Gray Level Histogram of a Gray-Scale Image

autor: Martinescu Sorin-Alexandru

Implementarea aplicatiei

Implementarea aplicatiei vine cu interfata grafica, pentru o mai usoara interactiune utilizator - proces de prelucare imagine.

Interfata grafica implica 3 butoane principale: ‘Select image’, ce are rolul de a selecta imaginea pe care dorim sa o filtram; daca nu selectam nici-o imagine si dam RUN, se va afisa o casuta (form) pop-up cu un mesaj ce ne atentioneaza ca nu am selectat nici o imagine.

Al doilea buton va preciza pathul unde vom salva histogramele si imaginile rezultate in urma proceselor de prelucrare; daca nu se alege path, se va folosi pathul setat default ‘C:\\Users\\Public\\Pictures’. Ultimul buton reprezinta butonul de RUN, si acesta da start proceselor de prelucrare imagine.

Implementarea algoritmi

Cand se selecteaza o imagine, aceasta este afisata in interfata grafica, si la finalul rularii procesarilor de imagine, se va afisa imaginea modificata si niste ferestre in care vor fi ilustrate histogramele pentru Gray scale si pentru EQ.

Algoritmele de prelucrare aplicate sunt urmatoarele: mai intai se aplica Gaussian Noise filter, pas precedat de GrayScale ce face conversia in gray a imaginii filtrate, iar la final pentru a obtine niste valori satisfacatoare pentru histogramele noastre se aplica Equalize histogram, ce este o tehnica pentru ajustarea intensitatii imaginii pentru ‘enhance contrast’.

Algoritmi / biblioteci

Codul apelat in main e formata din componente impartite in 3 directoare: histoLib, imgProc, lib. In ‘lib’, se afla bilblioteci java importate necesare reprezentarii histogramei.

In histoLib se afla: interfata histogram din care se mosteneste clasa abstracta ImplementHistogram, din care se mosteneste calasa abstracta AbstractHistoDraw din care se mosteneste clasa FinalHisto.

Din FinalHistose mosteneste ShowHistoAndData. In imgProc se afla: clasele: ConvertToGS, EQ, GaussianNoise, ce implementeaza interfata ImgProc.

Performante

Performantele aplicatiei sunt ‘satisfacatoare’, cu precizarea ca imaginea maxima recomandata pentru prelucrare trebuie sa fie sub 20MB. Precizez ca am realizat teste si cu samples ce depaseau 25 MB si uneori rula, uneori alicatia nu putea procesa din cauza depasirii de memorie din heap, deci, in concluzie, a se evita depasirea dimensiunii de 20 MB. Mai jos voi preciza datele de la rularea unui sample de aproximativ 20MB:

timp de procesare

Read time: 1308 ms
Filter time: 2193 ms
GrayScale time: 1479 ms
EQ time: 861 ms
Histogram EQ time: 3890 ms
Histogram GS time: 2657 ms
ImgWrite EQ time: 801 ms
ImgWrite GS time: 1199 ms
ImgWrite Filter time: 1279 ms
ImgWrite Initial time: 926 ms

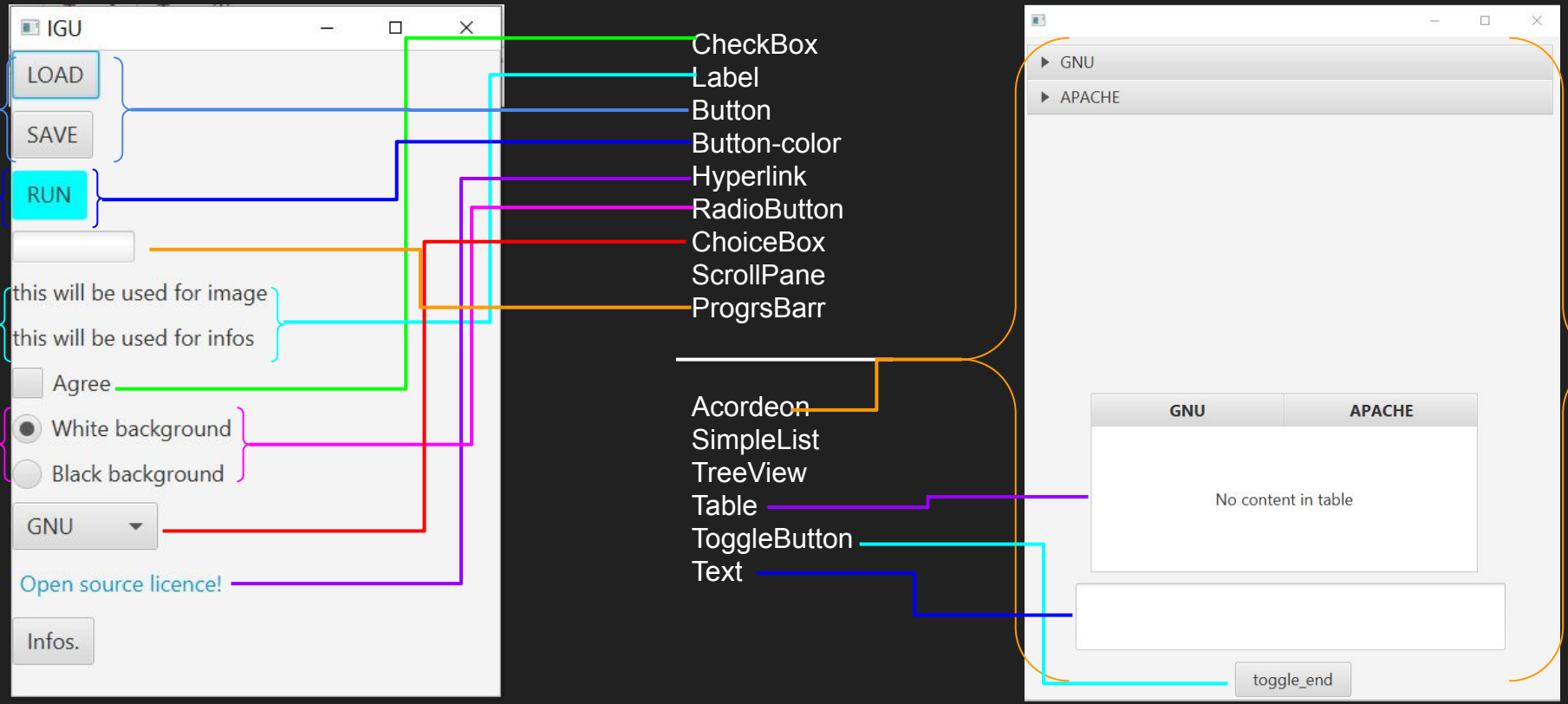
Concluzii procesare imagine

Din rulare se poate observa ca cea mai costisitoare operate este EQ, dar din histograma se trage concluzia ca: acel timp pierdut pe EQ este castigat deoarece histograma rezultata dupa EQ este mult mai clara si mai usor de interpretat.

Concluzii:

Implementarea fara EQ nu conduce la rezultate la fel de satisfacatoare, astfel efectul pozitiv adus la interpretarea datelor va amortiza timpul lung de procesare. Se observa ca dupa Gaussian filter, imaginea ar avea ‘purici’, dar acesta este un efect benefic in obtinerea unei histograme usor de interpretat.

UI / UX


IGU
LOAD
SAVE
RUN
this will be used for image
this will be used for infos
Agree
White background
Black background
GNU
Open source licence!
Infos.
CheckBox
Label
Button
Button-color
Hyperlink
RadioButton
ChoiceBox
ScrollPane
ProgrsBarr
Acordeon
SimpleList
TreeView
Table
ToggleButton
Text
GNU
APACHE
GNU
APACHE
No content in table
toggle_end

Demo

Pe langa documentatie, exista doua fisiere .gif (demourile), ce prezinta:

1. demo1.gif (un review rapid al codului)
2. demo2.gif (un review rapid al functionarii aplicatiei)

Multumesc!