A close up of a sign

Description generated with very high confidence 

**Extragerea conturului din imagini BMP**

**Master MPI**

**ACS 2018**

**Disciplina: Biometrie**

**Student: Vasile Razvan Ioan**

Contents

[**1. Introducere 3**](#_Toc503487553)

[**1.1 Obiectivul proiectului 3**](#_Toc503487554)

[**1.2 Domeniul studiat 3**](#_Toc503487555)

[**2. Considerente teoretice 3**](#_Toc503487556)

[**2.1 Canny Edge Detection 3**](#_Toc503487557)

[**2.1.1 Reducerea zgomotului din imaginea sursa 3**](#_Toc503487558)

[**2.1.2 Determinarea Gradientului Intensitate din imagine 4**](#_Toc503487559)

[**2.1.3 Suprimarea non-maximala 5**](#_Toc503487560)

[**2.1.4 Setarea pragului de histerezis 5**](#_Toc503487561)

[**2.1.5 Implementarea CannyEdge din bibilioteca Python OpenCV 6**](#_Toc503487562)

[**3. Descrierea aplicatiei 7**](#_Toc503487563)

[**3.1.1 Tehnologii utilizate 7**](#_Toc503487564)

[**3.1.2 Arhitectura aplicatiei 8**](#_Toc503487565)

[**3.1.3 Fluxul aplicatiei 10**](#_Toc503487566)

[**4. Concluzii 12**](#_Toc503487567)

[**5. Posibile imbunatatiri 13**](#_Toc503487568)

[**6. Bibliografie 13**](#_Toc503487569)

[**Figure 1 Ecuatia filtrului Gaussian 2D 4**](#_Toc503487274)

[**Figure 2 DIstributia filtrului Gaussian 2D 4**](#_Toc503487275)

[**Figure 3 Determinare Gradient si directie 5**](#_Toc503487276)

**[Figure 4 Suprimarea non-maximala 5](#_Toc503487277)**

**[Figure 5 Pragul de histerezis 6](#_Toc503487278)**

[**Figure 6 EdgeDetection Example 6**](#_Toc503487279)

[**Figure 7 Fluxul aplicatiei 7**](#_Toc503487280)

[**Figure 8 EdgeDetectGUI 8**](#_Toc503487281)

[**Figure 9 Image\_Edge\_Extraction 9**](#_Toc503487282)

[**Figure 10 Interfata START 10**](#_Toc503487283)

[**Figure 11 Consola Python 10**](#_Toc503487284)

[**Figure 12 Selectare sursa 11**](#_Toc503487285)

[**Figure 13 Informare 11**](#_Toc503487286)

[**Figure 14 Selectare destinatie 11**](#_Toc503487287)

[**Figure 15 Output 12**](#_Toc503487288)

[**Figure 16 LogFile 12**](#_Toc503487289)

# Introducere

## Obiectivul proiectului

**Dezvoltarea** unui sistem capabil sa converteasca o imagine sau un set de imagini oferite ca input de catre un utilizator din formatul sau in format .bmp. Ulterior acestei conversii, imaginile .bmp sunt procesate si prelucrate pentru a atinge scopul fundamental al aplicatiei: extragerea conturului dintr-o imagine data ca input.

## Domeniul studiat

**Imaginea digitală** este reprezentarea vizuală a unui obiect, a unei persoane sau a diverse peisaje aparținând mediului înconjurător, prin intermediul unui dispozitiv de tip optic – camera. Din punct de vedere structural, imaginea digitală este o matrice formată din puncte denumite pixeli.

Conceptul de achizitie si prelucrare a imaginii a aparut pentru prima oara in domeniile de specialitate odata cu lansarea primului dispozitiv capabil de a captura o imagine pentru ca ulterior sa o serveasca ca si output. “Camera obscura” a fost inventata in secolul XVI, fiind primul dispozitiv care functioneaza pe acelasi principiu ca si ochiul omenesc.

Evolutia domeniului IT a condus la aparitia unor gadget-uri care in trecut se puteau vedea doar in domeniul cinematografic, mai exact in filmele Science-Fiction. Camera Web, dar si camerele DSLR fac parte din aceasta categorie.

# Considerente teoretice

## Canny Edge Detection

**Canny Edge Detection** reprezinta probabil cel mai popular algoritm folosit pentru detectarea contururilor dintr-o imagine. Acest algoritm a fost dezvoltat de John Canny in 1986.

Complexitatea algoritmului consta in faptul ca pentru a ajunge la rezultatul dorit – extragerea conturului din imagine, trebuie urmata o serie de pasi bine determinati. Astfel se aplica multiple transformari imaginii sursa, in scopul determinarii imaginii destinatie.

Pasii urmati in construirea algoritmilor vor fi prezentati pe rand, in acest capitol.

### Reducerea zgomotului din imaginea sursa

Din cauza faptului ca extragerea conturului este susceptibila de crearea zgomotelor in imaginea rezultata, primul pas in procesul de extragere va fi reducerea, pe cat posibil a zgomotului folosind un filtru 5x5 Gaussian.

**Filtrul Gaussian de netezire (‘smoothing’)** este un operator de convolutie 2D, utilizat pentru blurarea imaginilor in vederea eliminarii detaliilor si zgomotului. In acest sens, filtrul Gaussian se aseamana cu **filtrul median**, insa utilizeaza un kernel diferit, sub forma unui clopot Gaussian.

Functia matematica definitorie pentru **filtrul Gaussian 2D** este izotropa. Forma sa, cat si distributia Gaussiana se pot regasi in figurile de mai jos.

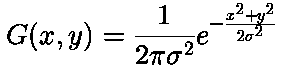
****

Figure 1 Ecuatia filtrului Gaussian 2D

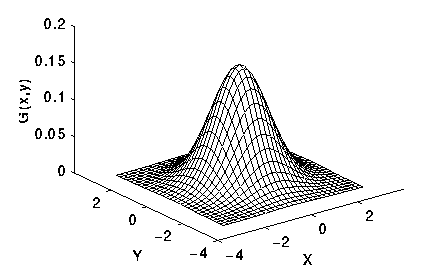


Figure 2 DIstributia filtrului Gaussian 2D

### Determinarea Gradientului Intensitate din imagine

Imaginea rezultata in urma operatiei de netezire Gaussiana este apoi filtrata cu un kernel Sobel, atat pe directie orizontala, cat si vertical. Astfel, se va obtine prima derivata pe directie orizontala (Gx) si directie verticala (Gy).

Din aceste doua imagini, se poate determina gradientul conturului, precum si o directie pentru fiecare pixel in parte, folosind urmatoarele formule matematice:

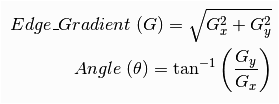


Figure 3 Determinare Gradient si directie

### Suprimarea non-maximala

Dupa obtinerea gradientului si a directiei de gradient, se efectueaza o procesare complete a imaginii, pentru a se elimina pixelii nedoriti. Pentru aceasta operatie, luand fiecare pixel in parte, acesta este verificat daca reprezinta un maxim local in vecinatatea sa si in directia gradientului.

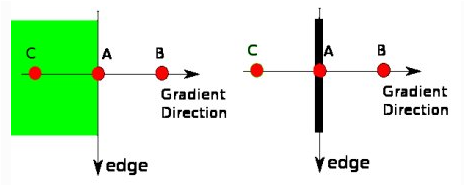


Figure 4 Suprimarea non-maximala

Direcția de înclinare este normală la margine. Punctele B și C sunt în direcții de gradient. Deci, punctul A este verificat cu punctele B și C pentru a vedea dacă acesta formează un maxim local. Dacă da, este considerată pentru etapa următoare, altfel este suprimată (pusă la zero). Astfel, **rezultatul obținut este o imagine binară cu "marginile subțiri".**

### Setarea pragului de histerezis

In aceasta etapa se determina care dintre mariginile determinate la pasul anterior sunt intr-adevar margini. Pentru ca aceasta operatiune sa fie posibila, este necesara definirea a doua valori de prag, **minVal** si **maxVal.**

Oricare dintre muchiile cu un gradient de intensitate mai mare decat **maxVal** sunt cu siguranta muchii ale conturului, iar cele mai mici de **minVal** nu reprezinta muchii, deci nu vor fi luate in considerare in rezultatul final.

Muchiile care se afla intre aceste doua praguri sunt muchii clasificate pe baza conectivitatii lor. Daca sunt conectate la pixeli ‘singuri’, acestea sunt considerate margini, in caz contrar nefiind luate in considerare.

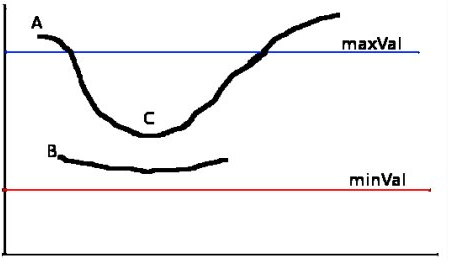


Figure 5 Pragul de histerezis

In acest pas se va mai realiza o eliminare suplimentara a pixelilor ‘zgomot’ intrucat muchiile vor fi considerate la final ca linii continue.

### Implementarea CannyEdge din bibilioteca Python OpenCV

**Bibilioteca Python OpenCV** inglobeaza toti pasii definiti anterior intr-o singura functie: **cv2.Canny().**

Functia **cv2.Canny():**

1. Primul argument: imaginea de intrare
2. Al doilea/treilea argument: minVal / maxVal
3. Al patrulea argument: aperture\_size – marimea kernelului Sobel pentru a gasi gradient de imagine (default = 3)

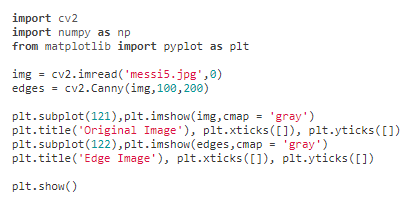


Figure 6 EdgeDetection Example

# Descrierea aplicatiei

Aplicatia construita dispune de 3 functionalitati fundamentale:

* Convertirea imaginilor din formate .PNG sau .JPG in format .BMP 24bit
* Extragerea conturului din imaginea .BMP 24 bit
* Afisarea pe ecran a rezultatelor, precum si salvarea acestora intr-un fisier destinatie, ales de utilizator

Solutia propusa se bazeaza pe sintaxa si functionalitatile limbajului de scripting Python, precum si pe cea a bibliotecilor inglobate in acesta – precum Tkinter (Interfata Grafica) sau OpenCV (Prelucrare&Procesare imagine)

Pentru a fi user-friendly, aplicatia dispune de o interfata grafica prin care utilizatorul poate interactiona cu sistemul, oferind posibilitatea acestuia atat de selectie a directoarelor de lucru cu imaginile (sursa <-> destinatie), cat si posibilitatea de a salva si vizualiza datele de iesire.

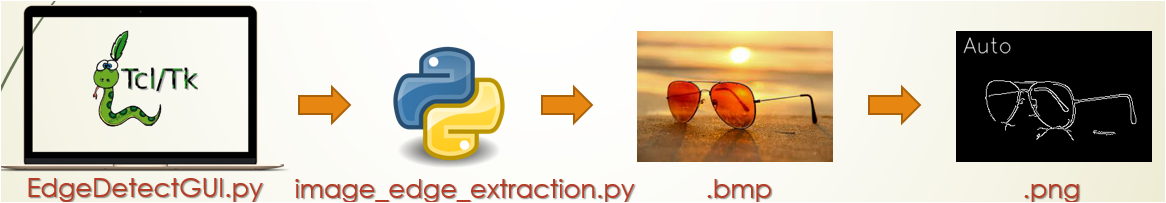


Figure 7 Fluxul aplicatiei

## Tehnologii utilizate

Prezenta aplicatie este construita in limbajul Python. Python este un limbaj de scripting compatibil cu orice sistem de operare, fiind capabil sa ruleze codul sursa pe orice calculator.

**Python** reprezinta un mediu de programare facil pentru lucrul atat cu o arie vasta de tipuri de date: de la procesare si prelucrare de imagini / video, prelucrari audio, prelucrari text, pana la prelucrarea datelor stiintifice.

Utilizarea sa cu preponderenta in ultimul timp se datoreaza arhitecturii sale robuste – Python fiind capabil de furnizarea unui output intr-un timp scurt.

Prima varianta stabila a Python a fost lansata in anul 1994, iar fata de versiunile anterioare, aceasta versiune a dispus de capabilitati sporite. Printre noile functionalitati ale Python 94 se puteau regasi functii de filtrare, functii de definire a colectiilor/structurilor de date, dar si functii de reducere a zgomotului din datele de intrare.

**Tkinter** este biblioteca standard de creare a GUI (Graphical User Interface) din Python. Importarea si folosirea acestui modul faciliteaza interactiunile utilizatorilor cu aplicatia.

**OpenCV**: Bibliotecă de vedere artificială destinată şi optimizată pentru aplicaţiile în timp real, care permite încărcarea, salvarea şi prelucrarea unei imagini sau a unui set de imagini. De asemenea, **OpenCV**  dispune de un set de functii prin care conturul imaginii poate fi extras, rezultand astfel o noua imagine ( **cv2.Canny()** ).

## Arhitectura aplicatiei

**Interfata grafica cu utilizatorul: EdgeDetectGUI**

**A screenshot of a cell phone

Description generated with very high confidence**

Figure 8 EdgeDetectGUI

**Modulul de extragere a conturului: Image\_Edge\_Extraction**

**A screenshot of a cell phone

Description generated with high confidence**

Figure 9 Image\_Edge\_Extraction

## Fluxul aplicatiei

1. Prin accesarea EdgeDetectGUI.py, se va porni o fereastra cu panoul de START al aplicatiei.

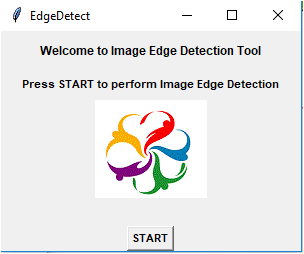


Figure 10 Interfata START

1. Apasarea butonului start conduce la deschiderea consolei de output Python in care este afisat progresul din cadrul fluxului aplicatiei

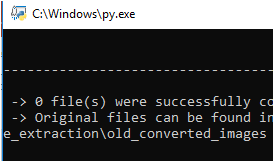


Figure 11 Consola Python

1. In paralel cu consola Python, se va deschide si un File Open Dialog, prin care utilizatorul selecteaza folderul cu imagini sursa

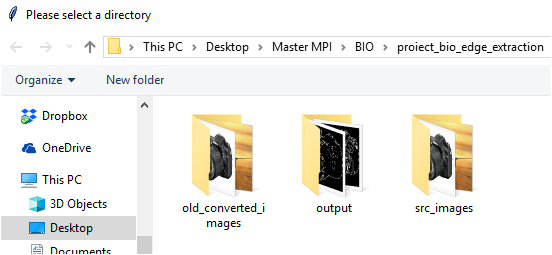


Figure 12 Selectare sursa

1. Scriptul Python parcurge fisierul pentru a gasi imagini de alt format decat .BMP. In cazul in care sunt prezente astfel de imagini, acestea sunt convertite automat la formatul .BMP 24 bit. La terminarea executiei convertirii, se afisaza un mesaj de informare.

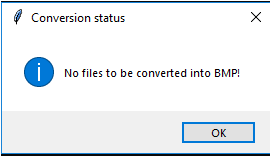


Figure 13 Informare

1. La apasarea butonului OK din pasul 4, va aparea o noua fereastra in care utilizatorul poate selecta directorul destinatie, in care rezultatele vor fi stocate sub forma de imagini .PNG. In aceasta fereastra, exista doua optiuni de selectare: adaugarea link-ului in form-ul present in fereastra (SEARCH) sau cautarea prin directoarele computerului (BROWSE).

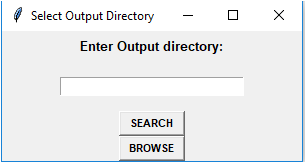


Figure 14 Selectare destinatie

1. In momentul selectarii folderului destinatie, imaginile rezultate prin cele trei metode de eliminare a conturului (wide, tight si auto) sunt salvate in acel fisier si ulterior sunt afisate pe ecran.

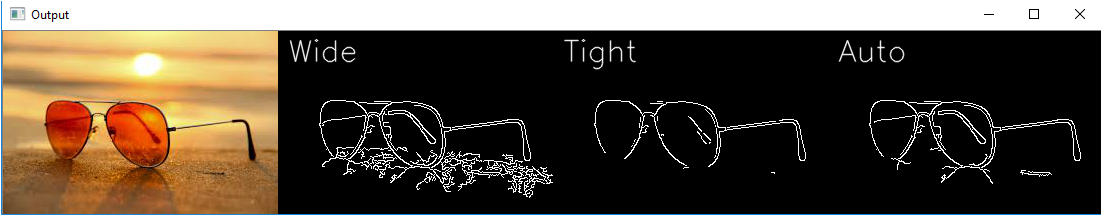


Figure 15 Output

1. Toate functiile sunt masurate ca si timp, iar acest lucru este scris in fisierul de log time.txt

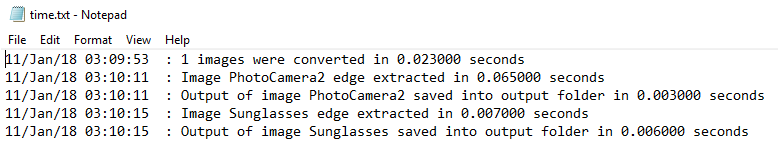


Figure 16 LogFile

# Concluzii

Aplicatia prezentata ofera intr-un timp de raspuns scurt output-ul dorit. Imaginile rezultate sunt salvate in folderul destinatie, selectat de catre utilizator.

De asemenea, aplicatia este capabila de extragere prin trei moduri diferite, oferind utilizatorului posibilitatea de a alege rezultatul care ii ofera calitatea dorita acestuia. Diferentele de calitate dintre rezultate se poate observa atat in interfata grafica, la afisarea rezultatului, cat si in fisierele generate de scriptul Python din folderul destinatie, selectat de asemenea de utilizator din interfata.

# Posibile imbunatatiri

* + **Interfață grafică pentru toate modulele aplicației**
  + **Dezvoltarea unui modul de comparare a calității imaginii pentru fiecare imagine de output**
  + **Dezvoltarea unui modul de editare a parametrilor prin care conturul este extras**
  + **Imbunatatirea modalitatii de convertire in .bmp**

# Bibliografie

* <https://wiki.python.org/moin/TkInter>
* <https://opencv.org/>
* <http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html>
* <https://en.wikipedia.org/wiki/Canny_edge_detector>
* <http://effbot.org/imagingbook/image.htm>
* <https://docs.python.org/3/library/tk.html>