Logo, icon

Description automatically generated

**Image Enhancement Application**

Student: Zilai Denis

Grupa: 30238

Procesare de Imagini

Contents

[1. Definirea temei 2](#_Toc103960331)

[2. Algoritmii principali utilizati 2](#_Toc103960332)

[3. Rezultate intermediare 2](#_Toc103960333)

[4. Test de evaluare/performanta 2](#_Toc103960334)

[5. Rezultatele finale 2](#_Toc103960335)

[6. Challenge-uri intampinate 2](#_Toc103960336)

# Definirea temei

Prezenta tema isi doreste a indeplini un mare scop principal, mai exact acesta fiind imbunatatirea calitatii imaginilor (HDR – High Dynamic Range), precum si manipularea anumitor proprietati ale acestor imagini:

* Alpha – contrast
* Beta – luminozitate
* Gamma – optimizarea utilizarii bitilor, partile intunecate ale imaginii -> mai luminate, iar partile luminoase ale imaginii -> mai intunecate

In materie de interfata si utilizare, la startul aplicatiei se poate observa aparitia a 3 imagini, acestea fiind rezultatele celor 3 algoritmi HDR utilizati pe o imagine specifica, precum si 2 ferestre care permit manipularea atributelor mai sus mentionate, prin intermediul unor trackbar-uri. Aceste atribute pot fi schimbate, pe baza imaginii rezultate in urma utilizarii algoritmului „Drago” HDR, insa se pot aplica oricarei alte imagini.

# Algoritmii principali utilizati

Pentru realizarea acestei aplicatii s-au utilizat 3 principali algoritmi pentru asa-numitul „Tone mapping”, care nu este altceva decat conversia unei imagini HDR, intr-o imagine cu 8 biti pe fiecare canal (imagine de 24 biti), cu scopul de a produce imagini de o calitate ridicata.

* Drago

Are ca si parametri urmatoarele:

createTonemapDrago

(

float gamma = 1.0f,

float saturation = 1.0f,

float bias = 0.85f

), unde bias este functia cu valori intre [0, 1].

* Reinhard

Are ca si parametri urmatoarele:

createTonemapReinhard

(

float gamma = 1.0f,

float intensity = 0.0f,

float light\_adapt = 1.0f,

float color\_adapt = 0.0f

), unde intensity este intre [-8,8]. Valori mai ridicate produc rezultate mai luminoase; light-adapt controleaza adaptarea luminii si are valori intre [0,1]. O valoare de 1 indica adaptare bazata doar pe o valoare pixel, iar 0 indica o adaptare globala; color\_adapt controleaza adaptarea cromatica si se afla in intervalul [0,1]. Canalele sunt tratate independent daca valoarea este setata cu 1, iar nivelul de adaptare este acelasi pentru fiecare canal daca valoarea este setata cu 0.

* Mantiuk

Are ca si parametri urmatoarele:

createTonemapMantiuk

(

float gamma = 1.0f,

float scale = 0.7f,

float saturation = 1.0f

), unde parametrul scale este factorul pentru scalare al contrastului. Sunt recomandate valori intre 0.6 si 0.9 pentru cele mai bune rezultate.

# Rezultate intermediare

Graphical user interface, application, Teams

Description automatically generatedRezultate tone mapping:

Graphical user interface, website

Description automatically generatedRezultate ajustari contrast si luminozitate:

Graphical user interface, website

Description automatically generatedRezultate corectie gamma:

# Test de evaluare/performanta

Pentru rezultate nu am utilizat un sablon pentru testare, incat nu era intocmai necesar. Rezultatele se pot observa cu ochiul liber, dar totusi am utilizat diferiti parametri pentru a testa algoritmii de tone mapping, alegandu-i pe cei care produc cel mai realist si fidel rezultat (a se consulta punctul 2). De asemenea, am utilizat debugger-ul oferit de IDE-ul PyCharm, pentru a verifica daca rezultatele sunt corecte.

# Rezultatele finale



Rezultatele finale au fost cele asteptate si descrise la punctul 1, in afara de o singura problema, aceasta fiind legata de trackbar-ul destinat modificarii luminozitatii imaginii. Rezultatul asteptat la miscarea cursorului pentru luminozitate la stanga, ar fi fost ca imaginea sa fie complet neiluminata, insa apare un fenomen ciudat, distorsionand culorile imaginii. In afara de aceasta problema, toate cerintele de la punctul 1 sunt indeplinite cu succes.

# Challenge-uri întâmpinate

Acest proiect s-a dovedit a fi unul foarte interesant, permitandu-ti sa manipulezi imaginile cu diverse scopuri. Am decis sa scriu acest proiect in limbajul Python, tocmai pentru a usura scrierea si intelegerea per total a codului, acesta putand accesa de asemenea libraria OpenCV. Printre singurele probleme intampinate pe parcursul acestui proiect, merita mentionata urmatoarea: la afisarea celor doua ferestre pentru manipularea atributelor, precum si rezultatul celor 3 algoritmi utilizati pentru tone mapping, imaginile finale au avut dimensiuni mult prea mari, neincapand pe ecran. Astfel, a fost nevoie de implementarea unei metode pentru rescalarea imaginilor, pentru a incapea pe ecran si pentru a putea realiza operatii. De asemenea, o alta problema intampinata a fost instalarea librariilor necesare (OpenCV, numpy).

In concluzie, aceste probleme au fost depasite si s-a putut ajunge la un rezultat satisfacator.

# Bibliografie

1. <https://docs.opencv.org/3.4/d3/dc1/tutorial_basic_linear_transform.html>
2. <https://www.geeksforgeeks.org/python-opencv-cv2-imshow-method/>
3. <https://www.tutorialkart.com/opencv/python/opencv-python-resize-image/>
4. <https://en.wikipedia.org/wiki/High_dynamic_range>
5. <http://qtpfsgui.sourceforge.net/help/en/tmap_ref_drago.html>
6. https://www.cl.cam.ac.uk/~rkm38/pdfs/mantiuk08datm.pdf