Dokumentation des Routenplanungstools mit parallelisiertem A\*

Luka Juric

Hochschule für angewandte Wissenschaften München

Modul: Routenplanung, Prof. Dr. Abmayr

WS 21/22

Libraries

*Parallel Computing Toolbox*

Erlaubt es SPMD Blöcke zu erstellen. Diese führen parallelisiert Funktionen aus und können sich gegenseitig Informationen zusenden.

*Symbolic Math Toolbox*

Erlaubt es die Präzision der Nachkomastellen zu erhöhen.

*Statistics Toolbox*

Berechnung von Entfernungen in verschiedenen Metriken.

*Mapping Toolbox*

~

Dateien

*shpToGeoEucAdj.m*

Eine der wichtigsten Funktionen. Transformiert das ShapeFile in Listen aus Tupeln aus den euklidischen (startEndList), den geographischen (startEndListGeo) Start- und Endpunkten der Straßensegmente besteht. Deren Verbindungen werden in eine Adjazenzmatrix gespeichert.

*addTraversalCache.m*

Die Traversal Datenstruktur besteht aus einem start- und endIndex, welcher für die Indizes aus der startEndList steht. Cached ein Traversal aus den Parametern.

*[aStarCollector.m aStarThread.m aStarPathFinderX.m]*

Der aStarPathFinderX ist die Hauptfunktion für den parallelisierten A\*. Dieser

startet 2x aStarThread.m- und 1x aStarCollector.m-Threads in einem SPMD-Block.

Die 2 aStarThreads füllen ihre closedListen mit dem A\* bis zu einer festgelegten Buffergröße aus. Sind beide Buffer voll, werden diese an aStarCollector übergeben, welcher beide ClosedListen vergleicht. Wird ein gemeinsamer Index gefunden, wird aus beiden ClosedListen und der Adjazenzmatrix 2 TraversalListen der jeweiligen Threads mit Indizes der Roadsegmente aus dem Shapefile erstellt. Da Geoshow in SPMD nicht funktioniert kann nicht innerhalb der Funktion gezeichnet werden. Deshalb müssen die TraversalListen zurückgegeben werden, um diese außerhalb eines SPMD-Blocks in 2 verschiedenen Farben zu zeichnen.

*searchTraversalCache.m*

Sucht im TraversalCache ob bereits eine TraversalListe mit jeweiligen start- und endIndex besteht.

*aStarPathFinder.m*

Bekommt die startEndList(/Geo) und Adjazenzmatrix aus shpToGeoEucAdj. Berechnet und zeichnet eine A\* Route. Führt davor jedoch einen Cache-Check durch.

*aStarConnectedFinder.m*

A\* ohne endIndex in closedList check. Gut zum finden zusammenhängender Graphen.

*calcWorldFile.m*

Berechnung eines ESRI-World Files mit Pixeldaten die aus GIMP extrahiert wurden. (myBoston.jgw)

*[checkLaplace.m createLaplace.m delEdgeLaplace]*

Überprüft die Validität, Erstellt eine vollständige, oder löscht eine Kante aus der Laplace Matrix.

*clearPointsAndPaths.m*

Entfernt alle Zeichnungen auf der Map.

*create\_kd\_tree.m*

Erstellt einen KD Tree, welcher so modifiziert wurde, dass dieser ab einer bestimmten Tiefe terminiert. Die Leafs enthalten die GeoListe einer Knotenpartition mit ihrem Index in der startEndList(/Geo) an der dritten Spalte.

*pointMatch.m*

Bekommt einen Punkt und eine Liste and Punken. Gibt aus dieser den euklidisch nächsten zurück.

*search\_kd\_tree.m*

Bekommt einen gewählten GeoPunkt. Returnt die GeoListe einer Knotenpartition mit den nächsten Punkten. (Diese wird bei der Punktwahl immer mit pointMatch kombiniert um anschließend den nächsten Punkt zur Wahl zu finden).

*<name>Sandbox.m*

Script File um die Funktion <Name> zu testen oder .mat Files zu bilden.

*[init\_shapefile.m, getting\_started.m]*

Funktionen aus der ersten Vorlesung. Erstere um das ShapeFile in NAD83 Geodaten umzuwandeln, zweitere um die Straßenklassen aus dem ShapeFile zu extrahieren.

*save\_kd\_tree.m*

Speichert KD Trees mit verschiedenen Depths ab.

*kdBenchmark.m*

Benchmarked die Suche von Knotenpartitionen anhand eines bestimmten Geo-Punkts in den KD Trees.

*Multithreading Ordner*

Experimente mit der Matlab Parallelized Toolbox.

*ui.mlapp*

Das Benutzerinterface.

*Multithreading Ordner*

*[antAlgo.m, calcAntTour.m]*

Der Ameisenalgorithmus aus der Vorlesung.

Messungen

