# Piccaso project

Andrei-Cristian Danaila

Universitatea DUNAREA DE JOS DIN gALATI

Facultatea de Automatica Calculatoare Inginerie Electrica si Electronica

Specializarea:Automatica si informatica aplicata

Email:andreidanaila895@gmail.com

Dragos Iulian Bobes

Universitatea DUNAREA DE JOS DIN gALATI

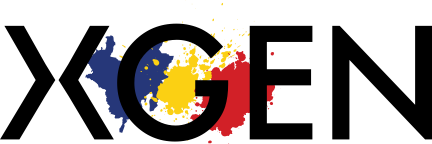
Facultatea de Automatica Calculatoare Inginerie Electrica si Electronica

Specializarea: Automatica si informatica aplicata

Email: dragosiulianbobes@gmail.com**Abstract**

The Piccaso project aims to create real-time drawings with an interactive graphical interface. Developed in Python 3.9.1, it utilizes OpenCV for image processing, Turtle for drawing, and NumPy for calculations. Images are read and processed to grayscale, then filtered with a Laplacian filter to enhance edge detection. Negative values are eliminated, and a threshold is applied to reduce pixel count for faster drawing. Contouring is divided into interior and exterior margins, optimizing complexity. Turtle library then renders the drawing based on processed coordinates. The program offers a user-friendly interface for selecting images stored on the PC.

**Keywords:** greafic interface,Python,Laplacian filter,threshold



## Introducere

Proiectul Picasso a fost realizat pentru a crea desene în timp real. Programul a fost dezvoltat cu dorința de a obține o interfață grafică interactivă, nu doar afișând anumite date, numere sau informații.

Proiectul realizează procesarea imaginii. Procesarea imaginii este folosită foarte des în zilele noastre în diverse domenii de activitate economică și nu numai.

Un bun exemplu unde întâlnim procesarea imaginii ar fi deblocarea facială folosită de tot mai multe smartphone-uri. Acestea folosesc camera frontală care scanează fața și analizează semnele distincte ale persoanei în cauză. [1]

Un al doilea exemplu de utilizare a procesării imaginii ar fi în domeniul automobilelor care au amplasate camere ce recunosc semnele de circulație, observă pericolele iminente și încearcă să le evite. [2]

Un exemplu de utilitate ridicată aș spune că ar fi procesarea imaginii în domeniul medical, unde se pot identifica tumori sau diferite malformații la nivelul organelor interne. [3]

**Prelucrarea imaginii**

În primul rând, trebuie să menționă că acest proiect este scris în limbajul interpretat Python, rulând pe versiunea 3.9.1. Bibliotecile utilizate pentru acest program sunt OpenCV, Turtle, NumPy,Tkinter și vom explica scopul fiecăreia în continuare. Biblioteca OpenCV-python este capabilă de multe lucruri, cum ar fi citirea și scrierea de imagini și videoclipuri, procesarea imaginilor, detectarea obiectelor și recunoașterea acestora . În acest proiect, biblioteca OpenCV-python a fost folosită pentru procesarea imaginilor. Biblioteca Turtle este folosită pentru a crea desene în timp real, dând impresia că desenul este realizat de o mână nevăzută, dar destul de precisă, urmând pași bine stabiliți de la început. De asemenea, am folosit biblioteca NumPy pentru a efectua anumite calcule matematice.

Biblioteca Tkinter este utilizată pentru a crea o interfață grafică din care am putea accesa orice fișiere de imagine descărcate sau existente pe calculatoarele noastre.

O imagine care conține text, Font, captură de ecran, număr

Descriere generată automat

Programul nostru funcționează în felul următor: primește o imagine în format PNG, JPG, JPEG, GIF sau WebP, dar trebuie să fie strict alb-negru (imaginea poate fi descărcată de pe internet).

Ca exemplu, vom folosi această imagine:

O imagine care conține schiță, vehicul, desen, Schiță

Descriere generată automat

După selectarea imaginii dorite, programul începe să lucreze. Folosește funcția cv2.imread pentru a citi imaginea și funcția cv2.flip pentru a corecta orientarea imaginii. Apoi, tot cu o funcție din biblioteca OpenCV, mai precis cv2.cvtColor, convertește imaginea la tonuri de gri pentru a facilita detectarea marginilor imaginii..

O imagine care conține schiță, Schiță, desen, diagramă

Descriere generată automat

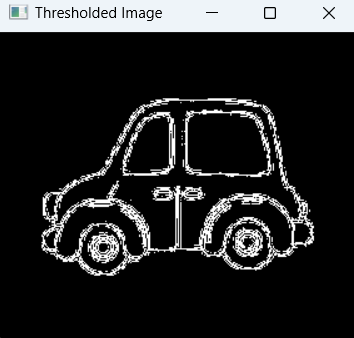
O imagine care conține vehicul, schiță, Vehicul de teren, mașină

Descriere generată automat

După efectuarea acestor operații, imaginea este trecută printr-un filtru Laplacian pentru a evidenția regiunile cu schimbări bruște în intensitatea luminii. Am folosit un filtru Laplacian deoarece aceste filtre calculează al doilea gradient al imaginii, ceea ce face ca regiunile cu schimbări abrupte în intensitate să iasă în evidență. După aplicarea filtrului Laplacian, vom obține valori negative și pozitive, ceea ce nu este foarte benefic pentru imaginea rezultată. Pentru a elimina valorile negative, vom folosi funcțiile NumPy care returnează valoarea absolută (np.absolute). Rezultatul obținut este introdus în interiorul funcției np.uint8, care returnează numere întregi fără semn pe 8 biți, ceea ce este necesar pentru a reprezenta o imagine validă.O imagine care conține text, vehicul, Vehicul de teren, schiță

Descriere generată automat

După toate aceste prelucrări de imagine, obținem o imagine al cărei contur este bine definit, dar vom avea foarte mulți pixeli, ceea ce presupune, de asemenea, foarte multe coordonate de urmărit de către biblioteca Turtle. Acest lucru va duce la un timp de desen foarte lung. Pentru a obține un timp favorabil de desen, va trebui să adăugăm un prag de 50 pe imaginea deja filtrată, ceea ce va seta toți pixelii cu o valoare mai mare de 50 la 255 (alb) și cei cu o intensitate mai mică de 50 vor fi setati pe negru



Pentru ca biblioteca Turtle să aibă continuitate, va trebui să împărțim desenul în margini interioare și exterioare: marginile exterioare sunt obținute prin funcția cv2.findContours, căreia îi vom da ca parametri: imaginea obținută după aplicarea pragului, funcția pentru găsirea contururilor exterioare, care va viza doar pixelii de la periferie, ignorându-i pe cei din interior, iar funcția cv2.CHAIN\_APPROX\_SIMPLE va stoca doar punctele de colț ale contururilor, ignorând punctele intermediare și reducând astfel complexitatea datelor. În același mod, se va proceda și pentru marginile interioare, doar că acum vom folosi funcția cv2.RETR\_LIST, care va lista toți pixelii indifferent de pozitia lor.

O imagine care conține vehicul, Vehicul de teren, roată, Piesă auto

Descriere generată automat

O imagine care conține text, vehicul, Vehicul de teren, proiectare

Descriere generată automat

**Analiza software**

O imagine care conține text, captură de ecran, Font

Descriere generată automat

După toate aceste prelucrări de imagine, biblioteca Turtle poate începe să îsi faca treaba. În primul rând, am inițializat viteza folosind turtle.speed, am ascuns pictograma pe care o foloseste librăria (adică aceea a unei broaște țestoase) pentru a folosi un efect de desenare mai natural folosind comanda turtle.hideturtle, am setat fundalul și culoarea desenului (am ales culoarea roșie), comenzile folosite fiind turtle.bgcolor și turtle.color. De asemenea, am impus o dimensiune maximă de 956 de pixeli în lățime și 720 în lungime. Pentru ca noua imagine sa fie parcursa de libraria Turtle, am folosit două structuri repetitive for pentru contururile exterioare și încă două pentru cele interioare. Biblioteca Turtle desenează practic pe baza coordonatelor pixelilor care trec prin filtrele aplicate anterior.

O imagine care conține text, captură de ecran

Descriere generată automat Am folosit și funcția select\_image pentru a crea o fereastră din care putem selecta orice imagine salvată în PC-ul nostru.O imagine care conține text, software, Software multimedia, captură de ecran

Descriere generată automat

## Concluzie

În concluzie, programul nostru realizează o prelucrare extinsă a imaginii, reușind să analizeze și să reproducă locația fiecărui pixel, realizând o redare destul de fidelă a imaginii originale. Un minus al programului ar fi că nu poate procesa desene complexe sau mari, ci doar desene de o mică complexitate.

O aplicație practică pentru această aplicație ar fi utilizarea ei de către disigneri de interior care ar putea vedea cu ușurința cum s-ar transpune anumite elemente într-un spatiu dat.

O altă utilitate ar fi folosirea aplicației în scop educativ în scolile primare pentru a ajuta copii să inteleagă intr-un mod interactiv si atractiv ce sunt coordonatele,la ce ne ajută ele si cum putem să le utilizam în viata de zi cu zi.

**Bibliografie**

|  |  |
| --- | --- |
| [1] | M. K. R. M. A. M. Mr. Guru Moorthy, "The Technology Behind Face Unlocking in Smartphones," *International Journal of Research Publication and Reviews,* vol. 4, no. 4, pp. 2070-2076, 2023. |
| [2] | K. Nair, "Image Processing in Automotive Industry," *International Journal of Advance Research in Science,Communication and Technology,* vol. 2, no. 3, 2022. |
| [3] | M. F. D. W. K. a. B.L.Trus, *Medical Image Processing,Analysis and Visualization in clinical research,* MD,USA, 2001. |
| [4] | "Opencv2.0 c reference.online," 2009. |
| [5] | C. Tomasi, "Bilateral filtering for gray and color image.on-line," 2009. |
| [6] | L. D. I. V. M. Barbu, Procesarea Datelor Utilizand Mediul Python.Lucrari Practice, Galati: Editura Fundatiei Universitare "Dunarea de Jos"din Galati, 2022. |
|  |  |
|  |  |