Meshing

Markus Schütz

e0825723

2013.10.06

Mathematica

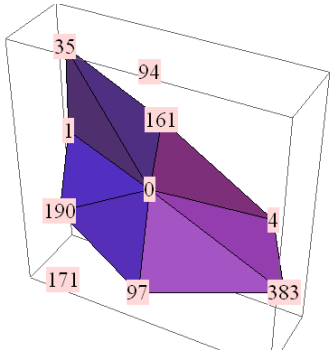
Die Mathematica Umsetzung befindet sich in

svn+ssh://projects.cg.tuwien.ac.at/data/svn/nn-connect3d/meshing_mathematica/

- meshing.nb Lädt eine Punktwolke und wandelt dieses in ein Mesh um.
- meshing.m Beinhaltet Funktionen in denen die Algorithmen umgesetzt wurden.

Zu beachten ist, dass indices in Mathematica von 1 starten. Code wie z.B. "index = 257+1" soll klarstellen, dass es sich hier um eine Umwandlung des zero based index 257 zu dem one based index 258 handelt. Zum Ausgeben werden die indices üblicherweise wieder zu zero based indices umgewandelt um sie mit den Ausgaben aus C++/CUDA vergleichen zu können.

Umbrella eines Vertex ermitteln



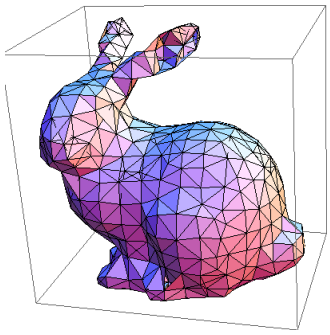
Siehe meshing.nb – Evaluate single vertex

Hier wird jeweils ein einzelnes vertex sowie dessen numNN Nachbarn ausgesucht und der Umbrella berechnet.

Verwendet wird dazu die Methode FindUmbrella:

1. Zuerst werden zum gegebenen Vertex dessen NN und Eigenvektoren bestimmt
2. Anschließend werden die Punkte in das Eigenvektorkoordinatensystem transformiert mit dem center-vertex als origin.
3. Bestimmen der Reihenfolge der Nachbarn um den center-vertex im oder gegen den Uhrzeigersinn.
4. Durch edge-flipping werden aus der Liste der im Uhrzeigersinn sortierten Nachbarn diejenigen entfernt, die bestimmte Kriterien nicht erfüllen

Vollständiges Mesh erstellen



Siehe meshing.nb – Evaluate all vertices

Durch anwenden von FindUmbrella auf alle Vertices wird ein vollständiges Mesh zusammengebaut.

Vergleich mit nnConnect

Siehe meshing.nb – Compare with nnConnect

In diesem Abschnitt wird die Anzahl der umbrella die in Mathematica und nnConnect ident sind berechnet.

C++/CUDA

Die CUDA Version ist in folgenden Dateien umgesetzt:

svn+ssh://projects.cg.tuwien.ac.at/data/svn/nn-connect3d/meshing_cuda

- src/Common.h Beinhaltet ein define zum festlegen der Anzahl der NN
- src/meshing.cpp Lädt Punktwolken, berechnet NN und Eigenvektoren
- src/meshing.cu Beinhaltet den CUDA Kernel

Zum laden der Punkte habe ich den Ply-Loader aus nnConnect kopiert.

Aufgerufen wird das Programm mit folgenden Parametern:

Meshing.exe <modelPath> <outPath> <blocksPerGrid> <threadsPerBlock> <#KernelRepetitions>

z.B.: bin\release\Meshing.exe resources\bun_zipper.ply out\bunny.obj 24 512 100

Linux

Zum Builden unter Linux müssen folgende skripte aufgerufen werden:

- compile_cpp.sh
- compile_cuda.sh
- compile_ply.sh
- link.sh

Windows

Für Windows gibt es die Visual Studio 2012 Solution Meshing.sln.

Berechnen der NN und Eigenvektoren

Siehe meshing.cpp – createVertexNeighborhoodProps

Mit der ANN Library werden zuerst für alle vertices deren NN bestimmt. Anschließend wird mit der EIGEN Library für jedes set von NN deren Eigenvektoren berechnet.

Sowohl die NN als auch die Eigenvektoren aller vertices werden in 2 Arrays gespeichert. Diese werden später CUDA mitgegeben.

CUDA Kernel

Siehe meshing.cu – umbrella() und edgeFlip()

Der Ablauf schaut folgendermaßen aus:

1. Transformieren der NN in den Eigenvektorspace mit dem center-vertex als dessen origin.
2. Speichern der im Uhrzeigersinn sortierten NN in der Variable orderedOuterKNN.
3. Edge-flipping auf die outer KNN. Jedes mal wenn ein vertex entfernt wird, werden alle Nachfolgenden vertices in der Liste um eins nach links verschoben und der Rest mit -1 aufgefüllt.
4. Das Edge-flipping wird solange wiederholt bis kein vertex mehr geflippt werden kann.
5. Die Liste der verbleibenden vertices wird in das Output Array geschrieben. Für jeden Vertex gibt es einen bestimmten offset im Array und eine fixe Anzahl an Plätzen (2+numNN) in die das Ergebnis gespeichert wird. Die erste und zweite Stelle sind für den vertex-index (möglicherweise obsolet) und die Anzahl der NN bestimmt. Anschließend kommen die verbliebenen NN.

Das Ergebnis des CUDA Kernels ist die Liste der UmbrellaVertices. Diese werden dann in C++ zu UmbrellaTriangles umgewandelt.

Speichern des Ergebnisses

In meshing.cu wird zuletzt noch die Anzahl der consensusTriangles berechnet, d.h. die Dreiecke die 3x vorkommen, und in consensusTriangles.txt gespeichert.

Die Dreiecke die an meshing.cpp zurückgegeben werden, werden dort in der saveMesh() Methode in ein obj File gespeichert.

Results

Alle Versuche sind auf einer GeForce GT 330M mit 24 Blöcken und 512 Threads getestet worden.

Model	#vertices	Durchläufe	Zeit	vertices/s
bun_zipper_res4.ply	453	1000	0.452s	~1m
bun_zipper_res3.ply	1889	1000	0.920s	~2m
bun_zipper.ply	35947	1000	14.849s	~2.4m