**Laborator 2. Instalare Open CV. Procesarea imaginilor**

Prelucrarea imaginilor este practic procesul care ne oferă realizarea caracteristicilor din imagini. Procesarea imaginilor se aplică atât pentru imagini, cât și pentru videoclipuri. Acestea sunt procedurile utilizate frecvent pentru a face ca instruirea să aibă mai mult succes în structurile de învățare profundă.

**Procesarea imaginii**

Prelucrarea imaginii începe cu recunoașterea datelor de către computere. În primul rând, se creează o matrice pentru date în format imagine. Fiecare valoare a pixelilor din imagine este procesată în această matrice. De exemplu, se creează o matrice de dimensiunea 200x200 pentru o imagine de dimensiunea 200x200. Dacă această imagine este colorată, această dimensiune devine 200x200x3 (RGB). De fapt, fiecare manipulare în procesarea imaginilor este o operație matricială. Să presupunem că este dorită o operație de estompare pe imagine. Un anumit filtru se deplasează peste întreaga matrice, făcând modificări fie asupra tuturor elementelor matricei, fie asupra unei părți a elementelor matricei. Ca urmare a acestui proces, partea necesară sau întreaga imagine devine neclară.

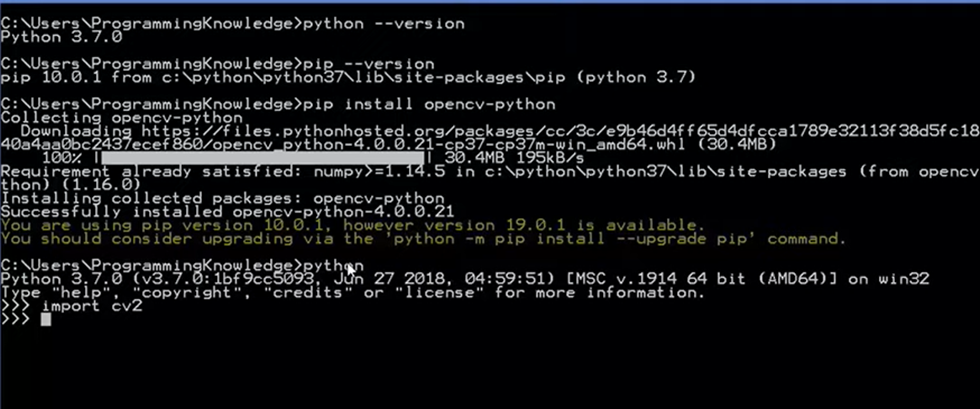
Prelucrarea imaginilor este necesară în multe cazuri. În general, aceste operații sunt aplicate pe datele formatului de imagine care vor fi utilizate în modele de învățare profundă. De exemplu, nu contează că datele sunt colorate în unele proiecte. În acest caz, utilizarea imaginilor color pentru antrenament va cauza pierderi de performanță. Una dintre cele mai utilizate structuri de învățare profundă a procesării imaginilor este Rețelele neuronale convoluționale (CNN). Această rețea determină atributele necesare pentru antrenament cu stratul convoluțional pe imagine. În acest moment, este posibil să trebuiască procesate doar anumite părți ale imaginilor care vor fi folosite pentru instruire. Proeminența liniilor mai rotunjite decât a liniilor ascuțite din imagini poate îmbunătăți uneori succesul antrenamentului.

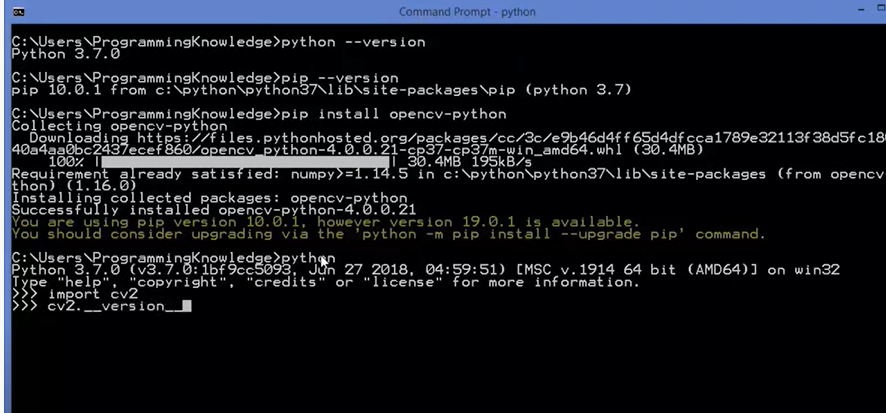
În astfel de cazuri, se utilizează tehnici de procesare a imaginilor.

Aceeași logică se bazează pe operațiunile programelor de optimizare a imaginilor utilizate în viața de zi cu zi pe lângă situațiile descrise mai sus. Există multe etape în procesarea imaginilor, cum ar fi îmbunătățirea calității imaginilor, realizarea restaurărilor pe imagini, eliminarea zgomotului, egalizarea histogramei.

**OpenCV**

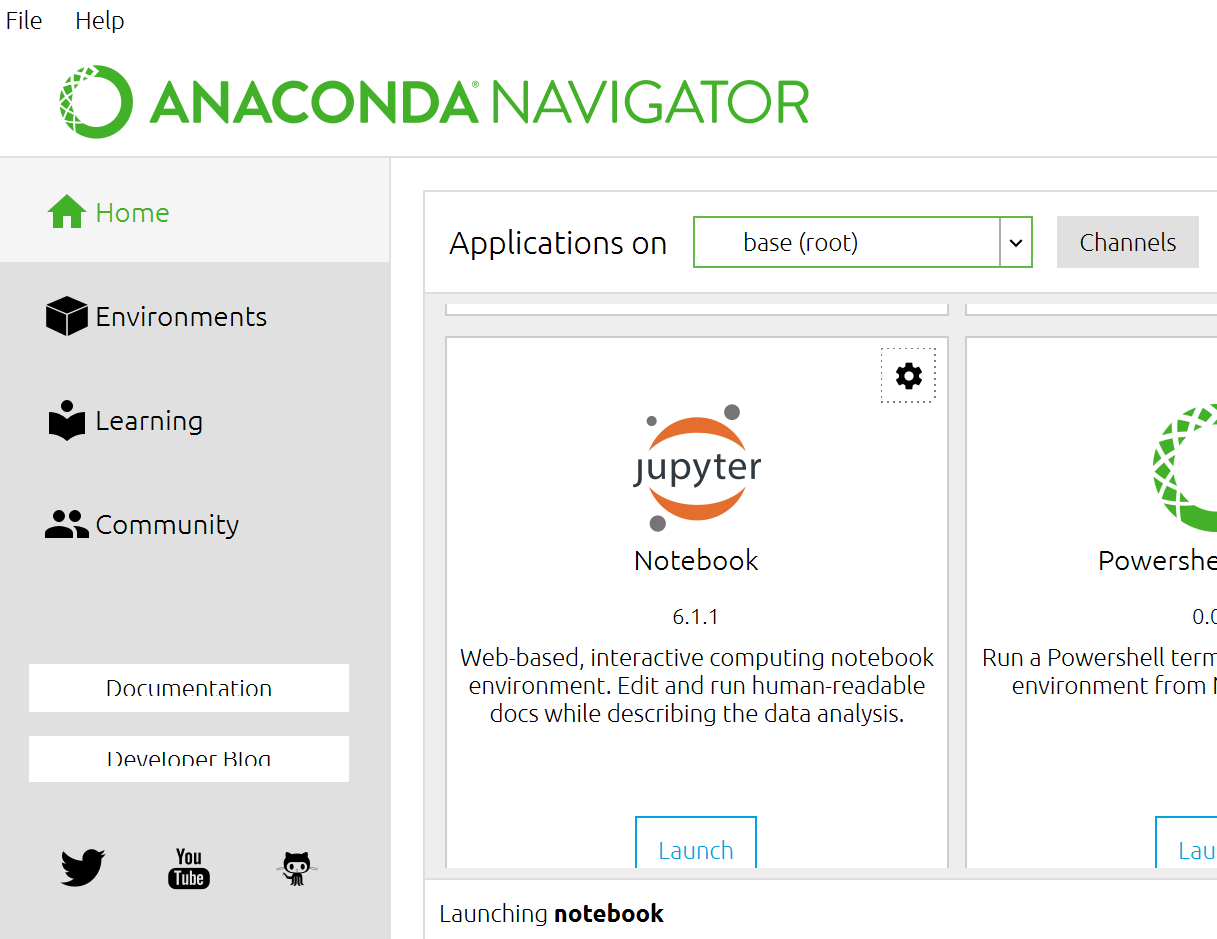
OpenCV este una dintre cele mai populare biblioteci utilizate pentru procesarea imaginilor [2]. Există multe companii care utilizează OpenCV, precum Microsoft, Intel, Google, Yahoo. OpenCV acceptă o mare varietate de limbaje de programare, cum ar fi Java, C ++, Python și Matlab. Toate mostrele din această lucrare sunt codificate în Python.

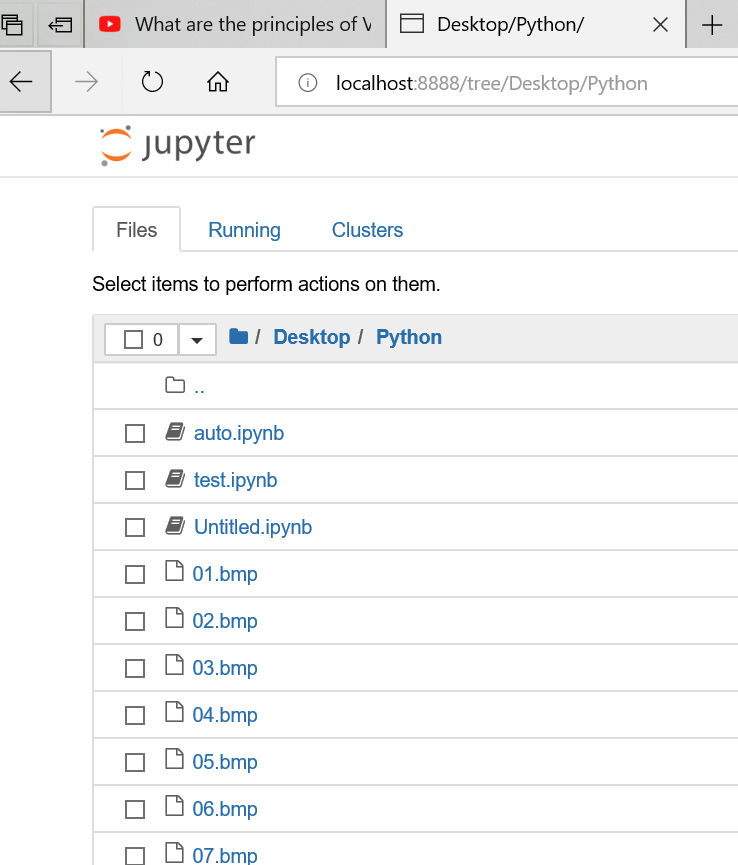




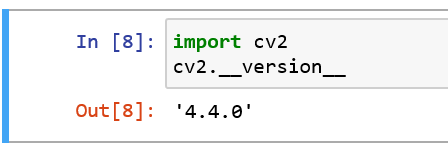
Exemplul 1.

Lansați Anaconda Navigator, environment (base) root, Jupyter Notebook.





Deschideți Fișierul Test în Python 3 și verificați versiunea de OpenCV instalată.



În primul rând, bibliotecile sunt importate. Există câteva funcții în OpenCV care nu funcționează stabil în fiecare versiune. Una dintre aceste funcții este „imshow”. Această funcție ne permite să vedem schimbările din imagine ca urmare a operațiunilor noastre. Biblioteca matplotlib va fi utilizată ca o soluție alternativă în această lucrare pentru cei care au astfel de probleme.

import cv2  
from matplotlib import pyplot as plt  
import numpy as np



Figura 1. Imaginea inițială (11.bmp)

Procesele care trebuie efectuate vor fi aplicate pe imaginea prezentată mai sus (Figura 1). Imaginea este citită inițial pentru a putea fi procesată.

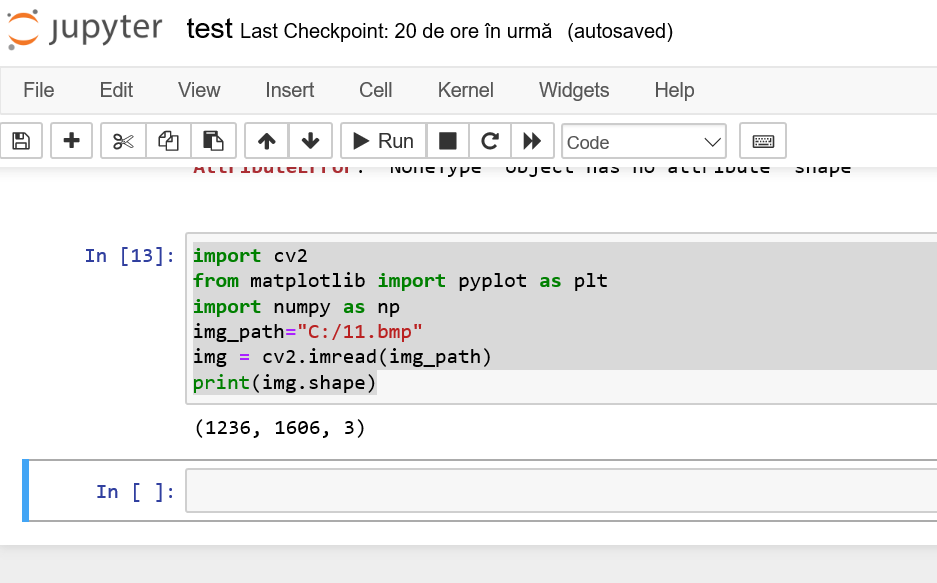


Figura 2. Dimensiunile imaginii

Dimensiunile imaginii sunt de 1236 x 1606 pixeli în Figura 2. Vedem rezultatul de 1236x1606x3 când vrem să imprimăm dimensiunile după procesul de citire. Deci, a fost creată o matrice până la dimensiunile imaginii și acestei matrici i se atribuie valorile fiecărui pixel al imaginii. Există 3 dimensiuni de la RGB, deoarece imaginea este colorată.

Dacă dorim să convertim imaginea în alb și negru, se utilizează funcția cvtColor.

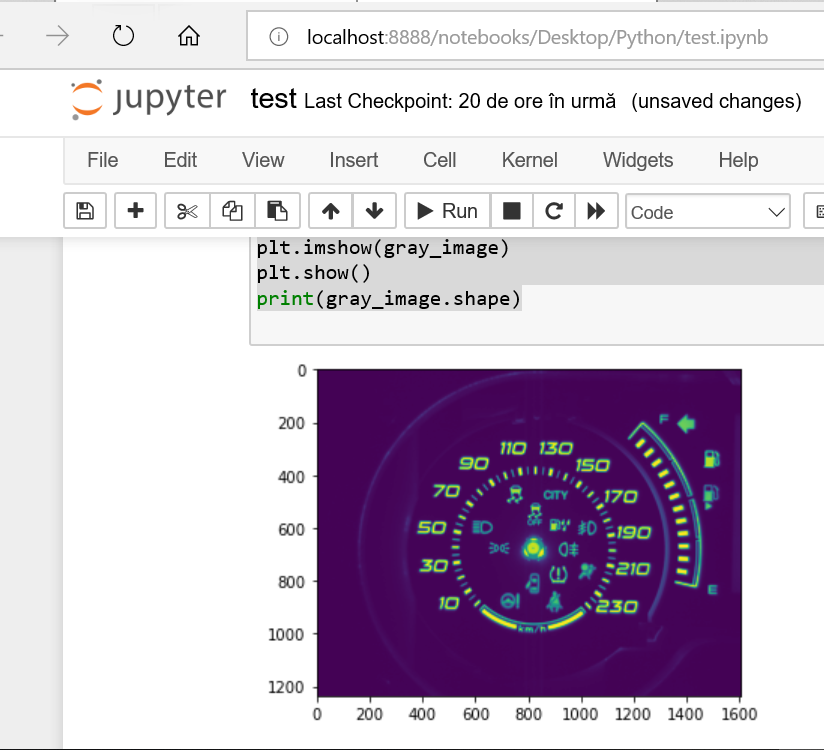
gray\_image = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

plt.imshow(gray\_image)

plt.show()

print(gray\_image.shape)

Dacă vrem să vedem schimbarea care are loc ca urmare a acestei funcții, folosim funcția imshow de la matplotlib.



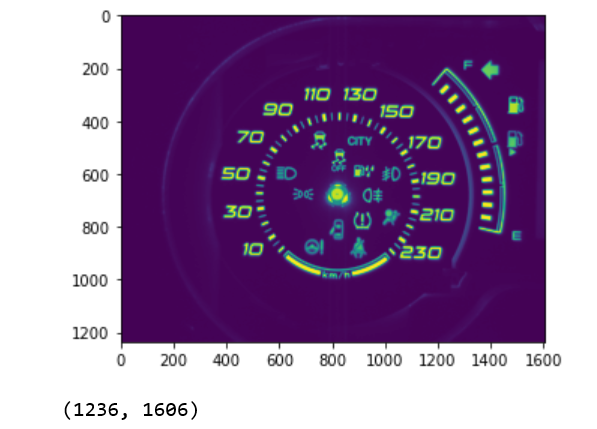


Figura 3. Versiunea Gray

Așa cum se arată în Figura 3, am convertit imaginea în alb și negru. Când le verificăm dimensiunile, nu mai există 3 dimensiuni din cauza RGB.

Când ne uităm la valorile matricei imaginii, vedem că aceasta constă din valori cuprinse între 0 și 255. În unele cazuri, este posibil să dorim ca această matrice să fie formată doar din valorile 0 și 255. Funcția de prag este utilizată în astfel de cazuri.

Blur (Estompare)

O altă metodă utilizată în procesarea imaginilor este estomparea. Acest lucru poate fi realizat cu mai multe funcții.

output2 = cv2.blur(gray\_image, (10, 10))  
plt.imshow(output2)  
plt.show()

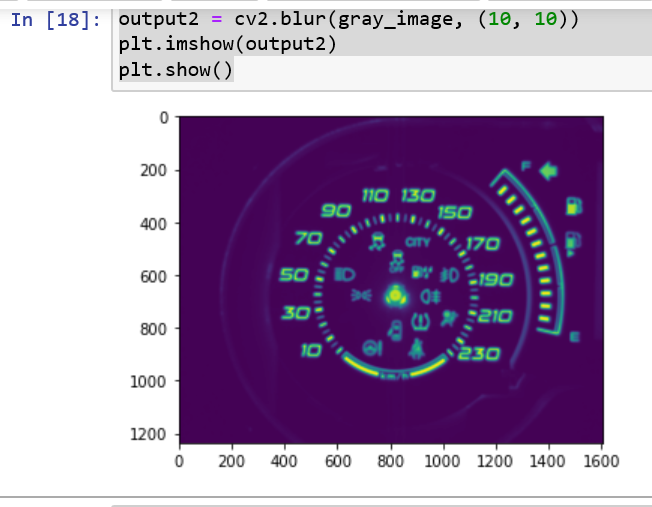


Figura 4 Blur

output2 = cv2.GaussianBlur(gray\_image, (9, 9), 5)  
plt.imshow(output2)  
plt.show()

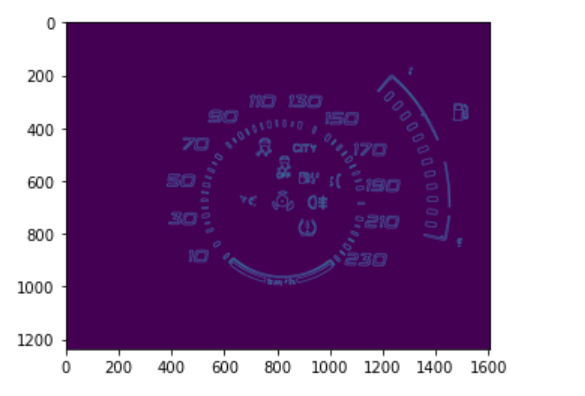


Figura 5. Gaussian Blur

Așa cum se vede în Figura 4 și Figura 5, imaginea alb-negru este estompată cu filtrele de estompare specificate și gradele de estompare. Acest proces este de obicei folosit pentru a elimina zgomotul din imagini. De asemenea, în unele cazuri, antrenamentul este grav afectat din cauza liniilor ascuțite din imagini. Este disponibil în cazurile în care este utilizat din acest motiv.

În unele cazuri, este posibil ca datele să fie rotite pentru mărire sau imaginile să fie utilizate, deoarece datele pot fi înclinate. Următoarele funcții pot fi utilizate în aceste cazuri.

(h, w) = img.shape[:2]  
center = (w / 2, h / 2)  
M = cv2.getRotationMatrix2D(center, 13, scale =1.1)  
rotated = cv2.warpAffine(gray\_image, M, (w, h))  
plt.imshow(rotated)  
plt.show()

În primul rând, centrul imaginii este determinat, iar rotația este realizată de acest centru. Primul parametru al funcției getRotationMatrix2D este valorile centrale calculate. Al doilea parametru este valoarea unghiului. În cele din urmă, al treilea parametru este valoarea scalării care trebuie aplicată după rotație. Dacă această valoare este setată la 1, va roti aceeași imagine numai în funcție de unghiul dat, fără nici o scalare, figura 5.

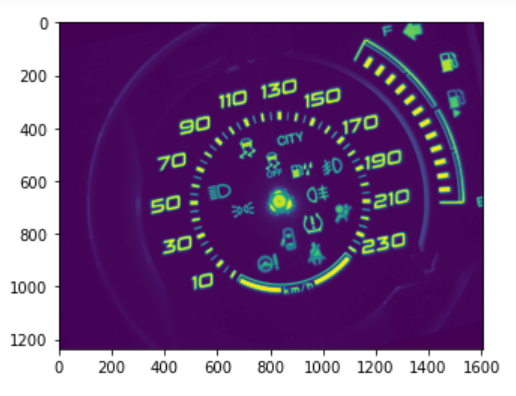


Figura 6 Rotire

**Exemplul 2**

Metodele menționate mai sus sunt adesea utilizate împreună în proiecte. Să facem un exemplu de proiect pentru o mai bună înțelegere a acestor structuri și procese.

Să presupunem că vrem să pregătim un pilot de conducere autonomă pentru vehicule. Când imaginea din Figura 7 este examinată pentru această problemă, pilotul nostru autonom ar trebui să poată înțelege calea și benzile. Putem folosi OpenCV pentru această problemă. Deoarece culoarea nu contează în această problemă, imaginea este convertită în alb și negru. Elementele matricei setează valorile 0 și 255 cu valoarea pragului determinată. Pentru a scăpa de zgomot, estomparea se realizează cu funcția GaussianBlur. Părțile până aici pot fi examinate în detaliu din Figura 8 până la 13.



Figura 7. Imagine inițială( auto.jpg)

Exercitiu. Cititi imaginea ca in exemplul 1. Afisati imaginea.

Apoi, obținem imaginea alb-negru:



Figura 8. Imaginea alb-negru

Aplicăm o funcție prag, selectarea valorii pragului este esențială pentru această funcție. Valoarea pragului este setată la 200 pentru această problemă. Putem clarifica alte detalii, deoarece va fi suficient să ne concentrăm pe marginea drumurilor și a benzilor de rulare.

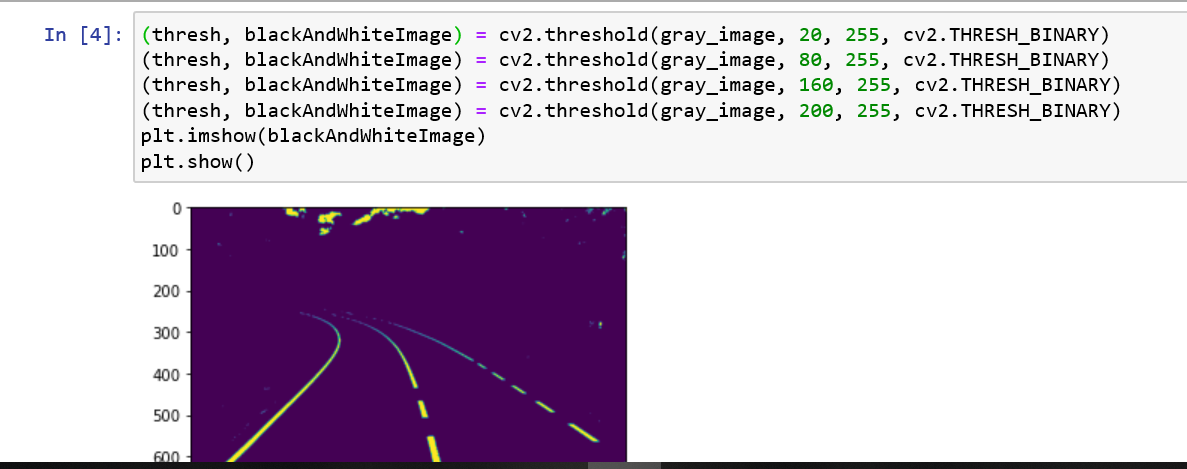


Figura 9. Aplicarea funcției prag



Figura 10. Rezultat prag 1

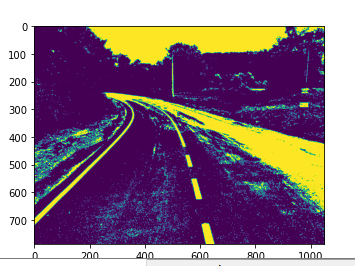


Figura 11. Rezultat prag 2

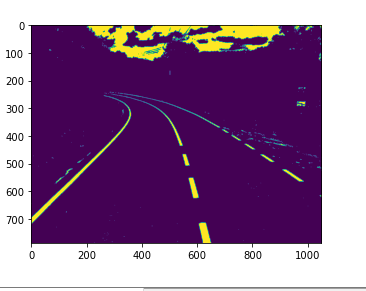


Figura 12. Rezultat prag 3

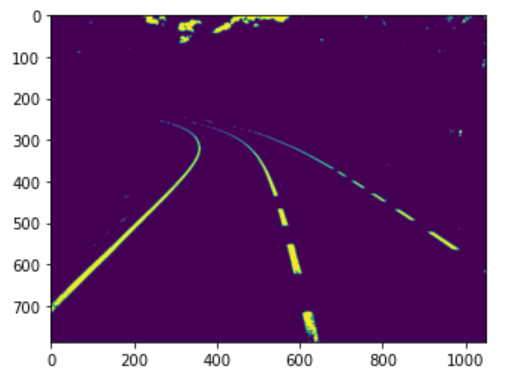


Figura 13. Rezultat prag 4

După aceste procese, se aplică detectarea marginii Canny.

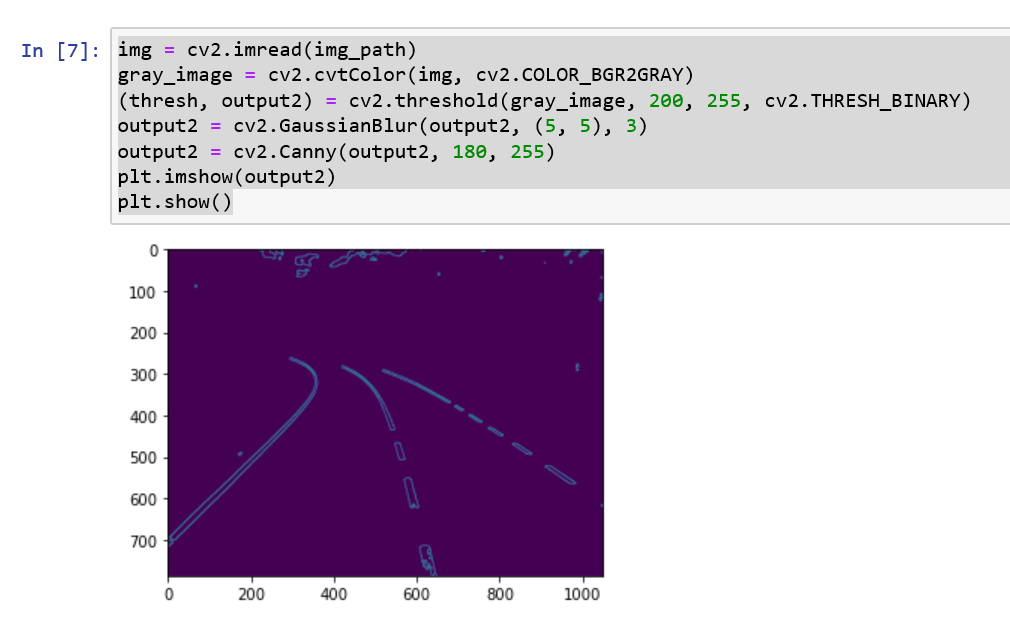


Figura 14. Rezultat Canny

Primul parametru pe care îl ia funcția Canny este imaginea căreia îi va fi aplicată operația. Al doilea parametru este valoarea pragului scăzut și al treilea parametru este valoarea pragului ridicat. Imaginea este scanată pixel cu pixel pentru detectarea marginilor. De îndată ce există o valoare mai mică decât valoarea pragului scăzut, se detectează prima parte a marginii. Când se găsește o valoare mai mare decât valoarea pragului mai mare, se determină cealaltă parte și se creează marginea. Din acest motiv, valorile parametrului prag sunt determinate pentru fiecare imagine și pentru fiecare problemă. Pentru a observa mai bine efectul GaussianBlur, să facem aceleași acțiuni fără a estompa de data aceasta.

Când funcția GaussianBlur nu este implementată, zgomotul este clar vizibil în Figura 14. Este posibil ca aceste zgomote să nu fie o problemă pentru proiectul nostru, dar vor avea un impact mare asupra succesului antrenamentului în diferite proiecte și situații. După această etapă, procesele se efectuează pe imaginea reală (standard) pe baza marginilor determinate. Pentru aceasta sunt utilizate funcțiile HoughLinesP și line.

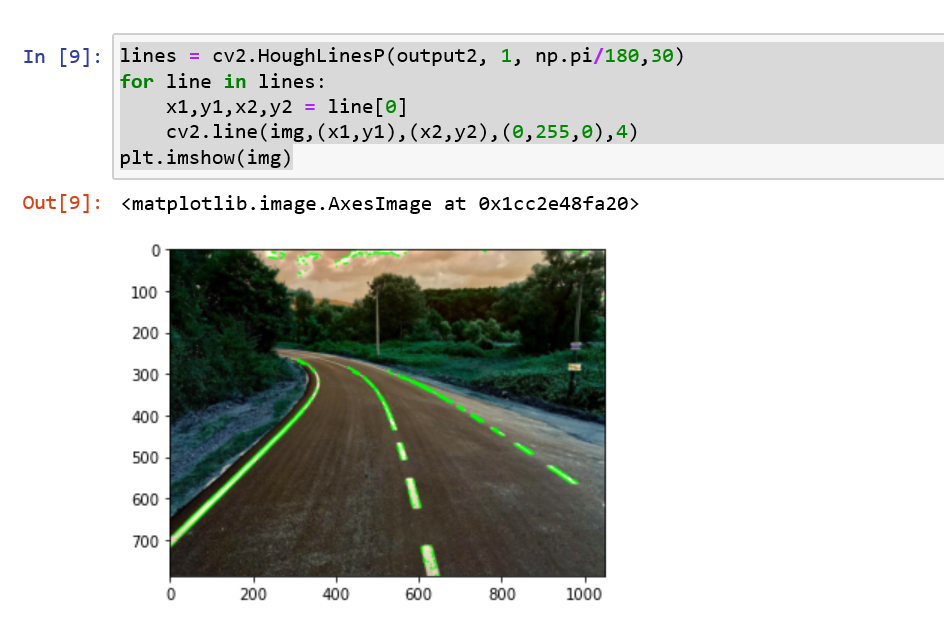


Figura 15. Imaginea cu aplicarea funcției HoughLinesP

După cum se vede în imaginea din Figura 15, limitele și benzile de drum au fost realizate frumos. Cu toate acestea, când Figura 15 este examinată cu atenție, vor fi observate unele probleme. Deși nu a existat nicio problemă în determinarea limitelor benzii și drumului, norii au fost percepuți și ca limite ale drumului. Metoda de mascare trebuie utilizată pentru a preveni aceste probleme.

def mask\_of\_image(img):

height = img.shape[0]

polygons = np.array([[(0,height),(2200,height),(250,100)]])

mask = np.zeros\_like(img)

cv2.fillPoly(mask,polygons,255)

masked\_image = cv2.bitwise\_and(img, mask)

return masked\_image

plt.imshow(img)

Putem face procesul de mascare cu funcția mask\_of\_image. În primul rând, zona care urmează să fie mascată este determinată ca un poligon. Valorile parametrilor sunt valori specifice datelor.

Masca (Figura 16) va fi aplicată pe imaginea reală. Nu există niciun proces pentru regiunile corespunzătoare zonei negre din imaginea reală. Cu toate acestea, toate procesele de mai sus sunt aplicate zonelor corespunzătoare zonei albe.

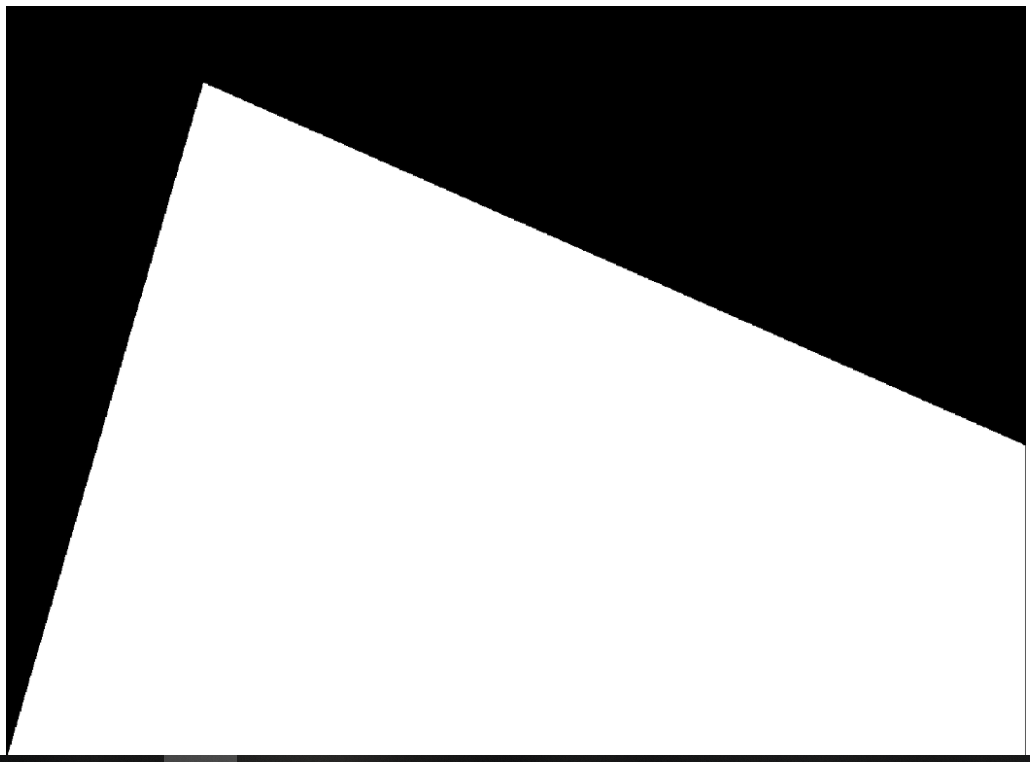


Figura 16. Masca

Așa cum se arată în Figura 17, ca rezultat al procesului de mascare, am rezolvat problema pe care am văzut-o în nori.



Figura 17. Imaginea cu masca

**Incarcati fisierul test.ipynb pe platforma**.