

**M09 – Autonóm jármű**

**Mérési útmutató**

***Irányítástechnika és képfeldolgozás laboratórium 1.***

Szemenyei Márton

Irányítástechnika és Informatika Tanszék  
2018

Tartalomjegyzék

[**1.** **Objektumkövetés konvolúciós hálók segítségével** 3](#_Toc523894152)

[**1.1.** **Neurális hálózatok alapjai** 3](#_Toc523894153)

[**1.2.** **Konvolúciós hálók** 8](#_Toc523894154)

[**1.3.** **Követés** 12](#_Toc523894155)s

[**1.4.** **Hasznos praktikák** 13](#_Toc523894156)

[**2.** **A mérés környezete** 17](#_Toc523894157)

[**2.1.** **A Python nyelv** 17](#_Toc523894158)

[**2.2.** **PyTorch** 20](#_Toc523894159)

[**3.** **Mérési feladatok** 22](#_Toc523894160)

[**4.** **Hasznos kódrészletek** 23](#_Toc523894161)

[**5.** **Ellenőrző kérdések** 25](#_Toc523894162)

1. **Autonóm járművek**

Blah

* 1. **Sávdetektálás**

Blah

* 1. **Matematikai becslések**

Blah

1. **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 felkeresni (<https://www.jetbrains.com/help/pycharm/quick-start-guide.html>). Függvények, objektumok esetében a ***Ctrl+P*** billentyűkombináció pop-up segítségként szolgálva mutatja nekünk a paramétereket.

* 1. **Raspberry Pi**

Blah

* 1. **Fontos Python könyvtárak**

Numpy, Scikit

Az OpenCV 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. Az OpenCV mérésben használt verziójának dokumentációja elérhető itt: <https://docs.opencv.org/3.4.2/index.html>

1. **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. Készítsen robusztus becslőt a sávok görbületének, az autó helyzetének és ez utóbbi változásának meghatározására!
3. Valósítson meg egy sávtartó algoritmust egyszerű fuzzy irányítás segítségével!
4. Ellenőrizze az algoritmus helyes működését előre felvett videófelvételek, valamint tényleges robotautó segítségével!
5. **Hasznos kódrészletek**

Blah

1. **Ellenőrző kérdések**
2. Ismertesse röviden a Hinge és a Kereszt-entrópia költségfüggvényeket! Mi az előnyük/hátrányuk?
3. Hogyan működik a gradiens módszer? Milyen fontos kiegészítései vannak?
4. Mire való a backpropagation? Hogyan működik?
5. Ismertesse egy konvolúciós háló és gyakori rétegei felépítését!
6. Mi az a saliency? Hogyan állítható elő? Mi a guided backpropagation? Mi értelme van?
7. Milyen módszerekkel kerülhető el az overfitting? Ismertesse ezeket egy-egy mondatban!
8. Milyen módszerekkel javítható egy neurális háló konvergenciája? Ismertesse ezeket egy-egy mondatban!
9. Ismertesse röviden a PyTorch könyvtárat! Milyen alapvető adattípusai, moduljai léteznek? Milyen szolgáltatásokat nyújt neurális hálók számára?
10. Készítsen egy egyszerű szkriptet Python nyelven:
    1. Töltse be a *torch* könyvtár *nn* nevű modulját!
    2. Definiáljon egy új, *MyNet* nevű osztályt, melynek ősosztálya az *nn.Module*
    3. Az osztály konstruktorában hozzon létre egy *mult* nevű **tag**változót, értéke legyen 5.
    4. Definiáljon egy *forward* nevű tagfüggvényt, melynek bemeneti paramétere *x*, és a *mult* nevű tagváltozóval szorozza meg, az eredményt pedig adja vissza.
11. Hogyan néz ki egyetlen tanulási ciklus PyTorch-ban? (Pszeudó kód elég)