

Comparație teoretica și experimentală a unor algoritmi de sortare de bază

Gabriel Mares, Andrei Jucan, Lascu-Adelin Crăciunesc

Iunie 2022

Cuprins

1 Introducere	3
1.1 Definirea problemei	3
1.2 Cazuri de utilizare ale aplicației	3
1.3 Utilizarea rezultatelor obținute	3
1.4 Identificarea cerințelor aplicației	3
1.5 Contribuții personale	4
2 Tehnologii utilizate	5
2.1 Descrierea stack-ului de tehnologii alese	5
2.2 Descrierea metodelor utilizate	5
2.2.1 Grayscale/cvtColor	5
2.2.2 Gaussian Blur	5
2.2.3 Canny	5
2.2.4 Hough Lines	6
3 Testarea Aplicației	7
4 Concluzii și direcții viitoare	9
5 Bibliografie	10
Bibliografie	10

1 Introducere

1.1 Definirea problemei

Cu toții știm cât de obosit este să conduci pe o durată lungă de timp, iar orice metodă de ușurare a acestui lucru inevitabil este bine venită. Astfel, cu sfârșitul celui de-al douăzecilea secol, focusul anumitor dezvoltatori și companii a fost acesta de a îmbunătăți experiența condusului unei mașini.

Printre acestea, se află și problema menținerii direcției traficului, ce se bazează pe problema detectării a benzii de circulație, astfel încât vehiculul folosit poate fi îndrumat în mod automat de către un calculator de bord în a controla vehiculul astfel încât conducătorul poate să ridice membrele de pe controalele mașinii pe distanțe lungi în condiții de siguranță.

1.2 Cazuri de utilizare ale aplicației

După cum am menționat mai sus, principalul caz de utilizare al acestui proiect este manevrarea automată a unui vehicul pentru a ușura procesul de deplasare. Acest lucru poate rezulta în mai multe beneficii: cost redus din punct de vedere al resurselor umane, reducerea accidentelor prin eliminarea riscului de eroare umană, dar și alte beneficii mai puțin relevante.

1.3 Utilizarea rezultatelor obținute

Rezultatele obținute pot fi folosite în a ghida un computer de bord în a menține direcția de mers a autovehiculului, aşadar aceste rezultate trebuie să fie cât se poate de precise în a nu indica mijlocul de transport în situații periculoase.

1.4 Identificarea cerințelor aplicației

Deoarece acuratețea rezultatelor aplicației este foarte importantă, am decis ca următorii doi KPIs ca fiind importanți:

- Cât timp este detectat drumul: Acest KPI are ca măsură durata de timp în care o bandă de circulație este detectată versus folosirea ultimelor benzi "bune" identificate de către aplicație. Mai multe detalii despre această interpretare se vor regăsi în capitolul următor. Pentru acest KPI, am decis ca 95% este un procent realizabil pentru aplicația noastră.
- Acuratețea detectării benzii de circulație: Acest KPI se bazează pe acuratețea cu care liniile delimitatorii unei benzi sunt detectate. Deoarece acest rezultat este ușor influențat de către diferiți factori (vremea la care se realizează detectia benzii, condițiile meteo, condițiile drumului sau altele). Astfel, am decis să fixăm 2 KPI-uri pentru această categorie: unul pentru situații normale sau ideale - 90%, respectiv pentru situații complexe - 75%.

1.5 Contribuții personale

Deoarece contribuția personală este importantă în aplicații de genul pe piața comercială relevantă acestor soluții, aplicația noastră realizează cea mai importantă operație din cadrul acestui proces - detectarea liniilor. Prin acest proces, detectem 2 linii potrivite pentru a forma o direcție de sens prin calcularea unghiului pe care acestea le formează față de axa Ox a frame-ului curent.

2 Tehnologii utilizate

2.1 Descrierea stack-ului de tehnologii alese

Stack-ul de tehnologii utilizat este cel cu care s-a lucrat în cadrul materiei curente: python3.6 ca și variantă de python utilizată pentru compatibilitate cu OpenCV - principala librărie folosită pentru operațiile rudimentare pentru realizarea operațiunii aplicației noastre.

2.2 Descrierea metodelor utilizate

2.2.1 Grayscale/cvtColor

Grayscale este, în principiu, prima operație realizată în cadrul aplicației. Metoda prin care se realizează această operație, cvtColor, ce aparține librăriei OpenCV, ne permite convertirea unei imagini dintr-un domeniu de culori în altul, în cazul nostru RGB (red, green, blue) to Grayscale [1].

Acest lucru ne ajută în a simplifica procesul de realizare a următoarelor metode, astfel reducând timpul de execuție per iterație a aplicației.

2.2.2 Gaussian Blur

Blur-ul este procesul prin care aplicăm unei imagini un kernel de normalizare sub următoarea formă: [2]

$$K = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Totuși, Gaussian Blur-ul funcționează pe același principiu, doar că kernel-ul gaussian este puțin mai detaliat, astfel încât noise-ul dintr-o imagine să fie cât se poate de redus [2] pentru, din nou, a ușura procesul de detectare a benzii de circulație.

2.2.3 Canny

Conform documentației openCV [3], canny este un proces prin care putem detecta marginile anumitor obiecte. Aceasta este compus dintr-un proces cu mai mulți pași:

- Reducere de noise: Acest proces este realizat de Gaussian Blur, proces explicat anterior.
- Afarea gradientului de intensitate: Imaginea trecută prin blur anterioara este apoi supusă unui Kernel Sobel în ambele axe Ox și Oy pentru a putea găsi gradientul de intensitate al marginilor. Astfel, putem grupa fiecare pixel pe o linie.

- Surpresia pixelilor nedoriți: După obținerea gradientului sus menținut, se scaneză încă odată imaginea obținută pentru a îndepărta pixeli nedoriți - pixeli ce nu fac parte dintr-o linie. Pentru asta, pentru fiecare pixel se realizează faptul dacă acesta este un maxim local în zona lui în direcția gradientului.
- Stabilirea marginilor: În acest pas, folosindu-ne de două valori, una minimală și una maximală, pentru a determina dacă un pixel chiar face parte dintr-o margine sau nu. În cazul aplicației noastre, valoarea minimală este 50, iar cea maximală este 150.

2.2.4 Hough Lines

Hough Lines, în contextul aplicației noastre, acesta fiind contribuția noastră personală, este procesul prin care ne folosim de rezultatele obținute de către operația de canny în a detecta 4 puncte sub forma (x_1, y_1, x_2, y_2) . Astfel, calculăm unghiul dreptei ce este analizată curent ca fiind $(y_2 - y_1)/(x_2 - x_1)$. Știm cum că acest unghi este pozitiv dacă se află în intervalul $(0, 1]$, respectiv $[-1, 0)$ dacă este negativ. Astfel, considerând noțiunea anterioară, putem echivala acest lucru cu faptul că un unghi "pozitiv" este linia dreaptă delimitatoare a benzii, iar cel negativ este linia stângă delimitatoare. Dacă aceste două linii nu pot fi găsite, se vor folosi ultimele linii (pentru fiecare direcție) știute ca fiind bune de către aplicație.

3 Testarea Aplicației

Testarea aplicației este realizată prin folosirea acesteia asupra mai multor exemple de videoclipuri de condus, acestea fiind realizate sub diferite condiții. Astfel, mai jos aveți o colecție de rezultate.



Figura 1: Autostrada puțin curbată

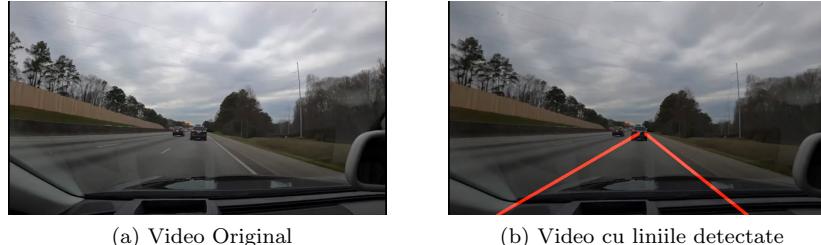


Figura 2: Autostrada dreaptă, bord vizibil

După cum se poate observa, în aceste exemple, nu doar că găsim câte două linii pentru fiecare frame, dar și găsim linii ce acoperă aproape 1:1, dacă nu chiar perfect, delimitatorii de bandă. Aceste două exemple respectă KPI-urile noastre în întregime.

Totuși, următorul exemplu este ceva mai complex atât din cauza unghiului în care s-a realizat filmarea, dar și posibilității de alegere a mai multor ”drepte” - parapeți, linii de delimitare nu doar a altor benzi, dar și a drumului în general.



Figura 3: Autostrada dreaptă, în urcare, parapeți vizibili

Acest lucru este rezultatul videoclipului în sine, acesta fiind foarte noisy din punctul nostru de vedere, astfel afectând metricele noastre. O nouă linie este găsită aproximativ în 70% din cadrele videoclipului, iar doar aproximativ 60% din aceste linii sunt corect detectate. Multe dintre aceste rezultate se duc fie spre parapetele din dreapta vizibil în mare parte din videoclip, fie undeva între a doua bandă și terenul verde dintre direcțiile de mers.

În general, suntem mulțumiți de rezultatele obținute în timpul acordat în cadrul modulului, dar desigur că există loc de îmbunătățiri.

4 Concluzii și direcții viitoare

În urma testelor noastre desfășurate pe diferite videouri, am observat o rată aproximativă de 75% de detectare de noi linii de benzi, iar dintre acestea, pe videoclipuri cu condiții ideale, 90% din detectii sunt bune, iar pe videoclipuri provocatoare, avem un procent de rezultate bune mult mai mic decât cel propus de aproximativ 45%.

Cu toate acestea, deși KPI-urile noastre nu au fost realizate în deplin, suntem conștienți unde am greșit. În primul rând, din lipsa exemplelor de videoclip, nu am putut detecta root cause-ul (sau root causes, la plural c:), dar putem să îmbunătățim aplicația prin adăugarea a două elemente importante în cadrul acesteia: calibrarea camerei realizat într-un mod mai rudimentar (noi am realizat acest lucru pe anumite rezultate, dar doar în mod superficial) și adăugarea unui bird eye view la nivel de imagine pentru a generaliza procesul de detecție.

5 Bibliografie

Bibliografie

- [1] https://docs.opencv.org/3.4/d8/d01/group__imgproc__color_conversions.html
- [2] https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html
- [3] https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html
- [4] <https://medium.com/@techreigns/how-does-a-self-driving-car-detect-lanes-on-the-road-fc5>