Sichtbarkeitsberechnung

Stand 14. Juli 2009

Dr. Liebau

GWR Gesellschaft für Wissenschaftliches Rechnen mbH

Zusammenfassung. Es werden die Definitionen zur Sichtbarkeit zwischen Paneelen und Cluster angegeben und entsprechende Datenstrukturen und Verfahren zur Berechnung vorgestellt. Weiterhin werden Algorithmen zur komprimierten Speicherung und Ausnutzung der Symmetrie beschrieben.

Inhaltsverzeichnis

2000 *Mathematics Subject Classification.* 65D18, 68U05.

Key words and phrases. Visibility, Clustering.


EUROPÄISCHE UNION
Europäischer Fonds
für regionale Entwicklung
Investition in Ihre Zukunft!
Allgemeine Informationen: www.efre.brandenburg.de

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Ministeriums für Wirtschaft des Landes Brandenburg und der EU gefördert. Die Verantwortung für den Inhalt der Veröffentlichung liegt beim Autor.

1. Einleitung

Die Berechnung der Sichtbarkeits-Funktion in der Integralgleichung zum Wärmeaustausch bzgl. diffuser Strahlung ist ein anspruchsvolles Problem aus dem Gebiet der „Algorithmischen Geometrie“. Vor allem in der Literatur zur Computer-Graphik finden sich zu diesem Thema eine große Anzahl von Artikeln (etwa Durand[**4**][**5**], Richard[**14**], Bittner[**2**] und viele mehr).

Der zu approximierende Operator

$$(K\,q)(\mathbf{x}) = \int_\Gamma k(\mathbf{x},\mathbf{y})\, vis(\mathbf{x},\mathbf{y})\, q(\mathbf{y})\; d\mathbf{y}, \quad \mathbf{x} \in \Gamma.$$

der Integralgleichung enthält neben dem Kern $k(\mathbf{x},\mathbf{y})$ die Funktion

$$(1) \qquad vis(\mathbf{x},\mathbf{y}) : \Gamma \to \{0,1\}, \quad vis(\mathbf{x},\mathbf{y}) = \begin{cases} 1 & \text{falls kein Hindernis zwischen } \mathbf{x} \text{ und } \mathbf{y} \text{ liegt} \\ 0 & \text{sonst} \end{cases},$$

die entscheidet, ob der Punkt $\mathbf{x}$ der Oberfläche Γ den Punkt $\mathbf{y}$ „sehen“ kann. Was der Fall ist, falls die Verbindungslinie ganz im Inneren des Objektes liegt. Auch wenn die Definition von vis relativ harmlos aussieht, ist doch die Berechnung der Sichtbarkeiten der rechenintensivste Schritt. Für eine Triangulierung ist (1) für alle Mittelpunkte der beteiligten Paneele zu bestimmen.

Definition 1. *Sichtbarkeitsproblem*

$$(2) \qquad \textit{Bestimme } vis(\mathbf{x},\mathbf{y}) \quad \textit{für alle Paare } (\mathbf{x},\mathbf{y}) \textit{ von Paneel-Mittelpunkten.}$$

(2) ist erst einmal ein $\mathcal{O}(n^2 \log n)$ Problem: $\log n$ Rechenoperationen für den einzelnen Sichtbarkeitstest und insgesamt n^2-Sichtbarkeitstest[1]. Der Abschnitt 4.1 zeigt wie sich die Symmetrie ausnutzen läßt und Abschnitt 5 geht näher auf das Komplexitätsproblem ein.

[1] Das Problem ist symmetrisch, außerdem kann sich ein Paneel nicht selbst sehen: die Komplexität verringert sich sofort auf $\mathcal{O}(\frac{n(n-1)}{2})$

2. Cluster

Die Oberfläche Γ sei in n disjunkte Dreiecke (Paneele, panels, faces, Facetten) zerlegt

$$\Gamma = \bigcup_{i \in \mathcal{I}} t_i, \qquad \mathcal{I} \text{ eine Indexmenge}, \quad \mathcal{P} := \bigcup_{i \in \mathcal{I}} t_i$$

Dabei sind $\mathcal{I}$ eine Indexmenge und $\mathcal{P}$ die Menge der Paneele.

DEFINITION 2 (Cluster). *Jede Teilmenge $\mathcal{C} \subset P$ ist ein Cluster. Ist $\mathcal{I}$ die Indexmenge zu $\mathcal{P}$, dann definiert jede Teilmenge von $\mathcal{I}$ in kanonischer Weise einen Cluster.*

Durch das Zusammenfassen von benachbarten Dreiecken können *Cluster* τ erklärt werden. Benachbarte Cluster können wiederum zu größeren Clustern vereinigt werden. Dieses Vorgehen kann solange durchgeführt werden, bis die gesamte Oberfläche in einem einzigen Cluster zusammengefaßt ist. Das Verfahren definiert in natürlicher Weise den *cluster tree* $\mathcal{T}$: die Blätter sind die ursprünglichen Paneele, die Oberfläche Γ ist die Wurzel von $\mathcal{T}$ und die restlichen Knoten sind die erzeugten Cluster. Eine formale Definition für $\mathcal{T}$ ist z. B. in Hackbusch[**8**] oder auch in Graham[**7**] gegeben.

DEFINITION 3 (Clusterbaum, Hackbusch[**8**]). *Die Menge aller Teilmengen von $\mathcal{P}$ sei mit $\mathcal{S}$ bezeichnet.*

(1) *Alle Knoten, d. h. Cluster, von $\mathcal{T}$ repräsentieren Teilmengen von $\mathcal{P}$.*

(2) *Γ ist die Wurzel von $\mathcal{T}$*

(3) *Jedes Paneel $t \in \mathcal{P}$ entspricht einem Blatt von $\mathcal{T}$.*

(4) *Ist $\tau \in \mathcal{T}$ kein Blatt. Dann sei $sons(\tau)$ die Menge der Söhne; für sie gilt $|sons(\tau)| \geq 2$. Weiter gilt*

$$\tau = \bigcup_{\tau' \in sons(\tau)} \tau'.$$

Der Cluster τ repräsentiert die Vereinigungsmenge seiner Söhne.

Die Menge der von $\tau \in \mathcal{T}$ erreichbaren Blätter werde mit `leaves`(τ) *bezeichnet:*

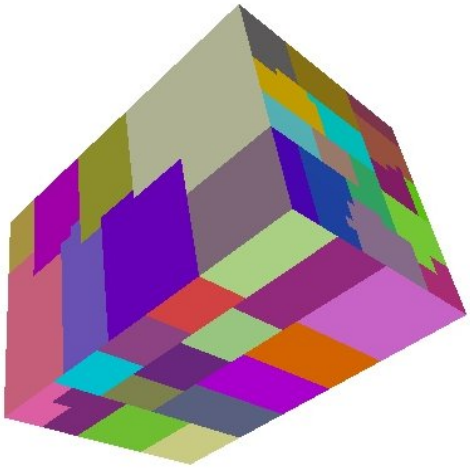
$$q \in \texttt{leaves}(\tau) \quad :\Leftrightarrow \quad \text{es gibt einen Pfad von } q \text{ nach } \tau.$$

Die Menge der Abkömmlinge (descendants) eines Clusters sei $\mathcal{D}(\sigma)$

$$\tau \in \mathcal{D}(\sigma) \quad :\Leftrightarrow \quad \text{es gibt einen Pfad von } \tau \text{ nach } \sigma.$$

Geeignete Teilmengen der Knoten des Clusterbaums liefern eine disjunkte Zerlegung der ursprünglichen Triangulierung (siehe Abbildung 1).

Der Algorithmus 1 erzeugt den Cluster-Baum rekursiv, beginnend mit den Blättern von unten nach oben (bottom-up). Die Clusterbildung berücksichtigt dabei über die Auswahl der Nachbarn bekannte geometrische Zusammenhänge (CAD-region).



(a)

ABBILDUNG 1. Cluster (a) Unterboden, (b) Testproblem "Quader"

BEMERKUNG 2.1.

(1) *Die Anzahl der Cluster ist nicht größer als* $2\,n$.

$$\sharp\mathcal{T} < 2\,n.$$

(2) *Die Stufe* l *des Clusterbaums enthält höchstens* 2^l *Cluster.*

BEWEIS. (2) ist offensichtlich, (1) folgt mit der geo. Reihe (Annahme $\log_2 n \in \mathbb{N}$)

$$1 + 2 + 4 + \ldots + \frac{n}{2} + n = \sum_{i=0}^{\log_2 n} 2^i = \frac{2^{\log_2 n+1} - 1}{2 - 1} = 2\,n - 1$$

□

DEFINITION 4 (Teil- und Schnittmenge). *Ein Cluster* τ *wrid Teilmenge des Cluster* σ *genannt, falls dieses für die entsprechenden Blätter gilt:*

$$\tau \subset \sigma \quad :\Leftrightarrow \quad \texttt{leaves}(\tau) \subset \texttt{leaves}(\sigma).$$

Genauso hat man für die Schnittmenge

$$\tau \cap \sigma := \texttt{leaves}(\tau) \cap \texttt{leaves}(\sigma).$$

BEMERKUNG 2.2. *Seien* τ *und* σ *Cluster eines Clusterbaums. Dann gilt: entweder ist ihre Schnittmenge leer oder sie sind ineinander enthalten:*

$$\tau, \sigma \in \mathcal{T} \quad \Rightarrow \quad \tau \cap \sigma = \emptyset \quad \vee \quad (\tau \subset \sigma \quad \vee \quad \sigma \subset \tau)$$

Algorithm 1 erzeuge binären Cluster-Baum

```
void CreateClusterTree (ClusterTree &T, list<cluster> &L)
{L ist eine Liste mit Clustern}
if L.size = 1 then
  T.root = L(0);
else
  list<cluster> h(L); {kopiere L nach h; nicht unbedingt nötig}
  L.clear();
  while  h.size() > 0  do
    wähle ein h(i) und einen Nachbarn h(j) von h(i)
    {erzeuge einen neuen Cluster}
    cluster τ; τ.father = nil;
    sons(τ)= (h(i), h(j)); h(i).father = τ; h(j).father = τ;
    L.pushback(τ){speichere τ}
    h.delete(i); h.delete(j); {lösche Cluster i und j}
  end while
  CreateClusterTree (T, L) {rekursiver Aufruf mit verkleinerten L}
end if

{die nachfolgenden Aufrufe erzeugen einen Clusterbaum}
liste<cluster> L(P) // alle Paneele in die Liste L
ClusterTree T;
CreateClusterTree (T,L);
```

BEWEIS. Ist $q \in \tau \cap \sigma$ ein Blatt, dann gibt es einen Pfad von q nach τ und einen Pfad von q nach σ. Aufgrund der Baumstruktur von $\mathcal{T}$ gibt es nur einen Pfad von q zur Wurzel $\texttt{root}(\mathcal{T})$, so daß dieser Pfad τ und σ enthält. Je nach Reihenfolge ist dann $\tau \subset \sigma$ oder $\sigma \subset \tau$. □

Jedem Knoten im Clusterbaum wird mittels einer Postorder-Traversierung des Clusterbaumes eine Nummer (`orderId`) zugewiesen, die dann eine Ordnung auf den Paneelen induziert (Algorithmus 8). Für einen Cluster σ sei diese mit $\sigma.OrderId$ bezeichnet. Zusätzlich wird noch die kleinste Ordnungszahl gespeichert, die in der Menge der Nachfolger des Clusters enthalten ist:

$$\sigma.minOrderId := \min\{\tau.OrderId \ : \ \tau \in \mathcal{D}\sigma\}$$

Die Ordnung auf der Clustermenge ist dann

$$p < q \quad :\Leftrightarrow \quad p.orderId < q.orderId, \quad p, q \in \mathcal{T}. \tag{3}$$

Insbesondere gilt (3) auch für die Paneele.

Algorithm 2 Ordnung bzgl. der Clusterknoten

```
void ClusterTree::MakeOrder(cluster t)
t->minOrderId = OrderId;
if  t->IsLeaf() then
  t->orderId = OrderId++;
else
  MakeOrder(t->left);
  MakeOrder(t->right);
  t->orderId = OrderId++;
end if

{/*------- Bestimmung der Ordnung ---------*/}
static ClusterTree::OrderId=0;
T.MakeOrder(T.root);
```

BEMERKUNG 2.3. *Sei σ ein innerer Knoten des Clusterbaums und* $\texttt{leaves}(\sigma)$ *wie oben die Menge der erreichbaren Blätter, dann gilt*

$$q < \sigma, \qquad \textit{für alle} \quad q \in \texttt{leaves}(\sigma).$$

Durch die Hinzunahme der kleinsten Ordnungsnummer hat man noch allgemeiner

BEMERKUNG 2.4. *Es gilt*

(1) *Für die zwei Cluster $\sigma, \tau \in \mathcal{T}$ gilt*

$$\tau \subset \sigma \quad \Leftrightarrow \quad \sigma.minOrderId \leq \tau.OrderId < \sigma.OrderId. \tag{4}$$

d. h. $\tau \subset \sigma \Leftrightarrow \tau.OrderId \in [\sigma.minOrderId, \sigma.OrderId)$.

(2)

$$t \in \texttt{leaves}(\mathcal{T}) \quad \Leftrightarrow \quad t.minOrderId = t.OrderId.$$

Sei $\mathcal{C} := (\sigma_0, \sigma_1, \ldots, \sigma_m)$ eine mittels der Relation "$<$" geordnete Menge von Clustern aus einem Clusterbaum $\mathcal{T}$, d. h.

$$(\sigma_0, \sigma_1, \ldots, \sigma_m) \quad \text{mit} \quad \sigma_i \in \mathcal{T}, \quad \sigma_{i-1} < \sigma_i, \quad i = 1, \ldots, m.$$

Der Menge $\mathcal{C}$ kann durch

$$\texttt{leaves}(\sigma_0, \sigma_1, \ldots, \sigma_m) := \bigcup_{i=0}^{m} \texttt{leaves}(\sigma_i) \tag{5}$$

die entsprechenden Blätter zugeordnet werden. Sollte nun für ein j

$$\sigma_j.minOrderId \leq \sigma_{j-1}.OrderId, \quad j \in \{1, \ldots, m\}, \tag{6}$$

sein, dann ist Aufgrund der Anordnung die Bedingung (4) erfüllt: $\sigma_{i-1} \subset \sigma_i$. Und es folgt

$$\bigcup_{i=0}^{m} \texttt{leaves}(\sigma_i) = \bigcup_{i=0, i\neq j-1}^{m} \texttt{leaves}(\sigma_i).$$

Damit können diejenigen Cluster, die (6) erfüllen, aus der Clustermenge gestrichen werden, ohne daß sich die Menge der erreichbaren Blätter (5) ändert.

SATZ 2.1.

$$\texttt{leaves}(\sigma_0, \sigma_1, \ldots, \sigma_m) = \texttt{leaves}(\sigma_{i_0}, \sigma_{i_1}, \ldots, \sigma_{i_k}), \quad k \leq m, \tag{7}$$

wobei die reduzierte Menge mittels (6) *bestimmt wird.*

3. Sichtbarkeit zwischen Paneelen und Clustern

DEFINITION 5 (Sichtbarkeit). *Seien p und q Paneele aus $\mathcal{P}$ mit Normalenvektoren n_p und n_q und den Mittelpunkten $\mathbf{x}$ und $\mathbf{y}$. Die Paneele induzieren mittels ihrer Mittelpunkte und der Normalenvektoren die Ebenen E_p und E_q. p sieht q (und umgekehrt) genau dann, wenn*

(a) $\mathbf{x} \in E_q^+$ *und* $\mathbf{y} \in E_p^+$

und

(b) *die Strecke $r : \mathbf{x} + t\,(\mathbf{y} - \mathbf{x})$, $t \in [0,1]$, mit keinem anderen Paneel aus $\mathcal{P}$ kollidiert:*

$$r(t) \cap \mathcal{P} = \{p, q\}.$$

Die formale Definition der Sichtbarkeitsfunktion ist demnach

$$vis(p,q) = \begin{cases} 1 & \textit{falls } \left(\mathbf{x} \in E_2^+ \wedge \mathbf{y} \in E_1^+\right) \wedge (r \cap \mathcal{P} = \{p,q\}) \\ 0 & \textit{sonst} \end{cases}. \tag{8}$$

Das Paneel p sieht einen Cluster τ genau dann, wenn p alle Blätter von τ sieht. In Zeichen

$$vis(p,\tau) := \prod_{l \in \texttt{leaves}(\tau)} vis(p,l). \tag{9}$$

Ein Cluster τ sieht den Cluster σ, falls jedes Paneel $p \in \texttt{leaves}(\tau)$ den Cluster σ sieht

$$vis(\tau,\sigma) := \prod_{p \in \texttt{leaves}(\tau)} vis(p,\sigma). = \prod_{p \in \texttt{leaves}(\tau)} \prod_{q \in \texttt{leaves}(\sigma)} vis(p,q) \tag{10}$$

BEMERKUNG 3.1. *Per Definition sieht sich ein Paneel selbst nicht, da $\mathbf{x} \notin E_p^+$ gilt.*

$$vis(p,p) = 0, \qquad \forall\, p \in \mathcal{P}.$$

Auch ergibt sich aus 5(a),(b) sofort die Symmetrie von vis:

$$vis(\mathbf{x},\mathbf{y}) = vis(\mathbf{y},\mathbf{x}) \quad \textit{für alle } \mathbf{x},\mathbf{y} \in \Gamma. \tag{11}$$

Zur Entscheidung, ob sich zwei Paneele (oder ein Paneel und ein Cluster) sehen, also ob sich zwischen ihnen in gerader Linie kein Objekt befindet, ist es sinnvoll die beteiligten Paneele in einer angepaßten Datenstruktur zu speichern. Möglich sind eine Bounding Volume Hierachy (BVH), der „Binary Space Partition Tree“ (BSP-Tree, Naylor[**13**], Berg[**3**]), der Kd-Tree und verwandte Strukturen wie die Kombination von Kd-Tree und einer octree Einteilung. Diese Datenstrukturen repräsentieren die relative Position der Paneele in der Szene zueinander.

Zur Berechnung der Sichtbarkeits-Funktion $vis(\mathbf{x},\mathbf{y})$, $\mathbf{x} \in t_1$ und $\mathbf{y} \in t_2$ wird der Strahl (ray) $r(t) = \mathbf{x} + t\,(\mathbf{y} - \mathbf{x})$ für $t \in (0,1)$ auf Kollisionen mit anderen Paneelen überprüft. Dieser Test kann mit Hilfe der Suchstruktur schnell ausgeführt werden. Fussell[**6**] und Subramanian, Fussell[**17**] beschreiben so ein Verfahren für einen Kd-Tree in Verbindung mit einer Bounding Volume Hierachy (BVH).

Da zu jedem Cluster $\tau \in \mathcal{T}$ sehr einfach eine Bounding-Box (Abkürzung: AABB für axis-aligned bounding box) erklärt werden kann, ist durch $\mathcal{T}$ schon eine BVH-Struktur vorgegeben. Der Sichtbarkeitstest kann entsprechend dem Algorithmus 3 durchgeführt werden. Der Algorithmus prüft nur diejenigen Cluster, deren Boundigbox auch vom Strahl r durchdrungen werden, alle anderen — und das sind die meisten — werden nicht betrachtet.

Algorithm 3 Sichtbarkeitstest zwischen Paneelen

```
  bool visibility (const Paneel &p, const Paneel &q) const;
Ensure: vis(p, q) nach Definition 5
  if  ! (p ∈ E_q^+ ∧ q ∈ E_p^+)  then
    return  false
  else
    ray r(p,q);
    return  ( CheckRay (SuperCluster,r) == NonBlocking )
  end if
```

Algorithm 4 Sichtbarkeitstest: Clusterbaum "gegen" ray

```
  enum TBlocking { Blocker, NonBlocking };
  {Anfangs und Endpunkt des rays zählen nicht als Blocker}
  TBlocking CheckRay (Cluster &τ, ray &r)
  if  τ->BB.InterSection( Ray) == empty  then
    return  NonBlocking
  else if  τ ∈ P then
    return triangleRayIntersection (τ,r);
  else
    TBlocking vis(NonBlocking)
    for all  Söhne s von τ  do
      vis = CheckRay (s, ray);
      if  vis == Blocker  then
        break;
      end if
    end for
    return  vis
  end if
```

4. Speicherplatzreduktion: komprimierte Sichtbarkeit

Eine einfache Möglichkeit eine einmal ermittelte Sichtbarkeit bzgl. eines Paneels p zu speichern, ist es die Indizes (oder Pointer) aller von p aus sichtbaren Paneele in einer Liste oder einer Menge zu speichern. Etwa

```
...
12    3 4 7 8 13
13    1 5 6 11 12
...
```

(Paneel 12 sieht die Paneele 3,4,7,8 und 13). Bei diesem Vorgehen muß mit einem Speicheraufwand von $\mathcal{O}(n^2)$ gerechnet werden[2]. Geht man von einem 4 Byte integer-Typ aus — das ist optimistisch — und einer durchschnittlichen Sichtbarkeit von 83.7 % des gesamten Modells[3], so sind dieses bei einem Problem mit $n = 20000$ Paneelen etwa 1 GB und 1 Million Paneele benötigen 2.5 Terabyte. Das ist aussichtslos!

Ein Teilaspekt der panel clustering Idee kann auf den Bereich „Sichtbarkeit“ übertragen werden: analog zum Begriff der *zulässigen Überdeckung* bzgl. eines Paneels kann auch eine minimale zulässige (Sichtbarkeit-)Überdeckung $C_{\rm vis}$ für ein Paneel angegeben werden: sind alle Paneele eines Clusters sichtbar, so wird der Cluster, nicht aber die Paneele selbst, gespeichert. Mit Hilfe dieser Datenstruktur ist dann der Sichtbarkeitstest selbst durchführbar:

$$vis(\mathbf{x},\mathbf{y}) = 1 \quad \Leftrightarrow \quad \exists\, \tau \in C_{\rm vis}(\mathbf{x}) \quad \text{mit} \quad \mathbf{y} \in \texttt{leaves}(\tau). \tag{12}$$

Bzgl. der Rechenzeit ist die Speicherung von sichtbaren Clustern in $C_{\rm vis}$ problematisch, da für eine Auswertung die Blätter erst ermittelt werden. müssen. Gegenüber der direkten Speicherung ergeben sich aber Vorteile im Hinblick auf den Speicherplatzbedarf, was vorläufig wichtiger ist als die benötigte Rechenzeit[4].

Der Aufwand zur Ermittlung dieser neuen Überdeckungen für alle Paneele ist zwar immer noch in der Größenordnung $\mathcal{O}(n^2)$, aber diese Berechnung muß für jede Geometrie nur einmal durchgeführt werden. Anschließend werden — wie beim panel clustering — alle $C_{\rm vis}$ mit einem Speicheraufwand, der wesentlich kleiner ist als n^2, im Cluster-Baum abgelegt.

---

[2] Das Ausnutzen der Symmetrie an dieser Stelle führt dann auf Suchprobleme.

[3] Der Wert 83.7 % stammt aus einem Modellproblem.

[4] Evtl. können die Kandidaten τ noch eingeschränkt werden: teile $C_{\rm vis}$ auf in zwei Mengen `leaves` und `cluster` $\Rightarrow$ die Auswertung von (12) wird schneller.

DEFINITION 6 (Überdeckung). *Eine Menge $\mathcal{C} = \{\tau_1, \tau_2, \ldots, \tau_m\}$, $\tau_i \in \mathcal{T}$, wird Überdeckung der Menge $S \subset \mathcal{P}$ von Paneelen genannt, falls*[5]

$$S = \bigcup_{i=1}^{m} \texttt{leaves}(\tau_i).$$

In Zeichen: $\mathcal{C} = cov(S)$. Die Menge aller Paneele in $\mathcal{C}$ ist $S =: \texttt{leaves}(\mathcal{C})$.

Eine Überdeckung $\mathcal{C}$ von S mit Clustern aus $\mathcal{T}$ heißt minimale Überdeckung, falls

$$|\mathcal{C}| \leq |\mathcal{D}|, \quad \text{für alle Überdeckungen } \mathcal{D} \text{ von } S.$$

Anmerkung: Es gilt immer $S = \texttt{leaves}(cov(S))$, aber i. Allg. nicht $\mathcal{C} = cov(\texttt{leaves}(\mathcal{C}))$, da die Überdeckung nicht eindeutig ist.

Algorithm 5 Markierung der minimalen Überdeckung

```
Covering::MarkMinCovering(cluster t)
if  t->IsLeaf() then
  return
else
  MarkMinCovering(t->left);
  MarkMinCovering(t->right);
  t->marker = t->left->marker and t->right->marker;
  if  t->marker  then
    { falls beide Söhne markiert sind, lösche die Markierung der Söhne }
    t->left->marker = t->right->marker = false
  end if
end if
```

Algorithm 6 Minimale Überdeckung

```
{/*———— erzeuge Minimale Überdeckung ————*/}
Covering::mincov (ClusterTree &T, SetOfPanels &S)
T.MarkLeafSet(S) { markiere alle Paneele aus S im Clusterbaum }
T.MarkMinCovering(T.root()) { Markierung der Cluster mit einer Postorder-Traversierung }
return T.GetMarkCluster();
```

BEMERKUNG 4.1. *Sind der Clusterbaum $\mathcal{T}$ und eine Menge von Paneelen S gegeben, so läßt sich die minmale Überdeckung $\mathcal{C}$ einer Menge S relativ leicht mit dem Algorithmus 6 finden. In Zeichen*

$$\mathcal{C} = mincov(S).$$

Der Aufwand ist von der Größenordnung $\mathcal{O}(n)$, da der gesamte Clusterbaum durchlaufen wird.

[5]Zur Ermittlung der Sichtbarkeiten ist eine disjunkte Überdeckung, wie beim panel clustering, nicht notwendig, sondern nur eine Überdeckung.

Algorithm 7 Bestimme die Blätter bzgl. einer Überdeckung

```
void ClusterTree::GetLeaves(cluster t, ListOfLeaves &leaves)
if  t->IsLeaf() then
  leaves += t;
else
  GetLeaves(t->left);
  GetLeaves(t->right);
end if
{/*------- GetLeaves ---------*/}
ListOfLeaves ClusterTree::GetLeaves (Covering &C)
ListOfLeaves result;
for σ ∈ C do
  GetLeaves(σ, result);
end for
return  result;
```

BEMERKUNG 4.2. *Ist $\mathcal{C}$ eine Überdeckung, dann liefert Satz 2.1 und Algorithmus* **??** *sofort eine reduziert Überdeckung.*

Frage: gibt es eine Variante, die mit $\mathcal{O}(m)$, $m = |S|$, auskommt? Wahrscheinlich ja:

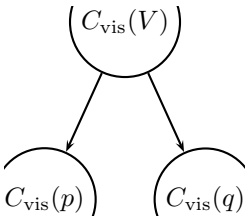
(1) markiere alle Leafs, die zu S gehören
(2) wähle ein Paneel q aus S
(3) bestimme von q ausgehend den größten Cluster σ in S mit $q \in \sigma$; die Menge $\texttt{leaves}(\sigma)$ wird dabei sukzessive bestimmt
(4) entferne $\texttt{leaves}(\sigma)$ aus S: $S := S \setminus \texttt{leaves}(\sigma)$
(5) falls $S = \emptyset$: stopp, sonst gehe zu (2)

Dieses Verfahren kommt mit einem Aufwand der Größenordnung $\mathcal{O}(m\,log(m))$ aus, wenn man annimmt, daß die Mengenoperation in konstanter Zeit durchgeführt werden. Es benutzt aber kompliziertere Operationen, so daß der Algorithmus 6, der nur Zeiger-, Boolsche-Operation und rekursive Aufrufe verwendet, für den ganzen Clusterbaum durchaus schneller sein kann.

BEMERKUNG 4.3. *Algorithmus 7 berechnet die Menge der Paneele, die in einer Überdeckung C liegen. Hier ist der Aufwand (maximal) in der Größenordnung $\mathcal{O}(|C| \log n)$, wenn man annimmt, daß eine push_back-Operation für die Liste* `ListOfLeaves` *in konstanter Zeit möglich ist.*

Master-Idee: Jetzt kommt eine weitere Idee ins Spiel, die sich wiederum auf das panel clustering-Verfahren zurückführen läßt: betrachten wir zwei Paneele p und q, die einen gemeinsamen Vorfahren V haben (p und q sind Brüder mit Vater V). In sehr vielen Fällen werden sich die Sichtbarkeits-Überdeckungen $C_{\text{vis}}(p)$ von p und $C_{\text{vis}}(q)$ von q nicht viel unterscheiden. Statt nun die in *beiden* Mengen vorkommenden

Cluster doppelt zu speichern, ist es ausreichend die betreffenden Cluster in dem gemeinsamen Vorfahren V abzulegen.



Damit ist ein Kompressionsverfahren für die Sichtbarkeiten festgelegt, daß aus zwei Komponenten besteht:

(1) sind von einem Paneel aus alle Söhne eines Cluster sichtbar, so wird der Cluster, d. h. die ihm eindeutig zugeordnete Nummer, in C_{vis} gespeichert und nicht die entsprechenden Blätter

(2) von Brüdern gemeinsam gesehene Cluster und Blätter werden im Vaterknoten gespeichert.

DEFINITION 7 (minimale Sichtbarkeitsüberdeckung). *Sei $p \in \mathcal{P}$ ein Paneel.*

- *Die Menge aller von p aus sichtbaren Paneele wird mit $S(p)$ bezeichnet.*

$$S(p) := \{q \in \mathcal{P} \ : \ vis(p,q) = 1\}.$$

- *Für einen Cluster $\tau \in \mathcal{T}$ sei*

$$S(\tau) := \cap_{p \in \texttt{leaves}(\tau)} S(p)$$

  die Kernsichtbarkeit des Clusters τ.
- *Eine Überdeckung von $S(p)$ wrid Sichtbarkeitsüberdeckung genannt und mit $C_{\text{vis}}(p)$ bezeichnet.*
- *Eine Überdeckung $C_{\text{vis}}(p)$ von $S(p)$ mit Clustern aus $\mathcal{T}$ heißt minimale Sichtbarkeitsüberdeckung, falls*

$$|C_{\text{vis}}(p)| \leq |D_{\text{vis}}(p)|, \quad \textit{für alle Überdeckungen } D_{\text{vis}}(p) \textit{ von } S(p).$$

  Sie wird mit $C_{\text{vis}}^{\min}(p)$ bezeichnet.

Nach dieser Definition sind die Paneele in $S(\tau)$ für jedes Paneel in $\texttt{leaves}(\tau)$ sichtbar:

$$p \in \texttt{leaves}(\tau), \quad q \in S(\tau) \quad \Rightarrow \quad vis(p,q) = 1.$$

Umgekehrt sieht auch jedes $q \in S(\tau)$ den Cluster τ:

$$q \in S(\tau) \quad \Rightarrow \quad vis(q,\tau) = 1.$$

Für $p, q \in \mathcal{P}$ mit $V = \texttt{father}(p) = \texttt{father}(q)$, $V \in \mathcal{T}$, d. h. die beiden Paneele sind Brüder, seien die Sichtbarkeiten $S(p)$ und $S(q)$ bzw. die Überdeckungen

$$C_{\text{vis}}(p) = cov(S(p)) \quad \text{und} \quad C_{\text{vis}}(q) = cov(S(q))$$

ermittelt.

- eine Überdeckung der Schnittmenge $S(p) \cap S(q)$ wird im Knoten V gespeichert:

$$C_{\text{vis}}(V) := C_{\text{vis}}(p, q) := C_{\text{vis}}(p) \cap C_{\text{vis}}(q). \tag{13}$$

- p erhält „nur“ diejenigen Paneele und Cluster, die nicht schon in $C_{\text{vis}}(V)$ liegen:

$$W := \{q \,:\, q \in C_{\text{vis}}(p) \,\wedge\, q \notin C_{\text{vis}}(p, q)\} \tag{14}$$

  Die in p gespeicherten Paneele sind dann $\texttt{leaves}(W) = S(p) \setminus S(V)$ (Mengendifferenz). Mit der de Morganschen Regel $X \setminus (A \cap B) = X \setminus A \cup X \setminus B$ wird in p gerade eine Überdeckung der Menge

$$S(p) \setminus S(q)$$

  gespeichert.
- genauso erhält q: $S(q) \setminus S(p)$

Demnach erhält der Vater V die Schnittmenge $S(V)$ und in den beiden Söhnen ist die symmetrische Differenz $S(q) \triangle S(p)$ gespeichert. Die minimale zulässige Überdeckung für das Blatt p ergibt sich dann aus den direkt zugeordneten Clustern und der im Vater gespeicherten Clustermenge:

$$S(p) = S(V) \cup (S(p) \setminus S(q)) \,.$$

Die Gleichheit kann natürlich auch formal bewiesen werden: sei $X := S(p) \setminus S(q)$. Wird nun die rechte Seite mit der leeren Menge ergänzt so gilt

$$X = S(p) \setminus S(q) = (S(p) \setminus S(p)) \cup (S(p) \setminus S(q)) = S(p) \setminus S(V)$$

(siehe oben). Es bleibt

$$S(V) \cup X = S(V) \cup (S(p) \setminus S(V)) = S(p).$$

Das Verfahren kann nun auf aller Cluster in ausgedehnt werden: für alle $\tau \in \mathcal{T}$: speichere in τ die Menge $S(\tau)$, während die Söhne σ_1 und σ_2 von τ jeweils einen Teil der symmetrischen Differenz $S(\sigma_1) \triangle S(\sigma_2)$ enthalten. Statt nun die Mengen $S(p)$ bzw. $S(\sigma)$ direkt zu speichern, werden Überdeckungen dieser Mengen in den Knoten des Clusterbaums $\mathcal{T}$ gespeichert.

FOLGERUNG 4.1.

$$q \in S(\tau) \quad \Leftrightarrow \quad vis(\tau, q) = vis(q, \tau) = 1$$

und

$$\sigma \in C_{\text{vis}}(\tau) \quad \Leftrightarrow \quad vis(\tau, \sigma) = 1$$

BEWEIS. Die zweite Beziehung folgt aus

$$vis(\tau, \sigma) = \prod_{q \in \texttt{leaves}(\tau)} vis(q, \sigma) = \prod_{q \in \texttt{leaves}(\tau)} \prod_{p \in \texttt{leaves}(\sigma)} vis(q, p)$$

□

Anmerkung Die Speicherung $C_{\text{vis}}(V)$ nach (13) ist mit Hinblick auf den Speicherplatz nicht optimal. Z. B. wäre denkbar, daß Paneel p einen Cluster σ sieht, der vom Bruder q nur teilweise gesehen wird: $vis(q, r) = 1$ mit $r \subset \sigma$. Da $C_{\text{vis}}(V)$ als Schnittmenge der Überdeckungen definiert ist, verbleiben sowohl σ wie auch r in ihren jeweiligen Knoten.

Alternativ zu (13) wäre

$$C_{\text{vis}}(V) := mincov(\texttt{leaves}(C_{\text{vis}}(p)) \cap \texttt{leaves}(C_{\text{vis}}(q))). \tag{15}$$

eine Möglichkeit zur Definition der im Vater V gespeicherten Menge. Entsprechend müßten die Mengen für die Söhne angepaßt werden.

4.1. Ausnutzen der Symmetrie. Aufgrund der Symmetrie (11) ist es ausreichend die Sichtbarkeit für jede Menge $\{p, q\}$, $p, q \in \mathcal{P}$, und nicht etwa für jedes geordnete Paar (p, q), zu bestimmen. Ist eine Ordnung für die Menge der Paneele definiert, so kann

$$W := \{(p, q) \,:\, p < q,\ p, q \in \mathcal{P}\}$$

gesetzt werden. Das Sichtbarkeitsproblem (2) ist dann

$$\text{bestimme} \quad vis(p, q) \quad \text{für alle } (p, q) \in W. \tag{16}$$

Die Sichbarkeitsfunktion vis und die Ordnung der Paneele legen die Matrix

$$V := (vis(p, q))_{p,q \in \mathcal{P}} \in \mathbb{B}^{n \times n}$$

fest. (16) bedeutet somit, daß nur die obere Dreiecksmatrix von V bestimmt wird. Die untere Dreiecksmatrix ergibt sich anschließend durch "Spiegeln" an der Hauptdiagonalen.

Jedem Knoten im Clusterbaum wird mittels einer Postorder-Traversierung des Clusterbaumes eine Nummer (`orderId`) zugewiesen, die dann die Ordnung auf den Paneelen induziert (Algorithmus 8).

$$p < q \quad :\Leftrightarrow \quad p.orderId < q.orderId, \quad p, q \in \mathcal{P}. \tag{17}$$

Algorithm 8 Ordnung bzgl. der Clusterknoten

```
void ClusterTree::MakeOrder(cluster t)
if  t->IsLeaf() then
  t->orderId = OrderId++;
else
  MakeOrder(t->left);
  MakeOrder(t->right);
  t->orderId = OrderId++;
end if

{/*------- Bestimmung der Ordnung ---------*/}
static ClusterTree::OrderId=0;
T.MakeOrder(T.root);
```

BEMERKUNG 4.4. *Sei σ ein innerer Knoten des Clusterbaums und* $\texttt{leaves}(\sigma)$ *wie oben die Menge der erreichbaren Blätter, dann gilt*

$$q < \sigma, \qquad \textit{für alle} \quad q \in \texttt{leaves}(\sigma).$$

Es folgt

$$\tau < p \quad \Rightarrow \quad (p,q) \notin W \quad \textit{für alle} \quad q \in \texttt{leaves}(\tau). \tag{18}$$

BEWEIS. Aus der Postorder-Traversierung folgt, daß erst die Söhne und dann der Knoten selbst nummeriert werden. Die Blätter erhalten also eine kleinere Nummer. Sei nun $q \in \texttt{leaves}(\tau)$ und $\tau < p$ für ein $p \in \mathcal{P}$.

$$\tau < p \quad \Rightarrow \quad q < \tau < p \quad \Rightarrow \quad (p,q) \notin W$$

□

(18) wird nun dahingegen angewandt, daß bei Abstieg im Clusterbaum nur Tests $vis(p,\tau)$ für $p < \tau$ durchgeführt werden. Die Algorithmen 9 – 12 zeigen ein mögliches Vorgehen auf.

Problem: ... das wirkt doch der minimalen Überdeckung entgegen ... i. Allg. keine kompakten Indexmengen ...

4.2. Algorithmen. Das Verfahren zur Lösung des Sichtbarkeitsproblems (2) besteht aus 3 Schritten

(1) Sichtbarkeitstest: Bestimme von einem Paneel p aus die Menge der sichtbaren Paneele unter Berücksichtigung der Symmetrie:

$$\text{bestimme } C_{\text{vis}}^{\text{sym}}(p) := \{q \in \mathcal{T} \,:\, q \in C_{\text{vis}}(p) \wedge p < q\}, \quad \mathbf{p} \in \mathcal{P}$$

(2) Kompressionsschritt: Für einen inneren Knoten des Clusterbaums ist die Kernsichtbarkeit nach xxx zu berechnen

(3) Vervollständigung: nachdem (1) und (2) für den gesamten Clusterbaum abgearbeitet sind, ergänze die fehlenden Sichtbarkeiten $vis(p,q)$ mit $(p,q) \notin W$. Das ist der Übergang von $C_{\mathrm{vis}}^{\mathrm{sym}}(p)$ zu $C_{\mathrm{vis}}(p)$. Da sich damit auch die Einträge in den Überdeckungen ändern, sollte auch ein weiterer Kompressionsschritt durchgeführt werden.

```
class VisClusterNode
{
  public:
        VisMath::IntSet2 Cvis;
        pVisClusterNode father, left, right;
        int             index;            /**< Index in den Vektoren V[i], N[i], M[i] .                */
        int             orderID;          /** Ordnung der Knoten bei Preorder-Traversierung             */
        bool    Touch;
public:
        bool            PlaneVis;           /**< Hilfsgroesse: Sichtbarkeit von einer Ebene aus  */
        bool            isvisadmis;
};
```

Algorithm 9 Caller zur Berechnung der minimalen Sichtbarkeitsüberdeckung

```
  IntSet MinimumVisCovering (cluster pleaf)
  {
  IntSet Cmin;
  ReSetAllMarker(T.root()); { Löschen aller Marker }
  VisDivide(pleaf, T.root(), Cmin); { Bestimme die minmale Überdeckung }
  return  Cmin;
  }
```

Algorithm 10 Plane-Box-Test: $\tau \in E_p^-$?

```
  {Der Test klärt, ob die durchs Paneel p induzierte Ebene E_p den Cluster, zumindest teilweise, sieht.
  Das ist äquivalent zu τ ⊄ E_p^-. Das Ergebnis ist false, falls die Boundingbox von τ gänzlich im nega-
  tiven Halbraum liegt, ansonsten true.}
Ensure: !(τ ⊄ E_p^-)
  bool PlaneBoxTest (cluster p, cluster tau)
  bool result = tau->BB.IsInBox(pleaf->Center)) or tau->BB.PlaneVisBox(p);
  if  ! result  then
    tau->isvisadmis = false;
    tau->SetValueTouch(true);
  end if
  return  result;
```

Algorithm 11 Rekursive Bestimmung von $C_{\text{vis}}(p)$

```
IntSet VisClustering::GetCvis (pVisClusterNode p, pVisClusterNode tau)
{
tau->isvisadmis = false;
{ Ausnutzen der Symmetrie }
if  !donotSymmetric then
  if  (tau->orderID <= p->orderID) then
    return VisMath::IntSet();
  end if
end if
if  tau->IsLeaf()  then
  if  visibility(p,tau)==IsVis then
    tau->isvisadmis = true;
    return VisMath::IntSet(tau->GetId());
  else
    return VisMath::IntSet();
  end if
else
  { liegt τ ganz im negativen Halbraum, so ist der Cluster nicht sichtbar }
  if  BackSideTest(p,tau) == VisGeo::OutSide  then
    return VisMath::IntSet();
  end if
  VisMath::IntSet left, right;
  left = GetCvis(p,tau->left);
  right = GetCvis(p,tau->right);
  if  tau->left->isvisadmis and tau->right->isvisadmis  then
    tau->isvisadmis = true;
    return VisMath::IntSet(tau->GetId());
  else
    return (left+right);
  end if
end if
}
```

Algorithm 12 Sichtbarkeitstest

```
{ Der Cluster τ ist zulässig bzgl. der Sichtbarkeit von pleaf aus, falls alle Paneele leaves(τ))
sichtbar sind. Es sind also alle Blätter zu prüfen.
Im Vektor TmpVis wird die Sichtbarkeit von pleaf aus zwischengespeichert. Ist schon der Vater von
τ geprüft worden, so sind auch schon alle Paneele, die zu τ gehören auf Sichtbarkeit geprüft und
brauchen nicht nocheinmal berechnet zu werden. }
void IsVisAdmissible (cluster p, cluster tau)
{
if  tau->GetTouch()  then
  return  tau->isvisadmis;
end if
if  tau->IsLeaf  then
  { Ausnutzen der Symmetrie und des logischen Kurzschlußes}
  tau->isvisadmis = (p->orderID < tau->orderID) and (visibility(pleaf, tau) == IsVis);
else
  tau->isvisadmis = IsVisAdmissible(pleaf, tau->left) and IsVisAdmissible(pleaf,tau->right);
end if
tau->SetValueTouch(true);
return  tau->isvisadmis;
}
```

4.3. Block-Cluster-Tree. Ausgehend vom Clusterbaum $\mathcal{T}$ kann ein zweiter Clusterbaum folgendermaßen definiert werden; vgl. auch Hackbusch[**8**] oder Sauter/Schwab[**16**]

DEFINITION 8. (1) *$\mathcal{T}_2$ ist eine Teilmenge von $\mathcal{T} \times \mathcal{T}$, d. h. jeder Knoten von $\mathcal{T}_2$ ist ein Produkt $\tau \times \sigma$ von zwei Clustern $\tau, \sigma \in \mathcal{T}$.*

(2) *$\Gamma \times \Gamma$ ist die Wurzel.*

(3) *Die Söhne eines Knoten $\nu = \tau \times \sigma$ sind*

$$S(\nu) := \begin{cases} \{\tau' \times \sigma' : \sigma' \in S(\sigma),\ \tau' \in S(\tau)\} & \text{falls weder } \sigma \text{ noch } \tau \text{ ein Blatt ist} \\ \{\tau \times \sigma' : \sigma' \in S(\sigma),\ \tau \in S(\tau)\} & \text{falls } \tau \text{ ein Blatt ist } \sigma \text{ jedoch nicht} \\ \{\tau' \times \sigma : \sigma \in S(\sigma),\ \tau' \in S(\tau)\} & \text{falls } \sigma \text{ ein Blatt ist } \tau \text{ jedoch nicht} \\ \emptyset & \text{falls } \sigma \text{ und } \tau \text{ Blätter sind} \end{cases}$$

DEFINITION 9 (v-zulässig). *Ein Knoten $\nu = \tau \times \sigma$ des Block-Clusterbaums $\mathcal{T}_2$ heißt genau dann v-zulässig, falls $vis(\tau, \sigma) = 1$.*

DEFINITION 10 (Überdeckung). *$\mathcal{C} \subset \mathcal{T}_2$ wird Überdeckung genannt, falls*

$$\bigcup_{b \in \mathcal{C}} b = \Gamma \times \Gamma.$$

Die Überdeckung hießt v-zulässig, falls alle Knoten entweder v-zulässig oder Blätter sind.

Ziel ist eine minimale zulässige Überdeckung zu finden. Dazu wird beginnend mit der Wurzel $\Gamma \times \Gamma$ eine Teilmenge des Block-Clusterbaum rekursiv konstruiert: ist ein zulässiger Knoten oder ein Blatt von $\mathcal{T}_2$ erreicht, so wird nicht weiter verfeinert, ansonsten werden die vier Söhne erzeugt.

5. Komplexität

Das folgende Beispiel zeigt eindrucksvoll welchen Einfluß die Komplexitätsklasse des Verfahrens auf die zu erwartende Rechenzeit hat: für eine Geometrie mit 1.6 Millionen Paneelen (= Freiheitsgraden) gibt die nachfolgende Tabelle 1 unter der Annahme, daß 100000 Sichtbarkeiten pro Sekunde gerechnet werden können, eine Abschätzung für die Rechenzeiten in Abhängigkeit von der Komplexitätsklasse des Verfahrens an. Zum Vergleich sind in Tabelle 2 die entsprechenden Werte für eine Geometrie mit nur 160000 Paneelen aufgelistet. Für ein Verfahren mit quadratischer oder größerer Komplexität ist die Berechnung des ersten Problems aussichtslos, es sei denn man hat Jahre Zeit um auf das Ergebnis zu warten.

Es gibt aber Hinweise, daß das Problem mit weniger Aufwand zu lösen ist. Navazo et al. [**12**] erkennen über konvexe Hüllen und Bounding-Boxen der Cluster Abschattungen/Blocker in der Geometrie. Das Verfahren hat die Chance bedeutend schneller zu sein, als der direkte Test, es werden aber in [**12**] keine expliziten Aussagen zur Komplexität des Verfahrens gemacht. In [**15**] benutzt Saona-Vázquez eine

Komplexität	$\mathcal{O}(n)$	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^{1.5})$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2 \log n)$
Rechenzeit	16 Sekunden	4 Minuten	5.6 Stunden	300 Tage	11.5 Jahre

TABELLE 1. Abschätzung der Rechenzeiten für die Sichtbarkeiten, $n = 1.6$ Million

Komplexität	$\mathcal{O}(n)$	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^{1.5})$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2 \log n)$
Rechenzeit	1.6 Sekunden	19 Sekunden	10.6 Minuten	3 Tage	36 Tage

TABELLE 2. Abschätzung der Rechenzeiten für die Sichtbarkeiten, $n = 160000$

vielversprechende octree-Datenstruktur zur Erkennung von nicht sichtbaren Bereichen (bzgl. eines in der Szene positionierten Beobachters). Interessante Zugänge finden sich auch bei Laine[**11**] (view cell und Abschätzungen für die Sichtbarkeiten über Nachbarschaftsbeziehungen) und Haumont et al. [**9**], die die experimentell bestimmte Größenordnung $\mathcal{O}(n^{1.44})$ für den Aufwand ihres Verfahrens angibt, was überaus gut wäre. Für das Beispiel mit den 1.6 Millionen Freiheitsgraden beläuft sich dann die Rechenzeit auf 143 Minuten.

In Abschnitt 6 wird ein Ansatz beschrieben, der mit Hilfe von konvexen Hüllen eine Beschleunigung der Sichtbarkeitsberechnung in Aussicht stellt.

5.1. Erweiterte Sichtbarkeitsfunktion. Für spiegelde Oberflächen sind in einer Strahlungsberechnung mittels raytracer nicht nur die direkt sichtbaren, sondern auch die indirekt über den Spiegel sichtbaren Paneele zu beachten. Es sind Ketten/Pfade der Art $vis^{ex}(x,r) = vis(x,y)\, vis(y,z)\, vis(z,r)$ zu berechnen. vis^{ex} entspricht dann den "erweiterten Sichtfaktoren" in der Literatur, etwa [**1**].

Für die Abschätzung des Aufwandes im Fall von spiegelnden Teilflächen wird die Aufwandsabschätzung

$$\text{Aufwand zur Berechnung der erweiterten Sichtfaktoren} = \mathcal{O}(n^2\, k^r) \tag{19}$$

aus Arques[**1**] verwendet. Dabei sind n die Anzahl aller Paneele, k die Anzahl der spiegelnden Paneele und r die Anzahl der berücksichtigten Reflexionen.

5.2. Speicherplatz und Rechenaufwandsabschätzungen. Die nachfolgenden Abschätzungen zum Rechen- und Speicheraufwand basieren auf der Behandlung des 1. Modellproblems „Würfel ". Das Problem besteht aus 6708 Paneelen mit 3392 Punkten. Für ein Paneel sind durchschnittlich etwa 83.7% aller Paneele sichtbar (16.3% liegen dann durchschnittlich im Schatten).

Die Tabelle 3 dokumentiert den reinen Rechenaufwand zur Berechnung sämtlicher nicht verschwindender Sichtfaktoren für das Modellproblem. Die Berechnung der Sichtbarkeit ist in der Rechenzeit von

Anzahl n der Paneele	n^2	Anzahl der Sichtfaktoren (nach Sichtbarkeitstest)	Rechenzeit [sec]	Sichtfaktoren pro Sekunde
6708	44997264	31556386	41.26	760000 bis 765000

TABELLE 3. Rechenaufwand zu den Sichtfaktoren (Lambert-Formel; ohne Sichtbarkeit)

41 Sekunden für die etwa 3×10^7 Werte nicht(!) berücksichtigt. Zur Speicherung aller nicht trivialen Sichtfaktoren sind etwa 300 MB nötig[6].

Die Rechnungen wurden auf einem PC mit einem AMD Athlon 64 3200+ (2.0 GHZ) Prozessor durchgeführt. Zur groben Einschätzung der Rechenleistung habe ich den linpack-Benchmark-Test in der Java-Version[7] ohne irgendwelche Optimierungen durchgeführt. Der schnellste Rechner erreicht gegenüber dem benutzten AMD64 einen 10-fachen besseren Mflop-Wert: etwa 2000 Mflops/s gegenüber 200 Mflops/s. Ein Standard P4 Rechner erreicht höchstens den Faktor 2.3.

Es lassen sich nun Prognosen für verschiedene Szenarien ableiten. Bei der Vorhersage der Rechenzeiten sei gegenüber dem AMD64 ein 5-fach schnellerer Rechner angenommen: dieser Computer berechnet 3.8 Millionen Sichtfaktoren pro Sekunde mittels der Lambertschen Summenformel; die Rechenzeit zur Ermittlung Sichtbarkeit ist auch hier nicht enthalten. Genau wie bei dem Würfel wird wieder eine 84 %-tige Sichtbarkeit vorausgesetzt und ein Verfahren zur Lösung des Strahlungsproblems angenommen, das die Kenntnis der Sichtfaktoren explizit benötigt. Für die Abschätzung des Aufwandes im Fall von spiegelnden Teilflächen wird die Aufwandsabschätzung (**??**) verwendet. Unter diesen Bedingungen kann die Rechenzeit mit

$$\text{Rechenzeit in Sekunden} = \frac{(0.837\,n)^2\,k^r}{3812500}$$

abgeschätzt werden. Trotz des beachtlichen Faktors im Nenner wächst der Ausdruck über alle (menschlichen) Grenzen. Tabelle[8] 4 ist eine Auswertung dieser Schätzung, (Zeitangaben dort: s Sekunden, h Stunden, y Jahre).

Der geschätzte Rechenaufwand bzw. die Rechenzeit nach Tabelle 4 kann sicherlich noch reduziert werden. So sind aus Symmetriegründen eigentlich nur $(n^2 - n)/2$ Sichtfaktoren nötig (das hilft allerdings nichts, wenn auch für diese nicht genügend Speicherplatz zur Verfügung steht, vgl. Tab. 5).

[6] 256MB zur Speicherung der Werte (8 Byte double) und 60MB für die Indizes (2 Byte integer). Unter Ausnutzung der Symmetrie ist noch ein Speicherplatz von 150 MB nötig.

[7] `http://www.netlib.org/benchmark/linpackjava`

[8] Abweichend von den anderen Werten wurde in der Tabelle 4 für $n = 20000$ $k = 100$ und $k = 20000$ gewählt. Außerdem sind zur Übersichtlichkeit nur einige Felder ausgefüllt

n	n^2	k	Anzahl (erweiterte) Sichtfaktoren			Rechenzeiten		
			$r=0$	$r=1$	$r=2$	$r=0$	$r=1$	$r=2$
8000	64×10^6	1000	45×10^9	4.4×10^{10}	4×10^{13}	11.76 s	3.2 h	136 d
		2500						
10000	10^8	1000	70×10^6		7×10^{13}	18.3 s		212.6 d
		2500		10^{11}			12.7 h	
20000	4×10^8	100	2.8×10^8	2.8×10^{10}		73.5 s		8.5 d
		20000		5.5×10^{12}	10^{17}		17 d	932 y
$10^6/16$	3.9×10^9	1000	2×10^9			717.7 s	8.3 d	
		2500						
10^6	10^{12}	1000	2.8×10^{11}			51 h	5.8 y	
		2500			4.3×10^{18}			36417.8 y

TABELLE 4. prognostizierte Rechenzeiten: $r=0$ diffuser Fall, $r=1$ eine Reflexion, $r=2$ zwei Reflexionen

Aber die Tendenz bleibt erhalten: ein naives Gauß-Seidel-Verfahren, eine Fixpunkt-Iteration oder das progressive refinement sind $\mathcal{O}(n^2)$ Algorithmen. Werden die berechneten Sichtfaktoren nicht gespeichert, sondern „zeilenweise“ bestimmt und sogleich im Iterationsverfahren zur Lösung der radiosity/rendering Gleichung benutzt, dann spiegeln die Rechenzeiten aus der Tabelle 4 den Aufwand für *einen* Iterationsschritt wider. Die Tests zum panel clusterting ergaben für das Würfel-Beispiel (Strahlung gekoppelt mit einem sehr einfachen 1-Schichtmodell, gesamt Simulationszeit von 900 Sekunden), daß für die zeitliche Lösung 37 einzelne Strahlungsprobleme zu lösen sind, bei etwa 6 Iterationsschritten pro Aufruf des Lösers (Konvergenzrate ≈ 0.32). Die oben genannten Verfahren mit zeilenweiser Bestimmung (und nicht Speicherung der Sichtfaktoren) braucht dann zur Lösung eines ähnlichen zeitlichen Problems, bei gleicher Konvergenzrate — und die ist wirklich gut — die 200-fachen(!) Zeiten der Tabelle 4. Selbst auf einem Parallelrechner mit 16 Rechnern und optimalen speed up würde für den Fall $n=10^6$ nur die Berechnung der Sichtfaktoren etwa 40 Stunden benötigen. Sind noch 1000 spiegelnde Teilflächen über die erweiterten Sichtfaktoren zu berücksichtigen, dann kann man sich auf eine mehrjährige Wartetzeit einrichten: $8.3\,d\times200\approx4.5\,y$.

Anzahl der Paneele	8000	10000	20000	50000	$10^6/16$	100000	250000	1000000
Speicherplatz in GB	0.2	0.3	1.3	8.1	25.48	32.6	203.9	3262.2

TABELLE 5. Speicherplatz; Voraussetzungen zur Sichtbarkeit wie oben, Ausnutzen der Symmetrie

Neben der reinen Berechnung der Sichtfaktoren ist auch noch die Bestimmung der Sichtbarkeit ein rechenzeitliches Problem. Die Beantwortung der Frage: „sieht Paneel p Paneel q?“, d. h. die Auswertung einer Funktion `visibility(p,q)` ist wesentlich für alle Algorithmen.

Wieder soll hier das Würfel-Beispiel als Modellproblem dienen. Tab. 6 zeigt den Rechenaufwand

n	n^2	Anzahl der Sichtfaktoren	Anzahl Sichtbarkeitstest	Rechenzeit Sichtbarkeitstest [sec]	Sichtbarkeitstests pro Sekunde
6708	44997264	31556386	10033689	20.81	482157

TABELLE 6. Rechenaufwand zur Sichtbarkeit, Würfel, AMD64

bzgl. der Sichtbarkeit für das Modellproblem. Die Anzahl der Sichtbarkeitstests ist geringer als die Anzahl der Sichtfaktoren, da durch Informationen über die Regionen und die Lage eines Paneels schon viele Fälle ausgeschlossen werden können: so ist z. B. für die Gruppe „Lampe“ bekannt, dass kein Paneel aus dieser Gruppe ein anderes Paneel aus der gleichen Gruppe sieht. Trotzdem liegt die Anzahl der Sichtbarkeitstests in der Größenordnung $\mathcal{O}(n^2)$ und die Tendenzen, die sich für die reine Berechnung der Sichtfaktoren in Tab. 4 zeigen, bleiben bestehen: siehe Tabelle 7. Hier sind die Rechenzeiten in der zweiten Zeile optimistisch geschätzt, da neben einem schnellen Rechner auch noch angenommen wird, dass nur ein Viertel aller möglichen Tests auch wirklich durchgeführt werden muß. Bei der pessimistischen Schätzung fällt diese Annahme weg (also die 4-fachen Zeiten).

Anzahl der Paneele	8000	10000	20000	50000	$10^6/16$	100000	250000	1000000
Rechenzeit Sichtbarkeitstest (optimistisch)	6.6 s	10.3 s	41.4 s	259 s	405 s	1037 s	1.8 h	28.8 h
Rechenzeit Sichtbarkeitstest (pessimistisch)	26.5 s	41.4 s	165 s	1037 s	27 min	69 min	7.2 h	4.8 d

TABELLE 7. Prognostizierte Rechenzeit für den Sichtbarkeitstest (diffuser Fall); Voraussetzungen wie oben.

Für Verfahren, die die Sichtbarkeiten nicht speichern, sind die Zeiten der Tab. 7 wieder *pro* Iterationsschritt zu verstehen. Für den Fall $n = 20000$, konservative Schätzung, instationäres Verfahren wie oben, keine Reflexionen, ist dann mit einer Rechenzeit zur Bestimmung der Sichtbarkeit von ca. 9 h zu rechnen.

6. Sichtbarkeit ein $\mathcal{O}(n^2 \log n)$-Problem?

Zur Durchführung des oben beschriebenen Verfahrens wird jedes Paneel gegen jedes andere auf Sichtbarkeit getestet, d. h. es liegt ein Verfahren mit mindestens quadratischer Komplexität vor. Der Test selbst

kann dank der Datenstruktur mit $\mathcal{O}(\log n)$ Operationen durchgeführt werden. Der gesamte Aufwand ist also von der Größenordnung $\mathcal{O}(n^2 \log n)$.

Die folgende Bemerkung gibt möglicherweise einen Hinweis dahingehend, daß für nicht pathologische Situationen ein Sichtbarkeitstest mit weniger (Rechen-)Aufwand möglich ist. Vorausgesetzt sei, daß die Paneele in einem Clusterbaum organisiert und für die Cluster Boundingboxen bekannt sind.

BEMERKUNG 6.1. *Sei $p \in \mathcal{P}$ ein Paneel mit Mittelpunkt $\mathbf{x}$, $\tau \in \mathcal{T}$ ein Cluster und*

$$\mathcal{CH} := hull(p \cup \{b : b \in \tau\})$$

die konvexe Hülle von $p \cup \tau$. Hinreichend für $vis(p, \tau) = 1$ ist:

- $\tau \subset \partial\mathcal{CH}$
- *das Innere der konvexen Hülle $\mathcal{CH}$ hat mit allen Clustern aus $\mathcal{C} \setminus \{\tau\}$ einen leeren Durchschnitt.*

  $$\overset{\circ}{\mathcal{CH}} \cap \sigma = \emptyset, \qquad \sigma \neq \tau,\ \sigma \in \mathcal{C}$$

  (auf $\sigma \neq \tau$ kann verzichtet werden, da das Innere von $\mathcal{CH}$ betrachtet wird).

Höchstwahrscheinlich sind die Bedingungen auch notwendig.

Sowohl für die Konstruktion der konvexen Hülle $\mathcal{CH}$, als auch für den Test $\mathcal{CH} \cap \sigma = \emptyset$? sollten schnelle Algorithmen zur Verfügung stehen, d. h. für beide „Probleme“ sollte es Algorithmen mit konstantem Aufwand geben. Mit der Annahme, daß eine Sichtbarkeitsüberdeckung aus $\mathcal{O}(\log n)$ Clustern besteht, ergibt sich das gewünschte Ergebnis: der hinreichende Sichtbarkeitstest der obigen Bemerkung kann *für alle* Paneele aus $\mathcal{P}$ mit einem Aufwand der Größenordnung $\mathcal{O}(n \log n)$ durchgeführt werden.

Ist evtl. ein Test mit (einer) Boundingbox von τ möglich?

BEMERKUNG 6.2.

$$\mathcal{CH}(p \cup \tau) = \mathcal{CH}(p \cup \mathcal{CH}(\tau)).$$

Ein (hinreichender) Test entsprechend Bem. 6.1 könnte wie folgt aussehen:

- Bestimme zu jedem Cluster die konvexe Hülle $\mathcal{CH}(\tau))$
- Teste (einmal) für alle Cluster: $\tau \subset \partial\mathcal{CH}(\tau)$, d. h. teste ob τ konvex ist
- Bestimme $\mathcal{CH}(p \cup \mathcal{CH}(\tau))$ und teste $\mathcal{CH}(p \cup \mathcal{CH}(\tau)) \cap \sigma = \emptyset$?

Der Test wird zu einem hinreichendem Test, falls σ durch $\mathcal{CH}(\sigma)$ ersetzt wird.

Die konvexe Hülle selbst kann durch (eine) Boundingbox ersetzt werden, falls

- τ konvex ist und
- $p \in E(t)^+$ für jedes Paneel $t \in \tau$.

Unter diesen Bedingungen ist der Test dann wieder hinreichend.

Probleme

- Behandlung von Paneelen, die in einer Ebene liegen
- (schnelle) Algorithmen für die konvexe Hülle

Literatur

[1] Didier Arques, Sylvain Michelin, and Benoit Piranda. Extending the zonal method to specular surfaces. In Vaclav Skala, editor, *WSCG '98 (Sixth European Conference in Central Europe on Computer Graphics and Visualization)*, pages 27–34, Plzen, Czech Republic, 1998. University of West Bohemia.

[2] J. Bittner and P. Wonka. Visibility in Computer Graphics. Technical Report TR-186-2-03-03, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Vienna, March 2003.

[3] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry — Algorithms and Applications*. Springer–Verlag, Berlin Heidelberg, 1997.

[4] F. Durand. A multidisciplinary survey of visibility, 2000.

[5] Frédo Durand, George Drettakis, and Claude Puech. Fast and accurate hierarchical radiosity using global visibility. *ACM Transactions on Graphics*, 18(2):128–170, 1999.

[6] D. S. Fussell and K. R. Subramanian. Fast ray tracing using K-d trees. Technical Report CS-TR-88-07, University of Texas, Dept. Of Computer Science, Austin, March 1988.

[7] I. G. Graham, L. Grasedyck, W. Hackbusch, and S. A. Sauter. Optimal panel-clustering in the presence of anisotropic mesh refinement. *SIAM J. Numer. Anal.*, 46(1):517–543, 2007.

[8] W. Hackbusch. Panel Clustering Techniques and Hierarchial Matrices for BEM and FEM. Technical Report 71, Max-Planck-Institut für Mathematik in den Naturwissenschaften, Leipzig, 2003.

[9] Denis Haumont, Otso Makinen, and Shaun Nirenstein. A low dimensional framework for exact polygon-to-polygon occlusion queries. In *Rendering Techniques*, pages 211–222, 2005.

[10] S. Havemann. Modellierung in der Computergrafik Polygonnetze und Freiformflächen. www.graphics.tu-bs.de/lvcg03/modellierung_cg, 2003.

[11] Samuli Laine. A general algorithm for output-sensitive visibility preprocessing. In *Proceedings of ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games*, pages 31–39. ACM Press, 2005.

[12] Isabel Navazo, Jarek Rossignac, J. Jou, and R. Shariff. Shieldtester: Cell-to-cell visibility test for surface occluders. *Comput. Graph. Forum*, 22(3):291–302, 2003.

[13] F.B. Naylor. A tutorial on binary space partitioning trees. Technical report, Spaztial Labs Inc., ?

[14] Jim Richard. Parallel hierarchical radiosity on cache-coherent multiprocessors.

[15] Carlos Saona-Vázquez. Computing visibility approximations.

[16] S. Sauter and Schwab C. *Randelementmethoden*. Teubner, Stuttgart, 2004.

[17] K. R. Subramanian and D. S. Fussel. A search structure based on k-d trees for efficient ray tracing. Technical Report Tx 78712-1188, University of Texas, Dept. Of Computer Science, Austin, 1992.

GWR Gesellschaft für Wissenschaftliches Rechnen mbH, Potsdamer Strasse 18 A, 15413 Teltow

E-mail address: liebau@gwr-net.de