Gegevensstructuren en Algoritmen : Practicum 3

Gilles Ceulemans – r0752986

18 mei 2020

Introductie

In dit practicum is het de bedoeling een Image Compositing algoritme te ontwikkelen dat 2afbeeldingen samenvoegt tot een afbeelding waarbij de grens tussen deze afbeeldingen zo min mogelijk zichtbaar moet zijn.

Als basis van de code wordt er gebruikt gemaakt van de methodes seam() en floodfill().
De methode Seam() bepaalt de grens tussen de 2afbeeldingen door gebruik te maken van Dijkstra's kortste pad algoritmen.

Floodfill() gaat ervoor zorgen dat de nieuw gecreëerde afbeelding zal ingevuld worden met pixels van de huidige afbeeldingen.

Het resultaat is een nieuwe afbeelding dat bestaat uit 2afbeeldingen die aan elkaar "genaaid" zijn.

Vraag 1 : Geef de grafe die als input dienen voor het kortste pad algoritme.

Afbeelding1:

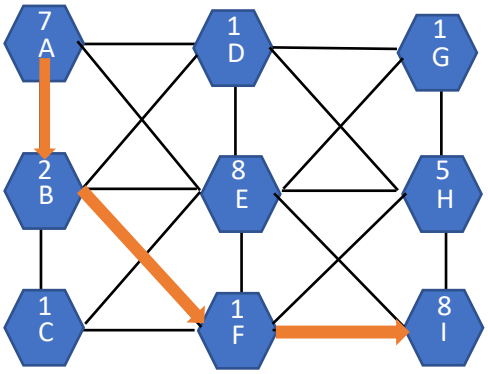
A(7,0,0)	D(0,1,0)	G(0,0,1)
B(2,0,0)	E(0,8,0)	H(0,0,5)
C(1,0,0)	F(0,1,0)	I(0,0,8)

Afbeelding2:

A(0,0,0)	D(0,0,0)	G(0,0,0)
B(0,0,0)	E(0,0,0)	H(0,0,0)
C(0,0,0)	F(0,0,0)	I(0,0,0)

Resultaat:

A(7)	D(1)	G(1)
B(2)	E(8)	H(5)
C(1)	F(1)	I(8)



De pixels worden voorgesteld door letters, de getallen die in de graaf staan stellen de kleuren afstand voor van de 2 gegeven afbeeldingen op die pixel.

Vraag 2 : gebruik van een andere kleur afstandsformule.

In plaats van de euclidische afstand met 3 kleur waarden te gebruiken zullen we nu gebruik maken van een euclidische afstands-formule die enkel 2 kleuren gebruikt

$$\sqrt{(rx-ry)^2+(gx-gy)^2+(bx-by)^2} \rightarrow \sqrt{(rx-ry)^2+(gx-gy)^2}$$

Om op deze vraag correct te kunnen antwoorden zullen we enkele gevallen moeten onderscheiden.
Deze gevallen zullen voornamelijk zijn met de hoeveelheid blauw er in de afbeeldingen zitten.

Geval1: 2 Afbeeldingen die enkel uit blauw bestaan moeten gecombineerd.
Als gevolg van de nieuwe functie zal elke kleurafstand per pixel 0 zal zijn.
Hierdoor kan er geen efficiënte keuze gemaakt worden. In dit geval gaat het Dijkstra algoritme de grens kiezen doormiddel van de kortste afstand in pixels naar de end-node.

Geval2: 2 Afbeeldingen die beide niet de kleur blauw bevatten.
In dit geval gaat de nieuwe functie sneller werken omdat je blauw sowieso kan laten vallen.
In de eerste functie zal er altijd nul nog worden berekent wat tijd zal kosten.
Dit verschil gaat echter zeer minimaal zijn.

Geval3: 2Afbeeldingen die alle kleuren bevatten.
Hier zal het afhangen van hoe doorslaggevend de kleur blauw kan zijn, bij pixels waarbij de groen- en rood- waarde bijna gelijk zijn zal blauw hier een belangrijke rol inspelen en zal er een betere keuze kunnen gemaakt worden met de eerste functie die blauw ook meer in rekening neemt. Als blauw niet zo doorslaggevend is zal de nieuwe functie die blauw niet in rekening neemt wellicht sneller werken omdat men hier dan een bewerking minder moet doen.

Vraag 3.1 : wat is de tijdscomplexiteit van een afbeelding met als grootte 1xN of N x 1?

De naad van de afbeelding zal maar 1 pad kunnen volgen waarbij de tijdscolmplexiteit ~N zal zijn.
Dit komt omdat we gebruik maken van een PriorityQueue dit telkens maar een element zal hebben.

Vraag 3.2 :Zal het Dijskstra algoritme even snel zijn in functie van het aantal pixels met grootte N x N?

Moesten we dit toepassen op een afbeelding met grootte N x N dan zal het Dijkstra-algoritme trager werken, dit is al te voorspellen omdat er nu telkens meerdere elementen in de PriorityQueue zullen zitten.

We zullen hier de best-case tijdscomplexiteit berekenen, dit is wanneer de seam volledig diagonaal zal lopen.

Lengte van het pad : $\sqrt{2N^2}$

De links boven hoek heeft slechts 3 buren, de rest van de pixels die niet gelegen zijn aan de buitenkant hebben steeds 8 buren. We hebben dus $8*(N-1)^2 + 3N$ Edges. En N^2 Vertices.
De tijdscomplexiteit van het Dijkstra algoritme zal dus $18*(N-1)^2*\log(N)$

Het Dijkstra algoritme zal dus aanzienlijk trager werken bij een grootte van N x N

Vraag 4 : Hoe voorkom je dat de seam complexe vormen aanneemt?

In de huidige code kan de seam bij elke node elke kant op gaan(boven, onder, links en rechts).
Moesten we in de code voorwaardes implementeren zodat de seam de buren die zich links, linksboven, boven en rechtsboven bevinden niet meer toevoege zodat het Dijkstra algoritme deze weg ook niet gaan verkennen.

Vraag 5 : hoe ziet de uitvoer er uit moest het algoritme het langste pad zoeken.

Door het langste pad te zoeken dan zou elke pixel deel moeten uitmaken van de seam.
Bij onze implementatie wordt de seam vervangen door de 2^{de} afbeelding. Aangezien de grens al de pixels Zal bevatten zal de uitkomst de 2^{de} afbeelding voorstellen.

Stel dat deze afbeeldingen niet even groot zijn zullen andere delen van de seam niet gekleurd worden.