Algoritmen en Datastructuren 2
Taak 1 - Grafen

Academiejaar 2022-2023 (1e zittijd)
Assistent: Youri Coppens (Youri.Coppens@vub.be)

De deadline voor deze taak vind je in sectie 4.

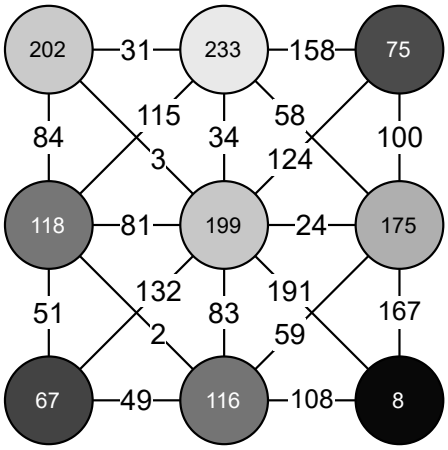
1 Inleiding

Digitale beeldverwerking is alomtegenwoordig in tal van toepassingen. Om verschillende objecten op een digitale foto te kunnen herkennen, moet men proberen pixels op te breken in visueel correcte *segmenten*. Als men bijvoorbeeld de achtergrond kan onderscheiden van de voorwerpen, kunnen verdere analyses de aanwezige objecten identificeren. Het proces om een afbeelding in delen op te splitsen noemt men *segmentatie* (in het Engels: *image segmentation*). Tegenwoordig wordt hiervoor regelmatig naar technieken uit de Artificiële Intelligentie gegrepen (bijv. Deep Learning), maar de Computer Vision (de discipline die zich hiermee bezighoudt) kent een rijke geschiedenis van andere technieken om beelden te segmenteren, waaronder het gebruik van grafen.

Een digitale foto is een gestructureerde verzameling van pixels. Segmentatie komt overeen met het opdelen van pixels in disjuncte delen. Een foto wordt doorgaans wiskundig voorgesteld als een matrix waarbij iedere plaats in de matrix overeenkomt met een pixel en zijn corresponderende kleurwaarde. Monochrome afbeeldingen hebben slechts één grijswaarde per pixel. Gewogen grafen zijn een alternatieve voorstelling voor monochrome afbeeldingen. Knopen (vertices) stellen de pixels voor en bogen (edges) leiden naar omliggende pixels in de afbeelding. Het gewicht op de boog duidt het absolute verschil in kleurwaarde aan tussen beide knopen. Een simpel voorbeeld hiervan is te zien in Figuur 1.

2 Opdracht

Het doel van deze taak is het **implementeren van een segmentatie-algoritme gebaseerd op grafen** in R^7RS, namelijk het algoritme van Felzenszwalb-Huttenlocher [1]. In het bijgeleverde bestand `a-d/segmentation.rkt` dien je de procedure `segment` te vervolledigen. Met het bestand `main.rkt` kun je je implementatie demonstreren.



Figuur 1: Voorbeeld van een gewogen graaf die een monochrome afbeelding kan voorstellen. Iedere knoop is een pixel waaruit bogen naar de omliggende pixels vertrekken. De knopen werden in deze figuur ingekleurd volgens hun respectievelijke grijswaarden (die ook in de knopen werden genoteerd). Merk op dat een gewogen graaf deze grijswaarden niet expliciet bewaart! Deze staan hier louter ter illustratie.

Het algoritme creëert segmenten vertrekkende van individuele knopen in de gewogen graaf $G = (V, E)$ met knopen V, bogen E en wegingsfunctie $w : E \mapsto \mathbb{R}$. De segmenten vormen een partitie van V. Een segment (component) zullen we aanduiden met C. De bogen bevatten informatie over kleurverschillen tussen naburige pixels. Indien deze verschillen voldoende klein zijn, kunnen naburige segmenten samengevoegd worden. Om te beslissen of twee segmenten al dan niet samengevoegd kunnen worden, wordt voor beide componenten het inwendige verschil bepaald en de kleinste van beide waardes gekozen (formule 1). Bogen die een lager gewicht hebben dan dit minimale inwendige verschil, kunnen segmenten doen samensmelten. De formule hiervoor is als volgt:

$$MInt(C_1, C_2) = \min(\iota(C_1) + \tau(C_1), \iota(C_2) + \tau(C_2)) \tag{1}$$

In deze formule wordt voor twee componenten C_1 en C_2 een som berekend. Die som bestaat enerzijds uit het inwendig verschil van de component, $\iota(C)$. In formule 2 zie je dat dit bepaald wordt door het grootste gewicht van een boog waarvan beide knopen in dezelfde component zitten. Een component met slechts 1 knoop heeft inwendig verschil nul.

$$\iota(C) = \begin{cases} 0, & \text{indien } |C| = 1 \\ \max_{e=\{u,v\} \in C} w(e), & \text{anders} \end{cases} \tag{2}$$

Voor beide componenten wordt bij hun respectievelijk inwendig verschil een toelatingsmarge opgeteld, $\tau(C)$. Formule 3 geeft aan dat die marge bepaald wordt door de verhouding van een constante k en het aantal knopen die reeds tot het component behoren, $|C|$. **In je taak mag je de constante k gelijkstellen aan 150.**

$$\tau(C) = k/|C| \tag{3}$$

Gegeven een gewogen graaf $G = (V, E)$ met n knopen, m bogen en wegingsfunctie $w : E \mapsto \mathbb{R}$ zal het algoritme als volgt te werk gaan:

1. Begin met een segmentatie $S = (C_1, \ldots, C_n)$ waarbij iedere knoop $v_i \in V$ in zijn eigen segment C_i zit.

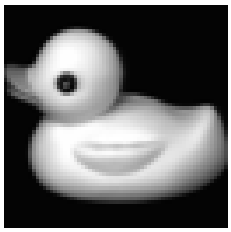
2. Sorteer de bogen van G volgens gewicht van klein naar groot.

3. Itereer over de m gesorteerde bogen $e \in E$ en voer het volgende uit: Beschouw de knopen u en v van e. Voeg de segmenten van u en v, C_u en C_v, samen indien beide knopen nog tot verschillende segmenten behoren, én het gewicht van de boog kleiner is dan het minimale inwendig verschil tussen beide componenten, m.a.w.: $C_u \neq C_v$ en $w(e) \leq MInt(C_u, C_v)$.

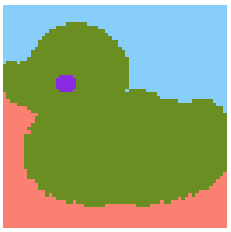
Je procedure `segment` dient als eindresultaat een cons-cel (pair) terug te geven die in de `car` het aantal segmenten weergeeft en in de `cdr` een node-indexed vector die per knoop aangeeft tot welk segment het behoort. Stel dat een 3×3 afbeelding gesegmenteerd zou zijn per rij, dan zou de output er als volgt moeten uitzien:
`(3 . #(1 1 1 2 2 2 3 3 3))`.

3 Testen en beoordeling

Om je implementatie te testen, gebruik je het bestand `main.rkt`, samen met de bijgeleverde modelfiguren in de folder `examples`. De figuren zijn een selectie uit de COIL-20 dataset [2] die gehalveerd werden in resolutie om de rekentijd te beperken. Het bestand bevat procedures om deze figuren correct in te lezen en om te zetten tot gewogen grafen. Het bestand bevat eveneens een procedure om het resultaat van de segmentatie te visualiseren. Figuur 2 geeft een voorbeeld van zo een visualisatie. Merk op dat je code ook getest zal worden met andere figuren, dus het is belangrijk dat je jouw oplossing grondig test. Bovendien wordt je code niet alleen beoordeeld op functionaliteit en performantie, maar ook op het gebruik van de juiste abstracties. Het documenteren van je code is verplicht en maakt integraal deel uit van de evaluatie.



(a) De originele afbeelding



(b) De resulterende segmentatie

Figuur 2: Visulatie van een segmentatie van een monochrome afbeelding volgens het algoritme beschreven in sectie 2 met constante $k = 150$. De pixels werden ingekleurd op basis van hun segment.

4 Oplevering

Upload je oplossing naar Canvas ten laatste op de aangegeven deadline die voor jou van toepassing is in de daarvoor voorziene ruimte in de 'Opdrachten' rubriek:

- **Deadline voor de Dagstudenten**: **zondag 30 april 2023 om 23u59**.
- **Deadline voor de Werkstudenten**: **maandag 29 mei 2023 om 08u00**.

Je oplossing zit in een zip-bestand met je naam en studentennummer, bijv. `Youri-Coppens-123456.zip`. Dit bestand bevat één enkele map met identieke naam, bijv. `Youri-Coppens-123456` met als inhoud:

1. Een `README.txt` bestand waarin je een **korte** beschrijving geeft van je oplossing. Je beschrijft en motiveert de keuzes die je gemaakt hebt om tot je oplossing te komen. Je licht ook kort toe welke bestanden van de `a-d` folder je gebruikt en/of gewijzigd hebt.
2. De folder `examples` waarin alle voorbeeldfiguren verzameld zijn.
3. De map `a-d` die de code bevat. Deze map bevat alle code die nodig is om het testbestand uit te voeren, inclusief alle gewijzigde bestanden. ADTs die niet nodig zijn voor deze taak mag je weglaten uit deze folder om de grootte van je inzending te beperken. Zorg er wel voor dat alle nodige ADTs om je code te runnen aanwezig zijn! Dit kan je best apart testen alvorens je code op te sturen. **Het moet namelijk mogelijk zijn om je code te runnen indien enkel en alleen deze `a-d` folder geïnstalleerd is via de DrRacket Package Manager**.
4. Het bestand `a-d/segmentation.rkt`, waarin het algoritme wordt geïmplementeerd volgens de instructies uit sectie 2.
5. Het bestand `main.rkt` waarin het algoritme wordt toegepast op alle voorbeeldfiguren.

5 Opmerkingen

- De API-beschrijving moet exact gevolgd worden: behoud de gevraagde procedurenamen alsook de proceduretyes. M.a.w. zorg dat deze het juiste aantal parameters hebben, in de juiste volgorde, en dat iedere procedure de gewenste output produceert. De beschrijvingen bevinden zich in de meegeleverde codebestanden.
- De deadline is strikt na te leven: taken die te laat worden ingediend, zullen niet verbeterd worden en als afwezig worden gequoteerd.
- Indien je vragen hebt over de taak, neem je tijdig contact op met de assistent.

De taak dient strikt individueel en op eigen kracht gemaakt te worden. Dat betekent dat je jouw taak op een zelfstandige basis moet maken en je jouw werk moet kunnen uitleggen, onder toezicht moet kunnen herimplementeren, en je werkwijze moet kunnen verdedigen. Werk (code, tekst, etc.) overnemen van of delen met derden (bv. medestudenten, websites, GitHub, etc.) is verboden. Elektronische hulpmiddelen worden gebruikt om alle inzendingen met online bronnen en met elkaar te vergelijken, zelfs over verschillende academiejaren heen.

De enige code die mag overgenomen worden, is de code gegeven in deze cursus: zowel de gegeven cursuscode op Canvas als de oplossingen van de WPO's van dit vak. **Indien je cursuscode of oplossingen van het WPO hergebruikt, vermeld je dit op zijn minst in de README.**

Elke handeling van een student die afwijkt van de gegeven instructies en niet in overeenstemming is met het examenreglement wordt beschouwd als onregelmatigheid. Plagiaat is eveneens een onregelmatigheid. Onder plagiaat wordt begrepen het gebruik maken van andermans werk, al dan niet in bewerkte vorm, zonder nauwkeurige bronvermelding (cf. OER[1], Artikel 118§2). Plagiaat kan betrekking hebben op verschillende vormen van werk zoals tekst, code, beeld, etc.

Elk vermoeden van plagiaat zal onverwijld aan de decaan van de faculteit worden gerapporteerd. Zowel gebruiker als verlener van zulke code zullen worden gerapporteerd en zullen worden behandeld volgens de plagiaatregels van het examenreglement (cf. OER, Artikel 118). De decaan kan beslissen tot (een combinatie van) de examentuchtsancties, gaande van 0/20 op het werkstuk tot een verbod tot (her)inschrijving voor één of meerdere academiejaren (cf. OER, Artikel 118§5).

Contacteer ons als je twijfelt of iets al dan niet als plagiaat zou beschouwd worden.

Referenties

[1] Pedro F. Felzenszwalb en Daniel P. Huttenlocher. "Efficient Graph-Based Image Segmentation". In: *International Journal of Computer Vision* 59 (sep 2004), p. 167–181. DOI: 10.1023/B:VISI.0000022288.19776.77.

[2] Sameer A Nene, Shree K Nayar en Hiroshi Murase. *Columbia Object Image Library (COIL-20)*. Tech. rap. CUCS-006-96. New York, NY: Columbia University, feb 1996. URL: https://www1.cs.columbia.edu/CAVE/publications/pdfs/Nene_TR96.pdf.

[1] Onderwijs- en Examenreglement