Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Irányítástechnika és Informatika Tanszék

M9 – Sávkövetés

Mérési Útmutató

Irányítástechnika és Képfeldolgozás Laboratórium

Szemenyei Márton

Tartalomjegyzék

1 Autonóm járművek

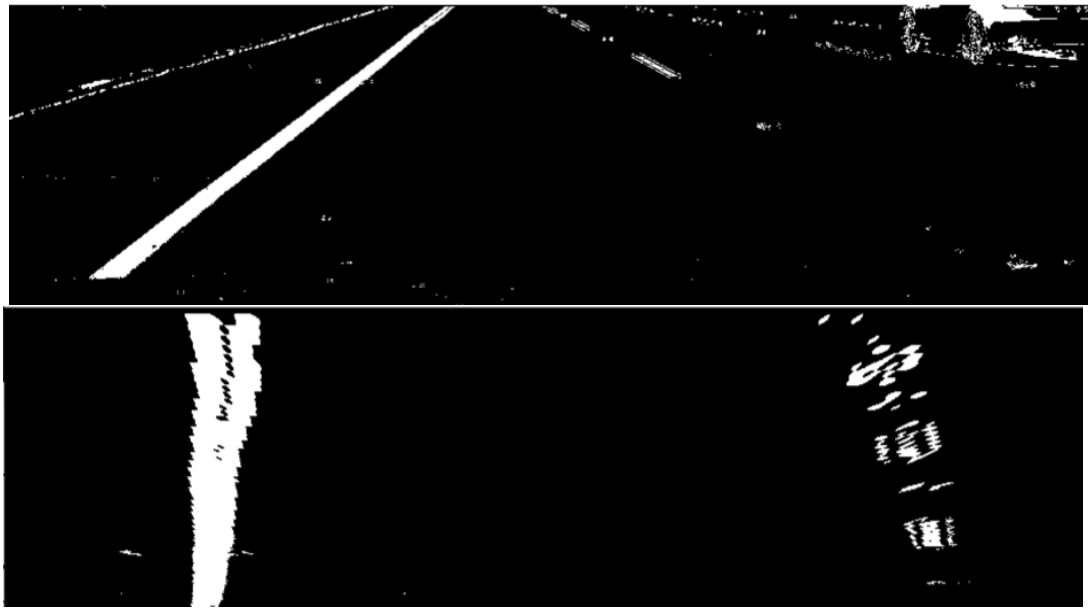
Napjainkban az autonóm járművek egyre hangsúlyosabb szerepet töltenek be a mindennapi életben. Kontrollált ipari környezetekben már jó néhány éve alkalmaznak ilyen robotokat, azonban a számítógépes látás és a mesterséges intelligencia rohamos fejlődésének köszönhetően már az utcákon is megjelentek önvezető autók formájában. A legmagasabb autonómiai szinttel rendelkező járművek fejlesztése azonban a jelen labor tárgyán messze túlmutat, így a mérés során a látás alapú autonóm viselkedések legegyszerűbb formájával, a sávdetektálással és -követéssel fogunk megismerkedni.

1.1. Sávdetektálás

A sávdetektálás elvégzésére alapvetően két fontos módszer létezik. Az egyik a sávok intenzitás alapú detektálása, míg a másik a sávok élkeresés alapján történő megtalálása. Mindkét módszer megbízhatósága önmagában kérdéses, így gyakorta szokás a kettő módszer párhuzamosan, egymás korrigálására felhasználni. Az intenzitás alapú detektálás legegyszerűbb módja a küszöbözés, amikor egy előre meghatározott küszöbérték egyik oldalán lévő pixeleket 0-ba, míg a másik oldalon lévőket 1-be állítjuk, így egy bináris képet kapunk. Fontos megjegyezni, hogy egy előre meghatározott küszöbérték használata esetén a fényviszonyok megváltozása a módszer eredményét is nagymértékben befolyásolhatja, így a gyakorlatban gyakran használunk adaptív – az adott képen lévő intenzitások eloszlásából meghatározott – küszöbértéket.

A Sávdetektálás elvégezhető képi élek segítségével is, melyek definíció szerint a képen található szomszédos pixelek között végbemenő nagy mértékű, egy irányú intenzitás változások. Lényeges tulajdonságuk, hogy az intenzitás csak az egyik irányban változik, míg a másikban konstans, valamint, hogy a változás éles, ugrásszerű. Az éldetektálás során rendkívül gyakran előfordul, hogy különféle egyszerű alakzatok (téglalap, kör) határvonalait keressük, amely esetben célszerű lehet a megtalált élpontokra egy egyenes modellt illeszteni, így a képen megtalált egyenes szerű elrendezésben található pontokat egy paraméteres modellel leírhatjuk, amely az alakzatok detektálását rendkívül megkönnyíti.

A probléma nehézsége, hogy természetesen a képen talált élpontok csak egy része fog egyenesekre illeszkedni, és azok közül is számos pont külön-külön egyenesre illeszkedik, így egyszerre kell azt meghatároznunk, hogy mely pontok illeszkednek egy egyenesre, és hogy milyen paraméterek írják le ezt az egyenest. Ha két kérdés közül bármelyikre tudnánk a választ, a probléma megoldása triviális volna. Erre a problémára az egyik legnépszerűbb algoritmus a Hough transzformációra épülő alakzat detektálás. A küszöbözés, éldetektálás, valamint a Hough-transzformáció részletei megtalálhatók a Számítógépes Látórendszerek c. tárgy jegyzetében [1].



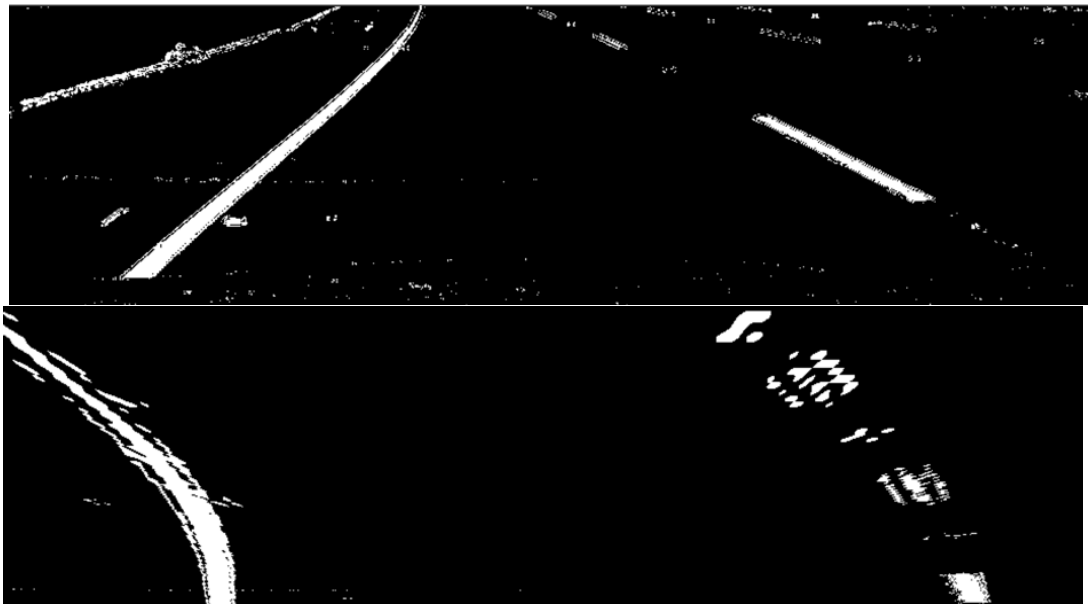
1.1. ábra. *Az él kép perspektív transzformáció előtt és után.*

Az egyenesek meghatározása a sávdetektálás szempontjából rendkívül hasznos lépés lehetne, azonban a valóságban a megtalált sávok számos esetben nem egyenes alakúak, hanem az út görbületének megfelelő módon változnak. Éppen ezért ilyen esetekben a Hough transzformáció nem fog pontos eredményt adni, ráadásul az egyenes görbületét sem képes meghatározni. E megfontolásokból érdemes egy másik, alternatív megoldást alkalmazni a sávok élképből történő meghatározására.

Ez a módszer első lépésként egy perspektív transzformációt alkalmaz a képen, aminek segítségével a képet egy számunkra kedvezőbb formára alakítjuk. A perspektív transzformáció egy olyan általános projektív transzformáció, amely egy tetszőleges kétdimenziós síkot egy másik kétdimenziós síkra vetít le. Ha belegondolunk, a képalkotás során pontosan ez történik: az autóút síkja a kamera képsíkjára vetül. Ennek a transzformációnak az egyik alapvető hatása, hogy számos geometriai jellemzőt (méret, szögek, párhuzamosság) torzít, melynek következtében a képen látható sávokat meghatározó egyenesek nem párhuzamosak. Ha azonban ezt a transzformációt meg tudjuk határozni, és valamilyen módon visszacsinálni, akkor egy sokkal könnyebben feldolgozható képet kapunk.

A perspektív transzformáció meghatározása rendkívül könnyen elvégezhető: mivel a transzformáció mátrixa 3x3-as, és ez egyik elemét szabadon megválaszthatjuk, ezért elég egyszerűen négy pontpárt kiválasztanunk, melyek értékeiből a transzformáció meghatározható. Ez a gyakorlatban ezt jelenti, hogy kiválasztunk az eredeti képen négy pontot, majd megadjuk, hogy hova szeretnénk, hogy ezek a pontok kerüljenek a transzformáció után. Mivel a jelen esetben célunk, hogy a sávot meghatározó két egyenes az új képen függőleges legyen, így ez a négy pont célszerűen a sáv széleinek 2-2 végpontja.

Miután ez a transzformált kép előállt, előállítunk egy speciális hisztogramot: a kép minden oszlopához összeszámoljuk, hogy hány darab egyes pixel található az adott oszlopban. Mivel a sávok görbülhetnek, ezért célszerű csak a kép alsó, a járműhöz közeli felén elvégezni ezt a számlálást. Ha az így kapott hisztogram két maximumát



1.2. ábra. *A perspektív transzformáció eredménye kanyarodó út esetén.*

megkeressük, akkor ebből megkaphatjuk a sávok pozícióit. A maximumok értékéből következtethetünk arra is, hogy az adott sáv folytonos vagy szaggatott felfestésű. Az eljárás utolsó lépéseként külön-külön összegyűjtjük a két megtalált vonalhoz tartozó pontokat az egész képről, majd a legkisebb négyzetek módszerének segítségével másodfokú polinomot illesztünk a pontokra, melyből a sáv görbülete meghatározható.

2 A mérés környezete

A mérés során a *PyCharm* elnevezésű IDE áll rendelkezésre, amely rendkívül sokoldalú szolgáltatásokkal könnyíti meg a szoftverfejlesztést, például konfigurálható automatikus formázási lehetőségek állnak rendelkezésünkre. További részletekért érdemes lehet a JetBrains ide vonatkozó weboldalát [2] felkeresni. Függvények, objektumok esetében a ***Ctrl+P*** billentyűkombináció pop-up segítségként szolgálva mutatja nekünk a paramétereket. A mérés során használt programnyelv a Python 3-as verziója lesz.

A Python programnyelvhez számos hasznos függvénykönyvtár tartozik, melyek a mérési feladatok megvalósítását nagymértékben megkönnyítik. A Python nyelv egyik rendkívül kényelmes funkciója a beépített package manager, amelynek segítségével az egyes könyvtárak automatikusan telepíthetők, telepítsük után pedig minden további beállítás nélkül használhatók. A Pythonhoz két ilyen package manager is tartozik, az egyik a Pip, amely a legtöbb telepíthető Python verzió mellé automatikusan települ, a másik pedig az Anaconda [3], ami a könyvtárkezelési funkciókon túl virtuális környezeteket is képes kezelni.

A Python egyik legfontosabb függvénykönyvtára a Numpy, amely tömbök kezelésére, illetve számtalan numerikus algoritmus használatára ad lehetőséget. A Numpy funkcionalitását kiegészíti a MatplotLib, melynek segítségével különböző ábrákat készíthetünk a tömbjeinkről. Egy harmadik rendkívül hasznos könyvtárcsalád a scikit, ami számos tudományos számításhoz szükséges alkönyvtárt foglal össze. A sckit-image képek kezelésére, a scikit-learn gépi tanulás algoritmusok használatára, míg a scikit-fuzzy fuzzy logika használatára ad lehetőséget. Ezek a könyvtárak tulajdonképpen együttesen kiadják a Matlab funkcionalitásának jelentős részét.

Az OpenCV [4] egy nyílt forráskódú számítógépes látás algoritmusokat tartalmazó függvénykönyvtár. Az OpenCV elsődleges nyelve a C++, azonban elérhetőek hozzá hivatalos wrapperek többek között Java és Python nyelven. Az OpenCV rengeteg hivatalosan támogatott algoritmust tartalmaz, melyen felül a külön letölthető Contrib modulban harmadik felek által kifejlesztett további funkciók is elérhetők.

3 Mérési feladatok

A mérés folyamán az alábbi feladatokat kell elvégezni:

1. Készítsen eljárást a sávokat jelentő élek detektálására! Az éldetektálás során kapott gradienseket szűrje nagyság és irány alapján, valamint végezzen szín alapú szűrést is!
2. Torzítsa a képet perspektív transzformáció segítségével úgy, hogy a sávok párhuzamosak legyenek, majd határozza meg a sávok helyzetét!
3. Készítsen robusztus becslőt a sávok görbületének, az autó helyzetének és ez utóbbi változásának meghatározására!
4. Valósítson meg egy sávtartó algoritmust egyszerű fuzzy irányítás segítségével!
5. Ellenőrizze az algoritmus helyes működését előre felvett videofelvételek segítségével!

4 Hasznos kódrészletek

Videó betöltése

```
clip = cv2.VideoCapture("original.mp4")
```

Képkocka olvasása videóból

```
success, img = clip.read()
if not success:
break
```

Szürkeárnyalatos konverzió

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Sobel operátor futtatása x és y irányban

```
sobelx = cv2.Sobel(gray, cv2.CV_32F, 1, 0, ksize=sobel_kernel)
sobely = cv2.Sobel(gray, cv2.CV_32F, 0, 1, ksize=sobel_kernel)
```

Abszolút érték, maximum számolása

```
abs = np.absolute(array)
max = np.max(abs_sobelx)
```

Típuskonverzió

```
converted = np.uint8(array)
```

Tartományon belüli küszöbözés

```
binary = cv2.inRange(rray, low_thresh, high_thresh)
```

Atan2 tömbön

```
dir = np.arctan2(y, x)
```

Elemenkénti logikai függvények

```
result = np.zeros_like(a)
combined[((a == 255) & (b == 255)) | (c == 255)] = 1
```

Képrészlet kivágása

```
roi = image[y:Y, x:X]
```

Perspektív transzformációhoz szükséges pontok

```
src = np.float32([[380, 0],[875, 235],[60, 235],[470, 0]])
dst = np.float32([[150, 0],[800, 260],[150, 260],[800, 0]])
```

Perspektív transzformáció és az inverzének számítása

```
M = cv2.getPerspectiveTransform(src, dst)
Minv = cv2.getPerspectiveTransform(dst, src)
```

Kép transzformálása

```
warped = cv2.warpPers
```

Hisztogram számítása

```
histogram = np.sum(warped[warped.shape[0]//2:,:], axis=0)
```

Hisztogram megjelenítése

```
plt.figure()
plt.plot(histogram)
plt.show()
```

Tömb shiftelése

```
a = np.roll(a,1)
```

Átlag számítás

```
avg = np.mean(a)
```

LS becslés

```
lineEst = np.linalg.lstsq(X,Y, rcond=None)[0]
```

Fuzzy változó

```
universe = np.linspace(-2, 2, 5)
var = ctrl.Antecedent(universe, 'name')
names = ['nb', 'ns', 'ze', 'ps', 'pb']
var.automf(names=names)
```

Fuzzy szabály készítése

```
rule0 = ctrl.Rule(antecedent=((error['nb'] & delta['nb']) |
(error['ns'] & delta['nb']) |
(error['nb'] & delta['ns'])),
consequent=output['pb'], label='rule pb')
```

Fuzzy szabályzó készítése

```
system = ctrl.ControlSystem(rules=[rule0, rule1, rule2, rule3, rule4])
controller = ctrl.ControlSystemSimulation(system)
```

Szabályzó bemenetének megadása

```
controller.input['error'] = devProc*4
controller.input['delta'] = slope*40
```

Szabályzó kimenetének számolása

```
controller.compute()
control = controller.output['output']
```

5 Ellenőrző kérdések

1. Mit jelent a küszöbözés? Hogyan lehet változó fényviszonyok esetében robusztusan elvégezni?
2. Ismertesse az éldetektálás első deriváltakon alapuló elvégzésének lehetőségeit!
3. Ismertesse az éldetektálás második deriváltakon alapuló elvégzésének lehetőségeit!
4. Hogyan lehet az éldetektálást konvolúciós szűrések segítségével elvégezni? Ismertessen és hasonlítson össze ilyen szűrőablakokat! Mi a közös tulajdonságuk?
5. Ismertesse a Canny algoritmus működését!
6. Mire jó a Hough transzformáció? Hogyan működik?
7. Milyen módszerrel lehet nem egyenes sávok pozícióját és görbületét könnyedén meghatározni?
8. Milyen programozási nyelvet használunk a mérés során? Miért előnyös ez a nyelv multi-platform fejlesztés esetén?

Bibliography

[1] M. Szemenyei, *Számítógépes Látórendszerek*, M. Szemenyei, Ed. BME, 2019. [Online]. Available: https://edu.vik.bme.hu/pluginfile.php/53608/mod_resource/content/0/jegyzetFull.pdf.

[2] *PyCharm Quick Start Guide*. [Online]. Available: https://www.jetbrains.com/help/pycharm/quick-start-guide.html.

[3] *Anaconda*. [Online]. Available: https://www.anaconda.com/.

[4] *OpenCV Documentation*. [Online]. Available: https://docs.opencv.org/4.1.1/.