - 2\mathbf{N} + \sum_{i=1}^{n} \left| \hat{\vec{q}}_i \right|^2 \mathbf{I} \right) r^\circ \tag{3.58}$$

mit

$$\mathbf{N} = \begin{bmatrix} v_1 & S_{yz} - S_{xy} & S_{xx} - S_{xz} & S_{xy} - S_{yz} \\ S_{yz} - S_{xy} & v_2 & S_{xy} + S_{yz} & S_{zx} + S_{xz} \\ S_{xx} - S_{xz} & S_{zx} + S_{xz} & v_3 & S_{zx} + S_{xz} \\ S_{xy} - S_{yz} & S_{zx} + S_{xz} & S_{zx} + S_{xz} & v_4 \end{bmatrix} \tag{3.59}$$

und

$$\begin{aligned} v_1 &= S_{xx} + S_{yy} + S_{zz} \\ v_2 &= S_{xx} - S_{yy} - S_{zz} \\ v_3 &= -S_{xx} + S_{yy} - S_{zz} \\ v_4 &= -S_{xx} - S_{yy} + S_{zz} \end{aligned} \tag{3.60}$$

und $\mathbf{I}$ als 4×4-Einheitsmatrix.

3. Man definiert nun die Matrix $\mathbf{A}$:

$$\mathbf{A} = (\sum_{i=1}^{n} \left|\hat{\vec{p}}_i\right|^2 \mathbf{I} - 2\mathbf{N} + \sum_{i=1}^{n} \left|\hat{\vec{q}}_i\right|^2 \mathbf{I}) \tag{3.61}$$

Bei der Suche nach dem kleinsten Eigenwert der Matrix $\mathbf{A}$ und der daraus resultierenden Rotation r°, formt man die Gleichung (3.58) um, so dass man die Matrix $\mathbf{A}$ herauslösen kann:

$$e = \sum_{i=1}^{n} r^{\circ T} \mathbf{A} r^\circ \tag{3.62}$$

Auf der Suche nach einer Lösung, die den Fehlerwert e minimiert, kann man sich bei den Eigenwerten bedienen. Die ermittelte symmetrische Matrix $\mathbf{A}$ besitzt vier Eigenwerte, λ_0, λ_1, λ_2 und λ_3. Gesucht ist das Quaternion r°, das Gleichung 3.62 minimiert. Dabei muss gelten: $r^\circ r^\circ$=1. Korrespondierende Eigenvektoren ($\vec{e}_0$, $\vec{e}_1$, $\vec{e}_2$ und $\vec{e}_3$) können mit folgender Gleichung gefunden werden:

$$\mathbf{A}\vec{e}_i = \lambda_i \vec{e}_i,\ i = 0, \ldots, 3 \tag{3.63}$$

Damit kann ein Quaternion als Linearkombination von Eigenvektoren aufgestellt werden.

$$r^\circ = a\vec{e}_0 + b\vec{e}_1 + c\vec{e}_2 + d\vec{e}_3 \tag{3.64}$$

Für die orthogonalen Eigenvektoren der Matrix $\mathbf{A}$ gilt:

$$r^{\circ T} \mathbf{A} r^\circ = a^2\lambda_0 + b^2\lambda_1 + c^2\lambda_2 + d^2\lambda_3 \tag{3.65}$$

Für die weitere Berechnung reicht es anhand dieser Überlegungen, den kleinsten Eigenwert der Matrix $\mathbf{A}$ zu bestimmen, um den korrespondierenden Eigenvektor als Quaternion r° für die Rotation zu erhalten.

Die Berechnung der Eigenvektoren und der zugehörigen Eigenwerte geschieht zum Beispiel mit dem Jacobi-Algorithmus [35].

4. Hat man das Quaternion $r^\circ = (r_0, r_1, r_2, r_3)^T$ berechnet, kann man mit diesem Wert auf die gesuchte 3×3-Rotationsmatrix $\mathbf{R}$ schließen, indem man das Rotations-Quaternion mathematisch umformt zu:

$$\mathbf{R} = \begin{bmatrix} r_0^2 + r_1^2 - r_2^2 - r_3^2 & 2(r_1r_2 - r_0r_3) & 2(r_1r_3 + r_0r_2) \\ 2(r_2r_2 + r_0r_3) & r_0^2 - r_1^2 + r_2^2 - r_3^2 & 2(r_2r_3 - r_0r_2) \\ 2(r_3r_2 - r_0r_2) & 2(r_3r_2 + r_0r_3) & r_0^2 - r_1^2 - r_2^2 + r_3^2 \end{bmatrix} \tag{3.66}$$

5. Da man $\mathbf{R}$ von $\vec{t}$ entkoppelt hat, ist es nun möglich, die Transformation $\vec{t}$ zu bestimmen. Die Rotation $\mathbf{R}$ wird in folgende Gleichung eingesetzt:

$$\vec{t} = \vec{z}_q - \mathbf{R}\vec{z}_p \tag{3.67}$$

Das Feinregistrierungsverfahren zeigt in der Praxis sehr gute Ergebnisse (vgl. Abbildung 3.25).

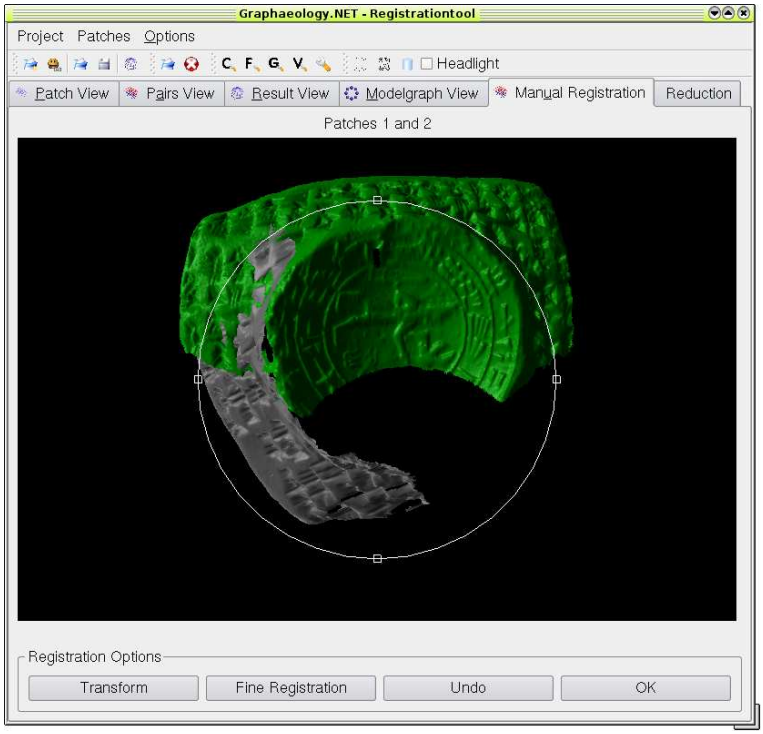

Abbildung 3.24: Feinregistrierung: Die Grobregistrierung per Hand bringt die beiden Scans annähernd gut zueinander. Eine Feinregistrierung ist in diesem Stadium der Registrierung noch nicht erfolgt.

3.3 Globalregistrierung

Ziel der Globalregistrierung

Eine Aufgabe der Globalregistrierung ist es, aus einer Menge von paarweisen Registrierungen, die in einem Grafen organisiert sind, mittels unterschiedlicher Verfahren einen „minimalen“ Spannbaum zu bilden. Die zweite Aufgabe besteht darin, Registrierungsfehler so „gering wie möglich“ zu halten. Die im Spannbaum gewählten, durch Kanten repräsentierten Registrierungen, ermöglichen es, jede enthaltene Abtastung des Modells (Knoten des Grafen) in ein gemeinsames, globales Koordinatensystem zu übertragen. Dieses globale Koordinatensystem wird vom Startelement bzw. der Wurzel des Spannbaums übernommen, alle anderen Abtastungen werden in dieses Koordinatensystem transformiert. Bei einer erfolgreichen Globalregistrierung werden schließlich alle Abtastungen des Objekts so in das globale Koordinatensystem integriert, dass ein vollständiges zusammenhängendes 3D-Modell entsteht. Es gibt verschiedene Strategien zur Auswahl der Spannbaumkanten und der Fehlerverteilung, um die bei der paarweisen Registrierung entstandenen Fehler zu minimieren. Der Fehler einer Grob- bzw. Feinregistrierung ist als Kantengewicht im Grafen gespeichert. Bei n Abtastungen und deren paarweiser Registrierung ergibt sich ein Graf mit n Knoten und $O(n^2)$ Kanten. Für die Globalregistrierung werden bei der Konstruktion des Spannbaums jene Kanten gewählt, die folgende Bedingungen möglichst optimal erfüllen:

- Der Fehler soll möglichst klein sein.
- Der Gesamtfehler im Spannbaum soll minimal sein.
- Die Fehlersumme entlang des Pfades von der Wurzel zu einem Blatt soll minimal sein.
- Der Registrierungsfehler soll an allen Stellen möglichst gut balanciert sein.

In der Regel können diese Bedingungen nicht alle gleichzeitig erfüllt werden, da z. B. eine Minimierung des Gesamtfehlers unter Umständen zu längeren Pfaden mit entsprechend höheren Fehlersummen führen

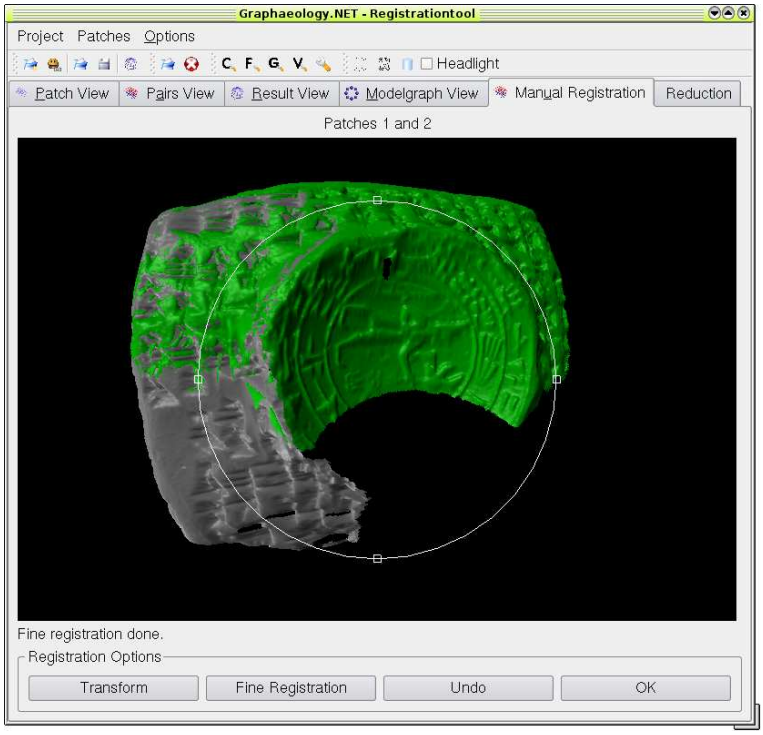

Abbildung 3.25: Feinregistrierung: Das Ergebnis einer Durchlaufes einer Feinregistrierung. Die Visualisierung der Feinregistrierung zeigt, wie perfekt diese Registrierung funktioniert. Auch durch mehrmaliges Ausführen der Feinregistrierung lassen sich noch minimale Verbesserungen erzielen.

kann. Die im Folgenden vorgestellten Verfahren unterscheiden sich unter anderem dadurch, dass die oben genannten Bedingungen unterschiedlich gewichtet werden.

3.3.1 Verfahren von Huber und Hebert

Vorverarbeitung

Das Verfahren von D. F. Huber und M. Hebert [19] arbeitet auf den numerisch bewerteten, paarweise grob- und optional feinregistrierten Abtastungen. Jedes Paar aus zwei miteinander registrierten Abtastungen stellt im Modellgraf eine Kante dar, die als Kantengewicht das Ergebnis der Bewertung erhält.

Verfahren (allgemein)

Das Grundkonzept des Algorithmus' ist die Berechnung eines minimalem Spannbaums bzgl. des Kantengewichts. Hierbei können möglicherweise lange Pfade vom Wurzelknoten zu den Blättern entstehen, was wiederum in einer hohen, lokalen Fehlersumme für den entsprechenden Pfad resultieren kann. Der Aufbau des Spannbaums erfolgt durch folgende Schritte:

1. Zunächst wird der minimale Spannbaum mit einem leeren Grafen initialisiert.
2. Auf Basis der Bewertung durch einen so genannten *free space violation*-Algorithmus (FSV), welcher zwei Abtastungen aufgrund einer angenommenen Scannerposition vergleicht, wird ein Fehlerwert bestimmt. Dieser Wert wird als Konsistenzmaß genutzt und dient als Kriterium, mit welchem die Liste aller möglichen Paarungen von Abtastungen sortiert wird.
3. Die Kante, welche die Paarung mit dem geringsten Fehler repräsentiert, wird in den Grafen aufgenommen, sofern dieser nicht bereits einen Pfad zwischen den beiden Knoten der Kante enthält.

4. Alle im bisher berechneten Grafen enthaltenen Registrierungen werden mit dem „Multiview-Registration“-Verfahren von P. Neugebauer [26] optimiert, welches die Anordnung mehrerer Abtastungen zueinander gleichzeitig verbessert.

5. Sollte der Graf den Konsistenztest mit dem FSV-Algorithmus nicht bestehen, wird die Kante verworfen und der Globalregistrierungsalgorithmus mit dem Grafen vor dem Einfügen dieser Kante (und somit auch vor der Veränderung des Grafen durch den Feinregistrierungsalgorithmus) fortgesetzt.

Das Ergebnis des Algorithmus' ist ein minimaler Spannbaum, welcher ein vollständiges optimiertes Modell repräsentiert.

Umsetzung und Ablauf in der PG

In der Implementierung des Registrierungswerkzeugs wird der als „min_span“ bezeichnete Algorithmus verwendet. Bei diesem erfolgt die Spannbaumberechnung ohne die „Multiview-Registrierung“ und den Oberflächenkonsistenz-Test. Das heißt, die Reihenfolge der in Punkt 2 des o. g. Algorithmus' erstellten Liste wird über die durch andere, im Registrierungstool implementierte Fehlerbewertungsverfahren, gegeben (siehe Kapitel 3.1.11). Die Berechnung des minimalen Spannbaums nach Prim [28] erfolgt ohne weitere Zwischenschritte.

Optional können alle durch die Kanten des minimalen Spannbaums ausgewählten, paarweisen Registrierungen noch mittels ICP-Feinregistrierung (siehe Kapitel 3.2) nachträglich optimiert werden. Dabei wurde bei der Implementierung nicht berücksichtigt, welche der beiden zueinander registrierten Abtastungen zur jeweils anderen registriert wird. So ist in diesem Fall immer die zweite, mit einer Kante gespeicherte Abtastung jene, die zur ersten Abtastung registriert wird. Die Reihenfolge, in der die Ansichtspaare feinregistriert werden, ist dabei nicht von Bedeutung, da das Ergebnis der Feinregistrierung eines Paares keinen Einfluss mehr auf die Berechnung des minimalen Spannbaums hat und die Berechnung der globalen Koordinaten erst im Anschluss an diese Feinregistrierung stattfindet. Daher handelt es sich hierbei auch nicht um ein Fehlerverteilungsverfahren im eigentlichen Sinn, sondern um eine Möglichkeit die Feinregistrierung automatisch auf alle für die Globalregistrierung genutzten Registrierungspaare anzuwenden.

Als weitere Option kann noch ein Fehlerverteilungs-Verfahren nach Sharp, Lee und Wehe [33] zur Nachbearbeitung eingesetzt werden (vgl. Kapitel 3.3.3).

Nachbearbeitung und Ergebnis

Das Ergebnis der Globalregistrierung ohne Fehlerverteilung ist in Abhängigkeit von der Qualität der Grob- und Feinregistrierung in den meisten Fällen sehr gut. In einigen Fällen muss jedoch der Fehler im Bereich der Registrierungskanten gleichmäßig verteilt werden. Dazu kann das Fehlerverteilungs-Verfahren nach Sharp, Lee und Wehe genutzt werden.

3.3.2 Verfahren von Pulli

Allgemein

Ziel des Pulli-Algorithmus' [29] ist es, durch möglichst kurze Pfadlängen die Fehlersumme in jedem Pfad gering zu halten. Dazu wird ein Spannbaum mit minimaler Tiefe berechnet, in dem iterativ jeweils der Knoten dem Grafen hinzugefügt wird, welcher einen möglichst hohen Grad an inzidenten Kanten zu den bereits im Spannbaum enthaltenen Knoten hat. Auf diese Weise wird ein Graf mit vielen kurzen Pfaden erzeugt, dessen Vorteil es ist, dass die Kanten nicht numerisch bewertet sein müssen.

In dem Verfahren nach K. Pulli wird von einer manuellen Registrierung ausgegangen, was eine Einschränkung der möglichen Kanten nicht erforderlich macht. Für jeden hinzugefügten Knoten wird eine Fehlerverteilung gegen alle seine Nachbarn durchgeführt und in dem Fall, dass die Änderungen an den bereits im Spannbaum enthaltenen Registrierungen zu groß ausfällt, wird ein Backtracking gestartet, bei dem die Fehlerverteilung für alle Nachbarknoten erneut aufgerufen wird. Das verhindert, dass ein schlecht registriertes Paar aus Abtastungen das gesamte Modell zu sehr beeinflusst.

Ein entscheidender Nachteil dieses Verfahrens im Zusammenhang mit der automatischen Registrierung ist, dass nicht immer Kanten mit minimalem Registrierungsfehler verwendet werden, da das Verfahren möglicherweise kürzere Wege (falls deren Kanten aufgrund ihres Fehlerwerts überhaupt in Frage kommen) gegenüber Wegen mit kleinerem Fehler bevorzugt. Dies führt unter Umständen zu schlechteren Ergebnissen als beim Verfahren von Huber und Hebert (vgl. Kapitel 3.3.1).

Umsetzung

Abweichend vom Original-Algorithmus werden in der PG-Umsetzung folgende Schritte durchgeführt:

1. Da keine manuelle sondern eine automatische Registrierung erfolgt, muss die Menge der möglichen Registrierungen (im Folgenden Kanten) in „gute“ und „schlechte“ Kanten unterteilt werden. Daher werden alle Kanten als „gut“ markiert, deren Bewertung kleiner als ein gegebener Parameterwert ist.
2. Der „minimale“ Spannbaum wird zunächst mit einem leeren Grafen initialisiert.
3. Die Liste der noch nicht verwendeten Knoten wird anhand der Anzahl der „guten“ inzidenten Kanten sortiert.
4. Der erste Knoten aus dieser sortierten Liste, von dem eine Kante zu einem bereits im Spannbaum enthaltenen Knoten existiert, wird dem Spannbaum und einer Warteschlange hinzugefügt.
5. Alle zum zuletzt hinzugefügten Knoten inzidenten Kanten, die als „gut“ bewertet wurden, werden dem Spannbaum hinzugefügt, sofern sie das Spannbaumkriterium „Kreisfreiheit“ nicht verletzen.
6. Solange die Warteschlange nicht leer ist, wird für den ersten Knoten dieser Warteschlange das unten erläuterte Sternfehlerverteilungsverfahren angewandt und der Knoten aus der Warteschlange entfernt. Das Fehlerverteilungsverfahren liefert einen Wert der relativen Änderung zurück, welcher als Entscheidungskriterium für das Backtracking verwendet wird. Sollte dieser Wert der relativen Änderung zu groß ausfallen, werden die den „zentralen Knoten“ umgebenden Knoten der Warteschlange hinzugefügt (näheres siehe Sternfehlerverteilungsverfahren).
7. Die Schritte 3. bis 6. werden wiederholt, bis der Spannbaum alle Knoten enthält oder keine Kante zwischen den verbleibenden Knoten und dem Spannbaum mehr existiert.

Sternfehlerverteilungsverfahren

Das eingesetzte Fehlerverteilungsverfahren basiert auf der Idee von Bergevin [2], dass in einem sternförmigen Grafen mit einer maximalen Pfadlänge von zwei, ein Element C existiert, von dem aus alle anderen Knoten N_i direkt erreichbar sind. Der Knoten C wird als „zentrales Element“ bezeichnet, die Knoten N_i sind Blätter. Jeder Weg von einem Blatt zu einem anderen Blatt hat die Länge zwei. Daher werden für alle möglichen Paarungen der Blätter die Transformationen $\boldsymbol{T}_{N_i \to N_j}$ berechnet, indem die beiden auf dem Pfad liegenden Transformationen $\boldsymbol{T}_{N_i \to C}$ und $\boldsymbol{T}_{C \to N_j}$ hintereinander ausgeführt werden,

$$\boldsymbol{T}_{N_i \to N_j} = \boldsymbol{T}_{N_i \to C} \circ \boldsymbol{T}_{C \to N_j} \tag{3.68}$$

Die Hintereinanderausführung entspricht einer „passenden“ Matrixmultiplikation. Die nun durch $\boldsymbol{T}_{N_i \to N_j}$ aneinander ausgerichteten Abtastungen werden anschließend mittels ICP (siehe Kapitel 3.2) feinregistriert. Hier wurde ebenfalls nicht berücksichtigt, welche mit welcher Abtastung eines registrierten Paares feinregistriert wird, da sich die Transformationen zu diesem Zeitpunkt noch auf die lokalen Koordinatensysteme beziehen (vgl. Kapitel 3.3.1).

Für die Neuberechnung der Transformation $\boldsymbol{T}_{N_i \to C}$ wird nun zunächst aus den zusätzlich berechneten Transformationen $\boldsymbol{T}_{N_i \to N_j}$ und den ursprünglichen Transformationen $\boldsymbol{T}_{N_j \to C}$ ein Mittelwert $\boldsymbol{T'}_{N_i \to C}$ berechnet, wobei n die Anzahl der Blätter ist:

$$\boldsymbol{T'}_{N_i \to C} = \frac{\sum_{j=1, j \neq i}^{n} \boldsymbol{T}_{N_i \to N_j} \circ \boldsymbol{T}_{N_j \to C}}{n-1} \tag{3.69}$$

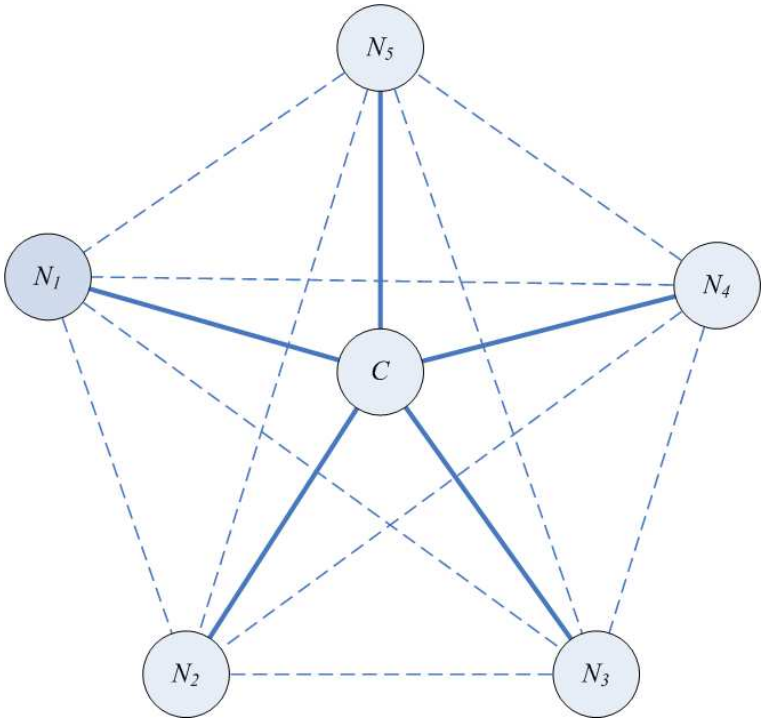

Abbildung 3.26: Sternfehlerverteilung: Sternförmiger Teilgraf um zentrale Ansicht C mit fünf Nachbarknoten N_i

Diese gemittelte Transformation $\boldsymbol{T'}_{N_i \to C}$ wird mit der ursprünglichen Transformation $\boldsymbol{T}_{N_i \to C}$ ebenfalls gemittelt:

$$\boldsymbol{T}^{neu}_{N_i \to C} = \frac{\boldsymbol{T'}_{N_i \to C} + \boldsymbol{T}_{N_i \to C}}{2} \tag{3.70}$$

Durch Aufsummieren des Differenzbetrages jeder Matrixzelle $t^{neu}_{x,y}$ der Transformationsmatrix $\boldsymbol{T}^{neu}_{N_i \to C}$ mit der Matrixzelle $t_{x,y}$ der Transformationsmatrix $\boldsymbol{T}_{N_i \to C}$ erhält man den Wert der relativen Änderung rcv bezüglich des Knotens N_i gemäß Formel (3.71).

Sei $T_i = (t_{x,y})_i$, mit $0 \leq x, y \leq 3$ die ursprüngliche Transformationsmatrix zwischen N_i und C, sowie $T^{neu}_i = \left(t^{neu}_{x,y}\right)_i$, mit $0 \leq x, y \leq 3$ die neue berechnete Transformationsmatrix, dann wird definiert:

$$rcv_i = \sum_{x=0}^{3} \sum_{y=0}^{3} \left| t^{neu}_{x,y} - t_{x,y} \right| \tag{3.71}$$

Der Durchschnitt der rcv_i-Werte über alle i bildet den Wert der gesamten relativen Änderung.

Nachbearbeitung und Ergebnis

Die Laufzeit des Algorithmus' und die Genauigkeit der Ergebnisse ist stark von zwei Parametern abhängig. Zum Ersten ist dies die Fehlergrenze, die angibt, ab wann eine Kante als eine „gute" Kante betrachtet werden kann, denn ein kleiner Wert sorgt für ein genaueres aber evtl. nicht zusammenhängendes Modell und ein großer Wert erhöht die Wahrscheinlichkeit eines zusammenhängenden Modells auf Kosten der Qualität. Zum Zweiten ist dies die Toleranzgrenze für rcv, denn je kleiner die Abweichung sein darf, desto häufiger wird die Fehlerverteilung aufgerufen. Da die Fehlerverteilung durch ihren ICP-Anteil rechenintensiv ist, erhöht eine niedrige Toleranz die Laufzeit.

3.3.3 Verfahren von Sharp, Lee und Wehe

Allgemein

Das Konzept der Fehlerverteilung in Kreisen von Sharp, Lee und Wehe [33] wurde für die Registrierung von Scans eines Drehtellerscanners entworfen. Der Modellgraf eines durch einen Drehtellerscanner erfassten Objektes ist aufgrund der Arbeitsweise des Scanners normalerweise ein Kreis, so dass es innerhalb dieses Kreises für jeden Knoten einen Weg zu sich selbst gibt, der alle anderen Knoten enthält (Abbildung 3.27). Dementsprechend existiert eine Folge von Transformationen, die einen Scan auf sich selbst abbildet. Im Idealfall ist die so berechnete Transformationsmatrix die Einheitsmatrix und der Gesamtregistrierungsfehler gleich null. Ziel der Fehlerverteilung ist es, den durch die Abweichung von der Einheitsmatrix bestimmten Fehler gleichmässig auf alle Kanten zu verteilen. Das Konzept des Verfahrens lässt sich auf spannbaumförmige Modellgrafen übertragen, indem durch das Hinzufügen von Kanten im Spannbaum Kreise erzeugt werden (Abbildung 3.28).

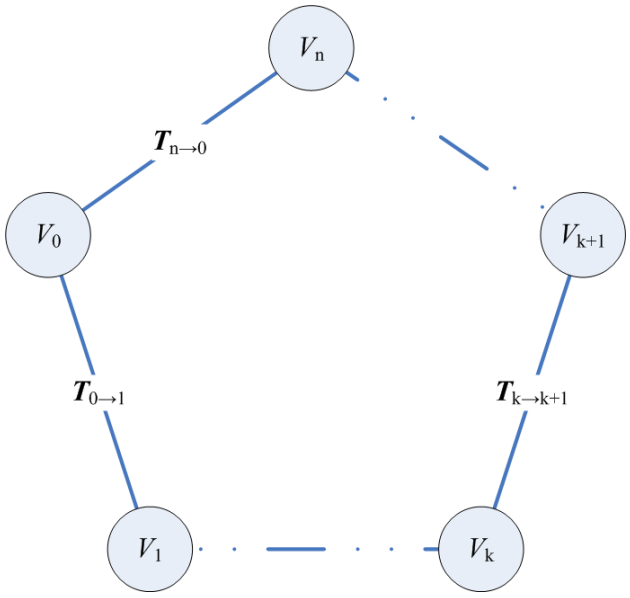

Abbildung 3.27: Globalregistrierung Sharp, Lee und Wehe: kreisförmiger Modellgraf

Vorverarbeitung

In einem berechneten Spannbaum werden durch rekursive Tiefensuche alle Pfade von der Wurzel zu den Blättern berechnet. Dann wird aus je zwei Pfaden ein Kreis gebildet, indem diese Pfade durch die Kante zwischen ihren beiden Endpunkten verbunden werden, sofern der Registrierungsfehler dieser Kante unterhalb eines Grenzwertes liegt. Diese Kante wird im Folgenden als Brückenkante bezeichnet. Die Kreise werden so gebildet, dass sie möglichst aus zwei Pfaden entstehen, die noch nicht Bestandteil eines anderen Kreises sind. Sobald alle Pfade in mindestens einem Kreis enthalten sind oder keine Kreise mehr gebildet werden können, weil der Fehler einer dafür benötigten Brückenkante den o. g. Schwellwert überschreiten würde, ist die Vorverarbeitung beendet.

Ablauf

Die berechnete Menge der Kreise wird dem Algorithmus zur Fehlerverteilung übergeben. Die Pfade, die nicht in mindestens einem Kreis liegen, werden nicht weiter bearbeitet. Kanten aus Pfaden, die in mehr als einem Kreis liegen, werden vom Algorithmus mehrmals behandelt. Dabei gehen die Veränderungen vorangegangener Durchläufe in die Fehlerberechnung mit ein.

Für jeden Kreis wird eine Transformation vom Wurzelelement des Spannbaums über alle Kanten des Kreises zurück zur Wurzel berechnet. Die Ergebnismatrix ist die Fehlermatrix $\boldsymbol{E}$, welche im Idealfall der Identitätsmatrix entspricht. Die Matrix für die gegengerichtete Transformation der Fehlermatrix ist

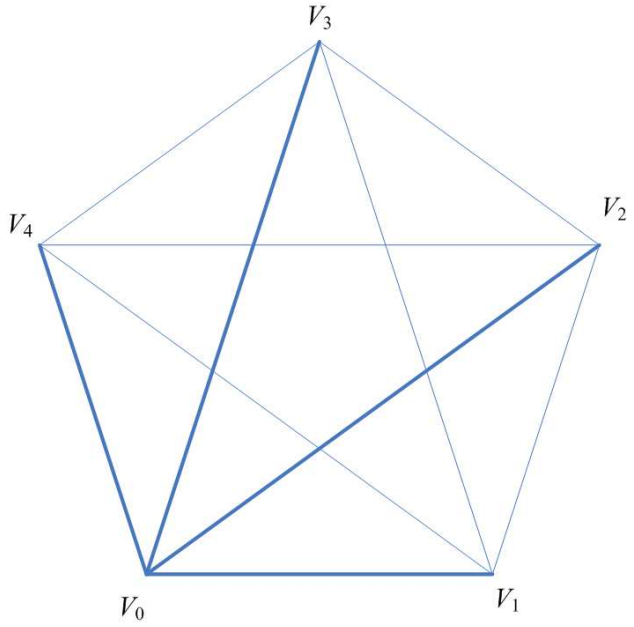

Abbildung 3.28: Globalregistrierung Sharp, Lee und Wehe: Spannbaum mit Wurzel V_0 und kreisbildenden Kanten

die Korrekturmatrix $\boldsymbol{C}$. Diese Korrekturmatrix $\boldsymbol{C}$ wird mit einem Skalar s multipliziert, der sich gemäß Formel (3.72) ergibt.

$$s = \frac{1}{\text{Anzahl der Kreiskanten}} \tag{3.72}$$

Diese Korrekturmatrix wird nun mit allen Transformationsmatrizen des Kreises multipliziert. Da die Korrektur gleichmäßig auf alle Kanten verteilt wird, wird erreicht, dass sich der Gesamtfehler innerhalb des Kreises minimiert.

Ergebnis

Tests mit dem Objekt „Siegelabdruck“ ergaben keine verwertbaren Ergebnisse, da entweder keine ausreichend genauen Brückenkanten zur Kreisbildung vorlagen oder Kanten mit hohem Fehler als Brückenkanten genutzt wurden. Dadurch verschlechterte sich das Gesamtergebnis durch die Verteilung des Fehlers deutlich. Weil die Fehlerverteilung in den von der PG betrachteten Modellen wenig zur Verbesserung beitragen konnte, wurde die Voreinstellung so gewählt, dass das Verfahren bei der Globalregistrierung nach D. F. Huber und M. Hebert [19] nicht aktiviert ist. Bei dem Verfahren von K. Pulli [29] wurde diese Art der Fehlerverteilung nicht implementiert.

3.4 Nachbearbeitung und Export

Punktwolkenbereinigung

Im Anschluss an die Berechnung des Spannbaums und die Fehlerverteilung werden die Punktwolken nacheinander in ein gemeinsames Koordinatensystem, welches durch das Koordinatensystem der zur Wurzel des Spannbaums gehörenden Punktwolke bestimmt wird, überführt. Gleichzeitig werden die Punktwolken von doppelten oder sehr nahe beieinander liegenden Punkten, die durch das Zusammenführen der Punktwolken entstanden sind, bereinigt, indem man beim Überführen einer Punktwolke in das Koordinatensystem der Zielpunktwolke folgenden Algorithmus anwendet, die bisher zusammengeführten und bereinigten Punktwolken werden dabei als „Zielpunktwolke“ bezeichnet:

1. Auf Basis der Zielpunktwolke Z wird ein k-d-Baum erzeugt.

2. Um eine Abtastung A der Zielpunktwolke Z hinzuzufügen, werden alle Punkte dieser Abtastung mit den im k-d-Baum gespeicherten Punkten verglichen und der jeweils nächste Nachbar ermittelt. Wenn die euklidische Distanz zum nächsten Nachbarn eine über einen Parameter definierte Schranke unterschreitet, wird zum einen der gefundene nächste Nachbar in einer Punktemenge L gespeichert, zum anderen wird die Position des neuen Punktes mit der Position des nächsten Nachbarn gemittelt. Anschließend wird der ggf. gemittelte neue Punkt in eine zweite Punktemenge N mit den neuen Punkten hinzugefügt. Die Punktmenge N enthält ebenfalls die Punkte aus A, deren Abstand zu ihren nächsten Nachbarn aus Z zu groß ist.

3. Sind alle Punkte der neuen Abtastung A im 2. Schritt verarbeitet, werden die Punkte der Menge der zu löschenden Punkte L aus der Zielpunktwolke Z entfernt und die Punktemenge der neuen Punkte N der Zielpunktwolke Z hinzugefügt.

Für die Bestimmung der nächsten Nachbarn mittels des k-d-Baums wird die quelloffene ANN-Bibliothek (siehe Anhang C) eingesetzt. Die Punktwolkenbereinigung wird für jeden neu hinzugefügten Scan durchgeführt, um Probleme im Überlappungsbereich von drei oder mehr Scans zu vermeiden. Die Zielpunktwolke bildet nach dem Hinzufügen des letzten Scans die Eingabe für die Rekonstruktion.

Rekonstruktion

Die Rekonstruktion soll über ein externes Werkzeug erfolgen und war nicht Aufgabe der PG.

Export

Der Export erfolgt im OFF-Format [30]. Eine detaillierte Beschreibung zu diesem Dateiformat befindet sich in Anhang B.2.

Kapitel 4

Reduktion

Die durch die Registrierung bzw. Rekonstruktion zur Verfügung gestellten Dreiecksnetze können eine erhebliche Größe erreichen. Sie können dann nicht mehr in Echtzeit dargestellt und auch nicht effizient in einem Client-Server-System übertragen werden. Hier setzt die Reduktion ein. Durch die in diesem Kapitel vorgestellten Reduktionstechniken wird die Anzahl der Dreiecke eines gegebenen Dreiecksnetzes stark verringert.

4.1 Allgemeines zur Reduktion

Um die Bedingungen einer Echtzeitdarstellung, ein hoher Detailgrad und eine geringe zu übertragende Datenmenge, zu realisieren, bieten sich vielfach aufgelöste Netze an [5]. Diese Netze sind vorberechnete Strukturen, aus denen durch eine Offline-Reduktion reduzierte Netze ohne komplette Neuberechnung gewonnen werden können.

Im Anwenderprogramm wird dann durch eine LOD-Technik (*Level-Of-Detail*) entschieden, welches reduzierte Netz aus der Struktur dargestellt werden soll. Als Ergebnis erhält man ein extrahiertes Netz, das in Echtzeit darstellbar ist. Dieses extrahierte Netz ist ein beliebiges Netz der Struktur, das durch Anwendung der inversen Reduktionsoperationen gewonnen werden kann. In einem Client-Server-System wird durch die Verwendung von vielfach aufgelösten Netzen die zu übertragende Menge an Daten erheblich reduziert[1]. Der Server überträgt initial das gröbste Netz der Struktur zum Client. Dann werden nur noch Netzmodifikationen an den Client gesendet. Durch dieses Verfahren müssen keine kompletten Netze übertragen werden [1].

Um ein vielfach aufgelöstes Netz zu erhalten, wird bei der Reduktion anhand einer Fehlerschranke ein gegebenes Netz M in ein reduziertes Netz M' überführt [5]. Diese vielfach aufgelösten Netze sind gegeben durch $N{=}(\Gamma_0,\ R,\ \prec)$ und bestehen aus einem Basisnetz Γ_0, einer Sammlung von Modifikationen $R{=}\{M_1,...,M_h\}$ und einer Abhängigkeitsrelation $\prec$ auf R. Das Basisnetz ist das gröbste Netz des vielfach aufgelösten Netzes. Eine Modifikation $M{=}(\Gamma_1, \Gamma_2)$ ist eine Grundoperation, wie z. B. Kantenkontraktion, um ein Netz Γ lokal zu verändern, wobei Γ_1 und Γ_2 Teilnetze mit identischem Rand sind. Im Prinzip wird bei einer Modifikation M das Teilnetz Γ_1 durch Γ_2 ersetzt und aus dem Netz Γ wird Γ' erzeugt. Um aus einem vielfach aufgelösten Netz ein extrahiertes Netz zu erhalten, müssen beispielsweise nicht die Kantenkontraktionen gespeichert werden, sondern die inversen Operationen $R{=}\{vsplit_0,...,vsplit_{n-1}\}$. Allgemein ergibt sich dann die Struktur $N{=}(\Gamma_0,\{vsplit_0,...,vsplit_{n-1}\})$. Die Relation $\prec$ ist in der Menge R enthalten. Die Operation $vsplit_i(s_i, l_i, r_i)$ fügt den Punkt $\vec{v_t}$ und zwei neue Facetten $\{v_s, v_t, v_l\}$ und $\{v_t, v_s, v_r\}$ ein (Abbildung 4.1).

[1] Das gilt natürlich nur dann, wenn nicht das gesamte Objekt mit allen Details angezeigt werden soll.

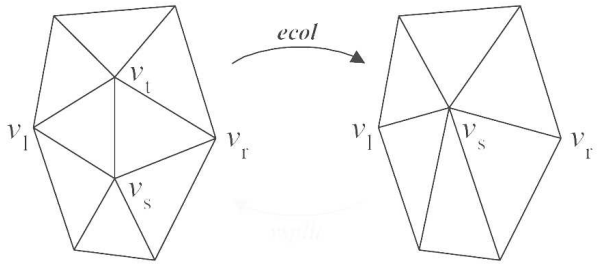

Abbildung 4.1: Reduktion: Durch eine Kantenkontraktion (*ecol*) wird ein neuer Punkt $\vec{v_s}$ und die korrespondierende Inverse (*vsplit*) erzeugt [17].

4.2 Verfahren

Die Grundoperation bzw. Modifikation der hier verwendeten Reduktion ist die Kantenkontraktion. Ein neuer Punkt, z. B. der Mittelpunkt der zu kontrahierenden Kante, wird eingefügt, und die Endpunkte der Kante werden entfernt.

Um die zu kontrahierenden Kanten zu erhalten werden die Kanten anhand einer definierten Fehlerschranke sortiert. Kanten, deren Fehler einen gegebenen Grenzwert (Fehlerschranke) überschreiten, werden nicht reduziert. Bei der implementierten Reduktion kann der Benutzer zwischen zwei Fehlermetriken wählen, die im Folgenden beschrieben werden.

4.2.1 Reduktion anhand der Kantenlänge

Bei der Reduktion auf Basis der Kantenlängenfehlermetrik basiert der Fehler bzw. die Fehlerschranke auf der Länge der Kanten. In jedem Reduktionsschritt wird eine Kante reduziert. Dazu wird die Länge der zu reduzierenden Kante geprüft. Wird eine zuvor festgelegte maximale Länge nicht überschritten, wird die Kante auf einen Knoten reduziert. Der dabei neu entstehende Knoten ist der Mittelpunkt dieser Kante, berechnet aus dem euklidischen Abstand der zwei Kantenendpunkte. Dann wird im Reduktionsschritt eine weitere Kante reduziert. Dieser Vorgang wird solange wiederholt bis eine vorher definierte minimale Dreiecksmenge erreicht wird.

4.2.2 Reduktion anhand der Krümmung

Bei der Reduktion auf Basis der Krümmungsfehlermetrik ergibt sich der Fehler der Kante anhand der Krümmung nach [21]. Der Krümmungswert W eines Punktes

$$W = \lambda_a \cdot \frac{A}{A_{max}} + (1 - \lambda_a) \cdot \frac{R'}{R'_{max}} \tag{4.1}$$

ergibt sich aus der Gauß-Krümmung K und der mittleren Krümmung H.

$$K = \frac{2\pi - \sum \phi}{\frac{1}{3}A} \tag{4.2}$$

$$H = \frac{\sum m(e_i)}{\frac{1}{3}A} \tag{4.3}$$

Aus den beiden Krümmungen ergibt sich $R' = \lambda_s \cdot R$ für Sattel-Punkte oder $R' = \lambda_f \cdot R$ für Endknoten scharfer Kanten mit

$$R = \sqrt{2H^2 - K} \tag{4.4}$$

Dabei sind λ_s und λ_f unterschiedliche Punktgewichte. Der Bezeichner λ_a ist ein Flächengewicht, A ist die Summe der Flächeninhalte der inzidenten Dreiecke des Punktes und A_{max} ist das Maximum der A-Werte aller Punkte. R'_{max} ist das Äquivalent zu A_{max} bzgl. R'. Der Winkel zwischen zwei benachbarten Kanten des Punktes wird mit ϕ bezeichnet. Mit e_i wird eine Kante benannt und $m(e_i)$ ist der Winkel zwischen den Normalen der zwei benachbarten Flächen einer Kante. Der Fehler der Kante ergibt sich aus dem Mittel der Fehler der Kantenendpunkte. Dann wird analog zur Reduktion auf Basis der Kantenlängenfehlermetrik weiter verfahren.

Kapitel 5

Client / Server

Im Folgenden werden die Module Client und Server beschrieben, die zur Übertragung und Darstellung der fertigen 3D-Netze über TCP/IP basierte Netzwerke dienen. Der Server ist eine Konsolenanwendung, welche auf eingehende TCP/IP Verbindungen wartet. Der Client ist eine fensterbasierte Anwendung, welche auf QT aufsetzt und OpenGL zur Darstellung der 3D-Netze nutzt.

5.1 Allgemeines zum Server

Der Server ist eine Konsolenanwendung, welche auf dem vordefinierten Port 44044 auf eingehende TCP/IP Verbindungen wartet. Die Anwendung reagiert ausschließlich passiv auf Anfragen des Clients und beantwortet diese. Nach einer eingehenden erfolgreichen Authentifizierung können alle zur Darstellung im Client notwendigen Daten abgerufen werden. Dazu gehören neben einer Liste aller auf dem Server verfügbaren Modelle auch Zusatzinformationen im HTML Format inklusive Bilder. Der Server benutzt zur Übertragung der großen Datenmengen einen Kompressionsmechanismus um Bandbreite zu sparen. Die Daten, die in einem speziellen binären Format vom Registrierungswerkzeug geliefert werden, werden in einer festgelegten Verzeichnisstruktur erwartet. Des Weiteren findet sich dort auch eine Liste mit autorisierten Benutzern und den benötigten Kennwörtern in einem XML basierten Format.

5.2 Allgemeines zum Client

Der Client ist eine fensterbasierte Anwendung, welche die Darstellung und Vermessung von 3D-Objekten ermöglicht. Die dargestellten Objekte erhält der Client über eine TCP/IP Verbindung zu einem Graphaeology.NET Server. Die Objekte werden lokal zwischengespeichert, so dass eine abgebrochene oder fertige Übertragung eines Objektes nicht erneut ausgeführt werden muss. Bei der Übertragung wird eine Datenkompression benutzt.

5.3 Kommunikationsprotokoll

Die Kommunikation zwischen Client und Server findet über TCP/IP Sockets statt. Die Verbindung wird vom Client aus initiiert. Ein Paket besteht in jedem Fall aus einem Header und einem Datenteil. In dem Header stehen alle notwendigen Informationen, um ein Paket vollständig einzulesen und zu interpretieren. Der Datenteil wird entsprechend der Informationen im Header interpretiert und ausgewertet. Es gibt folgende Paketarten:

- Anmeldung am Server + Antwort
- Anforderung der Modellliste + Antwort

- Setzen des ausgewählten Modells auf dem Server
- Anfrage von Zusatzinformationen eines Modells + Antwort
- Anfrage des Basisnetzes des ausgewählten Modells + Antwort
- Anfrage einzelner Abtastungsdaten des ausgewählten Modells + Antwort

Dabei erfolgt die Übertragung der einzelnen Abtastungsdaten komprimiert.

Kapitel 6

Implementierungsdetails

6.1 Implementierung Registrierungswerkzeug

Die Implementierung des Registrierungswerkzeug orientiert sich sehr stark an dem im Pflichtenheft aufgezeigten Arbeitsablauf. Dabei wurde das Programm konsequent modular entwickelt. Die einzelnen Programmkomponenten arbeiten in einer linearen Reihenfolge und benutzen jeweils die Ergebnisse der vorherigen Stufe. Es ist jedoch jederzeit möglich zu einer bereits durchlaufenen Stufe zurückzukehren um dort Einfluß auf die bisherigen Teilergebnisse zu nehmen.

Zunächst lädt ein Benutzer die zu registrierenden Scans und startet einen Registrierungsvorgang mit einem ausgewählten Verfahren. Nachdem jeder geladene Scan mit jedem anderen registriert wurde, wird das Ergebnis für jede Paarung angezeigt. Dabei sortiert das Programm die jeweiligen Paarungen nach der errechneten Bewertung. Auch hier kann der Anwender gezielt eingreifen und gegebenenfalls einzelne Paarungen nachregistrieren, andere Registrierungsverfahren durchführen oder eine manuelle Grobregistrierung vornehmen. Danach können optional sämtliche Paarungen feinregistriert werden, um eine möglichst genaue Ausrichtung zueinander zu erreichen. Als letzter Schritt versucht die Globalregistrierung aus den einzelnen Paarungen ein Gesamtobjekt zusammenzusetzen.

Es wurden für jeden Schritt der Registrierung verschiedene Registrierungsverfahren umgesetzt. Der Benutzer kann für jeden Schritt zwischen unterschiedlichen Verfahren wählen, um für den jeweiligen Satz an Abtastungen ideale Ergebnisse zu erzielen. Aufgrund der Beschaffenheit des Problems eignet sich nicht jedes Verfahren für jede Art von Scan.

6.1.1 Spin Tables, Winkelbach et al.

Ziel dieses Verfahrens ist es auf Basis von *Spin Tables* zwei Scans zu registrieren. Das Verfahren startet damit, dass eine `ANN`-Datenstruktur (vgl. Anhang C) für die Bewertungsfunktion initialisiert wird. In einer Schleife werden die beiden 64×64-Spin-Tables abwechselnd gefüllt, bis drei Punkte mit ähnlichem Radius und ähnlicher Höhe gefunden werden. Diese möglicherweise korrespondierenden Punkte werden aufeinander transformiert und eine Transformationsmatrix bestimmt. Die beiden Scans und die Transformationsmatrix werden an die Bewertungsfunktion übergeben. Die zurückgelieferte Bewertung wird mit der aktuell besten Bewertung verglichen und gegebenenfalls durch den neuen Wert ersetzt. Nach einer festgelegten Anzahl an Schleifendurchläufen wird die Transformationmatrix mit der besten Bewertung zurückgegeben.

Klassendiagramm-Beschreibung

In Abbildung 6.1 sieht man die für das *Spin Tables*-Verfahren benötigten Klassen. Der Algorithmus wird mit der öffentlichen Methode `start` gestartet. Dieser Methode werden zwei Zeiger auf ein Objekt der Klasse `Patch` und ein Zeiger auf ein Objekt der Klasse `Settings` übergeben. Über das Objekt `settings`

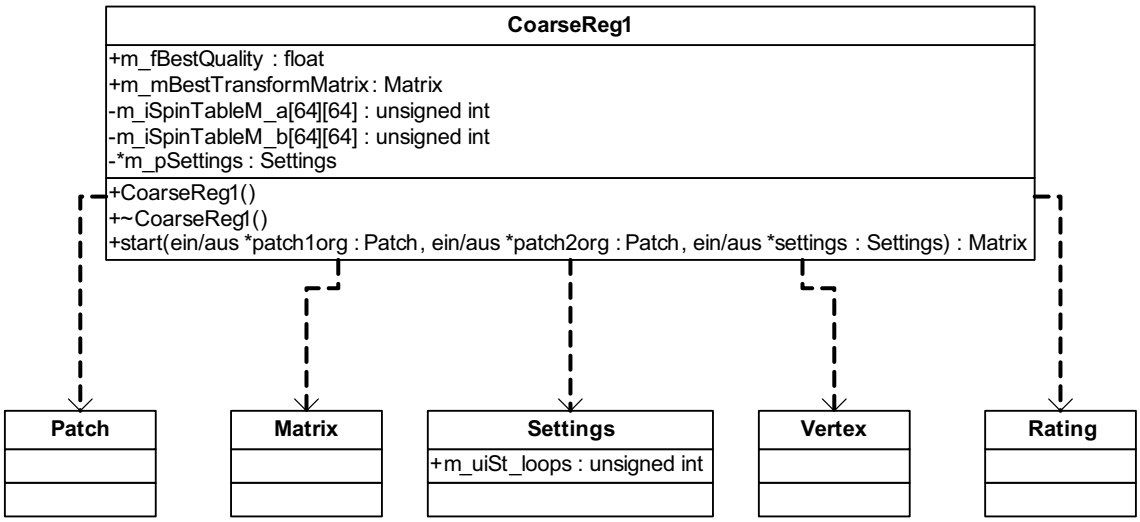

Abbildung 6.1: Auszug aus dem Registrierungswerkzeug-Klassendiagramm: Darstellung der für das Spin-Tables-Verfahren relevanten Klassen.

wird auf die Variable `m_uiSt_loops` zugegriffen, um die Anzahl der Schleifendurchläufe zu bestimmen. Die Werte der *Spin Tables* werden in den beiden 64×64 `unsigned int` Arrays `m_iSpinTableM_a` und `m_iSpinTableM_b` gespeichert. Die jeweils beste Bewertung und Transformation wird in `m_fBestQuality`, bzw. `m_mBestTransformationMatrix` gespeichert. `m_mBestTransformationMatrix` stellt gleichzeitig auch den endgültigen Rückgabewert der Methode `start` dar.

6.1.2 Spin-Images, Andrew Edie Johnson, Martial Hebert

Die Implementierung des in Kapitel 3.1.3 beschriebenen Verfahrens von Andrew Edie Johnson und Martial Hebert[20] entspricht weitestgehend der Vorgehensweise der Autoren. Zur groben Gliederung kann das Verfahren in die folgenden fünf Phasen eingeteilt werden:

1. Berechnung der *Spin-Images*.
2. Bestimmung der Korrelationskoeffizienten zwischen einem zufällig ausgewählten *Spin-Image* im ersten Teilnetz und n zufällig ausgewählten *Spin-Images* aus dem zu vergleichenden zweiten Teilnetz.
3. Erzeugen eines Histogramms in welches die Korrelationswerte eingetragen werden. Wiederholen der Schritte 2 und 3 für m zufällig ausgewählte *Spin-Images* aus dem ersten Teilnetz. Für die weitere Verwendung werden nur die Histogramme gespeichert, die so genannte Ausreisser enthalten.
4. Wahl von drei möglichst konvergen Punktpaaren aus den Punkten, die zu den besten Ausreissern in den Histogrammen gehören.
5. Berechnung der Rototranslation und Bewertung des Ergebnisses.

In der ersten Phase werden die *Spin-Images* erzeugt. Da die *Spin-Images* für ein Netz immer identisch sind, ist es nicht nötig die *Spin-Images* für den Vergleich zweier Teilnetze jedes Mal neu zu berechnen. Statt dessen werden die einmal berechneten *Spin-Images* auf die Festplatte geschrieben. Daher wird zunächst überprüft, ob sich zu den beiden zu vergleichenden Teilnetzen eine passende Datendatei auf der Festplatte befindet. Ist dies nicht der Fall wird mit der Berechnung der *Spin-Images* begonnen. Für die Berechnung werden nur die Punkte in einer lokalen Umgebung um den betrachteten Punkt benötigt. Daher bietet sich für eine effiziente Suche nach relevanten Punkten die Zerlegung der gesamten Punktwolke in *Bounding Boxes* an. Die Klasse `BoundingBoxedPatch` liefert genau diese Möglichkeit. Dieser Klasse wird über den Konstruktor ein `Patch` übergeben, sowie die Informationen, wie groß die *Spin-Images*

sind und welche Auflösung die einzelnen *Bins* haben. Die beiden zuletzt genannten Werte können vom Benutzer frei gewählt werden. Sie werden über die `Settings`-Klasse in den Attributen `m_iSi_numOfBins` und `m_fSi_binSize` übergeben. Mit diesen Informationen wird eine Zerlegung der Punktwolke berechnet, so dass sich die Anzahl der *Bounding Boxes* und die durchschnittliche Anzahl der Punkte pro *Bounding Boxes* die Waage halten. Nun wird in dem gewählten Teilnetz jeder i-te Punkt zur Berechnung eines *Spin-Images* benutzt. Der Wert von i lässt sich ebenfalls durch den Benutzer vorgeben und wird über das Attribut `m_iSi_Skip` der `Settings`-Klasse definiert. Zu dem Punkt existiert in der `Patch`-Klasse eine Punktnormale. Mit dem Punkt, der Punktnormalen und der Größe des *Spin-Images* lässt sich dann der Konstruktor für ein neues `SpinImage`-Objekt aufrufen. Zu diesem *Spin-Image* werden alle Punkte der lokalen Umgebung über die Methode `addPoint` hinzugefügt. Um die Menge der Punkte zu bestimmen werden die entsprechenden *Bounding Boxes* verwendet. Diese ergeben sich aus den Werten, welche die Methoden `getMatchingBBox` und `getNumOfBoundingBoxes` zurückliefern. Bei der Erzeugung der *Spin-Images* wird für jeden eingefügten Punkt der Wert 1 mit bilinearer Verteilung auf die vier *Bins* verteilt, deren Mittelpunkte ein, den Punkt umspannenden Würfel bilden. Das fertige *Spin-Image* wird dann mit der Methode `getData` ausgelesen und auf die Festplatte geschrieben und schließlich aus dem Speicher gelöscht.

Zu Beginn der zweiten Phase werden die *Spin-Images* zu den beiden zu vergleichenden Teilnetzen von der Festplatte geladen, allerdings nur dann, wenn sie sich noch nicht im Arbeitsspeicher befinden. Das bedeutet, wenn zum Beispiel die Rototranslationen für die Teilnetzkombinationen 1/2, 1/3, 1/4 usw. berechnet werden, muss nur das zweite Teilnetz nachgeladen werden. Nun werden für alle Kombinationen der *Spin-Images* der beiden Teilnetze die Korrelationskoeffizienten berechnet. Dazu wird jeweils die Methode `correlation` aus der `SpinImage`-Klasse aufgerufen. Diese liefert einen Fließkommawert, der dem Grad der Korrelation entspricht, zurück. Zusätzlich zum Verfahren nach Hebert und Johnson wurden einige Filter eingesetzt.

- Ist mindestens ein bestimmter Prozentsatz der *Bins* gefüllt? Diesen Prozentsatz kann der Benutzer frei wählen.
- Wurde mindestens ein bestimmter Prozentsatz Punkte für die Bildung des *Spin-Images* in Bezug auf die maximal verwendete Anzahl der Punkte verwendet? Auch diesen Prozentsatz kann der Benutzer frei wählen.

Der berechnete Korrelationskoeffizient, zusammen mit den Indizes der zugehörigen Punkte wird als `CorrelationPoint`-Typ in einen `vector` gespeichert. Immer wenn alle Korrelationskoeffizienten eines *Spin-Images* des ersten Teilnetzes mit allen *Spin-Images* des zweiten Teilnetzes berechnet wurden, wird Phase 3 fortgesetzt.

In der dritten Phase wird anhand der Korrelationen ein Histogramm gebildet. Dazu werden die Korrelationen an die Methode `Histogramm` übergeben. Dort werden die Korrelationskoeffizienten zunächst sortiert und auf ein Histogramm abgebildet. Dieses Histogramm wird auf Ausreisser untersucht. In der Statistik und im Dokument von Hebert und Johnson wird dabei auf die Dreiviertel-Methode verwiesen. Hier weicht die Implementierung ab, da sich diese Schranke als zu schwach erwiesen hat. Zusätzlich kann deshalb die Bedingung angegeben werden, dass der Ausreisser einen Mindestabstand zum nächstkleineren Wert im Histogramm haben muss. Dieser Wert wird vom Benutzer vorgegeben und über die `Settings`-Klasse und das Attribut `m_fSi_ausreisser` übergeben. Ein Wert von 0.85 bedeutet, dass der nächstkleinere Wert höchstens das 0.85-fache des aktuellen Wertes betragen darf. Sind diese Bedingungen erfüllt wird der `CorrelationPoint` für die weitere Verwendung gespeichert. Anschließend wird der `vector` mit den `CorrelationPoint`-Werten geleert und mit dem nächsten *Spin-Image* aus dem ersten Teilnetz in Phase 2 fortgesetzt.

Sind alle Kombinationen der *Spin-Images* berechnet beginnt in Phase 4 die letzte Auswertung. Die gefundenen Ausreisser werden sortiert. Dann wird unter den besten 10 Korrelationen die Dreierkombination gesucht, welche die geringsten Fehler in Bezug auf Winkel und Kantenlängen der beiden durch die Punkte aufgespannten Dreiecke bildet.

Mit den beiden gefundenen Dreiecken wird schließlich die Rototranslation berechnet.

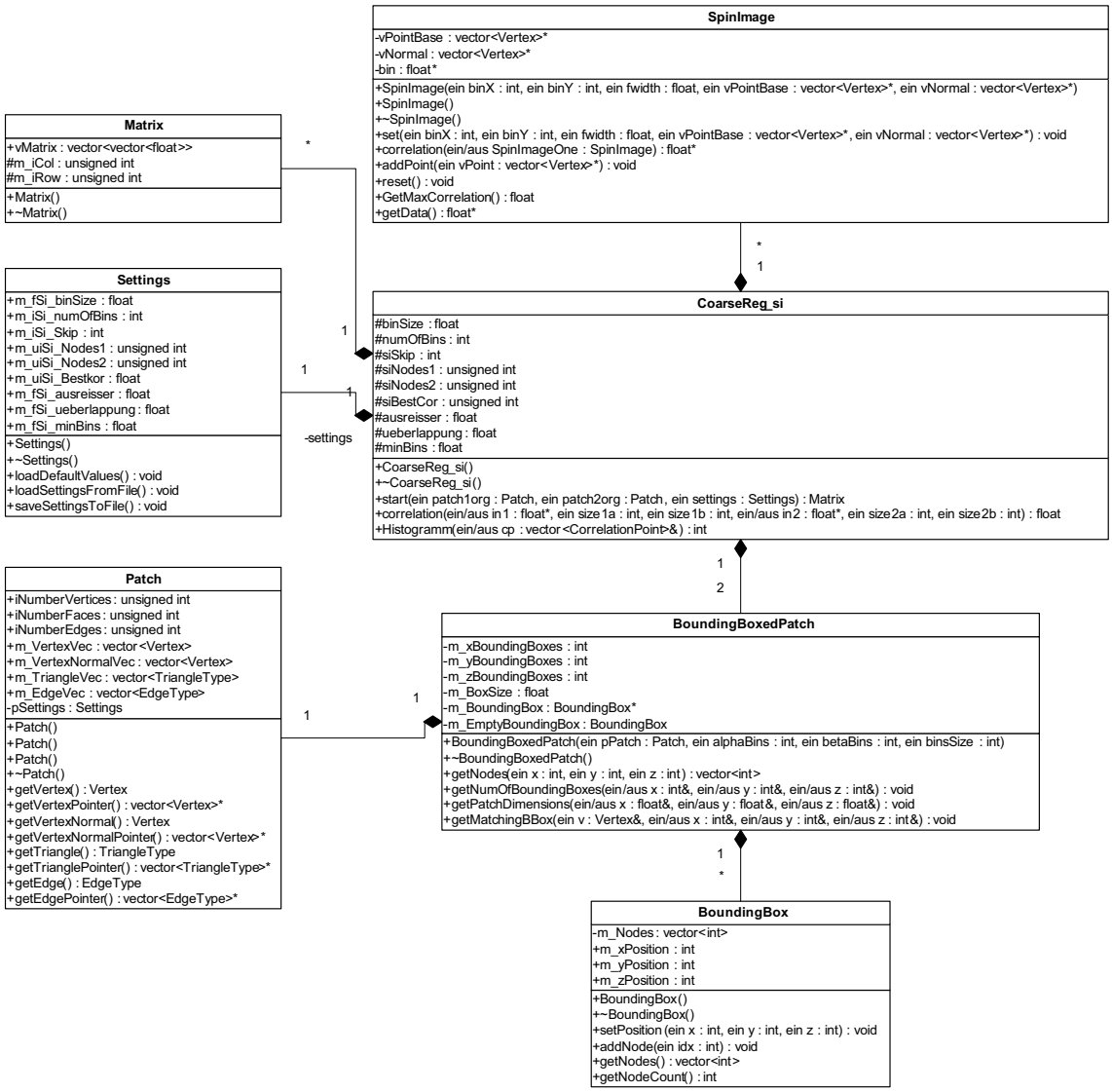

Abbildung 6.2: Auszug aus dem Registrierungswerkzeug-Klassendiagramm: Darstellung der für das Spin-Image-Verfahren relevanten Klassen

Beschreibung des Klassendiagramms

Viele der in Abbildung 6.2 benutzten Methoden und Attribute in den aufgeführten Klassen wurden bereits innerhalb der Implementierungsdetails beschrieben. Daher werden im Folgenden nur die wichtigen noch verbliebenen Elemente beschrieben.

Die `BoundingBox`-Klasse

Die `BoundingBox`-Klasse kapselt eine einzelne Zelle eines orthogonal und regelmäßig zerlegten Vektorraums im $\mathbb{R}^3$. Die Attribute `m_xPosition`, `m_yPosition` und `m_zPosition` enthalten die Indizes um die Box im Raum anordnen zu können. Ansonsten bietet die Klasse lediglich Methoden einen Punkt mit `addNode` hinzuzufügen, alle Punkte mit `getNodes` auszulesen und die Anzahl der Punkte mit `getNodeCount` zu bestimmen.

Die `BoundingBoxedPatch`-Klasse

Die `BoundingBoxedPatch`-Klasse zerlegt ein übergebenes Patch. Dazu werden die Punkte einzelnen *Bounding Boxes* zugeordnet und in ihnen gespeichert. Die Größe dieser Boxen hängt davon ab, wie groß die

Spin-Images werden sollen. Sie sind aktuell so bemessen, dass sie eine Außenkantenlänge haben, die genau der Dimension eines Spinimages entspricht. Daher müssen für die Verwendung bei der Berechnung der *Spin-Images* die umliegenden 26 *Bounding Boxes* mit verwendet werden.

Die wichtigsten Methoden sind demnach die Methode `getMatchingBBox`, welche die *Boundig Box* bestimmt in welcher sich der angegebene Knoten befindet und die Methode `getNodes`, welche das *Array* der enthaltenen Punkte zurückliefert.

Die `Matrix`-Klasse

Die `Matrix`-Klasse wird als Rückgabe-Typ für die Methode `start` aller Grobregistrierungsverfahren verwendet. Sie kapselt neben den reinen Werten der Matrix auch Methoden zur Addition und Multiplikation, sowie Methoden zur Berechnung der inversen oder transponierten Matrix. Diese Funktionen werden jedoch an dieser Stelle nicht benötigt, so dass das Klassendiagramm auf die reine Datenrepräsentation reduziert wurde.

Die `SpinImage`- und die `CoarseReg_si`-Klasse

Diese beiden Klassen wurden hinreichend detailliert in den Implementierungsdetails beschrieben, weswegen an dieser Stellen nicht mehr darauf eingegangen wird.

6.1.3 Range image registration driven by a hierarchy of surface differential features, P. Krsek

Das Grobregistrierungsverfahren von Krsek et. al. [22], das in Kapitel 3.1.4 beschrieben wurde, kann bei großen Patches zu langen Rechenzeiten führen. Um diesen Effekt abzuschwächen, wurde bei der Implementierung die Berechnung der Punktepaare eines Patches in einen Vorverarbeitungsschritt ausgegliedert. Dadurch müssen bei einer weiteren Grobregistrierung mit dem gleichen Objekt diese Daten nicht mehr berechnet werden.

Zu Beginn der Grobregistrierung wird also überprüft, ob für das Patch bereits Vorverarbeitungsinformationen im Speicher vorliegen. Wenn diese Daten fehlen, wird versucht von der Festplatte eine Datei mit den Vorverarbeitungsinformationen zu laden. Falls auch dies fehlschlägt, müssen die Informationen neu berechnet werden. Dafür werden zunächst alle Punkte des Patches ermittelt, bei denen die Krümmung einen Null-Durchgang aufweist. Aus diesen Punkten werden Kurven benachbarter Punkte und schließlich Punktpaare extrahiert:

- Eine Kurve wird mit einem beliebigen Punkt der Liste initialisiert. Dann werden solange die jeweils nächsten Knoten angefügt, wie eine definierte Abstandschranke eingehalten wird. Analog wird am Anfang der Kurve verfahren. Wenn für ein Kurvenende keine Knoten mehr zu finden sind, beginnt man mit einer neuen Kurve, bis letztendlich keine nicht zugeordneten Knoten mehr existieren.
- Für alle daraus entstandenen Kurven werden die kürzesten Verbindungslinien zu benachbarten Kurven berechnet und die zugehörigen Endpunkte als Punktepaare gespeichert.

Für die eigentliche Registrierung werden für alle Punktpaare im ersten Patch alle Punktpaare im zweiten Patch durchlaufen. Sind die Abstände der beiden Punktpaare ähnlich lang, so wird eine weitere Punktpaarpaarung gesucht, bei der ebenfalls die Abstände näherungsweise übereinstimmen. Wurde so eine weitere Paarung gefunden, so wird nun aus diesen Punkten eine starre Transformation errechnet. Danach wird diese Transformation bewertet. Ist die Transformation besser als die beste vorherige Transformation, so wird sie nun als neue beste Transformation gespeichert.

Beschreibung des Klassendiagramms

Abbildung 6.3 zeigt die Klassen, die für das Grobregistrierungsverfahren nach Krsek benötigt werden. Der Algorithmus wird durch den Aufruf der öffentlichen Methode `start` der Klasse `CoarseRegKrsek` für zwei übergebene Scans der Klasse `Patch` gestartet. Innerhalb dieser Methode werden folgende private

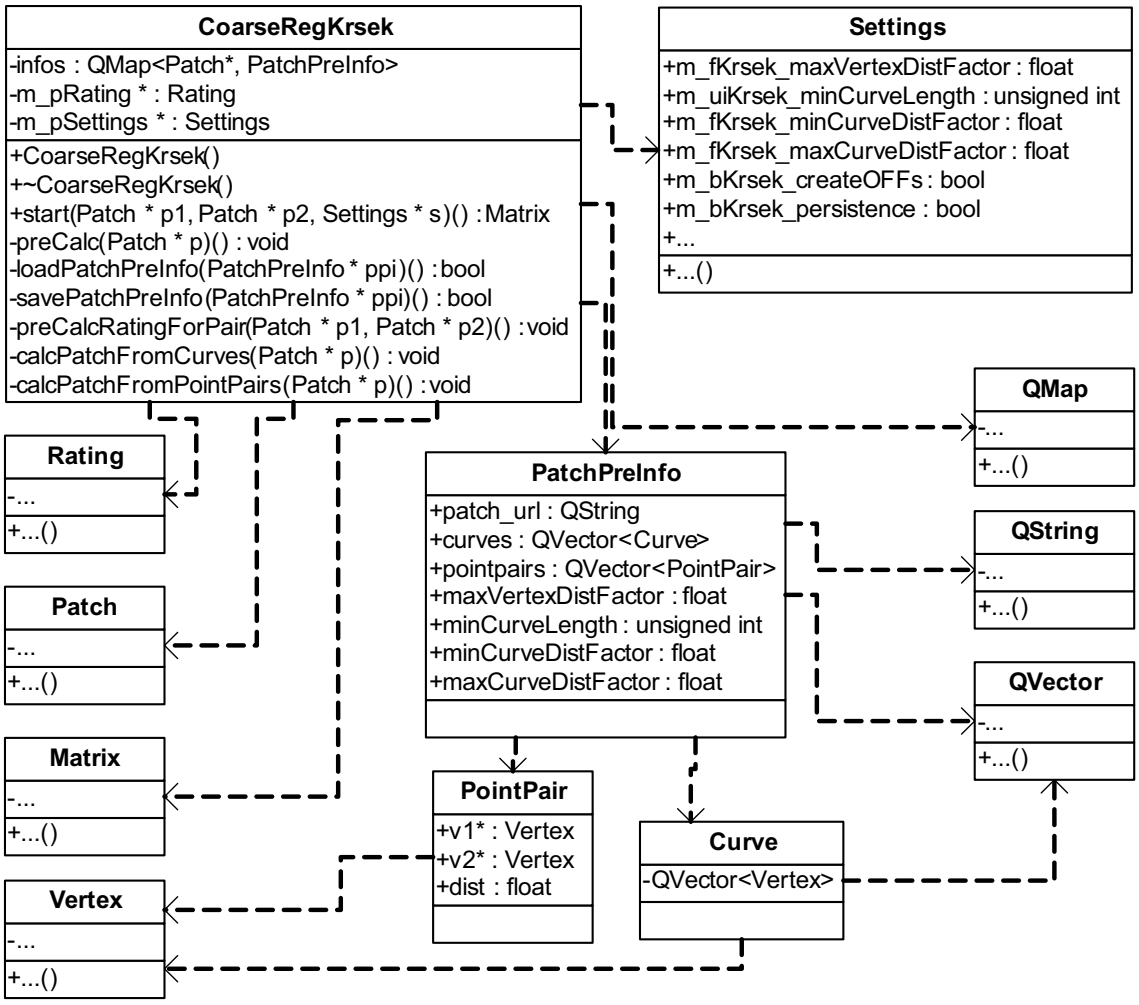

Abbildung 6.3: Auszug aus dem Registrierungswerkzeug-Klassendiagramm: Darstellung der für das Krsek-Verfahren relevanten Klassen

Methoden aufgerufen: `loadPatchPreInfo` bzw. `savePatchPreInfo` sind die Methoden, die die Vorberechnungsinformation von der Festplatte lesen bzw. auf sie schreiben. Alle Ergebnisse der Vorberechnungen werden in einem assoziativen Array `Qt::QMAP` gespeichert, in der für jedes Patch ein `PatchPreInfo`-Objekt abgelegt wird.

Durch den Aufruf von `preCalcRatingForPair` wird ein `Rating`-Objekt erzeugt und in der Membervariable `m_pRating` gespeichert. Dieses wird später dazu verwendet, um einen Registrierungsversuch zu bewerten. Die Methoden `calcPatchFromCurves` und `calcPatchFromPointPairs` erzeugen OFF-Dateien, in denen die Kurven bzw. Punktpaare dargestellt werden, so dass der Benutzer des Registrierungswerkzeuges die Zwischenergebnisse analysieren kann. Da das Verfahren an einigen Stellen durch Veränderung von Parametern durch den Benutzer beeinflusst werden kann, gibt es außerdem den Zeiger `m_pSettings` zu dem `Settings`-Objekt, in dem diese Parameter gespeichert sind.

Die Klasse `PatchPreInfo`hat keine Methoden sondern dient nur zum Speichern der vorberechneten Informationen.Die Klasse `Curve` beinhaltet die Liste der Kurven, die als Vektor von `Vertex`-Objekten gespeichert ist, und `pointpairs` die Liste der Punktepaare. Die weiteren Variablen der Klasse `PatchPreInfo` sind dafür da, um identifizieren zu können, unter welchen Parametern die Vorinformationen berechnet wurden.

Die Punktepaare werden als Objekt der Klassen `PointPair` gespeichert. Sie beinhaltet Zeiger auf die beiden Punkte des Paares und speichert den Abstand `dist` dieser Punkte.

6.1.4 A Frequency Domain Technique for Range Data Registration, Lucchese et al.

Implementierungsdetails

Das Grobregistrierungsverfahren nach Lucchese et al. bedient sich einer völlig anderen Sichtweise als alle anderen Grobregistrierungsverfahren. Durch die Betrachtung der Scans im Frequenzbereich eröffnen sich völlig neue Möglichkeiten. Das Verfahren wurde vollständig nach Lucchese et al. implementiert, allerdings erwiesen sich die Ergebnisse der Berechnung der Translation als äußerst unzuverlässig und ungenau. Allgemein durchläuft das implementierte Verfahren die folgenden vier Schritte:

- Bestimmung der Überlappungsbereiche
- Bestimmung der Rotationsachse
- Bestimmung des Rotationswinkels
- Bestimmung der Transformation

Das Verfahren nach Lucchese et al. sieht eine Bestimmung der Überlappungsbereiche nicht vor, setzt sie aber voraus. Da das Ziel des Registrierungswerkzeugs eine vollständig automatisch ablaufende Registrierung war, musste hier also eine Lösung her. Die Bestimmung der Überlappungsbereiche wird über ein abgewandeltes Spinimage Verfahren durchgeführt. Das Punktpaar mit der höchsten Korrelation ergibt das Zentrum des Überlappungsbereiches. Die Größe des Überlappungsbereiches wird nun durch die Voxelgröße bestimmt.

Um die Rotationsachse zu bestimmen müssen beide Scans zunächst in jeweils ein Voxelspaceformat gewandelt werden. Dazu dient ein dreidimensionales Array. Dazu werden die Eckpunkte der Scans in das Voxelarray projiziert. Dabei landet das vorher bestimmte Zentrum des Überlappungsbereiches in der Mitte des Voxelarrays und erzeugt hier eine Eins. Je nach Voxelgröße landen in dem Voxelarray kleinere oder größere Ausschnitte des Scans. Idealerweise sollte die Voxelgröße so bestimmt werden, dass nur die Punkte des tatsächlich überlappenden Bereiches in das Array projiziert werden.

Bis zu diesem Zeitpunkt befinden sich in den Voxelarrays ausschließlich Einsen und Nullen. Um den Einfluss von Scanungenauigkeiten zu mindern werden die beiden Voxelarrays nun geglättet. Die geschieht, indem man die Arrays mit einer Gaussfunktion faltet. Die Faltung im Ortsbereich wird durch eine komplexe Multiplikation im Frequenzbereich realisiert. Dazu werden die Voxelarrays mit Hilfe der `fftw` fouriertransformiert. Man erhält ein Array, welches jetzt nurnoch Frequenzen enthält. Multipliziert mit einem Gaussarray und nach anschließender Rücktransformation erhält man jeweils ein geglättetes Voxelspacearray für jedes Patch. Die Rücktransformation ist allerdings nicht notwendig, da ohnehin mit den Patches im Frequenzbereich weiter verfahren wird.

Die beiden Voxelspacearrays im Frequenzbereich werden nun als Grundlage für die Bestimmung der Rotationsachse genutzt. Lucchese definiert eine $\triangle(k)$ Funktion, die für jeden Vektor $\vec{k} = [x, y, z]$ einen Distanzwert berechnet. Da für viele dieser Vektoren mehrfach die Funktion aufgerufen wird, und die Vektoren aus diskreten Werten zwischen Null und der Voxelspacegröße stammen, wurden die Werte der Funktion vorberechnet und in ein dreidimensionales Array gespeichert.

Der nächste Schritt transformiert die Koordinatensysteme in sphärische Koordinaten. In der Implementierung wurde hierzu eine weitere $\triangle_S$ Funktion definiert, welche sphärische Koordinaten (ρ, ϕ, θ) annimmt und diese in kartesische Koordinaten umrechnet um letztendlich auf die vorberechneten Werte der $\triangle(k)$ Funktion zuzugreifen. Diese Funktion (in der Implementierung `DeltaS` genannt) wird benutzt um die Radialprojektion zu realisieren. Die Integration über θ zerfällt auf Grund der diskreten Arrays in eine einfache Aufsummierung entlang von θ. Die Radialprojektion erzeugt somit ein zweidimensionales Array, welches an den Indexpositionen die Winkel ρ und ϕ erwartet. Dieses Array wurde in 1-Grad-Schritten realisiert und wird vollständig berechnet. In Luccheses Artikel wird auf eine Möglichkeit hingewiesen durch geschicktes *Simulated-Annealing* eine vollständige Berechnung des Arrays zu vermeiden. Auf Grund der zahlreichen Minima und Maxima erschien diese Überlegung als unsinnig und wurde daher nicht implementiert. Während das gesamte Array berechnet wird, wird zusätzlich das globale Minimum des Arrays

berechnet. Dieses dient letztlich der Rotationsachsenberechnung, welche einfach aus der Rücktransformation der sphärischen Koordinaten des Minimums der Radialprojektion in kartesische Koordinaten ermittelt wird.

Es ist hierbei zu erwähnen, dass das Array der Radialprojektion zahlreiche lokale Minima und Maxima enthält und die Rotationsachse in den meisten Fällen leider nicht eindeutig bestimmbar ist. Dieser Umstand ist auf die Diskretisierung und die Scanungenauigkeiten zurückzuführen, da die Rotationsachse zumindest mathematisch exakt dem globalen Minimum entspricht und nicht etwa eine Näherung ist. Dieses Verhalten konnte belegt werden, sobald man zwei identische, lediglich gegeneinander verdrehte Scans registriert hat. Die Ergebnisse waren dann durchweg exakt.

Als nächster Schritt wurde die Bestimmung des Rotationswinkels implementiert. Dazu wurde eine Projektion entworfen, welche die Rotationsachse als eine der drei Basisachsen eines rechtwinkligen Koordinatensystems nutzt. Dies dient dazu, um über eine Phasenkorrelation den Rotationswinkel zu berechnen. Es werden hier zwei Funktionen (p_1 und p_2) definiert, welche die nun projizierten fouriertransformierten Voxelarraydaten entlang der Rotationsachse integriert. Diese beiden Integrationen liefern jeweils ein zweidimensionales Array. Stellt man dieses Array dar, so erhält man einen Ausreißer nach oben in der Mitte des Arrays. Die Darstellungen der beiden Arrays sind bis auf eine Rotation identisch. Auch hier wird über eine weitere Projektion in sphärische Koordinaten und Integration über die Länge des Polarkoodinatenvektors das Problem der Rotationswinkelbestimmung um eine Dimension vereinfacht. Die Werte beider Funktionen sind nun also zwei eindimensionale Arrays.

Mittels Fouriertransformation wird nun erneut in den Frequenzbereich übergegangen. In dem Artikel von Luccheses et al. wird der Rotationswinkel über eine Phasenkorrelation bestimmt. Vor der Projektion und Fouriertransformation lässt sich der Winkel ablesen durch eine geeignete Rotation des p_1-Arrays, so dass es mit dem anderen p_2 Arrays identisch ist. Nach der Projektion und Fouriertransformation lässt sich der Winkel ablesen, in dem man eines der beiden sich ergebenen eindimensionalen Arrays verschiebt, bis es sich mit dem anderen Array deckt. Dieser Betrag der Verschiebung entspricht dem Rotationswinkel. Die Ergebnisse der Phasenkorrelation waren höchst unzuverlässig und wurden nachdem sie fertig implementiert wurden deaktiviert. Stattdessen wird der Rotationswinkel nun durch Vergleich der beiden Maxima beider Arrays errechnet. Der Abstand der Maxima entspricht mit guter Näherung dem Rotationswinkel.

Weiterhin wird im Artikel die Translation bestimmt. Nachdem diese im Registrierungswerkzeug keine brauchbaren Ergebnisse lieferte und die Verschiebung unabhängig von der Genauigkeit der errechneten Translation ohnehin nur innerhalb der Überlappungsbereiche stattfindet, wurde auf eine Optimierung und weitergehende Analyse verzichtet.

Als Ergebnis der Frequenzregistrierung wird eine Matrix bestimmt, welche die Zentren beider Überlappungsbereiche zusammenzieht und den zweiten Scan um den berechneten Rotationswinkel um die berechnete Rotationsachse rotiert.

Klassendiagramm-Beschreibung

(vgl. Abbildung 6.4)

Die Klasse `CoarseReg_freq` führt die Frequenzregistrierung durch. Die einzige öffentliche Methode ist die Methode `start(...)`. Diese Methode erwartet Zeiger auf die beiden Patches und auf die Settingsklasse, aus der weitere Parameter übernommen werden. Die Methode berechnet mit Hilfe der Funktion `OnFindOverlap(...)`, welche die beiden Scans als Parameter erwartet, die beiden Zentren der Überlappungsbereiche und gibt diese zurück. Anschließend wird die `RSDiff` Klasse instanziiert, welche im Konstruktor die beiden Patches und die beiden Zentren, sowie die Auflösung des Voxelarrays übernimmt. Diese Klasse instanziiert ihrerseits zwei Objekte der Klasse `Rangescan`. Von dem Objekt der `RSDiff` Klasse werden nun nacheinander die beiden Funktionen `FillPField()` und `CreatePNFields()` aufgerufen. Die Funktion `FillPField()` berechnet die Funktion $\triangle(k)$ und die Radialprojektionen und damit die Funktion $P(\phi, \theta)$. Daraus ergibt sich bereits die Rotationsachse. Der Winkel wird über die Funktion `CreatePNFields()`berechnet. Dabei werden die Funktionen p_1 und p_2 berechnet und anschließend daraus der Rotationswinkel. Zuletzt wird in der Methode `start(...)`die Matrix berechnet, welche das

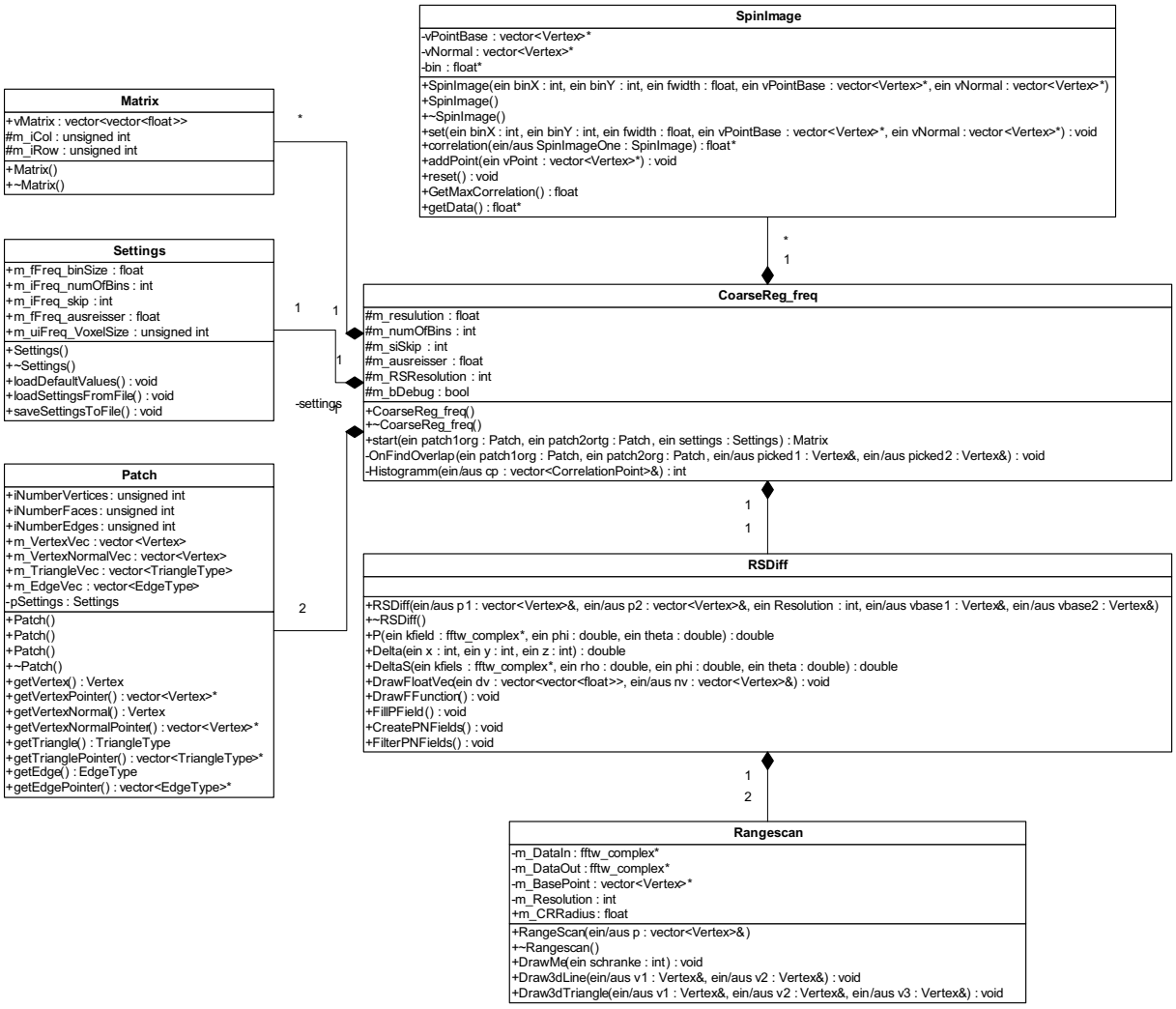

Abbildung 6.4: Auszug aus dem Registrierungswerkzeug-Klassendiagramm: Darstellung der für das Frequenz-Registrierungs-Verfahren relevanten Klassen

Zusammenziehen beider Zentren und die Rotation um den berechneten Rotationswinkel bewirkt und an das Reduktionswerkzeug zur Berechnung der Bewertung übergeben.

6.1.5 Mutual Information Registration, Josien P.W. Pluim

Implementierungsdetails

Das Mutual Information Registration Verfahren nutzt Erkenntnisse aus der Informationstheorie um eine Registrierung zwischen zwei Scans durchzuführen. Es ist ein randomisiertes Verfahren, so dass die Qualität des Ergebnisses sehr stark von der gewählten Anzahl der Stichproben und den dazugehörenden Transformationen abhängt. Das Verfahren erwartet als Eingabe zwei Abtastungen A und B, die registriert werden sollen, sowie die Anzahl an Stichproben x und den Skalierungsfaktor s. Aufgrund der Beschränkung auf zweidimensionale Bilder ergibt sich folgender Ablauf für die Registrierung:

- Parametrisierung von Abtastung A, liefert A' skaliert mit dem Faktor s.
- Parametrisierung von Abtastung B, liefert B' skaliert mit dem Faktor s.

- Berechnung des Entropiewertes von A'
- Berechnung des Entropiewertes von B'
- Bestimmung von x gefordeten Stichproben, bestehend aus einer zufälligen Translation und einer zufälligen Rotation von B', mit der Einschränkung, dass nach der Transformation B' einen nichtleeren Überlappungsbereich mit A' hat.
- Bestimmung der wechselseitigen Informationen für jede der x Stichproben.
- Rückrechnung der besten zehn Stichproben mit der maximalen Bewertung der wechselseitigen Informationen zu einer Transformation im dreidimensionalen Raum
- Bewerten der zehn Transformationen mit Hilfe der Bewertungsfunktion für Registrierungen
- Rückgabe der Transformation mit der besten Bewertung

Der Parameter x, der die Anzahl der Stichproben darstellt, kann vom Anwender frei festgelegt werden. Aufgrund der Randomisierung steigt die Wahrscheinlichkeit auf eine gute Stichprobe mit großem x an. Die Wahl eines Wertes für x ist daher stets ein Kompromiss zwischen Rechenzeit und der Wahrscheinlichkeit eine gute Stichprobe zu finden. Ein weiterer Einflussfaktor besteht durch die Skalierung der Bilder mit dem Faktor s. Dieser Wert wird ebenfalls vom Anwender festgelegt. Stark verkleinerte Bilder verringern die Rechenzeit des Verfahrens beträchtlich, führen aber zu Ungenauigkeiten. Große Bilder liefern präzisere Ergebnisse allerdings auf Kosten der Rechenzeit.

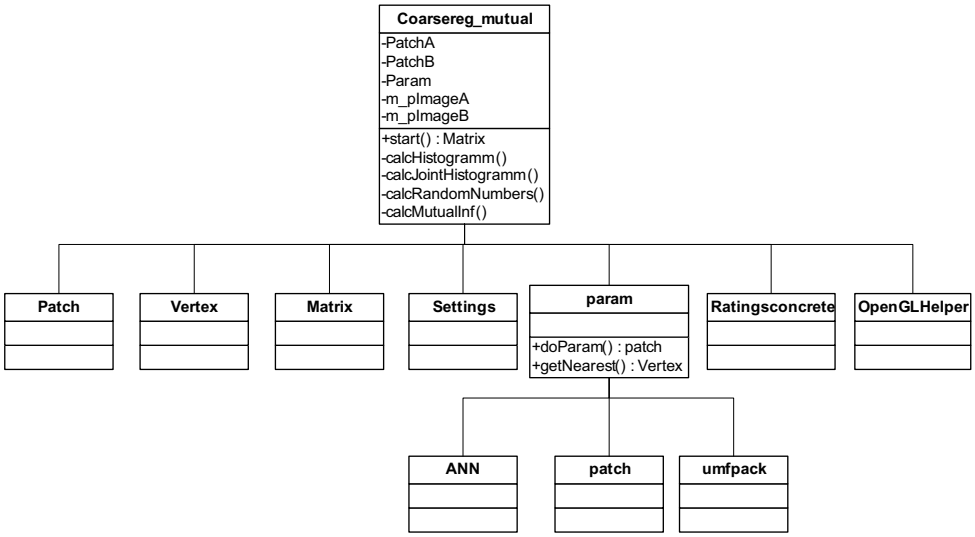

Abbildung 6.5: Auszug aus dem Registrierungswerkzeug-Klassendiagramm: Darstellung der für das Mutual Information Registration-Verfahren relevanten Klassen

Beschreibung des Klassendiagramms

Abbildung 6.5 zeigt die Klassen, die von dem Verfahren verwendet werden. Das Verfahren wird über die Methode `start` der Klasse `coarsereg_mutual` aufgerufen und liefert als Ergebnis eine Transformationsmatrix zurück.

6.1.6 Feature based matching of triangular meshes, Heinrich Müller

Implementierungsdetails

In diesem, an [12] angelehnten Verfahren, ist die Grundvoraussetzung zur Bestimmung der überlappenden, d. h. aufeinander passenden Bereiche, die Krümmung an den einzelnen Knoten der Scans sowie deren

Umgebung.

Einsprungpunkt des Algorithmus ist die Methode `start` der Klasse `coarsereg_mueller`. Der Methode werden die Zeiger auf die beiden Scans sowie die Einstellungen übergeben.

Nun wird zuerst die Menge der infrage kommenden Punkte mit der Methode `getsignificantcurvature` reduziert. Es werden nur Punkte akzeptiert, die ein bestimmtes Mindestmaß an Krümmung aufweisen. Dies geschieht für beide Scans, die Ergebnisse werden im Datentyp `vector` gespeichert.

Anschließend wird aus diesen Ergebnissen eine Menge an möglichen Dreickspaaren mittels der Methode `findmatching_cv`, wie in 3.1.7 beschrieben, berechnet. Die Dreieckspaare werden durch 2×3 Punkte dargestellt, wobei man die Transformation für die Scans bestimmt, indem man die Transformation für ein Dreieck auf das andere mittels der Klasse `openglhelper` berechnet. Jede dieser Berechnungen liefert eine Transformationsmatrix, und jede Möglichkeit wird nacheinander berechnet und mittels der Klasse `rating` bewertet. Dies realisiert die Methode `findbestmatching`. Zu Beginn ist die Bewertung der ersten Transformation der Referenzwert. Dieser wird immer dann ersetzt, wenn eine Transformation eine bessere Bewertung liefert. So wird am Ende die Transformation, d. h. deren Transformationsmatrix, mit der besten Bewertung als Ergebnis der Registrierung zurückgegeben.

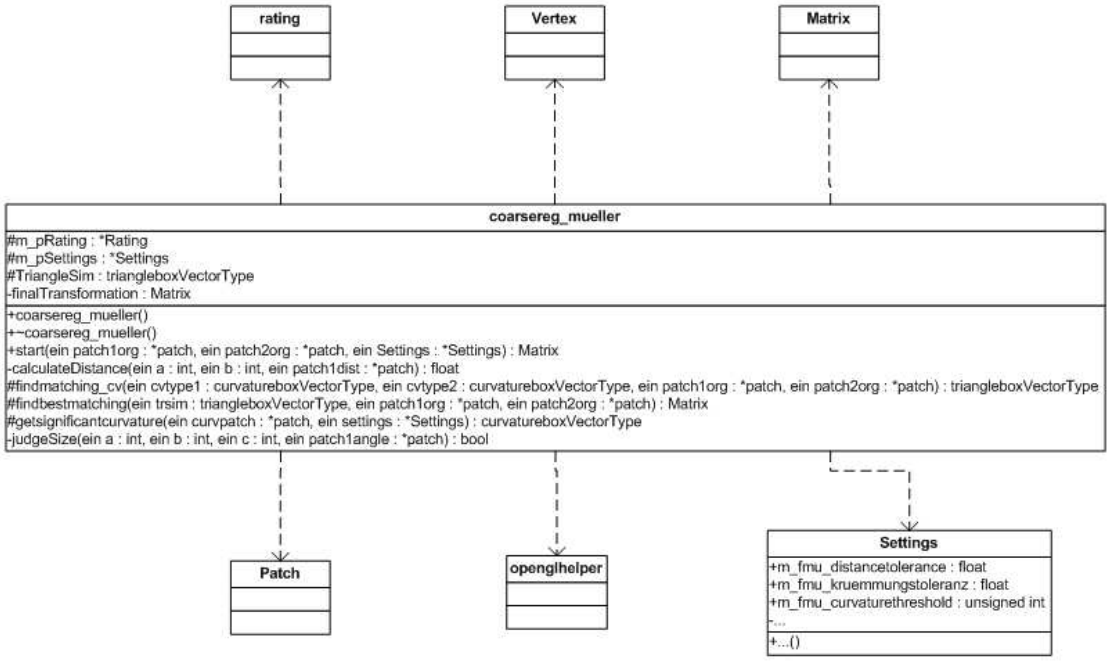

Abbildung 6.6: Auszug aus dem Registrierungswerkzeug-Klassendiagramm: Darstellung der für das Verfahren nach Müller relevanten Klassen

Beschreibung des Klassendiagramms

Das Klassendiagramm (Abbildung 6.6) zeigt die für das Verfahren wichtigen Klassen. Um das Verfahren zu starten wird die `start`-Methode der Klasse `coarsereg_mueller` aufgerufen. Während der Registrierung wird die Klasse `patch` und ihre Methoden benötigt, um auf die Scans und deren Eigenschaften zugreifen zu können. Um Zugriff auf die einzelnen Knoten der Patches zu haben, wird die Klasse `vertex` eingebunden. Des Weiteren wird für die Bewertung einer Transformation die Klasse `rating` benötigt und für Matrixoperationen die Klasse `Matrix`. Um daraus die Transformation der Scans zu generieren, wird die Klasse `openglhelper` benötigt. Zuletzt ist die Klasse `settings` vonnöten, um Zugriff auf die Registrierungseinstellungen zu haben.

6.1.7 Correlation of Distances to Neighbors, T. Hüntler, D. Frenzel

Die grobe Vorgehensweise dieser Grobregistrierung wurde schon in Kapitel 3.1.8 erläutert, soll hier aber näher erklärt werden.

Implementierungsdetails

Folgende Schritte werden durchgeführt:

- Finden von Punkten mit einem Krümmungswert innerhalb der Werte α und β. Dies wird für beide Netze durchgeführt.
- Berechnung der Distanzen der ausgewählten Punkte zu ihren Nachbarn der Stufe eins und zwei.
- Berechnung ähnlicher Punkte anhand ihrer Distanzen.
- Finden von ähnlich langen Kanten in Netz eins und zwei.
- Finden von ähnlich großen Dreiecken in Netz eins und zwei.
- Bestimmung der besten Transformation anhand einer Bewertungsfunktion.

Im ersten Schritt werden Punkte mit ähnlicher Krümmung aus Netz eins und zwei ausgewählt. Wenn die Werte innerhalb der Grenzen α und β liegen, werden die Punkte in je einen Vektor für Netz eins und Netz zwei gespeichert.

Nach der Auswahl der Punkte werden die Distanzen des jeweiligen Punktes zu seinen Nachbarpunkten berechnet. Die Werte werden als sortierte Vektoren am Punkt gesichert. Es existiert je ein Vektor für Stufe eins und Stufe zwei Nachbarn.

Der dritte Schritt ist die Bestimmung ähnlicher Punkte aus den zuvor berechneten Punkten. Dazu wird jeder Punkt aus Netz eins gegen jeden Punkt aus Netz zwei getestet. Verglichen wird zuerst die Anzahl der Nachbarn. Wenn sich der Wert unterscheidet wird der betreffende Punkt aus Netz zwei verworfen. Bei gleicher Nachbaranzahl werden die Distanzen aufsteigend verglichen. Wenn die Distanzen der Punkte nicht, oder nur kaum, voneinander abweichen, wird eine Bewertung an Punkt eins gespeichert. Diese Bewertung setzt sich aus dem Quadrat der Distanzabweichungen der zwei getesteten Punkte zusammen. Zusätzlich wird noch der Index des getesteten Punktes aus Netz zwei gesichert. Wenn sich die Distanzen der Punkte stark unterscheiden wird der getestete Punkt aus Netz zwei verworfen. Bei jedem Test wird die aktuelle Bewertung mit der gespeicherten verglichen. Der jeweils kleinste Wert wird gesichert. Wenn sich der Wert der Bewertung ändert wird gleichzeitig der Index des getesteten Punktes aktualisiert. Nachdem alle Punkte getestet wurden, wird ein Vektor aus Punktepaaren erstellt. Dazu wird der Vektor der ausgewählten Punkte aus Netz eins durchlaufen. An jedem Punkt steht der Index seines korrespondierenden Punktes in Netz zwei. Dadurch ergibt sich ein Punktepaar.

Die Punktepaare werden dann sukzessive zu Dreiecken zusammengesetzt, so daß in beiden Netzen zwei Dreiecke entstehen, bei denen die Eigenschaften der Eckpunkte (also die Distanzen zu den Nachbarn) des Dreiecks in Netz mit den Eigenschaften der Eckpunkte des Dreiecks in Netz zwei übereinstimmen. Außerdem wird sichergestellt, daß beide Dreiecke eine sehr ähnliche Geometrie haben, also annähernd deckungsgleich sind. Für jede der dabei entstehenden Dreicks-Paarungen wird dann die Transformationsmatrix bestimmt und bewertet. Die Transformation deren Matrix die beste Bewertung liefert wird als Ergebnis der Registrierung zurückgegeben.

Beschreibung des Klassendiagramms

Wie in der Abbildung 6.7 zu erkennen besteht das CDN-Verfahren aus einer Klasse `coarsereg_sun` und nutzt vorhandene Klassen des Registrierungs-Moduls.

Die Klasse `coarsereg_sun` enthält nur eine Methode. Diese Methode `start` der Klasse beinhaltet alle Berechnungen des Verfahrens. Übergabeparameter sind zwei Instanzen *patch*1 und *patch*2 der Klasse

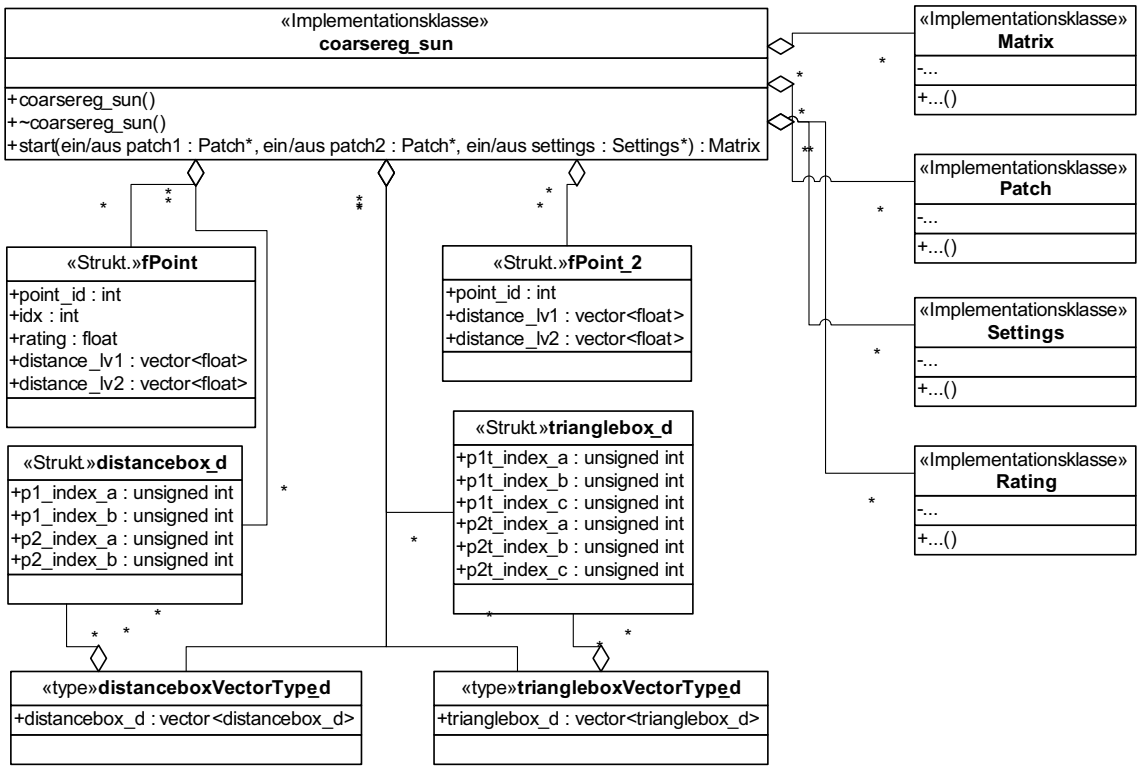

Abbildung 6.7: Auszug aus dem Registrierungswerkzeug-Klassendiagramm: Darstellung der für das CDN-Verfahren relevanten Klassen

`Patch`. Die Instanzen enthalten alle Daten eines Dreiecksnetzes inklusive der Krümmung. Ein weiterer Parameter ist eine Instanz der Klasse `Settings`. Rückgabe der Methode ist eine Transformationsmatrix. Diese Matrix ist eine Instanz der Klasse `Matrix`.

6.1.8 Registering Two Overlapping Range Images, Gerhard Roth

Für die Grobregistrierung nach Roth wird zunächst aus beiden Scans eine begrenzte Teilmenge an Punkten bestimmt. Die Anzahl der Punkte für den ersten Scan ist über die Registrierungoptionen festgelegt. Für diese Teilmenge werden Punkte mit der höchsten Krümmung genommen und die untere Krümmungsschranke bestimmt. Anhand dieser Krümmungsschranke werden die Punkte des zweiten Scans bestimmt. Die beiden Punktmengen werden mit Hilfe der Bibliothek *TetGen* delaunay trianguliert. Die sich ergebenen Dreiecke der Tetraeder werden in einem Vektor gespeichert und nach den Kantenlängen sortiert. Aus diesen beiden Vektoren werden Paare von Dreiecken bestimmt, die sich ähnlich sind. Diese Dreieckspaare werden ebenfalls in einem Vektor gespeichert. Im letzten Schritt wird dieser Vektor durchlaufen und die Dreieckspaare werden aufeinander transformiert und bewertet. Die Transformation mit der besten Bewertung wird zurückgegeben.

Klassendiagramm

siehe Abbildung 6.8

Beschreibung des Klassendiagramms

Die Klasse `CoarseReg_Roth` führt die Grobregistrierung nach Roth durch. Die einzige öffentliche Methode ist `start`. Diese startet den Grobregistrierungsalgorithmus. Der Methode werden zwei Zeiger auf

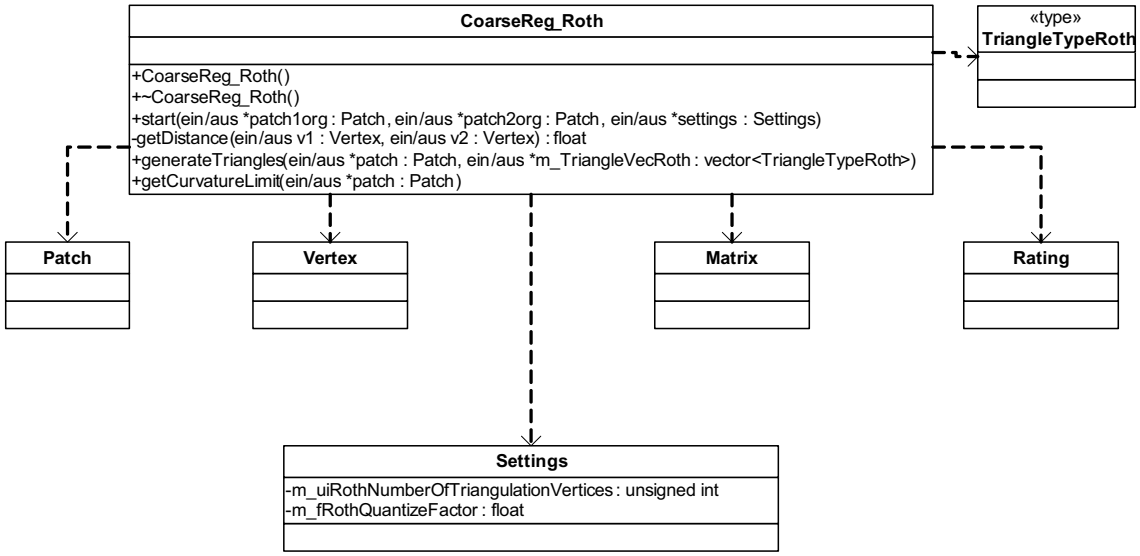

Abbildung 6.8: Auszug aus dem Registrierungswerkzeug-Klassendiagramm: Darstellung der für das Roth-Verfahren relevanten Klassen.

ein Objekt der Klasse `Patch` und ein Zeiger auf ein Objekt der Klasse `Settings` übergeben. Über die Methode `getCurvatureLimit` wird die Krümmungsschranke für den zweiten Scan bestimmt. Als Eingabe wird ein Zeiger auf ein Objekt der Klasse `Patch` erwartet. Die Delaunay-Triangulierung und somit die Dreiecke, werden für jedes Patch über die Methode `generateTriangles` bestimmt. Die Dreiecke werden in dem Vektor `m_TriangleVecRoth` vom Typ `TriangleRothType` gespeichert. Die Distanzen zwischen zwei Punkten werden über die Methode `getDistance` bestimmt. Diese erwartet als Eingabe zwei Punkte vom Typ `Vertex`. Die Anzahl der zu triangulierenden Punkte wird in der Klasse `Settings` als `m_uiRothNumberOfTriangulationVertices` ausgelesen. `m_fRothQuantizeFactor` gibt den Quantisierungsfaktor vor, mit der alle Distanzen zwischen zwei Punkten quantisiert werden.

6.1.9 The Normal Distributions Transform, Peter Biber, Wolfgang Straßer

Die Implementierung des Verfahrens von Peter Biber und Wolfgang Straßer [4] wurde nach den Vorgaben der Autoren implementiert und dementsprechend angepasst, so dass es auch auf dreidimensionalen Objekten arbeitet (siehe Kapitel 3.1.10). Das Verfahren gliedert sich in folgende Schritte:

- Einteilung des parametrisierten Abtastung in ein Raster von Zellen, für die jeweils die Normalverteilung der darin befindlichen Knoten berechnet wird.
- Berechnung verschiedener Bewertungen für partielle Ableitungen.
- Bestimmung des neuen Parameters t, indem die Funktion $f = -score(t)$ maximiert wird.
- Berechnung der Transformationsmatrix aus den Parametern in t.

Beschreibung des Klassendiagramms

Die Klasse `CoarseReg_ndt` (siehe Abbildung 6.9) führt die Grobregistrierung nach Biber und Straßer durch und wird durch den Aufruf der öffentlichen Methode `start` für zwei übergebene Abtastungen der Klasse `Patch` gestartet. Zusätzlich gibt es den Zeiger `m_underline_pSettings` zu dem `Settings`-Objekt, in dem die vom Benutzer veränderbaren Parameter gespeichert sind. Die `init_trafo`-Methode initalisiert die Transformation, indem die Mittelpunkte der Abtastungen übereinandergeschoben und um einen zufälligen Winkel gedreht werden. Die Bewertung der Transformation liefert die Methode `score`.

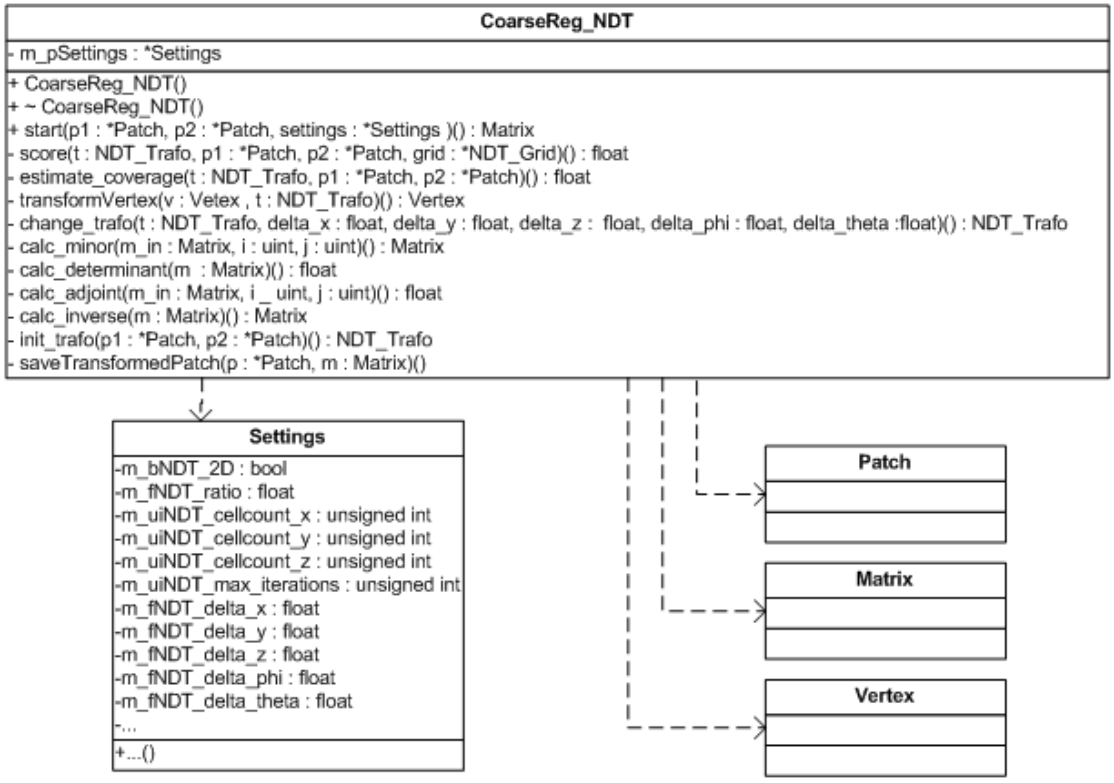

Abbildung 6.9: Auszug aus dem Registrierungswerkzeug-Klassendiagramm: Darstellung der für das NDT-Verfahren relevanten Klassen

Die Methoden `calc_minor`, `calc_determinante` und `calc_adjoint` werden zur Berechnung der inversen Matrix benötigt, welche die Methode `calc_inverse` liefert.

6.1.10 ICP

Implementierungsdetails

Die Implementierung der Feinregistrierung baut sich im Wesentlichen genauso auf, wie in Kapitel 3.2.2 beschrieben. Zuerst wird der Klasse `RefinedReg` durch die Settingseinstellungen `Settings` mitgeteilt, auf wieviel Prozent der Punktanzahl eines Scans sich die Ermittlung der Feinregistrierung beziehen soll. Im Gegensatz zu einer festen Wahl an Knoten kann somit der Benutzer selbstständig entscheiden, welchen Prozentwert er für seine Untersuchungen bevorzugt. Empfohlen wird ein Wert um 10%, dieser hat sich in experimentellen Untersuchungen als hinreichend dargestellt. Der Klasse liegen neben der Settingseinstellungen die Informationen zu Grunde welche Scans verwendet werden und welche Transformationsmatrix die Grobregistrierung geliefert hat. Die Feinregistrierung nimmt nun eine Knotenreduktion durch randomisierte Knotenauswahl (Anzahl der entnommenen Knoten stammen aus der Settingseinstellung) aus einem Scan vor. Danach wird nach der beschriebenen k-d-Baum-Suche die nächsten Nachbarn aus dem zweiten Scans ermittelt. Die k-d-Baum-Suche stützt sich auf die ANN-Bibliothek (vgl. Anhang C und Methode C.2 für die Suche der nächsten Nachbarn). Daraufhin startet die eigentliche Distanzminimierung der beiden Scans durch die Horn-Metrik. Für detailierte Angaben und mathematischen Hintergrund sei auf Kapitel 3.2.2 verwiesen. Die Methode `errorMetricHorn(...)` benötigt die beiden Punktlisten, vom Typ `VertexVectorType`, einem Typ, das ein Vektor aus Punkten (*Vertex*) darstellt. Die Methode berechnet die Zentroiden zusammen mit der Kreuzkorrelation. Für die Quaternionen wird der Eigenvektor ermittelt durch den Jacobi-Algorithmus, anschließend wird aus den Quaternion, die Rotationsmatrix ermittelt durch eine einfache Transformationsgleichung.

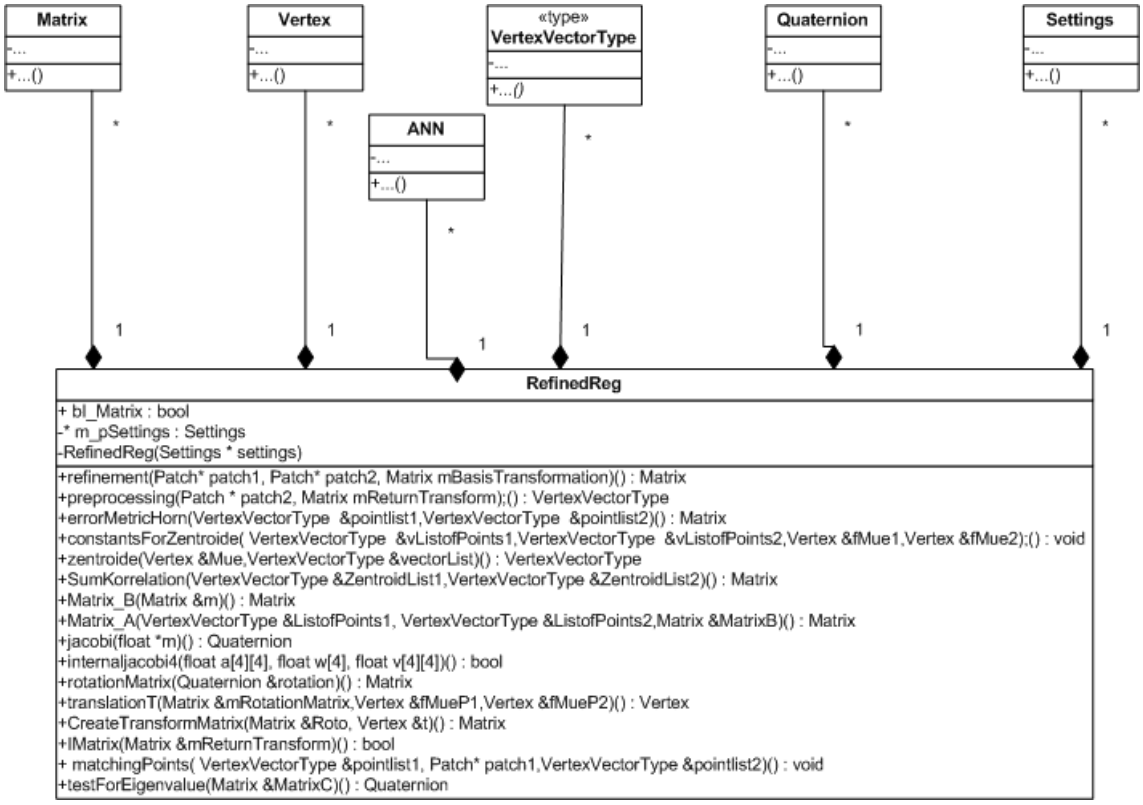

Abbildung 6.10: Auszug aus dem Registrierungswerkzeug-Klassendiagramm: Darstellung der für das ICP-Verfahren relevanten Klassen.

Beschreibung des Klassendiagramms

Wie in Abbildung 6.10 zu erkennen, sind die wesentlichen Klassen, die die Feinregistrierung verwendet: Die Klassen `Matrix`,`Vector`, `Settings`, `Quaternion` und `ANN`. Der Aufruf der Methoden dieser Klasse geschieht nach folgenden Schritten:

- **RefinedReg(Settings * settings)** Dies ist der Konstruktor mit Übergabe der Settingseinstellungen, in denen ein Prozentwert angegeben wird für die Ermittlung der randomisierten Punkte aus einem Patch,

- **Matrix refinement(Patch* patch1, Patch* patch2, Matrix mCoarseRegTransformation)** Aufruf der ersten Methode `refinement (...)` mit Übergabe der beiden zu feinregistrierenden Scans *patch1* und *patch2* vom Typ `Patch`.

- **VertexVectorType preprocessing(Patch * patch2, Matrix mReturnTransform)** Die Operation `preprocessing(..)` ist für die randomisierte Punktauswahl aus *patch1* zuständig und legt die herausgegriffenen Punkte in einem `VertexVectorType` ab.

- **matchingPoints(VertexVectorType &pointlist1, Patch patch1, VertexVectorType &pointlist2)** Diese Methode bestimmt mit Hilfe eines k-d-Baumes die nächsten Punkte aus *patch2*, die zu den ausgewählten Punkten *pointlist1* den geringsten Abstand aufweisen. Alle gefundenen Knoten werden in einer neuen Liste *pointlist2* vom Typ `VertexVectorType` gespeichert.

- **Matrix errorMetricHorn(VertexVectorType &pointlist1,VertexVectorType &pointlist2)** Die Hauptmethode in der die Fehlerberechung und Distanzminimierung der beiden Patches vereint sind.

ErrorMetricHorn

Die Methode `ErrorMetricHorn` arbeitet sehr effizient zur Ermittlung der Transfomationsmatrix, um zwei Scans so gut wie Möglich zueinander zu verschieben. Im Folgenden soll näher auf die Implementierung bzgl. des Klassendiagramms eingegangen werden.

- `constantsForZentroide(pointlist1,pointlist2,fMueP1,fMueP2)`:

  Die Operation `constantsForZentroide(...)` ist zusammen mit der folgenden Methode für die Zentroidenberechnung notwendig (vgl. Kapitel 3.2.2).

- `VertexVectorType zentroide(fMueP1,pointlist1)`:

  Für beide ermittelten Punktlisten aus *patch1* und *patch2* werden nun zwei neue Punktlisten vom Typ `VertexVectorType` erstellt, die die Zentroiden für alle Punkte darstellen.

- `SumKorrelation(vZentroidP,vZentroidQ)`:

  Diese Methode beschreibt die Summenkorrelation berechnet aus den ermitteleten Zentroiden-Listen.

- `testForEigenvalue(MatrixA)`:

  Diese Methode ist für die Eigenwertberechnung und Eigenvektorbestimmung des kleinsten Eigenwertes da. Die Ermittlung der Eigenwerte ist wichtig, um auf die Rotationsmatrix zu schliessen, nach Horn reicht es aus, die Eigenwerte zu berechnen und deren kleinster Eigenvektor stellt die Rotationsmatrix dar in Form eines Quaternions (Beweis und Vorgehen siehe 3.2.2)

- `rotationMatrix(r)`:

  Die Rotationsmatrix lässt sich aus dem ermittelten Quaternion zurücktransformieren.

- `translationT(rotationsMatrix,fMueP1,fMueP2)`:

  Aus der Rotationsmatrix rotationsMatrix und zwei Konstanten, die man aus der Zentroidenberechnung ermittelt hat *fMueP1* und *fMueP2* lässt sich durch einfache Rechnung die Translation bestimmen. Dazu wird *fMueP1*, die durchschnittliche Position des ersten Scans, mit der Rotationsmatrix multipliziert und anschliessend von *fMueP2* subtrahiert.

- `CreateTransformMatrix(rotationsMatrix,trans)`: Diese Methode berechnet die homogene 4×4 Transformations-Matrix.

6.1.11 Globalregistrierung nach Huber und Hebert

Die Implementierung dieses Verfahrens basiert auf der Idee von Huber und Hebert einen minimalen Spannbaum-Algorithmus zur Auswahl der paarweisen Registrierungen zu verwenden. Grundlage der Implementierung bildet ein einfacher minimaler Spannbaum-Algorithmus, was bei Huber und Hebert als ,,Min-Span" Version bezeichnet wird. Anders als im ursprünglichen Verfahren von Huber und Hebert wurde aber weder der Oberflächenkonsistenztest mittels ,,Free-Space-Violation" noch das Fehlerverteilungsverfahren nach Neugebauer implementiert. Dafür wurden eine optionale ICP-Optimierung aller verwendeten Registrierungen sowie ein optionales Fehlerverteilungsverfahren nach Sharp, Lee und Wehe implementiert.

Folgende Schritte werden durchgeführt:

- Sortieren der Kanten nach ihrer Fehlerbewertung.
- Berechnung eines minimalen Spannbaums.
- Wenn die ICP-Optimierung durchgeführt werden soll, werden alle benutzten Kanten feinregistriert.
- Wenn die Fehlerverteilung durchgeführt werden soll, wird die Fehlerverteilung in Kreisen gestartet.
- Setzen der globalen Transformationsmatrizen.

Schritt eins erfolgt mit dem Sortieralgorithmus für Vektoren aus der STL-Bibiliothek.

Schritt zwei berechnet einen Spannbaum nach Prim mit dem Scan 0 als Wurzelknoten.

In Schritt drei wird abhängig von der Variable *m_bGlobReg_DoFine* in *m_pSettings* die ICP-Feinregistrierung auf alle Kanten mit der Eigenschaft *usedInGlobalReg = true* angewendet.

In Schritt vier wird die Fehlerverteilung in Kreisen durchgeführt, wenn die Variable *m_bGlobReg_CDE* in *m_pSettings* gleich *true* ist.

Im fünften und letzten Schritt der Globalregistrierung von Huber und Hebert werden mittels der Methode `setGlobTransMats()` die Transformationen aller Scans bezüglich des durch das Wurzelelement gegebenen globale Koordinatensystems gesetzt, um die Übertragung der Punkte aller Scans in das globale Koordinatensystem zu ermöglichen.

Klassendiagramm Globalregistrierung

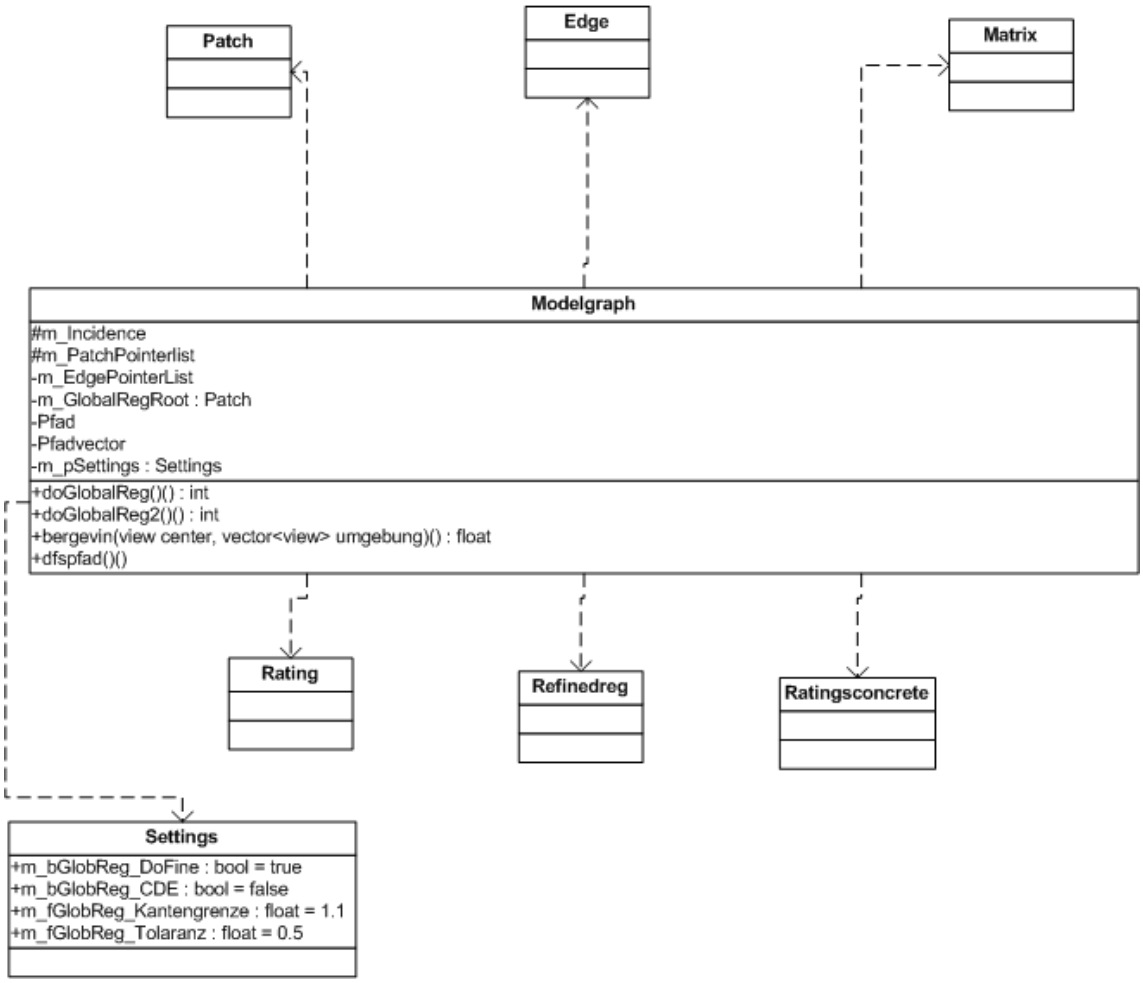

Abbildung 6.11: Auszug aus dem Registrierungswerkzeug-Klassendiagramm: Darstellung der für die Globalregistrierung relevanten Klassen.

Beschreibung des Klassendiagramms

Die Abbildung 6.11 zeigt die für die Globalregistrierung benötigten Klassen. Die Globalregistrierung wird mit der öffentlichen Methode `doGlobalReg` für das Huber und Hebert Verfahren bzw. mit `doGlobalReg2`

für das Pulli Verfahren gestartet. Beide Methoden greifen auf die *edge-pointer* Matrix *m_Incidence* zu um die Daten der Transformationen und Patches zu erhalten. Ein Zeiger auf ein Objekt von Typ `Settings` erlaubt es den Methoden auf die Werte der einstellbaren Parameter zuzugreifen. Objekte vom Typ `Matrix` werden zur Berechnung von Transformationen in den Fehlerverteilungsverfahren eingesetzt.

6.1.12 Fehlerverteilung nach Sharp, Lee und Wehe

Die Implementierung für die Fehlerverteilung in Kreisen, welches im Kapitel 3.3.3 beschrieben wird, richtet sich weitestgehend an der in dem Artikel von Sharp, Lee und Wehe [33] vorgestellten Idee zur Fehlerverteilung in kreisförmigen Grafen und die Erweiterung dieses Verfahrens auf spannbaumförmige Grafen.

Für die Fehlerverteilung erfolgen die Schritte:

- Berechnen aller Pfade von der Wurzel zu den Blättern des Spannbaums.
- Erzeugen von Kreisen aus diesen Pfaden.
- Berechnung der Fehlermatrix für jeden Kreis und gleichmässiges Verteilen des Fehlers auf alle Kanten des Kreises.

In Schritt eins der Fehlerverteilung werden mittels der Methode `dfspfad()` durch rekursive Tiefensuche alle Pfade von der Wurzel zu den Blättern bestimmt und in einem Vektor gespeichert.

In Schritt zwei werden je zwei Pfade aus diesem zu einem Kreis verbunden, wenn es zwischen den beiden Blättern eine Kante gibt deren Fehler unterhalb des Parameters *fGlobReg_Kantengenze* in *m_pSettings* liegt. Es werden bevorzugt zwei Pfade verbunden die noch in keinem Kreis liegen, um die Anzahl der Kreise gering zu halten. Wenn alle Pfade in mindestens einem Kreis liegen oder keine Kreise mehr gebildet werden können beginnt die eigentliche Fehlerberechnung und -verteilung.

Im letzten Schritt der Fehlerverteilung wird für jeden Kreis eine Transformation von der Wurzel über alle Kanten des Kreises zurück zur Wurzel berechnet. In Idealfall ist das Ergebnis die Identitätsmatrix und der Fehler gleich null. Ist dies nicht der Fall wird eine zur Ergebnistransformation inverse Transformation berechnet und die Matrix diese Transformation mit einem Skalar multipliziert, das sich als $\frac{1}{AnzahlKreiskanten}$ ergibt. Diese Korrekturmatrix wird jetzt zu alle Matrizen des Kreises zellenweise addiert, wobei auf die Ausrichtung der Transformation in der Kante geachtet wird.

6.1.13 Globalregistrierung nach Pulli

Die Implementierung des Pulli Globalregistrierungsverfahrens entspricht weitestgehend dem im Artikel von Pulli [29] beschriebenen Vorgehen. Lediglich bei der Auswahl des neu hinzuzufügenden Elements gibt es eine Abweichung. Das Verfahren ist langsamer als die Basisversion der Huber und Hebert Implementierung, allerdings ist das eingesetzte Fehlerverteilungsverfahren universeller.

Zunächst wird für jeden Knoten des Modellgraph der Knotengrad bzgl. der verbindenden Kanten bestimmt und diese Information zusammen mit dem Index des zum Knoten gehörenden Scans in einem Element von Typ `view` gespeichert. Eine Kante ist eine Verbindung, wenn ihr Fehlerwert kleiner ist als der Parameter *Kantengrenze* und sie nicht in der Gruppe *No Match* ist. Aus diesen `view`-Elementen wird das Element mit dem höchsten Knotengrad bestimmt und der zugehörige Patch als *Globalregroot* gesetzt. Dieser Patch bestimmt später das globale Koordinatensystem als Wurzelelement des Modellbaums. Das gewählte Element wird aus der Menge der noch nicht gewählten entfernt und in die Menge der bereits gewählten Ansichten eingefügt. Beide Mengen werden dabei durch Vektoren von `view`-Elementen dargestellt. Für eine neu ausgewählte Ansicht werden alle mit ihr nach den oben beschriebenen Kriterien verbundenen Ansichten ebenfalls hinzugefügt und die hinzugefügten aus dem Vektor noch nicht verwendeter Ansichten entfernt. Die neu ausgewählte Ansicht wird in eine Queue gespeichert, welche zur Steuerung der While-Schleife für die Fehlerverteilung benutzt wird. Innerhalb der Fehlerverteilung werden für das vorderste Element der Queue alle verbunden Knoten („Nachbarn") innerhalb der bereits ausgewählten

Ansichten bestimmt und die Ansicht sowie ihre Nachbarn als Parameter an die Bergevin Fehlerverteilungsmethode übergeben. Durch Vergleich zwischen dem Rückgabewert der Bergevin Methode und dem Parameter *Toleranz* wird entschieden ob die Nachbaransichten in die Queue gespeichert werden und somit ein backtracking und zusätzliche Fehlerverteilung durchgeführt wird oder nicht. Sobald die Queue leer ist endet die While-Schleife und eine neue Ansicht mit Verbindung zur Menge der ausgewählten Ansichten und maximalem Knotengrad innerhalb der noch nicht gewählten Ansichten wird gesucht und hinzugefügt. Das Verfahren terminiert sobald die Anzahl der gewählten Ansichten gleich der Anzahl der Knoten im Modellgraph ist (Erfolgreiche Berechnung eines Gesamtmodells) oder keine neue Ansicht gefunden werden kann (bei dieser Einstellung des Parameters Kantengrenze kann kein zusammenhängendes Modell berechnet werden).

6.1.14 Bergevin Fehlerverteilung

Das in der Pulli Globalregistrierung eingesetzte Fehlerverteilungsverfahren ist der Bergevin ,,Multi-View-Registration" Algorithmus [2], der in Kapitel 3.3.2 beschrieben wird. Die Implementierung richtet sich dabei nach den Beschreibungen im Artikel.

Die Bergevin Fehlerverteilung arbeitet auf einem sternförmigen (Teil-)Modellgrafen, der durch eine zentrale Ansicht c und die umgebenden Ansichten x_1 bis x_n gegeben ist. Zwischen je zwei Ansichten x_i und x_j wird aus den Transformationen $c \rightarrow x_i$ und $c \rightarrow x_j$ eine Transformation $x_i \rightarrow x_j$ berechnet und auf die so entstandene Transformation ICP angewendet. Nachdem für alle i, j die Transformationen berechnet wurden, werden für die Transformation c, x_i alle Transformationen von c über j nach i berechnet und durch zellenweises Addieren der Transformationsmatrizen und anschließende zellenweise Division durch die Anzahl der Transformationen gemittelt. Mit der gleichen Vorgehensweise wird die Orginaltransformation und die neue Transformation im Verhältniss 1 : 1 gemittelt. Diese gemittelte Transformation wird an der Kante c, x_i gespeichert. Die Summe der zellenweisen Differenz der original und der gemittelten Transformationsmatrix bildet den Wert für die relative Änderung.

Relative Änderung i=$\sum_n \sum_m T_{neui} < n, m > -T_{original} < n, m >$

Auch dieser Schritt wird für alle Kanten c, x_i durchgeführt. Der Durchschnitt aller relativen Änderungen ist der Rückgabewert der Methode.

Rückgabewert = $\frac{\sum_i Relative\ Änderung i}{Anzahl\ Umgebungsknoten}$

6.1.15 Kantenfilter

Beim Laden der Scans wird auf den Dreiecksnetzen ein Kantenfilter ausgeführt. Dieser soll Anomalien filtern, die durch ein fehlerhaftes Scannen bzw. Abtasten der Objekte entstanden sind. Dafür stehen dem Kantenfilter zwei Methoden zur Verfügung. Bei der Ersten werden Kanten gefiltert, die die durchschnittliche Kantenlänge um einen bestimmten Faktor überschreiten. Dieser Faktor kann in den Optionen des Registrierungswerkzeuges eingestellt werden. Alle Dreiecke, die an diesen überlangen Kanten grenzen werden ebenfalls gefiltert. Die zweite Methode filtert Randkanten im Dreiecksnetz. Diese Filterung kann in mehreren Stufen passieren. Die Anzahl dieser Stufen kann ebenfalls in den Optionen des Registrierungswerkzeuges angegeben werden.

6.1.16 Export und Verschmelzung

Es wird eine Punktwolke erzeugt, die durch einen `vector` von Punkten repräsentiert wird. Angefangen beim Wurzelpatch wird für alle Patches wie folgt verfahren: Alle schon in der Ausgabeliste vorhandenen Punkte werden in die ANN-Datenstruktur eingefügt. Dann wird auf jeden einzelnen Punkt des Patches die ermittelte globale Transformation angewandt. Dann wird für diesen Punkt mit Hilfe der ANN-Bibliothek der Abstand zum nächsten Punkt in der Ausgabeliste ermittelt. Ist Abstand kleiner als ein Schwellwert, so werden die beiden Punkte verschmolzen, das heißt der vorhandene Punkt wird durch das arithmetische Mittel der beiden Punkte ersetzt. Nachdem alle Punkte abgearbeitet wurden, werden alle inzidenten

Scans, welche noch nicht behandelt wurden, in die Liste der noch zu bearbeitenden Scans aufgenommen. Zum Schluß werden alle Punkte der Ausgabeliste in eine OFF-Datei geschrieben.

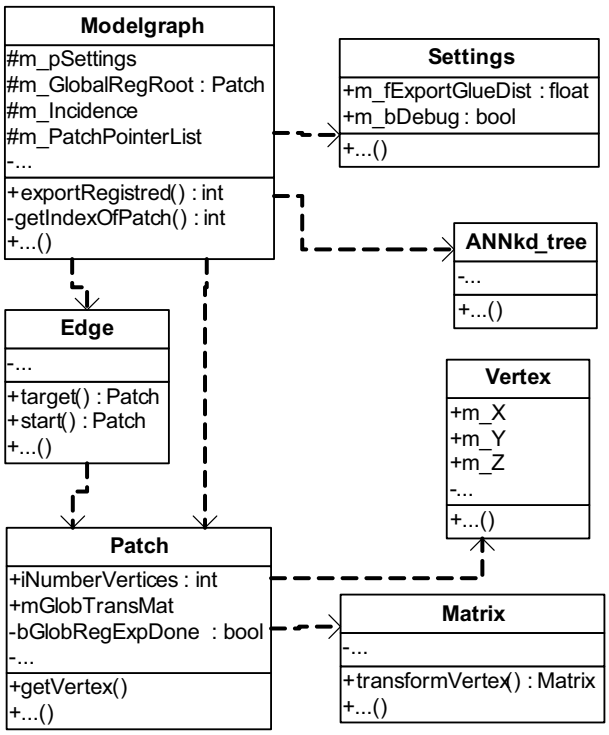

Abbildung 6.12: Auszug aus dem Registrierungswerkzeug-Klassendiagramm: Darstellung der für den Export relevanten Klassen.

6.1.17 Klassendiagramm Registrierungswerkzeug

Die Klassendiagramme zeigen die grundlegende Struktur des Registrierungswerkzeug (siehe Abbildung 6.14 und Abbildung 6.13). Dabei wurden aus Gründen der Übersichtlichkeit die eigentlichen Programmkomponenten von der Struktur der grafischen Benutzeroberfläche getrennt. Abbildung 6.14 verdeutlicht wie die einzelnen Grob-, Fein- und Globalregistrierungsverfahren in das Programm eingebunden sind. Für jedes Registrierungsverfahren existiert noch ein detailliertes Klassendiagramm welche die Abhängigkeiten der jeweiligen Verfahren aufzeigt.

6.2 Implementierung Reduktionswerkzeug

Aufgabe der Reduktion soll eine benutzerfreundliche und schnelle Reduktion der gegebenen 3D-Modelle sein. Um die Benutzerfreundlichkeit zu steigern wurde das Reduktions-Modul mit dem Registrierungs-Modul verschmolzen. Dadurch wird der Arbeitsablauf vereinfacht und die Produktivität erhöht. In diesem Abschnitt wird auf wichtige Details der Implementierung eingegangen.

6.2.1 Vorgehensweise

Die Vorgehensweise bei der Reduktion ist das Abarbeiten einer Prioritätswarteschlange. Zuerst werden für alle Kanten die Fehlerwerte berechnet. Dann werden die Kanten anhand dieser Fehler in die Warteschlange einsortiert. Jeweils eine Kante wird dann aus dieser Warteschlange entnommen und reduziert. Nachdem eine Kante reduziert wurde, werden die nötigen Änderungen in der gesamten Datenstruktur durchgeführt. Neu erzeugte Kanten werden anhand ihrer neu berechneten Fehler in die Warteschlange eingefügt. Alte

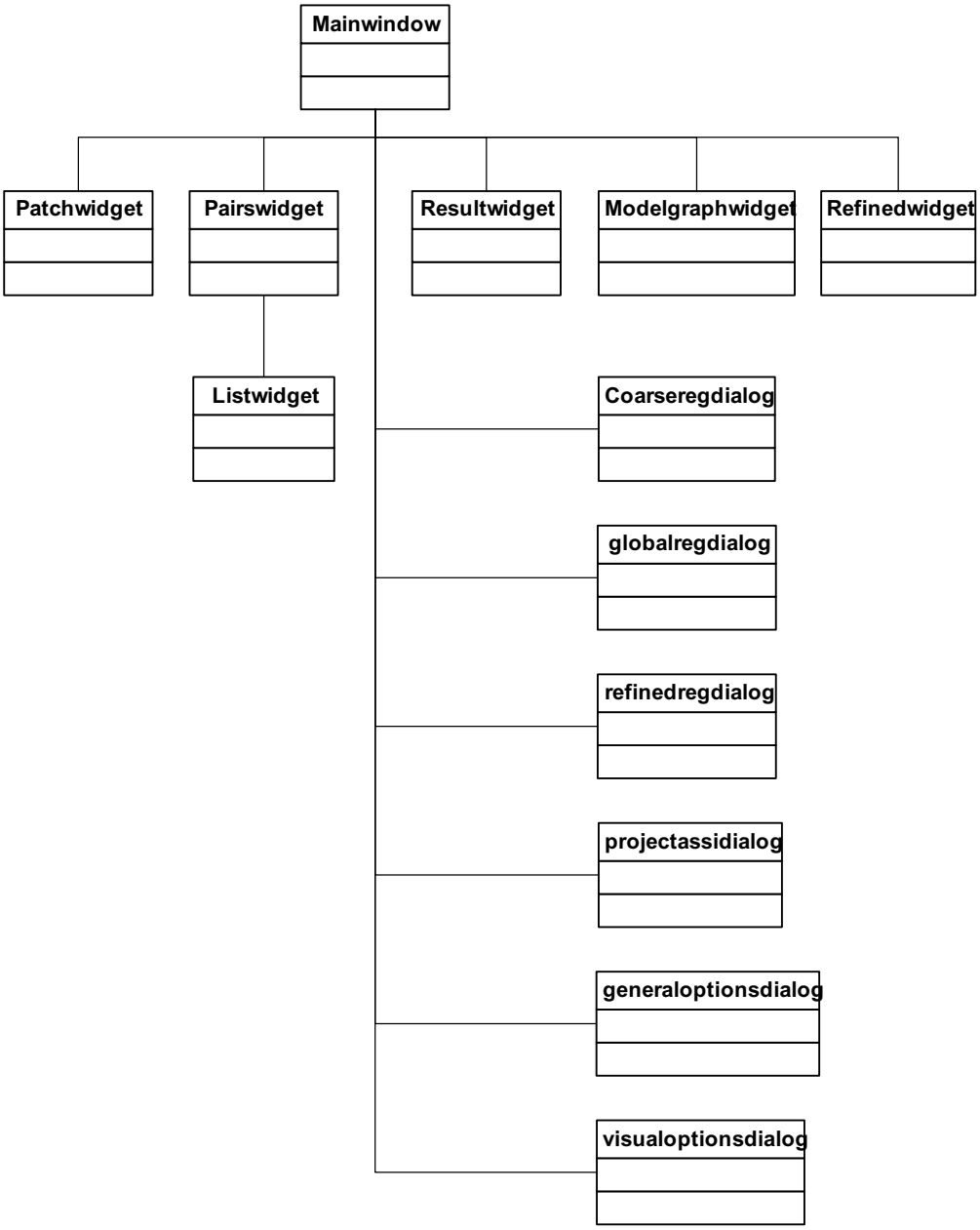

Abbildung 6.13: Klassendiagramm Registrierungswerkzeug - vereinfachtes Klassendiagramm der grafischen Benutzeroberfläche

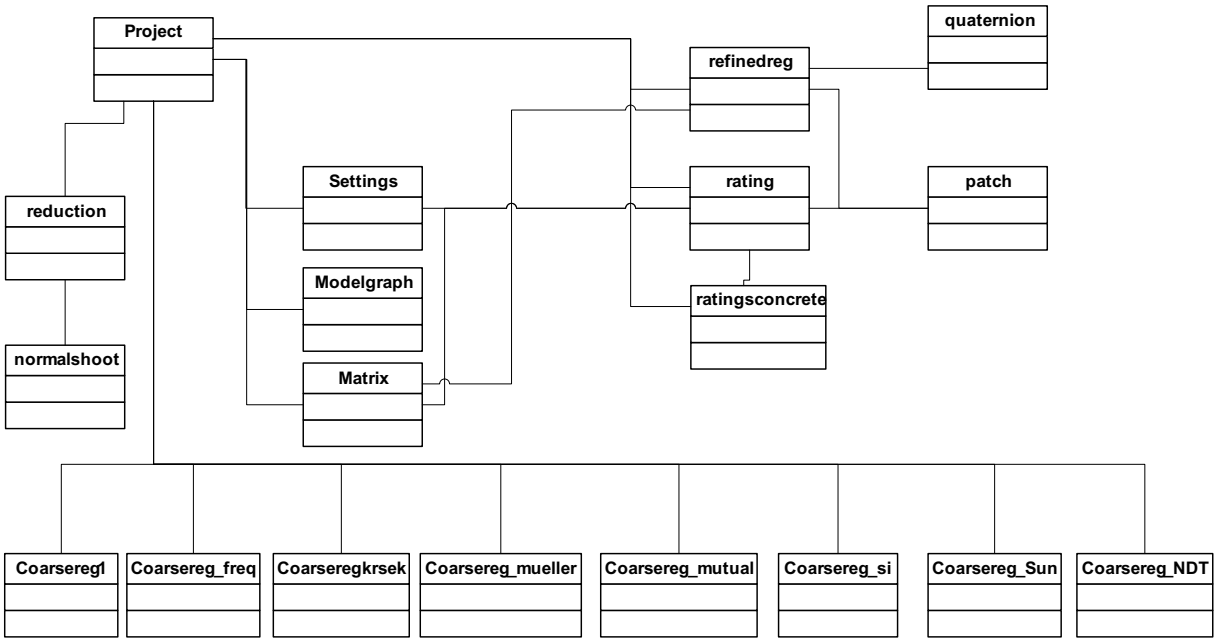

Abbildung 6.14: Klassendiagramm Registrierungswerkzeug - vereinfachtes Klassendiagramm der Komponenten des Programms

Kanten werden gelöscht, Dreiecke werden verändert und alte Dreiecke werden ebenfalls gelöscht. In dieser Warteschlange werden die Kanten anhand ihrer Fehler so sortiert, dass das Element mit dem geringsten Fehler am Anfang der Warteschlange steht. Diese Prioritätswarteschlange wird dann solange abgearbeitet bis ein Abbruchkriterium (Fehlerschranke) erreicht wird.

6.2.2 Heap

Nun hat diese Vorgehensweise bezüglich der Geschwindigkeit zwei Nachteile. Der erste Nachteil ist das Löschen und Einfügen von Kanten. Dies kann jedoch umgangen werden indem die zu löschenden Kanten nicht gelöscht sondern markiert werden. Wenn nun solch eine Kante an der Reihe ist, wird sie nicht weiter behandelt und direkt gelöscht. Somit ist ein aufwendiges Suchen und Sortieren nicht nötig. Neue bzw. geänderte Kanten, die eingefügt werden müssen, werden derart behandelt, dass sie direkt in der Warteschlange geändert werden. Dadurch wird ein komplettes Sortieren der Prioritätswarteschlange vermieden. Die neuen Kanten werden nur lokal in der Struktur sortiert.

Das Sortieren ist der zweite Nachteil. Hier kommt das *Heap-Sort*-Verfahren zum Einsatz. Dieses Verfahren sortiert die Warteschlange so, dass das erste Element die Kante mit dem geringsten Fehler ist. Die restlichen Elemente sind nur partiell sortiert. Im Kern von *Heap-Sort* steckt ein links-vollständiger Binärbaum (*Heap*). Die Kinderknoten eines Vaterknotens sind dabei höchstens so klein wie der Vaterknoten selbst. Nun wird aus der Warteschlange ein *Heap* gebildet. Dadurch steht die Kante mit dem kleinsten Fehler an erster Stelle. Wenn nun diese Kante entfernt wird, muss die *Heap*-Eigenschaft wiederhergestellt werden. Dies geschieht dadurch, dass das letzte, am weitesten rechts stehende Element in der untersten Ebene, die neue Wurzel wird. Dadurch wird allerdings die Ordnung verletzt. Diese wird aber dadurch wiederhergestellt, dass nun der Wurzelknoten mit seinen Kinderknoten verglichen und ggf. vertauscht wird. Dies setzt sich nach unten fort.

Der somit erzeugte *Heap* erlaubt es vorhandene Knoten auf- und absteigen zu lassen. Dadurch ist es möglich Elemente im *Heap* lokal zu sortieren. Wenn sich jetzt eine Kante und ihr Fehler während der Reduktion ändern, kann die Lage im *Heap* lokal korrigiert werden.

6.2.3 Datenverwaltung

Während der Reduktion werden normalerweise die Inversen der durchgeführten Operationen gespeichert. Durch diese Operationen und dem gegebenen Netz lassen sich alle Reduktionsschritte rückgängig machen. Dadurch kann man reduzierte Netze aus dem Originalnetz erhalten. Dieses Vorgehen impliziert jedoch eine Implementierung einer Darstellungsmöglichkeit der reduzierten Netze zur Prüfung der Güte der Reduktion. Um den Aufwand gering zu halten und dennoch eine Darstellung zu ermöglichen, werden temporäre OFF-Dateien der Reduktionsstufen gespeichert. Diese Dateien können im Registrierungsmodul geladen, gelöscht oder in einer für den Server verwendbaren Binär-Datei gespeichert werden. Die Anzahl der Reduktionsstufen wird dabei vom Benutzer bestimmt.

Der Nutzer muss am Anfang der Reduktion einen Grenzwert für den Fehler festlegen. Dieser Grenzwert unterscheidet sich von Netz zu Netz unabhängig von den Reduktionsstufen. Also muss dem Benutzer ein Maß gegeben werden, an dem er sich orientieren kann. Ein guter Anhaltspunkt für den Grenzwert ist der globale Mittelwert der Fehlermetrik. Deshalb werden schon beim Einlesen eines Dreiecksnetzes die Fehler der Kanten berechnet. Während der Reduktion wird nur eine Aktualisierung der Fehler durchgeführt.

6.2.4 Weitere Details

Die Reduktion wurde so implementiert, dass auch Dreiecksnetze mit Randkanten und Löchern reduziert werden können. Dazu werden beim Einlesen einer OFF-Datei die betreffenden Kanten bestimmt und markiert. Durch diese Markierung werden die Kanten nicht weiter behandelt bzw. reduziert. Weiterhin werden auch OFF-Dateien, die keine Kanten enthalten, verarbeitet. Die nötige Kantenmenge wird in dem Fall aus den Dreiecken berechnet. Dabei werden doppelte Kanten gefiltert. Wenn eine gegebene Datei Kanten enthält, werden diese nicht berücksichtigt.

6.2.5 Klassendiagramm Reduktionswerkzeug

Hauptbestandteil der Reduktion sind die Klassen `Reduction` und `Object3d`. Diese Klassen und weitere Strukturen werden nun im weiteren Verlauf näher erläutert. Eine grafische Darstellung der Klassen ist in Abbildung 6.15 zu sehen.

Wichtige Variablen und Strukturen

Die gesamte Reduktion baut auf die Datenstruktur `rData` auf. Diese Struktur besteht aus drei Vektoren. Der Vektor `rVertexVector` enthält die Knoten, die Dreiecke sind in `rTriangleVector` enthalten und die Kanten werden in dem Vektor `rEdgeVector` gespeichert. An den Knoten, Dreiecken und Kanten werden je Element weitere Werte gespeichert. So enthält ein Knoten, neben seinen Koordinaten im Raum, auch einen Vektor aus Indizes seiner adjazenten Dreiecke, einen Vektor aus Indizes seiner adjazenten Kanten, seine Krümmung und die Summe der Flächeninhalte der benachbarten Dreiecke. Die Vektoren der adjazenten Dreiecke bzw. Kanten dienen dem Zugriff auf diese Elemente. Dadurch können die Reduktion betreffende Kanten und Dreiecke schnell verändert werden. Ein Dreieck baut sich aus drei Indizes des Knotenvektors, einem Normalenvektor, dem Flächeninhalt und einer Markierung auf. Die drei Indizes repräsentieren dabei die Dreiecksknoten. Der Normalenvektor und der Flächeninhalt dienen der Krümmungsberechnung. Ob ein Dreieck in der Struktur noch aktuell ist zeigt die Markierung. Durch diese Vorgehensweise ist ein Löschen von alten Dreiecken im Dreiecksvektor nicht nötig. Eine Kante besteht aus einer Markierung, einem Paar von Indizes der Kantenendpunkte im Knotenvektor, einem Fehlerwert und einem zugehörigen Index im *Heap*. Die Markierung dient, wie bei den Dreiecken, ebenfalls der Unterscheidung ob die Kante aktuell ist oder nicht. Der *Heap* ist die zweite wichtige Datenstruktur in der Reduktionsklasse. Dieser *Heap* repräsentiert eine *Priority Queue*. Um nicht die Zugriffsmöglichkeit auf adjazente Kanten per Index zu verlieren und dennoch die Kanten anhand ihrer Fehler sortieren zu können, wird nicht die Kantenmenge an sich behandelt sondern ein Vektor bestehend aus Kantenindizes. Dieser Vektor *heap_vector* wird dann als *Heap* sortiert.

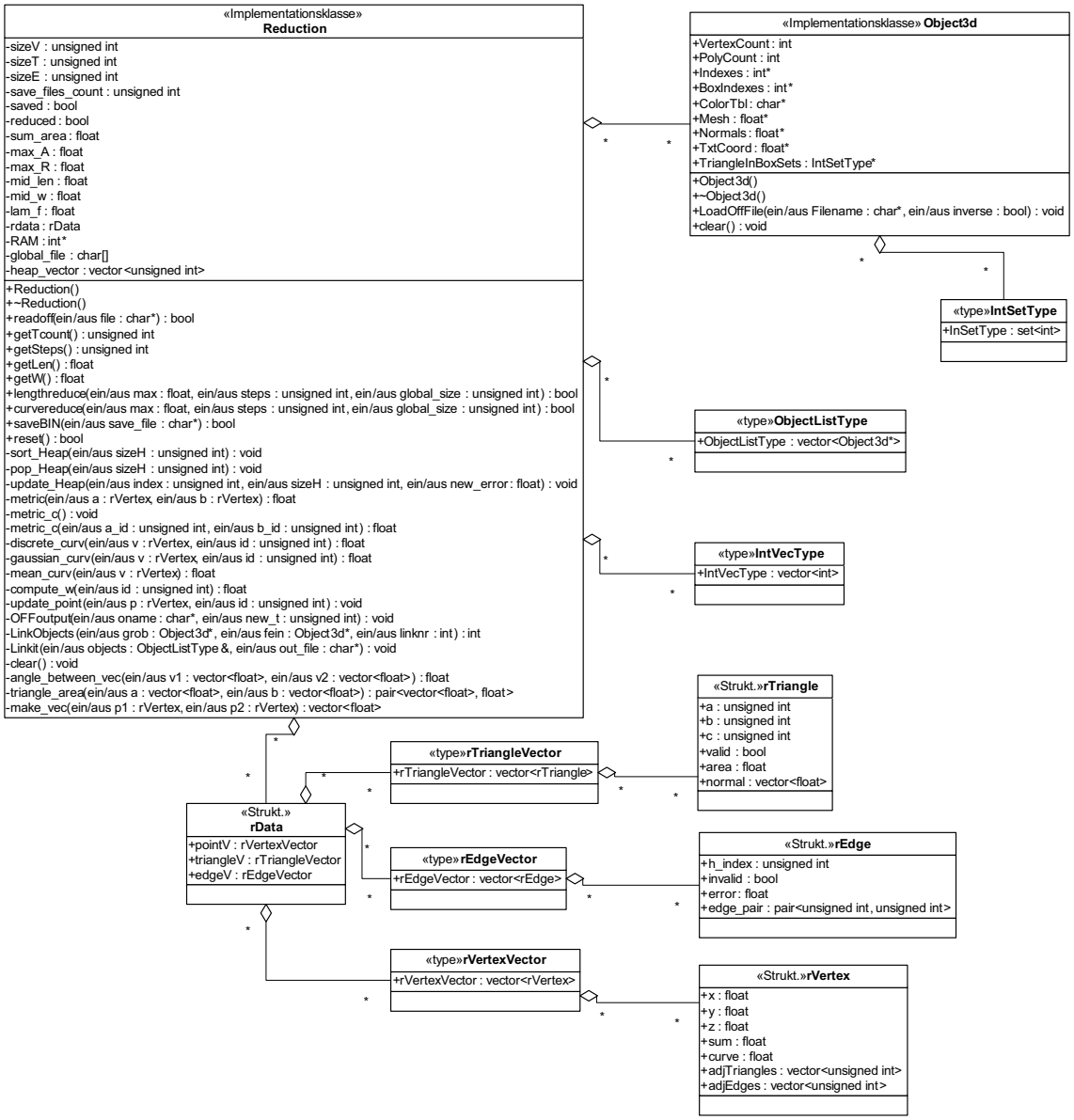

Abbildung 6.15: Klassendiagramm der Reduktion

`Reduction`-Klasse

In der Klasse ist eine Vielzahl von Methoden vorhanden. Es werden jedoch nur die Wichtigsten erläutert.

Die Funktion `readoff` liest eine OFF-Datei ein und baut die zuvor erwähnte Struktur *rData* auf. Als Parameter *file* wird der Dateiname übergeben. Während des Einlesens wird die Kantenmenge erstellt und doppelte Kanten gefiltert. Gleichzeitig wird für jede Kante die Kantenlänge durch `metric` berechnet. Diese Methode bekommt als Übergabeparameter zwei Knoten und gibt den Abstand dieser zueinander zurück. Ebenfalls wird für jeden Knoten die Krümmung berechnet. Dafür kommt die Methode `discrete_curv` zum Einsatz. Diese Methode ruft die Methoden `mean_curv` und `gaussian_curv` auf. Die erste Methode berechnet für einen übergebenen Knoten v dessen mittlere Krümmung und die zweite Methode berechnet für einen Knoten v und dessen Index *id* die Gauß-Krümmung. Dann werden diese Werte zurückgegeben und R (siehe 4.2) berechnet. Nachdem für jeden Knoten solch ein Wert berechnet wurde wird durch `compute_w()` der Wert W für alle Knoten berechnet (siehe 4.2). Der Parameter *id* ist dabei der Index des zu behandelnden Knoten. Wenn eine Datei erfolgreich eingelesen wurde wird *true* zurückgegeben, wenn nicht gibt die Methode `readoff()` den Wert *false* zurück.

Wenn die Datei eingelesen wurde kann die Reduktion durch den Aufruf von `lengthreduce` oder `curvereduce` gestartet werden. Übergabeparameter sind die Anzahl der zu reduzierenden Dreiecke *global_size*, Anzahl der Reduktionsstufen *steps* und der maximal erlaubte Fehler einer Kante *max*. Nach einer erfolgreichen Reduktion wird *true* zurückgegeben. Beide Funktionen arbeiten ähnlich und unterscheiden sich hauptsächlich in der Fehlerberechnung. Erst wird der Vektor *heap_vector* zu einem *Heap* sortiert. Bei der Reduktion anhand der Krümmung wird allerdings erst für jede Kante der Fehlerwert berechnet. Dann wird die Kantenmenge in zuvor festgelegten Stufen reduziert. Nach jeder Reduktionsstufe wird durch den Aufruf von `OFFoutput` eine temporäre OFF-Datei gespeichert. Der Parameter *oname* enthält den Namen der Datei und *new_t* gibt die Anzahl der Dreiecke an.

Innerhalb der Stufen wird jeweils eine Kante aus dem *Heap* entnommen und reduziert. Dazu wird der Fehler der Kante geprüft. Die Reduktion wird abgebrochen wenn der Grenzwert *max* überschritten wird. Falls der Grenzwert nicht überschritten wird, wird der Mittelpunkt der Kante berechnet. Dann werden die adjazenten Dreiecke des neuen Punktes berechnet und nicht mehr aktuelle Dreiecke markiert. Gleiches gilt für die adjazenten Kanten. Der neue Punkt wird dann an der Position eines alten Kantenendpunktes gespeichert. Dadurch ist es nicht nötig alle sich ändernden Dreiecke und Kanten zu behandeln. Durch die Indizierung ist ein Teil der neuen Dreiecke bzw. Kanten schon in der Struktur vorhanden. Die restlichen Dreiecke und Kanten erhält man, indem man die Indizes der Dreiecksknoten und Kantenendknoten auf den neuen Punkt setzt. Dies geschieht schon bei der Berechnung der adjazenten Dreiecke und Kanten. Nun müssen noch die Fehler der neuen Kanten bestimmt werden.

In der Methode `lengthreduce` wird dazu für jede neue Kante die Methode `metric` aufgerufen. Im Gegensatz dazu ist bei der Methode `curvereduce` mehr Aufwand nötig. Dort wird erst der W-Wert des neuen Punktes und seiner Nachbarn berechnet. `update_point` berechnet die dazu nötigen Werte der neuen Dreiecke und R des neuen Punktes p an der Stelle *id* im Knotenvektor. Die Variablen p und *id* werden der Methode als Parameter übergeben. Danach werden die R-Werte der Nachbarn berechnet. Durch `compute_w` wird dann W für den neuen Punkt und jeden seiner Nachbarn ermittelt. Dann wird für jede neue Kante der Fehler bestimmt. Zum Abschluss werden die Kanten im *Heap* sortiert.

Nach einer erfolgreichen Reduktion kann durch Aufruf von `saveBIN()` eine BIN-Datei der reduzierten Daten gespeichert werden. Der Name der BIN-Datei wird durch den Parameter *save_file* übergeben. In dieser Methode werden Instanzen der Klasse `Object3d` erzeugt. Diese Objekte enthalten die Daten der zuvor erstellten Reduktionsstufen und die Originaldaten. Die Methode ruft dann `Linkit()` auf. Erst diese Funktion verbindet die Objekte und speichert eine BIN-Datei. Der Parameter *out_file* enthält den Dateinamen der BIN-Datei und *objects* ist ein Vektor aus Instanzen der Klasse `Object3d`.

`Object3d`-Klasse

Die `Object3d`-Klasse dient dem Laden von (temporären) OFF-Dateien. Jede Datei wird durch eine `Object3d`-Instanz repräsentiert und der Klasse `Reduction` zur Verfügung gestellt. Durch die Methode `LoadOffFile(char* Filename, bool inverse)` wird eine Datei eingelesen und in temporäre Strukturen gespeichert. Die Parameter der Methode sind *Filename* und *inverse*. Die Variable *Filename* enthält den Dateinamen einer OFF-Datei und *inverse* gibt die Richtung der Normalen der eingelesenen Dreiecke an. Die Methode `clear` dient dem Löschen der durch die Methode zuvor angelegten Strukturen. Durch diese Klasse ist es der Reduktion möglich aus den erzeugten Reduktionsstufen eine BIN-Datei zu erzeugen.

6.3 Implementierung Client / Server

Ziel der Implementierung von Client und Server war es, ein performantes Werkzeug zur Betrachtung und Vermessung hochauflösender dreidimensionaler Netze, welche über ein Netzwerk übertragen werden sollen, zu entwickeln. Dabei wurde darauf geachtet, dass die benötigten Hardwareressourcen und Anforderungen an das Netzwerk in einem vertretbaren Rahmen bleiben. So ist es möglich einen einfachen gängigen PC als Client und Server einzusetzen. Das Netzwerk sollte mindestens 1 MBit Übertragungsrate bieten, ansonsten wird die Übertragung größerer Netze langsam und es kommt bei der Betrachtung im

Client zu Wartezeiten. Dieses Kapitel beschreibt wesentliche Implementierungsdetails des Clients und des Servers.

Multithreading

Der Client benutzt Multithreadingtechniken um im Hintergrund Daten zu empfangen und Anforderungen zu senden, während der Benutzer währenddessen uneingeschränkt die Benutzeroberfläche bedienen kann. Es gibt also einen Hauptthread, der die GUI und die Anzeige, sowie die Berechnung darzustellender Bildausschnitte übernimmt, und einen weiteren Thread, der alle Netzwerkaufgaben übernimmt. Bei der Berechnung von Bildausschnitten im Hauptthread wird festgestellt, welche Teile des Netzes noch nicht auf dem Client vorhanden sind. Diese werden zunächst in einer Liste gesammelt. Da der Netzwerkthread unter Umständen zum Zeitpunkt der Fertigstellung der Anforderungsliste beschäftigt ist, musste eine Lösung entwickelt werden, wie sich der Hauptthread sicher mit dem Netzwerkthread synchronisiert, ohne auf diesen warten zu müssen. Wartet man hingegen auf den Netzwerkthread, so friert die GUI ein und es ist während dieses Wartevorganges keine Bedienung des Clients mehr möglich. Für die Threadsynchronisation wurde eine Mutexgeschützte Queue eingesetzt. Der Mutex schützt die Queue vor gleichzeitigem Zugriff von Haupt- und Netzwerkthread. Ein ungeschützter Zugriff wäre nicht sicher und gerade bei dynamischen Datenstrukturen wie Queues würde dies irgendwann zum Absturz oder Einfrieren des Clients führen. Zu dem Zeitpunkt, zu dem der Client seine Anforderungslisten fertiggestellt hat, sichert er sich den Zugriff auf die Queue und schreibt einen Event in diese Queue. Anschließend gibt er die Queue wieder frei. Die Zeit, die dieser Vorgang dauert, ist so kurz, dass der Benutzer dies nicht bemerken kann. Auf der anderen Seite, im Netzwerkthread, wird in der `run` -Methode die Queue abgearbeitet. Dazu wird selbstverständlich auch jeder Zugriff auf die Queue durch einen Mutex geschützt. Das Entnehmen eines Events aus der Queue geht genausoschnell wie das Hereinschreiben eines Events. Daher wird der Mutex nur sehr kurz benötigt, was letztlich zu einer threadsicheren schnellen synchronisierten Kommunikation zwischen Haupt- und Netzwerkthread führt.

Berechnung der Bildausschnitte

Bei der Berechnung der Darstellung von Bildausschnitten gibt es grundsätzlich zwei Möglichkeiten. Entweder man berechnet bei jedem Bild den Bildausschnitt neu, oder man berechnet den Bildausschnitt nur zu günstigen Zeitpunkten neu. Letzteres hat den Nachteil, dass während der Benutzer sich in der Ansicht bewegt, unpassende Bildausschnitte mit geringerer Auflösung dargestellt werden, als dies theoretisch möglich wäre. Ein günstiger Zeitpunkt wäre z. B. dann gegeben, wenn der Benutzer die Maustasten loslässt oder die Maus für eine kurze Zeitdauer von z. B. 200ms nicht bewegt. Der Benutzer müsste also auf eine hochauflösende Ansicht während der Navigation verzichten. Die erste Methode hat diesen Nachteil nicht, jedoch setzt sie höhere Anforderungen an die Hardware des Client-PCs. Im Graphaeology-Client wurde eine Kombination von beiden Methoden implementiert. Einerseits ist es möglich in Echtzeit hochauflösende Bilder während der Navigation zu betrachten, andererseits gibt es eine Option, dass während der Navigation nur ein niedrigaufgelöstes Netz dargestellt wird. Diese Option heisst „Fast Display“, und kann im Bedienfeld „Option“aktiviert werden. Die Bildberechnung geschieht trotz allem für jedes darzustellende Bild. Dazu hat der Client eine Datenstruktur, aus der er sich schnell die gewünschten sichtbaren Bereiche extrahieren kann. Die Datenstruktur für den LOD Ansatz ist ein dynamischer zweidimensionaler Vektor mit der maximalen Größe [Anzahl Basisdreiecke] · [Anzahl Stufen]. In jedem Element dieses Vektors befindet sich ein Vektor, der die Indizes der zu der entsprechenden Stufe gehörigen Dreiecke enthält.

Während der Bildberechnung werden zunächst die Basisdreiecke durchlaufen. Anhand der Kameraposition des Betrachters wird ermittelt ob dieses Dreieck mindestens zum Teil sichtbar ist. Falls nicht, so wird mit dem nächsten Basisdreieck fortgefahren. Falls das Dreieck zum Teil sichtbar ist, wird die ungefähre Größe der Fläche berechnet, die das Dreieck besitzen würde, wenn es durch die Projektionsmatrizen läuft. Aus diesem Wert wird berechnet, welche Stufe für dieses Dreieck benutzt werden muss. Optimalerweise sollte das Dreieck dem Flächeninhalt eines einzelnen Pixels auf dem Bildschirm entsprechen. Benötigt das Dreieck mehr als einen Pixel, so ist die maximal darstellbare Auflösung noch nicht erreicht, während es, falls es weniger als einen Pixel Flächeninhalt hätte, zu klein wäre um sinnvoll Information darzustellen. Nach Berechnung der optimalen Stufe dieses Basisdreiecks wird ermittelt, ob die dazugehörige Stufe

bereits im Client vorhanden ist. Falls nicht, wird diese Stufe in ein `set` eingetragen. Ist die Stufe vorhanden so wird sie direkt mit Hilfe von Vertexarrays gezeichnet. Hierbei ist natürlich zu erwähnen, dass die Elemente der Datenstruktur bereits OpenGL konforme Vertexarrays enthält. Nach dem Durchlaufen der Basisdreiecke ist das Zeichnen bereits mit der maximalen verfügbaren Auflösung abgeschlossen. Die Stufen, die im `set` aufgenommen wurden, werden nun mit Hilfe einer Schnittbildung mit den bereits auf der Anforderungsliste stehenden Dreiecke verglichen. Daraus resultieren alle Dreiecke, die noch nicht vom Server angefordert wurden. Ist diese Menge sehr klein, so bedeutet dies, dass zur Darstellung einer hochauflösenden Ansicht beinahe genug Dreiecke vorhanden sind. Allerdings heisst dies nicht, dass das Modell bereits komplett auf dem Client vorhande ist. Daher werden kleine Mengen von Anforderungsstufen bis zu einer Mindestmenge mit noch nicht vorhandenen Stufen aufgestockt. Diese werden dann in einem Event zusammengefasst und in die Mutexgeschützte Queue des Netzwerkthreads geschrieben. Zuletzt wird die Menge der neu angeforderten Dreiecke mit der Menge der bereits angeforderten, aber noch nicht erhaltenen Dreiecke, vereinigt.

Verarbeitung eintreffender Daten

Der Netzwerkthread wartet nach Anforderung eines Pakets auf die entsprechende Antwort des Servers. Diese trifft komprimiert ein. Nach dem Entpacken wird für den Hauptthread ein Aktualisierungspaket zusammengestellt. Nachdem alle für dieses Paket notwendigen Daten zusammengestellt wurden, wird das Aktualisierungspaket über ein QT-Signal einer queued-Connection an den Hauptthread übertragen. Dieser ordnet die empfangenen, nun unkomprimierten Stufen in seine Datenstrukturen ein und erzwingt ein Neuberechnen der Ansicht. Das Neuberechnen der Ansicht bewirkt unter Umständen eine erneute Anforderung von Dreiecken, so dass die Ansicht nach und nach so hochauflösend wie möglich dargestellt wird.

Vermessungsberechnungen auf dem Client

Der Client ermöglicht mit dem Vermessungstool Punkte des Netzes zu markieren. Die euklidischen Abstände zwischen diesen Punkten werden aufsummiert und dargestellt. Hierzu ist grundsätzlich eine Transformation der zweidimensionalen Mauskoordinaten in einen dreidimensionalen Raumpunkt notwendig. Gängige Verfahren, die mit Hilfe von Schnittpunkten mit Dreiecken, die Raumpunkte berechnen sind viel zu langsam um auf hochauflösenden Netzen zu funktionieren. Daher wird mit Hilfe von OpenGL Funktionen die Tiefe des Pixels an der Mausposition aus dem `z-Buffer` ermittelt. Mit Hilfe von Rückprojektionen über die Projektionsmatrix und der Modelviewmatrix kann so der dreidimensionale Raumpunkt ermittelt werden. Die markierte Punkte werden in einer einfachen Liste gespeichert.

Schnittebenenberechnungen auf dem Client

Im Client ist es möglich eine Schnittebene zu benutzen um Querschnitte des betrachteten Objektes anzeigen zu lassen. Die Schnittebenen werden bei jeder Bewegung der Ebene in Echtzeit berechnet und zwischengespeichert. Die Darstellung und Navigation in einer berechneten Schnittebene geschieht so in gewohnter Geschwindigkeit. Zur Berechnung der Schnittebene wird jedes Dreieck der feinsten vorhandenen Stufe benutzt. Ein Dreieck schneidet die Schnittebene, falls mindestens ein Paar der drei Eckpunkte auf verschiedenen Seiten der Schnittebene liegen. Dies lässt sich mit Hilfe der Distanz aus den Normalen der Schnittebene berechnen. Das Vorzeichen der Distanz gibt an, auf welcher Seite ein Punkt liegt. Schneidet ein Dreieck die Schnittebene, so werden die zwei Schnittpunkte der Kanten des Dreiecks mit der Schnitteben berechnet. Beide Schnittpunkte bilden eine Strecke, die alle Schnittpunkte des Dreiecks mit der Ebene enthält. Diese Linie wird zwischengespeichert. Letztlich werden zur Darstellung der berechneten Schnittebene nurnoch alle Schnittlinien und die Ebene selbst gezeichnet.

Serverseitiges Einlesen der Daten

Hat sich ein Client für ein Modell entschieden, so läd der Server sich die entsprechende Datendatei in den Speicher. Dies ist notwendig, da das Auslesen der Daten von der Festplatte bei einer Anforderung

des Clients leider, trotz hochaktueller leistungsfähiger Hardware, eine maximale Übertragungsrate von wenigen MBit ermöglichte. Ein Bereitstellen der Daten aus dem Speicher inklusive Datenkompression ermöglichte dagegen Netto-Übertragunsraten mit mehr als 40 MBit. Damit sollte man auch für hochaktuelle DSL-Leitungen gerüstet sein. Nachteilig ist natürlich, dass nur eine begrenzte Zahl an Clients gleichzeitig an einem Server angemeldet sein können. Dieser Nachteil wird allerdings ohnehin aufgehoben durch die Anbindung des Servers an das Internet, welche wohl in den meisten Fällen der Flaschenhals der Übertragung sein dürfte.

6.3.1 Klassendiagramm Client / Server

Server

Der Server besteht im Wesentlichen aus drei Klassen: `Server`, `ClientSocket` und `Modelloader`.

`Server`-Klasse

Die Klasse `Server` basiert auf der Qt-Klasse `QTcpServer`. Diese Klasse liefert einen TCP-basierten Server, der es ermöglicht auf eingehende TCP-Verbindungen zu warten. Bei einer eingehenden Verbindung wird die Methode `showUserlist()` aufgerufen um den Benutzer der eingehenden Verbindung zu authentifizieren. Für eine eingehende Verbindung wird ein neues Objekt vom Typ `ClientSocket` erzeugt und in die Liste `m_ClientSocketList` der offenen TCP-Verbindungen eingetragen.

Die Methode `showModellist()` wird von der Instanz der `ClientSocket` aufgerufen wenn der Bunutzer die aktuell verfügbaren Modelle erfragt.

`ClientSocket`-Klasse

Die Klasse `ClientSocket` basiert auf der Qt-Klasse `QTcpSocket`. Diese Klasse liefert eine TCP-basierte Socket, die eine Stream-basierte TCP-Verbindung bereit stellt. Eine Instanz dieser Klasse stellt eine einzelne Verbindung zu einem Client dar. Die Attribute `m_model` und `m_user` enhalten den Namen des Modells und des Benutzers. `m_pOwner` ist ein Zeiger auf auf den Server über den die ClientSocket zum Beispiel auf die Methode zur Abfrage der verfügbaren Modelle zurückgreifen kann. `m_pFullData` ist ein Zeiger auf die zu übermittelnden Daten, die über eine Instanz des `m_pLoader` des `Modelloaders` gefüllt werden. Diese Daten werden dann über die Socket an den Client übermittelt. Die übrigen Methoden werden innerhalb der `ClientSocket`-Klasse aufgrufen, wenn eine entsprechende Anfrage vom Client erfolgt. Das bedeutet im Detail: `OnReadyRead()` wird immer dann aufgerufen, wenn neue Daten an der Socket zur Abholung warten. Diese Methode ist die Kernmethode dieser Klasse. Hier entscheidet sich anhand der empfangenen Daten, welche der hier im folgenden angegebenen Methoden aufgerufen wird.

- `LogIn(ProtLogin &a)`, wenn sich der Benutzer authentifiziert.
- `ChosenModel(ProtChosenModell &a)`, wenn der Benutzer ein bestimmtes Modell ausgewählt hat. Das führt dazu, dass eine Instanz des Modelloaders mit dem entsprechenden Modell erzeugt wird.
- `OnDisconnect()` zum Trennen der Verbindung.
- `SendModelList()`, wenn die Liste der verfügbaren Modelle angefragt wird.
- `SendBaseTriangles()`, um die Basisdreiecke an den Client zu schicken.
- `SendPatchData()`, wenn die Anfrage einer Verfeinerung zu einem bestimmten Basisdreieck in einer bestimmten Verfeinerungsstufe erfolgt.
- `SendModelInfo()`, wenn die Anfrage nach zusätzlichen Modellinformationen erfolgt.

`Modelloader`-Klasse

Die Klasse `Modelloader` stellt einen Container für die auf der Platte liegenden Daten zur Verfügung. Von die `ClientSocket` erfolgt eine Anfrage zu der Verfeinerung einer bestimmten Dreiecksliste. Der Modelloader stellt dann die benötigten Daten aus der Datei zusammen. Die wichtigste Methode stellt hier die `getPatchTriangles(int layer, int base, int &outByteSize)` dar. Diese Methode stellt an

Hand des Index' und der Verfeinerungsstufe eine Liste von Dreiecken und deren Eckpunkten zusammen. Diese Daten werden in der folgenden Reihenfolge in einem char-Array `m_pData` aneinandergehängt:

- Die Nummer der Verfeinerungsebene.
- Der Index des Basisdreiecks.
- Die Anzahl n der Verfeinerungsdreiecke.
- Insgesamt n 3-Tupel mit den Punktindizes der Dreiecke.
- Die Anzahl m der Punkte.¡/li¿
- m 4-Tupel aus dem Index des Punktes und den 3D-Koordinaten des Punktes.

Die Methode `getBaseTriangles(int &size)` liefert in analoger Weise die Daten für die Basisdreiecke. Diese Methode `check` gibt an, ob die Datendatei zum Lesen geöffnet werden konnte. Schließlich gibt es noch die Methoden `getPatchCount()` und `getNodeCount`, die die Anzahl der Dreiecke und Knoten zurück liefern.

Client

Der Client besteht aus einer Vielzahl Klassen, die jedoch in den meisten Fällen nur eine untergeordnete Rolle spielen. Daher zeigt das Klassendiagramm in Abbildung 6.17 nur die wesenlichen Klassen.

Auch innerhalb der Klassen wurde die Darstellung erheblich reduziert. Aus Platzgründen wurde auf die Angabe der Methoden-Signaturen verzichtet und es wurde ausschließlich wichtige Attribute angeben.

`ClientMainWindow`-Klasse

Die Klasse `ClientMainWindow` ist die Zentrale Klasse des Clients. Hier wird die grafische Oberfläche erzeugt und alle Komponenten der Anwendung zusammen geführt. Wichtigste Bestandteile dieser Klasse sind die beiden eingebetteten GLWidgets `m_pMainGL` und `m_pMiniGL`. Für die Kommunikation mit dem Server besitzt diese Klasse auch einen Instanz des `TCPThreads` `m_pTCPThread`. Und als viertes wichtigstes Element gibt es noch den `MeshDisposer` `m_pMeshDisposer`. Der `MeshDisposer` verwaltet die Daten des aktuell angezeigten Modells. Auf Anfrage wir ein Satz Dreiecke auswählt und darstellt. Der `timer` dient noch dazu festzustellen, ab wann der Client *idle* ist. Diese Information kann dazu genutzt werden Anfragen zu noch fehlenden Verfeinerungen des Netzes an den Server zu senden, so dass das Netz im Hintergrund im Lauf der Zeit vervollständigt wird. Die Methoden dienen in aller Regeln nur als Konnektoren zwischen Objekten und den Menüs.

`ClientMainGLWidget`-Klasse

Die Klasse `ClientMainGLWidget` stellt die Hauptdarstellungskomponente zur Verfügung. Diese Klasse enthält die Methoden zur Steuerung des Objektes, sowie Beleuchtung, Bemaßung und die Wahl der darzustellenden Objekte an Hand einer Sichtpyramide.

Für die Steuerung der Objekte kann der Benutzter zwischen dem `Trackball` `m_Trackball`, einer Steuerung, die auf Basis einer virtuellen Kugel die Objekte im Raum bewegt und der `Camera3d` `m_XSICam`, die der Steuerung in Softimages XSI nachempfunden ist wählen. Die bereits erwähnte Sichtpyramide `m_Frustum` auf Basis der Klasse `FrustumCulling` dient dazu, genau die Dreiecke des Basisnetzes zu bestimmen, die verfeinert dargestellt werden müssen. Alle anderen Bereiche des Objektes, die außerhalb der Sichtpyramide liegen, werden nur in ihrer gröbsten Darstellungsstufe dargestellt. Die Berechnung der Detailstufen übernimmt dann die Instanz des `MeshDisposer` `m_pMeshDisposer`.

Bei der Zeichnung kommt OpenGL-typisch die Methode `paintGL()` zum Einsatz. Je nach aktivierten Optionen werden aus dieser Methode heraus die Methoden

- `DrawMesh()` zur Darstellung des Objektes,
- `CalcSectionPlane()` zur Darstellung einer Schnittebene,

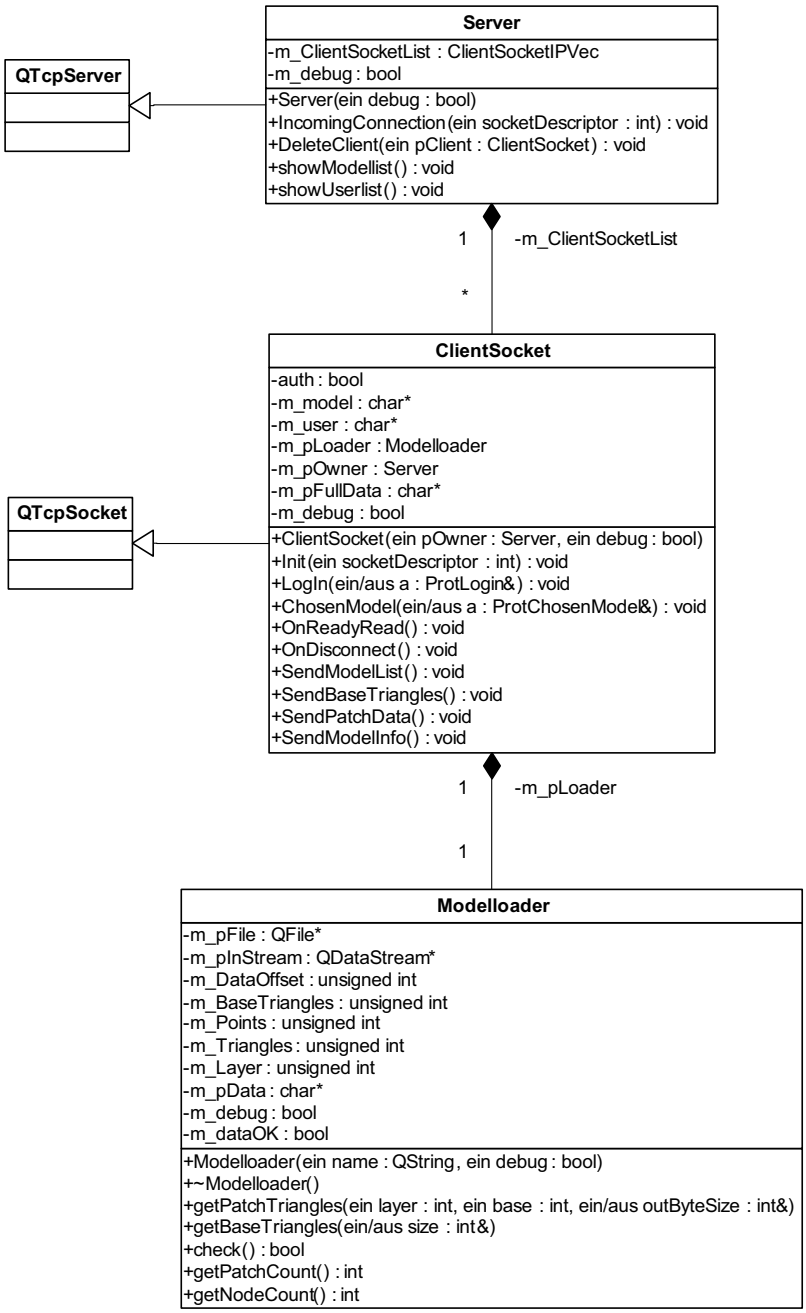

Abbildung 6.16: Klassendiagramm des Servers.

- `CalcSection()` zu Darstellung des Schnitts von Schnittebene und Objekt und
- `InsertMeasuringMarker(int x,int y)` zu Visualisierung eines Bemaßungspunktes

aufgerufen. Im letzten Fall wird zudem bei neu hinzugefügten Bemaßungspunkten – aktiviertes Bemaßungstool vorausgesetzt – der Bemaßungsdialog über das Signal `MeasuringLengthUpdate(QString str)` aktualisiert.

Sehr wichtig ist auch die Aktualisiertung der Gesamtansicht im `ClientMiniGLWidget`, die je nach gewählter Steuerung über das Signal `emitView(float* tMatrix, float distance, float w, float h)` oder `emitXSIView(float x1,float y1,float z1,float x2,float y2,float z2)` erfolgt.

`Trackball`-Klasse

Die Klasse `Trackball` stellt die Funktionalität eines virtuellen Trackballs zur Verfügung. Dieser wird über die Angabe des Radius mit `setRadius(float radius)` initialisiert und durch den Aufruf der Methode `drawTrackball()` innerhalb der `paintGL()`-Methode des GLWidgets schematisch dargestellt werden. Ausgangslage für die Rotation mit dem Trackball ist einen Eineheitsmatrix. Diese kann auch jeder Zeit wieder mit der Methode `initializeMatrix()` zurückgesetzt werden. Eine zusätzliche Matrix existiert zudem, um die Rototranslation der Schnittebene zu speichern. Diese Matrix kann analog über die Methode `initializeClippingPlaneMatrix()` mit der Einheitsmatrix initialisiert werden. Über die folgenden Methoden können nun die Hauptrototranslationsmatrix und Schnittebenenrototranslationsmatrix veändert werden:

- `addRotation(int x1, int y1, int x2, int y2)`: Angabe einer Rotatation über das Koordinatenpaar zweier Punkte in der Ebene, die Anfangs- und Endpunkt eine Mausbewegung darstellen. Mit der Methode `getSphereCoords(int x, int y)`, die selbst auch die Methode `clickpointIsInside(int x, int y)` verwendet werden die sphärichen Koordinaten der Klickpunkte bestimmt. Aus diesen lässt sich nun einfach eine Rotattionsachse und der zugehörige Winkel bestimmen. Schießlich werden diese Informationen in eine Rotattionsmatrix umgewandelt und über die Methode `appendMatrix` der aktuellen Rototranslationsmatrix hinzugefügt.

- `addRotation(float x, float y, float z, float a)`: Bei dieser Methode erfolgt die Angabe der Rotation direkt über die Rotationsachse und den Winkel. Dadurch muss nur noch eine entsprechende Rotattionsmatrix erzeugt und über die Methode `appendMatrix` der aktuellen Rototranslationsmatrix hinzugefügt werden.

- `addMove(int x1, int y1, int x2, int y2, float d, float h)`: Hier wird die Bewegung wieder über einen Anfangs- und einen Endpunkt einer Mausbewegung in der Ebene bestimmt. Die Parameter d und h geben die Distanz und die sichtbare Höhe einer zur Blickrichtung orthogonalen Ebene wieder. Damit wird gewährleistet, dass das Objekt unabhängig von der Distanz zum Betrachter der Maus in gleichbleibender Geschwindigkeit folgt. Auch hier wird wieder eine Matrix erstellt, die mit `appendMatrix` der aktuellen Rototranslationsmatrix hinzugefügt wird.

- `addMove(float x, float y, float z)`: In diesem Fall wird die Bewegung direkt in absoluten Werten angeben und auch hier wieder über eine Matrix er aktuellen Rototranslationsmatrix hinzugefügt.

- `addZoom(float z)`: Ein Zoom ist im Grunde genommen nur eine Bewegung entlang der Blickrichtung. Demnach genügt hier ein angepasster Aufruf von `addMove`.

- `addClippingPlaneRotation(int x1, int y1, int x2, int y2)`: Diese Methode funktioniert analog zu Methode `addRotation(int x1, int y1, int x2, int y2)`, mit dem Unterschied, dass nicht die Hauptmatrix, sondern die Matrix für die Schnittebene verändert wird.

- `addClippingPlaneMove(int x1, int y1, int x2, int y2, float d, float h)` Diese Methode funktioniert analog zu Methode `addMove(int x1, int y1, int x2, int y2, float d, float h)`, mit dem Unterschied, dass nicht die Hauptmatrix, sondern die Matrix für die Schnittebene verändert wird.

- `addClippingPlaneZoom(float z)` Diese Methode funktioniert analog zu Methode `addZoom(float z)`, mit dem Unterschied, dass nicht die Hauptmatrix, sondern die Matrix für die Schnittebene verändert wird.

Zu Verwendung dieser Matrix stehen jetzt mehrere Wege zur Verfügung. Zum Einen kann man sich die Matrix mit `getMatrix()` und die Schnittebenen-Matrix mit `getClippingPlaneMatrix()` übergeben lassen. Diese Matrizen kann man dann an OpenGL übergeben um eine entsprechende Rototranslation durchzuführen.

Die zweite Möglichkeit ist es direkt mit Hilfe der Methode `rotateByMatrix(Vertex &rotV)` die Rotation eines einzelnen Vektors anhand der Hauptmatrix, der Methode `rotateByInverseMatrix(Vertex &rotV)` die Rotation eines einzelnen Vektors anhand der Inversen der Hauptmatrix, der Methode `rotateByClippingPlaneMatrix(Vertex &rotV)` die Rotation eines einzelnen Vektors anhand der

Schnittebenenmatrix oder der Methode `rotate(int x1, int y1, int x2, int y2, Vertex &camPos)` direkt mit Hilfe der Anfangs- und Endposition einer Mausbewegung die Rotation dder Kamera zu brechnen.

`Camera3d`-Klasse

Die Klasse `Camera3d` ist das Gegenstück zur `Trackball`-Klassen und stellt die Navigation nach Softimage XSI zur Verfügung. Die Kameraposition kann über die Methoden `SetCam_xyz(float vx,float vy,float vz)` und `SetCam_dxdydz(float vx,float vy,float vz)` direkt gesetzt werden. Die Bewegungen werden nicht wie beim Trackball auf die Scene, sondern auf die Kameraposition bezogen. Somit kann die Rotation mit der Methode `RotateCam(float ax,float ay,float az)`, die Verschiebung entlang der x- und y-Achse der Projektionsebene über die Methoden `MoveLeft(float ax)` und `MoveUp(float ax)` und das Zoomen mit der Methode `AddDistance(float dx)` erfolgen.

Das Objekt kennt also jederzeit die Position der Kamera. Von der Darstellenden Klasse (z. B. `ClientMainGLWidget`) müssen nun lediglich die Methoden `GetDistance()`, `GetCamPosition(float &x,float &y,float &z)` und `GetEyePosition(float &x,float &y,float &z)` aufgerufen werden und die Werte den entsprechenden OpenGL-Methoden übergeben werden.

`ClientMiniGLWidget`-Klasse

Die Klasse `ClientMiniGLWidget` stellt ein kleines OpenGL-Widget dar, welches zur Orientierung am Objekt dient. In der Darstellung wird die im `ClientMainGLWidget` aktive Sichtpyramide schmatisch dargestellt. Es ist immer die gesamte Ansicht des Objektes zu sehen, so dass deutlich erkennbar ist, wo sich der Betrachter gerade befindet.

`FrustumCulling`-Klasse

Die Klasse `FrustumCulling` stellt die Funktionalität einer Sichtpyramide zur Verfügung. Diese Sichtpyramide wird analog zur der Sichpyramide bei OpenGL initialisiert. Die Methode `setCamInternals(float angle, float ratio, float nearD, float farD)` setzt den Innenwinkel, das Verhältnis von Höhe und Breite, sowie die nahe und entfernte Clipping-Ebene. Dann wird die Kameraposition über die Methode `setCamDef(Vertex &p, Vertex &l, Vertex &u)` gesetzt. Dabei ist p die Position der Kamera, l der Blickpunkt und u der Vektor, der angibt wo sich Oben befindet.

Wenn nun in der Methode `paintGL()` des `ClientMainGLWidget` alle Rototraslationen erfolgt sind kann man sich von OpenGL die benutzten Transformationsmatrix holen. Mit dieser Transformationsmatrix wird mittels `setMatrix(float* matrix)` auch der Pyramidenstumpf gedreht. Die Methoden `rotateByInverseMatrix(Vertex &v)` und rotateByMatrix(Vertex &v) dienen hierbei als Helfer, die einen Vektor anhand der (inversen) Matrix rotieren.

Nun kann man über die Methoden `pointInFrustum(Vertex &p)` und `triangleInFrustum(Vertex &p1, Vertex &p2, Vertex &p3, bool backface)` erfragen, ob sich ein Punkt inner- oder ausserhalb, beziehungsweise beim Dreieck auch im Schnittbereich befindet. Zusätzlich kann bei der Untersuchung berücksichtigt werden, ob die das Dreieck rückwärtig betrachtet als ausserhalb gekennzeichnet werden soll.

`TCPThread`-Klasse

Die `TCPThread`-Klasse basiert wie schon die `ClientSocket`-Klasse des Servers auf der Qt-Klasse `QTcpSocket`. Die mit q beginnnenden Methoden erzeugen die für eine Server-Anfrage notwendigen Daten und schicken diese über die Socket an den Server. Diese sind:

- `qLoginToGraphaeologyServer(char * username,char *password)`: Login mit Übergabe von Benutzername und Kennwort.
- `qGetObjectList()`: Die Anfrage nach den verfügbaren Modellen.
- `qSetSelectedModel(char* model)`: Die Auswahl eines bestimmten Modells.
- `qGetBaseMesh()`: Anfrage nach dem Basisnetz des ausgewählten Modells.
- `qGetPatches(QueueElementType &q)`: Anfrage nach einer Verfeinerung zu einem Bestimmten Basisdreieck in einer Bestimmten Reduktionsstufe. Die Daten sind hierbei in einem speziellen `QueueElementType` gekapselt.

- `qGetModelInfo()`: Fragt nach zusätzlichen Informationen zu einem Modell. Das ist zum Beispiel die HTML-Datei mit den zugehörigen Grafiken.

All diese Anfragen werden anhand der Elemente ausgelöst, die in der Request-Queue liegen. In die Queue hinein gelangen die Anfragen durch die entsprechenden öffentlichen Methoden, die ähnliche oder identische Namen ohne ein vorangestelltes *q* besitzen.

Wie der Name der Klasse schon suggeriert handelt es sich bei dieser Klasse um einen Thread. Die Hauptschleife `run()` wird kontinuiertlich aufgerufen und verarbeitet die Queueelemente der Anfrage-Queue. Immer wenn neue Pakete beim Client eintreffen werden diese Empfangen, verarbeitet und über ein entsprechendes Signal an die anfragenden klassen Verteilt. Auch hier sind die Namen der Methoden wie gehabt, nur dass ihnen ein `Sig` vorangestellt wurde.

`MeshDisposer`-Klasse

Der `MeshDisposer` ist die Klasse, welche die empfangenen Daten kapselt und ggf. persistent auf die festplatte schreibt. Er besitzt eine Referenz auf den `TCPThread` um gezielt fehlende Daten anzufragen.

Die wichtigste Methode für die darstellenden Klassen `ClientMainGLWidget` und `ClientMiniGLWidget` ist `CreateMesh(bool lowmesh)`. Diese Methode erzeugt und zeichnet das darzustellende Modell.

Die anderen Methoden dienen im Wesentlichen der Verwaltung der Daten. So dienen die Methoden `DumpIt()` und `ReadIt()` dazu die Daten in einen lokalen Modellcache zu schreiben und dort wieder auszulesen, sollte das Modell zu einem späteren Zeitpunkt wieder geöffnet werden. Mit der Methode `AddPatchPackage(UpdateMeshPackageVectorType & patch, NodeWithIndexVectorType &nodes)` werden die beim Server angefragten Daten in die Datenstruktur eingefügt. Sollten keine ansichtsrelevanten Daten mehr benötigt werden, können beliebige noch nicht sichtbare Verfeinerungen angefragt werden. Dies geschieht mit `AppendIdlePatches(DemandSetType &newDemand)`. Sollte der Benutzer die Daten neu vom Server laden wollen, so bietet sich dazu die Möglichkeit die Datenstruktur wieder zurückzusetzen. Dies geschieht mit der Methode `ClearCache()`.

Bliebe zum Schluss noch die Methode `OffExport(char * szFilename)` zu erwähnen, mit dem die Daten im Cache in eine OFF-Datei exportiert werden. Näheres zu diesem Dateiformat siehe Kapitel B.2.

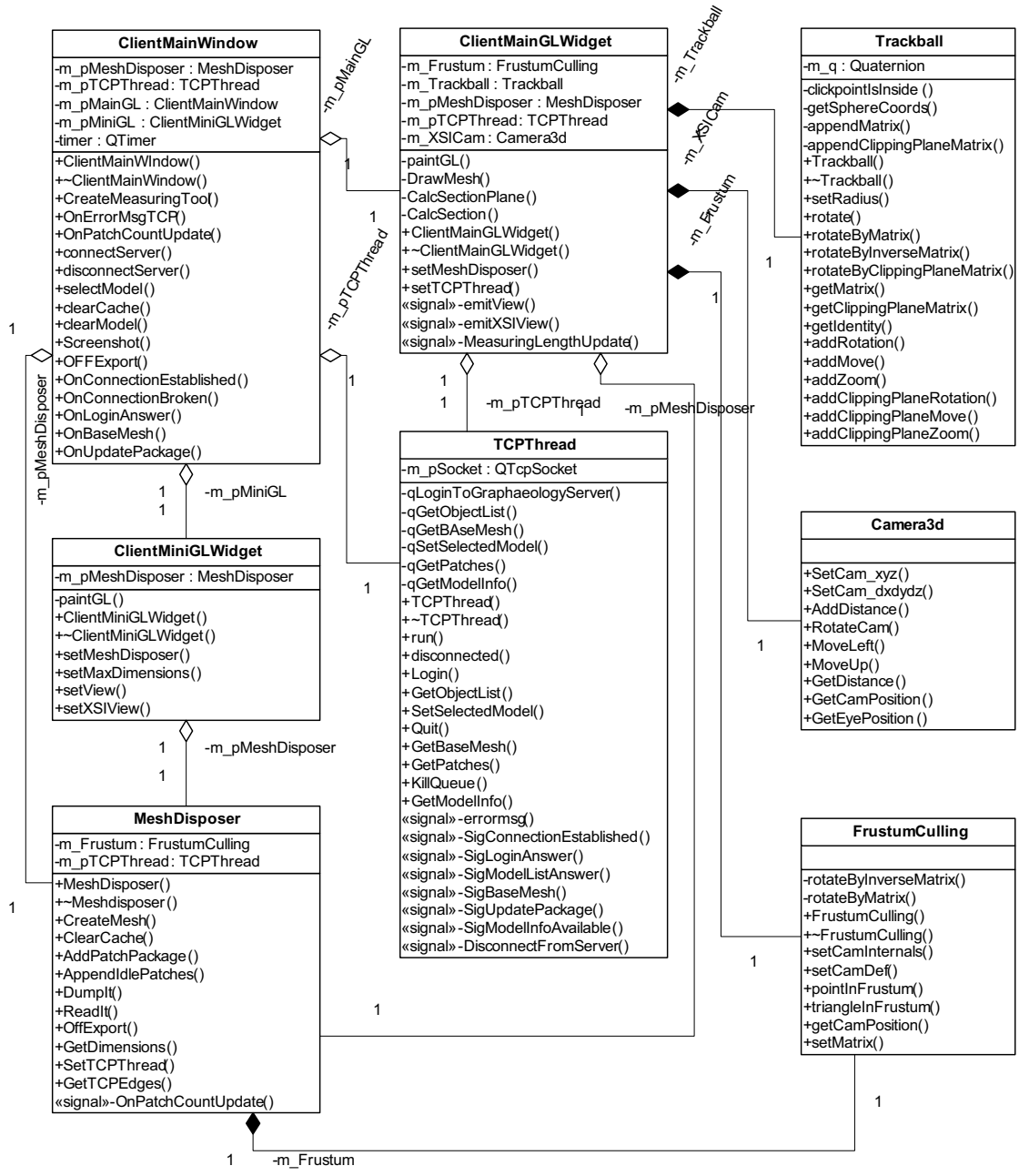

Abbildung 6.17: Stark vereinfachtes Klassendiagramm des Clients.

Kapitel 7

Evaluierung des Systems

In den folgenden Kapiteln werden die einzelnen Module und insbesondere die Registrierungsverfahren kritisch betrachtet. Dabei wird vorgestellt, welche Registrierungsverfahren effiziente Ergebnisse lieferten und welche sich nicht sinnvoll einsetzen ließen. Hierbei ist zu beachten, dass alle Verfahren nur in ihrer Basisvariante implementiert sind. Einige Verfahren ließen sich evtl. erweitern und optimieren.

7.1 Registrierung

7.1.1 Grobregistrierung

Spin Tables

Das Spin-Tables-Verfahren (siehe Kapitel 3.1.2) ist ein Registrierungsalgorithmus, welches zufällig drei Punkte aus jedem Scan auswählt. Daraus wird die Transformation berechnet und die Güte dieser Transformation bestimmt. Demzufolge hängt der Erfolg stark von der Bewertungsfunktion ab. Durch den Nichtdeterminismus kann das Verfahren daher sowohl sehr gute, als auch sehr schlechte Ergebnisse liefern.

Die Laufzeit hängt maßgeblich von der Anzahl der Durchläufe ab, die das Verfahren durchführt. Hier gilt, dass eine höhere Anzahl an Durchläufen mit größerer Wahrscheinlichkeit gute Ergebnisse liefert.

Das Registrierungsverfahren mittels Spin-Tables kann gute Ergebnisse liefern, ist jedoch für eine vollautomatische Grobregistrierung nicht geeignet. Probleme bereiten z. B. Objekte, die symmetrisch aufgebaut sind, d. h. wo unter Umständen mehrere Transformationen denkbar wären.

Spin-Images

Das Spin-Image-Verfahren (siehe Kapitel 3.1.3) hat auf allen zur Verfügung gestellten Datensätzen funktioniert, wenn die richtigen Parameter eingestellt wurden. Für die Parameter sollten folgende Bedingungen erfüllt sein:

- Der Kantenfilter (siehe 6.1.15) sollte ausgeschaltet sein.
- Die Bingröße sollte auf die Abtastauflösung eingestellt werden. Das ist zum Beispiel beim Armadillo 0.3, sowie bei der Lagerverwaltung und dem Siegelabdruck 0.25.
- Bei der Anzahl der horizontalen und vertikalen Bins in einem Spin-Image sollte 40 nicht unterschritten werden. Viel größere Werte führen letztendlich aber nur noch zu erheblich längeren Laufzeiten.

- Von einem Scan wird ein prozentualer Anteil Punkte betrachtet, der größer als zehn Prozent der Gesamtanzahl der Punkte sein sollte. Es ist von Vorteil, eine Primzahl[1] auszuwählen. Je kleiner der Wert gewählt wird, desto größer ist die Wahrscheinlichkeit einen Treffer zu landen, allerdings verlängert sich damit auch die Laufzeit.

Die Laufzeit für einen Durchlauf über die neun Scans des Siegelabdrucks beträgt bei sinnvoller Parameterwahl auf einem Intel Core Duo mit 2 GHz ungefähr zwei Stunden.

Krsek

Das Krsek-Verfahren (siehe Kapitel 3.1.4) erzeugt Kurven, die durch die Nulldurchgänge der Knotenkrümmungen laufen. Für alle Kurvenpaare werden dann die beiden Punkte gespeichert, die den geringsten Abstand haben. Es werden schließlich die Punktpaare von den zu registrierenden Scans miteinander verglichen und versucht korrespondierende Paare zu berechnen.
Wendet man es auf Objekte an, die nicht extrem glatt sind, dann berechnet es zu viele Kurven. Zusätzlich muss ein Kurvenverlauf in einem Patch nicht unbedingt dem Verlauf im anderen Patch entsprechen.
Außerdem wurde es scheinbar nur für kleinere Objekte entwickelt, da es auf den zur Verfügung stehenden Datensätzen extrem lange läuft.

Das Verfahren nach Krsek braucht sehr lange und liefert trotzdem schlechte Ergebnisse.

Frequency-based registration

Das frequenzbasierte Verfahren von Lucchese et al. sah sehr vielversprechend aus. Geht es doch, ausgehend von den Überlappungsbereichen, einen geraden Weg ohne Komplikationen und ohne große Fallunterscheidungen strikt der perfekten Registrierung entgegen. Die mathematischen Beziehungen aus dem Artikel geben einen klaren Weg vor, weit entfernt von einer Randomisierung, weit entfernt von einem `Brute-Force`-Verfahren. Dazu verspricht das Artikel noch eine Rechenzeit, die es ermöglichen soll in kurzer Zeit viele Scans perfekt zusammenzusetzen. Soweit zur Theorie. In der Praxis versagte das Verfahren bei allen getesteten Scans. Die Probleme sind bei den nicht exakt berechenbaren Überlappungsbereichen, der Diskretisierung der Fouriertransformationen und den diskreten Projektionen zu suchen. Das Verfahren nutzt in jedem Schritt die Rechenergebnisse des vorhergenden Schrittes. Dabei verschlechtert sich die Genauigkeit von Schritt zu Schritt. Beispielsweise führt ein nicht gut genug gewählter Überlappungsbereich zu störenden Elementen in den Voxelarrays (vgl. Kapitel 6.1). Dies führt zu Verfälschungen in den Frequenzbereichen. Eine nicht exakt berechnete Rotationsachse führt dann mit Sicherheit zu einem falsch berechneten Rotationswinkel. Die laut Artikel am Ende zu berechnende Translation beruht auf projizierten Koordinatensystemen anhand der Rotationsachse und des berechneten Rotationswinkels. Die Genauigkeit der Translation ließ auch unter definierten Testbedingungen stark zu wünschen übrig und war alles andere als zuverlässig. Ein Kontakt zu Lucchese, dem Autor des Verfahrens, zeigte, dass Lucchese mit denselben Problemen zu kämpfen hatte, und diese bis heute nicht gelöst werden konnten.

Damit ist das Verfahren nicht für eine automatische Registrierung einzusetzen, die Herangehens- und Betrachtungsweise allerdings so weit bekannt einzigartig und evtl. ausbaufähig.

Mutual Information

Das Verfahren (siehe Kapitel 3.1.6) arbeitet nur, wenn der Kantenfilter durch Setzen eines hohen Werts außer Kraft gesetzt wird, da ansonsten nicht parametrisiert werden kann (siehe Erläuterungen in Kapitel 2.1).
Das Problem des Verfahrens besteht darin, dass unendlich viele Transformationen möglich sind und es keinen Ansatz gibt, die bestmögliche zu finden. Daher werden zufällige Transformationen ausgewählt und getestet. Die Laufzeit hängt demzufolge von der Anzahl der zu testenden Transformationen ab. Theoretisch sollte also eine größere Anzahl von Transformationen ein besseres Ergebnis bringen. Dies ließ

[1] Der Chinesischer Restwertsatz liefert die Begründung dafür, denn die Wahrscheinlichkeit ist minimal, dass der Wert ein Teiler der Anzahl der Abtastpunkte des Scans in einer Dimension ist.

sich aber in der Praxis nicht beobachten.
Grundsätzlich funktioniert das Verfahren, denn beim Ausrichten zweier gleicher Patches gegeneinander war es möglich eine gute Transformation zu finden, wenn man genug Transformationen getestet hat. Für andere Scans war es allerdings nicht möglich gute Transformationen zu finden. Evtl. wäre zur Verbesserung der Ergebnisse eine andere als die hier gewählte Parametrisierung besser geeignet.

Müller

Das Verfahren nach Müller (siehe Kapitel 3.1.7) funktioniert nur auf geglätteten Datensätzen, da ansonsten zu viele ähnliche Krümmungen das Resultat verfälschen. Da das Verfahren verschiedene Transformationen mit Hilfe der Bewertungsfunktion durchtestet, kann es für gleiche Scan-Kombinationen in verschiedenen Durchläufen zu unterschiedlichen Ergebnissen kommen. Die Güte des Ergebnisses ist wie bei anderen Verfahren auch von der Bewertungsfunktion abhängig.

Der geglättete Spongebob-Datensatz konnte zu 3/4 automatisch zusammengesetzt werden, lediglich bei einer Scan-Kombination war manuelles Grobregistrieren notwendig.

Das Verfahren arbeitet auf kleineren Scans relativ zügig. Für einen einzelnen Registrierungsvorgang liegt die Rechenzeit im Bereich weniger Sekunden. Wenn man aber die Parameter für die Toleranzen zu großzügig wählt, dann kann das Verfahren schon mal mehrere Minuten brauchen.

CDN

Das CDN-Verfahren (siehe Kapitel 3.1.8) arbeitet nur auf Rasterscans. Da es die Distanzen zu den Nachbarpunkten als Grundlage für Bestimmung von Ähnlichkeiten benutzt und anschließend mögliche Transformationen der Reihe nach durchtestet, erhält man mal erstaunlich gute und mal weniger gute Transformationen. Schwachpunkt ist auch hier wieder, wie bei den Spin-Tables, die Bewertungsfunktion.

Das Verfahren hat den Vorteil, dass es zügig arbeitet, zur vollautomatischen Registrierung ist es allerdings nicht geeignet. Um einen kompletten Datensatz zusammenzusetzen sind manuelle Korrekturen nötig.

Roth

Das Verfahren nach Roth (siehe Kapitel 3.1.9) ist ein deterministisches Verfahren, das aber auf die nichtdeterministische Bewertungsfunktion zugreift. Daher können die Ergebnisse von Fall zu Fall etwas unterschiedlich sein.

Das Verfahren arbeitet vergleichsweise schnell, ist aber wie das Spin-Tables-Verfahren stark von der Bewertungsfunktion abhängig.

Für eine vollautomatische Grobregistrierung ist das Verfahren nicht geeignet, da erstens gute Parametereinstellungen für eine Scankombination nicht automatisch gut für eine zweite Kombination von Scans desselben Datensatzes sein müssen und zweitens die Anzahl an schlechten Transformationen, die das Verfahren liefert, einfach zu groß ist.

NDT

NDT (siehe Kapitel 3.1.10) wurde dafür entwickelt, dass sich mobile Roboter in einem Raum zurechtfinden. Es erhält einen 2D-Scan und versucht diesen in einem vorhandenen Grundriss des Raums wiederzufinden. Das Verfahren arbeitet über ein Newtonverfahren, es beginnt an einem Punkt und läuft von da an zum nächsten lokalen Maximum. Dies funktioniert aber hier nicht, da die Lage der 3D-Objekte unbekannt ist. Es bleibt festzustellen, dass das Verfahren in der Regel keine zufriedenstellenden Ergebnisse liefert. Zudem kommt es vor, dass das Ergebnis nach wenigen Iterationen besser sein kann als das Ergebnis nach vielen Iterationen, da das Verfahren anfällig dafür ist, lokale Optima zu berechnen und nicht globale.

7.1.2 Feinregistrierung

Die Feinregistrierung (Kapitel 3.2) arbeitet im Rahmen der natürlichen Grenzen für ein ICP-Verfahren. Das bedeutet, dass zwei Scans, die durch ein Grobregistrierungsverfahren ausreichend gut aneinander ausgerichtet worden sind, in angemessener Zeit durch das Feinregistrierungsverfahren korrekt feinregistriert werden.

7.1.3 Globalregistrierung

Die Ergebnisse der Globalregistrierung sind in erster Linie von den Ergebnissen der vorangegangenen Schritte abhängig. Liegen genügend gute Teilergebnisse vor, ist es mit beiden Globalregistrierungsverfahren möglich, ein zusammenhängendes Gesamtmodell zu erzeugen.

Pulli

Die Ergebnisse des Pulli-Verfahrens (Kapitel 3.3.2) sind neben den vorangegangenen Ergebnissen noch entscheidend vom Parameter „Kantengrenze“ abhängig. Wird dieser Parameter zu klein gewählt, ist das Ergebnismodell nicht zusammenhängend. Ist der Parameterwert zu groß, werden unter Umständen falsche Kanten gewählt, was zu schlechten Ergebnissen führt. Bei den Tests hat sich ein Parameterwert von 1.1 als effektiv erwiesen. Das im Pulli-Verfahren eingesetzte Fehlerverteilungsverfahren erzeugt bei gegebenen guten Ausgangsregistrierungen sehr gute Scanübergänge. Da es auch ICP verwendet, ist die Laufzeit in der von der Pg implementierten Fassung entsprechen hoch.

Huber und Hebert

Der Spannbaumansatz des Huber und Hebert Verfahrens (Kapitel 3.3.1) stellt sicher, dass immer ein zusammenhängendes Modell mit minimalem Gesamtfehler berechnet wird. Wahlweise kann noch eine zusätzliche ICP-Optimierung der verwendeten Registrierungen erfolgen. Diese hat jedoch nur geringen Einfluss auf das Endergebnis. Das nachschaltbare Fehlerverteilungsverfahren nach Sharp, Lee und Wehe hatte bei den Tests eher negative Folgen, da durch die zusätzlichen Kanten zur Kreisbildung sehr große Fehler in das Modell übertragen wurden. Das Fehlerverteilungsverfahren ist somit nur in Ausnahmefällen sinnvoll (Kapitel 3.3.3). Das Verfahren ohne ICP-Optimierung ist sehr schnell, da es nur auf den berechneten Werten der Transformationen arbeitet und nicht auf den Daten der Scans.

Testergebnisse

Beide Globalregistrierungsverfahren lieferten mit den Standardeinstellungen gute Ergebnisse bei den Testobjekten *Siegelabdruck* (siehe Abbildung 7.1 und 7.2) nach Grobregistrierung mit dem Spin-Image-Verfahren und anschließender Feinregistrierung und *Polymesh/Amadillo* (siehe Abbildung 7.3 und 7.4) nach Grobregistrierung mittels *Feature-Based-Matching* und anschließender Feinregistrierung.

7.2 Reduktion

Am Anfang der Implementierung musste entschieden werden, welche Grundoperation der Reduktion genutzt werden soll. Ebenfalls wurden Fehlermetriken geprüft und ausgewählt. Die damit verbundenen Probleme und Ergebnisse sollen in diesem Abschnitt beschrieben werden.

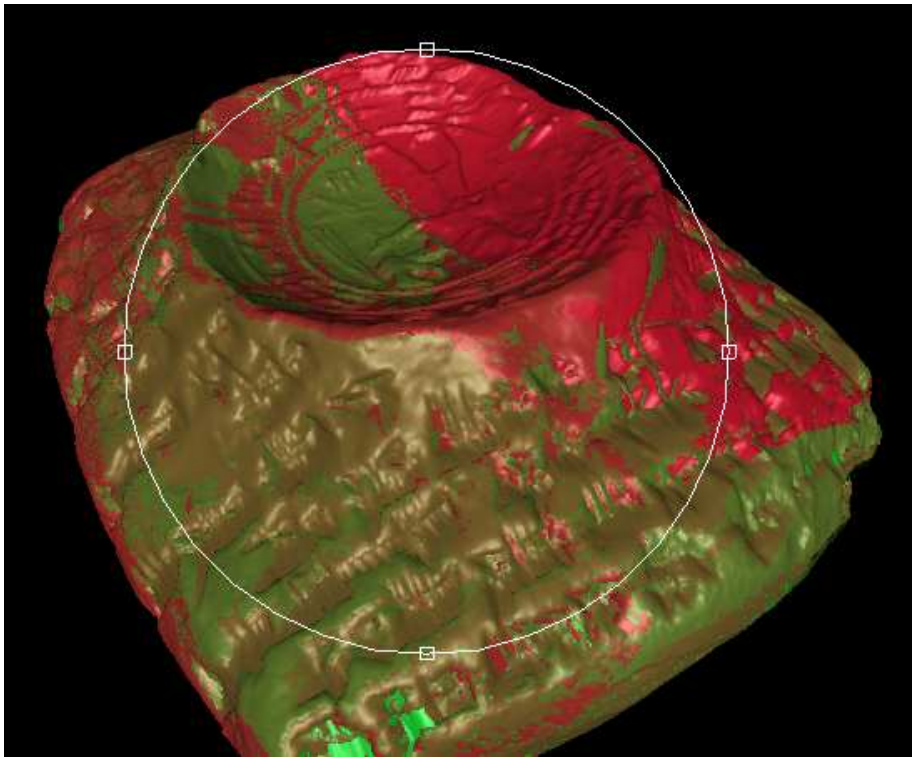

Abbildung 7.1: Ergebnis der Globalregistrierung mit dem Verfahren nach Huber und Hebert

7.2.1 Wahl der Grundoperation

Am Anfang musste zwischen den Grundoperationen, dargestellt in Abbildung 7.5 und 7.6, eine geeignete gewählt werden.

Dabei muss beachtet werden, dass die Wahl der Grundoperation entscheidend bezüglich der Wahl der Fehlermetrik ist. Nicht jede Metrik eignet sich für jede Grundoperation gleich gut oder lässt sich für jede Operation geeignet implementieren. Die Art der Berechnung der Fehler hängt stark von der Grundoperation ab. Viele Fehlermetriken können aber auch auf die jeweilige Operation angepasst werden.

Da bei der Kantenkontraktion nur wenige Sonderfälle während der Reduktion beachtet werden müssen, wurde diese gewählt. Ebenfalls ist bei dieser Grundoperation, im Gegensatz zum Knotenentfernen, kein weiterer Arbeitsschritt, wie z. B. die Triangulierung von Löchern, nötig. Die Wahl der Kantenkontraktion schien anfangs sinnvoll, erwies sich während der Implementierung aber als nicht so gut. Sinnvolle Metriken bezüglich der Kantenkontraktion sind rechenaufwendiger als für das Knotenentfernen, da bei der Kantenkontraktion zwei Knoten und gegebenenfalls ihre Nachbarschaft betrachtet werden müssen. In der Literatur wird weitestgehend das Knotenentfernen oder die Halbkantenkontraktion durchgeführt.

7.2.2 Wahl der Fehlermetrik

Nach der Wahl der Grundoperation musste eine geeignete Fehlermetrik gefunden werden. Dabei bot sich die Kantenlänge als einfaches Fehlermaß an. Der Fehler einer Kante berechnet sich dabei durch den euklidischen Abstand der Kantenendpunkte. Diese Fehlermetrik ist durch ihre einfache Berechnung schnell und liefert akzeptable Resultate. Dennoch ist diese Fehlermetrik nicht optimal, da bei sehr starker Reduktion viele Details eines gegebenen Dreiecksnetzes verloren gehen. Als zweite Fehlermetrik traf die Wahl auf den GJK-Algorithmus [15]. Dieser Algorithmus berechnet die minimalen Distanzen zwischen Objekten. Es wurde versucht den Algorithmus auf die Kantenkontraktion anzupassen. Dies erwies sich als nicht erfolgversprechend, da bei der hier durchgeführten Kantenkontraktion immer minimale Distanzen von null entstehen. Es ergeben sich Kontaktpunkte zwischen den alten Kanten und den neu entstehenden Kanten.

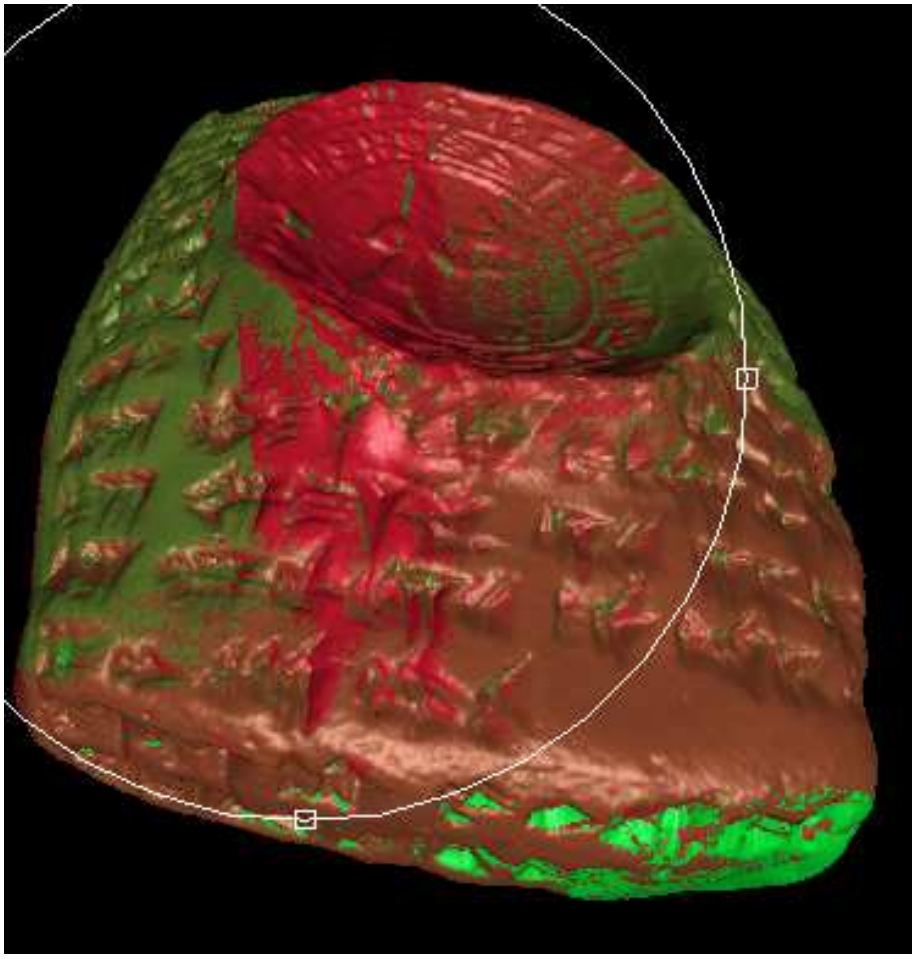

Abbildung 7.2: Ergebnis der Globalregistrierung mit dem Verfahren nach Pulli

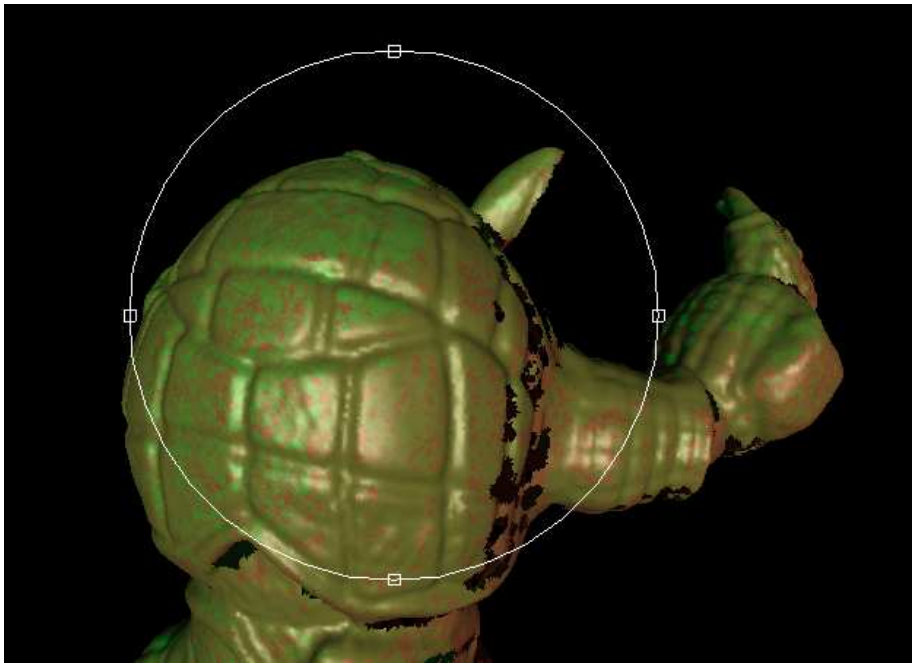

Abbildung 7.3: Ergebnis der Globalregistrierung mit dem Verfahren nach Huber und Hebert

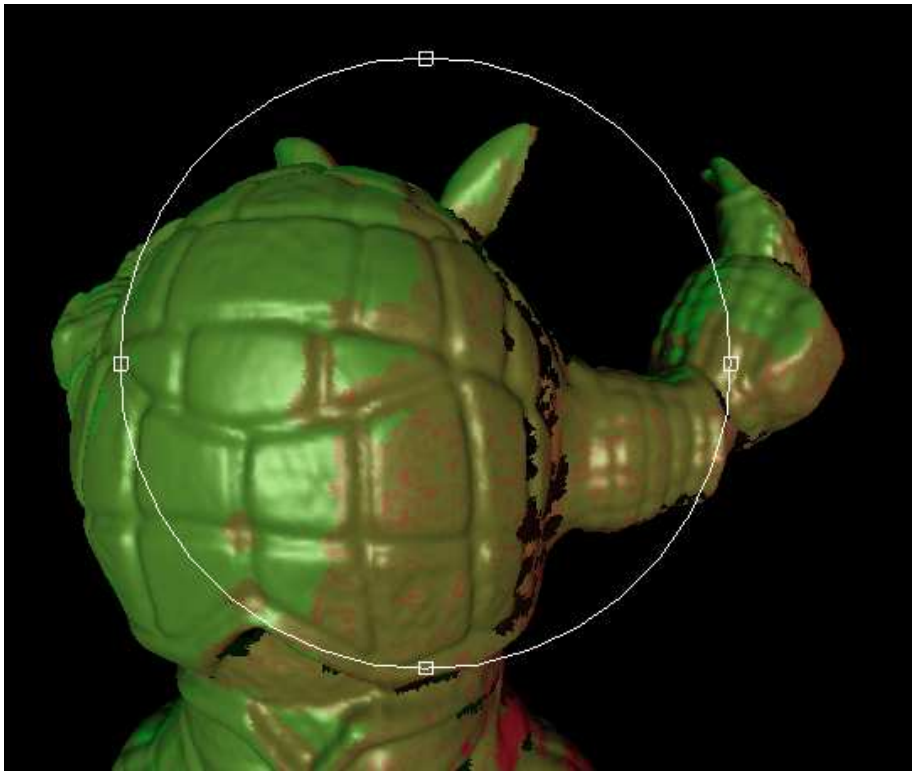

Abbildung 7.4: Ergebnis der Globalregistrierung mit dem Verfahren nach Pulli

Dies gilt ebenfalls für die Dreiecke. Als nächstes Fehlermaß wurde dann die Hausdorffdistanz nach [23] gewählt und implementiert. Die Hausdorffdistanz wurde aufgrund der Ähnlichkeit zum GJK-Algorithmus gewählt. Bei dieser Fehlermetrik wird die maximale Distanz zwischen Objekten berechnet. Während der Implementation fiel auf, dass diese Fehlerberechnung nur sinnvoll ist, wenn der neu berechnete Punkt der Kantenkontraktion nicht der Mittelpunkt der kontrahierten Kante ist. Dies ist wichtig, da sonst die Distanz immer der halben Kantenlänge der kontrahierten Kante entspricht. Da bei der hier durchgeführten Kantenkontraktion der neu entstehende Punkt als Mittelpunkt der alten Kante gewählt wurde, wurde dieses Verfahren aufgegeben. Eine Berechnung der optimalen Lage des neuen Punktes wäre, bezüglich dieses Verfahrens, zu aufwendig gewesen. Als zweite Fehlermetrik wurde dann die Krümmung nach [21] gewählt. Die Implementation erwies sich als relativ einfach wurde aber dennoch leicht abgewandelt. Bei der hier implementierten Form wird nicht die maximale Veränderung der Krümmung der Nachbarpunkte einer Kante als Fehler für diese gewählt, sondern die mittlere Krümmung der Nachbarpunkte. Die Berechnung dieser Fehlermetrik ist schnell, liefert aber früh sehr große Dreiecke. Zu erwähnen wäre noch, dass die Reduktion anhand der Krümmung stark von den Gewichten λ_f, λ_s und λ_a abhängt. Im Vergleich schneidet die Reduktion anhand der Kantenlänge besser ab.

7.3 Server

Für die Evaluierung des Clients wurden zum Zeitpunkt der Evaluierung gängige Notebooks benutzt. Die folgenden Daten geben die Konfiguration der zwei verwendeten Systeme wider:

System 1:

Arbeitsspeicher:	1GB
Prozessor:	Pentium M 1,8 GHz
Grafikkarte:	ATI X700
Konfiguration des Treibers:	1280x800
System:	Kubuntu 6.10 (edgy eft)

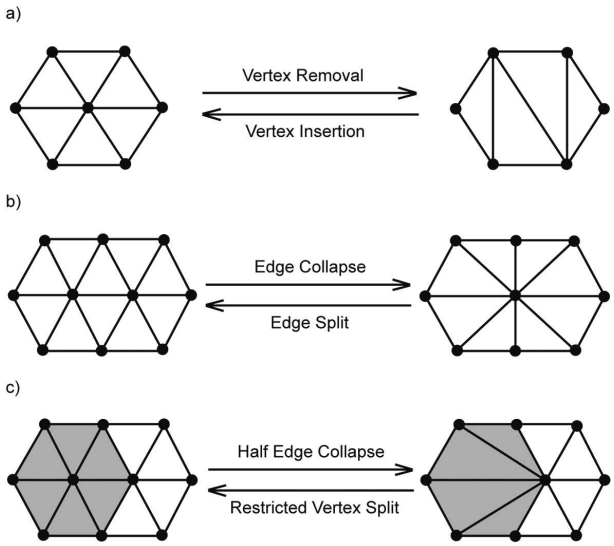

Abbildung 7.5: a) *Vertex Removal* bezeichnet die Knotenentfernung und *Vertex Insertion* bezeichnet die Inverse. b) *Edge Collapse* (*ecol*) bezeichnet die Kantenkontraktion und *Edge Split* bzw. *Vertex Split* (*vsplit*) bezeichnet die inverse Knotenaufspaltung. c) *Half Edge Collapse* bezeichnet die Halbkantenkontraktion und *Restricted Vertex Split* bezeichnet die Inverse [25].

System 2:

Arbeitsspeicher:	1GB
Prozessor:	Pentium Core Duo 2 GHz
Grafikkarte:	ATI FireGL v5200
Konfiguration des Treibers:	1920x1200
System:	Debian 4.0 (etch)

7.3.1 Stabilität

Testobjekte

Für den Stabilitätstest wurden die folgenden Testszenarien erzeugt:

1. Im Modell-Verzeichnis des Servers liegt keine Datendatei. Da der Server ausschließlich auf Basis der Verzeichnisse die Liste der verfügbaren Modelle aufbaut, können auf diese Weise Fehler entstehen.

2. Im Modell-Verzeichnis des Servers liegt zwar eine Datendatei, der Kopiervorgang oder das Erzeugen dieser Datendatei ist jedoch aus irgend einem Grund fehlgeschlagen, so dass die Datendatei leer ist.

3. Die Netzwerkverbindung zwischen dem Client und dem Server wird während einer Übertragung aus einem unbekannten Grund getrennt und bleibt über einen längeren Zeitraum (mehr als eine Minute) getrennt.

4. Die Netzwerk-Verbindung zwischen dem Client und dem Server wird während einer Übertragung aus einem unbekannten Grund getrennt und bleibt über einen kurzen Zeitraum (weniger als eine Minute) getrennt.

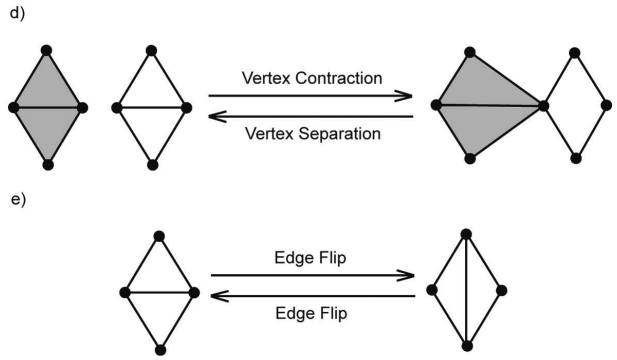

Abbildung 7.6: d) *Vertex Contraction* bezeichnet die Knotenkontraktion und *Vertex Separation* bezeichnet die Inverse. e) *Edge Flip* bezeichnet die Knotenvertauschung [25].

Testergebnisse

1. Die Verbindung wird vom Server getrennt. Eine erneute Verbindung zum Server ist jederzeit möglich.
2. Die Verbindung wird vom Server getrennt. Eine erneute Verbindung zum Server ist jederzeit möglich.
3. Die Verbindung läuft nach 60 Sekunden in einen Timeout. Dadurch wird der `Modelloader` aus dem Speicher entfernt und der Speicher wieder freigegeben. Eine erneute Verbindung zum Server ist nach Wiederherstellung der physikalischen Netzwerkverbindung jederzeit möglich.
4. Die Verbindung stockt, läuft dann aber nach Wiederherstellung der physikalischen Netzwerkverbindung normal weiter. D. h. insbesondere, dass die Anfrage nach Verfeinerungen und die entsprechenden Antworten ohne Eingriff des Benutzers fortgesetzt wird.

7.3.2 Effizienz

Speicherbedarf

Der Speicherbedarf entspricht genau der Größe der Datendatei. Das bedeutet:

1. Für die Anzahl der Dreiecke aller Reduktionsstufen 12 Byte je Dreieck.
2. Für die Anzahl der Punkte aller Reduktionsstufen 12 Byte je Punkt.
3. Für die Liste der Verfeinerungsdreiecke je Basisdreieck und Stufe können nur approximierte Angaben zum Speicherbedarf erfolgen, da diese von der Anzahl und der Stärke der Reduktionsstufen abhängt. Bei fünf Stufen mit einem Fehler, der ungefähr 20% der ursprünglichen Punkte reduziert, ergibt sich ein Betrag von durchschnittlich 12 Byte je Dreieck.

Der Server lief über zwei Monate ohne Unterbrechung auf dem Graphaeology.NET-Server als Dienst und musste zwischenzeitlich zahlreiche Anfragen beantworten. Dabei konnten keine Speicherlecks festgestellt werden, der Speicherbedarf blieb über den gesamten Zeitraum konstant.

Kompression

Die durchschnittlich erzielte Kompressionsrate liegt bei 1:2. Das genügt bei einer heute üblichen DSL-Verbindung für eine brauchbare Übertragungsgeschwindigkeit und belastet den Client und den Server nur in geringer Weise.

7.4 Client

Für die Evaluierung des Clients wurden zwei zum Zeitpunkt der Evaluierung gängige Notebooks verwendet. Die folgenden Daten geben die Konfiguration der zwei verwendeten Systeme wieder:

System 1:

Arbeitsspeicher:	1GB
Prozessor:	Pentium M 1,8 GHz
Grafikkarte:	ATI X700
Konfiguration des Treibers:	1280x800
System:	Kubuntu 6.10 (edgy eft)

System 2:

Arbeitsspeicher:	1GB
Prozessor:	Pentium Core Duo 2 GHz
Grafikkarte:	ATI FireGL v5200
Konfiguration des Treibers:	1920x1200
System:	Debian 4.0 (etch)

7.4.1 Stabilität

Testobjekte

Für den Stabilitätstest wurden die folgenden Testszenarien erzeugt:

1. Im Modell-Verzeichnis des Servers liegt keine Datendatei. Da der Server ausschließlich auf Basis der Verzeichnisse die Liste der Verfügbaren Modelle aufbaut, können auf diese Weise Fehler entstehen.
2. Im Modell-Verzeichnis des Servers liegt zwar eine Datendatei, der Kopiervorgang oder das Erzeugen dieser Datendatei ist jedoch aus irgend einem Grund fehlgeschlagen, so dass die Datendatei leer ist.
3. Die Netzwerk-Verbindung zwischen dem Client und dem Server wird während einer Übertragung aus einem unbekannten Grund getrennt und bleibt über einen längeren Zeitraum (mehr als eine Minute) getrennt.
4. Die Netzwerk-Verbindung zwischen dem Client und dem Server wird während einer Übertragung aus einem unbekannten Grund getrennt und bleibt über einen kurzen Zeitraum (weniger als eine Minute) getrennt.
5. Der Server wird aus einem unbekannten Grund während der Übertragung eines Modells beendet.

Testergebnisse

Die Testergebnisse der oben aufgeführten Szenarien wurden mit den beiden oben aufgeführten Systemen erzeugt. Dabei wurde der Client auf dem Rechner 1 und der Server auf dem Rechner 2 gestarten.

Die Ergebnisse sind:

1. Die Verbindung wird vom Server getrennt. Der Client liefert eine entsprechende Statusmitteilung und läuft weiter. Es kann ohne Probleme auf Objekte des lokalen Caches zugegriffen werden. Eine erneute Verbindung zum Server ist jederzeit möglich.
2. Die Verbindung wird vom Server getrennt. Der Client liefert eine entsprechende Statusmitteilung und läuft weiter. Es kann ohne Probleme auf Objekte des lokalen Caches zugegriffen werden. Eine erneute Verbindung zum Server ist jederzeit möglich.

3. Die Verbindung läuft nach 60 Sekunden in einen Timeout. Der Verbindungsstatus des Clients wechselt von *Verbunden* zu *Getrennt* und der Client liefert eine entsprechende Statusmitteilung Es kann ohne Probleme auf Objekte des lokalen Caches zugegriffen werden. Eine erneute Verbindung zum Server ist nach Wiederherstellung der physikalischen Netzwerkverbindung jederzeit möglich.

4. Die Verbindung stockt, läuft dann aber nach Wiederherstellung der physikalischen Netzwerkverbindung normal weiter. D. h. insbesondere, dass die Anfrage nach Verfeinerungen und die entsprechenden Antworten ohne Eingriff des Benutzers fortgesetzt wird.

5. Die Verbindung läuft nach 60 Sekunden in einen Timeout. Der Verbindungsstatus des Clients wechselt von *Verbunden* zu *Getrennt* und der Client liefert eine entsprechende Statusmitteilung Es kann ohne Probleme auf Objekte des lokalen Caches zugegriffen werden. Eine erneute Verbindung zum Server ist nach Wiederherstellung des Servers jederzeit möglich.

7.4.2 Effizienz

Speicherbedarf

Der Speicherbedarf ist entsprechend der Größe er Binärdatei. Das bedeutet:

1. Für die Anzahl der Dreiecke aller Reduktionsstufen 12 Byte je Dreieck.

2. Für die Anzahl der Punkte aller Reduktionsstufen 12 Byte je Punkt.

3. Für die Liste der Verfeinerungsdreiecke je Basisdreieck und Stufe können nur approximierte Angaben zum Speicherbedarf erfolgen, da diese von der Anzahl und der Stärke der Reduktionsstufen abhängt. Bei fünf Stufen mit einem Fehler, der ungefähr 20% der ursprünglichen Punkte reduziert, ergibt sich ein Betrag von durchschnittlich 12 Byte je Dreieck.

Geschwindigkeit

Für die Messung der Geschwindigkeit wurde ein *Framecounter* in das OpenGL-Widget eingebaut. Dieser Zähler bestimmt die möglichen Aktualisierungen pro Sekunde. Als Testobjekt dient ein Modell mit dem Namen „HeadMax“. Dieses Modell besitzt 446720 Dreiecke.

Anzahl der Bilder pro Sekunde:

$$\begin{array}{ll} \text{Rechner 1:} & \approx 80 \text{ fps} \\ \text{Rechner 2:} & \approx 100 \text{ fps} \end{array}$$

Kapitel 8

Zusammenfassung und Ausblick

Zusammenfassung

Mit der Zusammenfassung mehrerer völlig unterschiedlicher Registrierungsverfahren in einem einzigen Programm ist es der Projektgruppe gelungen verschiedene Verfahren gleichzeitig einzusetzen und gegenüberzustellen. Da es bis dato kein Registrierungsverfahren gibt, welches diese Aufgabe bei jeder Art von Scan in zufriedenstellender Qualität und Laufzeit bewältigt, war es wichtig zumindest die wichtigsten bisher veröffentlichten Verfahren zur Verfügung zu stellen. Die Gegenüberstellungen haben gezeigt, dass eine vollautomatische Registrierung zwar in vielen Fällen möglich ist, dabei aber eine lange Laufzeit von mehreren Stunden in Kauf genommen werden muss. Des Weiteren hat sich gezeigt, dass die Information, wie die Scans in welcher Ausrichtung erstellt wurden, sehr wichtig ist und sich nur schwer rekonstruieren lässt. In dem Verfahren wurde auf Zusatzinformationen dieser Art verzichtet, da einige Verfahren diese Informationen nicht einzusetzen vermögen. Allgemein geht man davon aus, dass die Kenntnis der Orientierung des Objekts im Scanner jedoch den Suchraum bei der Registrierung erheblich verkleinern kann. Leider hat sich auch gezeigt, dass einige vielversprechende Verfahren für eine vollständige Registrierung gänzlich ungeeignet sind. In den zugehörigen Artikeln der Autoren, die diese Verfahren entwickelt haben, wird dennoch auf die universelle Einsetzbarkeit bei extrem kurzen Laufzeiten hingewiesen.

Da das Registrierungswerkzeug jedoch unter jedem Umstand, unabhängig von der Qualität der Scans, ein zufriedenstellendes Endergebnis liefern sollte, war die Implementierung einer manuellen Grobregistrierung zwingend erforderlich. Diese ist sehr benutzerfreundlich und führt zu guten Ergebnissen, die zusätzlich noch mit der implementierten Feinregistrierung verbessert werden können. So ist es auch möglich Scans zusammenzusetzen, welche untereinander nur einen geringen Überlappungsbereich aufweisen.

Aufgrund der fehlenden Oberflächenrekonstruktion ist der Übergang zwischen globalregistriertem Modell und Reduktion in diesem Bericht nicht beschrieben.

Bei der Implementierung des Client- und Servermoduls gab es hingegen weniger Probleme. Die Übertragung der Modelle zwischen Client und Server läuft effizient und zuverlässig. Zur Datenkompression wurden bewährte Techniken eingesetzt. Der Algorithmus der Darstellung des Modells im Client wurde bewusst sehr einfach und sehr schnell gehalten, um eine flüssige Darstellung im Client zu erreichen. Die Speicherverwaltung sowohl im Client, als auch im Server ist ebenfalls effizient, so dass auch große Modelle auf heute verfügbaren Heim-PCs dargestellt werden können.

Ausblick

Da es immer noch keine vollautomatische, schnelle und zuverlässige Registrierung gibt, ist anzunehmen, dass dieser Bereich noch weiter erforscht werden kann. Allerdings ist hier auch anzumerken, dass die Scanner, die heute entwickelt werden, immer schneller werden und damit Scans mit großem Überlappungsbereich in kurzer Zeit erzeugen können, was für jedes Registrierungsverfahren nur von Vorteil sein kann. Zusätzlich liefern hochwertige Scanner auch die Position und Ausrichtung der Scaneinheit, so dass eine vollständige Registrierung noch weiter vereinfacht werden kann.

Selbstverständlich ist es möglich jedes einzelne Verfahren weiter zu optimieren oder sogar mehrere Verfahren zu kombinieren. So ist es z. B. vorstellbar, dass das Spinimageverfahren zusätzlich Krümmungswerte der einzelnen Punkte berücksichtigt, um so schneller zu sicheren Ergebnissen zu kommen.

Das Reduktionsverfahren könnte ebenfalls weiter verfeinert werden, so dass z. B. Inschriften in den Artefakten durch die Reduktion erkannt und so lange wie möglich geschützt werden. Somit ist es möglich auch bei einer geringen Auflösung Schriften abzulesen.

Die Module Client und Server sind ebenfalls erweiterbar. Im Client könnte man sich vorstellen, dass auch Objekte darstellbar sind, die von ihrer Größe bereits nicht mehr in den Arbeitsspeicher des PCs passen. Hierbei muss natürlich mit Einschränkungen in der Darstellungsgeschwindigkeit gerechnet werden. Eine weitere Möglichkeit den Client und Server zu erweitern wäre eine detailliertere Benutzerverwaltung, die es ermöglicht verschiedene Objekte nur bestimmten Nutzern zur Verfügung zu stellen. Des Weiteren wäre es möglich innerhalb der Objekte Markierungen zu setzen, die auch anderen Betrachtern eingeblendet werden könnten.

Anhang A

Handbuch

A.1 Kurzbeschreibung

Das Programmpaket Graphaeology.NET wurde von der Projektgruppe 489 des Fachbereichs Informatik der Universität Dortmund entwickelt. Es hat die Aufgabe, reale Objekte, speziell archäologische Steine, so aufzubereiten, dass sie in digitaler Form über das Internet zu betrachten sind.

Damit das Programmpaket die Steine digital verarbeiten kann, müssen zu jedem Objekt mehrere Höhenmodelle, zweieinhalbdimensionale Netze, als Datei im OFF-Format vorliegen. Diese Modelle könnten z. B. mit Hilfe eines taktilen oder eines Laserscanners gewonnen werden.

Diese Netze können dann von dem ersten Modul von Graphaeology.NET, dem Registrierungswerkzeug, zu einem einzelnen dreidimensionalen Modell zusammengesetzt werden. Dieses Modell wird in einem programmspezifischen Dateiformat B.1 abgespeichert, um es später optimal übertragen zu können.

Das zweite Modul, der Server, hat die Aufgabe diese Modelle über das TCP/IP-Protokoll zur Verfügung zu stellen und bei Anfrage progressiv zu übertragen.

Das dritte Programm im Paket ist der Client mit dem die Modelle über das Internet vom Server angefordert und in einem Betrachtungsfenster dargestellt werden können.

A.2 Lizenz

Die von der Projektgruppe entwickelte Software unterliegt der GNU General Public License (GPL).

Da Graphaeology.NET selbst Software nutzt die der GPL und der LGPL unterliegt, kommt nur die nichtveröffentlichung der Software oder die Veröffentlichung unter der GPL in betracht. Die GPL ist ein Lizenzmodell für freie Software und wurde von der Free Software Foundation entwickelt. Dabei gelten im Wesentlichen die folgenden Regeln:

- Die Software darf uneingeschränkt für jeden Zweck genutzt werden.
- Die Software (mit Quellcode) darf kostenlos verteilt werden.
- Die Software darf verändert und für eigene Bedürfnisse angepasst werden.
- Die veränderte Software unterliegt den gleichen Regeln wie die Ausgangssoftware (kostenlos, Weitergabe und Quellcode offenlegen etc.)

Aus diesem Grunde liegt die entwickelte Software als Quellcode vor. Es wird darauf verzichtet die komplette GPL ins Handbuch aufzunehmen. Eine Kopie der GPL ist in dem Quellcode enthalten. Der genaue Text der GPL befindet sich auch im Internet unter [8]. Desweiteren gibt es eine inoffizielle deutsche Übersetzung der GPL, welche unter [9] zu finden ist.

A.3 Installation

Für die Installation sämtlicher Programme der Projektgruppe sind die folgenden Abhängigkeiten auf dem jeweiligen System zu erfüllen:

- GCC mit Fortran in Version ≥4.0
- QT in Version ≥4.1
- fftw3
- zlib-develop

Es sind weitere Bibliotheken notwendig (siehe Anhang C). Diese werden jedoch nur lokal installiert. Daher wird in den jeweiligen Abschnitten auf diese Bibliotheken hingewiesen und beschrieben wie diese zu installieren sind.

A.3.1 Installation des Registrierungswerkzeugs für Linux

Das Registrierungswerkzeug liegt als Archivdatei vor, welche erstmal entpackt werden muss. Dafür genügt der Aufruf von `tar -jxvf regtool.tar.bz2`. Nun müssen einige, mit dem Registrieungswerkzeug mitgelieferte, Bibliotheken installiert werden.

Installation der Bibliotheken

Vor der Übersetzung des Registrierungswerkzeugs werden vier Bibliotheken installiert. Dieses wird in diesem Abschnitt erläutert. Zuerst wird die TetGen-Bibliothek übersetzt. Der Quelltext befindet sich unter `/regtool/tetgen`. In diesem Verzeichnis ist `make tetlib` auszuführen. Nun sollte die ANN-Bibliothek übersetzt werden. Der Quelltext hierzu findet sich unter `/regtool/ann`. Hier muss nun `make linux-g++` ausgeführt werden. Anschließend ist GotoBLAS zu kompilieren. Dazu ist die Datei `/regtool/lgslibs/UFconfig/UFconfig.mk` zu editieren. Sollte auf dem System die BLAS-Bibliothek installiert sein, so sollte aus der Zeile `UMFPACK_CONFIG` der Schalter `-DNBLAS` entfernt werden. Dies führt zu einer möglichen Beschleunigung des Registrierungswerkzeugs. Nun muss in das Verzeichnis `/regtool/lgslibs/GotoBLAS` gewechselt werden und dort `quickbuild.32bit` oder `quickbuild.64bit`, je nach verwendeter Architektur, ausgeführt werden. Nach abgeschlossener Kompilierung kann mit der Installation von UMFPACK fortgefahren werden. Dafür genügt nach dem Wechsel zu `/regtool/lgslibs/UMFPACK` das ausführen von `make`. Nun sind alle erforderlichen Abhängigkeiten übersetzt und das eigentliche Registrierungswerkzeug kann kompiliert werden.

Kompilieren des Registrierungswerkzeugs

Nach einem Wechsel zurück in das Verzeichnis `/regtool` kann das Kompilieren des Registrierungswerkzeugs durch Aufruf von `./rebuild` begonnen werden. Nach erfolgreicher Übersetzung steht das Registrierungswerkzeug unter `/regtool/bin` zur Verfügung und kann hier durch den Aufruf von `./regtool` gestartet werden. Zu der Benutzung des Registrierungswerkzeugs sei auf den Anhang A.4 verwiesen.

A.3.2 Installation des Client- und Server-Moduls für Linux

Die Installation des Client- und Server-Moduls ist einfach. Zunächst ist die entsprechende Archivdatei zu entpacken. Dies geschieht durch Eingabe von `tar -jxvf clientserver.tar.bz2`. Dann kann mit der Übersetzung begonnen werden, indem in `/clientserver` einfach `./rebuild` aufgerufen wird. Nun stehen unter `/clientserver/bin` die entsprechenden Programme zur Verfügung. Zu ihrer Bedienung sei hier auf die Anhänge A.5 und A.6 verwiesen.

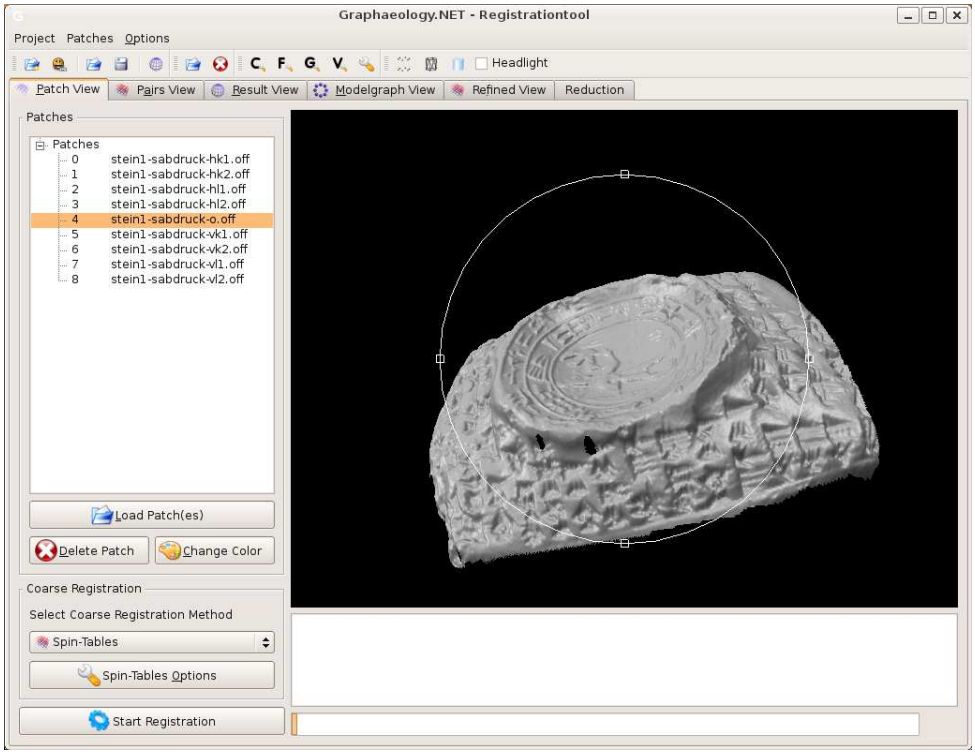

Abbildung A.1: Registrierungswerkzeug: Screenshot der Scan-Ansicht

A.4 Registrierungswerkzeug

Das Registrierungswerkzeug wurde entwickelt, um aus einer Reihe von zusammengehörigen, zweieinhalbdimensionalen Scans eines realen Gegenstandes ein 3D-Objekt zu erzeugen, das diesen Gegenstand vollständig darstellt. Um dieser Anforderung gerecht zu werden, wurde das Programm in verschiedene Teile aufgeteilt. Für eine möglichst intuitive Bedienung des Programms sind die einzelnen Schritte unter jeweils einem eigenen Karteikartenreiter untergebracht. Man kann so Schritt für Schritt die notwendigen Aufgaben ausführen.

A.4.1 Ablauf

Laden → Grobregistrierung → Feinregistrierung → Globalregistrierung → Export → Reduktion

Zuerst importiert man alle benötigten Objekte (im Folgenden *Scans* oder *Patches*) in das Projekt. Dann versucht die Grobregistrierung alle Paarungen von zwei Scans möglichst gut gegeneinander auszurichten. Die Feinregistrierung verbessert das Ergebnis der Grobregistrierung. Die Globalregistrierung ermittelt danach aus den Paarungen die Lage der einzelnen Scans im zu erzeugenden Ergebnisobjekt. Ist das Ergebnis zufriedenstellend, so können alle Objekte mit der Export-Funktion zu einem neuen Objekt verschmolzen werden. Möchte man dieses Objekt mit Hilfe des Servers bereitstellen, so können auch noch die verschiedenen Reduktionsstufen erzeugt werden.

Das Programm ist in sechs Ansichten gegliedert, welche nun erläutert werden.

A.4.2 Patch View

In der ersten Ansicht, dem *Patch View* (siehe Abb. A.1), wird festgelegt, welche Objekte zum Projekt gehören. Hierzu werden die folgenden Funktionen angeboten:

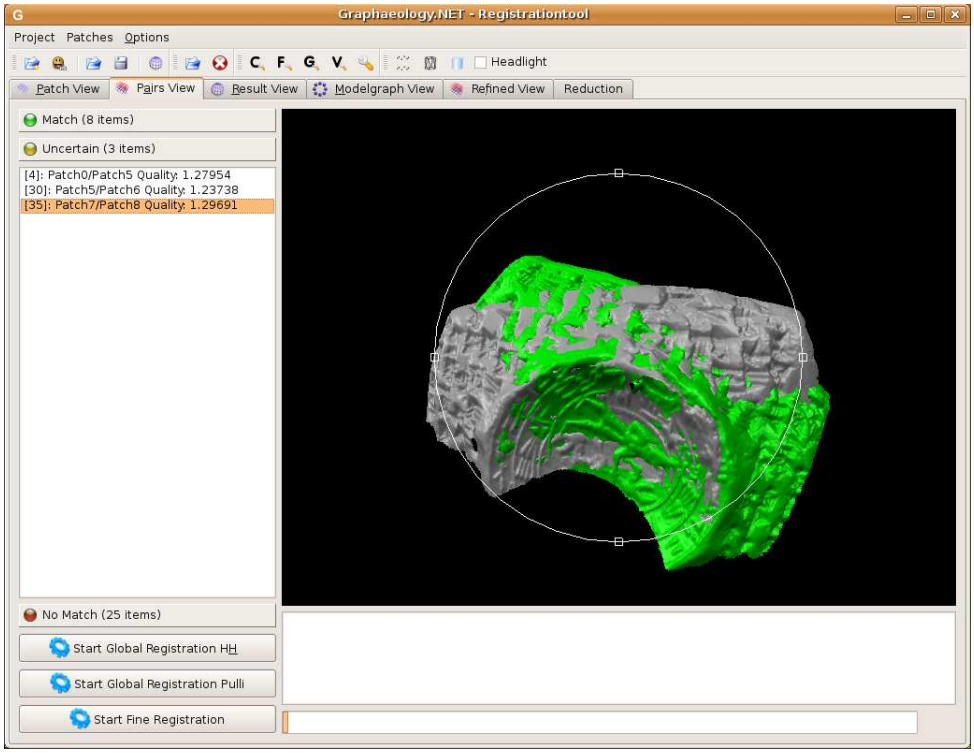

Abbildung A.2: Registrierungswerkzeug: Screenshot der Paarungs-Ansicht

- **Load Patch(es)** – Öffnet einen Dialog zum Laden weiterer Scans.
- **Delete Patch** – Löscht den ausgewählten Scan.
- **Change Color** – Ändert die Darstellungsfarbe der Scans in dieser Ansicht.

Die geladenen Dateien werden auf der linken Seite des Fensters aufgelistet. Um sich ein Objekt anzeigen zu lassen, muss auf den entsprechenden Dateinamen geklickt werden, wodurch es im großen rechten Bereich des Hauptfensters dargestellt wird. Durch eine Trackball-Navigation, die im Anhang A.6.3 erklärt wird, kann die Ansicht des Objekts durch den Benutzer beliebig verändert werden.

Nach dem Laden der Scans wird die Grobregistrierung aller Scans gestartet. Hierzu kann man zuerst mittels des Auswahlfelds **Select Coarse Registration Method** das zu verwendende Grobregistrierungsverfahren auswählen. Mit dem Button darunter gelangt man zum Optionen-Dialog[1] des gewählten Verfahrens (zum Beispiel **Spin-Table Options**). Wenn alle Einstellungen vorgenommen sind, kann man über den Button **Start Registration** die Grobregistrierung starten. Dies kann abhängig vom gewählten Verfahren und der Anzahl der Scans sehr lange dauern.

Ist die Grobregistrierung abgeschlossen, wechselt das Programm automatisch zur nächsten Ansicht, dem *Pairs View* (siehe Abb. A.2).

A.4.3 Pairs View

Die Grobregistrierung erzeugt zu jeder Paarung zweier Scans eine möglichst gute Transformation und bewertet diese auch, siehe Kapitel 3.1.11. Je nach Güte der ermittelten Transformation, wird ein Paar in die Gruppe *Match* (gut), *Uncertain* (unsicher) oder *No match* (keine gute Transformation gefunden) einsortiert.

[1] Eine genaue Erklärung des Dialogs erfolgt in Anhang A.4.9.

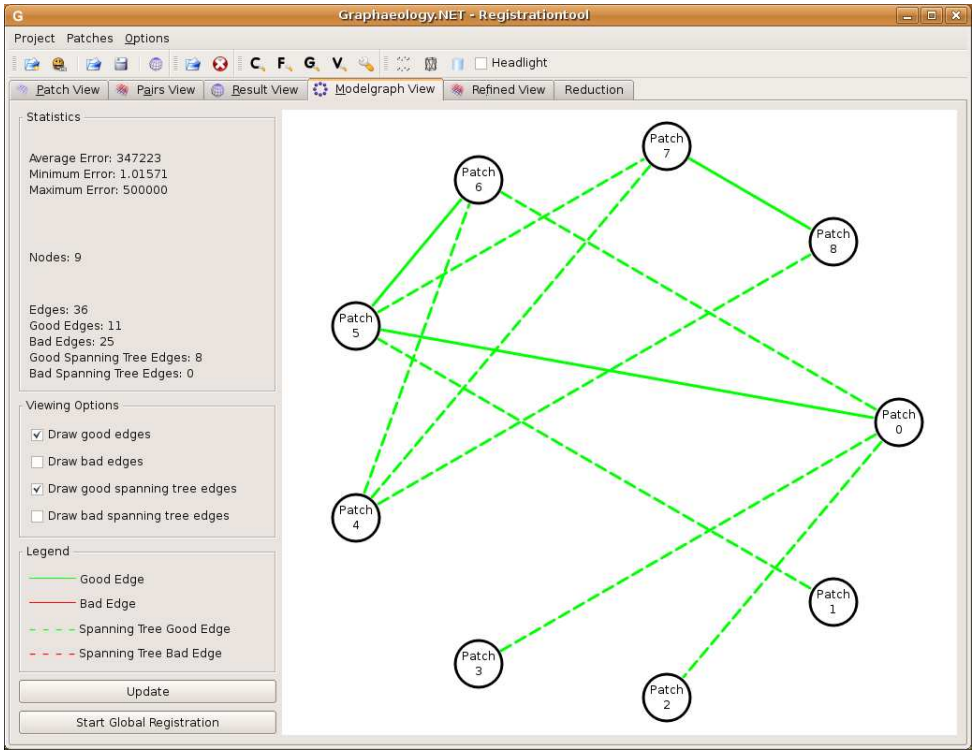

Abbildung A.3: Registrierungswerkzeug: Screenshot der Modellgraf-Ansicht

Klickt man mit der rechten Maustaste auf eines dieser Paare, so erscheint ein Kontext-Menü, welches die folgenden Möglichkeiten bietet: Zum einen kann eine Paarung über **Move to ...** in eine der beiden anderen Gruppen verschoben werden. Zum anderen kann man für eine Paarung ein beliebiges Grobregistrierungsverfahren erneut manuell starten oder eine Feinregistrierung durchführen. Außerdem kann eine Paarung gelöscht werden. Für Paarungen, bei denen kein automatisches Grobregistrierungsverfahren zu einem Erfolg führt, ist es möglich eine manuelle Grobregistrierung zu starten, wodurch automatisch zum *Refined View* gewechselt wird.

Falls man mit der Registrierung aller Paarungen zufrieden ist, wird durch **Start Global Registration HH** bzw. **Start Global Registration Pulli** die in Kapitel 3.3 beschriebene Globalregistrierung nach Huber und Hebert [19] bzw. Pulli [29] gestartet.

A.4.4 Modelgraph View

In der Ansicht *Modelgraph View* (Abb. A.3) kann man sich anzeigen lassen, welche Paarungen gut oder schlecht bewertet und welche von ihnen für die Globalregistrierung verwendet wurden. Die Knoten in dem Modellgrafen repräsentieren die einzelnen Scans. Eine Kante zwischen zwei Knoten zeigt an, dass eine Grobregistrierung für die entsprechenden Scans berechnet wurde. Grüne Kanten symbolisieren passende, rote Kanten nicht passende Kombinationen. Die gestrichelten Kanten zeigen den Spannbaum, den die Globalregistrierung erstellt hat.

A.4.5 Refined View

Hier lässt sich eine manuelle Grobregistrierung durchführen (siehe Abb. A.4). Hierzu selektiert man bei gedrückter Strg-Taste je drei korrespondierende Punkte in beiden Scans. Um die Scans zu drehen oder verschieben kann wiederum auf die Trackball-Navigation zurückgegriffen werden. Nach erfolgreicher Auswahl der Punkte klickt man auf **Transform**. Über **Fine Registration** kann man das erzielte Ergebnis weiter

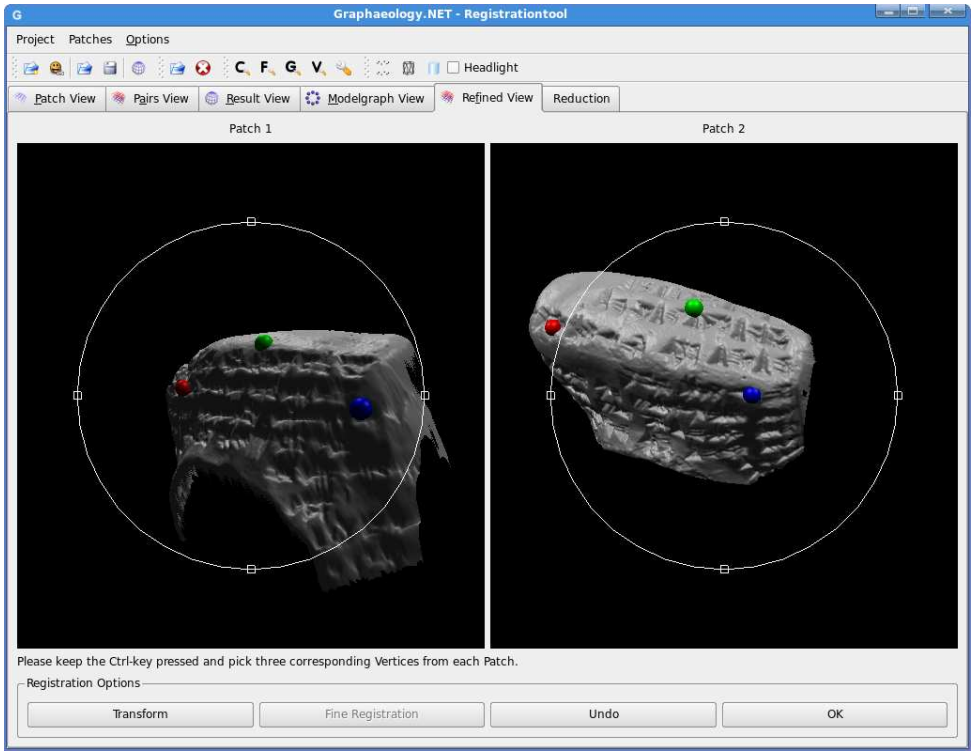

Abbildung A.4: Registrierungswerkzeug: Screenshot der manuellen Registrierungsansicht

verfeinern. Ist das Ergebnis zufriedenstellend, kann die Änderung an der Paarung mit **OK** übernommen werden.

A.4.6 Reduction

Damit das bei der Globalregistrierung entstandene komplette Modell dem Server zur Verfügung gestellt werden kann, muss es in verschiedenen Qualitätsstufen abgespeichert werden. Das dafür benötigte Reduktionswerkzeug wird unter dem Karteikartenreiter *Reduction* (Abb. A.5) bereitgestellt. Hier kann u.a. die Anzahl der zu erzeugenden Stufen, der maximale Fehler und die Anzahl der zu reduzierenden Dreiecke vom Benutzer variiert werden. Mit **Load Object** wird das Modell geladen und durch **Start Reduction** wird das Modell dann entsprechend den Vorgaben reduziert. **Save Structure** speichert schließlich das reduzierte Modell so, dass es vom Server eingelesen werden kann. Wenn nötig, kann die gesamte Struktur durch **Reset** zurückgesetzt werden.

A.4.7 Die Werkzeugleiste

Die Werkzeugleiste bietet dem Nutzer einen direkten Zugriff auf viele nützliche Funktionen und Optionen, ohne dass diese über das Menü erreicht werden müssen. Anbei ist die Werkzeugleiste abgebildet (Abb. A.6), deren Funktionen nun erläutert werden:

1. Beginnt ein neues Projekt
2. Öffnet den Projekt-Assistenten
3. Lädt ein Projekt
4. Speichert das aktuelle Projekt
5. Exportiert das Ergebnis als OFF-Datei

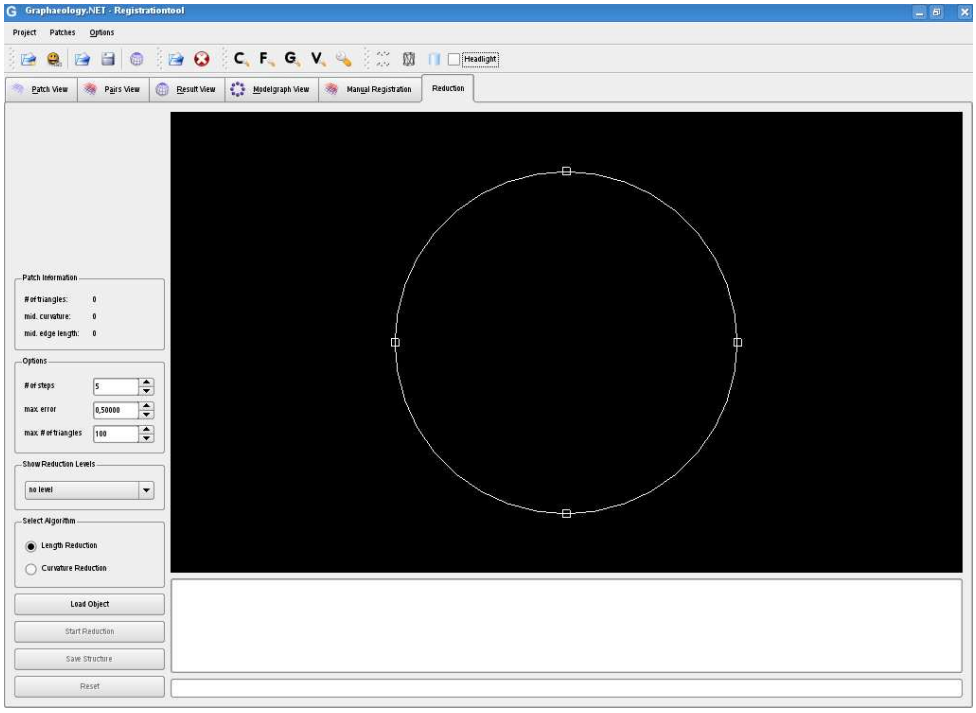

Abbildung A.5: Registrierungswerkzeug: Screenshot der Reduktionsansicht

Abbildung A.6: Registrierungswerkzeug: Die Werkzeugleiste

6. Lädt ein Patch in das Projekt
7. Löscht ein Patch aus dem Projekt
8. Zeigt die Grobregistrierungs-Optionen
9. Zeigt die Feinregistrierungs-Optionen
10. Zeigt die Globalregistrierungs-Optionen
11. Zeigt die Ansichts-Optionen
12. Zeigt die sonstigen Optionen (z. B. Schwellwerte für den Kantenfilter)
13. Wechselt zur Punktwolkenansicht eines Objektes
14. Wechselt zur Drahtgitteransicht eines Objektes
15. Wechselt zur Flächenansicht eines Objektes
16. Schaltet eine Kopflampe zur besseren Ausleuchtung des Objektes an/aus

A.4.8 Export

Um das bei der Globalregistrierung entstandene Modell nicht nur im Client-Modul betrachten zu können wird zusätzlich eine Exportfunktion angeboten. Diese speichert das Modell im OFF-Dateiformat, das von verschiedensten 3D-Betrachtungsprogrammen eingelesen werden kann. Diese Exportfunktion erreicht man im Hauptmenü im Abschnitt Projekt unter **Export to OFF-File...**.

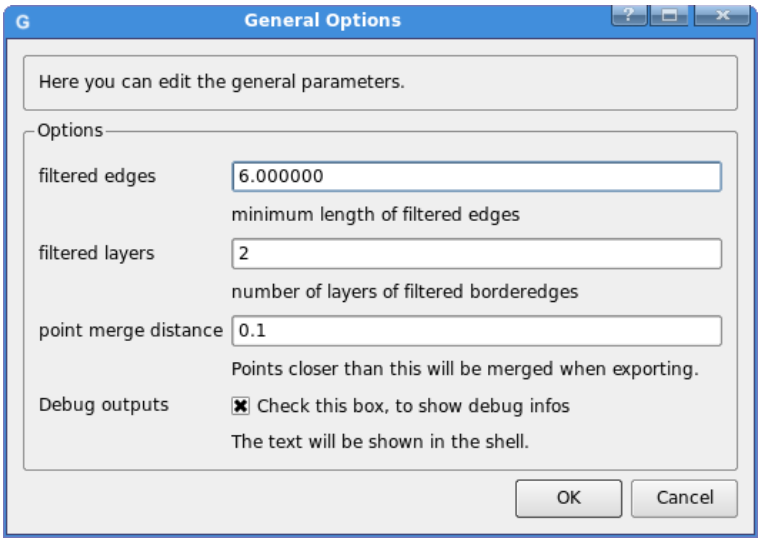

Abbildung A.7: Registrierungswerkzeug: Screenshot des Dialogs für allgemeine Optionen

A.4.9 Optionen

Die verschiedenen Unterpunkte der Optionen-Dialoge für Grobregistrierung, Feinregistrierung, Globalregistrierung, Ansicht (*Viewing Options*) und Allgemeines (*General Options*) werden in diesem Abschnitt erläutert.

Allgemeines

Allgemeine Einstellungen für das Registrierungswerkzeug können im *General Options*-Dialog (Abb. A.7) geändert werden. Dieser ist über das Menü im Abschnitt *Options* zu erreichen.

- **Filtered Edges** – Alle Kanten in Objekten, die länger sind als der angegebene Wert, werden herausgefiltert.
- **Filtered Layers** – Anzahl der Ebenen vom äußeren Rand der Scans gesehen, die vom Kantenfilter betrachtet werden.
- **Point Merge Distance** – Beim Export werden Punkte verschiedener Scans, welche näher beieinander liegen als der angegebene Wert, zu einem Punkt verschmolzen. Hier empfiehlt sich die Abtastauflösung des Scans anzugeben.
- **Debug outputs** – Legt fest, ob das Programm auf der Konsole Statusmeldungen ausgeben oder sich still verhalten soll. Die Statusmeldungen können hilfreich sein, da das Programm teilweise sehr lange rechnet.

Grobregistrierung

Um die Parameter der einzelnen Grobregistrierungsverfahren einzusehen bzw. zu ändern, klickt man entweder im *Patch View* auf den Optionen-Button oder man klickt auf *Coarse Registration Options* unter Optionen im Hauptmenü. Dadurch öffnet sich das Dialogfenster aus Abbildung A.8.

- **Global Settings**
  - **Method** – Welche Methode soll standardmäßig zur Grobregistrierung verwendet werden?
  - **Rating** – Welches Bewertungsverfahren soll standardmäßig verwendet werden?

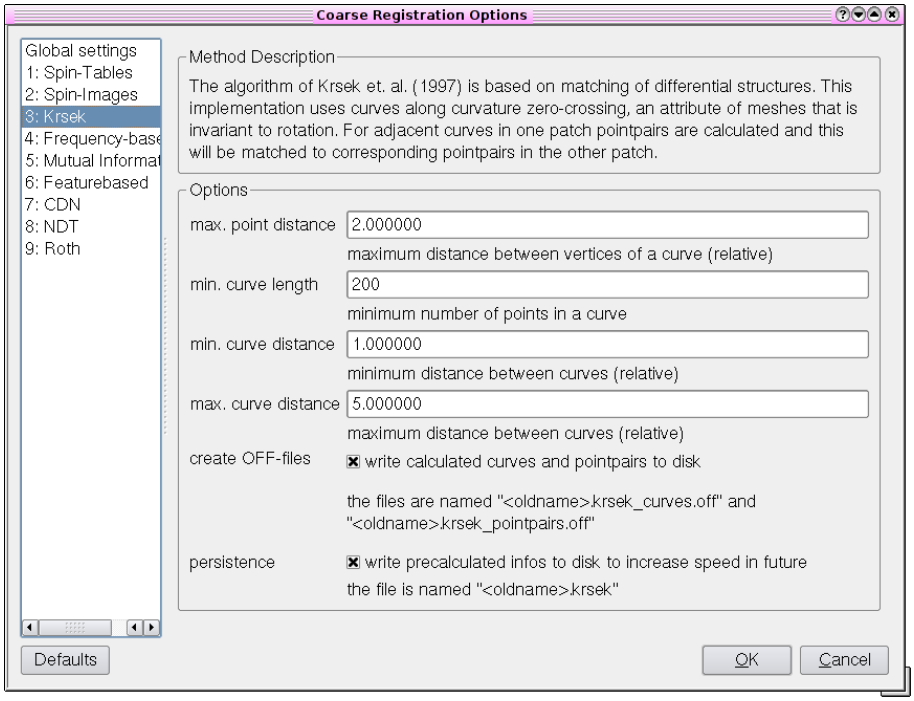

Abbildung A.8: Registrierungswerkzeug: Screenshot des Dialogs für Grobregistrierungs-Optionen

  - **Percent** – Prozentsatz der zur Bewertung benutzten Punkte.
  - **Covering** – Gibt an, zu wieviel Prozent sich die Patches einer Paarung mindestens überlappen müssen.
  - **Epsilon** – Gibt die Größe der Epsilon-Umgebung an.
  - **Defaults** – Wähle eine Vorgabe für die Parameter des Bewertungsverfahrens.
  - **Load defaults** – Lade die selektierte Vorgabe.

- Spin-Tables Verfahren
  - **loops** – Anzahl der Durchläufe

- Spin-Images Verfahren
  - **bin size** – Gibt die Größe eines Array-Feldes an.
  - **num of bins** – Definiert die Größe eines Spin-Images.
  - **skip** – Mit diesem Parameter kann angegeben werden zu jedem wievielten Punkt ein Spin-Images erzeugt werden soll.
  - **nodes patch 1** – Bestimmt die Anzahl der zu suchenden Ausreißer-Korrelationen für Scan 1.
  - **nodes patch 2** – Bestimmt die Anzahl der zu suchenden Ausreißer-Korrelationen für Scan 2.
  - **best correlation** – Definiert eine obere Schranke für Ausreißer.
  - **runaways** – Gibt den Abstand zur nächstkleineren Korrelation in Prozent an.
  - **overlapping** – Möglichkeit zum Einstellen des benötigten prozentualen Überlappungsbereichs.
  - **catched points** – Gibt die Mindestanzahl an aufgesammelten Knoten im Spin-Image in Relation zum Spin-Image mit den meisten Knoten an, um die Nutzung leerer Spin-Images zu verhindern.
  - **min bins** – Bestimmt den prozentualen Anteil der Array-Elemente mit mindestens einem Punkt.

- Verfahren nach Krsek

  - **max. point distance** – Definiert den maximalen Abstand zwischen Knoten einer Kurve, relativ zum durchschnittlichen Knotenabstand.
  - **min. curve length** – Hier kann die minimale Anzahl an Knoten in einer Kurve eingestellt werden.
  - **min. curve distance** – Bestimmt den minimalen Abstand benachbarter Kurven, relativ zum durchschnittlichen Knotenabstand.
  - **max. curve distance** – Bestimmt den maximalen Abstand benachbarter Kurven, relativ zum durchschnittlichen Knotenabstand.
  - **create .OFF files** – Durch Auswahl dieser Option werden die berechneten Kurven und Punktpaare als OFF-Dateien auf die Festplatte geschrieben, so dass sie vom Anwender analysiert werden können.
  - **persistence** – Schreibt die Vorberechnungen für einzelne Scans auf die Festplatte, so dass eine erneute Registrierung desselben Scans beschleunigt wird.

- Frequenzbasiertes Verfahren
  - **bin size** – Gibt die Größe eines Array-Feldes an.
  - **num of bins** – Definiert die Größe eines Spin-Images.
  - **skip** – Mit diesem Parameter kann die Anzahl der zu überspringenden Spin-Images definiert werden.
  - **runaways** – Gibt den Abstand zur nächstkleineren Korrelation in Prozent an.
  - **bounding box intersect** – Dieser Parameter gibt an, wie fein das Objekt durch die Hüllquader zerteilt wird. Die Hüllquader dienen der disjunkten Objektzerlegung, um schnell auf bestimmte Bereiche des Netzes zugreifen zu können. Je größer der Wert für diesen Parameter gewählt wird, desto feiner wird das Netz zerlegt. Standard ist vier, während Werte größer zehn nicht empfehlenswert sind.
  - **voxel space size** – Dieser Parameter gibt die Größe der Voxel an.

- Mutual-Information Verfahren
  - **number of samples** – Gibt die Anzahl der Stichproben an.
  - **scale factor** – Bestimmt den Skalierungsfaktor.
  - **sort of MI** – Hiermit kann zwischen dem klassischen und dem normalisierten Verfahren ausgewählt werden

- Merkmalsbasiertes Verfahren
  - **distance tolerance** – Dieser Wert definiert die tolerierte Abweichung für den Abstand zweier Punkte.
  - **curvature tolerance** – Bis zu diesem Toleranzwert werden Abweichungen der Krümmungen akzeptiert.
  - **curvature threshold** – Mit diesem Parameter kann durch Angabe eines Schwellwerts die Anzahl der berücksichtigten Krümmungen verringert werden.

- Normalverteilungstransformation nach Biber und Strasser
  - **dimension** – Mit diesem Parameter kann die Art des Verfahrens ausgewählt werden: Das 2D-Verfahren entspricht dem ursprünglichen Algorithmus von Biber und Strasser und arbeitet auf parametrisierten Scans. Das 3D-Verfahren ist eine angepasste Version, um direkt auf den Scans arbeiten zu können.
  - **ratio** – Definiert den relativen Anteil der zu verwendenen Knoten. Kleinere Werte erhöhen die Geschwindigkeit, aber verringern die Qualität.
  - **cellcount x** – Dieser Wert gibt die Anzahl der Zellen in x-Richtung an.
  - **cellcount y** – Dieser Wert gibt die Anzahl der Zellen in y-Richtung an.

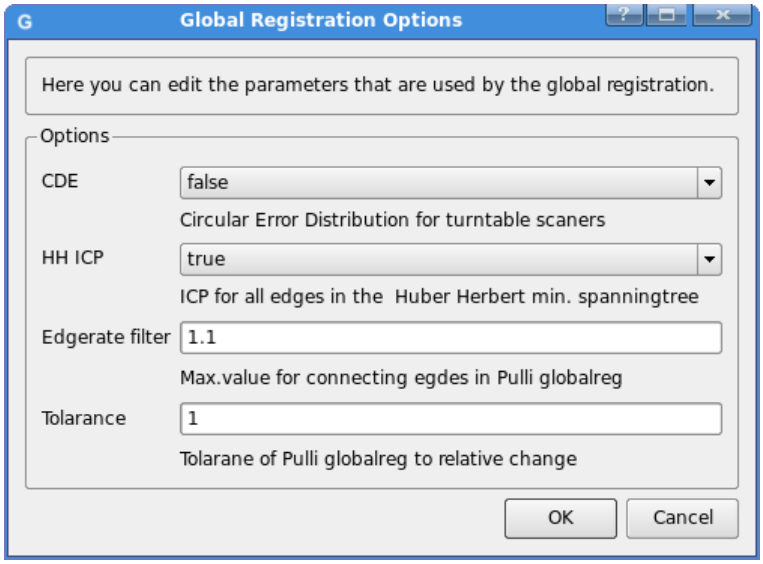

Abbildung A.9: Registrierungswerkzeug: Screenshot des Dialogs für Globalregistrierungsoptionen

  - **cellcount z** – Dieser Wert gibt die Anzahl der Zellen in z-Richtung an, falls das 3D-Verfahren verwendet wird.
  - **iterations** – Die Anzahl der Schritte des Newton-Verfahrens kann mit diesem Parameter beschränkt werden.
  - **delta translation** – Bestimmt die Schrittweite der partiellen Ableitung nach den Translationsvariablen X, Y und Z.
  - **delta rotation** – Bestimmt die Schrittweite der partiellen Ableitung nach den Rotationsvariablen φ und θ.
- Verfahren nach Roth
  - **quantize factor** – Dieser Wert bestimmt den Quantisierungsfaktor.
  - **triangulation vertices** – Mit diesem Parameter definiert man die Anzahl der Triangulationsknoten.

Feinregistrierung

Der Parameter der Feinregistrierung kann ebenfalls unter *Optionen* im Hauptmenü eingestellt werden.

- **Ratio** – Gibt an, wie groß der Anteil an Punkten ist, der für die Feinregistrierung benutzt wird. Der Wert ist eine Zahl zwischen null und eins.

Globalregistrierung

Im *Global Registration Options*-Dialog (Abb. A.9) können die Einstellungen für die Globalregistrierung geändert werden.

- **CDE** – Gibt an, ob bei der Globalregistrierung nach Huber/Hebert versucht werden soll den addierten Winkelfehler zu minimieren (*Circular Error Distribution*). Dies bietet sich nur bei Drehtellerscannern an.
- **HH ICP** – Gibt an, ob bei der Globalregistrierung nach Huber/Hebert für alle verwendeten Paarungen (noch einmal) eine Feinregistrierung durchgeführt werden soll.

- **Edgerate filter** – Verwendet nur Paarungen für die Globalregistrierung nach Pulli, deren Bewertung besser ist als der angegebene Wert.
- **Tolerance** – Toleranz der Globalregistrierung nach Pulli.

Anzeige

- **Draw Trackball** – Soll der Kreis angezeigt werden, der die Grenzen des Trackballs markiert?

A.5 Server

Der Server ist ein Konsolenprogramm ohne grafische Benutzeroberfläche, das die mit dem Registrierungswerkzeug erzeugten Modelle bei Anfrage über eine TCP/IP-Verbindung überträgt.

Da diese Übertragung nur durch den anfordernden Client angestoßen wird, ist eine Interaktion zwischen Benutzer und Server weder möglich noch nötig. Nachdem der Server gestartet wurde, werden alle Aufgaben automatisch erledigt.

Damit der Server jedoch wie gewünscht funktioniert, muss er vorher entsprechend konfiguriert werden. Dazu gehört zuerst einmal das Anlegen einer Verzeichnisstruktur, so dass der Server die zu übertragenden Modelle findet. Außerdem muss eine Datei mit der Benutzerverwaltung erstellt werden, in der festgelegt wird, wer sich mit dem Server verbinden darf.

A.5.1 Verzeichnisstruktur

Die Modelle, die der Server zur Verfügung stellen soll, müssen alle in einem Verzeichnis `models` gespeichert werden. Darin muss für jedes Modell ein Ordner mit dem Namen erzeugt werden, der später im Client angezeigt werden soll. In diesen Ordner muss die vom Registrierungswerkzeug erzeugte Datei unter dem Namen `daten.bin` gespeichert werden.

Wenn gewünscht, kann zusätzlich zum Modell auch eine Beschreibung übertragen werden, die vom Client in einem kleinen Informationsfenster angezeigt wird. Diese Beschreibung muss im HTML-Format vorliegen. Es ist zu beachten, dass nur eine kleine Auswahl an HTML-Befehlen im Informationsfenster interpretiert werden kann. Alle Verweise in den HTML-Dateien müssen die relative Adresse `cache/<dein Modellname>/` vorangestellt haben. Zusätzlich zu der Modelldatei `daten.bin` müssen alle für die Beschreibung benötigten Dateien (z. B. ein Vorschaubild) ebenfalls in das Verzeichnis des entsprechenden Modells kopiert werden. Die HTML-Seite, die als Startseite angezeigt werden soll, muss dabei `index.html` heißen.

Um dem Server mitzuteilen, welche Dateien zu den Zusatzinformationen gehören, muss schließlich eine Datei `daten.files` erstellt werden, in der alle Dateien aufgelistet sind.

Das folgende Beispiel zeigt den Verzeichnisaufbau für zwei Modelle, die auf dem Server bereitgestellt werden sollen: Das Modell „Siegelabdruck“ beinhaltet Zusatzinformationen und ein Vorschaubild, das Modell „Lagerverwaltung“ enthält keine Beschreibung. Die Datei `userlist.xml` ist für die Benutzerverwaltung erforderlich und wird im Anhang A.5.2 beschrieben.

Beispiel für die Verzeichnisstruktur:

```
userlist.xml
models/Siegelabdruck/daten.bin
                     /daten.files
                     /index.html
                     /more.html
                     /preview.jpg
      /Lagerverwaltung/daten.bin
```

Inhalt der Datei `daten.files`:

```
index.html
more.html
preview.jpg
```

Möglicher Inhalt der Datei `index.html`:

```
<html><head></head><body>
  <h1>Siegelabdruck</h1>
  <p>Hier k&ouml;nnte dein Text stehen...
     (<a href="cache/Siegelabdruck/more.html">mehr</a>)</p>
  <img src="cache/Siegelabdruck/preview.jpg">
</body></html>
```

A.5.2 Benutzerverwaltung

Um den Benutzerkreis des Servers zu beschränken, wird in einer Benutzerliste bestimmt, wer sich mit dem Server verbinden kann. Jeder Client, der eine Verbindung mit der Server aufbauen möchte, muss einen Benutzernamen und ein Passwort übertragen, die dann mit der auf dem Server gespeicherten Benutzerliste verglichen werden. Diese Benutzerliste wird in der Datei `userlist.xml` im XML-Format gespeichert.

Der Aufbau dieser XML-Datei kann am besten durch die Dokumenttypdefinition der Benutzerliste beschrieben werden.

```
<!ELEMENT userlist (user*)>

<!ELEMENT user EMPTY>
<!ATTLIST user
      pass    CDATA    #REQUIRED
      div     CDATA    #REQUIRED
      name    CDATA    #REQUIRED
>
```

Die Attribute `pass` und `user` beinhalten dabei Passwort bzw. Benutzername. `div` wurde vorsorglich eingefügt, um sp“ater die Zugriffskontrolle erweitern zu können. Zum aktuellen Stand wird dieser Parameter jedoch nicht berücksichtigt.

Ein Beispiel für eine Benutzerliste mit zwei registrierten Benutzern könnte folgendermaßen aussehen:

```
<?xml version="1.0"?>
<!DOCTYPE userlist SYSTEM "https://graphaeology.net/downloads/userlist.dtd">

<userlist>
    <user pass="secret"  div="" name="Mr. Stony" />
    <user pass="private" div="" name="Old Mole"  />
</userlist>
```

Hinweis: Da die Passwörter unverschlüsselt in der Datei gespeichert sind, muss darauf geachtet werden, dass durch eine entsprechende Rechteverwaltung nur autorisierte Benutzer die Datei ändern und auslesen dürfen.

A.5.3 Bedienung

Um den Server zu starten, muss in der Konsole `./server` aufgerufen werden. Dadurch verbindet sich der Server automatisch mit dem Port 44044 und wartet auf Anfragen eines Clients. Es ist jedoch darauf zu achten, dass das richtige Arbeitsverzeichnis verwendet wird. Der Server muss also in dem Verzeichnis gestartet werden, in dem der Ordner `models` und die `userlist.xml` liegen.

Es gibt verschiedene Parameter, die dem Server beim Start übergeben werden können:

`-u:` Durch diesen Parameter wird beim Start des Programms die Liste der auf dem Server registrierten Benutzer in der Konsole ausgegeben.

`-m:` Beim Start des Servers werden die Namen der vorhandenen Modelle in der Konsole ausgegeben.

`-p:` Die auf den Parameter `-p` folgende Portnummer bestimmt, mit welchem Port sich der Server verbinden soll.

`-d:` Debugausgaben werden in die Konsole geschrieben, wodurch unter anderem die Aktivität des Servers verfolgt werden kann.

`--help:` Zeigt die möglichen Parameter des Servers an.

An der Ausgabe in der Konsole erkennt man, ob der Start des Servers erfolgreich war. Das nächste Beispiel zeigt den Start des Servers am Port 1234 mit Ausgabe der Modellliste:

```
du@pc:~/graphaeology.net/bin> ./server -p 1234 -m
Modellist:
  Lagerverwaltung
  Siegelabdruck
Try listening on port 1234... success! running...
```

A.6 Client

Das Client-Modul ist das Programm, mit dem man sich die Modelle ansehen kann, die von einem Server angeboten werden. Dafür wird eine Verbindung zum Server aufgebaut. Dann werden, abhängig von der aktuellen Position im Betrachtungsfenster, die Daten eines Modells progressiv übertragen. Damit die einmal angeforderten Daten beim nochmaligen Betrachten nicht erneut vom Server geladen werden müssen, besitzt der Client einen Cache, in dem die bereits geladenen Teile der Modelle lokal gespeichert werden.

Bevor die Bedienung des Betrachtungsfensters beschrieben wird, wird im Folgenden zuerst die Erstellung einer Verbindung zu einem Server und die Auswahl eines Modells erklärt.

A.6.1 Verbindungen

Damit die Daten eines Modells auf den Client übertragen werden können, muss eine Verbindung mit einem Server hergestellt werden. Andernfalls können nur die Modelle angezeigt werden, die bereits im Cache des Clients gespeichert sind.

Durch Aufruf von `Program > Connect...` im Hauptmenü wird dder Verbindungsdialog geöffnet (siehe Abb. A.10). Hier werden in einer Liste die bereits erzeugten Verbindungskonfigurationen angezeigt. Über die Schaltflächen `New`, `Edit` und `Delete` kann eine neue Verbindungskonfiguration erstellt bzw. vorhandene beearbeitet oder gelöscht werden. Durch Anklicken von `Save & Connect` werden alle im Dialogfenster durchgeführten Änderungen gespeichert und versucht, eine Verbindung zu dem ausgewählten Server aufzubauen.

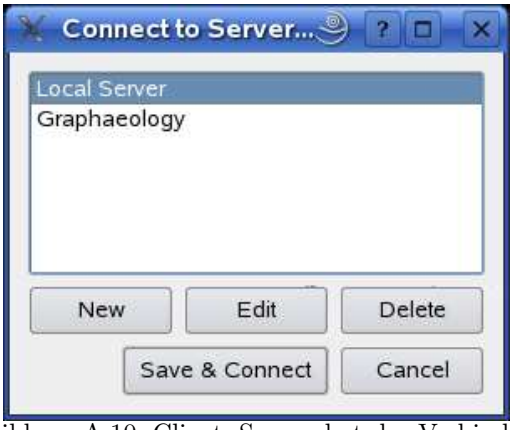

Abbildung A.10: Client: Screenshot des Verbindungsdialogs

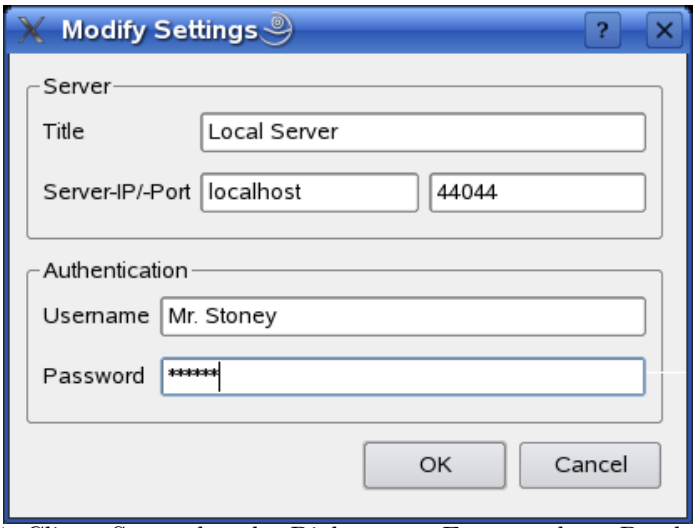

Abbildung A.11: Client: Screenshot des Dialogs zum Erzeugen bzw. Bearbeiten einer Verbindung

Neue Verbindungskonfiguration herstellen

Durch Auswahl von `New` im Verbindungsdialog wird ein neues Fenster geöffnet, in dem die Eigenschaften einer Verbindung eingetragen werden können (siehe Abb. A.11). `Title` ist dabei der frei wählbare Name, unter dem die aktuelle Verbindungskonfiguration im Verbindungsdialog angezeigt wird. Bei `Server-IP/-Port` sind die IP-Adresse und die Portnummer des Servers einzutragen.

Da der Server nur Verbindungen zu registrierten Benutzern zulässt, müssen in den Feldern `Username` und `Password` die beim Server hinterlegten Benutzerinformationen eingetragen werden.

A.6.2 Modellmenü

Durch Aufruf von `Model > Select...` im Hauptmenü des Clients wird der Modelldialog (siehe Abb. A.12) aufgerufen. Hier kann ein Modell ausgewählt werden, das daraufhin im Betrachtungsfenster dargestellt wird.

In dem Modelldialog wird zwischen drei Arten von Modellen unterschieden:

1. Modelle, die nur auf dem Server vorhanden und noch nicht auf dem Client gespeichert sind. Man

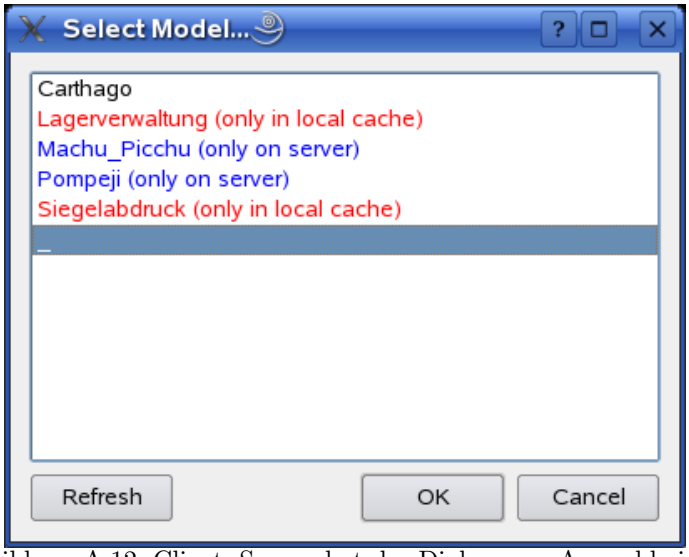

Abbildung A.12: Client: Screenshot des Dialogs zur Auswahl eines Modells

erkennt sie an der Bezeichnung `(only on server)`, die an den Modellnamen angehängt wird.

2. Modelle, die nicht auf dem Server vorhanden, sondern nur auf dem Client gespeichert sind. Man erkennt sie an der Bezeichnung `(only in local cache)`, die an den Modellnamen angehängt wird.

3. Modelle, die sowohl auf dem Server als auch auf dem Client vorhanden sind.

Falls keine Verbindung zu einem Server aufgebaut wurde, werden natürlich nur die Modelle, die auf dem Client gespeichert sind, in der Modelliste angezeigt.

Über den Menüpunkt `Model > Clear cache` besteht die Möglichkeit, die Daten eines Modells komplett vom Client zu entfernen und so zum Beispiel neuen Speicherplatz zu schaffen.

A.6.3 Hauptfenster

Das Hauptfenster (siehe Abb. A.13) besteht aus mehreren Teilen. Der große linke Bereich wird vom Betrachtungsfenster eingenommen, in dem das Modell angezeigt wird. Auf der rechten Seite werden zwei weitere Fenster dargestellt: Oben ist ein Vorschaufenster, in dem eine Übersicht über das komplette Modell gegeben wird. Darunter werden zusätzliche Informationen zum Objekt als HTML-Daten angezeigt, falls sie vom Server zur Verfügung gestellt werden. Alternativ kann dort durch Auswahl des entsprechenden Tabulators zu den Optionen des Betrachtungsfensters oder zu einem Messwerkzeug gewechselt werden.

Bedienung des Betrachtungsfensters

In dem Betrachtungsfenster wird das vom Server übertragene Modell dargestellt. Um sich das Modell von allen Seiten ansehen zu können, kann der Benutzer durch Mausgesten den Blickwinkel ändern. Die standardmäßig eingestellte Steuerung heißt *Trackball-Navigation*. Als Alternative wird zusätzlich die *XSI-Navigation* angeboten. Weitere Funktionen in dem Betrachtungsfenster sind das Anzeigen einer Schnittebene durch das Modell und die Verschiebung der virtuellen Lichtquellen. Die Befehle zur Bedienung sind im Folgenden erklärt:

Trackball-Navigation

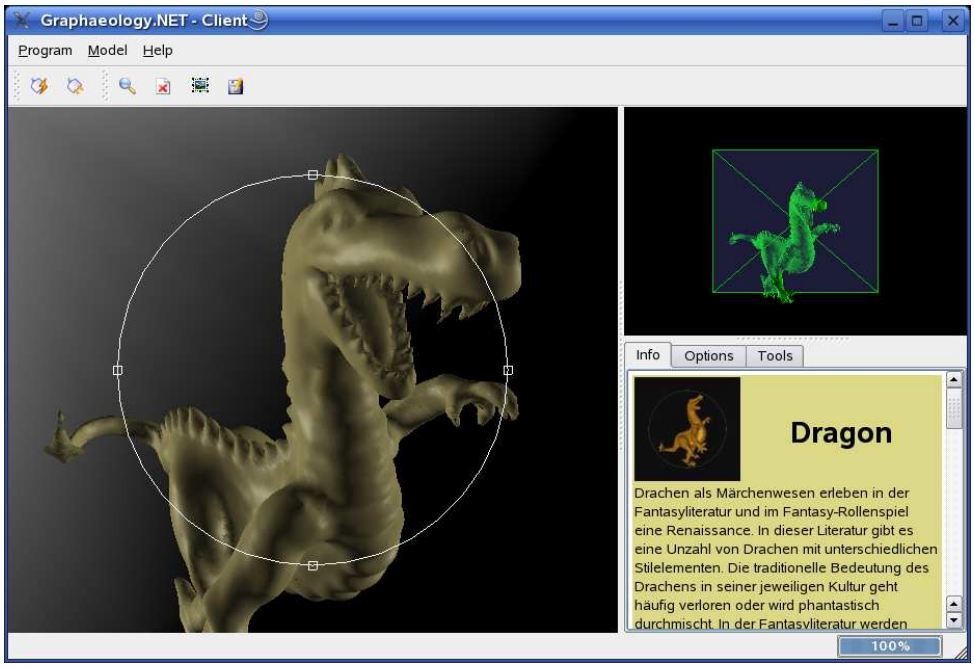

Abbildung A.13: Client: Screenshot des Hauptfensters

Verschieben Um das Modell zu verschieben, muss die rechte Maustaste gehalten und gleichzeitig die Maus verschoben werden. Das Modell bewegt sich dann entsprechend der Position des Mauszeigers.

Rotieren Eine Drehung des Modells erfolgt durch Halten der linken Maustaste. Bei Bewegungen des Mauszeigers innerhalb des weißen Kreises (des Trackballs), rotiert das Modell um die waagerechte und senkrechte Achse. Bei Bewegungen außerhalb des Trackballs erfolgt eine Rotation um die Achse senkrecht zum Bildschirm.

Zoomen Durch Drehen des Mausrads kann in das Modell hinein- bzw. aus ihm herausgezoomt werden. Gleichzeitiges Drücken der Strg-Taste erhöht die Zoomgeschwindigkeit.

XSI-Navigation

Verschieben Durch Halten der rechten Maustaste und gleichzeitiger Mausbewegung kann das Modell verschoben werden. Bei der XSI-Navigation erfolgt jedoch im Gegensatz zur Trackball-Navigation die Verschiebung entgegen der Mausbewegung.

Rotieren Die aktuelle Blickposition wird durch das Halten der linken Maustaste um einen festgelegten Drehpunkt rotiert. Dieser Drehpunkt wird durch einen blauen Fleck im Betrachtungsfenster dargestellt.

Rotationspunkt setzen Durch Drücken der mittleren Maustaste wird der Drehpunkt entsprechend der aktuellen Mauszeigerposition neu gesetzt.

Zoomen Durch Drehen des Mausrads kann in das Modell hinein- bzw. herausgezoomt werden. Gleichzeitiges Drücken der Strg-Taste erhöht die Zoomgeschwindigkeit.

Lichtquellen Um die Beleuchtung des Modells zu variieren, besteht zusätzlich die Möglichkeit, die Lage der virtuellen Lichtquellen zu verschieben. Durch gleichzeitiges Halten der Shift-Taste und der linken Maustaste kann die erste Lichtquelle um das Modell rotiert werden. Falls eine zweite Lichtquelle aktiviert ist, kann diese durch Drücken der Shift- und der rechten Maustaste bewegt werden.

Schnittebene Die Schnittebene, die der Benutzer durch das Modell legen kann, wird entsprechend der Trackball-Navigation und gleichzeitigem Drücken der Strg-Taste bewegt. Das gleichzeitige Halten der Strg-Taste und rechter Maustaste verschiebt, das Halten der Strg-Taste und linker Maustaste dreht die Ebene.

Optionen der Ansicht

Im Hauptfenster hat man unten rechts unter dem Tabulator `Options` die Möglichkeit, die Eigenschaften des Betrachtungsfensters einzustellen. In folgender Liste werden die einzelnen Optionen kurz erläutert:

`XSI-Navigation:` Durch Auswahl dieser Option kann die Steuerung im Betrachtungsfenster vom Trackball auf die von XSI bekannte Navigation gewechselt werden.

`Enable Headlight:` Diese Option setzt die Position einer Lichtquelle auf die jeweils aktuelle Blickposition.

`Enable secondary light:` Mit diesem Parameter kann eine zweite Lichtquelle angeschaltet werden.

`Enable resolution coloring:` Die Aktivierung dieser Option färbt die Teilflächen des Modells in verschiedenen Farben ein, wodurch erkannt werden kann, in welcher Auflösung die einzelnen Teilflächen dargestellt werden. Rote Färbung bedeutet dabei, dass die gröbste Darstellung verwendet wird, ein helles Grün entspricht der feinsten Darstellung.

`Draw mesh:` Wenn diese Option deaktiviert ist, wird das Modell nicht mehr angezeigt. Dies ist jedoch nur sinnvoll in Verbindung mit der nächsten Option.

`Show sectionplane:` Hiermit kann eine Schnittebene in das Betrachtungsfenster eingefügt werden, auf der der Schnitt des Modells mit der Schnittebene markiert ist.

`Flat shading:` Durch Selektion dieser Option wird ein einfacheres Schattierungsverfahren verwendet, so dass die Übergänge der Teilflächen des Modells nicht mehr geglättet werden.

`Display trackball theme:` Gibt an, ob der Kreis des Trackballs dargestellt werden soll.

`Fast display:` Wenn diese Option aktiviert ist, wird beim Verschieben nur das gröbste Netz angezeigt, um die Geschwindigkeit zu erhöhen.

Messwerkzeug

Mit dem Messwerkzeug können Wege auf der Oberfläche abgelaufen werden, um Entfernungen zu messen. Das Werkzeug wird im Tabulator `Tools` unten rechts im Hauptfenster aktiviert (siehe Abb. A.14).

Um nun eine Entfernung zu messen, muss im Betrachtungsfenster die Strg-Taste gehalten werden und dann durch Drücken der linken Maustaste die Punkte auf dem Streckenzug ausgewählt werden. Der Abstand wird im Messfenster dargestellt. Um die Größe der Eckpunktdarstellung des Streckenzugs anzupassen, kann der Schieberegler verwendet werden.

Wenn ein neuer Streckenzug abgemessen werden soll, kann durch Klicken von `Reset` der aktuelle Streckenzug gelöscht werden und die Abstandsmessung zurückgesetzt werden.

A.6.4 Überblick über alle Menüfunktionen

`Program > Connect...:` Öffnet den Verbindungsdialog zum Verwalten der Verbindungskonfigurationen.

- `New:` Öffnet den Dialog zum Erzeugen einer neuen Verbindungskonfiguration.
  - `OK:` Erzeugt eine neue Verbindungskonfiguration mit den eingegebenen Werten.
  - `Cancel:` Schließt den Dialog ohne Speicherung.
- `Edit:` Öffnet den Dialog zum Bearbeiten der ausgewählten Verbindungskonfiguration.

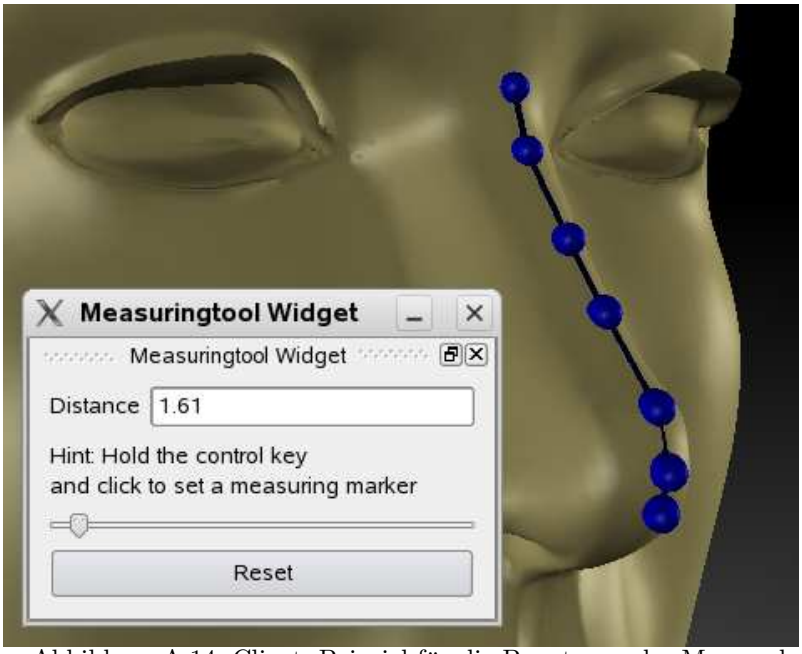

Abbildung A.14: Client: Beispiel für die Benutzung des Messwerkzeugs

`OK:` Speichert die ausgewählte Verbindungskonfiguration mit den geänderten Werten.

`Cancel:` Schließt den Dialog ohne Speicherung.

`Delete:` Löscht die ausgewählten Verbindungskonfigurationen aus der Liste.

`Save & Connect:` Speichert alle Änderungen in der Liste der Verbindungskonfigurationen und versucht, eine Verbindung mit der ausgewählten Verbindungskonfiguration aufzubauen.

`Cancel:` Schließt den Dialog ohne Speicherung.

`Program > Disconnect:` Trennt die aktuelle Verbindung.

`Program > Exit:` Beendet das Programm.

`Model > Select modell...:` Öffnet den Modelldialog.

`Refresh:` Lädt die Liste der Modelle neu.

`OK:` Schließt den Dialog und öffnet das ausgewählte Modell im Betrachtungsfenster.

`Cancel:` Schließt den Dialog ohne Änderungen.

`Model > Clear cache:` Löscht alle lokalen Daten zum aktuellen Modell.

`Tools > Screenshot:` Erzeugt ein JPEG-Bild von der aktuellen Ansicht im Betrachtungsfenster.

`Tools > OFF-Export:` Erzeugt eine OFF-Datei von dem aktuellen Modell.

`Help > About:` Öffnet das Informationsfenster.

A.7 Schnittstellen zwischen den Modulen

Es gibt insgesamt drei Schnittstellen:

Eingang des Registrierungswerkzeugs: Die Scans, die in das Registrierungswerkzeug geladen werden, liegen im OFF-Dateiformat vor, das in Anhang B.2 beschrieben ist.

Schnittstelle zwischen Registrierungswerkzeug und Server: Die vom Registrierungswerkzeug erstellten Objekte, die vom Server bereitgestellt werden sollen, werden in einem von der Projektgruppe entwickelten Dateiformat gespeichert (siehe Anhang B.1).

Schnittstelle zwischen Server und Client: Die Objekte, die im Client dargestellt werden sollen, werden über das TCP/IP-Protokoll übertragen. Eine Spezifikation des Protokolls wird in Anhang 5.3 kurz skizziert.

Anhang B

Dateiformate

B.1 Binärformat der Server-Dateien

Da die Reduktion im Registrierungs-Modul durchgeführt wird, muss eine Verbindung zwischen dem Registrierungs- und dem Client-Server-Modul hergestellt werden. Dies geschieht über eine Datei, die durch die Reduktion erzeugt und vom Server geladen wird.

Die gewonnenen Daten der Reduktion des Registrierungs-Moduls werden als binärer Datenstrom in einer Datei abgespeichert. Es wurde explizit ein binäres Dateiformat gewählt um eine gewisse Platzersparnis, Sicherheit und Effektivität bei Zugriffen zu gewährleisten. Der folgende Abschnitt beschreibt die Spezifikation der Datei, die von der Reduktion des Registrierungs-Moduls erzeugt und vom Server eingelesen wird.

Für die Zuordnung der Elemente zueinander werden mit einer Ausnahme Indizes benutzt. Diese Indizes bestehen aus einer 4-Bytes Ganzzahl. Nach den Kopf-Informationen kann eine beliebige Anzahl an Bytes (`RESERVED`), die für zusätzliche Informationen, wie z. B. für die HTML-Informationen genutzt werden können, folgen. Diese können auch zu einem späteren Zeitpunkt eingefügt werden und werden bei der ersten Implementierung nicht beachtet. Im Folgenden werden nun die einzelnen Segmente der Datei nacheinander beschrieben.

Kopf-Informationen

Die Kopf-Information (engl. *Header*) der Datei besteht aus fünf *Offsets* und einer 4-Bytes Fließkommazahl. Der Datentyp der *Offsets* ist eine 4-Bytes Ganzzahl.

`BN_OFFSET` Ein *Offset*, der die Stelle in der Datei angibt, an der die Definition des Basis-Netzes beginnt.

`TL_OFFSET` Ein *Offset*, der die Stelle in der Datei angibt, an der die Definition der Dreiecke beginnt.

`NO_OFFSET` Ein *Offset*, der die Stelle in der Datei angibt, an der die Definition der Knotenliste startet.

`RT_OFFSET` Ein *Offset*, der die Stelle in der Datei angibt, an der die Definition der Verfeinerungsoperationen startet.

`NL_OFFSET` Ein *Offset*, der die Stelle in der Datei angibt, an der die Definition der Nachbardreiecke startet.

`ASPECT` Ein Fließkommawert, der die Auflösung des Scanners angibt.

`RESERVED` Beliebige Anzahl an Bytes für zusätzliche Informationen.

Daten Basisnetz

Das Basisnetz wird durch eine beliebige Anzahl von Dreiecken beschrieben.

Daten Dreiecksliste

Die Dreiecksliste ist eine beliebige Folge von Dreiecken. Ein Dreieck ist eine Folge von genau drei Knotenindizes.

Daten Knotenliste

Die Knotenliste ist eine beliebige Folge von Knoten. Ein Knoten ist eine Folge von genau drei 4-Bytes Fließkommazahlen. Die erste Fließkommazahl ist die x-Koordinate, die zweite Fließkommazahl ist die y-Koordinate und die dritte Fließkommazahl ist die z-Koordinate.

Daten Verfeinerungsoperationen

Die Verfeinerungsoperationen bestehen aus Verfeinerungsknoten. Ein Verfeinerungsknoten enthält dabei zwei Indizes der beiden zuvor reduzierten Knoten und zwei Indizes der beiden zuvor reduzierten Dreiecke. Wenn keine Knoten und keine Dreiecke mehr angehängt werden, wird als Index -1 eingetragen. Darüber hinaus werden noch Informationen, wo die Indizes der benachbarten Dreiecke relativ zum `NL_OFFSET` in der Datei stehen und wie viele es sind, angehängt. Dies geschieht durch einen *Offset* `N_OFFSET`. Der *Offset* speichert in einer 4-Bytes Ganzzahl die Stelle in Bytes, wo die Liste der zu diesem Verfeinerungsknoten gehörenden Nachbardreiecke beginnt. Ebenfalls wird die Anzahl der Nachbardreiecke des Verfeinerungsknotens in einer 4-Bytes Ganzzahl `N_COUNT` gespeichert.

Dabei ist exakt darauf zu achten, dass die Informationen zu den Verfeinerungen jedes Knotens in der selben Reihenfolge wie die Knoten auftauchen, so dass über den Knotenindex n der zugehörige n-te Verfeinerungsknoten der Verfeinerungsoperationen ausgelesen werden kann.

Daten Nachbarliste

Die Nachbarliste besteht aus einer beliebigen Anzahl Dreiecksindizes. Diese Indizes beinhalten die Position der Nachbardreiecke aller Knoten in der Datei.

B.2 Object File Format (OFF)

Die Schnittstelle zwischen dem Registrierungs- und dem Reduktionswerkzeug bildet das „Object File Format“ (OFF) [30]. Der Vorteil dieses Formates ist, dass es recht einfach aufgebaut ist und sich relativ effizient maschinell verarbeiten lässt. Weiterhin ist es bereits etabliert, und es existieren verschiedene gute Programme, um Dateien im OFF zu visualisieren. Das OFF repräsentiert die Geometrie eines Modells, indem es die Polygone der Modelloberfläche spezifiziert. Jede OFF-Datei beginnt mit dem Schlüsselwort OFF, gefolgt von der Anzahl der Knoten, der Polygone und der Kanten. Danach werden die Koordinaten aller Knoten aufgelistet, wobei jede Zeile den Koordinaten eines Knotens entspricht. Im nächsten Abschnitt werden die Polygone über ihre Knoten definiert. Jede Zeile entspricht einem Polygon. Die erste Zahl gibt die Anzahl der Eckpunkte des Polygons an, danach folgen entsprechend viele Knotenindizes. Zu beachten ist, dass das in der Projektgruppe zu übergebende Modell ein Dreiecksnetz darstellt und auf drei Dimensionen beschränkt ist. Abbildung B.2 zeigt den beispielhaften Aufbau einer OFF-Datei für einen Würfel.

B.3 Projektdatei des Registrierungswerkzeugs

Das Registrierungstool erlaubt es, Projekte zu speichern und zu laden. Ein Projekt wird in einer auf XML basierenden Dateii mit der Endung `.489` gespeichert. Der zugehörige DTD ist in Abbildung B.3 aufgelistet.

```
OFF
8 6 0
-0.5 -0.5 0.5
0.5 -0.5 0.5
-0.5 0.5 0.5
0.5 0.5 0.5
-0.5 0.5 -0.5
0.5 0.5 -0.5
-0.5 -0.5 -0.5
0.5 -0.5 -0.5
4 0 1 3 2
4 2 3 5 4
4 4 5 7 6
4 6 7 1 0
4 1 7 5 3
4 6 0 2 4
```

Abbildung B.1: Dateiformat OFF: Beispiel für eine OFF-Datei, beginnend mit dem Schlüsselwort OFF. Das Objekt besteht aus acht Knoten, sechs Polygonen und null Kanten. Danach folgt ein Block mit der Koordinatenbeschreibung der Knoten, abschließend die Definition der Polygonzüge. Ein Polygon besteht aus vier Knoten, die aus den nachfolgenden Indizes zusammengesetzt werden. Die Indizierung ist entsprechend der Position der Knoten im Knotenblock der OFF Datei zu sehen.

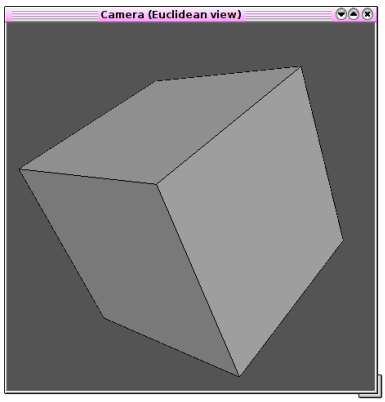

Abbildung B.2: Dateiformat OFF: Darstellung der Beispieldatei in Geomview [14].

Das Wurzelelement *regtoolProject* enthält ein *info*-Element und beliebig viele *patch*- und *edge*-Elemente. Im *info*-Element werden der Name des Projekts und der Pfad zum Arbeitsverzeichnis gespeichert. Ein *patch*-Element repräsentiert einen Scan im Projekt und speichert den Pfad zur zugehörigen OFF-Datei und die fortlaufende Nummer im Projekt. Die *edge*-Elemente stellen die Kanten im *Modelgraph* zwischen zwei Scans dar. Sie enthalten die Nummern der beiden Scans, sowie die Transformation zwischen den beiden und den Fehlerwert der Transformation.

Abbildung B.3: Inhalt der Datei regtoolProject.dtd

```
<!ELEMENT regtoolProject (info, patch*, edge*)>

<!ELEMENT info   (projectName,workingDir,)>
<!ELEMENT projectName    (#CDATA)>
<!ELEMENT workingDir     (#CDATA)>



<!ELEMENT patch (filename, filepath)EMPTY>
<!ATTLIST patch
        number              PCDATA      #REQUIRED
>

<!ELEMENT filename       (#CDATA)>
<!ELEMENT filepath       (#CDATA)>


<!ELEMENT edge (transformation)>
<!ATTLIST edge
        patch1number        PCDATA       #REQUIRED
        patch2number        PCDATA       #REQUIRED
        error               CDATA       #REQUIRED
        group               CDATA       #REQUIRED
>

<!ELEMENT transformation>
<!ATTLIST transformation
        m00                 CDATA       #REQUIRED
        m01                 CDATA       #REQUIRED
        m02                 CDATA       #REQUIRED
        m03                 CDATA       #REQUIRED
        m10                 CDATA       #REQUIRED
        m11                 CDATA       #REQUIRED
        m12                 CDATA       #REQUIRED
        m13                 CDATA       #REQUIRED
        m20                 CDATA       #REQUIRED
        m21                 CDATA       #REQUIRED
        m22                 CDATA       #REQUIRED
        m23                 CDATA       #REQUIRED
        m30                 CDATA       #REQUIRED
        m31                 CDATA       #REQUIRED
        m32                 CDATA       #REQUIRED
        m33                 CDATA       #REQUIRED
>
```

Listing B.1: regtoolProject.dtd

Anhang C

Bibliotheken

C.1 UMFPACK, AMD und GotoBLAS

Die UMFPACK-Bibliothek liefert Funktionen zum Lösen von linearen Gleichungssystemen. Diese Bibliothek wird vom Parametrisierer (siehe Kapitel 2.1) benötigt, da dieser sehr große Gleichungssysteme berechnen muss. Da Matrizen dieser Gleichungssysteme nur sehr schwach besetzt sind, fiel die Wahl auf die UMFPACK-Bibliothek, da sie auf diesen Fall spezialisiert ist. UMFPACK liefert dabei einen Satz von Funktionen zum Lösen von unsymmetrischen, schwach besetzten Gleichungssystemen der Form $\boldsymbol{A}\vec{x} = \vec{b}$. Es nutzt hierzu die unsymmetrische Multifrontal-Methode zusammen mit LU-Faktorisierung in speziellen Varianten für den schwach besetzten Fall. Um gute Leistungen zu erbringen, setzt UMFPACK auf BLAS bzw. GotoBLAS auf. BLAS steht dabei für *Basic Linear Algebra Subroutines*. Bei GotoBLAS handelt es sich nach Angaben des Autors um die im Moment schnellste Implementierung von BLAS. Bei der Übersetzung wird die Bibliothek auf die verwendete Rechnerarchitektur hin optimiert, um die Anzahl der Treffer im Prozessorcache möglichst hoch zu halten. Dadurch ist es möglich auch diese sehr großen Gleichungssysteme in Echtzeit zu lösen. Weiterhin benutzt UMFPACK die AMD-Bibliothek, diese liefert Routinen zum Umordnen spärlich besetzter Matrizen vor der LU-Faktorisierung.

C.2 ANN

ANN ist eine Bibliothek, die komplett in C++ geschrieben ist. Sie unterstützt die exakte und approximierte Suche von nächsten Nachbarn in verschiedenen Dimensionen. Die Autoren sind David M. Mount von der *University of Maryland* und Sunil Arya von der *Hong Kong University of Science and Technology*. ANN steht für *Approximate Nearest Neighbor* und bietet u.a. auch eine Testumgebung, um Testdaten zu erzeugen, zu sammeln und zu analysieren.

Bei dem Nächster-Nachbar-Problem ist eine Punktmenge P in einem d-dimensionalen Raum gegeben. Für diese Menge wird eine passende Datenstruktur erzeugt, um für einen Punkt $\vec{q}$ effizient die nächsten k Nachbarn zu bestimmen. ANN ist für Datenmengen ausgelegt, die klein genug sind, so dass sie in den Arbeitsspeicher passen.

Die ANN-Bibliothek unterstützt für einen Punkt q im d-dimensionalen Raum die Suche der nächsten Nachbarn in einem k-d-Baum.

Die Ausführung der Methode *annkSearch* für einen Punkt q, gibt die Indizes der nächsten k Nachbarpunkte zurück. Das Array nn_idx beinhaltet die Indizes der nächsten Nachbarpunkte, und das Array $dists$, die entsprechende Distanz zu den nächsten Nachbarpunkten. Dabei müssen die Arrays nn_idx und $dists$ ebenfalls mindestens k Elemente enthalten. Man erhält Zugriff auf den nächsten Nachbarn durch $nn_idx[0]$, $nn_idx[0]$, usw. . Um den Schwellwert etwas zu approximieren, kann man optional $eps > 0$ wählen. Damit würde der „i“. Nachbar eine Approximation des „i + eps“. Nachbarn sein.

```
virtual void ANNkd_tree::annkSearch(
ANNpoint q,  // für den Punkt q wird die Suche ausgeführt
int k,  // Angabe, wie viele nächsten Nachbarn gesucht sind
ANNidxArray nn_idx, // Array mit den Indizes der nächsten Nachbarpunkte
ANNdistArray dists, // Array mit der Distanzwerten zu den nächsten Nachbarpunkten
double eps=0.0);  // optionaler Schwellwert
```

Abbildung C.1: Methode annkSearch der ANN-Bibliothek mit den Parameterübergaben.

Im Graphaeology.NET-Projekt wird ANN in der Version 1.1.1 eingesetzt. Sollten die Autoren dieser Bibliothek in neueren Versionen keine Änderungen an der Schnittstelle vornehmen, sollte es keine Probleme geben, diese mit der Registrierungssoftware einzusetzen. Der Quellcode von ANN ist über folgende Internetseite zu beziehen:

http://www.cs.umd.edu/~mount/ANN

Dort findet man ebenfalls ein ausführliches Handbuch mit Installations- und Kompilierungsanweisungen. Zu beachten sind weiterhin die Nutzungsbedingungen, die für diese Software gelten.

C.3 TetGen

Tetgen ist ein Programm bzw. eine Bibliothek, das u.a. aus einer dreidimensionalen Punktwolke eine Delaunay-Tetrahedralisation und eine konvexe Hülle erzeugt. Der Quellcode wurde komplett in C++ geschrieben. Dieser kann zu einer ausführbaren Datei oder einer Bibliothek kompiliert werden. Hierbei werden alle gängigen Betriebssysteme unterstützt.

Im Graphaeology.NET-Projekt wird Tetgen in der Version 1.4.1 eingesetzt. Sollte der Autor dieser Bibliothek in neueren Versionen keine Änderungen an der Schnittstelle vornehmen, sollte es keine Probleme geben, diese mit der Registrierungssoftware einzusetzen. Der Quellcode von TetGen ist über folgende Internetseite zu bekommen:

http://tetgen.berlios.de

Dort findet man ebenfalls ein ausführliches Handbuch mit Installations- und Kompilierungsanweisungen. Zu beachten sind weiterhin die Nutzungsbedingungen, die für diese Software gelten.

C.4 FFTW

FFTW [11] [10] (Fastest Fourier Transform in the West) ist eine frei verfügbare Bibliothek für nichtkommerzielle Anwender unter der GPL-Lizenz, die eine schnelle Berechnung der diskreten Fouriertransformation ermöglicht. Diese Bibliothek kann sowohl für eindimensionale als auch mehrdimensionale diskrete Fouriertransformationen eingesetzt werden. Die Berechnungsmöglichkeiten schließen komplexe, reale, parallele und symmetrische Transformationen mit ein. FFTW ist in ANSI C geschrieben und ist somit auf verschiedenen Plattformen einsetzbar.

Anwendung findet diese Bibliothek im Graphaeology.NET-Projekt ausschließlich für die Grobregistrierung im Frequenzbereich (vgl. Kapitel 3.1.5). FFTW wird in der Version 3.1 eingesetzt.

Weitere Informationen bietet die Internetseite www.fftw.org, auf der das Handbuch und die Installationshinweise zu finden sind. Für das Graphaeology.NET-Projekt ist eine Installation durch den Benutzer nicht erforderlich.

Literaturverzeichnis

[1] View-Dependent Streaming of Progressive Meshes. In: *SMI '04: Proceedings of the Shape Modeling International 2004 (SMI'04)*. Washington, DC, USA : IEEE Computer Society, 2004, S. 209–220

[2] Bergevin, R. ; Soucy, M. ; Gagnon, H. ; Laurendeau, D.: Towards a General Multi-View Registration Technique. In: *IEEE Trans. Pattern Anal. Mach. Intell* 18 (1996), Nr. 5, S. 540–547

[3] Besl, P. J. ; McKay, N. D.: A Method for Registration of 3-D Shapes. In: *IEEE Transactions on Pattern Analysis and machine Intelligence* 14 (1992), Februar, Nr. 2, S. 239–258

[4] Biber, P. ; Strasser, W.: The Normal Distributions Transform: A New Approach to Laser Scan Matching. In: *IEEE/RJS International Conference on Intelligent Robots and Systems*, 2003

[5] De Floriani, L. ; Magillo, P.: Multiresolution mesh representation: Models and data structures. In: *Multiresolution in Geometric Modelling*. Springer, 2002, S. 363–418

[6] Floater, M. S. ; Hormann, K.: Parameterization of Triangulations and Unorganized Points. In: Iske, A. (Hrsg.) ; Quak, E. (Hrsg.) ; Floater, M. S. (Hrsg.): *Tutorials on Multiresolution in Geometric Modelling*. Berlin, Heidelberg : Springer, 2002 (Mathematics and Visualization), S. 287–316

[7] Floater, M. S. ; Hormann, K.: Surface Parameterization: a Tutorial and Survey. In: Dodgson, N. A. (Hrsg.) ; Floater, M. S. (Hrsg.) ; Sabin, M. A. (Hrsg.): *Advances in Multiresolution for Geometric Modelling*. Berlin, Heidelberg : Springer, 2005 (Mathematics and Visualization), S. 157–186

[8] Free Software Foundation: *GNU General Public License (GPL)*. Version: 2006. `http://www.fsf.org/licensing/licenses/gpl.html`

[9] Free Software Foundation: *Inoffizielle Übersetzung der GNU GPL*. Version: 2006. `http://www.gnu.de/documents/gpl.de.html`

[10] Frigo, M. ; Johnson, S. G.: The Design and Implementation of FFTW3. In: *Proceedings of the IEEE* 93 (2005), Nr. 2, Special issue on „Program Generation, Optimization, and Platform Adaptation“, S. 216–231

[11] Frigo, M. ; Johnson, S. G.: *FFTW-Version 3.1*. Version: 2006. `www.fftw.org`

[12] Fröhlich, M. ; Müller, H. ; Pillokat, C. ; Weller, F.: Feature-Based Matching of Triangular Meshes. In: *Geometric Modelling* Bd. 14, Springer, 1999 (Computing Supplement), S. 105–118

[13] Garimella, R. V. ; Swartz, B. K.: Curvature Estimation for Unstructured Triangulations of Surfaces / Los Alamos National Laboratory. 2003 (LA-UR-03-8240)

[14] Geomview: *Geomview*. `www.geomview.org`

[15] Gilbert, E. G. ; Johnson, D. W. ; Keerth, S. S.: A fast procedure for computing the distance between complex objects in three-dimensional space. In: *IEEE Journal of Robotics & Automation* RA-4 (1988), April, Nr. 2, S. 193–203

[16] GRAPHAEOLOGY.NET: *PG489 Seminarband der PG-Fahrt*. 2006 Lehrstuhl Informatik VII, Universität Dortmund

[17] HOPPE, H.: View-dependent refinement of progressive meshes. In: *SIGGRAPH*, 1997, S. 189–198

[18] HORN, B. K. P.: Closed-form solution of absolute orientation using unit quaternions. In: *Journal of the Optical Society of America A* 4 (1987), April, Nr. 4, S. 629–642

[19] HUBER, D. ; HEBERT, M.: Fully automatic registration of multiple 3D data sets. In: *Image and Vision Computing* 21 (2003), July, Nr. 7, S. 637–650

[20] JOHNSON, A. ; HEBERT, M.: Surface Registration by Matching Oriented Points. In: *International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, 1997, S. 121–128

[21] KIM, S. J. ; JEONG, W. K. ; KIM, C. H.: LOD Generation with Discrete Curvature Error Metric. In: *In Proceedings of Korea Israel Bi-National Conference*, 1999, S. 97–104

[22] KLEIN, R. ; LIEBICH, G. ; STRASSER, W.: Mesh reduction with error control. In: *VIS '96: Proceedings of the 7th conference on Visualization '96*. Los Alamitos, CA, USA : IEEE Computer Society Press, 1996, S. 311–318

[23] KLEIN, R. ; LIEBICH, G. ; STRASSER, W.: Mesh Reduction with Error Control. In: *IEEE Visualization '96*, 1996, S. 311–318

[24] LUCCHESE, L. ; DORETTO, G. ; CORTELAZZO, G. M.: A Frequency Domain Technique for Range Data Registration. In: *IEEE Trans. Pattern Anal. Mach. Intell* 24 (2002), Nr. 11, S. 1468–1484

[25] MÜLLER, H.: *Geometrisches Modellieren*. Vorlesungsfolien, 2005 Universität Dortmund, Deutschland

[26] NEUGEBAUER, P. J.: Geometrical cloning of 3D objects via simultaneous registration of multiple range images. In: *SMA '97: Proceedings of the 1997 International Conference on Shape Modeling and Applications (SMA '97)*. Washington, DC, USA : IEEE Computer Society, 1997, S. 130

[27] PLUIM, J. P. W. ; MAINTZ, J. B. A. ; VIERGEVER, M. A.: Mutual Information Based Registration of Medical Images: A Survey. In: *IEEE Trans. Med. Imaging* 22 (2003), Nr. 8, S. 986–1004

[28] PRIM, R. C.: Shortest Connection Networks and Some Generalizations. In: *Bell System Technical Journal* 36 (1957), S. 1389–1401

[29] PULLI, K.: Multiview Registration for Large Data Sets. In: *3dim* 00 (1999), S. 160

[30] ROST, R.: *OFF – A 3D Object File Format*. Digital Equipment Corporation, 100 Hamilton Ave. Palo Alto, Ca. 94301, 6. November 1986 (aktualisiert am 12. Oktober 1989)

[31] ROTH, G.: Registering Two Overlapping Range Images. In: *3dim* 00 (1999), S. 191

[32] RUSINKIEWICZ, S. ; LEVOY, M.: Efficient Variants of the ICP Algorithm. In: *3dim* 00 (2001), S. 145–152

[33] SHARP, G. C. ; LEE, S. W. ; WEHE, D. K.: Multiview Registration of 3D Scenes by Minimizing Error between Coordinate Frames. In: *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part II*. London, UK : Springer-Verlag, 2002, S. 587–597

[34] SMITH, S. M. ; BRADY, J. M.: SUSAN - A New Approach to Low Level Image Processing. In: *Int. J. Comput. Vision* 23 (1997), Nr. 1, S. 45–78

[35] WIKIPEDIA: *Das Jacobi-Gesamtschrittverfahren, nach Carl Gustav Jakob Jacobi*. Version: 2006. `http://de.wikipedia.org/wiki/Jacobi-Verfahren` Wikipedia, Die freie Enzyklopedie

[36] WIKIPEDIA: *k-d-Baum*. Version: 2006. `http://de.wikipedia.org/wiki/K-d-baum` Wikipedia, Die freie Enzyklopedie

[37] WIKIPEDIA: *Quaternion*. Version: 2006. `http://de.wikipedia.org/wiki/Quaternion` Wikipedia, Die freie Enzyklopedie

[38] WINKELBACH, S. ; RILK, M. ; SCHÖNFELDER, C. ; WAHL, F. M.: Fast Random Sample Matching of 3d Fragments. In: *DAGM-Symposium*, 2004, S. 129–136