
Team 12

Frédéric Blondeel	26h
Martijn Debeuf	35h
Toon Sauvillers	h
Dirk Vanbeveren	21h
Bert Van den Bosch	h
Seppe Van Steenberghe	1h

Inhoudsopgave

1	Introductie	2
2	Delaunay triangulatie	2
2.1	S-hull	2
2.2	Bowyer-Watson	2
3	Testen	3
4	Valkuilen	5
5	Besluit	5

1 Introductie

Wanneer schermen herkent en geïdentificeerd zijn, moeten ze ook nog aan elkaar gelinkt worden. Hiervoor zal eerst elk scherm zijn eigen positie toegewezen krijgen. Daarna kunnen zijn burens bepaald worden met behulp van deze positie. Dit geeft de nodige informatie om bijvoorbeeld een slang van scherm naar scherm te laten bewegen. Delaunay triangulatie is een goede manier om de correcte burens te vinden. Dit verslag behandelt de keuzes die gemaakt zijn alsook een uitleg bij het gebruikte algoritme en zijn tijdscomplexiteit. De mogelijke beperkingen worden met deze kennis geduid.

2 Delaunay triangulatie

Om verschillende punten mooi te verbinden, kan er gebruik gemaakt worden van een Delaunay triangulatie. Deze wordt toegepast op de middelpunten van de schermen. Wanneer deze triangulatie geprojecteerd wordt, zullen de liggingen van de schermen mooi worden weergegeven. Bij een Delaunay triangulatie worden alle punten zo verbonden dat ze verscheidene driehoeken vormen. De zijden mogen niet overlappen en de kleinste hoek in de configuratie moet maximaal zijn. [1]

De triangulatie kan op vele manieren worden berekend, in dit verslag zijn er twee soorten algoritmen onderzocht. Het S-hull algoritme en het Bowyer-Watson algoritme.

2.1 S-hull

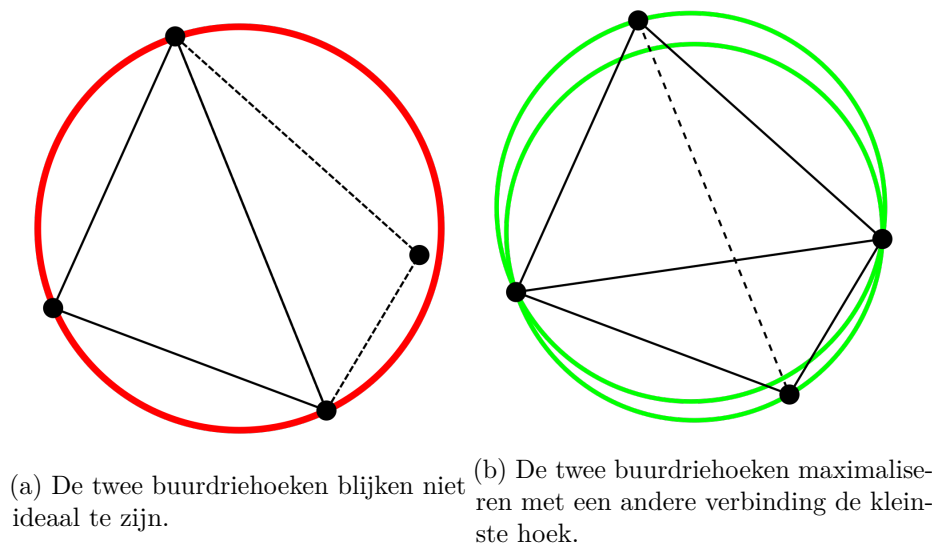
Het eerste algoritme is het S-hull algoritme. Het is een $O(n \log(n))$ algoritme, gebruikmakend van een radiaal propagerende sweep hull. S-hull start met twee punten, één daarvan is willekeurig geselecteerd. Het tweede punt is het dichtsbijzijnde. Vervolgens zal er een derde worden gezocht dat samen met de twee voorgaande een zo klein mogelijke cirkel maakt. Het middelpunt van deze cirkel wordt gebruikt om alle punten te sorteren t.o.v. de afstand met het gevonden middenpunt. Deze gesorteerde punten zullen één voor één toegevoegd worden aan de configuratie.

Nadat ze sequentieel zijn toegevoegd, is er een niet overlappende triangulatie gekomen. Om een Delaunay triangulatie te bekomen moet er echter ook gekeken worden naar de hoeken. Elke driehoek zal nu worden afgegaan en er wordt gekeken of er geen andere driehoek kan gevormd worden met de buurdriehoeken zodat de minimale hoek gemaximaliseerd wordt, zie figuur 1. [2]

Dit algoritme is echter niet gebruikt. Ook al is het algoritme $O(n \log(n))$ het vereist een zeer goede data management om deze snelheid te behalen. [3] Er is daarom gekozen voor een simpeler algoritme, namelijk het Bowyer-Watson algoritme.

2.2 Bowyer-Watson

Het Bowyer-Watson algoritme is wel geïmplementeerd. Het heeft een tijdscomplexiteit van $O(n^2)$, dit is aanzienlijk trager dan het S-hull algoritme. [4] Het vereist echter een minder complex data management en is daarom ook makkelijker te implementeren. De tijdscomplexiteit is ook relatief te zien, er zijn namelijk maar maximaal 120 detecteerbare schermen. Meer kunnen er door het identificatiemechanisme niet worden gevonden. Gezien het laag aantal punten, moet het algoritme niet nodeloos complex worden en kan er gekeken worden naar ‘tragere’ methoden, zoals deze Bowyer-Watson.



Figuur 1: Het maximaliseren van de kleinste hoek [1]

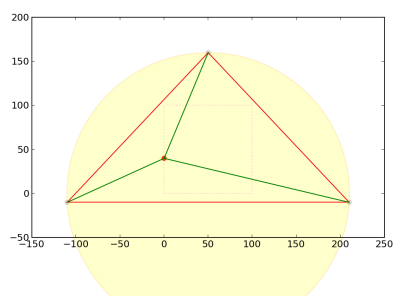
Bowyer-Watson gaat er van uit dat punten enkel worden toegevoegd in een al bestaande Delaunay triangulatie. Als eerste worden er twee superdriehoeken gezocht. Deze driehoeken zullen alle te trianguleren punten bevatten. De implementaties waarop het algoritme is gebaseerd [4] [5] stelden beiden een ‘superdriehoek’ voor, zie figuur 2a. Echter is het eenvoudiger om een omkaderende vierhoek te vormen en deze op te delen in twee driehoeken. Vervolgens worden alle punten één voor één toegevoegd.

Voor elk punt worden alle driehoeken gezocht waarvan het punt in de omschreven cirkel zit. Wanneer twee driehoeken eenzelfde zijde delen, wordt deze verwijderd. Alle punten van de omschreven veelhoek van de twee driehoeken worden nu verbonden met het toegevoegde punt, zie figuur 2b. Met deze werkwijze zal er op elk moment een Delaunay triangulatie zijn en moeten de driehoeken achteraf niet meer overlopen worden. Met het gevolg dat het data management voor deze methode minder complex is.

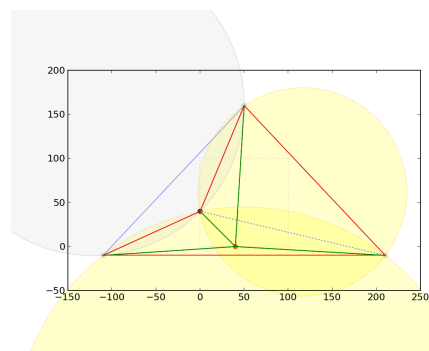
Eens alle punten toegevoegd zijn, worden de driehoeken die één of meer hoeken van de omkaderende vierhoek bevatten verwijderd, zie figuur 2c. Aangezien deze driehoeken aan de buitenkant liggen is dit toegestaan. Er zal een Delaunay triangulatie overblijven van alle onderzochte punten.

3 Testen

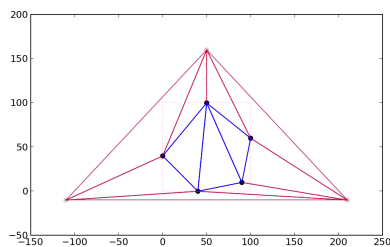
Voor de correctheid van het Bowyer-Watson algoritme zijn bepaalde subalgoritmen heel belangrijk. Zo is er een subalgoritme om te controleren of een punt wel degelijk binnen een cirkel ligt. Deze werd getest met een paar cases waar het punt in de cirkel, op de cirkel en uit de cirkel ligt. Bij het subalgoritme waar twee driehoeken gevormd worden die alle punten omvatten wordt er gekeken of elk punt in één van beide ligt. Hierbij wordt gebruik gemaakt van meetkundige stellingen. Een punt ligt binnen een driehoek wanneer het voor elke zijde van deze driehoek aan dezelfde kant ligt, bij het doorlopen van deze zijden in éénzelfde richting. Dit wordt gecontroleerd aan de hand van het kruisproduct tussen de vector van de zijde en de vector van het punt. [6]



(a) De rode superdriehoek waarin alle punten zullen worden toegevoegd.



(b) Een nieuw punt wordt toegevoegd. In geel de omschreven cirkels. De gestippelde zijde wordt verwijderd, de groene toegevoegd.



(c) In rood alle driehoeken verbonden met de superdriehoek, deze worden uiteindelijk verwijderd.

Figuur 2: Het Bowyer-Watson algoritme [4]



Figuur 3: Van colineaire punten kan geen triangulatie gevonden worden.



Figuur 4: Van bijna colineaire punten kan geen triangulatie gevonden worden door afrondingsfouten bij berekeningen.

4 Valkuilen

In theorie kan er van elke opstelling waarbij de punten niet allemaal colineair zijn een triangulatie worden gevonden, zie figuur 3. Echter door afronding bij de berekeningen zal er bij bijna colineaire punten geen juiste configuratie gevonden worden, zie figuur 4.

5 Besluit

De Delaunay triangulatie kan gebruikt worden om buurschermen te detecteren en de onderlinge oriëntatie te vinden. Het zorgt voor een niet-overlappende configuratie die de kleinste hoek maximaliseert. Enkel wanneer alle punten colineair zijn of er te weinig punten zijn, zal er geen enkele triangulatie gevonden worden. Met behulp van meetkunde kan er gecontroleerd worden of de juiste aan te passen driehoeken gekozen worden. Een supervierkant vervangt de superdriehoek zoals gebruikt in de bronnen.

Referenties

- [1] Delaunay triangulation. https://en.wikipedia.org/wiki/Delaunay_triangulation.
- [2] David Sinclair. S-hull: a fast sweep-hull routine for delaunay triangulation. s-hull.org, 2010.
- [3] Erik Thune Lund. Implementing high-performance delaunay triangulation in java. mathesis, Department of Informatics, University of Oslo, 2014.
- [4] Bowyer-watson algorithm. https://en.wikipedia.org/wiki/Bowyer%E2%80%93Watson_algorithm.
- [5] S.W. Sloan and G.T. Houlsby. An implementation of watson’s algorithm for computing 2-dimensional delaunay triangulations. Technical report, Department of Civil Engineering, University of Newcastle and Department of Engineering Science, University of Oxford, 1984.
- [6] Dirk Roose. *Toepassingen van meetkunde in de informatica*. Scientica Cursusdienst, 2007.