Rezumat pentru lucrarea de licență
“Tehnici de vectorizare a imaginilor de tip raster”

Motivația

Într-o zi căutam un cadou pentru o prietenă prin zona Pieței Unirii, iar unul dintre lucrurile care mi-a atras atenția au fost tricourile cu texte sau desene tipărite pe ele. Trecând prin toate modele pe care le aveau nu am găsit ceva destul de potrivit. Am întrebat vânzătoarea dacă poate să facă tricouri personalizate, ea spunând că pozele trebuie să fie în format vectorial.

Din curiozitate am căutat pe internet dacă exista tool-uri sau aplicații care pot transforma o imagine raster într-una vectorială. În momentul în care am scris licența, pe internet am putut găsi doar aplicații care: permiteau doar vizualizarea imaginilor vectorizate(nu și salvarea lor), convertirea avea doar trei opțiuni de rafinitate (low,medium,high), aveau un timp de procesare foarte mare sau costurile lor începeau de la 295 $ până la 1500 $. Niciuna dintre aplicații sau tool-uri era freeware, toate fiind ori shareware ori trebuia plătită o sumă foarte mare de bani.

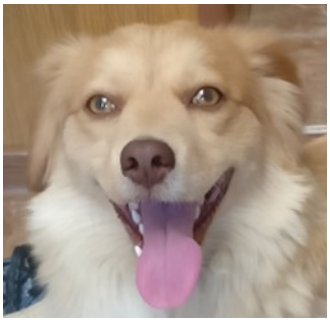
Văzând toate aceste implementări diferite oferite la prețuri relativ mari, am decis să creez câteva implementări pentru vectorizarea unei imagini, și să public codul sursă pe GitHub.

Diferența dintre o imagine vectorială și una raster

Imaginile vectoriale sunt formate din primitive geometrice (puncte, segmente de dreaptă , curbe , poligoane etc.). Imaginile raster sunt formate din pixeli. Un pixel este un singur punct sau cel mai mic element care poate fi afișat pe ecranul device-ului. În comparație cu imaginile bazate pe pixeli , cele bazate pe primitive geometrice pot fi mărite oricât fără pierderea calității.

Despre implementarea lucrarii

În proiectul meu am creat trei clase de vectorizare care extind clasa BaseVectorizer. Clasa aceasta a fost creată pentru a ușura utilizarea codului în restul aplicației, astfel reducând cantitatea de cod care trebuie scrisă și lăsând posibilitatea de a integra cu ușurință vectorizatoarele în codul pentru interfața grafică. Fiecare dintre vectorizatoare permite o reglarea a gradului de similaritate pentru a lăsa mai multă libertate utilizatorului.



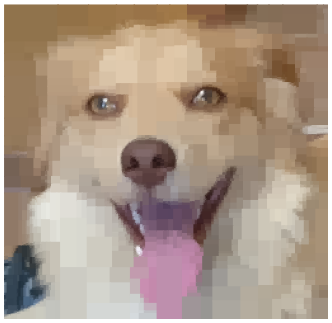
Exemplu de imagine raster

RectangleVectorizer

Acest vectorizator creează o listă de dreptunghiuri care acoperă întreaga suprafață a pozei originale. Fiecare dintre dreptunghiuri are o culoare care este calculată făcând media culorilor pixelilor care aparțin dreptunghiului respectiv. Procesul de divizare în dreptunghiuri este recursiv, încercând prima dată un dreptunghi care ocupă toată suprafața imaginii, și dacă acesta nu este considerat bun

(conține cel puțin doi pixeli care sunt prea diferiți din punct de vedere al culorilor lor) el va fi fragmentat în 4 dreptunghiuri și procesul continuă de acolo.

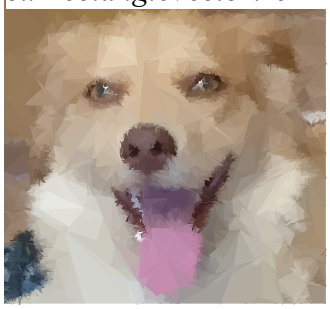
Pentru a accelera procesarea dreptunghiurilor, după prima divizare cele patru dreptunghiuri generate vor fi mai departe procesate de câte un thread separat. Fiecare dintre thread-uri va continua să creeze lista ei proprie de dreptunghiuri. Execuția se termină când toate thread-urile nu mai pot diviza dreptunghiurile lor.



Exemplu de procesare cu RectangleVectorizer

TriangleVectorizer

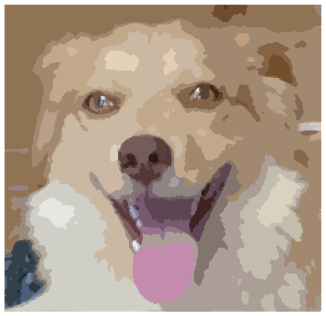
Acest vectorizator este foarte similar cu RectangleVectorizer prin simplul fapt că în loc să divizeze în patru dreptunghiuri, acesta la fiecare pas recursiv creează două triunghiuri. Atunci când am scris lucrarea, vectorizatorul acesta folosea doar două thread-uri de execuție pentru accelerare. Acestea sunt asignate primelor două triunghiuri dreptunghice formate prin tăierea imaginii dreptunghice dea-lungul diagonalei.



Exemplu de procesare cu TriangleVectorizer

PolygonVectorizer

Acest vectorizator este diferit față de celelalte două prin faptul că nu am putut să folosesc mai multe thread-uri de execuție. El generează o listă de poligoane care acoperă întreaga suprafață a pozei originale. Procesul de divizare în poligoane constă în căutarea zonelor de culori care sunt similare și continue. Mulțimea de poligoane descoperită nu este cea mai eficientă deoarece am ales să folosesc metoda Greedy pentru descoperirea lor.



Exemplu de procesare cu PolygonVectorizer

Din momentul în care se descoperă un pixel care nu a fost încă asociat niciunui poligon, din acel punct se vor căuta toți ceilalți pixeli vecini care au culori suficient de similare. Când nu mai găsește pixeli învecinați se calculează în timp liniar perimetrul poligonului și apoi este adăugat la listă.

Concluzie

Aplicația a fost un succes parțial din punctul meu de vedere, ea oferind o viteză de procesare relativ mare față de majoritatea aplicațiilor pe care le-am găsit și mai multă libertate în selectarea gradului de rafinitate dar, din punct de vedere calitativ este mai slab decât majoritatea.

Imaginile vectoriale sunt exportate în formatul SVG și SVGZ (care este de fapt SVG comprimat cu zip). Formatele respective pot fi vizualizate cu majoritatea browser-elor de pe calculatoare și telefoane și de asemenea pot fi editate folosind diverse tool-uri precum InkScape.

Semnătură