Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Békési Gergő Bendegúz

Mérésautomatizálás mesterséges intelligencia segítségével

Diplomatervezés 2

KONZULENSEK:

Dr. Ekler Péter

Urbin Ágnes

Szőke Szilárd

BUDAPEST, 2022

DIPLOMATERVEZÉS 2. FELADAT

Békési Gergő Bendegúz

MSc Villamosmérnök hallgató részére

Mérésautomatizálás mesterséges intelligencia segítségével

Napjainkban az elektronika és autóipari elektronika terén végbemenő bővülés és számos 21. századi folyamat következményeként fókuszba kerül a teljesítménytranzisztorok iparága. A növekvő elektrifikáció és a technikai fejlődés miatt egyre nagyobb disszipációsűrűséggel kell megküzdeniük ezeknek az eszközöknek, így fontos a termikus vizsgálatuk, amely T3Ster mérés segítségével hatékonyan elvégezhető. Ennek automatizációja, robotizált elvégzése fontos feladat amelyhez mind hardveres, mind pedig szoftveres kihívások is kapcsolódnak. A hallgató feladata a mérésautomatizálás látórendszeres támogatása, a mérőrobot számára a megfelelő képbemenet biztosítása és feldolgozása mesterséges intelligencia felhasználásával.

A hallgató feladatának a következőkre kell kiterjednie:

- Mutassa be, foglalja össze az automatizálandó mérés alapelveit!
- Elemezze automatizáltsági fok szempontjából a különböző mérési módszereket!
- Válasszon megfelelő hardver eszközöket a méréshez!
- Készítsen hagyományos gépi látásra és mesterséges intelligenciára alapuló szoftveres megoldást az automatizáció megvalósítására! Ezeket vesse össze!
- Ellenőrizze a fejlesztett látórendszer pontosságát!

Tanszéki konzulens: Dr. Ekler Péter, egyetemi docens

Külső konzulens: Szőke Szilárd (Robert Bosch Kft.)

Budapest, 2022. február 14.

Dr. Charaf Hassan

egyetemi tanár

tanszékvezető

Összefoglaló

You insist that there is something that a machine cannot do. If you will tell me precisely what it is that a machine cannot do, then I can always make a machine which will do just that. (Neumann János)

Napjainkban a növekvő elektrifikáció, az okos-eszközök elterjedése, a villamosenergia-fogyasztási igények növekedése, az elektronika és autóelektronika terén végbemenő bővülés, valamint a villamos autók egyre népszerűbbé válása miatt különös hangsúly helyeződik az elektronikai iparra, valamint azon belül is a teljesítménytranzisztorok iparágára. Ezen elektronikai eszközöknek a nagyobb leadott villamos teljesítményéhez az egyre kisebb felületükön, egyre nagyobb hőteljesítményt kell leadniuk, ami komoly műszaki kihívást eredményez, hiszen épp a túl magas hőmérséklet a vezető meghibásodási ok esetükben. Így kiemelten fontos termikus vizsgálatuk, melyet termikus tranziens méréssel, T3Ster méréssel lehet elvégezni. Ezen mérési folyamat automatizálásában, gyorsításában, gazdaságosabbá tételében vettem részt a Robert Bosch Kft. ThID csapatában, amely termikus modell identifikációval foglalkozik. Egy LabVIEW környezetben vezérelt robothoz fejlesztettem olyan komplex elektronikai eszközt, látórendszert, amely segíteni tudja az automatikusan mérni kívánó robotot a mérendő MOSFET-ek és más elektronikai komponensek megtalálásában, a hozzájuk való navigációban. A látórendszer hardver- és szoftverfejlesztését is én végeztem. Megfelelő szenzorral rendelkező kamerát és hozzá optikát választottam. Ezekhez egyszerre akár több vezérlő számítógép kiszolgálására is képes szerver platformot valósítottam meg egy Raspberry Pi 4 Model B segítségével. Autodesk Inventor Professional 2022 szoftverben megterveztem a mechanikai integrációt segítő konstrukciót, majd 3D nyomtatóval kinyomtattam. Munkám leghangsúlyosabb részében mesterséges intelligencia, teljesen konvolúciós neurális hálók, valamint számítógépes látás algoritmusainak segítségével szemantikus szegmentációt és kulcspont keresést végeztem a robot navigálásához. A teljesen konvolúciós neurális hálókat saját kóddal implementáltam, a tanításhoz szükséges adathalmazt az éles mérés esetében is használt kamerával és optikával készítettem el és kézzel annotáltam. A neurális hálókat komplex vizsgálat alá vetettem. Automatizált futtatások segítségével 21380 alkalommal tanítottam be és értékeltem ki hálókat, módosítva a mélységet, a csatornaszámokat, a felbontást, és a batch méretet. Ezeket a nyílt forráskódú Python 3 programozási nyelven elsősorban az OpenCV, Pytorch és Torchvision könyvtárak segítségével végeztem. A mérendő komponensekhez való navigálást robusztusan megvalósító, 100 µm és 1° pontosságú elektronikai rendszert készítettem, amelyhez 94 % pontosságú IoU érték társult a szemantikus szegmentáció esetében, amelyet valós, eredeti mérési környezetben teszteltem. Szakirodalmi vizsgálatom során nem találtam olyan megoldást, amely ilyen pontossággal használt volna szemantikus szegmentációt és helymeghatározást elektronikai komponensekre. Így munkám felhasználható további tudományos és ipari újításokhoz, projektekhez. Ezen felül újszerű megoldásnak számít a teljesen konvolúciós neurális hálók és a SIFT algoritmus együttes használata, melyekkel ezeket az eredményeket elértem. Ezen megoldásomnak kiterjeszthetőségét is igazoltam azzal, hogy hiperparaméter optimalizációs algoritmus alkalmazásával továbbfejlesztettem és implementáltam közvetlen kötésű réz kerámiahordozók esetén is.

Abstract

Due to global trends such as electrification, proliferation of smart devices, increasing electrical energy consumption, the expansion of electronics and automotive electronics and the growing popularity of electric cars, special emphasis is being placed on the electronics industry, including the power transistor industry. These electronic devices must deal with more and more heat power, because of their bigger electrical energy consumption, although their surfaces are getting smaller and smaller by the time. This results in a serious technical challenge, because too high temperature is the leading cause of failure. Thus, their thermal examination and identification is extremely important, which can be performed by thermal transient tester (T3Ster) equipment. As a member of ThID team at Robert Bosch Kft. I participated in a project that automated and accelerated T3Ster measurements. Our main profile is thermic identification and with our method we also made T3Ster measurements more competitive and economic. I developed a vision system as a complex electronic device for a robot controlled in a LabVIEW environment. My system can help the robot to find the MOSFETs and other electronic components. After that the robot can automatically measure them. I also developed the hardware and software for the system. I chose a camera with the right sensor and optics and implemented a server platform capable of serving several computers that would control or monitor the measurement at the same time with the help of a Raspberry Pi 4 Model B. In Autodesk Inventor Professional 2022, I designed the tool that supports mechanical integration and then printed it with a 3D printer. In the most emphatic part of my work, I performed semantic segmentation and keypoint detection for robot navigation using artificial intelligence, fully convolutional neural networks, and computer vision algorithms. I implemented the fully convolutional neural networks and created the dataset necessary for the teaching myself. The dataset is made with the camera and optics that are used in real measurements and annotation happened manually. I have studied neural networks comprehensively. Using automated runs, I taught and evaluated networks 21380 times, modifying depth, number of channels, resolution, and batch size. I did these in the open-source Python 3 programming language primarily using the OpenCV, Pytorch and Torchvision libraries. All in all, I created an electronic system with a robust implementation of navigation to the components, that are measured. I achieved 100 µm and 1° accuracy, which was associated with an IoU value of 94% accuracy for semantic segmentation. I tested my system in the real, original measurement environment. So, my work is absolutely applicable for additional scientific and industrial projects. Besides using the Fully Convolutional Networks and SIFT algorithm together, that helped to reach presented results, is a new approach. I also presented the extendibility of my system at the end of my work by adapting it to another problem.

Dolgozat felépítése, bevezetés

Dolgozatom 7 fejezetből és a hozzájuk kapcsolódó függelékekből épül fel. Az 1. fejezet rövid gazdasági kitekintését követően a 2. fejezetben és az ezt kiegészítő függelékekben bemutatom munkám központi automatizált mérésének, a termikus tranziens mérésnek alternatíváit és a munkám szempontjából legjobb alternatíva, a T3Ster mérés elméleti alapjait. Az elméleti alapokat a 3. fejezetben ültetem át a gyakorlatba. A Robert Bosch Kft. ThID csapatában dolgoztam és segítettem a mérésautomatizálás folyamatát. Közös munkánk több lépcsőjét is ismertetem ebben a fejezetben, bemutatom a termikus tranziens mérés módszertanának fejlődését. Feladatom volt egy olyan komplex elektronikai eszköz, egy látórendszer, fejlesztése, amely segítségével egy általunk épített robot képes T3Ster mérést végezni. Ezen eszköz hardverfejlesztését a 4., szoftverfejlesztését az 5. fejezetben tárgyalom. Az egyes hardver eszközök kiválasztásának és megtervezésének folyamatát külön részletezem. A 4.1 fejezetben térek ki a megfelelő szenzor és optika, a 4.2 fejezetben pedig az ideális egykártyás számítógép platform és a mechanikai konstrukció kiválasztására és megtervezésére. A szoftveres működéssel kapcsolatban az 5.1 fejezetben írom le a választott egykártyás számítógéphez tartozó funkciók megvalósítását, az 5.2 fejezetben pedig az egyes optikai hardvererőforrásokhoz elengedhetetlen szoftveres kalibrációt végzem. Ezt követően az 5.3 fejezetben leírom a folyamatot, amely segíti a mérőrobot navigációját, az 5.4 és 5.5 fejezetekben pedig részletesen elemzem és bemutatom az ezt támogató szemantikus szegmentációs és kulcspont kereső technikákat. Végül 5.6 fejezetben matematikailag optimalizálom, az 5.7 fejezetben pedig kiértékelem módszerem. A 6. fejezetben bemutatom a kiterjesztési, továbbfejlesztési lehetőségeket, a záró, 7. fejezetben pedig röviden összefoglalom munkámat.

Munkám egy az autóipari kutatás-fejlesztésben Kaliforniától Japánig elterjedten használt, a Budapesti Műszaki és Gazdaságtudományi Egyetemen kifejlesztett mérés automatizálására fókuszál. A mérésautomatizálást segítő gépi látást, mesterséges intelligenciát és neurális hálókat alkalmazó módszerek ezen a területen még nem vetették meg a lábukat, éppen ezért kapcsolódó publikációból igen kevés található. A téma nyitottságának és újszerűségének ellenére találtam valamelyest iránymutatásra alkalmas, technikámhoz kötődő cikkeket, melyeket az 5.4.2.1 fejezet kiemelt alfejezetében foglaltam össze.

Tartalomjegyzék

1 Az elektronikai komponensek globális trendjei

Napjainkban az aktív elektronikai komponensek globális piaca folyamatos növekedésen megy keresztül. Ennek oka, hogy a technológiai innovációban kulcsfontosságú szerepük van, illetve, hogy az elektronikai eszközök terén folyamatosan növekvő fogyasztói igények kielégítéséhez is nélkülözhetetlenek. Minden jel arra mutat, hogy az elkövetkező 6-7 évben jelentős növekedés várható az aktív elektronikai komponensek, illetve azon belül a MOSFET-ek piacán.

1.1 Aktív elektronikai komponensek piaca

Az [1] szerint 2020 bázisév 273,7 milliárd amerikai dolláros piaci méretéhez képest 2021 és 2028 között mintegy 9,2 %-os összetett éves növekedési ráta várható az aktív elektronikai komponensek piacán. Ez azt jelenti, hogy 2028-ra már ez a növekedés 2020-hoz képest elérheti a 102,2 %-ot, ami 553,4 milliárd amerikai dolláros piaci méretet jelentene (1) és (2) alapján.

$$CAGR = \left(\frac{EV}{BV}\right)^{\frac{1}{n}} - 1 \tag{1}$$

ahol

$CAGR$	az összetett éves növekedési ráta,
EV	a teljes növekedés után i érték,
BV	a teljes növekedés előtti érték,
n	az évek száma.

Így:

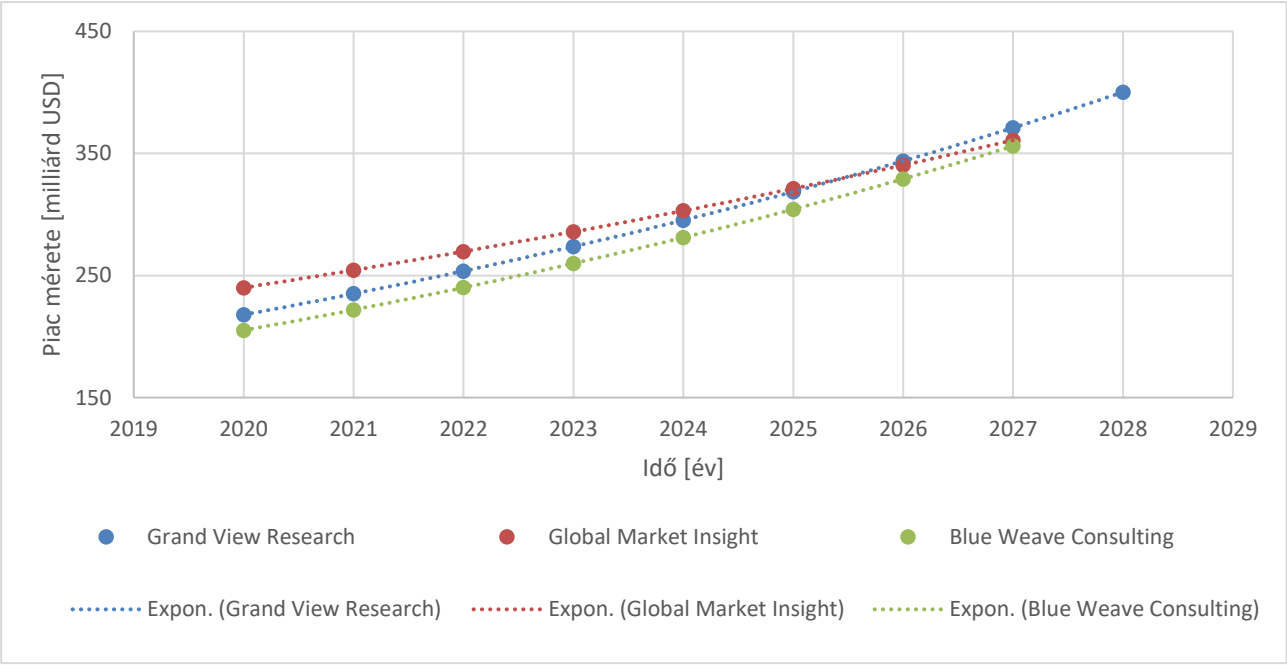
$$EV = (CAGR + 1)^n \cdot BV \tag{2}$$

Hasonló tendenciákat jósol [2] és [3] is. Előbbi egy picit szerényebb 260,015 milliárd dollárról 495,35 milliárd dollárra való növekedést jósol 2020 és 2028 között. Utóbbi pedig a becsült 2019 végi 273,25 milliárd dollárról 2027 végére egy erőteljes 521,11 milliárd dolláros méretet mond. Ezáltal látható, hogy bár a különböző piackutatók kis mértékben különböző méretűnek ítélik meg a piac jelenlegi és jövőbeli méretét is, azonban az egyértelmű, hogy mindenki jelentős növekedésre számít. Erről a tendenciáról szolgál információval az 1. ábra.

1. ábra: Aktív elektronikai konponensek piacának várható méretbeli alakulása különböző kutatások szerint [1][2][3]

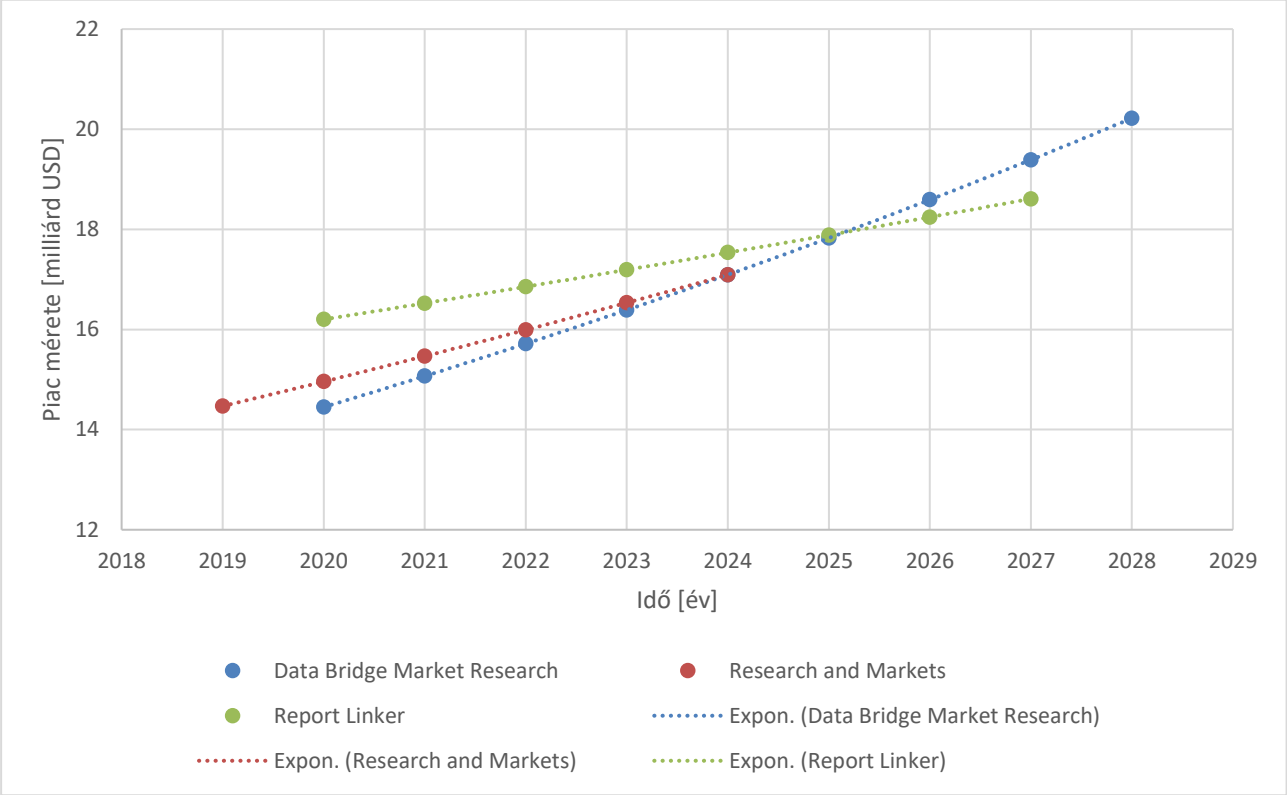
1.2 Autóipari elektronika és teljesítménytranzisztorok piaca

Hasonló trendek figyelhetők meg az autóipari elektronika terén is, ami nem meglepő, hiszen az egyik legkomolyabb felhasználója az aktív elektronikai komponenseknek. Így [4] szerint a 2020-as 217,86 milliárd amerikai dolláros szintről 2028-ra akár 400,27 milliárd dollárra bővülhet a piac. Ennél a 7.9 %-os összetett éves növekedési rátánál 2021 és 2027 között szerényebb becslést ad [5] és bátrabbat jósol [6], 240 milliárdról 360,87 milliárdra 6 %-os és 205,1 milliárdról 356,09 milliárdra 8,2 %-os értékekkel. Ezt a trendet mutatja be a 2. ábra.



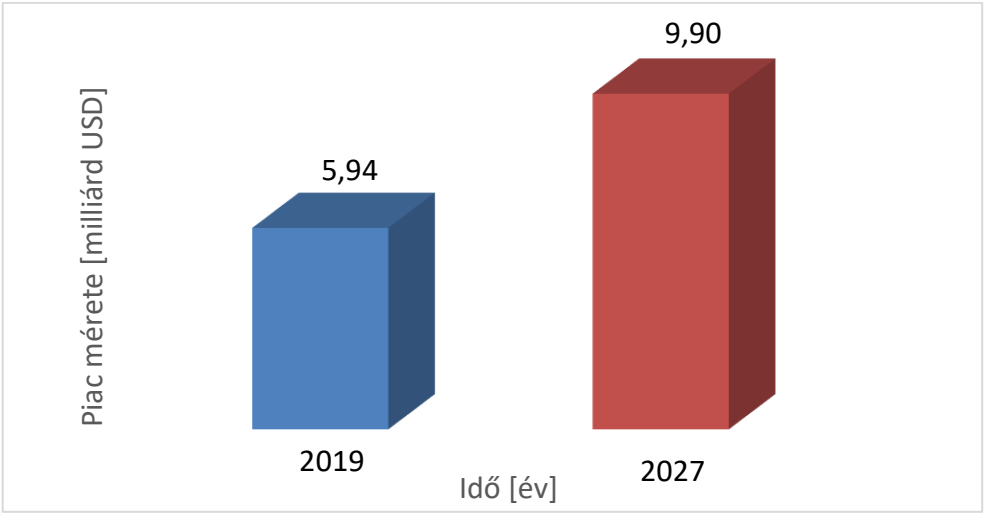
2. ábra: Az autóipari elektronika piacának várható méretbeli alakulása különböző kutatásokban

Jellemzően ezen a területen hangsúlyos a teljesítménytranzisztorok használata. Az aktív elektronikai komponensek közül ez a részpiac is növekvő tendenciát mutat majd a jövőben a gazdasági szakemberek szerint. [7], [8] és [9] arra világít rá, hogy a jelenleg 15-16 milliárd dolláros piac 7-8 éven belül meghaladhatja a 20 milliárd dollárt is, ahogy azt a 3. ábra is mutatja.



3. ábra: Teljesítménytranzisztorok piacának várható méretbeli alakulása különböző kutatásokban

Ezen belül is a [10] alapján kiemelkedő a MOSFET-ek piacán tapasztalt növekedés. Itt a 2019-es 5,43 milliárd dolláros piac a Research and Markets szerint 2027-re 9,90 milliárd dollárosra bővül, ami 6,60 %-os összesített éves növekedési rátát jelent. Ezt illusztrálja a 4. ábra.



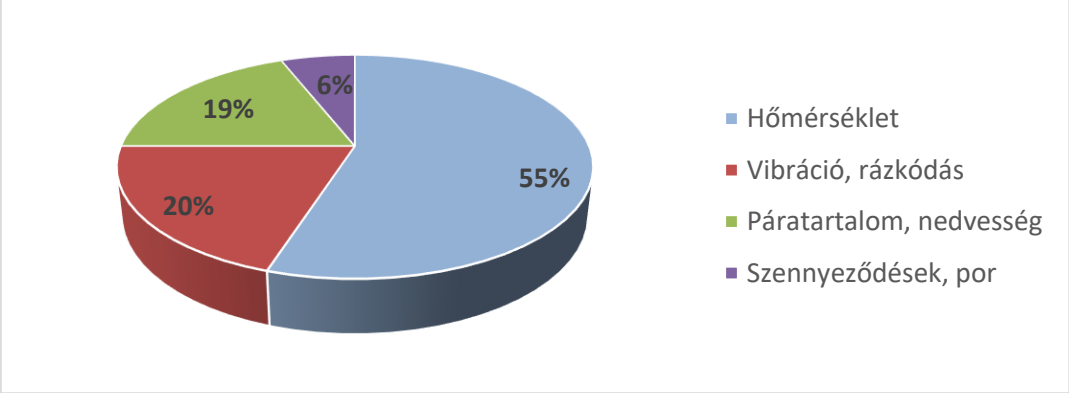
4. ábra: MOSFET-ek piacának várható méretbeli alakulása [10]

2 A teljesítménytranzisztorok termikus vizsgálata

Az 1. fejezetben bemutatott globális trendekből jól látható, hogy a jövő technológiáinak szempontjából különösen fontos a teljesítménytranzisztorok területén végzett kutatás, fejlesztés. Az új, innovatív ipari megoldások sokszor kisebb tranzisztor méretet vagy nagyobb teljesítményt igényelnek. Egységnyi felületre több tranzisztort integrálva, vagy nagyobb teljesítményű tranzisztorokat használva megnő a teljesítmény disszipációs sűrűség. Éppen ezért kiemelt figyelmet kell fordítani a tranzisztorok termikus vizsgálatára. A következőkben indoklom automatizált mérésem szükségességét és bemutatom annak elméleti alapjait. A vázolt problémára nyújt megoldást az úgynevezett T3Ster mérés, amelyet fejlesztett elektronikai eszközömmel gyorsabbá és hatékonyabbá tettem. A termikus tranziensek mérése egy nagyon izgalmas és kihívást jelentő terület mind fizikai leírás, mind méréstechnika szempontjából. Ebben a témakörben sikerült lehetővé tennem egy nagyságrendileg 8-10 órás aktív emberi munka 15 perces konfigurációval és ezt követő automatizált méréssel való kiváltását.

2.1 Elektronikai rendszerek megbízhatóságának összefüggése a hőmérséklettel

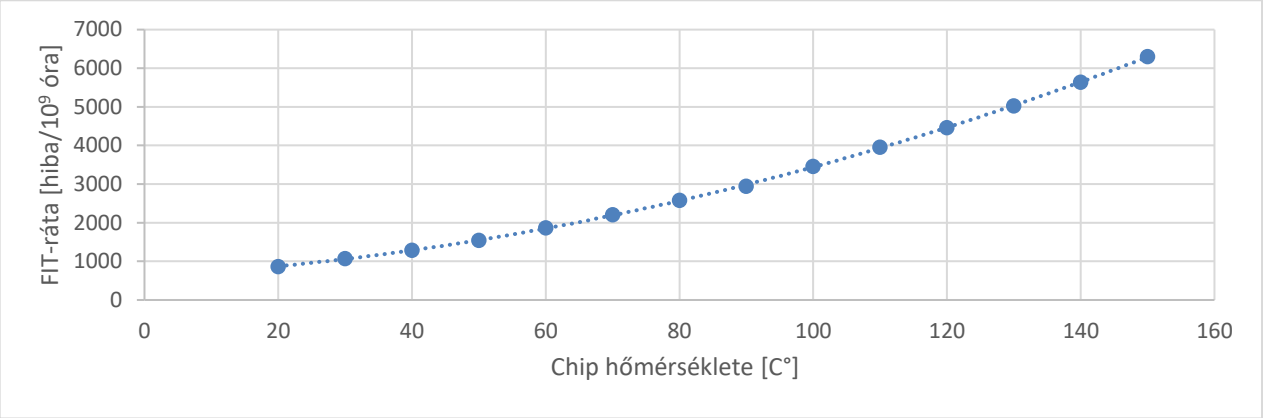
Megbízható elektronikai eszközök fejlesztéséhez elengedhetetlen azok meghibásodásának fő okaival tisztában lenni. Erről nyújt információt az 5. ábra.



5. ábra: Elektronikai eszközök meghibásodásának főbb okai [11]

Ezek alapján látható, hogy a legyakoribb meghibásodást kiváltó ok elektronikai eszközöknél a hőmérséklet, ami több, mint a hibák felét okozza [11]. Ezt követően a vibráció, a rázkódás, a páratartalom, a nedvesség, a különböző szennyeződések és a por jelentősen elmarad. Mindemellett az is megfigyelhető a 6. ábra esetében, hogy ahogy egy komponensnek, az ábra esetén éppen egy MOSFET-nek, egyre inkább megnő a chip hőmérséklete, úgy egyre jobban növekszik a kiesési valószínűsége, azaz FIT-rátája (Failures In Time rate). Következésképpen csökken az eszköz élettartama és egyre kevésbé lesz üzembiztos. Így a megbízhatóság, valamint a gyártáshoz felhasznált alapanyagmennyiség miatt a gazdaságosság és fenntarthatóság egyik nagyon fontos tényezője, hogy a komponensek üzem közben a megfelelő hőmérsékleti tartományban legyenek amellett, hogy ellátják feladatukat a teljes rendszerben. Ennek a tartománynak a meghatározásához

és olyan eszközök fejlesztéséhez, amelyek képesek ebben a tartományban megfelelően üzemelni, mérni kell valamilyen módon a komponenseket érő hőigénybevételt, kvantifikálni kell azt.



6. ábra: MOSFET meghibásodási rátája [12]

2.2 Hőmérséklet meghatározása

Létezik néhány hagyományos módszer is elektronikai komponensek hőmérsékletmérésére. Ezek előnye, hogy viszonylag olcsó és egyszerű megoldási alternatívák, azonban az egyes komponensek belső hőmérsékletéről nem szolgálnak információval. Ebből a szempontból jobb választás lehet a chip hőmérsékletről is információval szolgáló termikus tranziens mérést alkalmazni.

2.2.1 Hőelem használata

Egyszerű megoldás lehet például a hőelemek használata. Két különböző fémet villamosan összekötve, jellemzően hegesztve, a kötést a mérendő felületre helyezve, a különböző fémek mentén más hőmérsékletgradiens eloszlás jön létre, ezáltal köztük villamos feszültség keletkezik. Ez a mért feszültség függ a két vezeték anyagi minőségétől, valamint a hőmérsékletkülönbségtől, amely a kötés és a feszültségmérés helye között fennáll. Így ismerve egy adott vezetékpárra jellemző karakterisztikát, feszültségmérésre vezethető vissza a hőmérsékletmérés.

2.2.2 Hőkamera használata

Másik lehetséges módszer a hőkamera alkalmazása, amely azt használja ki, hogy minden anyag bocsát ki elektromágneses sugárzást töltött részecskéinek hőmozgása okán. Ez a sugárzás alakítható képpé a hőkamera érzékelője által úgy, hogy eltérő színekkel jeleníthetők meg az eltérő hőmérsékletű felületek. Éppen ezért, a felületi mérés miatt nem kielégítő ez a megoldás sem, ahogy a termoelem használat sem. Így nem kapható információ a vizsgált komponens belső hőmérsékletéről.

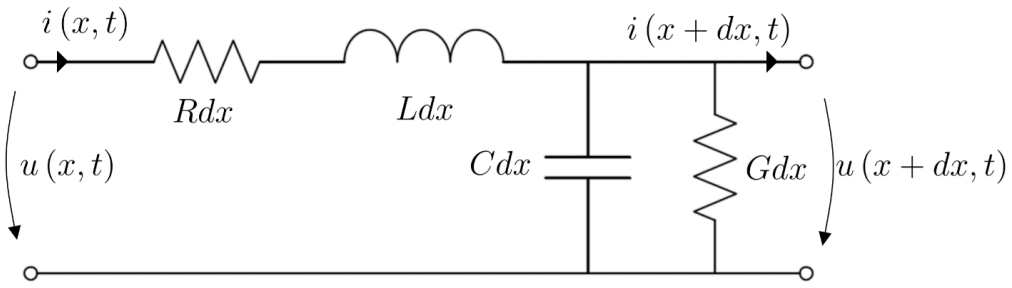
2.2.3 Termikus tranziens mérés, T3Ster használata [13]

A T3Ster [14] (Thermal Transinet Tester) mérés közvetve, megfelelő szoftveres támogatással kiegészítve [15], arról szolgál információval, hogy a hőmérséklet hogyan változik időben a komponens belső szerkezetében.

Éppen ezért működésének megértéséhez mindenképp szükséges ismerni a villamos- és termikus rendszerek közti analógiát.

2.2.3.1 Villamos- és termikus rendszerek közti analógia [16]

A villamos- és termikus rendszerek közti analógia megmutatható egy elemi távvezeték szakasz és egy kizárólag egydimenziós energiaáramlást lehetővé tévő elemi térfogatelem közti hasonlóság ismertetésével.



7. ábra: Elemi távvezeték modell

Egy elemi távvezeték szakasz feszültségének és áramának hely szerinti deriváltja egyszerűen felírható az alábbiak szerint [17]:

$$\frac{\partial}{dx}u(x,t) = -Ri(x,t) - L\frac{\partial}{dt}i(x,t) \tag{3}$$

$$\frac{\partial}{dx}i(x,t) = Gu(x,t) - C\frac{\partial}{dt}u(x,t) \tag{4}$$

ahol a 7. ábra alapján:

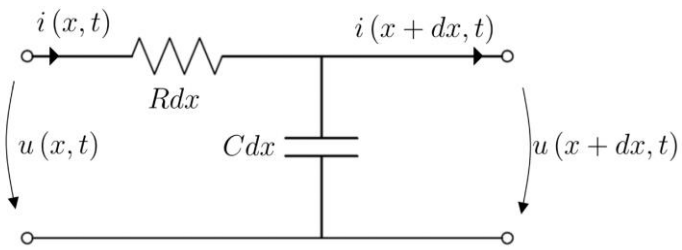
u	a feszültség,
x	a helyparaméter,
t	az idő,
R	az ellenállás,
i	az áramerősség,
L	az induktivitás,
G	a vezetőképesség,
C	a kapacitás.

Sokszor elektronikai számításoknál elosztott RC-tápvonal alkalmazandó, ami azt jelenti, hogy élni lehet $L = 0$ és $G = 0$ peremfeltételekkel. Ezek alapján:

$$\frac{\partial}{dx}u(x,t) = -Ri(x,t) \tag{5}$$

$$\frac{\partial}{dx}i(x,t) = -C\frac{\partial}{dt}u(x,t) \tag{6}$$

Tehát a modell jelentősen egyszerűsödik:



8. ábra: Elemi távvezeték RC-modell

Termikus rendszerek esetén célszerű az energiamegmaradás törvényéből kiindulni. Ekkor x irányban felírható a be- és kilépő hőáram különbsége [18]:

$$\dot{Q}(x) - \dot{Q}(x+dx) = \frac{\partial}{\partial x}\left(-\lambda \frac{\partial}{\partial x} T(x,t) dydz\right) dx \tag{7}$$

ahol a 9. ábra alapján:

$\dot{Q}$ a hőáram,

x, y, z a helyparaméterek,

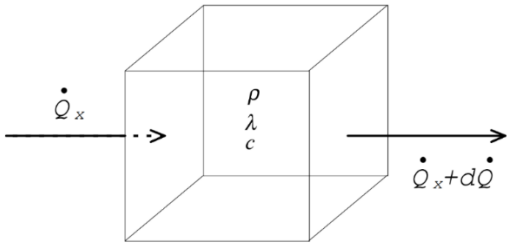
λ a hővezetési tényező,

t az idő,

λ a hővezetési tényező,

T a hőmérséklet.

Mivel jellemzően egydimenziós hőáramlással közelíthető a félvezetők hővezetése, így y és z irányban nem szükséges ezt az egyenletet felírni.



9. ábra: Elemi térfogatelem modell

Felírva a dV térfogat energiamérlegét, feltételezve, hogy forrásmentes, azaz nincs keletkező energia, elmondható, hogy a többlet bemenő energia megegyezik az entalpia megváltozásával, azaz:

$$-\frac{\partial}{\partial x}\left(-\lambda \frac{\partial}{\partial x} T(x,t)\right) dxdydz = \rho c_p dxdydz \frac{\partial}{\partial t} T(x,t) \tag{8}$$

ahol a (7) egyenlet kapcsán nem említettek:

λ a hővezetési tényező,

ρ a sűrűség,

c_p az izobár fajhő.

Itt, ha $dV \neq 0$ és λ hőmérsékletfüggése elhanyagolható:

$$\lambda \frac{\partial^2}{\partial x^2} T(x,t) = \rho c_p \frac{\partial}{\partial t} T(x,t) \tag{9}$$

Ekkor felhasználható az úgynevezett másodfajú peremfeltétel, mely szerint a dV határán a $\dot{q}$ hőáramsűrűség ismert:

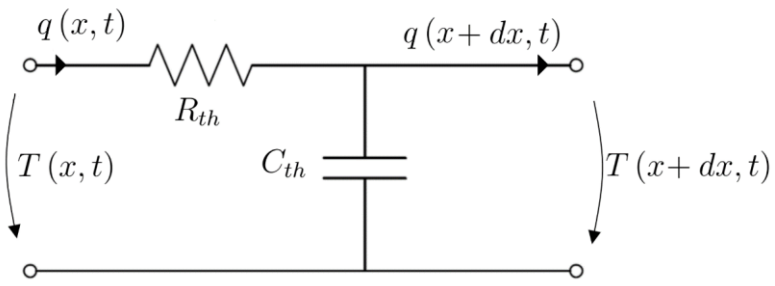
$$\dot{q}(x,t) = -\lambda \frac{\partial}{\partial x} T(x,t) \tag{10}$$

Így (9) és (10) alapján:

$$\frac{\partial}{\partial x} T(x,t) = -\frac{1}{\lambda} \dot{q}(x,t) \tag{11}$$

$$\frac{\partial}{\partial x} \dot{q}(x,t) = -\rho c_p \frac{\partial}{\partial t} T(x,t) \tag{12}$$

Így már szembetűnő az (5), (6) és a (11), (12) egyenletek közötti hasonlóság. Látható az analógia. A hőmérsékletkülönbség feleltethető meg a potenciálkülönbségnek, a hővezetési tényező reciproka az ellenállásnak, a hőáramsűrűség az áramerősségnek, a sűrűség és az izobár fajhő szorzata pedig a kapacitásnak. Így a termikus problémák visszavezethetők RC-modellek vizsgálatára.



10. ábra: Termikus RC-modell

2.2.3.2 Az elosztott paraméterű rendszer vizsgálata [13]

A fent bemutatott RC-hálózatokkal modellezhetők az elektronika kapcsán vizsgálandó félvezető eszközök, amelyek lényegében ebben az esetben termikus rendszerek. Ahogy arra a 9. ábra kapcsán is utaltam, sokszor a hővezetés egydimenziósnak tekinthető, a többi irányban elhanyagolható. Ez előnyös, hiszen így lehetőség nyílik minden egyes, az elektronikai eszközön belüli, hőtanilag különböző réteg vizsgálatára. Itt fontos kiemelni, hogy egy elosztott rendszerről van szó, azaz a térben is külön elhelyezkedő rétegeknek, más és más hőellenállása és hőkapacitása van. Ebből következőleg az elosztott paraméterű rendszerben az egyes RC-tagoknak más-más időállandója lesz. Ekkor úgynevezett Foster-modellel vagy Cauer-modellel írható le a rendszer.

A mérés matematikai alapjai [19]

Egy ilyen elosztott paraméterű RC rendszer egy Dirac-delta gerjesztésre a súlyfüggvényt, másként mondva Green-függvényt adja válaszként. Ebből már bármilyen gerjesztésre származtatható a válaszfüggvény a konvolúciós integrál segítségével:

$$T(t) = \int_0^\infty W(y)P(t-y)dy \tag{13}$$

azaz:

$$T(t) = W(t) \otimes P(t) \tag{14}$$

ahol:

T	a hőmérséklet,
t	az idő,
W	a súlyfüggvény,
P	a gerjesztés függvénye,
y	a változó.

Igazából a súlyfüggvény, amely a rendszer karakterisztikáját jellemzi, nem ismert. Éppen ezt kell meghatározni, így nem konvolúcióra, hanem dekonvolúcióra van szükség, ahogy az a (15) egyenletben látható:

$$W(t) = T(t) \otimes^{-1} P(t) \tag{15}$$

Azonban a Dirac-delta gerjesztés gyakorlatban nagyon nehezen megvalósítható. Éppen ezért lehet egy jobb megoldás a sokkal könnyebben kivitelezhető egységugrás függvény, amelyre a válasz az átmeneti függvény, ahogy azt a (16) egyenlet mutatja:

$$a(t) = W(t) \otimes h(t) \tag{16}$$

ahol a (13) és (14) kapcsán nem említett:

a	az átmeneti függvény,
h	egységugrás függvény.

Így:

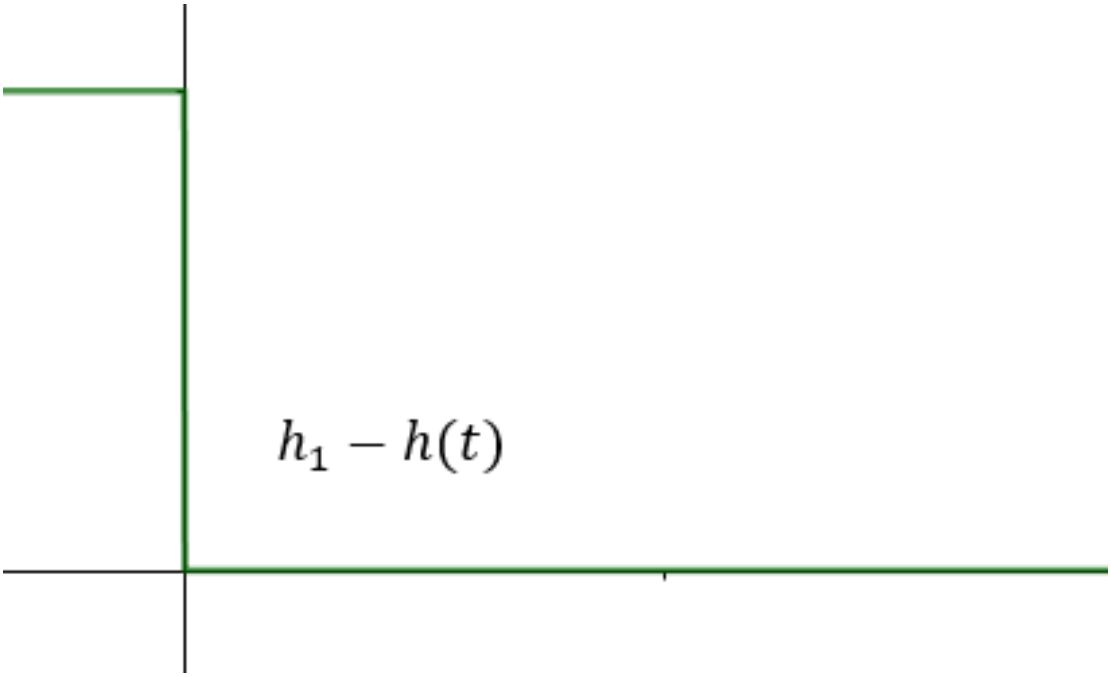
$$a(t) = \int_{-\infty}^\infty W(y)h(t-y)dy \tag{17}$$

$$a(t) = \int_0^\infty W(y)dy \tag{18}$$

Ahonnan:

$$W(t) = \frac{d}{dt} a(t) \tag{19}$$

Tehát ennek a módszernek a legfőbb előnye a Dirac-deltára adott súlyfüggvény, mint válasz mérésével szemben, hogy az egységugrás gerjesztés könnyebben kivitelezhető. Ez egy jobb megoldás, azonban gyakorlatban célszerű lehet nem egységugrás függvénnyel, hanem annak transzformáltjával dolgozni. Ennek oka, hogy az egységugrás esetén az „ugrás" utáni konstans teljesítmény tartása nem egyszerű feladat. Egész pontosan $h_1 - h(t)$ függvény használható, ahol h_1 konstans és értéke $h(\tau)$, úgy, hogy $\tau > 0$. Ezt a függvényt, amelyet akár lehetne „egység leugrás" függvénynek is nevezni, a 11. ábra szemlélteti.



11. ábra: „Egység leugrás" függvény

Így összefoglalva az előző gondolatokat elmondható, hogy komponensek T3Ster eszközzel történő karakterizációja során egy konstans teljesítmény érték lekapcsolásánál szükséges figyelni a csökkenő hőmérsékletet, mint választ. A hűlő eszköznek ez a válasza elektromos tulajdonság alapján mért. Minden félvezető rendelkezik több úgynevezett TSP (temperature sensitive parameter), hőmérséklet-érzékeny paraméterrel. Egy egyszerű dióda esetén ez például a nyitóirányú feszültség. A T3Ster ezt a feszültséget méri nagy pontossággal és mintavételezéssel, a hőmérséklet pedig ez alapján szoftveresen számított. Tehát elmondható, hogy a mért $a(t)$ válasz alapján határozható meg a vizsgált R-C rendszer karakterisztikája. A tokon belüli hőellenállások és hőkapacitások eloszlása, azaz a chip és a teljes komponens struktúrája így vizsgálható. Részletesebb leírás a félvezetők TSP értékéről az F1 függelékben, az átmeneti- és struktúrafüggvényről az F2.1-ben és F2.2-ben található.

3 Termikus tranziens méréstechnika fejlődése automatizáltság szempontjából

Ahogy a bevezetésben is említettem, csapatommal a 2.2.3 fejezetben bemutatott mérést, termikus karakterizációt végeztük és végezzük. Ahogy arról az 1. fejezeteben írok, ez mind gazdasági, mind technológiai szempontból egy előremutató és intenzív fejlesztést igénylő mérnöki tevékenység. A versenyképesség növeléséhez és az innovációhoz mindenképpen a folyamatok automatizálására van szükség, nem csupán a gyártás, de a kutatás és fejlesztés területein is. Ahhoz, hogy korszerű módon, jó minőségű modelleket lehessen alkotni az egyes elektronikai eszközökről, komponensekről, bizonyos termékfejlesztéshez szükséges modellidentifikációs folyamatokat érdemes robotizálni. Ezzel csökken az emberi hibafaktor és növekszik az egységnyi idő alatt elvégezhető mérések száma. Így nagyobb lesz a hozzáadott érték és gazdaságosabb lesz a folyamat.

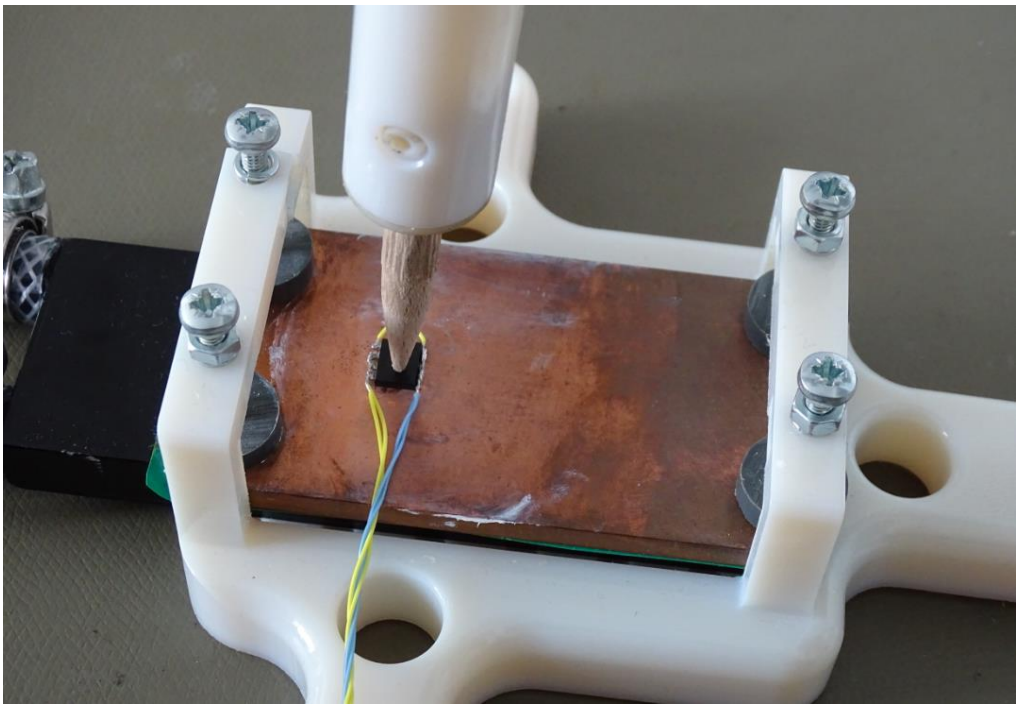
3.1 Mérési módszertan fejlődése

Csapatommal a 3. fejezet bevezetésében leírtaknak megfelelően folyamatosan igyekszünk megújulni, innovatív technológiákkal minél versenyképesebb termékeket alkotni. Ezt tesszük a termikus tranziens mérésünk automatizálásával is. A 2.2.3 fejezetben bemutatott folyamatot és a hozzá szükséges elektronikai eszközt, a T3Ster-t a Budapesti Műszaki és Gazdaságtudományi Egyetemen, a Villamosmérnöki és Informatikai Karon, az Elektronikai Eszközök Tanszéken fejlesztették ki. Mára az egész világon alkalmazzák ezt a módszert integrált áramkörök és diszkrét alkatrészek tokozásának roncsolásmentes strukturális vizsgálatára és termikus szempontból való optimalizálására. Azonban legjobb tudásom szerint, amikor ezt a munkát írom még egyik ipari résztvevő sem állt elő sem itthon, sem külföldön ezen mérés többféle komponens vizsgálatának automatizálását célzó, számítógépes látásra épülő projekttel. Így ez a munka még a kutatás-fejlesztés terén is úttörőnek számít. A következő pontokban ezen fejlesztés gondolatmenetének, kialukálásának lépéseit mutatom be.

3.1.1 Manuális mintapreparálás forrasztással és vezetékekkel

Az első módszer, amit használtunk az egyes MOSFET-ek méréséhez, egy folyamatos aktív emberi munkát és jelenlétet igénylő megoldás volt. Az egyes alkatrészekre forrasztás segítségével vezetékeket erősítettünk és ezen keresztül fűtöttük az egyes eszközöket, majd mértük a válaszjelüket is. Ezt természetesen mind a T3Ster segítségével tettük. Itt az apró és sokszor nehezen hozzáférhető MOSFET-lábak igen körülményessé tették a mintapreparálást. Ügyelni kellett rá, hogy a villamos vezető forrasz ón a mérés közben ne érjen hozzá a földpotenciálon lévő hűtést megvalósító rézlaphoz, valamint értelemszerűen két egymástól villamosan elszigetelt láb vezetékeinek forrasz anyaga sem kontaktálhatott egymással. Ráadásul ezt a mintaelőkészítést nem csupán olyan nagynak mondható házú MOSFET-ek estén végeztük el, mint mondjuk egy DPAK, vagy egy

D2PAK, hanem például, mint az LFPAK88, LFPAK56, vagy az LFPAK56D. Itt természetesen a preparálási idő nehezen kiszámítható, mivel nehezen lehet felforrasztani az adott vezetékeket az adott apró lábakra. Valamint miután ez sikerült, azután is különösen nagy figyelmet kellett fordítani arra, hogy ne történjen nem kívánt zárlat bármely két anyag között, illetve ne következzen be mechanikai roncsolódás. Egy ilyen előkészítést követően a szabad hőáram útját biztosító rézlapra lehetett helyezni a felvezetékezett félvezető eszközt, szükség esetén hővezető pasztával. Ezután egy rugós leszorítóval biztosítani lehetett a szoros érintkezést a hűtő felülettel és indulhatott a T3Ster mérés.

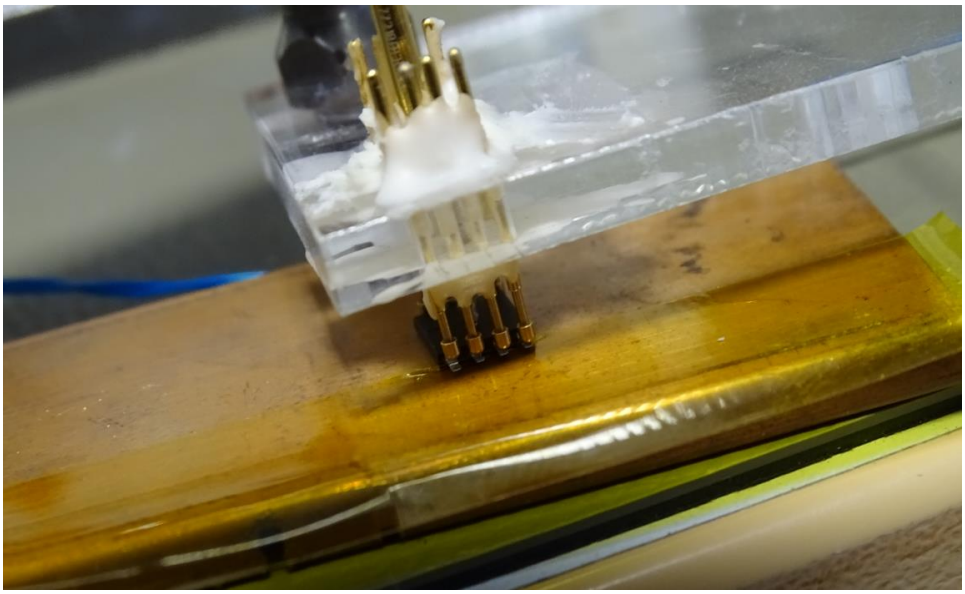


12. ábra: Manuálisan preparált MOSFET mérése

Minden egyes MOSFET-et így kellett előkészíteni a méréshez megfelelve a JEDEC (Joint Electron Device Engineering Council) szabványainak, például a JESD51-1 [21] és JESD51-14 [22] szabványoknak, ami igen hosszadalmas és fárasztó munka volt. Felmerült bennünk a gondolat, hogy jó lenne valahogy ezt időhatékonyabban csinálni, automatizálni a folyamatot. Fontosnak láttuk elkerülni a nehezen kiszámítható időigényű, minden minta esetén szükséges forrasztást. Továbbá hosszú távon növelni szerettük volna a versenyképességünket, azzal, hogy egységnyi idő alatt több mintát tudjunk mérni. Így találtunk ki egy olyan módszert, amely segítségével az egy eszközre jutó mintapreparációs idő lecsökkenthető.

3.1.2 Tűágy használata

A 3.1.1 fejezet végén említett módszer lényege, hogy ne kelljen minden alkalommal, minden új mintánál elvégezni a vezetékek lábakra forrasztását, kontaktálások ellenőrzését. Így került kialakításra a 13. ábra szerinti tűágy.

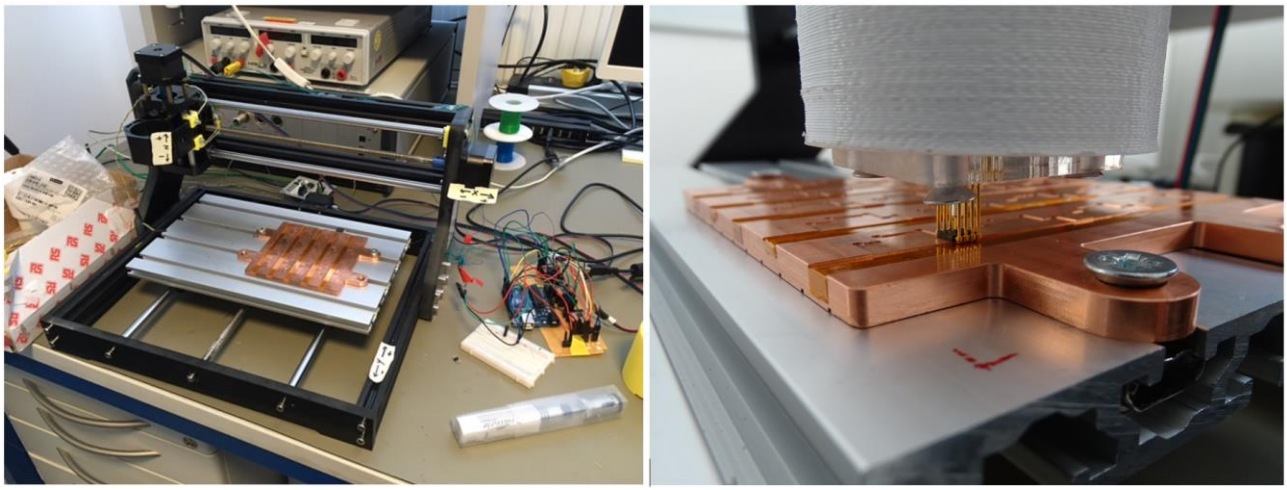


13. ábra: Tűágy használata

Ennek segítségével fenti irányból lehetséges villamosan vezető tűk segítségével kontaktot kialakítani az egyes mérendő eszközökkel. Ezáltal csak a tűágy elkészítése időigényes, de ez az idő megoszlik az összes mérendő minta között. Így összességében egy minta bruttó mérési ideje kevesebb. Ez jó, azonban még ez is igényel emberi jelenlétet, hiszen amikor vége van egy mérésnek, akkor ki kell cserélni a mintát a tűk alatt. Természetesen ez is időbe telik és nem tesz lehetővé például olyat, hogy éjszaka is mehessen számos mérés felügyelet nélkül. Ezért érkeztünk el arra a pontra, hogy egy robotizált rendszer kialakítását céloztuk meg.

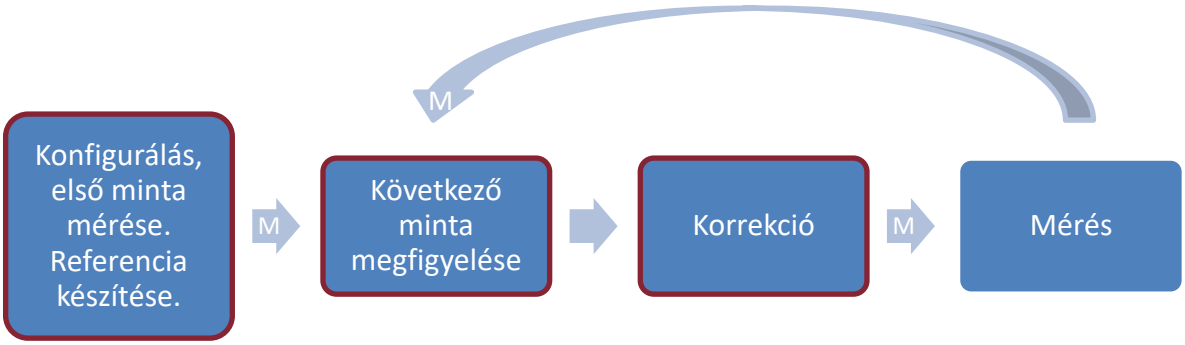
3.1.3 Robot készítése

Ez az intelligens rendszer egy rövid konfiguráció után automatikusan mérő robot és a hozzá tartozó hardver- és szoftverkeretrendszer. Ennek a keretrendszernek része egy speciális hűtő rézlap, amin külön helyek vannak kialakítva többek között a 3.1.1-ben felsorolt komponens házaknak, egymástól fix távolságra.



14. ábra: Mérés robot segítségével

Az első komponens fölé kell csak beállítani a robotot, ez a konfiguráció része, utána pedig automatikusan működik a rendszer. Az egészet egy átalakított CNC 3018 Pro marógép valósítja meg. Ez úgy került átépítésre, hogy a gravírozó motorjának és szerszámának helyére került be a mérést végző tűágy. A robot vezérlése LabVIEW környezetben állapotgép modell alapján történik. Erről illusztráció a F3 függelékben található. Itt számítja ki a robot, hogy melyik irányba hány lépést haladjanak léptető motorjai, melyek 100 µm pontossággal képesek különböző mozgásokat megvalósítani. A fix MOSFET-helyekkel a vezérlés egyszerűnek tűnhetne, azonban valójában a rézlapon kialakított fix helyekből elcsúszhatnak a komponensek és egy minimális elcsúszás is problémát jelenthet, hiszen a tűágy tűi az akár néhány tized milliméter széles lábakkal kontaktálnak. Ilyen elcsúszás például azért is történhet, mert a vízszintes síkban a 14. ábra szerinti vertikális elmozdulás esetén igazából a rézlap mozdul el, nem pedig a mérést végző tűágy, eközben pedig vibráció léphet fel. Így kell egy rendszer, ami intelligens módon és precízen inputot képes adni a robotnak a mérendő eszközök hollétéről. Cél, hogy ennek pontossága a 100 µm pontossággal közel megegyező legyen. Ezt a feladatot én vállaltam. Ki kellet alakítanom egy látórendszert, amely ki tudja szolgálni a LabVIEW-kódot korrekciós adatokkal.



15. ábra: Termikus tesztelés folyamata

Ahogy az a 15. ábrán is látható, a látórendszer a robotirányítási folyamat jelentős lépéseit befolyásolja. A bordó színű keretezéssel jelölt lépésekre van hatással. A robot mozgását az „M” betűvel jelölt nyilak reprezentálják. A konfigurálás az első minta mérése előtt történik. Ekkor kézi rávezetéssel a tűágy kontaktál az első komponenssel. Innentől kezdve autonóm módon működik a rendszer. Miután sikeresen megtörtént az első mérés, a robot fix offset-tel elmozdul, hogy elkészíthesse a fotót a jól lemért referencia objektumról. Ezt követően ciklikusan mindig a soron következő minta fölé viszi a kamerát és arról is képet készít. Ezen két kép összevetése alapján tudja meghatározni fejlesztett szoftverem az előző jól lemért mintához képest szükséges pozíciókorrekciót. Ezt a korrekciós mozgást is elvégzi a robot az offset-en túl, majd pedig méri a mintát.

4 Látórendszerhez szükséges eszközök, hardverek meghatározása, kiválasztása

Feladatom tehát egy olyan rendszer elkészítése volt, amely képes kameraképek alapján elmozdulást számolni két elektronikai komponens, mint célobjektum között. Mivel igen kis méretű alkatrészek mérésének automatizálását szolgálja a rendszer, így a minimális alapelvárás 500 µm pontosság volt. Azonban az igazán nagyszerű, hosszútávon működő és robosztus megoldásnak a 100 µm-es léptetőmotoros pontosság megközelítése lett kitűzve. Jelenleg még nem képes a robot tűágya a függőleges tengelye körüli elforgásra, azonban hosszútávon ez is cél az ilyen módon esetleg elfordult komponensek miatt. Így a látórendszernek ezt a paramétert is számítania kell.

4.1 Szenzor és optika kiválasztása [23]

A megfelelő látórendszer elemek kiválasztásához célszerű összefoglalni az induló paramétereket. A megfigyelés alá vetett objektum látószöge becsülhető zérusnak, hiszen a 3.1.3 fejezetben említett offset-nek köszönhetően elérhető, hogy a robot épp a komponensek fölé vigye a kamerát és ott megálljon. Így az egyes komponensek mindig a látórendszer optikai tengelye mentén helyezkedjenek el. A legkisebb megfigyelni kívánt méretet célszerű a kívánt pontosságnak választani, azaz 100 µm-nek. A tárgytávolság nem egy fix érték, előfordulhat, hogy plusz hőcserélő kerül az aktuálisan mért komponens rézlapja alá, vagy esetleg egy plusz hűtőbordákkal felszerelt test. Kezdeti becslésnek a tárgytávolság nagyságát így 20 és 100 mm közöttinek feltételeztem. Előbbire egy igen nagy hőcserélő elhelyezése esetén lehetne csak szükség, de vélhetőleg ez bőséges alulbecslése a tárgytávolságnak. Utóbbi akkor fordulhat csak elő, ha a robot teljesen felemelkedve, messziről szemléli a munkateret. Így elmondható, hogy ezen határok megválasztásával egy flexibilis, több funkcióra képes és igen sok igényt robusztusan kielégítő megoldást céloztam meg. A legkisebb megfigyelni kívánt mérethez hasonlóan szintén a kicsiny megfigyelni kívánt komponensek okán jó döntés kisebb, például nagyságrendileg 2,5 µm-es pixelméretet választani. Ahogy a széles tárgytávolság tartomány, úgy ez is nehezíti az eszközválasztást. Előbbi azért, mert az optika fókusztávolságának széles tartományban állíthatónak kell lennie, utóbbi pedig azért, mert ezzel számos szenzor kiesik a választható lehetőségek közül. Éppen ezért a számítás lényegében végezhető iteráció alapú megközelítéssel. Ha az adott pixelmérethez nem található igazán jó megoldás, akkor picit azt módosítani lehet. Az itt tárgyalt mennyiségek összegzése az 1. táblázatban található.

1. táblázat: Szenzor- és optikaválasztás kiinduló paraméterei

Mennyiség	Jelölése	Értéke, értéktartománya
Látószög	φ	$0°$
Előjelhelyes tárgytávolság	s	$[-100\ mm; -20\ mm]$
Legkisebb megfigyelendő méret	x	$0{,}1\ mm$
Pixelméret	px	$2{,}5\ \mu m$

4.1.1 Fókusztávolság meghatározása

Fontos információ, hogy a pixelméret megválasztásánál a pixel melyik méretét kell meghatározni. Jellemző kameraválasztási alapelv, hogy ha elsősorban fix állású, vízszintes vagy függőleges orientációjú az objektum, akkor a téglalap alakú pixel egyik oldalát lehet pixelméretnek tekinteni. Abban az irányban lényeges a pixelméret, amelyre merőlegesen haladnak a legkisebb megfigyelendő képrészletek. Azonban, ha nem fix az objektum orientációja, akkor a biztonság irányába eltérve, a pixelátlót célszerű jellemző méretnek tekinteni. Jelen alkalmazás esetében is így kell tenni, hiszen ahogy arról a 4. fejezet bevezetésében is szó volt, elforgatott objektumok detektálása is szükséges. [24]

Ilyen módon általánosságban, ha csak tárgyakról kép felvétele a cél, akkor ésszerű legalább 2 pixelen ábrázolni a legkisebb méretet. Egy élátmenet vagy egy határfelület meghatározása már akár így is működhet, azonban jelen esetben a robot irányításához bizonyos jellegzetességeket is jó lehet megfigyelni ezen a kicsiny alkatrészen, így a legkisebb méretet tartalmazó pixelek számát legalább 5 értékűre választottam, de nem vetettem el azt sem, hogy akár 15 pixelen láthassam ezt a méretet igen kicsiny komponensek esetén. Ekkor:

$$5 \leq a \leq 15 \tag{20}$$

Amelyből a 2,5 µm pixelméret okán:

$$12{,}5\ \mu m \leq a \cdot px \leq 37{,}5\ \mu m \tag{21}$$

ahol:

a	azon pixelek száma, amin x tárgyméret képe látható,
$a \cdot px$	x képének mérete.

Innen számítható az N nagyítás:

$$N = \frac{a \cdot px}{x} \tag{22}$$

Amivel pedig meghatározható az s' képtávolság:

$$s' = N \cdot |s| \tag{23}$$

Így pedig megkapható az f fókusztávolság.

$$f = \frac{1}{\frac{1}{s'} - \frac{1}{s}} \tag{24}$$

Figyelembe véve x képének mérettartományát, illetve a lehetséges tárgytávolságok tartományát a fókusztávolságra is egy tartomány adódik. Ezt az F4 függelék 5. táblázata tartalmazza, jól látható, hogy igen széles.

4.1.2 Kamerarendszer kiválasztása

A 4.1.1 fejezet foglalta össze a látórendszer hardverválasztásának 0. lépését. Ezt követően kezdődik a kamerarendszer kiválasztásának iterációja. Mivel konkrét termékek közül kell választani, így az egyes paraméterek esetében jellemzően diszkrét értékek, vagy értéktartományok adottak. Azaz egyik paraméter sem választható meg tetszőleges igények szerint egy folytonos értékkészletből. Tehát pontosan az F4 függelékben található 5. táblázat fókusztávolság tartományát kielégítő optika és pontosan a 4.1 fejezet felsorolása szerinti pixelméretű, az aktuálisan vizsgált teljes komponens megfigyeléséhez kellően nagy felbontású szenzor nem található. Így több lehetséges szenzort és optikát is megvizsgáltam, mind műszaki, mind gazdasági szempontból. Itt azt az eszközrendszert mutatom be, amely végül kiválasztásra került, de az F5 függelékben felsorolás szintjén más, műszakilag megfelelő alternatívát is megemlítek.

Az 5. táblázatnak elsősorban az alsó sorait, azon belül is legfőképp a bal oldali oszlopait érdemes szemügyre venni. Ennek oka, hogy értelemszerűen jó lenne minél több pixelen látni a vizsgálandó méretet és ahogy arra a 4.1 fejezetben utaltam, a nagyon kicsi tárgytávolságok kevésbé relevánsak. Ehhez a körülbelül 8 – 27 mm fókusztávolság tartományhoz jó választás lehet a Waveshare Electronics 8-50mm Zoom Lens optikája [26]. Ahogy az a termék nevéből is egyértelmű, a fókusztávolsága 8 és 50 mm között állítható, valamint 3 megapixeles felbontásra képes. Ezzel a fókusztávolság tartománnyal inverz módon kiszámítható a tárgytávolság és a nagyítás, amely segítségével ismerve a pixelméretet ellenőrizhető, hogy kellő mennyiségű pixelre esik-e a legkisebb tárgyméret képe. Azonban ezzel két probléma is van. Egyrészt a pixelméret még nem fix érték, a kiválasztandó szenzortól függ. Másrészt pedig az optikának is van térbeli kiterjedése, így az is távolabbi pontra kényszeríti a szenzort a megfigyelt tárgytól. Az első probléma egyes szenzorok összehasonlító vizsgálatával oldható meg, a második pedig úgy, hogy ha az objektív elejétől mért távolság és a fókusztávolság függvényében vizsgálom az a paraméter értékét és nem a tárgytávolság és a fókusztávolság függvényében. Ehhez első lépésként a (24) egyenlet alapján:

$$s' = \frac{s \cdot f}{s + f} \tag{25}$$

Továbbá (23) alapján:

$$N = \frac{s'}{|s|} \tag{26}$$

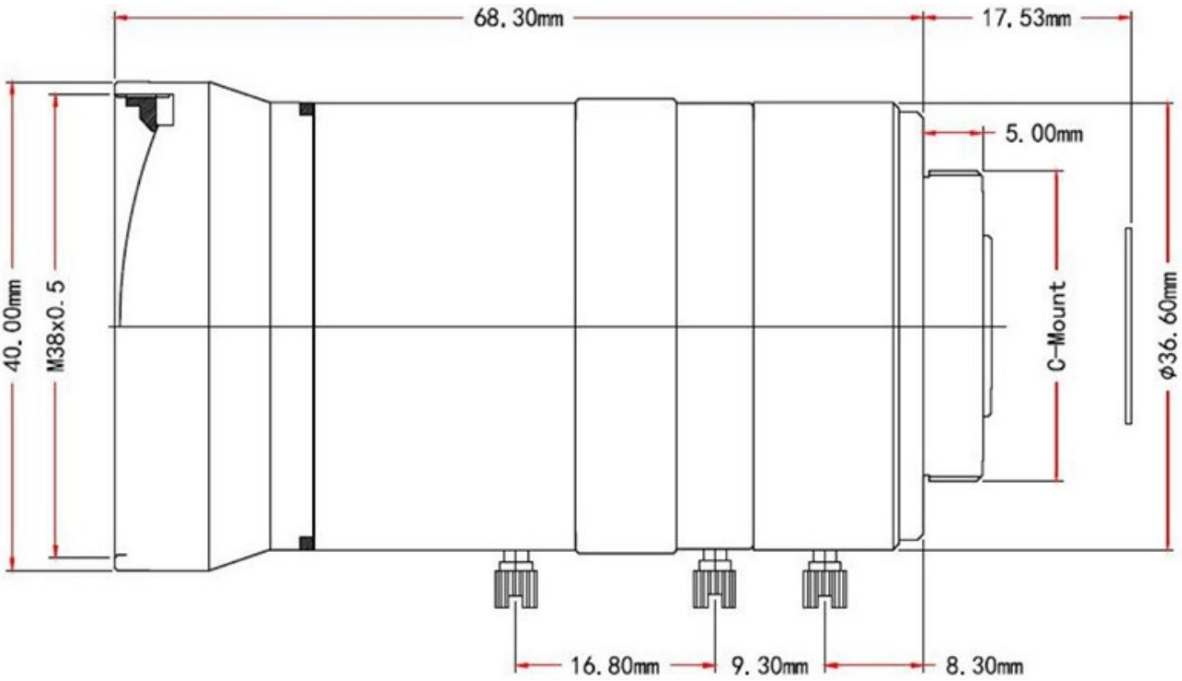
Illetve (22) szerint:

$$a = \frac{N \cdot x}{px} \tag{27}$$

Összevonva (25), (26) és (27) egyenleteket:

$$a = \frac{x}{px} \cdot \frac{1}{|s|} \cdot \frac{1}{\frac{1}{f} + \frac{1}{s}} \tag{28}$$

Itt lényeges kitérni az optika méreteire. Ebben segít a 16. ábra.



16. ábra: Waveshare Electronics 8-50mm Zoom Lens [26]

Legyen a teljes hossz jelölése a 16. ábra alapján:

$$d = 68{,}30\ mm + 17{,}53\ mm = 85{,}83\ mm \tag{29}$$

[25] alapján a csomópont megkapható, a szenzortól a fókusztávolság felmérésével az optikai tengely mentén és innen tovább mérhető a tárgytávolság. Ez azért van így, mert az optikai rendszer tárgy-, és képtere azonos törésmutatójú és így a csomópontok és a főpontok egybeesnek. Így már kiszámítható a tárgytávolság nagysága az objektívtől mért távolság, a teljes objektívhossz és a fókusztávolság függvényében.

$$|s| = |s_o| + d - f \tag{30}$$

ahol, az idáig nem jelölt:

s_o a tárgy objektívtől mért távolsága.

Előjelhelyesen:

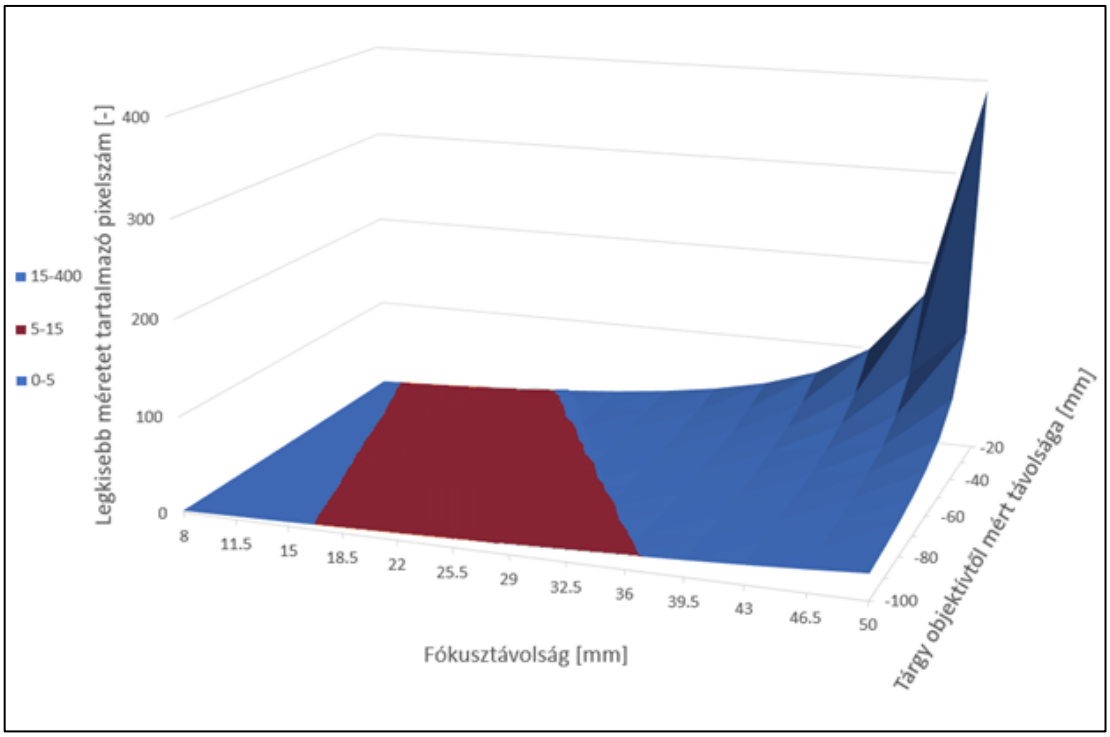
$$s = s_o - d + f \tag{31}$$

Ez pedig behelyettesíthető (28)-ba, így:

$$a = \frac{x}{px} \cdot \frac{1}{|s_o - d + f|} \cdot \frac{1}{\frac{1}{f} + \frac{1}{s_o - d + f}} \tag{32}$$

Azaz az említett két problémából már csak a megfelelő szenzor kiválasztását kell megoldani. Egy ilyen alkalmas választás a Sony IMX477 1,55 µm-es függőleges és vízszintes pixelméretével, valamint 12,3 megapixeles felbontásával [27]. Ez 2,2 µm-es diagonális pixelméretet és az optikáénál nagyobb felbontást jelent, tehát ebből a szempontból nem ront a konstrukción. Ezekkel az adatokkal vizsgálva $a(s_o, f)$ függvény az F4 függelék 6. táblázatában kerül leírásra.

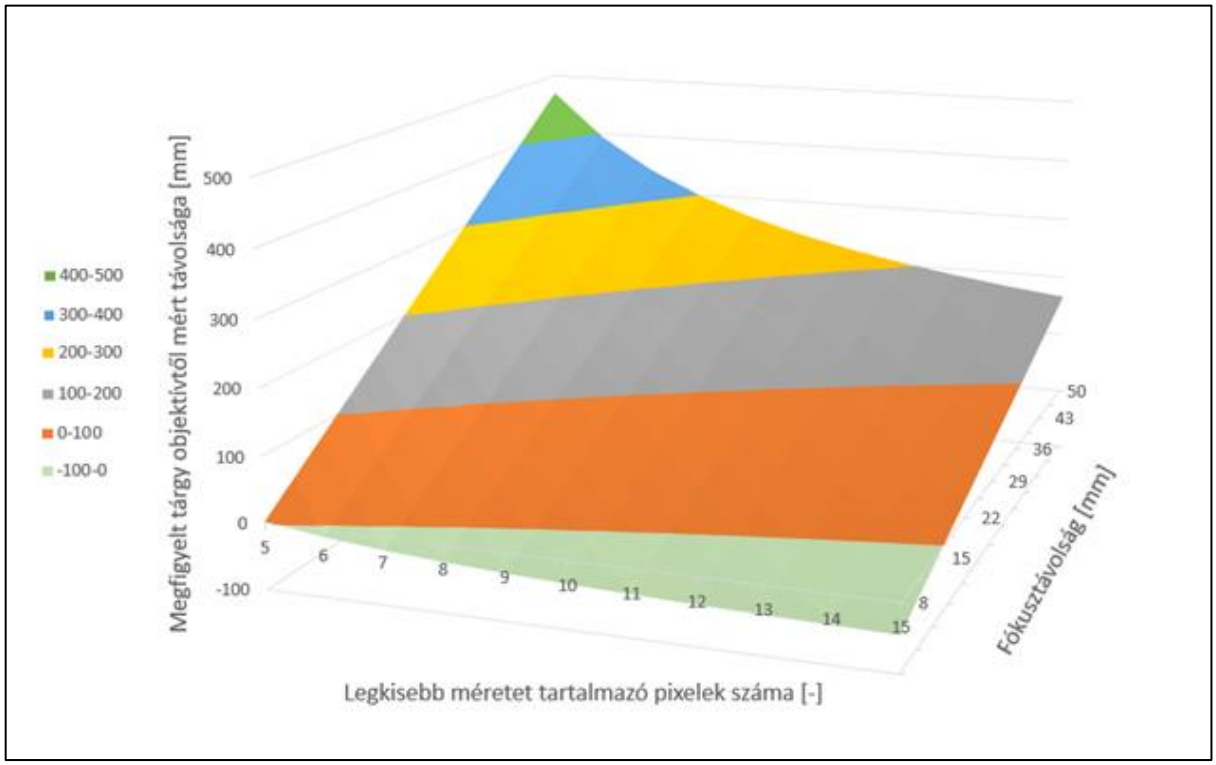
Ez még inkább szemléletes 3 dimenziós felületen bemutatva. A túl kevés pixelen való megfigyelés eredménye a pontatlan robotnavigáció lenne, míg a túl sok pixelen ábrázolás azt eredményezné, hogy a teljes komponens nem férne bele a képbe, így túl kevés lehetne a szolgáltatott információ.



17. ábra: Különböző objektívtől mért távolságokhoz és fókusztávolságokhoz tartozó legkisebb méretet tartalmazó pixelszám

A 17. ábra jól mutatja, hogy bármely objektívtől mért távolság esetén található olyan fókusztávolság beállítás, amely megfelelő pixelszámot eredményez. Ezen kedvező beállítások tartományát a bordó színnel jelölt terület jeleníti meg. Így a 4.1 fejezetben említett flexibilis, több funkcióra képes és igen sok igényt robusztusan kielégítő megoldást sikerült megvalósítanom. Ez a flexibilitás jól megfigyelhető az F4 függelék 7.

táblázatában és a 18. ábra esetén is, ahol látható, hogy milyen objektívtől mért távolságok érhetők el különböző fókusztávolság és legkisebb méretet tartalmazó pixelszám beállításokkal.

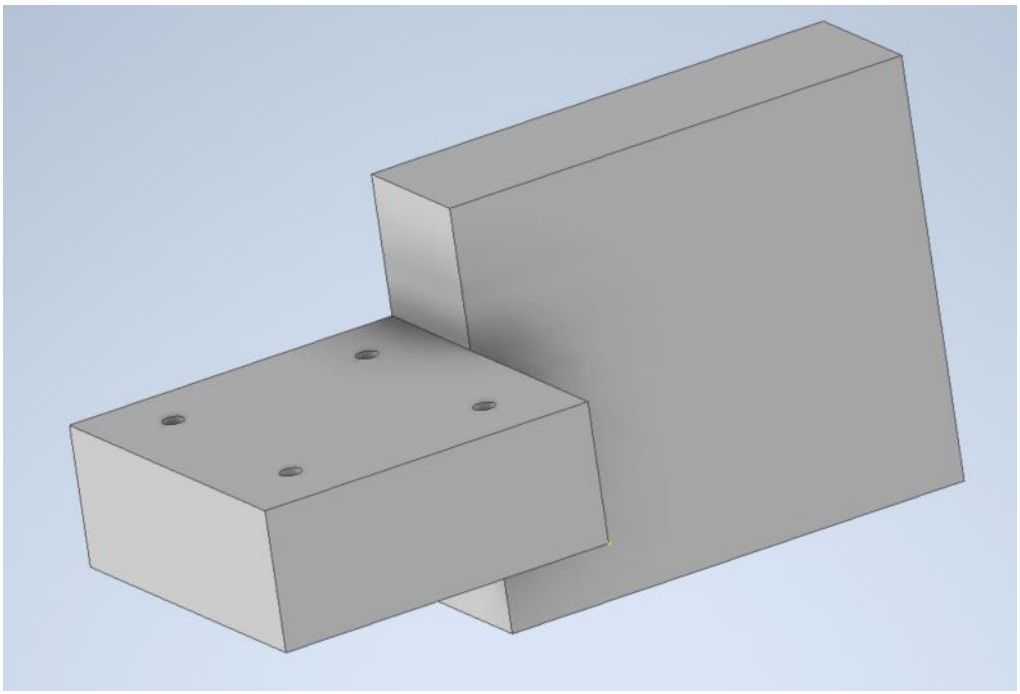


18. ábra: : Elérhető objektívtől való távolságok különböző fókusztávolságokkal és legkisebb méretet tartalmazó pixelszámokkal

A 18. ábra esetén értelemszerűen a negatív tartomány csupán elméleti jellegű, hiszen nem lehet a megfigyelni kívánt tárgyat az objektíven belülre helyezni. A releváns része a kívánt tartomány, azaz az objektívtől mért 20 – 100 mm-es tartomány. Látható, hogy itt minden definiált legkisebb méretet tartalmazó pixelszám megvalósítható a rendszerrel, tehát a megoldás minden igényt kielégítő. Ezen felül még fontos kiemelni, hogy egy ilyen kamerarendszer kiválasztásánál fontos paraméter a szenzorformátum. A szenzor és az optika ezen jellemzője egybe kell, hogy essen. Amennyiben az optika szenzorformátuma túl nagy, akkor az általa alkotott kép széle kilóg a szenzorról. Ha pedig túl kicsi, akkor a kép széle sötét marad és középen egy körben látható a hasznos információ, a kép. A szenzorformátum az említett Sony IMX477 és a Waveshare Electronics 8-50mm Zoom Lens esetében megegyezik, tehát kompatibilisek egymással.

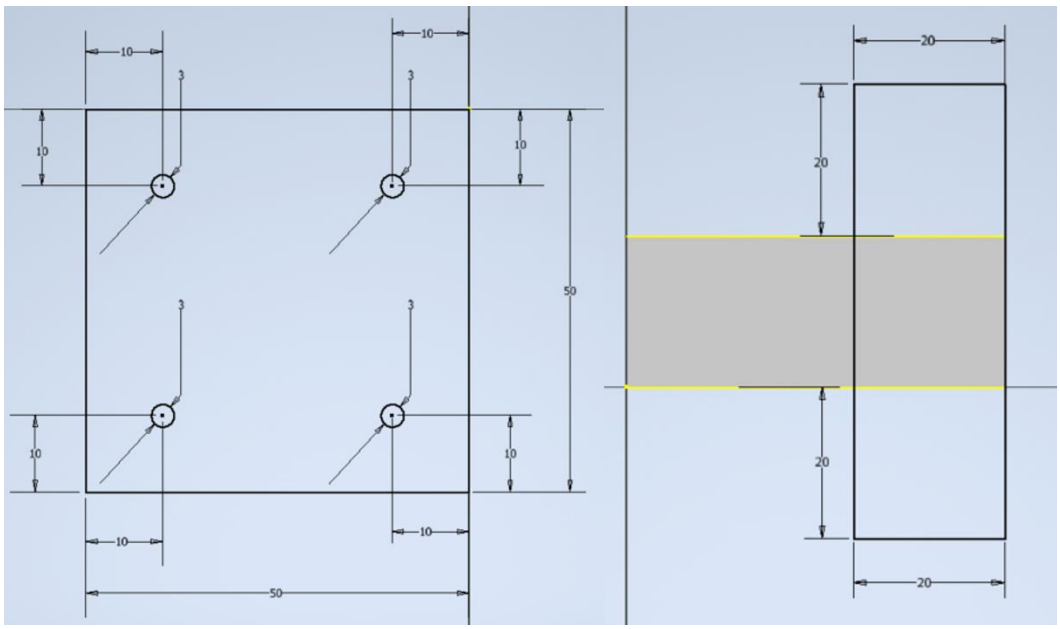
4.2 Hardver erőforrások és mechanikai konstrukció

Ezzel az optikai rendszer már összeállt. A kiválasztott szenzor megtalálható, a Raspberry Pi HQ Kamerában [32], így egy Raspberry Pi kiváló lehetőséget nyújt az önálló kameraplatform elkészítésére. Nekem a Raspberry Pi 4 Model B [33] hardver állt rendelkezésemre, amelyet a saját gyári szalagkábelével csatlakoztattam a kamerához. Ez a 2000 mm-es kábel kellően hosszú ahhoz, hogy megoldható legyen, hogy a Raspberry Pi ne mozogjon együtt a robottal, csak a kamera legyen rögzítve hozzá. Így kameratartót is terveztem a rendszerhez. Ezt Autocad Inventor Professional 2022 szoftverben tettem meg.



19. ábra: Kamera tartó Autocad Inventor Professional 2022 szoftverben

A 19. ábra ezt a mechanikai megvalósítást segítő tartó eszközt illusztrálja. Tervezése során a robothoz igazított méreteit a 20. ábra mutatja be. Bal oldali részén a 19. ábra bal oldali blokkjának felülnézetének méretezése látható. Jobb oldali részén pedig a jobb oldali blokk oldalnézetének méretezése. Ezen kettő rajzból az első esetében 20 milliméterrel húztam ki a bal oldali blokk testét, a második esetében pedig 70 milliméterrel a jobb oldali blokk testét.



20. ábra: Kameratartó méretei

A Creality CR-X Pro 3D nyomtatójával nyomtattam ki az alkatrészt. A nyomtatás eredménye az F6 függelékben a 79. ábrán figyelhető meg.

5 Szoftveres megvalósítás

Ebben a fejezetben áttekintem rendszeremhez fejlesztett szoftverem, annak felhasználói szempontból lényeges konfigurációs lépéseit és az azokat segítő szkripteket. Ezt követően pedig részletezem és tárgyalom az egyes autonóm működésű alrendszerek megvalósítását, az alkalmazott és implementált kódokat, algoritmusokat és a felhasznált szakirodalmakat.

5.1 Kamerakép továbbítása a Raspberry Pi szerverről a kliensgépeknek

Rendszerem szerver oldalán folyamatosan fut egy szkript a 4.2 fejezetben megemlített Raspberry Pi 4 Model B platformon. Létrehozok egy *socket*-et, majd lehetővé teszem, hogy akár több *socket* is csatlakozzon ugyanahhoz a porthoz. Ezután IP címet és port számot adok meg. Ezt követően várok a csatlakozásra. Amint az megtörtént, elkészítem az aktuális fotót a kamerával és elküldöm. Kliens oldalon, ha a kód valamelyik részében képet akarok fogadni, akkor létrehozok egy *socket*-et, kapcsolódok a szerver IP címének és port számának segítségével, majd csomagokban fogadom a fájlt és lementem. Ezen képfájlok segítségével valósíthatók meg az egyes konfigurációs lépések is és a mérés közben is így készülnek a fotók.

5.2 A látórendszer konfigurációja

Ahogy arra már a 3.1.3 fejezetben utaltam, az automatizált mérést egy ember által végzett konfiguráció előzi meg. Ezen alfejezet mutatja be a szoftver alrendszer konfigurációját, amelynek alapvetően három része van:

- Az optikai rendszer torzításának kalibrációja.
- Az optikai rendszer pozíciójának és orientációjának kalibrációja.
- A mérési minták elhelyezése és az első mintához való kontaktáláshoz a robot pozíciójának beállítása.

Ezt követően pedig indulhat a mérés 3.1.3-nak és a 15. ábrának megfelelően, amelyhez a rendszerem biztosítja a szükséges bemeneti adatokat. Az itt felsorolt pontok közül az első kettőhöz tartozó szoftveres megvalósítást az 5.2.1 és 5.2.2 fejezetek mutatják be.

5.2.1 Optikai rendszer torzításának kalibrációja

Adott esetben a konfiguráció részének tekinthető a torzítás kalibrálása, hiszen, ha például változik a mérendő komponens mérete, vagy más távolságból kell megfigyelni az objektumot, akkor előfordulhat, hogy állítani kell a fókusztávolságon. Azonban, ha egymás után többször mérendők ugyanolyan komponensek, akkor fix állású marad a fókusztávolság és a korábbi kalibráció is megfelelő, hiszen az az optikai rendszer ugyanazon állapotához tartozik.

5.2.1.1 Torzítás kalibrációhoz felhasznált összefüggések

Azért jön létre a radiális torzítás, mert az optikai rendszerbe nem párhuzamosan érkező fénysugarakat, különösképpen a rendszer lencséjének szélénél érkezőket, nem tökéletesen egy pontba gyűjti össze a rendszer, így a szenzoron is máshol, másik pixelnél jelenik meg a kép. [34] alapján a torzítás (33) szerint számolandó:

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = \begin{pmatrix} x_u \\ y_u \end{pmatrix} + \begin{pmatrix} x_u \\ y_u \end{pmatrix} \cdot \sum_i k_i \cdot r^{2i} \tag{33}$$

ahol:

$\begin{pmatrix} x_d \\ y_d \end{pmatrix}$ a torzított pixel pozíció x és y koordinátája,

$\begin{pmatrix} x_u \\ y_u \end{pmatrix}$ a helyes pixel pozíció x és y koordinátája,

r a középponttól mért távolság,

k_i a torzítást leíró i-edik konstans együttható.

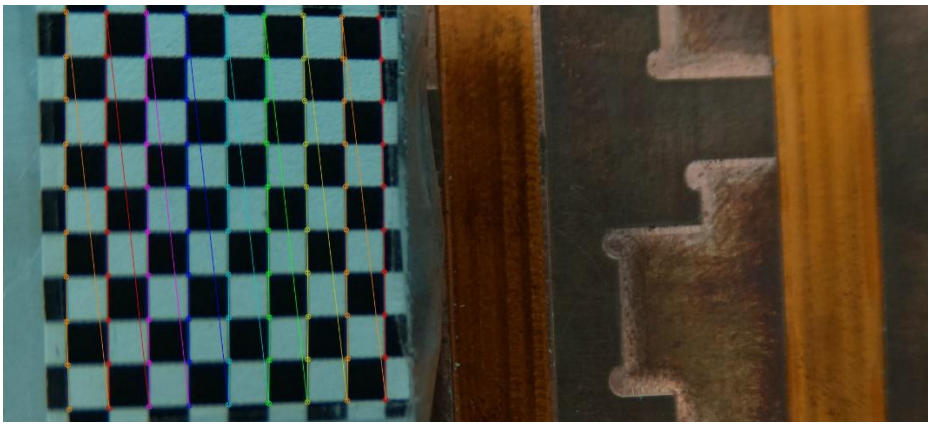
Itt ezen a ponton még a vetítés mátrixa nem ismert, de [35] szerint így is indítható egy algoritmus. Ha bizonyos pontok relatív pozíciói ismertek a képen és ezek világkoordináta-rendszerbeli helye tudható, akkor a (33) egyenletet rendezve:

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} - \begin{pmatrix} x_u \\ y_u \end{pmatrix} = \begin{pmatrix} x_u \\ y_u \end{pmatrix} \cdot \sum_i k_i \cdot r^{2i} \tag{34}$$

Ilyenkor (34) bal oldala a torzítás pixelben számítva. A jobb oldal nagyságának különböző pozíciókra való minimalizálásával megkapható az a torzítási együttható együttes, amelyre a legkisebb négyzetes hibájúak a torzított pixel pozíciók. Ezen nemlineáris minimalizálásra, amely során természetesen a triviális, minden i-re $k_i = 0$ megoldás kerülendő, az OpenCV a Levenberg–Marquardt algoritmust használja. [36] Megjegyzendő, hogy ahogy említettem, itt a kameramátrix még nem ismert, de a valós térbeli pontok vetítéséhez szükséges, ugyanis ezekre a vetített pontokra alkalmazható (33) és (34). Így az OpenCV a széles körben ismert SVD felbontás szerint számítja a mátrixot. [37]

5.2.1.2 Torzítás kalibráció megvalósítása

Az 5.2.1.1 fejezet szerint említett ismert relatív pozíciójú pontok egy sakktábla pontjai. Erről egy kalibrációs képet mutat meg a 21. ábra. Ehhez hasonló fotókat kell készíteni a kamerával, és ezeken futtatni a kódomat, amihez az OpenCV mellett többek között a Python numpy könyvtárát is használom.



21. ábra: Torzítás kalibrálása sakktábla használatával

A 21. ábra alapján a világkoordináta-rendszert a z sík mentén a sakktábla síkjába rögzítem és ahogy az látható, csak a belső sarokpontokat figyelem, amelyekből $9 \cdot 10 = 90$ darab van. Az ábrán berajzolt narancssárga, piros, rózsaszín, kék, türkizkék, zöld, citromsárga, narancssárga, piros útvonalon végighaladva, a z síkban kaphatók rendre a (0, 0, 0), (1, 0, 0), … (9, 8, 0) pontok. Ezeknek a pontoknak létrehozom *numpy* segítségével a $9 \cdot 10$-es tömbjét. Ezt követően létrehozok a valós pontoknak és a képpontoknak egy-egy listát, amelybe majd minden kép minden pontja tárolható lesz, majd beolvasom a képek listáját és elkezdem az iterációt a képeken. Minden egyes képet szürkeárnyalatossá konvertálok, majd az *OpenCV findChessboardCorners* függvényével megkeresem az aktuális sakktábla sarkait. Ez megkapja a képet, a sakktábla dimenzióit és kaphat különböző flag-eket is. Ezek közül érdemes lehet beállítani például az adaptív küszöbözést, hogy a megvilágításra ne legyen érzékeny az algoritmus, illetve a normalizálást a hisztogramkiegyenlítés érdekében. Az így talált sarkokat pontosítani lehet a *cornerSubPix* függvénnyel [39]. Ezt követően a létrehozott listákhoz hozzáadom a világ és a kép pontjait, majd haladok a következő képre. Az iteráció végén meghívom a *calibrateCamera* függvényt, megadva ezeket a listákat. Ez az 5.2.1.1 fejezetnek megfelelően működik. Így megkapható a vetítés mátrixa és a torzítást leíró konstans együtthatók, amiket lementek. Fontos megjegyezni, hogy a kalibrációs képek elkészítése esetén ügyelni kell rá, hogy kellő mennyiségű és minőségű kép készüljön a numerikus stabilitás megőrzésének érdekében. A minőség egyik fontos jellemzője, hogy a sakktáblák lehetőleg minél közelebb legyenek a képek széléhez, hiszen itt a legnagyobb a radiális torzítás. Továbbá megjegyzendő még, hogy az itt tárgyalt függvények, bár Pythonból kerülnek meghívásra, azonban a futási idő mérséklése érdekében valójában C++ környezetben futnak le.

5.2.2 Optikai rendszer pozíciójának és orientációjának kalibrációja [40]

A fentebb említett konfigurációnak teljes mértékben része a kamera pozíciójának és orientációjának kalibrációja. Ez minden indított mérés előtt elvégzendő, hiszen a kamera és a 3.1.3 szerinti speciális hűtő rézlap egymáshoz viszonyított helyzetét nem lenne egyszerű minden mérés előtt 100 µm pontossággal ugyanúgy beállítani. Erre kísérletet lehet tenni, azonban az igazán jó és robosztus megoldáshoz ilyen pontosságú emberi lépést nem szabad előfeltételezni.

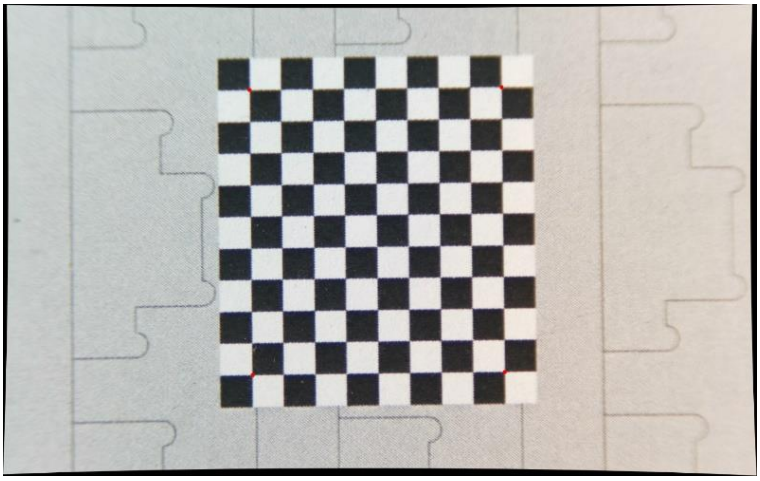
A pozíció és orientáció esetleges változásának eredményeként ugyanaz a valós alakzat a képen máshogy jelenik meg. A képre az egyes alakzatok perspektív vetítés eredményeként kerülnek. A perspektív vetítés nem távolságtartó, nem aránytartó, nem szögtartó, nem párhuzamosságtartó, azonban kollineáció, azaz egyenestartó. Tulajdonképpen a lyukkamera modell leírója, hiszen egy ponton haladnak keresztül a vetítő egyenesek. Mint sok más térbeli leképezés, úgy ez is egy mátrix segítségével jellemezhető:

$$\begin{pmatrix} s \cdot x' \\ s \cdot y' \\ s \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ c_1 & c_2 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{35}$$

ahol:

s	a skálázási faktor,
x'	a vetített képen a vízszintes koordináta,
y'	a vetített képen a függőleges koordináta,
$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$	a transzformáció orientációs részét leíró mátrix,
$\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$	a transzformáció transzlációs részét leíró vektor,
$(c_1 \quad c_2)$	a projekcióvektor,
x	az eredeti vízszintes koordináta a tárgytérben,
y	az eredeti függőleges koordináta a tárgytérben.

Mivel a leképezés mátrixát 8 konstans definiálja, így 4 eredeti-vetített pontpár ismerete esetén az számítható. Ez a mátrix az alkalmazás szempontjából azért fontos, mert kijelöl a vetített képen egy négyszög tartományt, amit téglalap tartománnyá alakít és ebben a téglalaptartományban lévő objektum hordozott információi alapján lehet navigálni a robotot. Ezt a gyakorlatban úgy valósítom meg, hogy a rézlap egy nyomtatott, markereket tartalmazó mását a rézlapra illesztem és készítek róla egy képet. Ez megfigyelhető a 22. ábra esetén.



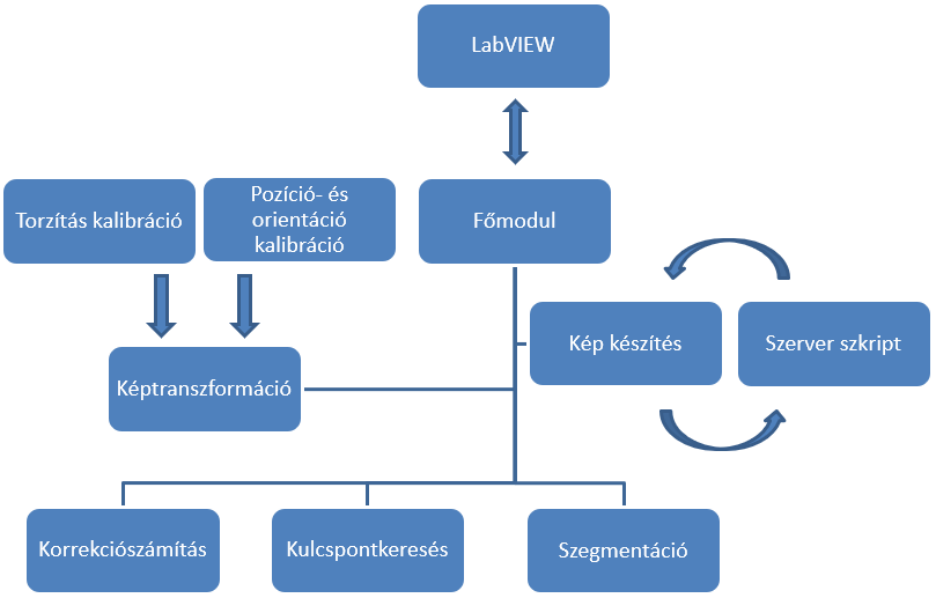
22. ábra: Pozíció- és orientáció kalibráció képe

Látható, hogy itt már elvégeztem a torzításkalibrációt, a kép görbe külső vonalait beljebb húzta az 5.2.1 fejezet szerinti algoritmus. Ezen a rajzon a sakktábla úgy van kialakítva, hogy a pirossal jelölt négy belső sarka által meghatározott téglalap pontosan egy mérési helyet jelöljön ki, ahol komponens lehet. A mérés előtti konfiguráció esetében ezen az egy helyen végzi el ezt a kalibrációt a robot, azonban a 3.1.3 fejezetben leírt mérési algoritmus minden mintájánál alkalmazza a kapott transzformációt.

Tehát a teljes kód a következőképpen jellemezhető. Alkalmazom a torzításkalibrációt az OpenCV *getOptimalNewCameraMatrix* és undistort függvényeinek segítségével. Fontos bemeneti paraméterek az 5.2.1 szerint kiszámított vetítés mátrix és a torzítási együtthatók. Ezt követően megkeresem a 22. ábra pirossal jelölt négy pontját a megfelelő indexelés és a *findChessboardCorners* függvény segítségével. A *cornerSubPix* függvénnyel ismét pontosítom a sarkokat, majd (35) alapján a *getPerspectiveTransform* függvénnyel megkapom a perspektív vetítés mátrixát, amit lementek. A későbbiek folyamán az egyes képek beolvasásánál a *warpPerspective* függvény képes elvégezni a megfelelő transzformációt.

5.3 Korrekció meghatározása

Most már rendelkezésre állnak a megfelelő minőségű képek. Így minden esetben két kép összevetéséből kell eldönteni, hogy mekkora a szükséges korrekció, ahogy azt a 3.1.3 fejezetben a 15. ábra kapcsán is tárgyaltam. Az ötletem az, hogy először elvégzek egy szemantikus szegmentációt, amely meghatározza azt a tartományt a már jó minőségű, torzítatlan képeken, ahol a mérendő komponens helyezkedik el. Ezt követően pedig jellegzetes pontokat keresek ezeken a képeken, amelyek összetartoznak. Ezek közül kiválasztom a biztos pontokat és ezek átlagos elmozdulása alapján meghatározom a valós elmozdulást. Illetve az általuk alkotott vektorok átlagos elforgása alapján meghatározom az átlagos elforgást. Ezeket az értékeket adom vissza a robotnak visszatérési értékként. Ez alapján a teljes szoftver modulárisan a következőképpen szemléltethető:



23. ábra: Szoftver moduláris felépítése

A főmodul a LabVIEW szoftverrel kommunikál. Az 5.2.1 és 5.2.2 fejezetekben tárgyalt előzetesen futtatott kalibrációs szkriptek segítségével elvégezhető az 5.1 fejezetben említett szerverről küldött képen egy transzformáció, amely a jó minőségű, lényeges tartományt állítja elő a képből. Ebből a tartományból *szegmentálható az elektronikai komponens*, amelyen ezután *kulcspontok kereshetők* és *meghatározható az elmozdulása* az előző képhez képest. A következő fejezetekben ezt a három folyamatot fogom tárgyalni.

5.4 Szemantikus szegmentáció

A szemantikus szegmentáció lényege, hogy egy képen minden pixel osztályát meghatározza. Így a kimenetének felbontása megegyezik a bemenet felbontásával és minden egyes pixel esetén annyi féle kimenet jelenhet meg, ahány osztály közül választani lehet az adott feladat szerint. Esetemben azt kell eldönteni minden pixelről, hogy az komponens vagy nem komponens, így kétféle érték adódhat, 0 és 1. Ezen döntés széles körben használt és elfogadott minőségi jellemzője az IoU (Intersection over Union) érték, amely számszerűsíti a szegmentáció kimenete és a valós szegmentáció közti átfedési arányt. Ennek leírása a következő:

$$IoU = \frac{szegmentáció\ kimenete\ \cap\ valós\ szegmentáció}{szegmentáció\ kimenete\ \cup\ valós\ szegmentáció} \tag{36}$$

Jellemzően kiértékelésnél minden egyes osztályra kiszámítják az IoU értéket és átlagolják ezeket.

Egy másik lehetséges minőségi jellemző a pixel pontosság. Itt a helyesen osztályozott pixelek számát vetik össze az összes pixel számával:

$$Pixel\ pontosság = \frac{pontosan\ osztályozott\ pixelek\ száma}{összes\ pixel\ száma} \tag{37}$$

Ennek használata például tanuló algoritmus futtatása során hasznos, azonban önmagában nem alkalmas igazán a szegmentáció minősítésére. Ennek oka, hogy abban az esetben amikor egy képen az egyik osztály nagyon kis arányban jelenik meg, akkor ennek nagymértékű fel nem ismerése sem okoz jelentős hibát a pixel pontosságban. Így nem megfelelő színvonalú szemantikus szegmentációk is megfelelőnek tűnhetnek, ha csak ez a jellemzés használt.

Munkám során szemantikus szegmentációra kipróbáltam hagyományos számítógépes látásban használt módszert és mesterséges intelligencia alapú neurális hálókat használó módszert is. Elsőként a konvencionális megoldást mutatom be.

5.4.1 Szemantikus szegmentáció hagyományos számítógépes látás segítségével

Munkám ezen része épít a K-means algoritmusra, amelyet az 5.4.1.1 fejezetben ismertetek. Ennek az algoritmusnak már hosszabb ideje jelentős szerepe van mind statisztika, mind mintafelismerés terén [41], de

még manapság is születnek újszerű és érdekes felhasználásai. Például használják olyan módon, hogy igyekeznek kiszűrni a fényességből fakadó képen megjelenő különbségeket [42], de alkalmazható hagyományosan is például műholdképek szegmentációjának segítésére [43].

5.4.1.1 A K-means algoritmus működése

Az algoritmus lényege a képfeldolgozásban, hogy különálló klaszterekbe sorolja a kép minden egyes pixelét. Ezen klaszterek száma *K*, melyet a felhasználó bemenő paraméterként ad meg az algoritmusnak. Tulajdonképpen az alábbi kifejezés minimalizálása a cél [41] :

$$E(m_1, m_2, \dots, m_M) = \sum_{i=1}^{N} \sum_{k=1}^{M} I(x_i \in C_k) \cdot \left|\left|x_i - m_k\right|\right|^2 \tag{38}$$

ahol:

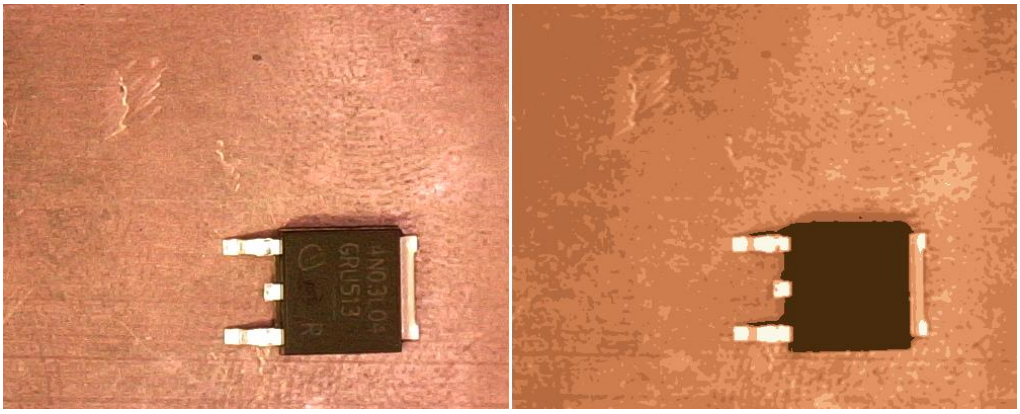
x_i	az i-edik pixel vektora, amelynek, elemei a színcsatornák értékei, másképp fogalmazva i-edik pont,
N	a pixelek száma,
M	a klaszterek száma,
C_k	a k-adik klaszter,
m_k	a k-adik klaszterbe tartozó vektorok összege elosztva M-mel, a centrum, a klaszterközéppont,
$I(X)$	1, ha X igaz, 0, ha X hamis,
E	a klaszterezési hiba.

Ez a klaszterezési hiba értelemszerűen függ az egyes klaszter középpontoktól. Tulajdonképpen összegzi az egyes klaszterekbe tartozó pontok klaszterközéppontoktól való négyzetes távolságainak összegeit. Ennek minimalizálása az $\{x_i | i = 1, \dots, N\}$ pontok terébe véletlenszerűen generált *K* darab centrum inicializálásával kezdődik. Második lépésként minden egyes klaszterközépponthoz hozzárendeli az algoritmus azokat a pontokat, melyek ahhoz a centrumhoz vannak a legközelebb. Harmadik lépésként pedig felülírja a klasztercentrumokat úgy, hogy azok, a klaszterbe tartozó pontok számtani közepét alkossák. Ezt követően a második és a harmadik lépés ismétlődik, amíg állandóvá, vagy kellően kevéssé változóvá nem válnak a centrumok, vagy el nem ér egy felhasználó által megadott értéket az iterációk száma [44].

5.4.1.2 Hagyományos látás alapú algoritmus működése

Eredetileg a képek pixelei kétdimenziós tömbként vannak eltárolva. Első lépésként ezt átalakítom egy egydimenziós tömbbé úgy, hogy soronként haladok az eredeti kétdimenziós tömbben. Ez azért szükséges, mert az 5.4.1.1 fejezet szerinti K-means algoritmus egy ilyen ponthalmazzal dolgozik. Szintén lényeges, hogy el kell végezni egy típuskonverziót, mivel az OpenCV-ben a K-means megvalósítását végző *kmeans* függvény [44] 32 biten ábrázolt lebegőpontos számokat vár bemenetként, így ilyenné kell alakítani az egyes pixelek eredetileg egész számmal jelölt színcsatornáit. Ez azért fontos, mert az OpenCV függvényei, ahogy azt az 5.2.1.2 fejezetben már kiemeltem, a jobb futási idő érdekében C++ nyelven vannak megírva és a gyengén

típusos Pythonban csupán a meghívásuk történik. Ezt követően már meghívható a függvény. Bemenetként meg kell adni a pixelek tömbjét, a klaszterek számát, a megállási kritériumot, valamint egy flag-ként azt, hogy véletlenszerűen generálja a klaszterek középpontjait az algoritmus. Megállási feltételnek a legalább 10 iteráció elérése és a legfeljebb 1 nagyságú centrum elmozdulás közül azt választom, amelyik hamarabb teljesül. Klaszterszámnak az 5 bizonyult a legjobb választásnak.



24. ábra: Az eredeti kép és a K-means algoritmus eredményeként kapott kép

Ennek oka az, hogy ahogy az a fenti ábrán is megfigyelhető a szemantikus szegmentáció megvalósítható fényesség alapon, ugyanis a legfényesebb rész a komponensek fém része, a legsötétebb rész pedig a házuk lesz. Ez alapján, ha túl kicsiny klaszterszámot adok meg, akkor a rézlap fényesebb részeit is besorolja a legfényesebb osztályba az algoritmus, míg, ha túl nagyot, akkor a sötét színű házon belül is megtalálja a legsötétebb részeket és így csak kisebb arányban tudom szegmentálni a képet. Többféle centrumszám esetén klaszterezett képeket mutat be az F7 függelék, ahol az ezen képekhez tartozó szemantikus szegmentáció képei is fel vannak tüntetve. Ezen szegmentált képek elkészítési módszerének részletezését a következő bekezdésben folytatom. Megjegyzendő még, hogy ahogy az a 24. ábra esetén is megfigyelhető itt másik rézlapon van elhelyezve a MOSFET, a 3.1.3 fejezethez képest. Erre részletesen ezen fejezet végén térek ki.

Tehát a fényesség alapú megközelítés jó módszernek bizonyul a 24. ábra esetében. Jellemzően a fényesség mértékét meghatározó számérték a piros, zöld és kék színcsatornák felvett értékeinek lineáris kombinációjaként áll elő. Ez a lineáris kombináció azért nem egyszerű átlagolást jelent, mert az emberi szem sokkal érzékenyebb a zöld színre, mint a kékre. A telített két szín sötétnek látszik, így kevéssé járul hozzá a fényerő növekedéséhez a kék színcsatorna értékének növelése. Ez az eltérő súlyozás egy elterjedten alkalmazott eljárás. [45]-[47] Én a (39) egyenlet szerinti lineáris kombinációt alkalmaztam.

$$fényesség = 0{,}3 \cdot piros + 0{,}59 \cdot zöld + 0{,}11 \cdot kék \tag{39}$$

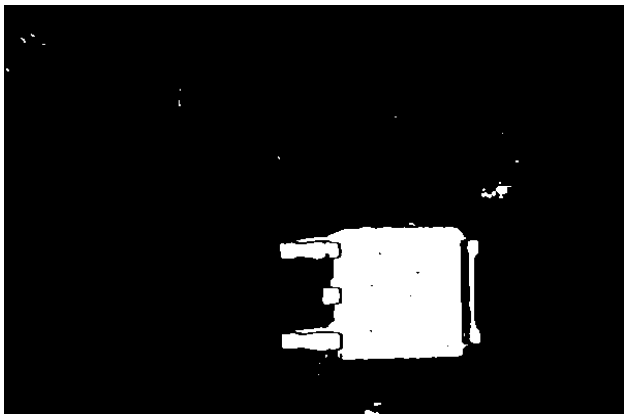
A már említett *kmeans* függvény általam felhasznált visszatérési értékei a pontokhoz tartozó osztályok tömbje és a klasztercentrumok értékeinek tömbje. Így a centrumok tömbjét, mint mátrixot szoroztam össze a fényességet kiszámító vektorral az alábbi módon:

$$\begin{pmatrix} C_{1R} & C_{1G} & C_{1B} \\ & \vdots & \\ C_{MR} & C_{MG} & C_{MB} \end{pmatrix} \cdot \begin{pmatrix} 0{,}3 \\ 0{,}59 \\ 0{,}11 \end{pmatrix} = \begin{pmatrix} C_{1L} \\ \vdots \\ C_{ML} \end{pmatrix} \tag{40}$$

ahol:

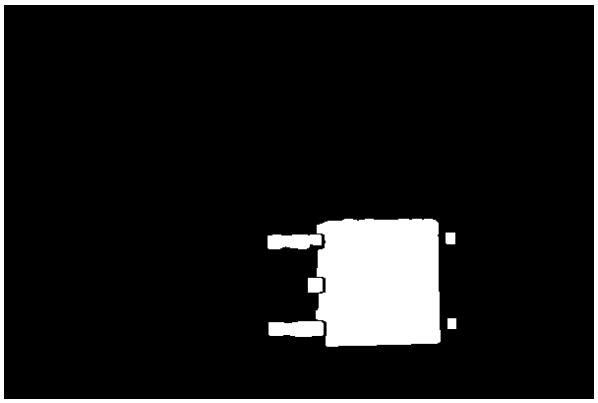
M	a klaszterek száma,
C_{kR}	a k-adik klaszter piros összetevője,
C_{kG}	a k-adik klaszter zöld összetevője,
C_{kB}	a k-adik klaszter kék összetevője,
C_{kL}	a k-adik klaszter fényességértéke.

A kapott fényességértékek közül kiválasztottam a legnagyobbat és a legkisebbet. Az ezekhez tartozó klaszterek színét fehérre állítottam, a többit pedig feketére egy olyan képen, amit már visszaformáztam az eredeti kétdimenziós pixelek tömbjévé. Itt már ismét 8 biten ábrázolt egész számokat használtam a színek meghatározására. Ezzel az alábbi képhez jutottam:



25. ábra: A K-means algoritmust követő fényesség alapú szegmentáció eredménye

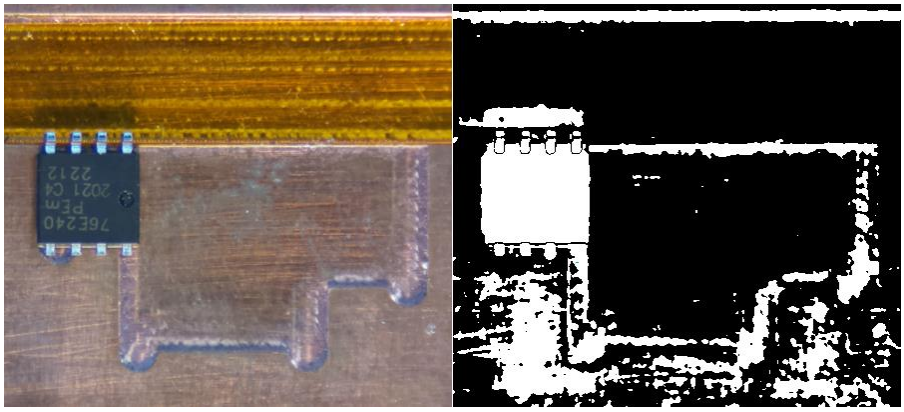
Jól megfigyelhető, hogy vannak fekete részek a MOSFET-en és fehér részek a rézlapon. Ezek a problémák még orvosolandók. Előbbit a következőképpen oldom meg. Az OpenCV *floodFill* függvényével [48] a bal felső sarokban lévő pixeltől indulva kitöltöm fehér színűre az egybefüggő fekete részeket, így csupán a MOSFET belső fekete pontjai maradnak meg. Ezt követően invertálom a képet, ezzel megkapva egy fekete képen fehér színnel a MOSFET eredetileg fekete belső pontjait. Ez utóbbi képből és az eredeti problémás képből egy logikai *vagy* művelettel megkapható a 25. ábra első problémától mentes verziója. A második probléma morfológiai transzformációkkal szüntethető meg. Gyakorlati tapasztalatok alapján egy 5·5-ös 1-es értékeket tartalmazó kernellel végzett egymás utáni kétszeri erózió és dilatáció az OpenCV *erode* és *dilate* függvényeivel eltünteti a felesleges fehér részeket és nem rontja jelentős mértékben a helyes szegmentációt sem. [49] Így a 26. ábra szerinti végleges szegmentációt kapom. A köztes lépések eredményei fellelhetők az F8 függelékben.



26. ábra: Hagyományos számítógépes látással végzett szemantikus szegmentáció eredménye

Ez az eredmény a végső cél, azaz a navigáció szempontjából jónak tűnhet, hiszen jól sikerült kiválasztani a MOSFET-hez tartozó pixeleket. Viszonylag kis területek elvesztésével járt és nincsenek fals módon szegmentált pixelek, azaz eltüntettem a rézlap csillogásából következő pontszerű, elszórt, felesleges szegmentációkat a képről.

Azonban a módszer *nem kellően robosztus*. Kifejezetten ügyelni kell a megvilágításra és közel homogén hátteret kell biztosítani az egyes komponenseknek. Előbbi még nagy odafigyeléssel orvosolható lehet. Bár a cél az, hogy a termikus méréseknek emberi felügyelet nélkül minden napszakban, így éjszaka is futniuk kell, mégis biztosítható az időben állandó megvilágítás. Ez úgy tehető meg, ha leárnyékolom a mérőteret a külső fényt kizárva és saját világítótesteket helyezek fel. Viszont az utóbbi elvárás nem teljesíthető, hiszen a speciálisan kialakított hűtő rézlap, amelyen lehetséges mérni olyan házú MOSFET-eket mint a DPAK, D2PAK, LFPAK88, LFPAK56, vagy az LFPAK56D, ezen komponenseknek megfelelő foglalatokat tartalmaz, ami megakadályozza a homogén vizualitást. Ráadásul CNC marással kerül kialakításra a rézlap felülete, így ez is gátolja a homogén színeloszlást. A rézlapot szemlélteti a 14. ábra és a 79. ábra valamint az F9 függelékben is megfigyelhető modellje. Egy ezen rézlapon készült fénykép esetén a következő szegmentáció kapható:



27. ábra: Hagyományos gépi látás használata az inhomogén színű rézlap esetén

Ez egyértelműen nem kielégítő. Ha az elmozdulást az itt szegmentált fehér színű rész kulcspontjai alapján kellene meghatározni, akkor jóval eltérne a becsült érték a valóstól, hiszen fix pontok is szegmentálásra

kerültek. Ennél jobb eredményre van szükség. Ezért kezdtem el foglalkozni mesterséges intelligencia alapú módszerekkel, melyeket az 5.4.2 fejezetben mutatok be részletesen.

5.4.2 Szemantikus szegmentáció mesterséges intelligencia és neurális hálók segítségével

A fenti 27. ábra kapcsán megmutatkozott, hogy jobb módszerre van szükség a hagyományos látás nyújtotta alternatívánál. Igen sok kutatást, cikket, publikációt, könyvet programkódot vizsgáltam meg, hogy megtaláljam ezt a jobb módszert. A következő fejezetben ezen kutatómunkámat foglalom össze tömören.

5.4.2.1 Szakirodalmi összefoglaló

A szemantikus szegmentációt nem véletlen olyan nehéz jól megvalósítani, hiszen a számítógépes látás egyik legnagyobb kihívása. Ahogy azt az 5.4 fejezetben is említettem, feladata, hogy egy adott képen pixelenként döntést hozzon arról, hogy az adott pixel milyen osztályhoz tartozik. Ahogy a tudomány sok más területén, úgy itt is nagy szerepe van a mélytanulási módszereknek. Számos alkalmazása van a mélytanulással megvalósított szemantikus szegmentációnak. Orvosi alkalmazási példákat mutat be [50]-[54], vagy éppen önvezető autókhoz használja fel [55]-[58]. Kiemelkedő alkalmazási példa még az élővilággal, állatokkal kapcsolatos szemantikus szegmentáció [59]-[61], illetve a gyalogosok szegmentációja [62][63], amely szintén az autonóm közlekedést segíti elő. De ezen kívül számos téren használnak még szemantikus szegmentációt, például anyagvizsgálat során hibakeresésre [64], vagy műholdképek vizsgálatára [65][66]. A különböző alkalmazásokhoz használt technikákat más-más szerzők eltérően csoportosítják. [67] átfogó képet nyújt a mélytanuláson alapuló szemantikus szegmentációról és három nagy részre osztja fel a különböző technikák szakirodalmait. Nevezetesen régió alapú, a teljesen konvolúciós háló alapú és a gyengén felügyelt módszerekre. De [68] ennél már jóval részletesebb felosztást alkalmaz, tíz kategóriába sorolja az egyes módszereket. Ezek a módszerek szétbontva a Markov-mezőkre és feltételes véletlenszerűségre, ezek kiváltására, az egyes részletek felbontás változtatására, a visszacsatolt neurális hálókra, a régiókra, a dekonvolúcióra, az egyes részek kiemelésére, a gyenge- és közepes megerősítésre, az időbeliségre és a részlet kódolásra alapuló technikák. [69] egy szinttel mélyebb felbontást is alkalmaz. A korábban említett eljárásokat unimodális – multimodális megfontolás szerint választja ketté. Ez tulajdonképpen az egy képkockát felhasználó metódusok és az egymás utáni, időben máskor készült képekből, vagy egyszerűen több képből összeállított három dimenzióra alapuló eljárások szembeállítása. De [70]-[72] csoportosításai sem megegyezők. Alkalmazások tekintetében számos helyen már igen jól teljesített ezen felsorolt technikák közül a teljesen konvolúciós háló, azaz FCN (fully convolutional network). Olyan különböző, szerteágazó példák említhetők, mint az útszegmentáció [73], a nagy felbontású légi felvételek szegmentációja [74], fa fajok szegmentációja [75], vagy az MR vizsgálatok képein végzett szegmentáció [76]. Nem véletlen, hogy ezt a módszert választották [73]-[76] szerzői és még sokan mások [77]-[80]. Az ezelőttiekben felsorolt szemantikus szegmentációt áttekintő cikkekből is egyértelműen kiderül, hogy az FCN kiemelkedőnek mondható. [68] szerint az FCN hozta meg az igazi áttörést a mélytanulásra alapuló szemantikus szegmentációban. [69] szerint

is a korábban említett Makov-mezőkre és feltételes valószínűségekre alapuló technikákat, valamint az egyszerű konvolúciós neurális hálókat használó módszereket követően jöttek igazán jó szemantikus szegmentációs megoldások, amelyeket az FCN hozott el. Hasonlóan [70] alapján, a közelmúlt sikerei a szegmentációban elsősorban az FCN-nek köszönhetők. Illetve [71] szerint drasztikusan megnövelte a pontosságot, [72] pedig kiemeli, hogy a jelen állapot szerint legsikeresebb state-of-the-art mélytanulási szemantikus szegmentációs technikák alapja minden esetben az FCN. Ahogy azt [67]-[72] jól összefoglalja, számos FCN-alapú technika született már. Ezek különböző méretű, így különböző hardverigényű megoldások. A 2. táblázat FCN módszereket és azok megfontolásain alapuló további technikákat mutat be.

2. táblázat: FCN módszerek és hasonló technikák

Jellemző / Módszer	Benchmark	Átlagos IoU [%]	Forrás
SDN	PASCAL VOC 2012 test	79,9	[81]
OA-Seg	PASCAL VOC 2012 test	74,1	[82]
DeconvNet	PASCAL VOC 2012 test	69,6	[83]
FC-DenseNet103	CamVid dataset	66,9	[84]
SegNet	CamVid dataset	60,1	[85][85]
FCN-8s	PASCAL VOC 2011 segval	62,7	[86]
FCN-16s	PASCAL VOC 2011 segval	62,4	[86]
FCN-32s	PASCAL VOC 2011 segval	59,4	[86]

Az itt feltüntetett pontosságot legjobban jellemző átlagos IoU értékek számítási módszere a korábbi 5.4 fejezetben részletesen tárgyalt. Az egyes benchmarkok nemzetközileg elfogadott, ismert adathalmazok, melyek szemantikus szegmentáció ellenőrzésére kifejezetten alkalmasak. [87]-[89]

Elektronikai komponensek szemantikus szegmentációjához kapcsolódó cikkek

Bár látható, hogy a mélytanulás alapú szemantikus szegmentáció népszerű és széles körben alkalmazott, de elektronikai komponensek kapcsán, különösen MOSFET-ek esetében igen alacsony mértékben publikált a téma, korszerűnek mondható, tudományos vizsgálatra érdemes. Kifejezetten tranzisztorok szemantikus szegmentációjáról nem is találtam folyóirat-, vagy konferenciacikket. Hibakeresési módszer található ebben a témában [90], vagy nyomtatott áramkörök esetén is [91], azonban ez más technikát igényel. A leginkább kapcsolódó szakirodalom is egyedi példányokkal foglalkozó (instance) szegmentációt végzet kapacitások, és ellenállások esetén. [92] Ennek sikere saját alkalmazásom szempontjából kétes értékű lenne, mivel a közölt tanító és tesztelő adathalmazon egyszínű, fehér háttér előtt volt minden egyes komponens és csupán az árnyékuk lehetett zavaró. Így nem lenne kellően robosztus a módszer alkalmazásomhoz. Nyomtatott áramkörök esetén találhatók még érdekes, komponensekkel foglalkozó publikációk, azonban ezek nem szegmentációval, hanem detekcióval [93], vagy nagyobb elemek mélység alapú szegmentációjával

foglalkoznak [94]. A fellelt elektronikai komponensekhez kapcsolódó cikkek közül egyik sem származik 2020-nál régebbről, így magabiztosan kijelenthető, hogy újszerű és nyitott témáról lehet beszélni.

5.4.2.2 Megvalósítás

A műszakilag megfelelő, szemantikus szegmentációra alkalmas, teljesen konvolúciós neurális háló létrehozásához saját kódot implementáltam. A tanításhoz szükséges adathalmazt pedig az éles mérés esetében is használt kamerával és optikával készítettem el. Az annotációt kézzel végeztem a tanító és validáló képeken. Módosítottam a mélységet, a csatornaszámokat, a felbontást és a batch méretet, így automatizált futtatások segítségével 21380 tanítottam be és értékeltem ki neurális hálókat. A következőkben külön-külön jellemzem az egyes ehhez készített összefüggő kódrészleteim, osztályaim, függvényeim, és a teljes háló működését. Ehhez felhasználtam a Python *Pytorch* [95]*, Torchvision* [96]*, Pillow* [97]*, Numpy* [98]*, Matplotlib* [99]*, OS* [100] *és Glob* [101] könyvtárait. Megjegyzem, hogy ezek neveit a továbbiakban kisbetűvel szerepeltetem, hű maradva ezzel kódom formátumához.

Adat-augmentációs függvény

Ahogy a mesterséges intelligencia tanítás más területein is, úgy itt is szükséges a robosztus tanításhoz a megfelelő adatmennyiség. Ezért valósítottam meg saját adat-augmentációt. Ehhez létrehoztam egy osztályt, amelynek a *()*, azaz *__call__* operátorát felülírtam. Ezt úgy tettem meg, hogy amennyiben a *__call__* meghívásra kerül és adott a kép a *pillow* segítségével, akkor a *random* által generálódik egyenletes eloszlással egy szám 0 és 1 között, ami ha 0,5-nél nagyobb, akkor tükrözésre kerül a kép a vízszintes tengelye mentén, egyébként nem. A felülírt függvény visszatér a képpel.

Adathalmaz elkészítése

Az adathalmaz szoftveres elkészítése a következőképpen történt. Származtattam a *pytorch torch.utils.data.DataLoader* osztályából egy saját osztályt. Ennek *__init__* függvényét felülírtam úgy, hogy az megkapja azt, hogy milyen adat-augmentációt kell végrehajtani a képek és címkék közül az éppen adott képen vagy címkén, illetve, hogy hol található a képeknek és címkéiknek a gyökérmappája. Előbbiekből attribútumokat készítek, utóbbit pedig az *os.path.join* és a *glob.glob1* függvény segítségével felhasználom arra, hogy elkészítsem a képek és a címkék elérési útvonalának listáját, amit sorba rendezek. Ezekből is attribútumot készítek.

Ezt követően indexelhetővé teszem az osztályt, így felülírom a *__getitem__* függvényt. Beállítom, hogy a képek elérési útvonalának listájának attribútumát indexelve visszakapható legyen az adott kép, illetve ugyanezt teszem a címkék esetén is. A *pillow* könyvtár *Image* moduljának *open* függvényével ezután egy *image* és egy *label* változóba elmentem az adott képnek és címkéjének referenciáját. Végül azonos, de random-generált *seed* változóval meghívom a *random.seed* függvényt kétszer és amennyiben volt megadva

adat-augmentáció a képhez vagy a címkéjéhez, akkor azt végrehajtom a hívások után és visszatérek a referenciákkal.

Konvolúciós réteg elkészítése

Egy konvolúciós rétegbe hálómban beletartozik a konvolúció művelete mellett a nemlinearitás és a normalizálás is. Ezt egy osztályként származtatom le a *pytorch.nn* modulból. Itt is felülírom a konstruktorfüggvényt, amely megkapja a bemeneti és kimeneti csatornaszámot, a konvolúciós kernel méretét és a lefelé skálázás mértékét is jellemző *stride* paramétert, amely azt adja meg, hogy hány pixeles lépéssel haladjon végig a képen a kernel. Ezt követően az ősosztály inicializálása után *convolution* attribútumnak létrehozok egy *pytorch.nn.Conv2d* osztálybéli objektumot. Ez hívásnál megkapja a csatornaszámokat, a *stride*-ot, illetve, hogy hány 0-val töltse ki a kép széleit, azaz a *padding*-et. Utóbbit a kernel méretének felének egészrészére állítom. Szintén létrehozok attribútumnak egy normalizálást, amelyhez a *pytorch.nn.BatchNorm2d* függvényt használom, ami megkapja a kimeneti csatornaszámot.

Ezen osztály egy objektumából a következő objektumba, azaz az egyik rétegből a következő rétegbe a *forward* függvénnyel lépek, amely visszatér a normalizálással, ami megkap egy javított lineáris függvényt, a *ReLU*-t (Rectified Linear Unit), az elsőként létrehozott *convolution* attribútum behelyettesítésével. Ez egy nemzetközileg elfogadott és széles körben alkalmazott módszer [102][103].

Transzponált konvolúciós réteg elkészítése

Ezt az osztályt a konvolúciós réteghez formálisan nagyon hasonló módon hozom létre. Ugyanazt az ősosztályt használom és a felülírt konstruktorfüggvény is ugyanazokat a bemeneteket kapja meg. De a *convolution* attribútum itt egy *pytorch.nn.Transpose2d* osztálybéli objektum. Ez is megkapja a bemeneti és a kimeneti csatornaszámot, valamint a *stride* és a *padding* paramétereket. A konvolúciós rétegemhez képest annyi az eltérés, hogy itt a transzponált konvolúció sajátossága miatt a kép szélein keletkező pixeleket el kell hagyni, amit egy paraméter formájában jelezni kell. A *forward* függvény megegyezik a konvolúciós rétegem *forward* függvényével.

Teljesen konvolúciós architektúrák megalkotása

Ezzel már minden építőkő rendelkezésemre áll a teljesen konvolúciós architektúra elkészítéséhez. Ahogy azt az 5.4.2.2 fejezet bevezetésében is említettem, 21380 alkalommal tanítottam be hálókat, ennyi féle hálóállapotból indítottam tanítást. Ehhez az egyik változó paraméterem a háló mélysége volt, azaz az, hogy hány konvolúciós réteget és transzponált konvolúciós réteget alkalmazok. Így több, egész pontosan 5 féle architektúrát készítettem.

Minden esetben szintén a *pytorch.nn* modulból származtatom osztályomat, melynél inicializálásnál csupán a bemeneti csatornaszámot kell megadni. Első lépésként egy konvolúciós réteget hozok létre attribútumként,

amely a 3 színcsatornát kapja bemenetként és az architektúra bemeneti csatornaszámát adja vissza. Ezután ***n*** darab leskálázás és konvolúció követi egymást, amiket szintén konvolúciós rétegek valósítanak meg. A leskálázások közül az ***i***-edik paraméterei: 2^{i-1}-szer az architektúra bemeneti csatornaszáma, ennek a kétszerese, a kernel mérete, ami 3 és a *stride*, ami 2. A konvolúció esetében pedig mind a két paraméter 2^{i}-szer az architektúra bemeneti csatornaszáma. Ezt követően attribútumot csinálok ***n*** db transzponált konvolúciós rétegből, melyek közül az ***i***-edik paraméterei 2^{n-i+1}-szer és 2^{n-i}-szer az architektúra bemeneti csatornaszáma. Végül pedig létrehozok egy osztályozót, amely a *pytorch.nn* modul *Conv2d* osztályának egy példánya. Paraméterei az architektúra utolsó transzponált konvolúciós rétegének kimeneti csatornaszáma, az osztályok száma, ami esetemben 2, illetve a kernel mérete és a hozzá tartozó *padding*. Ezen utóbbi két értéket 5-re és 2-re állítottam be, hogy az utolsó réteg nagyobb területre lásson rá.

Az itt megírt *forward* függvényben először el kell tárolni lokális változókban az ***n + 1*** darab konvolúció eredményeinek referenciáját. Ezután pedig ezeket hozzá kell adni a megfelelő transzponált konvolúciók kimeneteihez. Végül pedig vissza kell térni az osztályozó utolsó módosított transzponált konvolúciós rétegre adott visszatérési értékével. Ez az összeadás, azaz a felskálázó és leskálázó részek összekötése egy elismert és biztos módszer a tanítás konvergenciájának segítésére [104][105].

Tanító és validáló függvények létrehozása

A *trainer* függvényem bemenetei az aktuális epoch sorszáma, a hálóra mutató referencia, egy, a saját megírt adathalmazom osztályába tartozó objektumpéldány, a *pytorch.optim* könyvtár egy optimalizációs algoritmusa nulla kezdeti gradienssel, a *pytorch.nn* modul egy kritérium-osztálya, valamint az aktuálisan használt képek szélessége és magassága. A pontosság méréséhez létrehozok három lokális változót: egy olyat, ami számlálja, hogy hány képet használt fel az algoritmus, egyet, ami megmutatja, hogy ezek között átlagosan milyen pontossággal dolgozott, valamint egy olyat, ami megmondja az aktuális epochra jutó hibát. Ezt követően szükséges még a *pytorch.nn.Module* osztály *train* függvényével tanítási módba állítani a hálót, majd egy ciklussal végig lehet iterálni a bemenetként megadott adathalmazomon, mivel annak felülírtam a *__getitem__* függvényét és így az egyes képek és címkéik kaphatók vissza. Ezeknek az *enumerate* segítségével a sorszámát is elkérem. Itt megjegyezném, hogy kétféle hardvert használtam tanításra. Hangsúlyosan egyet, amely rendelkezik NVIDIA Quadro RTX 3000 videókártyával és verziószáma 452.57 és kisebb mértékben egy másikat, amelyben NVIDIA Quadro P 2000 videókártya van 385.54-es verziószámmal. Előbbihez CUDA 11.0.3 Update 1, utóbbihoz CUDA 9.0 (9.0.76) drivert tudtam használni [106]. A cikluson belül meghívom a megépített hálót és kimenetét odaadom a *__call__* operátorral meghívott kritérium osztálynak bemenetként. Ez visszaad egy *loss* értéket, aminek minden egyes paraméterére elvégzendő deriválást a *backward* függvény végzi el. Ezen számítás segítségével már frissíthetők az optimalizáló paraméterei a *step* függvénnyel. A frissítés után foglalkozok a statisztikával, növelem a *loss*-ban tárolt hiba skalár értékével az aktuális epochra jutó hibát és minden egyes kép minden egyes pixele esetén a *pytorch.max* segítségével a

háló kimenetéből megkapom a legvalószínűbbnek becsült osztályt. Módosítom a felhasznált képek változóját és a pontosság változóját, majd logolom az aktuális epoch számot, a *loss* értéket és a pontosságot ezután pedig visszatérek utóbbi kettővel.

A *validation* függvény ettől csak nagyon kis mértékben tér el. Itt a fent említett képszámláló, pontosságot meghatározó és egy epochra jutó átlagos pontosságot meghatározó változókon túl az 5.4 szerinti IoU értéket is számolom és logolom.

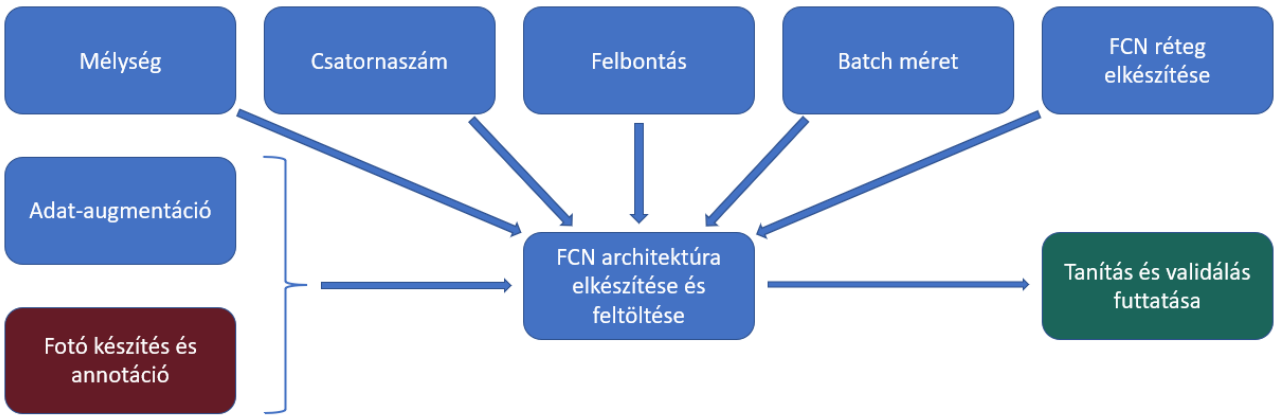
Futtató függvény létrehozása

Így már elkezdhető a tanítás, ehhez használok egy saját *run* függvényt. Ez megkapja, hogy milyen mélységű hálót milyen bemeneti csatornaszámmal használjon, valamint az egyes képek méretbeli paramétereit és a batch méretet. Először a reprodukálhatóság végett minden használt eszköznek fixálom a belső állapotát a megfelelő *seed* segítségével, aminek értékét 84210-ra választottam. Ezek az eszközök a *random*, a *numpy.random*, a *pytorch,* és a *pytorch.cuda.*

A tanító, validáló és tesztelő adathalmazok összeállítása az erre alkalmas függvényemmel történik. Itt a kizárólag tanításnál alkalmazott saját adat-augmentáción túl további transzformációkat is definiálok. A képek és a címkék esetén is átméretezést végzek. Az első esetben bilineáris interpolációval, a második esetben pedig a legközelebbi érték elvén. Ezt követően a bemenetben adott batch mérettel létrehozom a tanító adatbázist. Végül a tanítást végző ciklus előtt definiálom a kritérium-osztályt, az optimalizációs algoritmust és a lépéshossz módosításának módját. Kritériumom a nemzetközi gyakorlatban jelenleg egyik legjobbnak ítélt kereszt-entrópia [107][108], algoritmusom hasonló megfontolások alapján az Adam algoritmus [109][110], a lépéshosszt pedig 0.001-re állítom és minden futtatott 10 epoch után tizedére csökkentem úgy, hogy összesen 50 epochot futtatok.

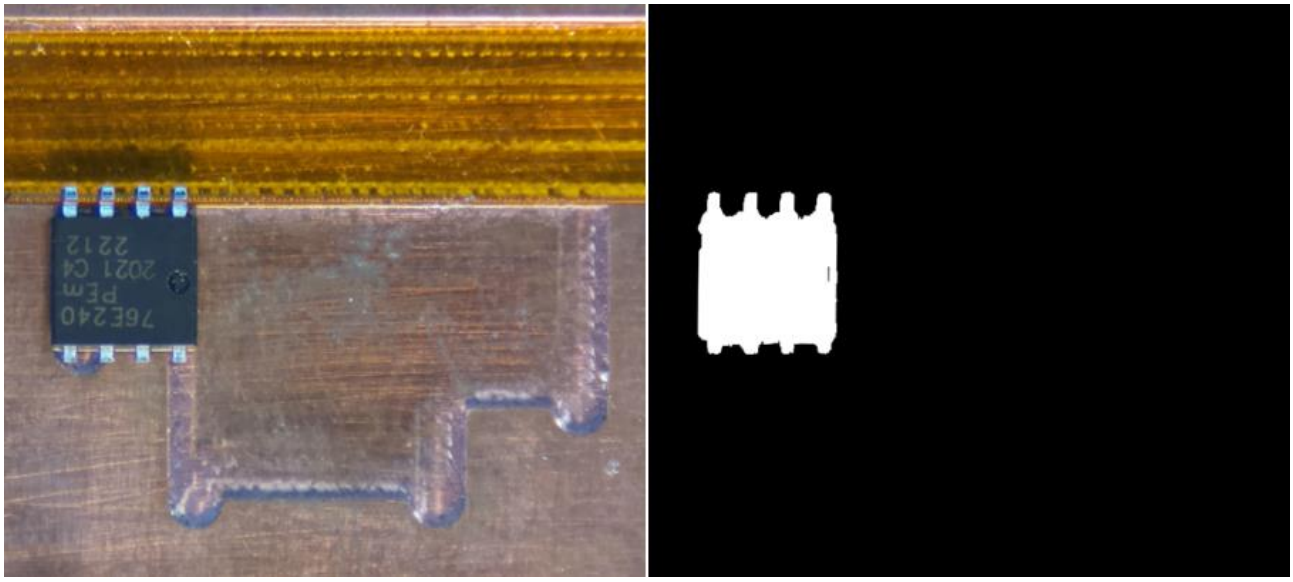
Main függvény létrehozása

Legvégül megírtam a main függvényt, amellyel 5 féle teljesen konvolúciós architektúrát hozok létre ***m = 2, 3, 4, 5, 6*** értékekkel, az utolsó kivételével 8, 16, 32, 64 csatornaszámokkal. Az ***m = 6***, 64 kombinációt a GPU memóriájának mérete nem tette lehetővé. Ezeket az opciókat még variálom felbontás és batch méret szerint. Alkalmazott felbontásaim függőleges-vízszintes pixelszám bontásba: 64-128, 128-192, 192-320, 256-384, 320-512, 384-640, 448-704, 512-832, 576-896, 640-1024, 704-1152, 768-1216 és 832-1344, batch méreteim pedig 128, 64, 32, 16, 8 és 4. Természetesen itt is korlátozó tényező a GPU memória, nagyobb mélység vagy csatornaszám esetén kisebb felbontás engedhető meg és fordítva is ugyanez igaz. Minden lehetséges esetben 50 tanítást végeztem. A tanító-validáló-tesztelő képek aránya 233-111-111 volt. A szükséges annotációkat kézzel végeztem a makesense.ai segítségével, ahogy azt az 5.4.2.2 fejezet bevezetésében is említettem. A teljes adatkészítést, háló építést és futtatást jól jellemzi a 28. ábra. Külön a neurális hálóról illusztráció az F10 függelékben található.



28. ábra: Szemantikus szegmentáció algoritmusának jellemzése teljesen konvolúciós neurális háló esetén

A bordó színnel jelölt rész manuálisan történt azonban ezt csak egyszer kellett megcsinálni, most már működőképes eszköz áll rendelkezésemre. Az eredmény jól megfigyelhető a 29. ábrán.



29. ábra: Mesterséges intelligencia használata inhomogén színű rézlap esetén

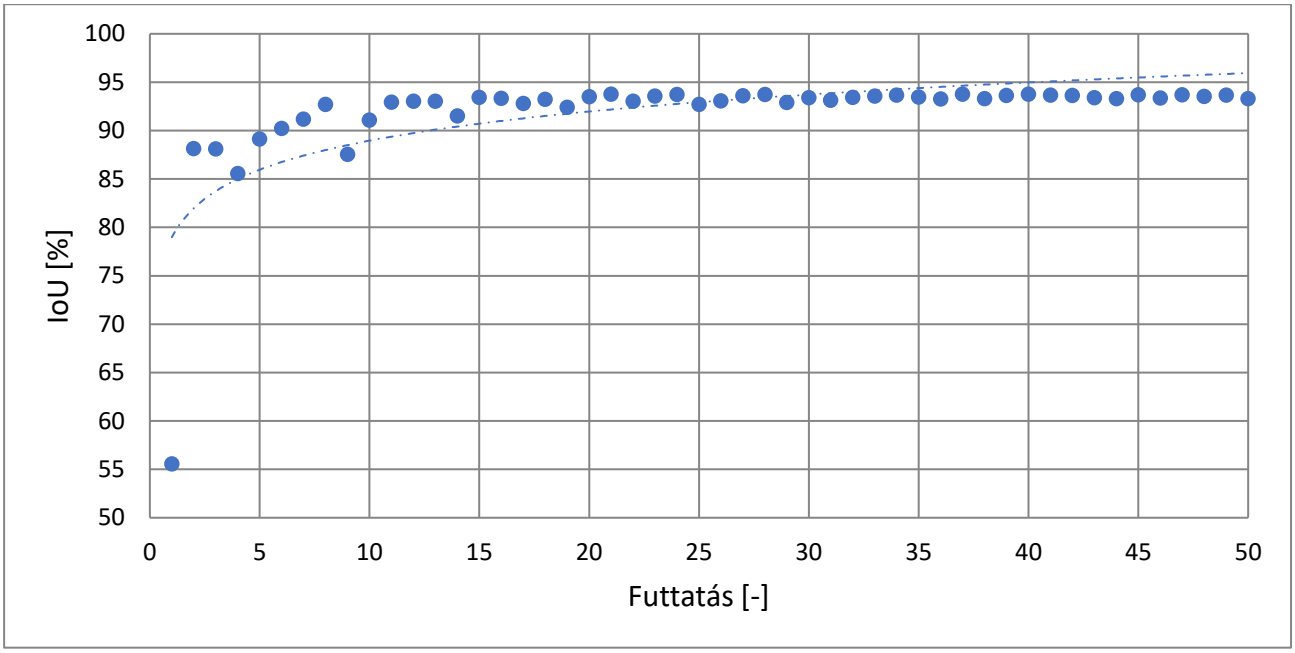
Ezt érdemes összevetni a korábbi 27. ábra szemantikus szegmentációjával. Így látható igazán, hogy mekkora előrelépés a megalkotott mesterséges intelligencia alapú algoritmus.

5.4.2.3 Szegmentáció kiértékelése

Ahogy azt már az 5.4 fejezetben felvezettem, a szemantikus szegmentáció minőségét jellemző nemzetközileg elfogadott érték az IoU (Intersection over Union). A 2. táblázatban azt is megmutattam, hogy különböző ismert módszerekkel nemzetközi gyakorlatban széleskörűen használt benchmarkokon milyen eredmények érhetők el.

Az én esetemben egy speciális feladathoz kerestem megoldást. Szemantikus szegmentáció szempontjából a feladat lényege az volt, hogy egy fényes rézlapra helyezett elektronikai komponensnek minél pontosabban ismerjem fel a pixeleit. Jellemzően ezek a komponensek MOSFET-ek és színük döntően kétféle: ezüstös fém,

és fekete. Ez a probléma értelemszerűen kevésbé komplexnek mondható, mint mondjuk egy PASCAL VOC 2012 adathalmaz [87] különböző közlekedési eszközeinek és állatainak szegmentálása, éppen ezért elvárható egy jobb eredmény. Igyekeztem a már ismert 2. táblázatban feltüntetett neurális hálóknál kisebb, kompaktabb hálókat építeni. Ennek oka az volt, hogy vélhetőleg egy 20 objektum kategóriára kifejlesztett komplex háló, az én speciális problémámra nem a legalkalmasabb. Ezt a várakozásomat támasztották alá eredményeim is.



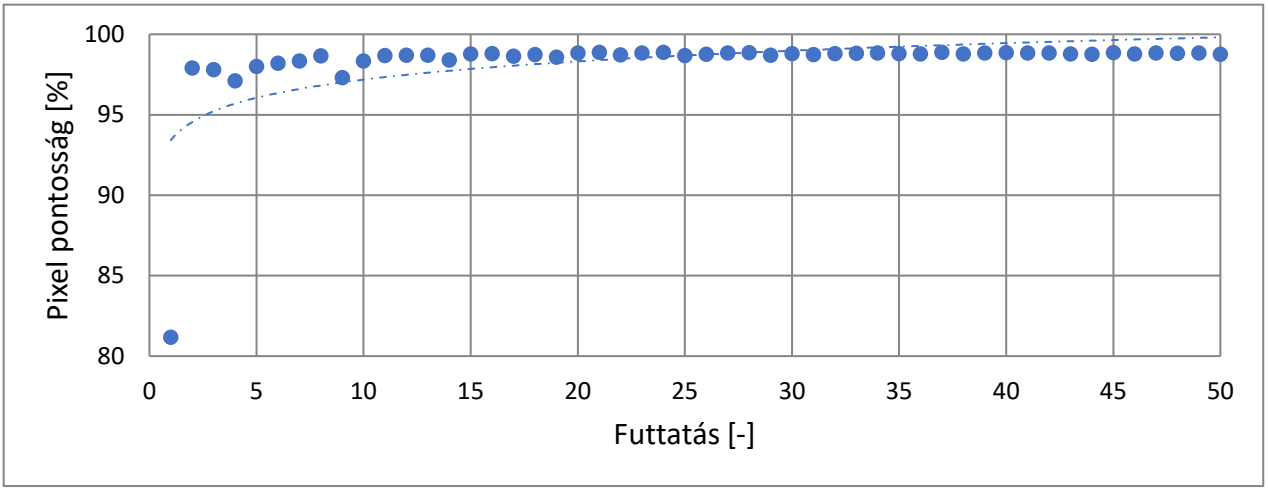
30. ábra: A legjobb eredményt elért háló futásának eredményei epochonként

A 30. ábra a legjobb validációs IoU értéket adó háló epoch-onkénti futtatási eredményeit mutatja be. A problémám megoldására ilyen módon a legalkalmasabbnak a 8 bemeneti csatornával rendelkező, ***m = 5*** mélységgel jellemzett, 4 batchmérettel dolgozó 576 · 896 felbontást használó háló teljesített a legjobban. Ezáltal látható, hogy nem a legkomplexebb háló eredményezi a legjobb megoldást. Legjobb validációs eredménye 93,78 % volt, a teszt adathalmazon pedig 94,01 %-ot ért el. Az, hogy mely hálóim érték el a legjobb eredményeket, szintén alátámasztja azt, hogy a háló komplexitás és az elért eredmény minősége között nincs egyértelmű korreláció. A második helyen a 8 bemeneti csatornával rendelkező, ***m = 5*** mélységgel jellemzett, 4 batchmérettel dolgozó 768 · 1216 felbontást használó háló végzett 93,73 %-os IoU értékkel. Ezt követte a 64 bemeneti csatornával rendelkező, ***m = 4*** mélységgel jellemzett, 4 batchmérettel dolgozó 192 · 320 felbontást használó háló 93,72 %-kal. Ezen legjobb hálók hasonló komplexitásúak, hiszen az 1. és 2. között csupán a felhasznált felbontásban van minimális különbség, a 3. pedig bár eggyel kevesebb réteggel rendelkezik, de ezt kompenzálja a nagyobb csatornaszáma. Ez azt sugallja, hogy körülbelül ezen a ponton, a 4-5 réteg határánál 4 esetében nagy csatornaszám, 5 esetében kis csatornaszám esetén lehet az optimum. Ezt igazolja az is, hogy a 93,5 % felett teljesítő 8 hálóból 5, köztük az első 4, ilyen paraméterekkel rendelkezett. Ezen hálók eredményeit tartalmazza a 3. táblázat.

3. táblázat: 93,5 % felett teljesítő hálók

Háló / IoU [%]	Mélység [-]	Csatornaszám [-]	Felbontás [h · w]	Batch méret [-]	Pixel pontosság [%]
93,78	5	8	576 · 896	4	98,88
93,73	5	8	768 · 1216	4	98,88
93,72	4	64	192 · 320	4	98,84
93,58	5	8	704 ·1152	4	98,84
93,56	3	32	320 · 512	4	98,81
93,55	6	16	384 · 640	4	98,79
93,53	4	64	256 · 384	4	98,85
93,51	4	16	448 · 704	4	98.83

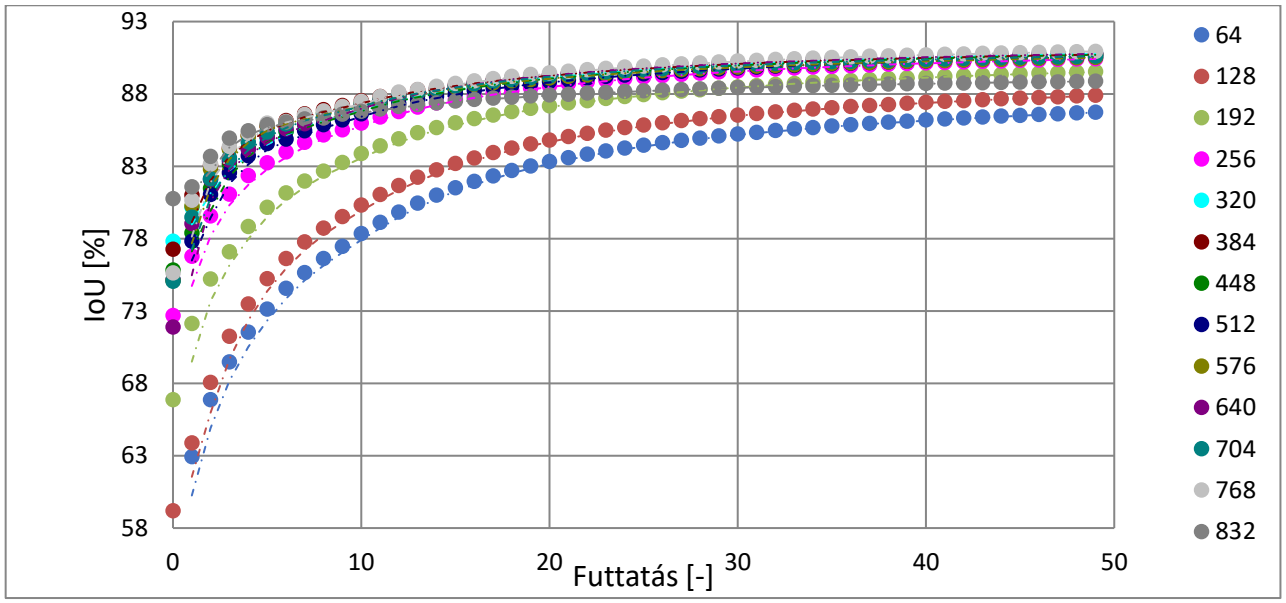
Megjegyzendő még, hogy egyértelműen látható, hogy ilyen kicsinek mondható rendelkezésre álló adat esetén batch méretnek a 4 volt a legmegfelelőbb. Ezen hálók epoch-okra bontott eredményei megtekinthetők az F11 függelékben. Szintén kiemelendő, hogy mivel képeim esetében közelítőleg azonos a 0, azaz nem MOSFET és az 1, azaz MOSFET osztályba tartozó pixelek aránya így az egyes hálók IoU értékei és pixel pontosság értékei között jelentős korreláció van. Amelyek az egyik mérőszám tekintetében jól teljesítenek, azok a másik esetében is, és amelyek az egyik tekintetében nem, azok a másik esetében sem. Ez olyannyira igaz, hogy ez a legjobb nyolc IoU értékkel rendelkező háló adja a legjobb nyolc pixel pontosságot is. Ezek a pontosságok 98,8 – 98,9 % körül mozognak, ahogy az a 3. táblázatban megfigyelhető. Az első sor hálójának eredményei alább láthatók, míg a többi eredmény az F12 függelékben.



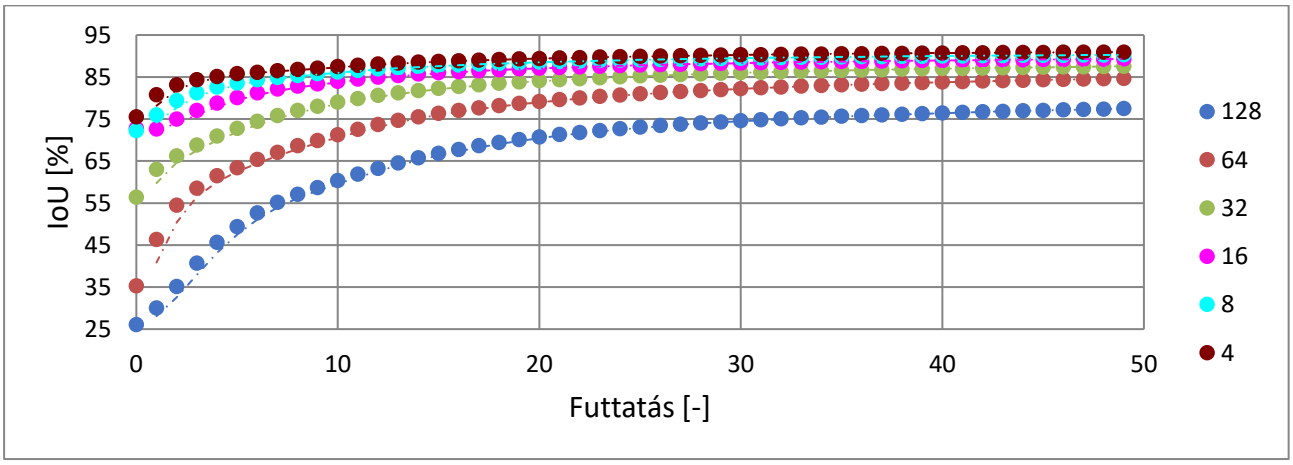
31. ábra: Legjobb háló pixel pontossága

Jellemzően a 7. és 8. háló esetén, ahol nagyobb pixel pontosság társul kisebb IoU értékhez, ott több a fals negatív pixel meghatározás. Ez jobb, mintha fals pozitív pixeleket adott volna vissza a háló, hiszen a szegmentáció célja meghatározni az 5.5 fejezetbéli kulcspontkereséshez a megfelelő területet. Így ha nagyon sok lenne a fals pozitív találat, akkor olyan pontok is maradhatnának a kulcspontkeresési tartományban, amik nem a MOSFET-hez tartoznak igazából és eltorzítanák az elmozdulás becslését. A következő négy ábra

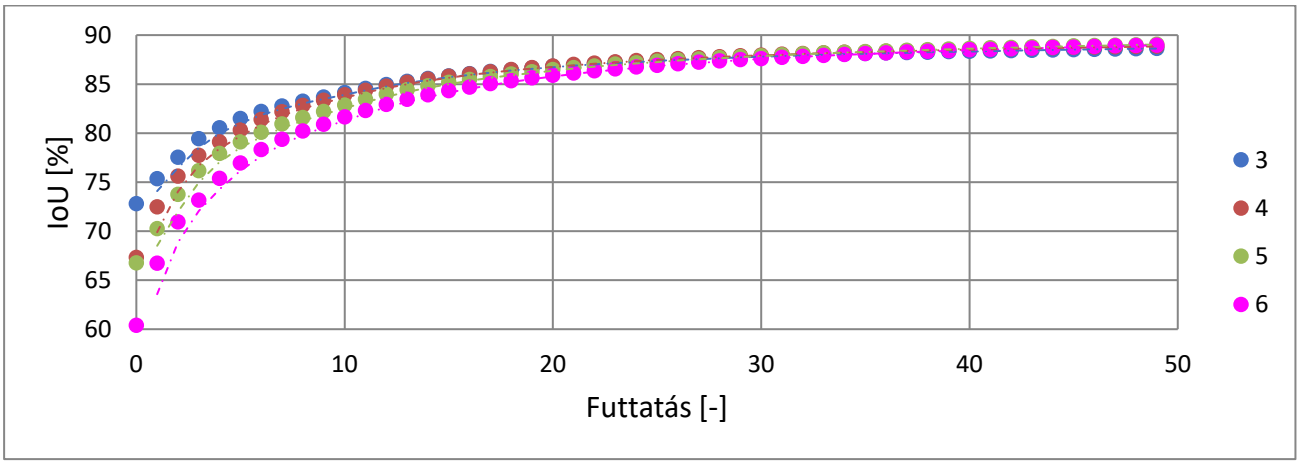
összefoglalja összes futtatásom átlagos eredményét képfelbontás, batch méret, háló mélység, és bemeneti csatornaszám szempontjából. A képfelbontás esetén a függőleges pixelszám van feltüntetve.



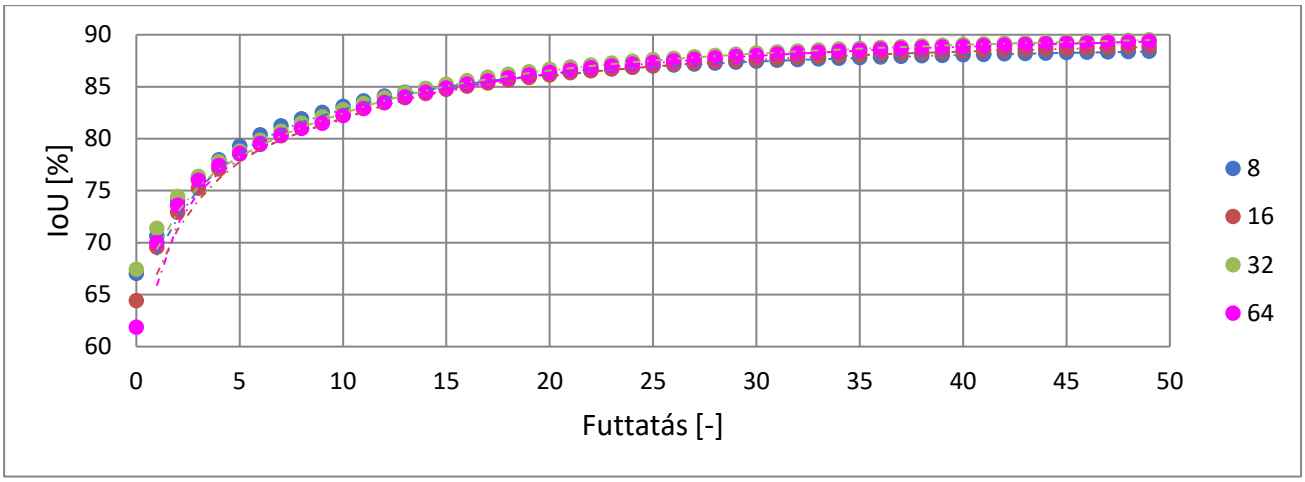
32. ábra: Különböző felbontású képeket használó hálók eredményei



33. ábra: Különböző batch mérettel dolgozó hálók eredményei



34. ábra: Különböző mélységű hálók eredményei



35. ábra: Különböző bemeneti csatornaszámú hálók eredményei

Általánosságban elmondható, hogy az átlagolások esetén sokkal inkább monoton viselkedéshez hasonlító javulás figyelhető meg az IoU érték esetén az iterációk számának növelésének függvényében. Olyan tendenciák is észrevehetők, amiket a legjobb nyolc futtatás kapcsán még nem sikerült meglátni. Például, hogy a nagyobb felbontású képek jobb eredményt biztosítanak, illetve, hogy noha a legjobb eredményeket nem ez produkálta, de általánosságban egy picivel jobb eredményt ad a 32 bemeneti csatornaszám a többinél. A 3. táblázathoz hasonlóan szintén a 4 batch méret megfelelősége mellett tanúskodik a 33. ábra és a 34. ábra is visszaigazolja, hogy az 5 a legjobb háló mélység. Pixel pontosság szempontjából ugyanezen kiértékelések hasonló eredményre vezetnek. Az ezekhez kapcsolódó diagramok az F13 függelékben találhatók. Ezzel kiértékeltem szemantikus szegmentációimat. Eredményeimet később, a kulcspont keresés tárgyalását követően nemzetközi kontextusba is helyezem az 5.7 fejezetben.

5.5 Kulcspont keresés [112][113]

Az 5.4 fejezetben meghatároztam azt a tartományt, ahol a megfelelő kulcspontok kereshetők az elmozdulás kiszámításához. Ezeket a kulcspontokat a SIFT (Scale Invariant Feature Transform) detektor segítségével találtam meg, melynek szabadalma 2020 márciusában járt le, így már díjmentesen alkalmazható az iparban.

Ez az algoritmus alapvetően sarokpontokat keres és ezek lokális környezete alapján készít egy, a perspektív torzítás kivételével minden transzformációra invariáns leírót. Ehhez az egyes éldetektálást igénylő problémáknál elterjedten alkalmazott DoG (Difference of Gaussians) szűrőket alkalmazza [114]. Ezeket több méretben végigfuttatva a képen 3 paraméter szerint keres lokális maximumokat az algoritmus. Ezek a szűrőméret, valamint az x és y koordináták. Az ilyen lokális maximumok nem diszkrét értelemben értendők, hanem folytonosan, mivel a SIFT többváltozós polinomot illeszt a kiszámított pontjaira. Ennek eredményeként *pixel szintűnél nagyobb pontosság érhető el vele*, ami az én alkalmazásom esetében különösen fontos. A generált leíró 128 elemű és egy 1 dimenziós tömbben kerül eltárolásra. Ez minden detektált pont $16 \cdot 16$ elemű környezetében különböző irányú intenzitásgradiensekből tevődik össze

súlyozással, ugyanis a sarokponttól távolabbi deriváltakat kisebb mértékben veszi figyelembe a kód, mint a közelebbieket. A legnagyobb derivált iránya lesz a referenciairány, ahonnan számítható az adott pont leírójának, összesített intenzitásgradiensének forgatása. Így például, ha lefuttatom egy képen és elforgatottján, akkor ugyanúgy megtalálja az egyes kulcspontokat a kód. A 10 fokonként kapott 36 derivált között, ha van olyan, amelynek értéke eléri az eredeti sarokpont 80 %-át, akkor ehhez is készül egy leíró. Továbbá még ezt a már elforgatott 16 · 16-os környezetet 16 darab 4 · 4-es környezetre osztva és 1-re normálva ez az algoritmus ismét elvégzendő 10 helyett 45 fokos felbontással ezzel biztosítva a skálainvarianciát. Ez adja a végső leírót. Tehát az 1-re való normálással biztosítható a lineáris fényesség változásra való érzéketlenség. Amennyiben ez a változás nemlineáris, az sem ront nagy mértékben a módszer hatékonyságán, mivel a normálás igazából úgy történik, hogy az egyes gradiensekhez meghatározott egy küszöbérték (0,2), és a normálás előtt az ennél kisebb értékeket erre módosítja a kód. Ezzel az egyes leírók orientációja hangsúlyosabb lesz, mint a nagyságuk.

5.6 Korrekciószámítás

Így tehát ha már megtörtént a referencia és a vizsgálandó képen is a szemantikus szegmentáció és SIFT detektorral meg lettek keresve ezeken a kulcspontok és leíróik, akkor már elvégezhető a korrekciószámítás.

5.6.1 Eltolás korrekció

Először az eltolás korrekcióját mutatom be. Ehhez első lépésként az OpenCV BFMatcher osztályának match függvényével összepárosítom a leíróik alapján az egyes pontokat [115]. Ez alapján konfidencia sorrendben megkaphatom az egyes talált pontpárokat. A felhasználó által előre beállítható konfigurációs állományban megadható, hogy ezen pontpárok közül hány darabot használjon fel az algoritmusom a továbbiakban. Jellemzően az optimális érték a 20 – 30 tartományba esik, innen minden tesztelésemnél minden lehetőség kielégítő volt. Erre az 5.7 fejezetben részletesebben is kitérek. Ezt követően kiszámítom ezen pontpárok x és y koordinátáinak különbségeit és a *numpy.mean,* illetve a *numpy.stdev* függvények segítségével meghatározom ezen különbségek átlagát és szórását. Ezt követően a szóráson kívülre eső különbségekhez tartozó pontpárokat elhagyom. Ezzel a 20 – 30 kiválasztott már eleve jó minőségű detekcióból megmarad a jellemzően 15 – 20 legjobb. Azért szükséges két lépcsőben két különböző minőségi szűrést végezni, mert így lehet csak kiszűrni az olyan hibákat, mint például, amikor a SIFT detektor egy D2PAK két különböző lábát detektálja azonosnak. Az ilyen hibából jellemzően kevés van és kiugróan más távolságot eredményez a többi értékhez képest, így a szórás elhagyásával biztosan eltűnik. Végül átlagot számítok a megmaradt különbségekből és ezzel megkapom a megfelelő eltolás korrekciót.

5.6.2 Elforgás korrekció

A két korrekció közül az elforgás korrekció a nehezebben számítható. Alapvetően az 5.6.1 fejezetben említett ***n*** érték megválasztásánál a felhasználó csak páros értéket választhat. Ennek oka, hogy az így kiválasztott

legjobb pontpárokat az elforgatás számításhoz párokba rendezem és vektorokat definiálok köztük. Így a referenciaképen és a vizsgálandó képen is keletkeznek vektorok és ezek elforgásainak átlagából megkapható az elforgás értéke. Ez matematikailag indokolható. Tulajdonképpen a MOSFET a képen egy síkbeli alakzat, amelynek kezdetben súlypontja legyen az origóban, amely körül történik egy elforgatás, majd a teljes MOSFET egy eltolást is elszenved. Ezek a transzformációk a MOSFET egy tetszőleges *i*-edik pontján következő változtatásokat eredményezik:

$$\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \begin{pmatrix} x_i \cdot \cos(\alpha) - y_i \cdot \sin(\alpha) + a \\ x_i \cdot \sin(\alpha) + y_i \cdot \cos(\alpha) + b \\ 1 \end{pmatrix} \tag{41}$$

ahol:

x_i	az i-edik pont x koordinátája,
y_i	az i-edik pont y koordinátája,
α	az elforgatás szöge,
a	az eltolás x menti komponense,
b	az eltolás y menti komponense.

Továbbá tekinthető a MOSFET egy tetszőleges *j*-edik pontjából egy tetszőleges *k*-adik pontjába mutató vektor és ezen vektor pontjaiból kapott elforgatott és eltolt pontok vektora (41) alapján:

$$\overrightarrow{JK} = \begin{pmatrix} x_k - x_j \\ y_k - y_j \\ 0 \end{pmatrix} \tag{42}$$

$$\overrightarrow{J'K'} = \begin{pmatrix} x_k \cdot \cos(\alpha) - y_k \cdot \sin(\alpha) + a \\ x_k \cdot \sin(\alpha) + y_k \cdot \cos(\alpha) + b \\ 1 \end{pmatrix} - \begin{pmatrix} x_j \cdot \cos(\alpha) - y_j \cdot \sin(\alpha) + a \\ x_j \cdot \sin(\alpha) + y_j \cdot \cos(\alpha) + b \\ 1 \end{pmatrix} \tag{43}$$

Ami tovább alakítva:

$$\overrightarrow{J'K'} = \begin{pmatrix} (x_k - x_j) \cdot \cos(\alpha) - (y_k - y_j) \cdot \sin(\alpha) \\ (x_k - x_j) \cdot \sin(\alpha) + (y_k - y_j) \cdot \cos(\alpha) \\ 0 \end{pmatrix} \tag{44}$$

Ez pedig éppen azt adja, mintha az origó körül α szöggel elforgattam volna a J pontból K pontba mutató vektort. Következésképpen a vektorok hajlásszöge megadja a MOSFET elforgatását.

Tehát az egyes pontpárok pontjai által meghatározott helyvektoroknak ezt követően meghatározom az x tengellyel bezárt szögét a *mat.atan* függvény segítségével és ezeket a szögeket egymásból kivonva, megkapom az egyes vektorok hajlásszögét. Ezen hajlásszögek esetén is az 5.6.1 fejezethez hasonlóan átlagot és szórást számolok és csak a szóráson belüli eredményekből számítom ki végleges eredményem.

5.7 Eredmények kiértékelése

Ezzel elérkeztem szoftverem eredményeinek kiértékeléséhez. Szemantikus szegmentáció terén már az 5.4.2.3 fejezet kapcsán végeztem minőségjellemzést munkámmal kapcsolatban. Most ezt a korábban hivatkozott szakirodalmakkal is összevetem, ezt követően pedig megvizsgálom, hogy a teljes rendszer teljesíti-e a 100 µm eltolási és 1° elforgási pontosságot

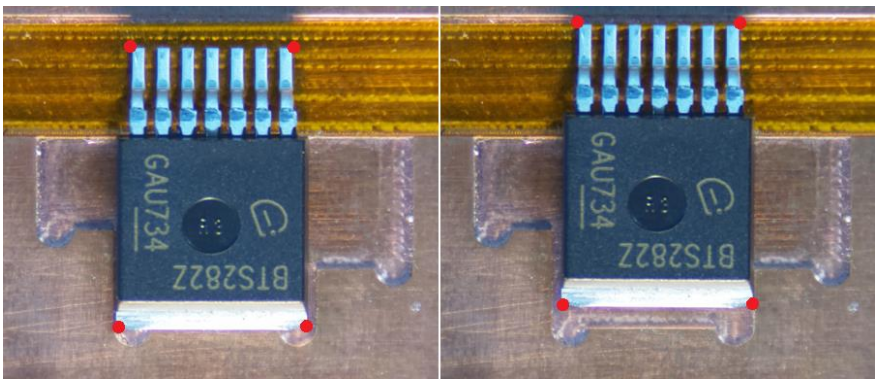
5.7.1 Szemantikus szegmentáció nemzetközi vonatkozásban

Általánosságban, szemantikus szegmentáció esetén az egyes FCN-hálók IoU értékei 60 – 80 %-os tartományban mozognak az egyes nagy benchmarkokat vizsgálva, ahogy azt a Szakirodalmi összefoglalóban, az 5.4.2.1 fejezetben tárgyaltam. [81]-[89] Ennél speciálisabb feladatoknál természetesen jobb eredmény is elérhető különböző architektúrákkal. Ahogy említettem, esetemben nagyon szegényesen publikált az elektronikai komponensek szemantikus szegmentációjának területe, de más területekről megvizsgálhatók speciális alkalmazások. Például [50] bevezetésében több városi alkalmazás is összegyűjtésre került 80 – 90 %-os IoU értékkel. Az orvosi alkalmazások közül kiemelném [53]-t, ahol polypus colorectalis szegmentáció esetén 93,21%-os átlagos IoU értéket értek el. A nagy erőkkel kutatott és a jövő önvezető autóinak fejlesztése szempontjából igen fontos útszegmentálásnál [55] 85,9 % és 96,7 % közti értékeket tudott felsorakoztatni. Számítógépes látás szempontjából rosszabb minőségű, havas környezetben [56] szerint ez a szám inkább csak 50 % körül mozog. Fém felületek hibáit [64] több módszerrel 80 – 90 % közti értékekkel határozta meg, illetve [66] műholdképek alapján egyes városnegyedeket legfeljebb 87,43 %-kal szegmentált. Ezáltal az én újnak mondható alkalmazásom 94 % feletti teljesítménye abszolút megállja a helyét a speciális esetek között, azoknál nem teljesít rosszabbul.

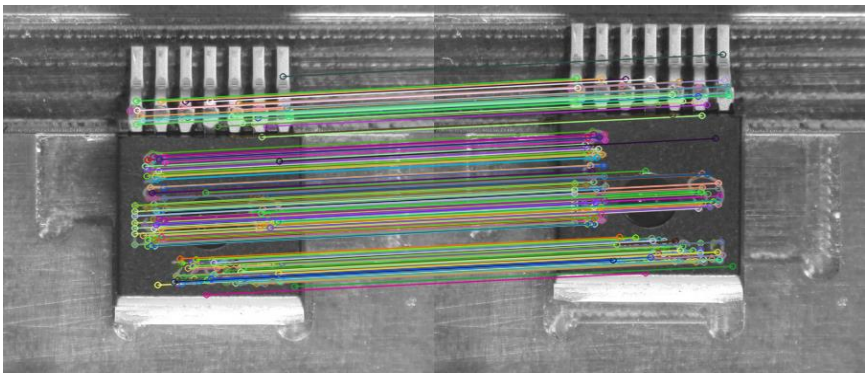
5.7.2 Pontossági vizsgálat

Teljes rendszerem felé fontos elvárás volt részemről és csapatom részéről is a biztos, robosztus működés. Éppen ezért részletes elemzést végeztem, valós mérési körülményeket létrehozva. Felszereltem az F6 függelék szerint a kamerám és futtattam a kódom, ami így képet készített egy komponensről. Ez volt az a komponens, ami a 15. ábra szerint a referenciának feleltethető meg. Majd egy másik, de ugyanabba a mintahalmazba tartozó komponenst elhelyeztem máshová és erre is lefuttattam a kódom, ami így a 23. ábrának megfelelően működött és visszaadta a korrekció értékeket, az eltolás vektort és az elforgást. Ezeket úgy ellenőriztem, hogy 10 µm pontosságú tolómérővel megmértem egyes pontjainak az elmozdulását és ezek vektoraiból számítottam elforgást is. Minden egyes komponens szimmetria tengelyével párhuzamosan vett leghosszabb méretét tekintettem a komponens hosszának. Ezen hossz két végpontjából indulva megkerestem a komponens szimmetriatengelytől legtávolabbi részeit úgy, hogy a szimmetriatengelyre merőlegesen haladtam. Ezáltal 4 pontot határoztam meg, amelyeket kulcspontnak tekintve hajtottam végre tolómérővel a mérés algoritmusát. Ezt 50 esetben végeztem el, ahol minden alkalommal megfelelő becslést

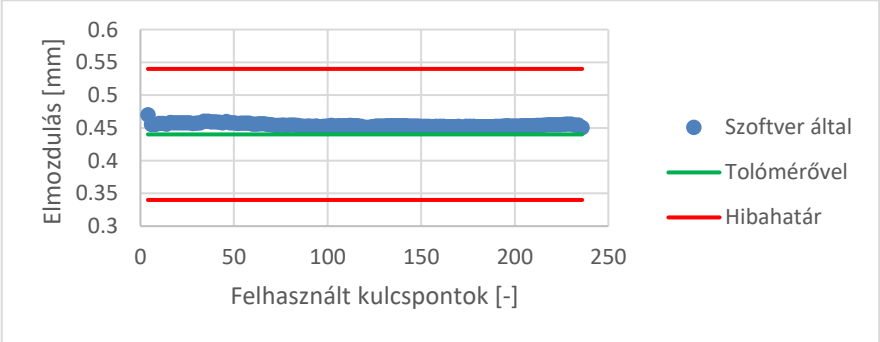
kaptam. Ezek közül ötöt itt részletesen is megmutatok, jelölve a szoftver által talált és a tolómérővel vizsgált kulcspontokat az egyértelműség kedvéért. Ezt a vizsgálatot összekötöttem annak vizsgálatával, hogy a felhasználó által milyen 5.6.1 fejezet szerinti felhasznált pont szám megadás a célszerű. Az alábbi ábrákon sorban láthatók az egyes referencia komponensek, vizsgálandó komponensek, a keletkező eltolást jellemző kulcspontok, valamint az x és y tengely menti eltolást, illetve a z tengely körüli elforgást jellemző korrekciók a felhasznált kulcspontok számának függvényében. Látható, hogy a releváns, gyakorlatban reális tartományokban végeztem vizsgálatot. Ez elmozdulás terén a tengelyek menti 0 – 1,5 mm tartományt, elforgásban a 0 – 3,5° tartományt jelentette. Minden egyes futtatásnál a vizsgálandó objektumról új kép is készült és ezzel futott le más választott kulcspont számmal az algoritmus, ezzel azt is vizsgálva, hogy az esetleges képi zajok mennyire befolyásolják a becslést.



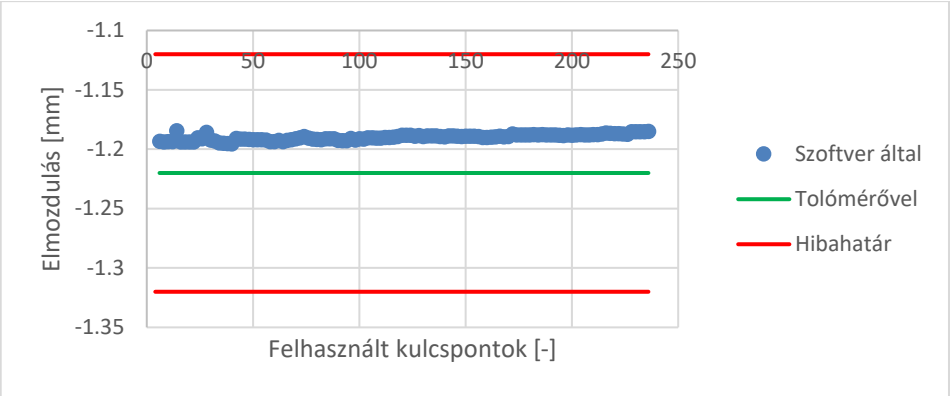
36. ábra: Első tesztobjektum és elmozdulása



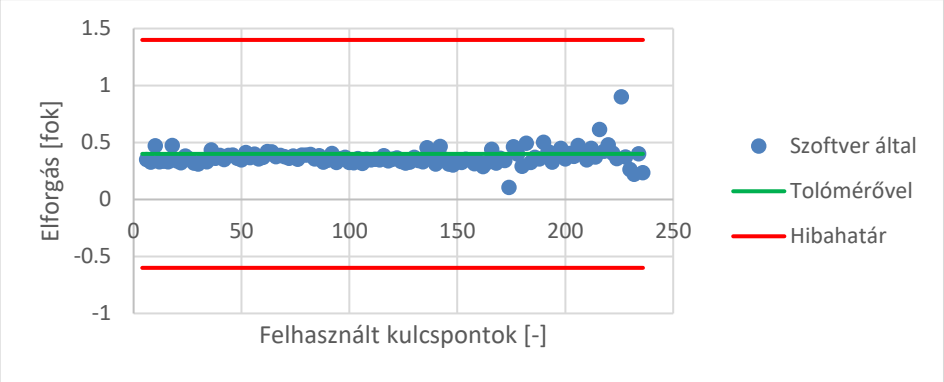
37. ábra: Elmozdulást leíró kulcspontok 1.



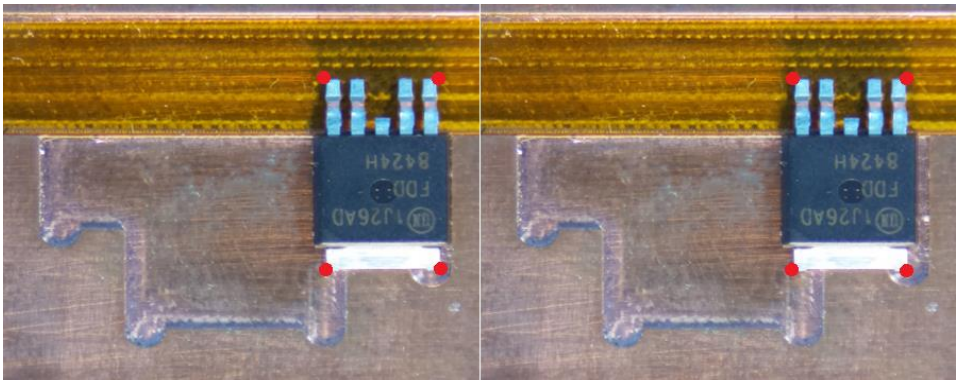
38. ábra: X tengely mentén számított elmozdulás 1.



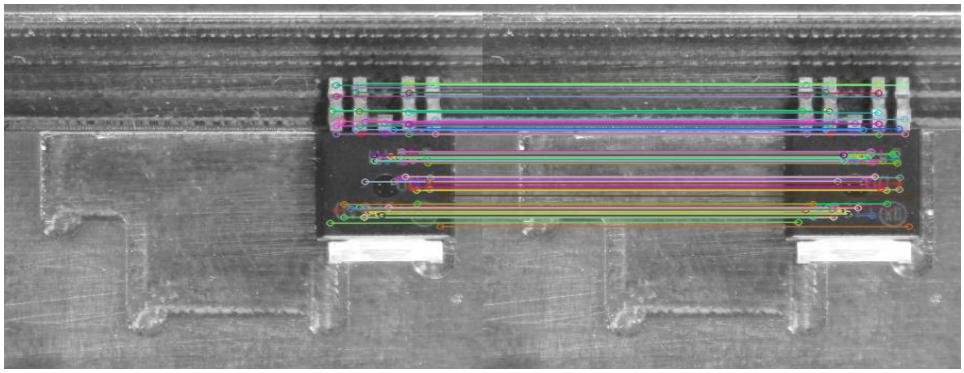
39. ábra: Y tengely mentén számított elmozdulás 1.



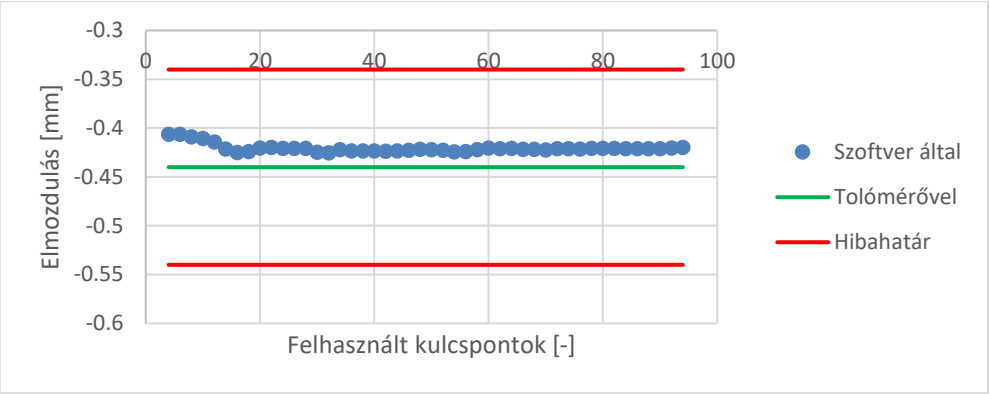
40. ábra: Z tengely körül számított elforgás 1.



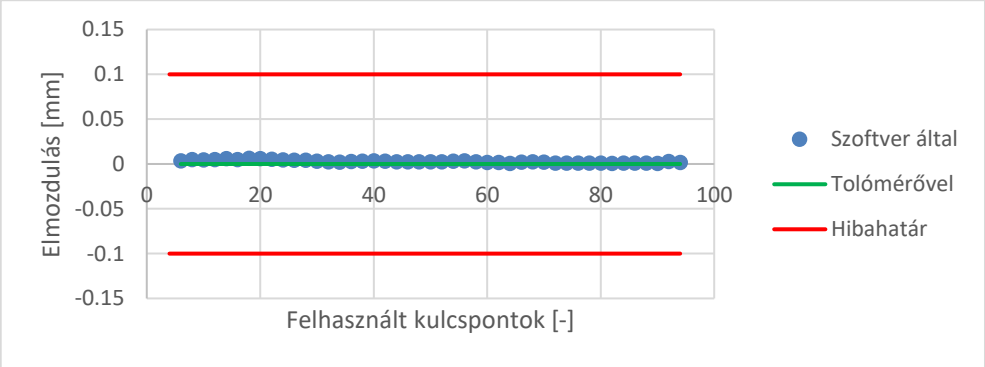
41. ábra: Második tesztobjektum és elmozdulása



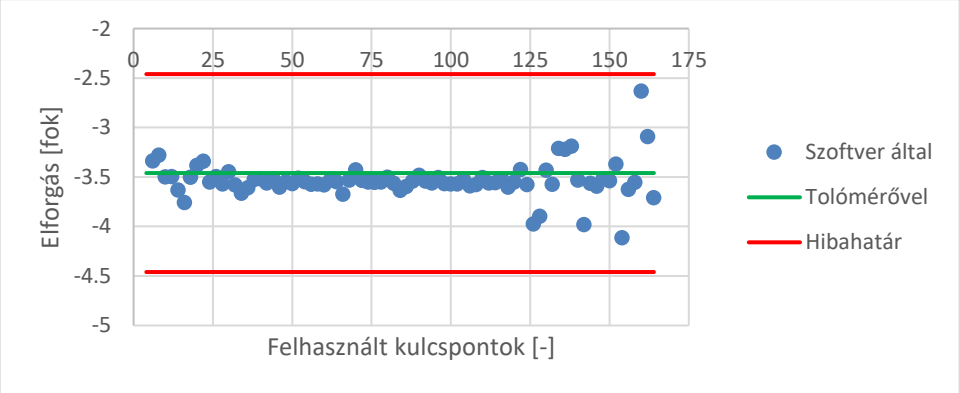
42. ábra: Elmozdulást leíró kulcspontok 2.



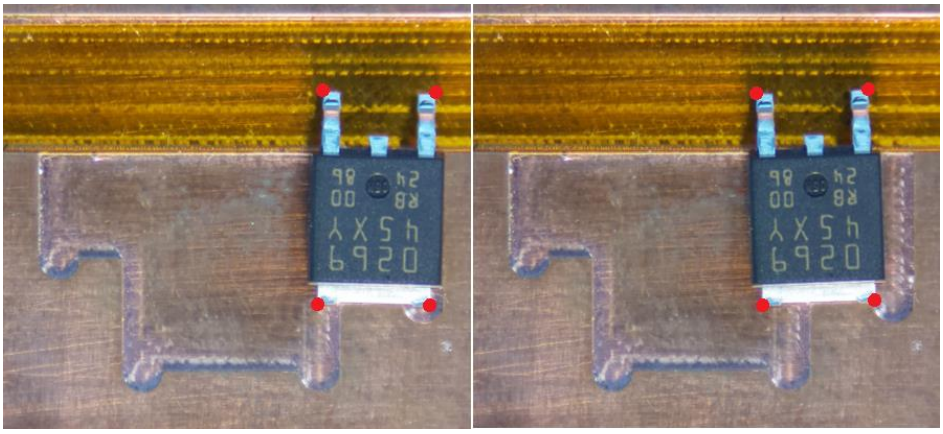
43. ábra: X tengely mentén számított elmozdulás 2.



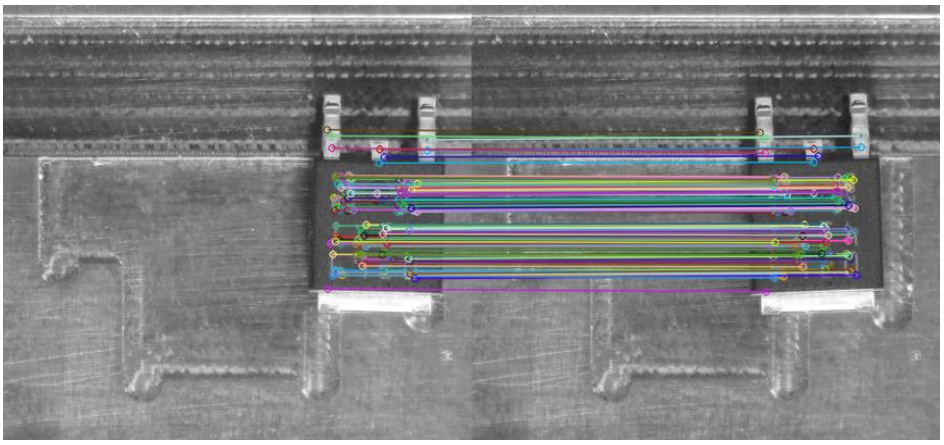
44. ábra: Y tengely mentén számított elmozdulás 2.



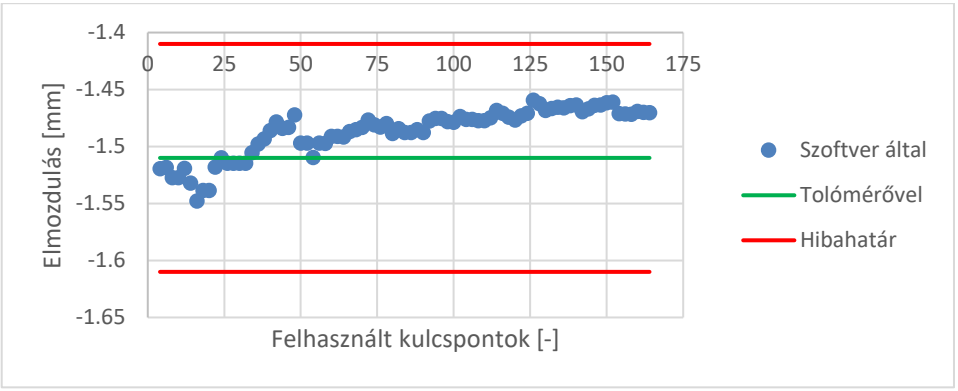
45. ábra: Z tengely körül számított elforgás 2.



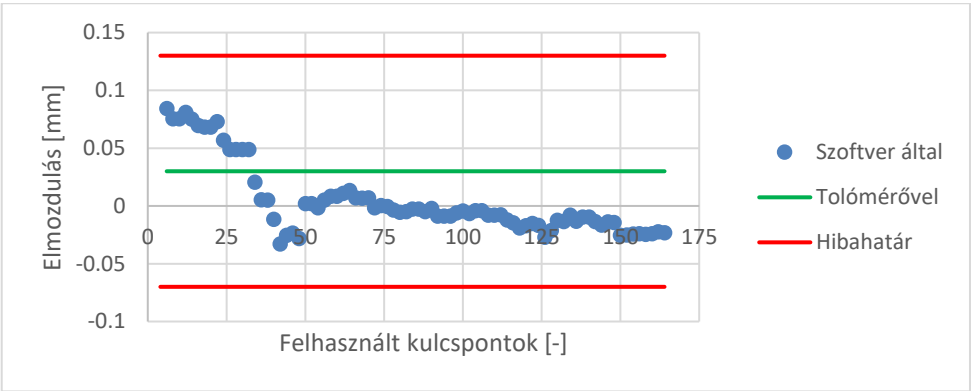
46. ábra: Harmadik tesztobjektum és elmozdulása



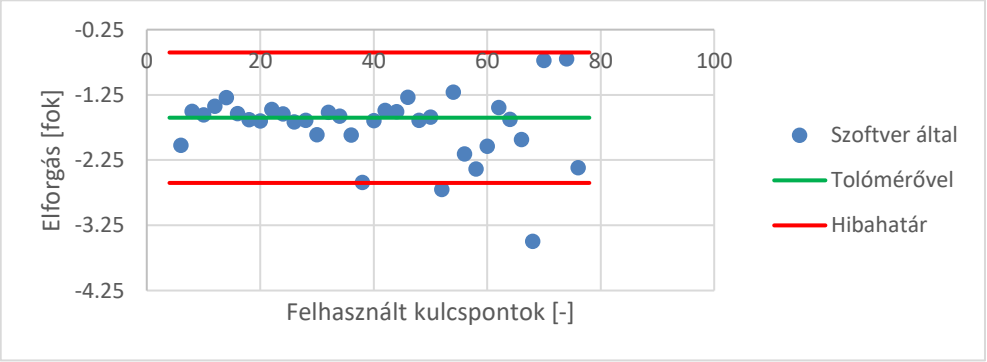
47. ábra: Elmozdulást leíró kulcspontok 3.



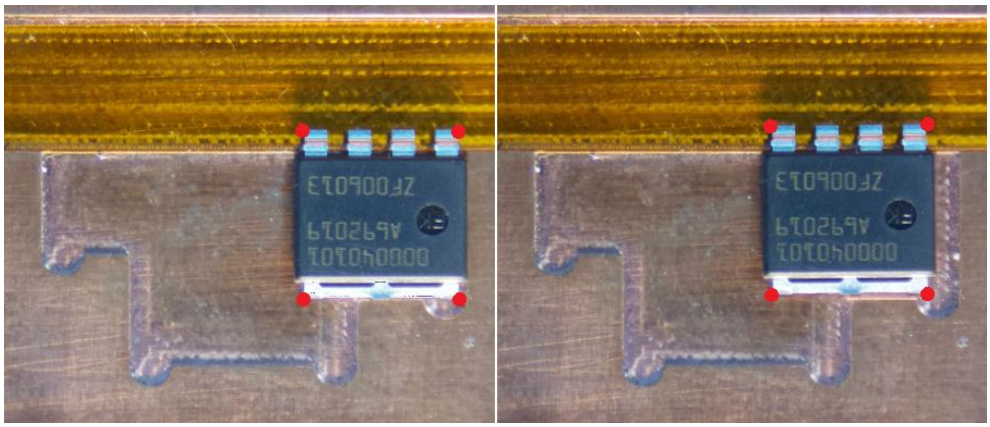
48. ábra: X tengely mentén számított elmozdulás 3.



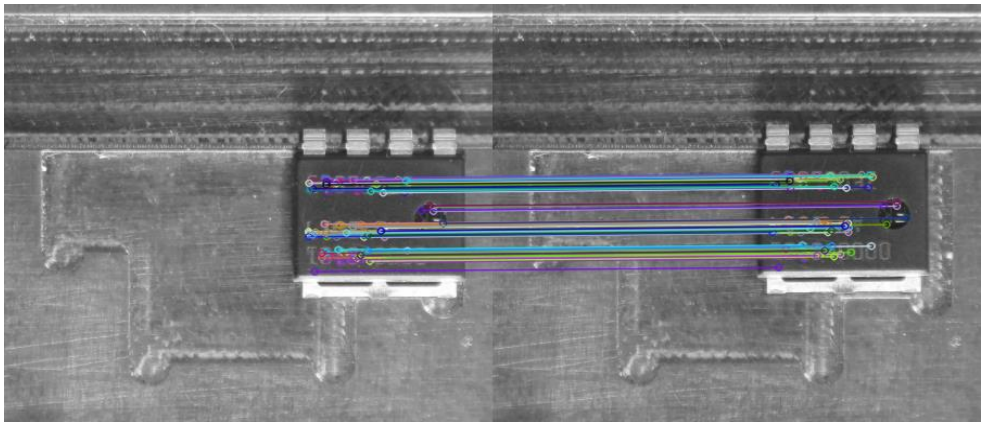
49. ábra: Y tengely mentén számított elmozdulás 3.



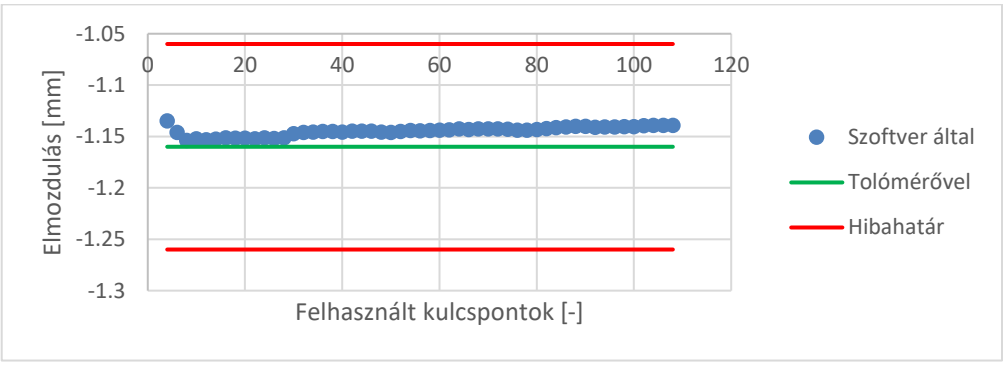
50. ábra: Z tengely körül számított elforgás 3.



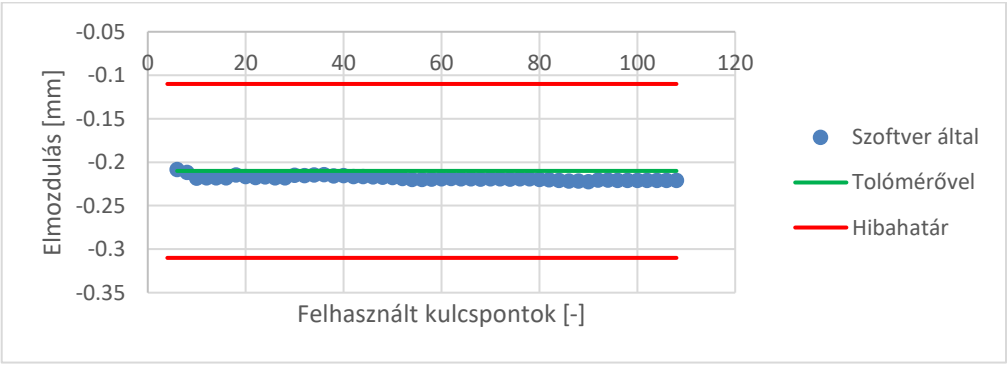
51. ábra: Negyedik tesztobjektum és elmozdulása



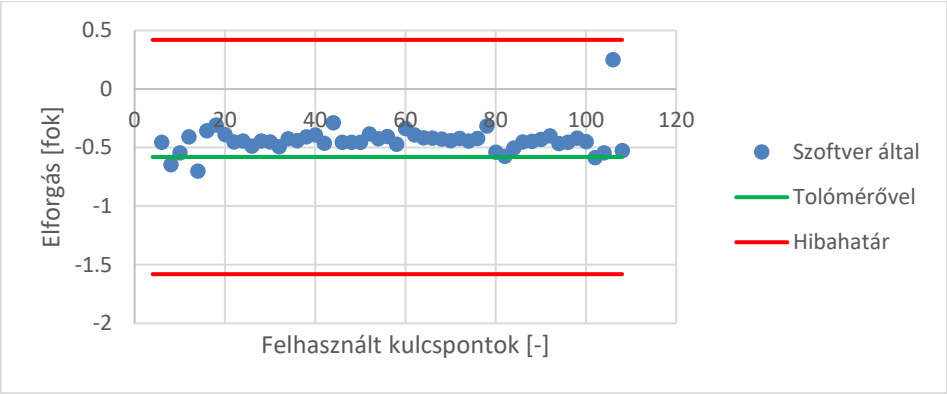
52. ábra: Elmozdulást leíró kulcspontok 4.



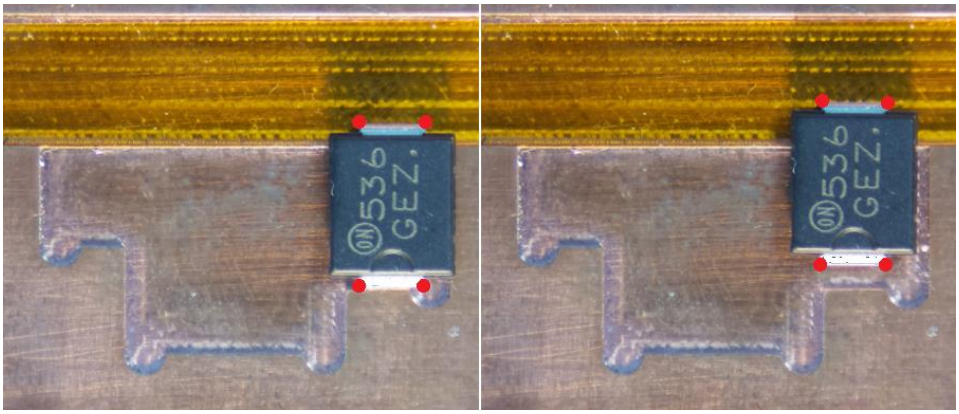
53. ábra: X tengely mentén számított elmozdulás 4.



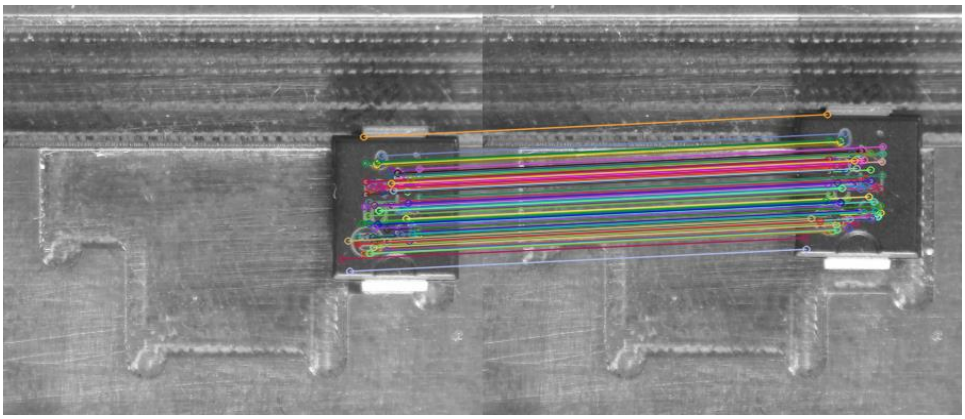
54. ábra: Y tengely mentén számított elmozdulás 4.



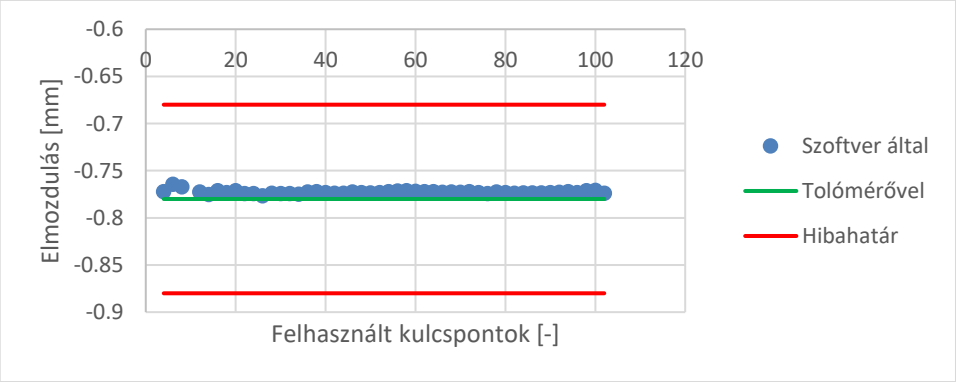
55. ábra: Z tengely körül számított elforgás 4.



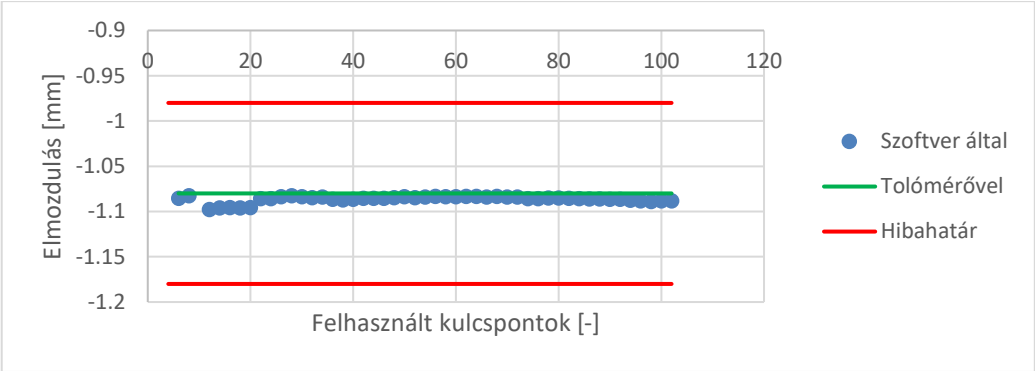
56. ábra: Ötödik tesztobjektum és elmozdulása



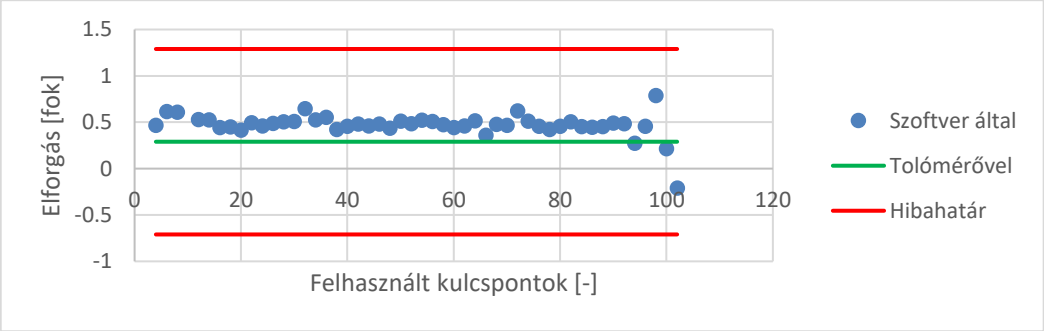
57. ábra: Elmozdulást leíró kulcspontok 5.



58. ábra: X tengely mentén számított elmozdulás 5.



59. ábra: Y tengely mentén számított elmozdulás 5.

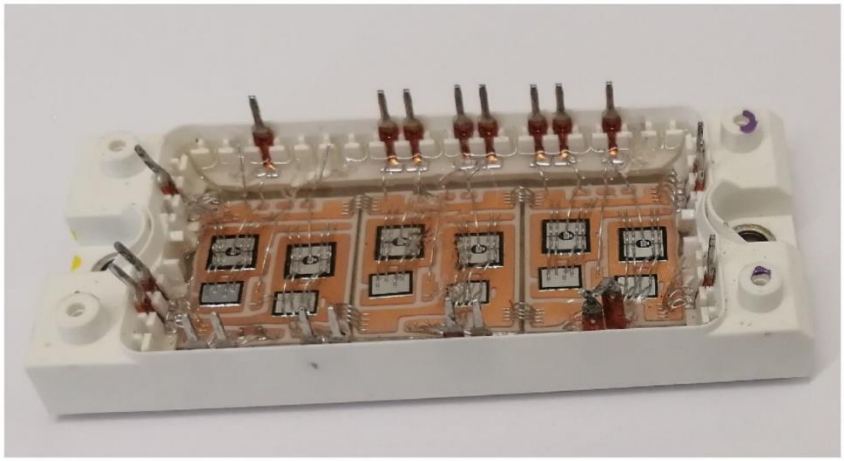


60. ábra: Z tengely körül számított elforgás 5.

Látható, hogy a felhasználandó kulcspont számra a 20 – 30 tartomány tényleg egy jó választás, itt az összes lehetőség mind elforgás, mind elmozdulás szempontjából elfogadható, de egyébként is csak a 3. objektum elforgásánál van bárhol a megengedettnél nagyobb eltérés. Egy-egy helyen megfigyelhető, hogy lépcsőzetesen változnak a becsült értékek a felhasznált kulcspont szám függvényében. Ennek oka az, hogy ahogy növelem a felhasznált kulcspontok számát, sokáig előfordulhat, hogy ez nem változtat jelentősen a szóráson, így a második minőségellenőrző szűrőn nem jutnak át igazán rossz becslést adó pontok. Viszont ha egy igazán rossz pont kerül be ezzel a növeléssel, akkor hirtelen megugrik a szórás és ezzel több pont átjut a második minőségellenőrzésen, ami jelentős becslésváltozást eredményez. Azonban ezek az értékek is végig a hibahatáron belül maradnak. Ezen kívül jellemzően kis offset hibával bőven a megengedett tartományon belül mozog a becslés. Ennek alakulásában a legtöbb esetben egyértelmű zajmentesség, folytonosság figyelhető meg, ami igazolja, hogy a képi zajok nem okoznak jelentős problémát a korrekció meghatározásában. Ez egyébiránt nemcsak a szoftver helyes működését, a torzításkorrekciókat, a szegmentáció megfelelőségét és kulcspontok általi korrekciókat igazolja, hanem a megfelelő hardverválasztást a kamera és optikája esetén, illetve a megfelelő hardvertervezést, a kameratartó esetében. A rendszer robusztusan működőnek mondható, hiszen mindig talál legalább kétszer annyi kulcspontot, mint ami feltétlenül szükséges és ezekből minden esetben hibahatáron belüli becslést ad minden elvégzett teszt esetében. Ezek a tesztek kiemelendő, hogy a valós mérések körülményeivel megegyező körülmények között készültek.

6 Továbbfejlesztési lehetőségek

A dolgozatban ismertetett rendszer kiterjeszthetőségét is szeretném alátámasztani. Így ebben a fejezetben röviden bemutatom, hogy hogyan lehetséges egy másik alkatrész esetén is mérésautomatizálást megvalósítani vele. A közvetlen kötésű réz kerámiahordozók (direct bonded copper, DBC) is az elektronikai ipar gyakran használt eszközei. Termikus méretezésük gyakran egy biztonságkritikus feladat, ugyanis olyan jellegű alkalmazásokba kerülnek beépítésre, mint a szervomotor vagy az automata váltó, ahol különösen veszélyes az üzem folyamán bekövetkező meghibásodás. Éppen ezért esetükben is igen fontos a T3Ster mérés hatékony alkalmazása.



61. ábra: DBC technológiát használó tápmodul illusztrációja [116]

A DBC-k kapcsán történő mérésautomatizálásban mindent a komponens mérésautomatizálásnak megfelelően csinálok, kivéve a szemantikus szegmentáció hiperparamétereinek meghatározását.

6.1 A szemantikus szegmentáció neurális architektúrájának módosítása

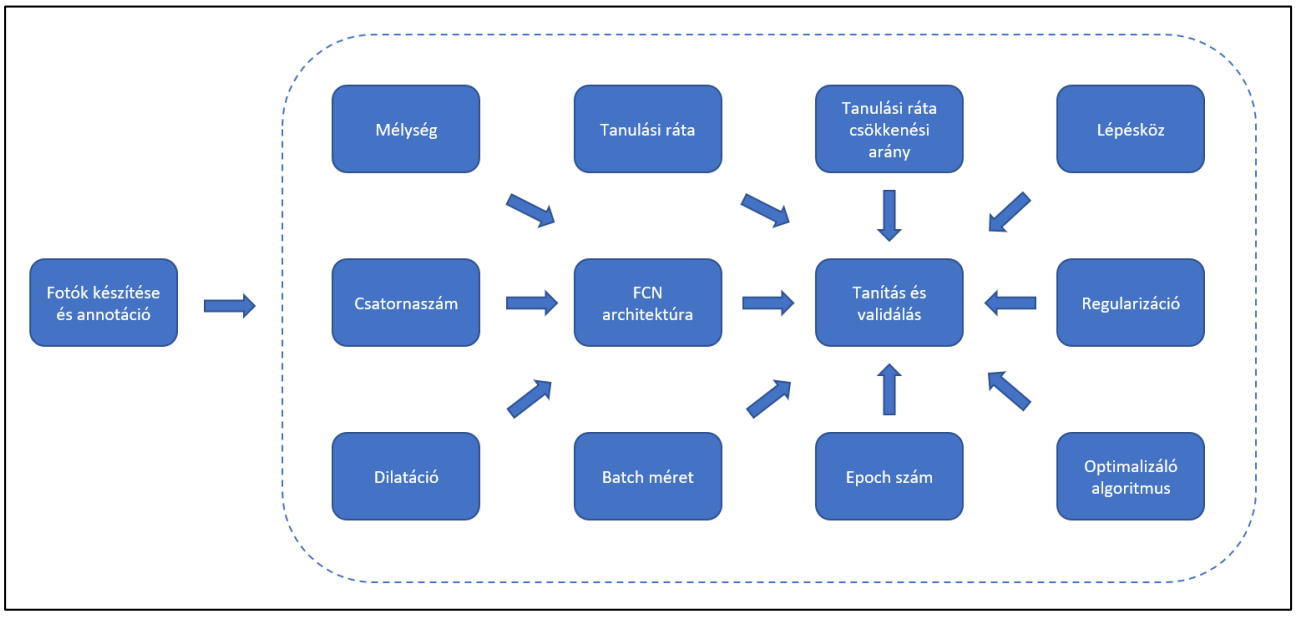
Az 5.4.2.2 fejezet szerinti megvalósításnak tehát néhány pontján változtatok. Egyrészt nem rács optimalizálással igyekszem bejárni a hiperparaméter teret, hanem ehhez optimalizációs algoritmust keresek. Ez egyúttal nagyobb, több dimenziós hiperparaméter teret enged meg, mivel így hamarabb érhető el az optimális konfiguráció [117]. Így több beállítást is módosíthatok, ezzel bemutatva, hogy ebből a szempontból is rugalmas a fejlesztett rendszer. A DBC-k esetében a felbontást 256 · 384 értéken fixálom, azonban a tanulási rátát, annak csökkenési arányát, a csökkenés lépésközét, valamint az alkalmazott epoch számot is variálom a batch méret, a csatornaszám és a háló mélység mellett. Továbbá a túlillesztés és az alulillesztés közötti optimalizáció érdekében variálva alkalmazok regularizációs súlyozást, illetve a hiperparaméter optimalizációs algoritmusra bízom a paraméter optimalizáló algoritmus kiválasztását is. Ezeken kívül még azt is lehetővé teszem, hogy dilatáció kerülhessen az egyes konvolúciós rétegekbe. Ez a paraméter azt mutatja meg, hogy a k · k méretű konvolúciós kernel egymás mellett lévő pixeleket dolgozzon fel, vagy alkalmazzon, esetemben egyetlen, pixel kihagyást. A változtatható hiperparamétereket a 4. táblázat foglalja össze.

4. táblázat: Optimalizálandó hiperparaméterek tartományai

Hiperparaméter	Minimum érték	Maximum érték
Háló mélység	1	7
Csatornaszám	6	16
Tanulási ráta	10^{-4}	10^{-2}
Tanulási ráta csökkenési aránya	0.1	0.5
Tanulási ráta lépésköze	10	30
Regularizációs súlyozás	10^{-5}	10^{-3}
Batch méret	1	16
Dilatáció	0	1
Epoch szám	40	90

A táblázat nem tartalmazza, de az optimalizáló algoritmust a 5.4.2.2 fejezetben említett Adam algoritmus és a Sztochasztikus Gradiens Módszer [118] közül választottam, mivel előbbi a nemzetközi gyakorlatban jelenleg az egyik legjobbnak ítélt, utóbbi pedig akkor lehet célravezető, ha a tanításom nem bizonyulna elég komplexnek az adathalmazhoz képest.

A tanító-validáló képek aránya 337-85 volt. A teljes adatkészítést, háló építést és futtatást jól jellemzi a 62. ábra:



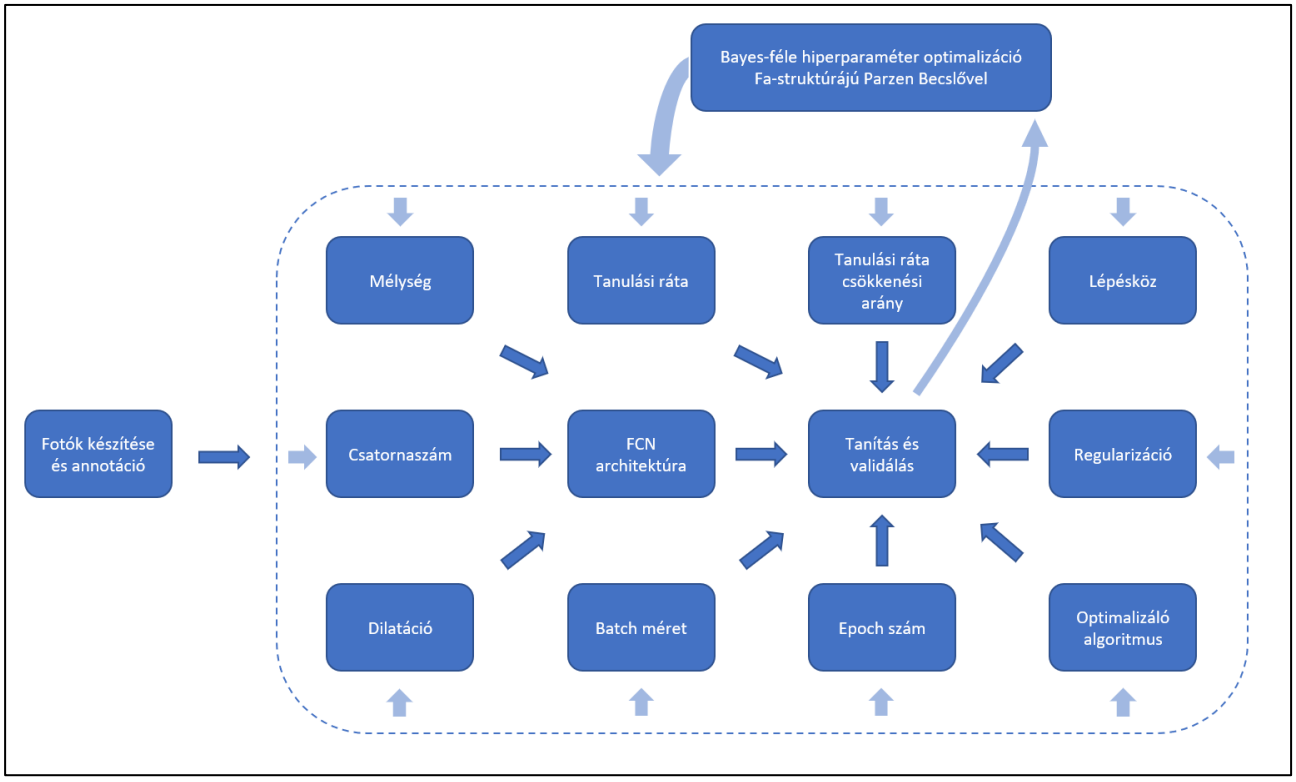
62. ábra: A DBC mérésautomatizáláshoz készített FCN architektúra tanításának folyamata

Azonban arról még nem esett szó, hogy a rendszer hiperparamétereit, vagyis az optimalizáló algoritmust, a dilatációt, a batch méretet, a csatornaszámot, a mélységet, az epoch számot, a regularizációs súlyozást, a

tanulási rátát, valamint a tanulási ráta csökkenési arányát és csökkenési lépésközét hogyan optimalizálom a 4. táblázatban említett tartományokban. Erről szól a 6.1.1-es fejezet.

6.1.1 Hiperparaméter-optimalizáció

A hiperparaméter-optimalizálás a tanuló algoritmusok fejlesztésének egyik legsarkalatosabb pontja [119]. Ha egy adatalapú problémához szeretnék minél jobb megoldást találni, akkor a problémamegoldó modellem kiválasztását követően, annak beállításait, hiperparamétereit is finomhangolnom kell, hogy minél pontosabb megoldást kaphassak. Jelen probléma esetében a mély neurális hálók használata azért jó választás, mert igazoltan alkalmasak olyan bonyolult nemlineáris összefüggések megtanulására, mint amilyenek a szemantikus szegmentációban is vannak. Ezen konvolúciós neurális hálók hiperparamétereinek optimalizálásához pedig találni kell egy jó metodikát. Azért is fontos a jó metodika megtalálása, mert jellemzően a sok adattal dolgozó konvolúciós neurális hálós megoldások esetében az egyes beállítások szerinti tanítások kiértékelése hosszú folyamat, így ahhoz, hogy minél gyorsabban minél jobb megoldást lehessen kapni, jó módszer szükséges. A 62. ábra így kiegészíthető az alábbi módon:



63. ábra: DBC-k FCN architektúra hiperparaméter-optimalizációjának folyamata

6.1.1.1 Manuális-, rács-, random- és algoritmikus optimalizálás

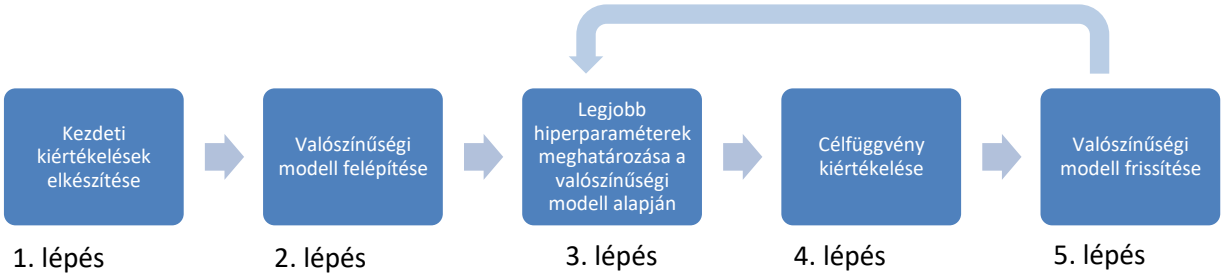
A hiperparaméterek optimalizálására többféle lehetőség is létezik [117][119]. A szoftverfejlesztési szempontból legegyszerűbb a manuális módszer. Ekkor kézzel történik az egyes értékek módosítása és az új futtatások indítása is, így figyelve, hogy milyen beállítások esetén kapható jobb megoldás a tanuló

algoritmustól. Egy kicsivel nagyobb odafigyelést igénylő megoldás, amikor több egymásba ágyazott ciklus segítségével egy adott tartományon belül az összes lehetséges modell konfiguráció megvizsgálásra kerül Ez abból a szempontból előnyösebb, hogy nem igényel folyamatos emberi jelenlétet, azonban továbbra is rendkívül időigényes. Így, ha az optimalizálás gyorsasága is szempont, akkor a hiperparamétertérnek csak egy kisebb környezetben vett lokális optimumát lehet vele megtalálni. A random keresés is egy lehetőség, azonban itt fennál a veszélye a legjobb hiperparaméter konfigurációk kihagyásának. Ugyanakkor gyorsabb lehet, mint a rács optimalizálás.

A leghatékonyabb és legkorszerűbb módszer pedig az algoritmikus optimalizálás, mivel ez egy adott futtatáson belül intelligensen képes kihasználni a korábbi iterációk eredményeit. Alapvetően a legtöbb korszerű hiperparaméter optimalizáló algoritmus alapja a Bayes-optimalizáció [120], noha vannak olyan esetek, amikor olyan speciális az optimalizálandó rendszer, hogy lehetőség van gradiens-alapú optimalizációra [121], vagy kellően gyorsan kiértékelhető a rendszer ahhoz, hogy alkalmazhatók legyenek evolúciós algoritmusok [122]. A Bayes-féle hiperparaméter optimalizáció lényege, hogy az optimalizációs iteráció minden lépésében felépít egy valószínűségi modellt az optimalizálandó célfüggvényről, azaz a neurális háló optimalizációs térbeli hibafüggvényéről, és ezt a modellt használja arra, hogy meghatározza, hogy a következő lépésben milyen hiperparaméter konstrukcióval kell kiértékelni a tényleges célfüggvényt. A Bayes-féle optimalizációt megvalósító algoritmusok, másként a szekvenciális modellalapú optimalizálási (Sequential Model-Based Optimization, SMBO) algoritmusok között a legjelentősebb különbség az, hogy milyen módon készítik el a célfüggvény valószínűségi modelljét. A két legnépszerűbb algoritmus, a Gauss-folyamat (Gaussian Process, GP) és a Fa-struktúrájú Parzen Becslő (Tree-stuctured Parzen Estimator, TPE) közül az utóbbit választottam, mivel ez több teszt alapján is eredményesebb volt [119]. Használták már például napelemes előrejelzés optimalizációhoz [123], forgógép hibadiagnosztikához [124], földcsuszamlás-érzékenység értékeléshez [125], és egészségmenedzsment alkalmazásokhoz [126], de nagyon sok más alkalmazása is van [127-130].

6.1.1.2 Bayes-féle optimalizálás Fa-struktúrájú Parzen Becslővel

A Bayes-féle hiperparaméter optimalizálás alapvetően a következő folyamatábrával írható le.



64. ábra: Bayes-féle optimalizáció

Látható, hogy az 1. és a 2. lépést csak egyszer kell elvégezni, míg a 3., 4. és 5. lépést ciklikusan ismételni kell. A megállási kritériumot jellemzően időhöz, iterációszámhoz, vagy célfüggvény értékhez szokás kötni. Az 1. lépés a szimulációs tér kezdeti véletlenszerű feltérképezését jelenti, hogy legyen alapja a valószínűségi modell felépítésének. Esetemben az *Optuna* [131] kísérletezési alapon meghatározott legjobb alapbeállítását, a 10 random iterációs beállítást használtam. A 2. és 5. lépés a TPE specifikus modellezés, a 3. lépés a TPE specifikus kritérium alapján való döntés, a 4. lépés pedig a neurális hálós futtatás.

Érdemes elsőként a kritériummal foglalkozni. Mitől lesz jó egy hiperparaméter konstrukció? Ennek meghatározására a TPE a Várható Fejlődés (Expected Improvement, EI) függvényét használja:

$$EI_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - \mathrm{y}) \cdot \mathrm{p(y|x)dy} \tag{45}$$

ahol az eddig nem említett

x az aktuális hiperparaméter konstrukció,

y^* a célfüggvény küszöbértéke. Az *Optuna* kísérletezési alapon meghatározott legjobb alapbeállítása szerint az alsó decilis, a 25. iteráció után pedig az alsó decilis és a 25. legkisebb érték közül a kisebbik,

y a célfüggvény aktuális értéke,

$\mathrm{p(y|x)}$ pedig a célfüggvény feltételes valószínűségi modellje.

Már ebből a formulából is világosan látszik, hogy olyan célfüggvény értékeket keresek ezzel az algoritmussal, amik kisebbek, mint a küszöbérték, hiszen ekkor lehet pozitív az integrál értéke. Tehát minimalizálom a hibát. Ismerve a várható fejlődés fogalmát célszerű rátérni a TPE legfőbb elemére, a valószínűségi modellre:

$$EI_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - \mathrm{y}) \cdot \frac{\mathrm{p}(x|y) \cdot \mathrm{p(y)}}{\mathrm{p(x)}} \mathrm{dy} \tag{46}$$

ahol az eddig nem említett

$\mathrm{p}(x|y)$ a hiperparaméter konstrukció feltételes valószínűségi modellje,

$\mathrm{p(y)}$ a célfüggvény valószínűségi modellje,

$\mathrm{p(x)}$ pedig a hiperparaméter konstrukció valószínűségi modellje.

Továbbá a TPE bevezeti $\mathrm{l(x)}$ és $\mathrm{g(x)}$ függvényeket úgy, hogy:

$$\mathrm{p}(x|y) = \begin{cases} l(x) \mid y < y^* \\ g(x) \mid y \geq y^* \end{cases} \tag{47}$$

Itt az $l(x)$ és $g(x)$ modellek Gaussi Keverékmodellként (Gaussian Mixture Model, GMM) [132] állnak elő úgy, hogy minden egyes adatra egy eloszlást kell illeszteni a hozzá jobbról és balról legközelebbi adatok közül a távolabb lévő távolságával megegyező szórással. Így, ha (46)-ból kiemelem $\frac{1}{\mathrm{p(x)}}$ -t:

$$EI_{y^*}(x) = \frac{1}{\mathrm{p(x)}} \int_{-\infty}^{y^*} (y^* - \mathrm{y}) \cdot \mathrm{p}(x|y) \cdot \mathrm{p(y)dy} \tag{48}$$

Akkor a Várható Fejlődés (47) alapján felírható úgy is mint:

$$EI_{y^*}(x) = \frac{1}{\mathrm{p(y < y^*)} \cdot \mathrm{l(x)} + (1 - \mathrm{p(y < y^*)}) \cdot \mathrm{g(x)}} \int_{-\infty}^{y^*} (y^* - \mathrm{y}) \cdot \mathrm{l(x)} \cdot \mathrm{p(y)dy} \tag{49}$$

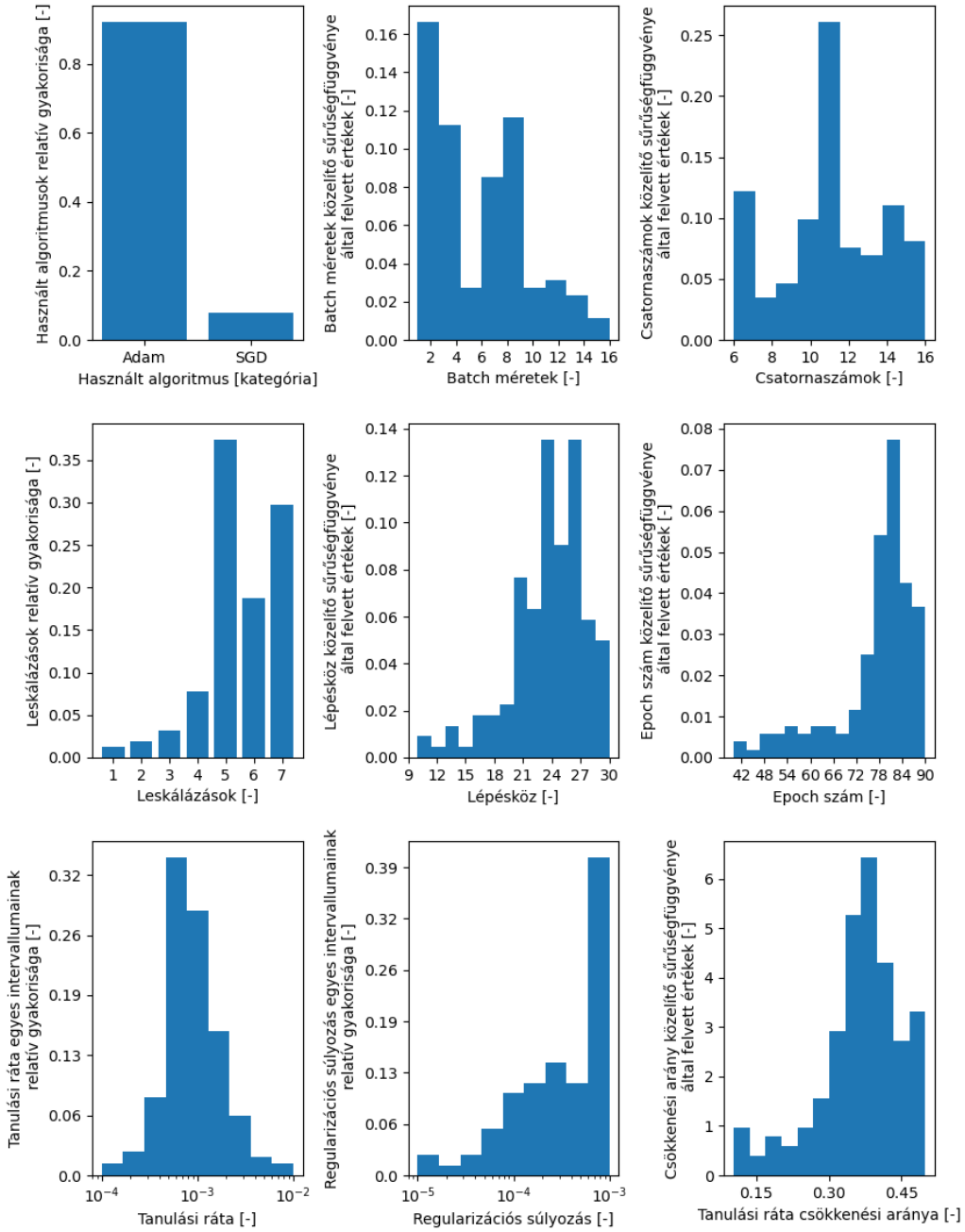
Amiből kiszámítható:

$$EI_{y^*}(x) = \frac{\mathrm{p(y < y^*)} \cdot y^* - \int_{-\infty}^{y^*} \mathrm{p(y)dy}}{\mathrm{p(y < y^*)} + (1 - \mathrm{p(y < y^*)}) \cdot \frac{\mathrm{g(x)}}{\mathrm{l(x)}}} \tag{50}$$

Ebből pedig látszik, hogy ha a $\frac{\mathrm{g(x)}}{\mathrm{l(x)}}$ kifejezés csökken, akkor az (50) tört nevezője is csökken, ezáltal a tört, azaz az $EI_{y^*}(x)$ nő, vagyis ekkor van a legnagyobb esélyem javítani a becslésemen. Ez gyakorlatilag azt jelenti, hogy a küszöbérték feletti célfüggvény értékek küszöbérték alatti célfüggvény értékekhez viszonyított aránya csökken. Tehát gyakorlatilag a kisebb hibájú becsléseket eredményező hiperparaméter konstrukciókhoz közeli konstrukciót igyekszik választani mindig a TPE algoritmus az optimalizálásnál.

6.1.2 Szegmentáció kiértékelése

A Parzen Becslő az 6.1.1.2 fejezetben leírtaknak megfelelően ott választja meg a hiperparamétereket a szimulációs terében, ahol a legjobban teljesít az állapotbecslés. Ezt a teljesítményt esetemben a legszigorúbb IoU metrika jelentette, hiszen ez képes a legátfogóbban értékelni egy becslés eredményét az 5.4-es bevezetésben leírtak szerint. Az optimalizálás eredményét a 65. ábra összegzi. Egyértelműen az Adam algoritmus bizonyult jobbnak optimalizálás szempontjából. Ez nem is annyira váratlan, hiszen a szemantikus szegmentáció egy komplex probléma a készített tanító adathalmaz pedig ehhez mérten kicsi, így nem szükséges az SGD módszert alkalmazni. A Batch méretek közül a kisebbek bizonyultak megfelelőnek, tehát akkor működik jobban a tanítás, ha egyszerre csak kevesebb kép alapján kell a konvolúciós kernelek súlyparaméter-optimalizációját végezni. A csatornaszámok terén a 11-es érték körül volt az optimum, tehát ennyi párhuzamos kernel mellett működik a legjobban a rendszer. A leskálázások, azaz a mélység terén is az

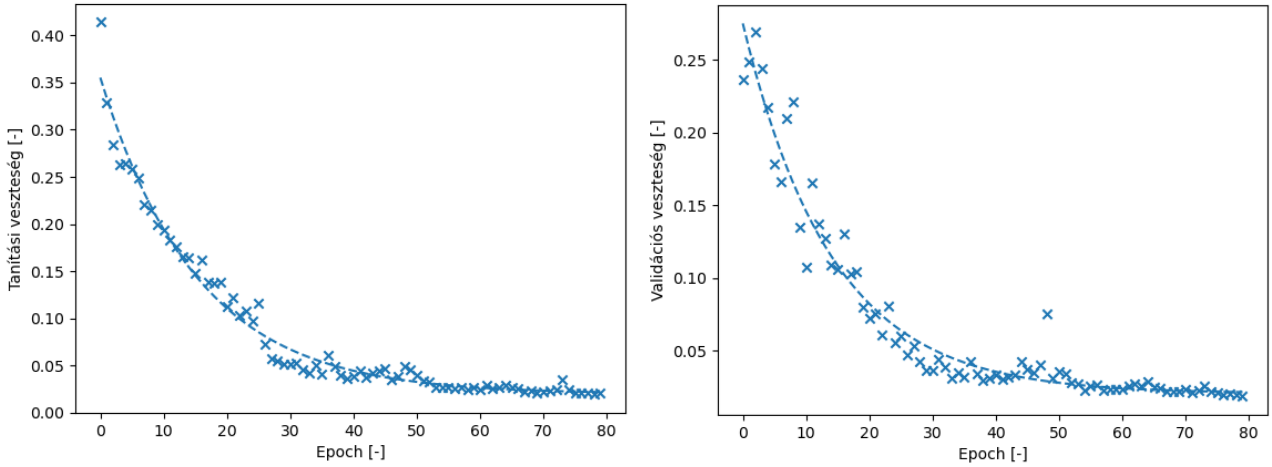


65. ábra: Hiperparaméter optimalizálás erdményei

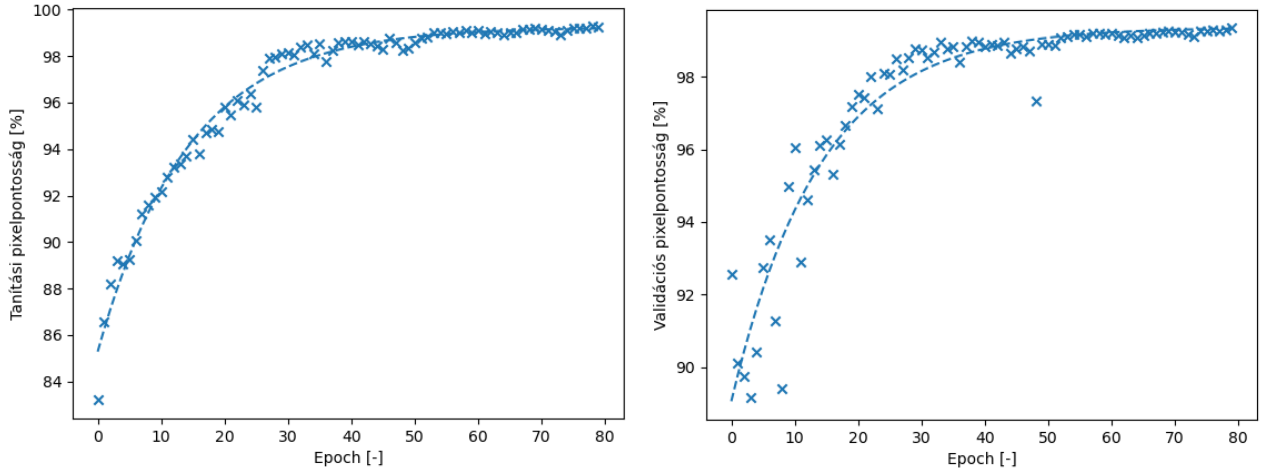
optimalizációs intervallum egy belső pontja bizonyult a legjobbnak, nevezetesen az 5. Tehát nem jó túl sok leskálázást alkalmazni, mert ekkor elveszhetnek a szemantikus szegmentációhoz is fontos információk, de ha nagyon kicsi a háló mélysége az sem jó, mivel ekkor nem nyerhetők ki kellően komplex képjellemzők. A lépésköz optimuma a 21-27 tartományba esett. Ennél kisebb esetben túl hamar csökken túl kicsire a tanulási ráta, ennél nagyobb esetben pedig nem lesz kellő finomságú a súlyparaméter optimalizáció. Az Epoch számok közül egy nagy, 80 körüli volt az optimális. Ez arra is utal, hogy nem kell tartani a túlillesztés problémájától [133] hiszen ez a probléma még a tipikus előfordulási esetében, a hosszú tanítások esetén sem lép fel. A

tanulási ráta nagyon szépen felveszi a nagyságrendileg 0.001-körüli irodalmi értéket [109], tehát ennek a paraméternek a szempontjából nem egyedi ez a probléma. Ez a ráta a választott lépésközönként nagyságrendileg 0,35 körüli értékkel szorzódik meg, ez az ideális csökkenési arány. A regularizációt nagyobb súlyok mellett kell figyelembe venni, ami az említett eredményes túlillesztés-elkerülést hozza magával. A dilatáció értékei pedig nem ábrázoltak, ezek megoszlása 50-50% tehát az egyes rétegek látóterétől független a szegmentáció eredménye, azaz érdemesebb a dilatáció nélküli esettel dolgozni a számítási idő mérséklésének érdekében.

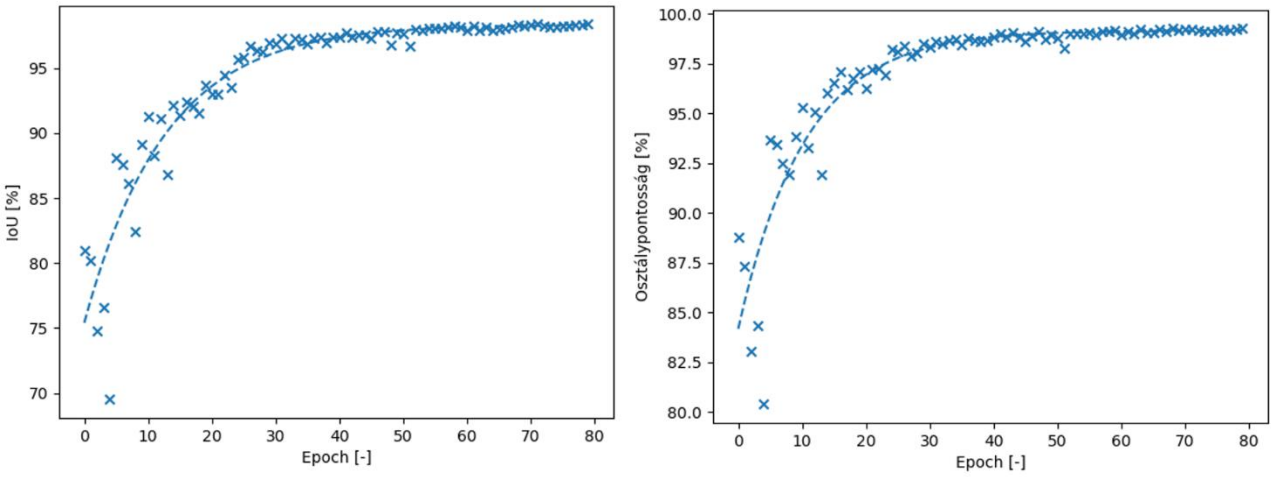
A legjobban teljesítő, 98,37%-os IoU értéket produkáló megoldás az Adam algoritmust használja, 2-es batch méret, 11-es csatornaszám, 5-ös mélység, 26-os lépésköz, 80-as epoch szám, 0,000583-as tanulási ráta, 0.000987-es regularizációs súlyozás és 0,37-es csökkenési arány mellett, dilatáció nélkül. Ennek a tanulási folyamatát a 66. - 68. ábrák jellemzik.



66. ábra: A legjobb futtatás veszteségei

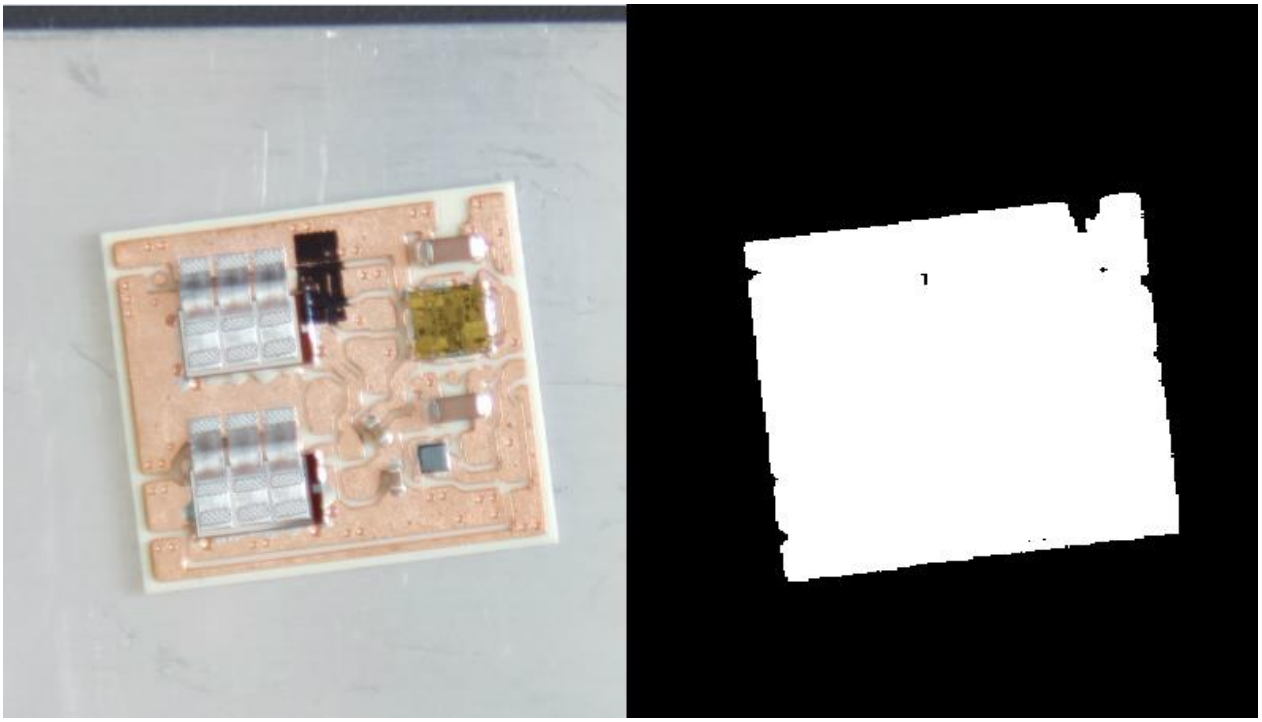


67. ábra: Legjobb futtatás pixelpontosságai



68. ábra: A legjobb futtatás osztálypontosság értékei

Látható, hogy a Fa-struktúrájú Parzen becslőnek köszönhetően jól választott architektúra és módszertan szép egyenletes konvergenciát mutat. A veszteség, és a pixelpontosság értékekre is igaz, hogy a validáció teljes mértékben hasonlít a tanításra viselkedés és legjobb eredmény szempontjából is, így kizárható a túlillesztés. A kapott 98,37%-os IoU érték a még a MOSFET-ek szemantikus szegmentációjának értékéhez képest is kiemelkedően jó.



69. ábra: Szemantikus szegmentáció eredménye

A kulcspontkeresés és az elmozdulás és elforgás kiszámítása a MOSFET-ek esetével megegyező módon történik. Itt is teljesülnek ugyanazok a sikerkritériumok. Az 5.7.2-es fejezethez hasonló, ezzel kapcsolatos vizualizációk az F14 függelékben találhatók.

7 Összefoglalás

Komplex elektronikai eszközt, látórendszert készítettem, amely egy autóiparban elterjedten alkalmazott nemzetközi renoméjú elektronikai komponensek termikus karakterizációját segítő mérés, a T3Ster mérés automatizált működésének lehetővé tételéhez járul hozzá. Több hónapon keresztül én magam is végeztem ezt a mérést hagyományos módszerekkel, így ezalatt az idő alatt igen jól megismerhettem. A robotizáltan végezhetőségének előfeltétele, hogy megfelelő információ álljon rendelkezésre a mérendő objektumok hollétéről.

Ezen probléma megoldásához több hardver és szoftver eszközt is fejlesztettem, valamint meglévő eszközök közül választottam ki a legalkalmasabbakat és dolgoztam velük. Ilyen módon szenzort és ezáltal kamerát, valamint optikát választottam megfelelő optikai számításokat végezve, több elfogadható műszaki megoldás közül a gazdaságilag optimálisat kiválasztva. Ezt követően a kamera felszereléséhez megfelelő mechanikai konstrukciót terveztem, a képének továbbításához pedig alkalmas beágyazott platformot választottam és valósítottam meg rajta egy szervert. A kamera képén két korrekciót is végeztem, amit saját algoritmussal, az OpenCV könyvtár függvényeit felhasználva tettem meg. Ezek a torzítás- és perspektív vetítésből fakadó korrekciók voltak. Következő lépésként szemantikus szegmentációt végeztem mesterséges intelligencia és neurális hálók segítségével, amelyhez alapos irodalomkutatást folytattam. Ezzel kapcsolatban kiemelném az 5.4.2.1 és 5.4.2.3 fejezeteket. A tanításhoz szükséges képeket a valós mérési körülményeknek megfelelően készítettem és kézzel annotáltam. Ilyen speciális probléma megoldásához elvárható a pontos szemantikus szegmentáció. Szoftverem ennek megfelelően nemzetközileg is jónak mondható, 94 %-os IoU értéket produkált a teszt adathalmazon. Végül a szemantikus szegmentációval kiválasztott tartományon kulcspont keresést végeztem, amit a SIFT algoritmussal tettem meg. Ezen kulcspontok minőség szerinti szűrésére kétlépcsős algoritmust valósítottam meg, a lehető legpontosabb becslés, legrobusztusabb működés érdekében. Fejlesztett rendszerem szinergiát mutat a mérések kiértékelésének folyamatával is, ugyanis készített képei felhasználhatók az egyes komponensek termikus elemzésénél az aktuális hővezető paszta mennyiség, rézlapon való elhelyezés és az egyéb környezeti parméterek mérés utáni megtekintésére. Ez a csapatom szimulációs mérnökeinek jelentős segítséget nyújt munkájukban.

Munkám skálázható, felhasználható további tudományos és ipari újításokhoz, projektekhez. A már megalkotott rendszer minimális módosításával a 6. fejezetnek megfelelően lehetőség nyílt a közvetlen kötésű réz kerámia hordozóra épített teljesítményelektronikai modulok mérésére is hasonló technikával. A jövőben néhány további dolgot is szeretnék még kipróbálni és vizsgálni rendszeremre való hatásukat. Hardver oldalról ilyen például a megvilágító rendszer építés és a polárszűrő alkalmazás, amelyekkel esetlegesen javíthatnám fejlesztett eszközöm megbízhatóságát. Szoftver oldalról pedig vizsgálnám neurális hálóim további paramétereinek, függvényeinek és beállításainak változtatását. Ezek további vizsgálatával lehetséges, hogy elérhető a sikeresen teljesített 100 µm-nél és 1°-nál még nagyobb pontosság, amivel akár még kisebb komponensek is mérhetők.

8 Irodalomjegyzék

[1] Grand View Research: Active Electronic Components Market Size, Share & Trends Analysis Report By Product Type (Vacuum Tubes, Semiconductor Devices), By End User (Automotive, Consumer Electronics), By Region, And Segment Forecasts, 2021 – 2028, 2021.

[2] The Market Write Up: Active Electronic Components Market Size, Growth Drivers, Regional Outlook And Forecast 2021-2028, 2021.

[3] Market Watch: Active Electronic Components Market Size, Share, Report 2027, 2021.

[4] Grand View Research: Automotive Electronics Market Size, Share & Trends Analysis Report By Component (Electronic Control Unit, Sensors, Current Carrying Devices), By Application, By Sales Channel, By Region, And Segment Forecasts, 2021 – 2028, 2021.

[5] Global Market Insight: Automotive Electronics Market Size By Vehicle Type (Passenger Cars, Commercial Vehicles), By Application (Advanced Driver Assistance System (ADAS), Body Electronics, Infotainment & Communication [Audio, Display, Navigation, Head-Up Display, Communication], Powertrain [Engine Controllers, Transmission Drivetrain, Exhaust, xEV], Safety Systems), COVID-19 Impact Analysis, Regional Outlook, Growth Potential, Competitive Market Share & Forecast, 2021 – 2027, 2021.

[6] Blue Weave Consulting: Global Automotive Electronics Market, By Vehicle Type (Passenger Cars, Commercial Vehicles); By Application (ADAS (Adaptive Cruise Control), Blind Spot Detection, Parking Assistance, Automated Emergency Braking, Infotainment & communication, Safety System, and Others); By Sales (OEM and Aftermarket), By Electronics Component (Electronic Control Unit, Sensors, Current-Carrying Devices, and Others); By Region (North America, Europe, Asia Pacific, Middle East & Africa, and Latin America); Trend Analysis, Competitive Market Share & Forecast, 2021-2027, 2021.

[7] Data Bridge Market Research: Global Power Transistor Market – Industry Trends and Forecast to 2028, 2021.

[8] Research and Markets: Global Power Transistors Market - Forecasts from 2019 to 2024, 2019.

[9] Report Linker: Global Power Transistors Industry, 2021.

[10] Research and Market: Global Power MOSFET Market by Type, Power Rate and Application: Global Opportunity Analysis and Industry Forecast, 2020-2027, 2020.

[11] H. Wang, K. Ma, and F. Blaabjerg, “Design for reliability of power electronic systems,” in Proc. 38th Annu. Conf. IEEE Ind. Electron. Soc., 2012, pp. 33–44.

[12] P. I. Prodanov, “Research of Reliability of Power Semiconductors in Dependence of Thermal Modes”, Proc. of 21th International Symposium on Electrical Apparatus and Technologies SIELA, 2020.

[13] C. J. M. Lasance, A. Poppe: Thermal Management for LED Applications, New York, USA: Springer Publishing, 2014

[14] “Simcenter T3STER: Characterize the thermal properties of components“, hozzáférés: https://www.plm.automation.siemens.com/global/en/products/simcenter/t3ster.html , [Utolsó elérés: 2021-09-20]

[15] “Mentor Graphics - MicReD Products: T3Ster Master“, hozzáférés: https://t3ster-master.software.informer.com/ , [Utolsó elérés: 2021-09-20]

[16] F. N. Masana, “A new approach to the dynamic thermal modelling of semiconductor packages“, Microelectronics Reliability, vol. 41, pp. 901-912., 2001.

[17] G. Fodor: Elektromágneses terek, pp. 163-166., Budapest, Magyarország, Műegyetemi Kiadó, 2005.

[18] G. Gróf: Hőközlés, pp. 10-18., 54-59., Budapest, Magyarország, 1999.

[19] A. Poppe, “The transient and multichip issues“, 11th THERMINIC Workshop, Thermal measurements and modelling, Belgirate, Italy, 2005.

[20] V. Székely, “Identification of Networks by Deconvolution: Chances and Limits“, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 45., 1998.

[21] JEDEC JESD51-1 Standard, “Integrated Circuits Thermal Measurement Method - Electrical Test Method (Single Semiconductor Device)“, 1995.

[22] JEDEC JESD51-14 Standard, “Transient Dual Interface Test Method for the Measurement of the Thermal Resistance Junction to Case of Semiconductor Devices with Heat Flow Trough a Single Path“, 2010.

[23]Gy. Ábrahám, K. W. Gerőfy, A. Antal, G. Kovács: Műszaki Optika, ch. 1., 7., Budapest, Magyarország, BME MOGI, 2015.
[24]N. Sischka, "Pixel Sizes and Optics", Automate 2017 Conference, Detroit, Michigan, USA, 2017.
[25]Gy. Ábrahám, G. Kovács, A. Antal, Z. Németh, Á. L. Veres: Jármű Optika, ch. 1., Budapest, Magyarország, BME MOGI, 2014.
[26]"Waveshare Electronics: 8-50mm Zoom Lens", hozzáférés: https://www.waveshare.com/wiki/8-50mm_Zoom_Lens_for_Pi , [Utolsó elérés: 2021-09-20]
[27]"SONY: IMX477", hozzáférés: https://www.sony-semicon.co.jp/products/common/pdf/IMX477-AACK_Flyer.pdf , [Utolsó elérés: 2021-09-20]
[28]"Edmund Optics: (8-48mm FL), 6X Manual Zoom Video Lens", hozzáférés: https://www.edmundoptics.eu/p/8-48mm-fl-6x-manual-zoom-video-lens/10695/ , [Utolsó elérés: 2021-09-20]
[29]"Edmund Optics: EO-10012C ½" CMOS Color USB Camera", hozzáférés: https://www.edmundoptics.eu/p/eo-10012ccmos-color-usb-camera/22777/ , [Utolsó elérés: 2021-09-20]
[30]"Hangzhou Ai Ke Electronics: ACM117VP104013CR1", hozzáférés: https://aico-lens.com/product/10-40mm-f1-3-12mp-p-iris-motorized-autofocus-c-mount-4x-zoom-cctv-lens-acm117vp104013cr1/ , [Utolsó elérés: 2021-09-20]
[31]"The Imaging Source: DFK 33UX226", hozzáférés: https://www.theimagingsource.com/products/industrial-cameras/usb-3.0-color/dfk33ux226/ , [Utolsó elérés: 2021-09-20]
[32]"Raspberry Pi: HQ Camera", hozzáférés: https://www.raspberrypi.org/products/raspberry-pi-high-quality-camera/ , [Utolsó elérés: 2021-09-20]
[33]"Raspberry Pi: 4 Model B", hozzáférés: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/ , [Utolsó elérés: 2021-09-20]
[34]P. Drap, J. Lefèvre, "An Exact Formula for Calculating Inverse Radial Lens Distortions", Sensors, MDPI, 2016.
[35]"OpenCV: Camera Calibration, Python Documentation", hozzáférés: https://docs.opencv.org/3.4.15/dc/dbb/tutorial_py_calibration.html , [Utolsó elérés: 2021-10-04]
[36]"OpenCV: Camera Calibration, Github" https://github.com/opencv/opencv/blob/master/modules/calib3d/src/calibration.cpp , [Utolsó elérés: 2021-10-04]
[37]"Georgia Tech Computing: Computer Vision Lecture 3 Camera Calibration, 2020." https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj3/html/kshu6/index.html , [Utolsó elérés: 2021-10-04]
[38]"OpenCV: findChessboardCorners, Documentation", hozzáférés: https://docs.opencv.org/3.4.15/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a [Utolsó elérés: 2021-10-04]
[39]"OpenCV: cornerSubPix, Documentation", hozzáférés: https://docs.opencv.org/3.4.15/dd/d1a/group__imgproc__feature.html#ga354e0d7c86d0d9da75de9b9701a9a87e [Utolsó elérés: 2021-10-04]
[40]R. Cipolla: Computer Vision, Handout 3: Projection, University of Cambridge Engineering Part IIB, 2020
[41]A. Likas, N. Vlassis, J. J. Verbeek, "The global k-means clustering algorithm", Pattern recognition, Elsevier, vol. 36, pp. 451–461., 2003.
[42]X. Zheng, Q. Lei, R. Yao, Y. Gong, Q. Yin, "Image segmentation based onadaptive K-means algorithm", EURASIP Journal on Image and Video Processing, 2018.
[43]M. Barthakur, K. K. Sarma, "Semantic Segmentation using K-means Clustering and Deep Learning in Satellite Image", 2nd International Conference on Innovations in Electronics, Signal Processing and Communication, 2019.
[44]"OpenCV: K-means, Github" https://github.com/opencv/opencv/blob/master/modules/core/src/kmeans.cpp , [Utolsó elérés: 2021-10-06]

[45]F. Pierre, J. F. Aujol, A. Bugeau, V. T. Ta, “Luminance-Hue Specification in the RGB Space“, International Conference on Scale Space and Variational Methods in Computer Vision, pp. 413-424., 2015.
[46]X. Li, Y. Wu, “Image object detection algorithm based on improved Gaussian mixture model“, Journal of Multimedia, pp. 152-158., 2014.
[47]C. Clausen, H. Wechsler, “Color image compression using PCA and backpropagation learning“, Pattern Recognition, vol. 33., pp. 1555-1560., 2000.
[48]“OpenCV: floodFill, Github“ https://github.com/opencv/opencv/blob/master/modules/imgproc/src/floodfill.cpp [Utolsó elérés: 2021-10-07]
[49]“OpenCV: Morphological Operations, Github“ https://github.com/egonSchiele/OpenCV/blob/master/modules/imgproc/src/morph.cpp [Utolsó elérés: 2021-10-07]
[50]S. A. Taghanaki, K. Abhishek, J. P. Cohen, J. Cohen-Adad, G. Hamarneh, “Deep semantic segmentation of natural and medical images: a review“, Springer, 2020.
[51]F. Jiang, A. Grigorev, S. Rho, Z. Tian, Y. Fu, W. Jifara, K. Adil, S. Liu, “Medical image semantic segmentation based on deep learning“, Springer, 2017.
[52]B. Kayalibay, G. Jensen, P. van der Smagt, “CNN-based segmentation of medical imaging data“, Germany, 2017.
[53]W. T. Xiao, L. J. Chang, W. M. Liu, “Semantic segmentation of colorectal polyps with DeepLab and LSTM networks“, International Conference on Consumer Electronics-Taiwan, 2018.
[54]Y. Wu, L. Lin, J. Wang, S. Wu, “Application of semantic segmentation based on convolutional neural network in medical images“, Journal of Biomedical Engineering, 2020.
[55]R. Fan, H. Wang, P. Cai, M. Liu, “Sne-roadseg: Incorporating surface normal information into semantic segmentation for accurate freespace detection“, European Conference on Computer Vision, 2020.
[56]Z. Pan, T. Emaru, A. Ravankar, Y. Kobayashi, “Applying semantic segmentation to autonomous cars in the snowy environment“, 36th Annual Conference of the Robot Society of Japan, Nagoya, 2018.
[57]L. Sun, K. Yang, X. Hu, W. Hu, K. Wang, “Real-Time Fusion Network for RGB-D Semantic Segmentation Incorporating Unexpected Obstacle Detection for Road-Driving Images“, IEEE Robotics and Automation Letters, vol. 5., 2020.
[58]] B. Li, S. Liu, W. Xu, W. Qiu, “Real-time object detection and semantic segmentation for autonomous driving“, Proceedings of the Multispectral Image Processing and Pattern Recognition: Automatic Target Recognition and Navigation, 10608, International Society for Optics and Photonics, 2018.
[59]S. Liu, M. Li, M. Li, Q. Xu, “Research of animals image semantic segmentation based on deep learning“, Concurrency and Computation: Practice and Experience, vol. 32, 2018.
[60]R. Sahu, “Semantic body parts segmentation for quadrupedal animals “, IEEE International Conference on Systems, Man and Cybernetics, 2016.
[61]I. Alonso, A. Cambra, A. Munoz, T. Treibitz, A. C. Murillo, “Coral-Segmentation: Training Dense Labeling Models With Sparse Ground Truth“, Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 2874-2882, 2017.
[62]G. Brazil, X. Yin, X. Liu, “Illuminating pedestrians via simultaneous detection & segmentation“, Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 4950-4959, 2017.
[63]G. Cheng, J. Y. Zheng, “Semantic Segmentation for Pedestrian Detection from Motion in Temporal Domain“, IEEE 25th International Conference on Pattern Recognition, 2020.
[64]X. Tao, D. Zhang, W. Ma, X. Liu, D. Xu, “Automatic Metallic Surface Defect Detection and Recognition with Convolutional Neural Networks“, Applied Sciences, vol. 8, 2018.
[65]V. V. Kniaz, “Conditional GANs for semantic segmentation of multispectral satellite images“, Proceedings of Image and Signal Processing for Remote Sensing XXIV, Berlin, Germany, 2018.
[66]M. Wurm, T. Stark, X. X. Zhu, M. Weigand, H. Taubenböck, “Semantic segmentation of slums in satellite images using transfer learning on fully convolutional neural networks“, ISPRS Journal of Photogrammetry and Remote Sensing, vol. 150, pp. 59-69., 2019.
[67]Y. Guo, Y. Liu, T. Georgiou, M. S. Lew, “A review of semantic segmentation using deep neural networks“, International Journal of Multimedia Information Retrieval, vol. 7, pp. 87–93., 2018.

[68]F. Lateef, Y. Ruichek, “Survey on semantic segmentation using deep learning techniques“, Neurocomputing, vol. 338, pp. 321-348, 2019.
[69]B. Li, Y. Shi, Z. Qi, Z. Chen, “A Survey on Semantic Segmentation“, IEEE International Conference on Data Mining Workshops (ICDMW), 2018.
[70]H. Yu, Z. Yang, L. Tan, Y. Wang, W. Sun, M. Sun, Y. Tang, “Methods and datasets on semantic segmentation: A review “, Neurocomputing, vol. 304, pp. 82-103, 2018.
[71]S. Hao, Y. Zhou, Y. Guo, “A Brief Survey on Semantic Segmentation with Deep Learning“, Neurocomputing, vol. 406, pp. 302-321, 2020.
[72]A. G. Garcia, S. O. Escolano, S.O. Oprea, V. V. Martinez, and J. G. Rodriguez, “A Review on Deep Learning Techniques Applied to Semantic Segmentation“, 2017., arXiv:1704.06857. [Online] hozzáférés: https://arxiv.org/abs/1704.06857 [Utolsó elérés: 2021-09-24]
[73]R. Kestur, S. Farooq, R. Abdal, E. Mehraj, O. S. Narasipura, M. Mudigere, “UFCN: a fully convolutional neural network for road extraction in RGB imagery acquired by remote sensing from an unmanned aerial vehicle“, Journal of Applied Remote Sensing, 2018.
[74]Z. Wu, Y. Gao, L. Li, J. Xue, Y. Li, “Semantic segmentation of high-resolution remote sensing images using fully convolutional network with adaptive threshold“, Connection Science, vol. 31, 2019.
[75]D. L. Torres, R. Q. Feitosa, P. N. Happ, L. E. Cué, L. Rosa, J. M. Junior, J. Martins, P. O. Bressan, W. N. Gonçalves, V. Liesenberg, “Applying Fully Convolutional Architectures for Semantic Segmentation of a Single Tree Species in Urban Environment on High Resolution UAV Optical Imagery“, Sensors, vol. 20, 2020.
[76]W. Bai, O. Oktay, M. Sinclair, H. Suzuki, M. Rajchl, G. Tarroni, B. Glocker, A. King, P. M. Matthews, D. Rueckert, “Semi-supervised Learning for Network-Based Cardiac MR Image Segmentation“, pp. 253-260., 20th International ConferenceQuebec City, QC, Canada, Proceedings, Part II, 2017.
[77]C. V. Dung, L. D. Anh, “Autonomous concrete crack detection using deep fully convolutional neural network“, Automation in Construction, vol. 99., pp. 52-58.
[78]X. Ma, X. Deng, L. Qi, Y. Jiang, H. Li, Y. Wang, X. Xing, “Fully convolutional network for rice seedling and weed image segmentation at the seedling stage in paddy fields“, PloS one, 2019.
[79]L. Tai, H. Ye, Q. Ye, M. Liu, “PCA-aided fully convolutional networks for semantic segmentation of multi-channel fMRI“, International Conference on Advanced Robotics (ICAR), 2017.
[80]G. Chen, X. Zhang, Q. Wang, F. Dai, Y. Gong, K. Zhu, “Symmetrical Dense-Shortcut Deep Fully Convolutional Networks for Semantic Segmentation of Very-High-Resolution Remote Sensing Images“, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2018.
[81]J. Fu, Y. Wang, H. Lu, “Stacked deconvolutional network for semantic segmentation“, IEEE Transactions on Image Processing, 2017.
[82]Y. Wang, J. Liu, Y. Li, J. Yan, H. Lu, “Objectness-aware semantic segmentation“, Proceedings of the ACM on Multimedia Conference, ACM, pp. 307–311., 2016.
[83]H. Noh, S. Hong, B. Han, “Learning deconvolution network for semantic segmentation“, Proceedings of the IEEE International Conference on Computer Vision, pp. 1520–1528., 2015.
[84]S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, Y. Bengio, “The one hundred layers tiramisu: fully convolutional densenets for semantic segmentation“, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, pp. 1175–1183., 2017.
[85]V. Badrinarayanan, A. Kendall, R. Cipolla, “Segnet: a deep convolutional encoder-decoder architecture for image segmentation“, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39., no. 12., 2017.
[86] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, “Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition“, pp. 3431–3440., 2015.
[87]“Papers with Code: Semantic Segmentation on PASCAL VOC 2012 test”, hozzáférés: https://paperswithcode.com/sota/semantic-segmentation-on-pascal-voc-2012 , [Utolsó elérés: 2021-09-25]
[88]“Papers with Code: CamVid (Cambridge-driving Labeled Video Database)”, hozzáférés: https://paperswithcode.com/sota/semantic-segmentation-on-pascal-voc-2012https://paperswithcode.com/dataset/camvid , [Utolsó elérés: 2021-09-25]

[89]"Papers with Code: Semantic Segmentation on PASCAL VOC 2011", hozzáférés: https://paperswithcode.com/sota/semantic-segmentation-on-pascal-voc-2011 , [Utolsó elérés: 2021-09-25]
[90]Y. Li, S. Liu, C. Li, Y. Zheng, C. Wei, B. Liu, Y. Yang, "Automated defect detection of insulated gate bipolar transistor based on computed laminography imaging", Microelectronics Reliability, vol. 115., 2020.
[91]W. Shi, Z. Lu, W. Wu, H. Liu, "Single-shot detector with enriched semantics for PCB tiny defect detection", The Journal of Engineering, 2020.
[92]Z. Yang, R. Dong, H. Xu, J. Gu, "Instance Segmentation Method Based on Improved Mask R-CNN for the Stacked Electronic Components", Electronics, 2020.
[93]J. Lian, L. Wang, T. Liu, X. Ding, Z. Yu, "Automatic visual inspection for printed circuit board via novel Mask R-CNN in smart city applications", Sustainable Energy Technologies and Assessments, vol 44., 2021.
[94]D. Li, C. Li, C. Chen, Z. Zhao, "Semantic Segmentation of a Printed Circuit Board for Component Recognition Based on Depth Images", Sensors, vol. 20., 2020.
[95]"Pytorch: Documentation", hozzáférés: https://pytorch.org/docs/stable/index.html [Utolsó elérés: 2021-10-08]
[96]"Torchvision: Documentation", hozzáférés: https://pytorch.org/vision/stable/index.html [Utolsó elérés: 2021-10-08]
[97]"Pillow: Documentation", hozzáférés: https://pillow.readthedocs.io/en/stable/ [Utolsó elérés: 2021-10-08]
[98]"Numpy: Documentation", hozzáférés: https://numpy.org/doc/ [Utolsó elérés: 2021-10-08]
[99]"Matplotlib: Documentation", hozzáférés: https://matplotlib.org/ [Utolsó elérés: 2021-10-08]
[100] "OS: Documentation", hozzáférés: https://docs.python.org/3/library/os.html [Utolsó elérés: 2021-10-08]
[101] "Glob: Documentation", hozzáférés: https://docs.python.org/3/library/glob.html [Utolsó elérés: 2021-10-08]
[102] J. Thanaki: Python natural language processing, pp. 397-398., 2017.
[103] M. Szemenyei: Számítógépes Látórendszerek, pp. 110-111. 2021.
[104] M. Szemenyei: Számítógépes Látórendszerek, pp. 129-130. 2021.
[105] H. Yang, J. Liu, H. Sun, H. Zhang: "PACL: piecewise arc cotangent decay learning rate for deep neural network training", IEEE Access, vol. 8., 2020.
[106] "NVIDIA: CUDA Toolkit Release Notes", hozzáférés: https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html [Utolsó elérés: 2021-10-09]
[107] L. Feng, S. Shu, Z. Lin, F. Lv, L. Li, B. An, "Can cross entropy loss be robust to label noise? ", Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, 2020.
[108] Z. Zhang, M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels", 32nd Conference on Neural Information Processing Systems, Montréal, Canada, 2018.
[109] D. P. Kingma, J. Ba, "Adam: A method for stochastic optimization", ICLR, 2015.
[110] I. K. M. Jais, A. R. Ismail, S. Q. Nisa, "Adam optimization algorithm for wide and deep neural network", Knowledge Engineering and Data Science, 2019.
[111] "Alpha: Make Sense", hozzáférés: https://www.makesense.ai/ , [Utolsó elérés: 2021-10-09]
[112] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", International Journal of Computer Vision, 2004.
[113] "OpenCV: SIFT, Documentation", https://docs.opencv.org/3.4.15/d7/d60/classcv_1_1SIFT.html , [Utolsó elérés: 2021-10-09]
[114] L. Assirati, N. R. Silva, L. Berton, A. A. Lopes, O. M. Bruno, "Performing edge detection by Difference of Gaussians using q-Gaussian kernels", Journal of Physics Conference Series, 2013.
[115] "OpenCV: BFMatcher, Documentation", https://docs.opencv.org/4.5.3/d3/da1/classcv_1_1BFMatcher.html , [Utolsó elérés: 2021-10-09]
[116] A. Yahyaee, A. S. Bahman, and F. Blaabjerg, "A Modification of Offset Strip Fin Heatsink with High-Performance Cooling for IGBT Modules," Applied Sciences, vol. 10, no. 3, 2020.
[117] M. Feurer and F. Hutter, "Hyperparameter optimization," in Automated machine learning, Springer, Cham, pp. 3–33, 2019.

[118] L. Bottou, “Stochastic Gradient Descent Tricks,” in Neural Networks: Tricks of the Trade: Second Edition, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 421–436, 2012.
[119] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl, “Algorithms for Hyper-Parameter Optimization,” Proc. Neural Information and Processing Systems, 2011.
[120] P. I. Frazier, “A tutorial on Bayesian optimization,” arXiv preprint arXiv:1807. 02811, 2018.
[121] Y. Bengio, “Gradient-based optimization of hyperparameters,” Neural computation, vol. 12, no. 8, pp. 1889–1900, 2000.
[122] D. Simon, Evolutionary optimization algorithms. John Wiley & Sons, 2013.
[123] M. Massaoudi, H. Abu-Rub, S. S. Refaat, M. Trabelsi, I. Chihi and F. S. Oueslati, ”Enhanced Deep Belief Network Based on Ensemble Learning and Tree-Structured of Parzen Estimators: An Optimal Photovoltaic Power Forecasting Method, ” in IEEE Access, vol. 9, pp. 150330-150344, 2021.
[124] J. Liang et al., “Intelligent fault diagnosis of rotating machinery using lightweight network with modified tree-structured parzen estimators,” IET Collaborative Intelligent Manufacturing, 2022.
[125] G. Rong et al., “Comparison of Tree-Structured Parzen Estimator Optimization in Three Typical Neural Network Models for Landslide Susceptibility Assessment,” Remote Sensing, vol. 13, no. 22, 2021.
[126] H.-P. Nguyen, J. Liu, and E. Zio, “A long-term prediction approach based on long short-term memory neural networks with automatic parameter optimization by Tree-structured Parzen Estimator and applied to time-series data of NPP steam generators,” Applied Soft Computing, vol. 89, p. 106116, 2020.
[127] S. M. Hernandez and E. Bulut, “WiFi Sensing on the Edge: Signal Processing Techniques and Challenges for Real-World Systems,” IEEE Communications Surveys & Tutorials, pp. 1–1, 2022.
[128] M. Zhao and J. Li, "Tuning the hyper-parameters of CMA-ES with tree-structured Parzen estimators," 2018 Tenth International Conference on Advanced Computational Intelligence (ICACI), pp. 613-618, 2018.
[129] S. Yoo, M. A. Haider, and F. Khalvati, “Estimating optimal depth of vgg net with tree-structured parzen estimators,” Journal of Computational Vision and Imaging Systems, vol. 3, no. 1, 2017.
[130] Z. Wang, L. Wang, Q. Liang, W. Luo, Q. Gong and S. Li, "Research on Automated Reinforcement Learning: based on Tree-structured Parzen Estimators and Median Pruning," 2021 33rd Chinese Control and Decision Conference (CCDC), pp. 5461-5465, 2021.
[131] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A Next-generation Hyperparameter Optimization Framework,” in Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019.
[132] B. Resch, “Mixtures of gaussians,” A Tutorial for the Course Computational Intelligence. Signal Processing and Speech Communication Laboratory, vol. 24, 2010.
[133] X. Ying, “An overview of overfitting and its solutions,” in Journal of physics: Conference series, vol. 1168, p. 022022, 2019.

Függelék

F1 Félvezető eszközök viselkedésének hőmérsékletfüggése [13]

Félvezető eszközök chip hőmérsékletének érzékelésére a leggyakrabban használt paraméter jellemzően a diódák nyitóirányú feszültsége. A TSP érték idealizált esetben alapvetően levezethető a félvezetők alapösszefüggéseiből. Tanulságos lehet erre kitérni szilíciumdiódák esetén, amelyek viselkedése jó közelítéssel ideális egy igen széles hőmérséklet és áramtartományban. Ez a levezetés egy-egy más anyagra is jó lehet így például szilícium-karbidra is. Szükség esetén a T3Ster Master szoftver segítségével akár egy polinommal is közelíthető a TSP érték amennyiben jelentős nemlinearitás tapasztalható.

Az ideális dióda karakterisztikája:

$$I = I_0(e^{\frac{U}{m \cdot U_T}} - 1) \tag{51}$$

ahol:

I	a dióda árama,
I_0	a p-n átmenet telítési állandója,
U_T	a termikus feszültség,
U	a dióda feszültsége,
m	az idealitási tényező.

Itt:

$$U_T = \frac{k \cdot T}{q} \tag{52}$$

ahol:

k	a Boltzmann-állandó,
T	a Kelvinben mért hőmérséklet,
q	az elektron töltésének abszolút értéke.

Valamint nyitóirányban felírható a következő alak:

$$U = m \cdot U_T \cdot \ln\left(\frac{I}{I_0}\right) + I \cdot R_s \tag{53}$$

ahol:

R_s a soros parazita ellenállás.

Továbbá valójában a termikus feszültségen túl a p-n átmenet telítési állandója is függ a hőmérséklettől:

$$I_0 \sim n_i^2 \sim T^3 e^{\left(-\frac{W_g}{k \cdot T}\right)} \tag{54}$$

ahol:

n_i az intrinsic töltéshordozó koncentráció,

W_g a tiltott sáv szélessége.

Az idealitási tényező egy eszközspecifikus mennyiség, normál üzemi tartományban 1 az értéke. Nagyon kicsiny áramok tartományában 2 a rekombináció jelensége miatt, valamint nagy áramok tartományában szintén 2 az ambipoláris diffúzió jelenségének következtében. Egy adott tartományra az idealitási faktor (53) két kiválasztott pontjából számítható. Többek között ezen alapösszefüggések és gondolatok felhasználásával kapható meg a feszültség hőmérsékletfüggése.

$$\frac{dU}{dT} = \frac{U - 3 \cdot m \cdot U_T - \frac{W_g}{q}}{T} \tag{55}$$

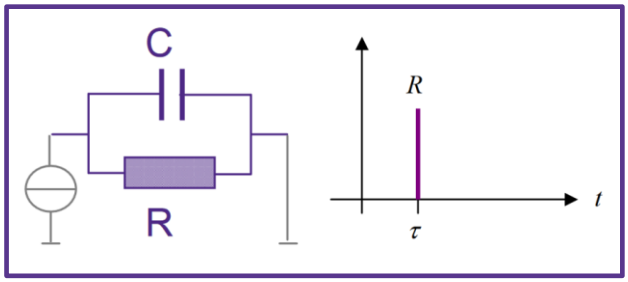
Egyébiránt pedig szélesebb áramtartományokra is ez alapján becsülhető meg a hőmérsékletfüggés, annak figyelembevételével, hogy nagy áramok esetén a soros parazita ellenállás értéke változhat. Így érdemes numerikus számításokat végezni és azokat összevetni az egyes tapasztalati értékekkel.

F2 Átmeneti- és struktúra függvény

Ahhoz, hogy érthető legyen, hogy miképp lehet kikövetkeztetni a chip és a teljes komponens struktúráját az átmeneti függvényből, részletesen ismerni kell azt. Erre szolgál a következő fejezet.

F2.1 Átmeneti függvény [19]

A 70. ábra szerint megfigyelhető az átmeneti függvény egyetlen RC-tag esetén. A jelölések megegyeznek a 2.2.3.2 fejezetben tárgyaltakkal.



70. ábra: Egyetlen RC-tag [19]

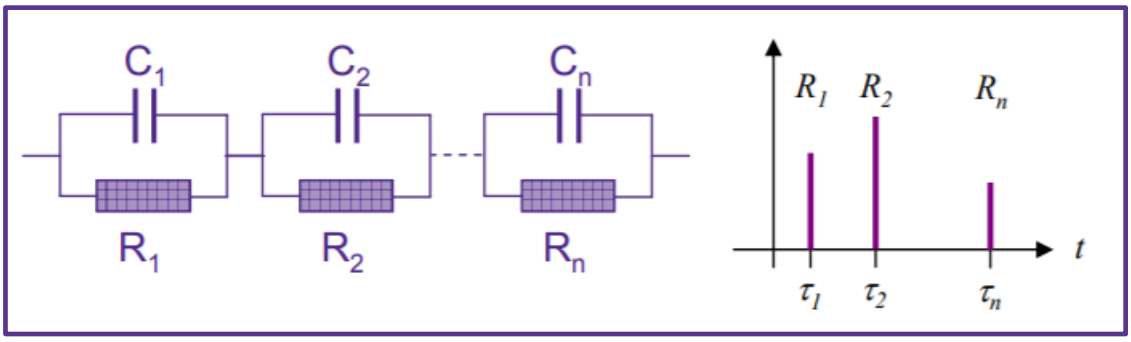
Az RC-tag jellegzetes mennyiségei az amplitúdó, amely az R-től függ és az időállandó, amely az alábbi egyenlet alapján számolandó:

$$\tau = R \cdot C \tag{56}$$

Ebből:

$$a(t) = R \cdot \left(1 - e^{-\frac{t}{\tau}}\right) \tag{57}$$

Sok ilyen párhuzamos RC-tag soros összekapcsolása is hasonló átmeneti függvényt eredményez. Minden egyes taghoz saját időállandó és ellenállás tartozik. Erre mutat rá a 71. ábra és az (58) egyenlet.



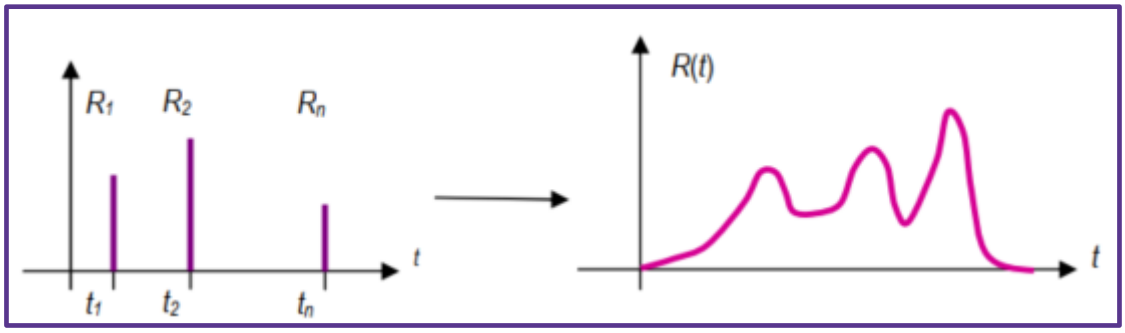
71. ábra: Sorosan kapcsolt RC-tagok válasza [19]

$$\tau_i = R_i \cdot C_i \tag{58}$$

Ezek alapján az átmeneti függvény:

$$a(t) = \sum_{i=1}^{n} R_i \cdot \left(1 - e^{\frac{-t}{\tau_i}}\right) \tag{59}$$

A rendszert egyértelműen definiáló, karakterisztikát jellemző értékek az ellenállás és időállandó értékek. Ezek ismeretében leírható a rendszer. Folytonos időben n tart a végtelenhez, melynek következtében integrálást kell alkalmazni összeadás helyett.



72. ábra: Diszkrét és folytonos rendszer [19]

$$a(t) = \int_0^{\infty} R(\tau) \cdot \left(1 - e^{\frac{-t}{\tau_i}}\right) d\tau \tag{60}$$

Az itt említettek lényegében az időállandó spektrumról szolgálnak információval, melynek jellegzetes tartományai ismertek [19]:

- 100 µs - 10 ms: félvezető chip
- 10 ms - 50 ms: chip alatti struktúra
- 50 ms - 1 s: komponens további struktúrája
- 1 s - 10 s: komponens háza
- 10 s - 10000 s: hűtő funkciót ellátó alkatrészek

Az időállandók széles skálája megnehezíti az adatok mintavételezését. Ahhoz, hogy a legkisebb időállandókkal rendelkező RC-tagokról is megfelelő információt lehessen szerezni, ezek reciprokánál nagyobb mintavételi frekvenciát kell alkalmazni. Tehát mintegy 1 µs-mal a gerjesztés után célszerű megkezdeni a mérést. Valamint logaritmikus időléptéket kell alkalmazni, hogy egyszerűbben megfigyelhető legyen a válaszfüggvény. Ezen válaszfüggvény a terminológia szerint termikus impedancia görbeként említendő. A logaritmikus áttérés eredményeként:

$$\frac{d}{dz} a(z) = \int_0^{\infty} R(\xi) \cdot e^{z-\xi-e^{z-\xi}} d\xi \tag{61}$$

ahol:

z az idő természetes alapú logaritmusa.

Így (15) analógiájára:

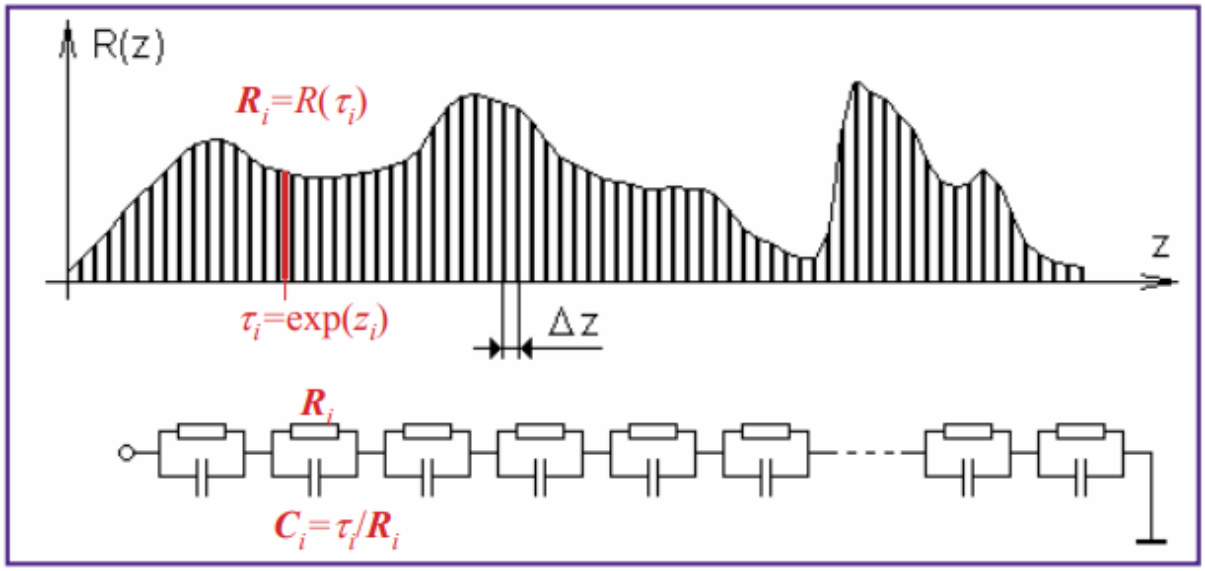
$$R(z) = \left(\frac{d}{dz} a(z)\right) \otimes^{-1} e^{z-e^z} \tag{62}$$

Gyakorlatban a mérési zaj csökkentéséhez több egymás után elvégzett mérés átlagolása használatos. A T3Ster Master szoftver a dekonvolúciót Bayes-iterációval valósítja meg. Ezen algoritmus iterációs

számának beállítása fontos, mérlegelést igénylő feladat, hiszen minél magasabb az iterációs szám, annál több jellegzetességet lehet megfigyelni az adott struktúrában, de ezzel a mérési zaj is egyre több artefaktumot hoz létre. Ezen módszer eredményeként kapott adatok elemzéséhez használatos a struktúrafüggvény.

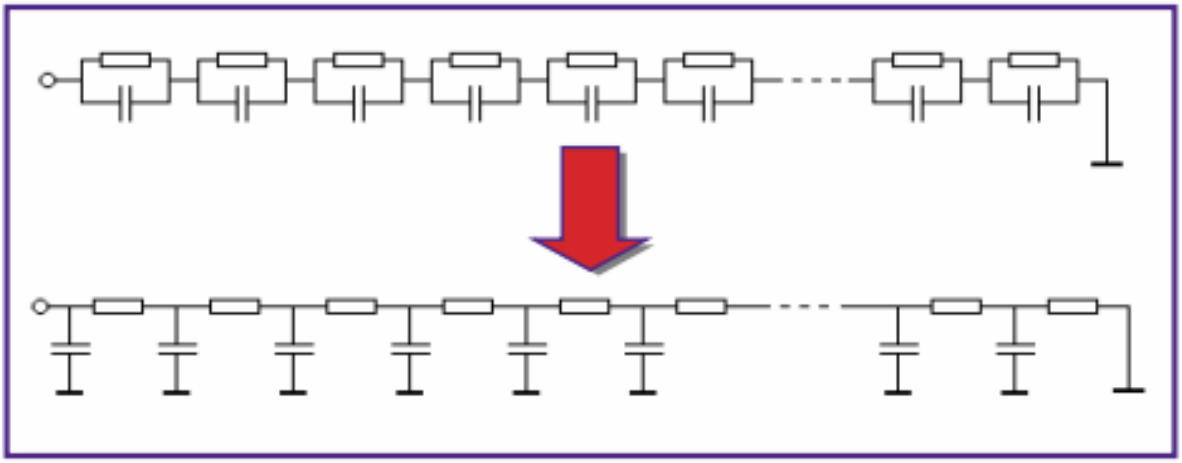
F2.2 Struktúra függvény [19]

Az F2.1-ben említett folytonos időállandó spektrum végtelen számú időállandót reprezentál. Azonban ezeket az elemzéshez diszkretizálni kell, pont úgy, területszámítással, mintha egy numerikus integrálást kellene végezni inverz módon. Ehhez nyújt segítséget a 73. ábra.



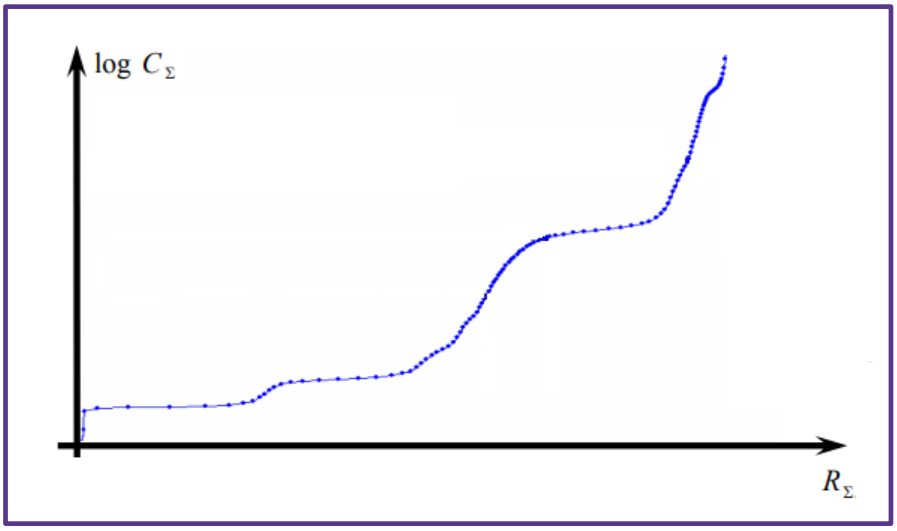
73. ábra: Diszkretizált időállandó spektrum [19]

Itt megfigyelhető a Foster-impedancia modell, ami lényegében a már korábban tárgyalt 71. ábra szerinti kapcsolást használ. Ez a modell inkább elméleti szempontból lényeges. NID [20] (Network Identification by Deconvolution) alkalmazásához át kell térni a praktikusabb Cauer-modellre. Itt a villamos reprezentációval leképezett termikus rendszer földpotenciálú része tulajdonképpen a környezet, ahova a hő elvezetésre kerül.



74. ábra: Áttérés Cauer-modellre [19]

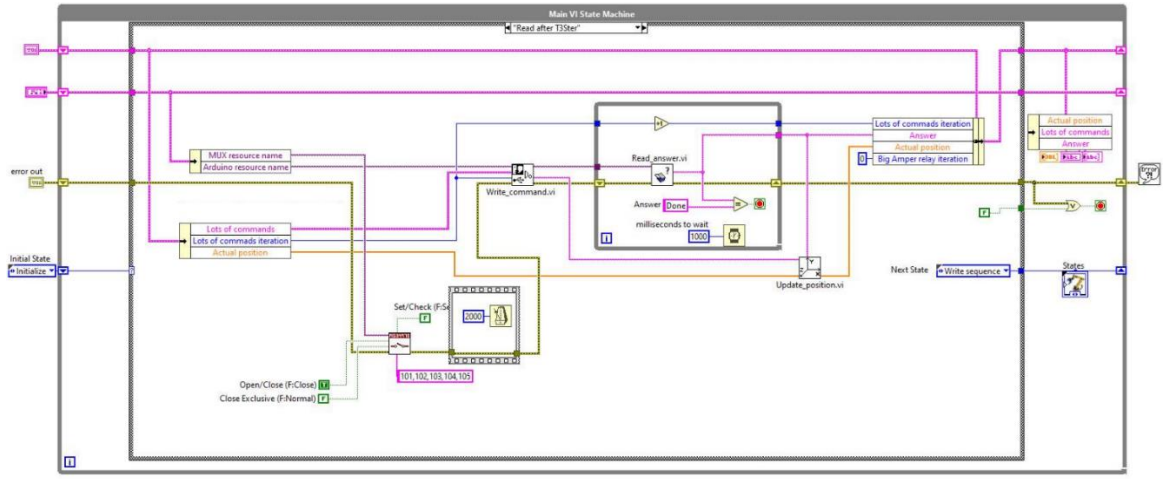
Így ez alapján készíthető egy egyértelmű hőellenállás-hőkapacitás reprezentáció. Tulajdonképpen ez a kumulatív struktúrafüggvény. A kapacitást logaritmikus léptékben szokás ábrázolni az ellenállás függvényében a jobb megfigyelhetőség érdekében.



75. ábra: Kummulatív struktúrafüggvény [19]

Megjegyzendő, hogy ennek deriválásával kapható a differenciális struktúrafüggvény, amely szintén fontos a struktúra vizsgálatának szempontjából, a meredekségváltozásokat jobban kiemeli. Mindkét struktúrafüggvény esetében a vízszintes tengely bal oldalán reprezentált hőforrástól, a jobb oldalon reprezentált környezet felé áramlik a hő. A kummulatív struktúrafüggvény esetében a vízszinteshez közeli részek nagysága az azon részhez tartozó kevéssé jó hővezető anyag mennyiségéről szolgál információval. A meredekebb részek a jobb hővezető anyagokhoz tartoznak, mint például a szilícium chip, vagy mint a rézből készült villamosan vezető részek. Ennek megfelelően a differenciális struktúrafüggvény esetében az egyes lokális maximumok mutatják meg a jó-, a lokális minimumok pedig a rossz hővezető anyagokat.

F3 Illusztráció a LabVIEW környezetről



76. ábra: Illusztráció a robot LabVIEW környezetéről

F4 A fókusztávolság, a legkisebb méretet tartalmazó pixelszám és az elérhető objektívtől való távolság tartománya

5. táblázat: Különböző tárgytávolságokhoz és képméretekhez tartozó fókusztávolságok tartománya milliméterben számítva ($f(a,s)$)

s [mm] / a [-]	- 100	- 90	- 80	- 70	- 60	- 50	- 40	- 30	- 20
5	11,11	10,00	8,89	7,78	6,67	5,56	4,44	3,33	2,22
6	13,04	11,74	10,43	9,13	7,83	6,52	5,22	3,91	2,61
7	14,89	13,40	11,91	10,43	8,94	7,45	5,96	4,47	2,98
8	16,67	15,00	13,33	11,67	10,00	8,33	6,67	5,00	3,33
9	18,37	16,53	14,69	12,86	11,02	9,18	7,35	5,51	3,67
10	20,00	18,00	16,00	14,00	12,00	10,00	8,00	6,00	4,00
11	21,57	19,41	17,25	15,1	12,94	10,78	8,63	6,47	4,31
12	23,08	20,77	18,46	16,15	13,85	11,54	9,23	6,92	4,62
13	24,53	22,08	19,62	17,17	14,72	12,26	9,81	7,36	4,91
14	25,93	23,33	20,74	18,15	15,56	12,96	10,37	7,78	5,19
15	27,27	24,55	21,82	19,09	16,36	13,64	10,91	8,18	5,45

6. táblázat: Különböző objektívtől mért távolságokhoz és fókusztávolságokhoz tartozó legkisebb méretet tartalmazó pixelszám ($a(s_o, f)$)

s_o [mm] / f [mm]	- 100	- 90	- 80	- 70	- 60	- 50	- 40	- 30	- 20
8	2,14	2,28	2,43	2,60	2,80	3,03	3,31	3,64	4,05
11,5	3,21	3,42	3,66	3,94	4,26	4,63	5,08	5,63	6,31
15	4,38	4,68	5,02	5,42	5,89	6,44	7,11	7,94	8,99
18,5	5,65	6,06	6,53	7,08	7,73	8,51	9,47	10,67	12,22
22	7,05	7,59	8,21	8,94	9,82	10,89	12,22	13,92	16,17
25,5	8,60	9,29	10,09	11,06	12,22	13,66	15,49	17,88	21,14
29	10,31	11,19	12,22	13,47	15,01	16,94	19,43	22,79	27,56
32,5	12,23	13,33	14,65	16,26	18,28	20,86	24,29	29,06	36,18
36	14,38	15,76	17,44	19,52	22,16	25,64	30,40	37,33	48,37
39,5	16,81	18,54	20,68	23,37	26,87	31,59	38,34	48,75	66,92
43	19,58	21,76	24,48	27,99	32,67	39,22	49,07	65,52	98,57
46,5	22,77	25,52	29,02	33,64	40,01	49,35	64,38	92,58	164,74
50	26,48	29,97	34,52	40,71	49,59	63,43	87,99	143,57	389,83

7. táblázat: Elérhető objektívtől való távolságok különböző fókusztávolságokkal és legkisebb méretet tartalmazó pixelszámokkal ($s_0(a, f)$)

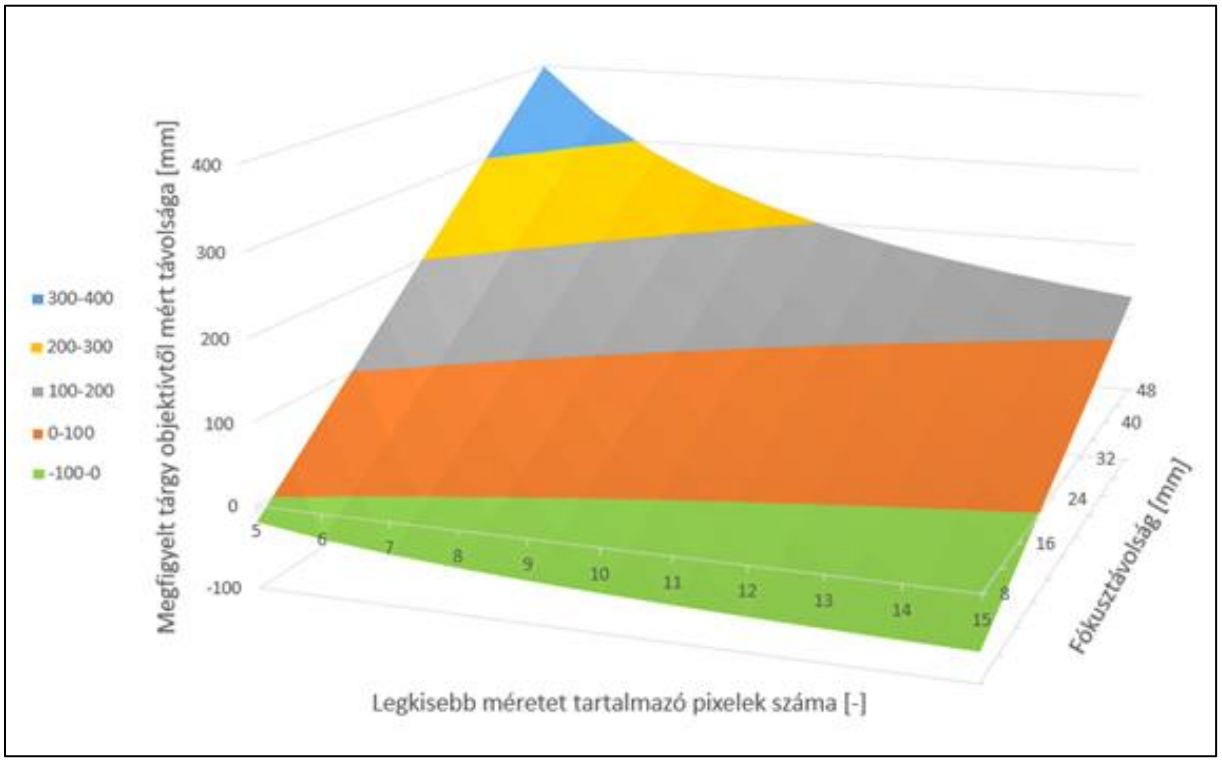
a [-] / f [mm]	5	6	7	8	9	10	11	12	13	14	15
8	2,90	-9,22	-17,88	-24,38	-29,43	-33,47	-36,77	2,90	-9,22	-17,88	-24,38
11,5	41,72	24,29	11,85	2,51	-4,75	-10,56	-15,31	41,72	24,29	11,85	2,51
15	80,53	57,81	41,57	29,40	19,93	12,35	6,15	80,53	57,81	41,57	29,40
18,5	119,35	91,32	71,30	56,28	44,60	35,26	27,62	119,35	91,32	71,30	56,28
22	158,17	124,84	101,03	83,17	69,28	58,17	49,08	158,17	124,84	101,03	83,17
25,5	196,99	158,35	130,75	110,06	93,96	81,08	70,54	196,99	158,35	130,75	110,06
29	235,81	191,87	160,48	136,94	118,63	103,99	92,00	235,81	191,87	160,48	136,94
32,5	274,62	225,38	190,21	163,83	143,31	126,90	113,47	274,62	225,38	190,21	163,83
36	313,44	258,90	219,94	190,72	167,99	149,81	134,93	313,44	258,90	219,94	190,72
39,5	352,26	292,41	249,66	217,60	192,66	172,72	156,39	352,26	292,41	249,66	217,60
43	391,08	325,93	279,39	244,49	217,34	195,62	177,86	391,08	325,93	279,39	244,49
46,5	429,90	359,44	309,12	271,37	242,02	218,53	199,32	429,90	359,44	309,12	271,37
50	468,72	392,96	338,85	298,26	266,70	241,44	220,78	468,72	392,96	338,85	298,26

F5 Kamerarendszer további alternatívái

Ebben a függelékben a 4.1.2 szerinti kamerarendszer további alternatívái kerülnek bemutatásra. Itt a 4.1 fejezetei szerinti részletes levezetések alapján közlöm az eredményeket.

F5.1 Edmund Optics 8-48 mm FL 6X Manual Zoom Video Lens [28] és 10012C ½" CMOS Color USB Camera [29]

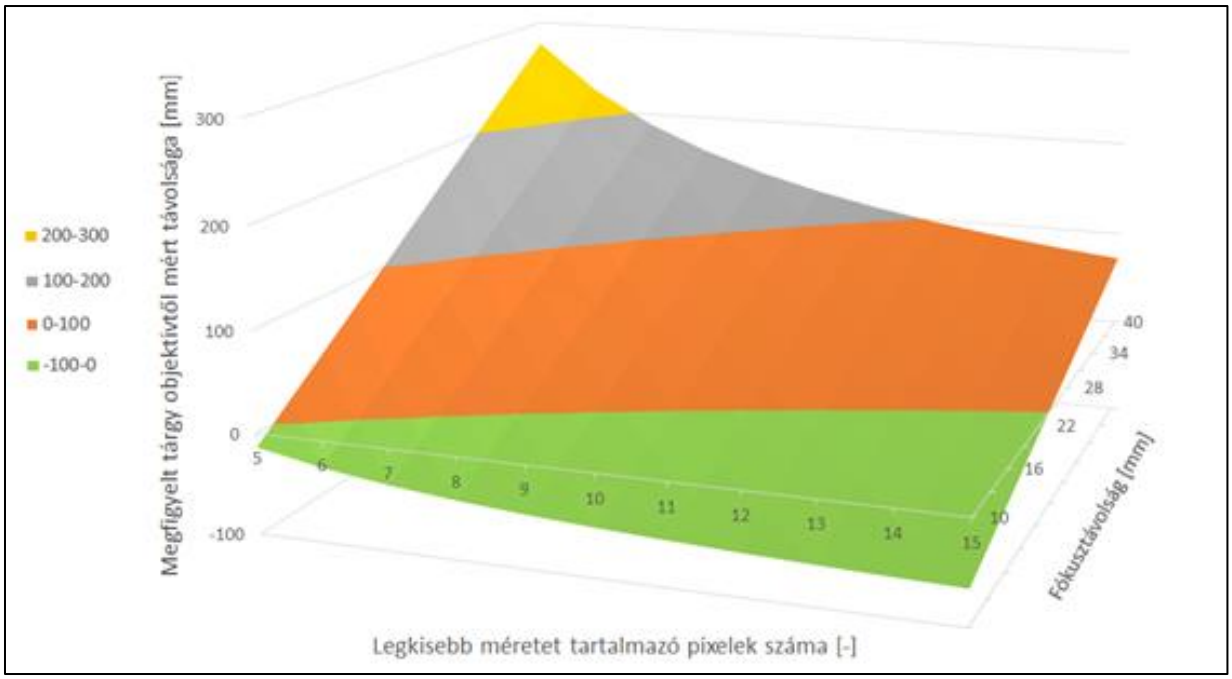
Az egyik lehetséges alternatíva az Edmund Optics 8-48 mm FL 6X Manual Zoom Video Lens optikájának és a 10012C ½" CMOS Color USB Camera kamerájának kombinációja. Itt is megegyezik a szenzorformátum. A választás műszaki megfelelőségéről a 77. ábra tanúskodik. Az objektív teljes hossza itt 103 mm, a fókusztávolság 8 és 48 mm között állítható a szenzor diagonális pixelmérete pedig 2,36 µm.



77. ábra: Edmund Optics alternatívái

F5.2 Hangzhou Ai Ke Electronics ACM117VP104013CR1 [30] és The Imaging Source DFK 33UX226 [31]

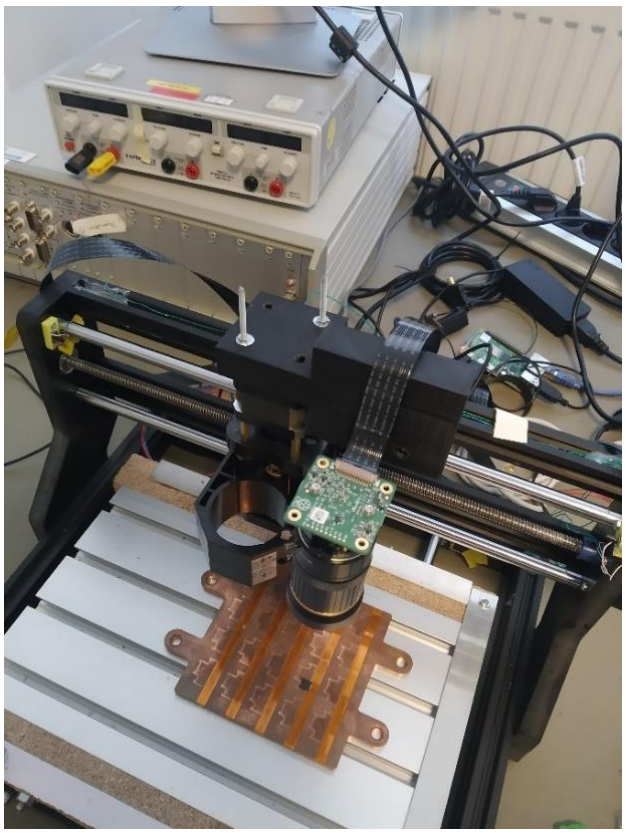
A másik további lehetséges alternatíva a Hangzhou Ai Ke Electronics ACM117VP104013CR1 optikája és a The Imaging Source DFK 33UX226 kamerája. Ezek is jó megoldást nyújtanak a 78. ábra alapján.



78. ábra: Harmadik kamera-optika alternatíva

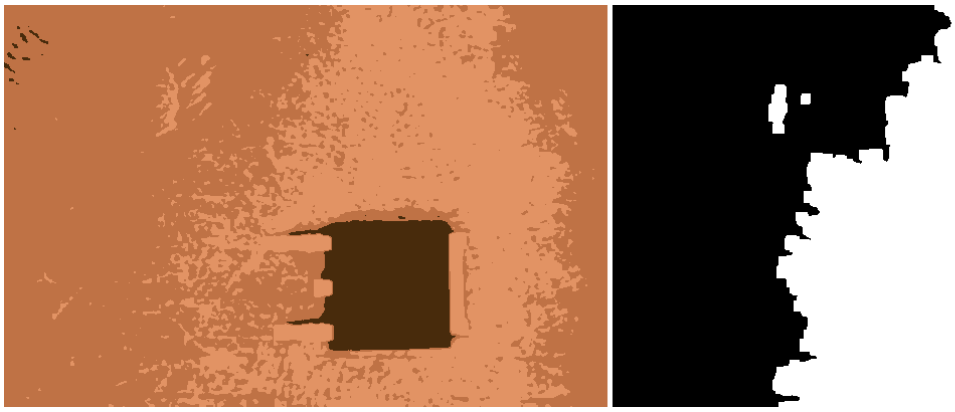
Ebben az esetben az objektív teljes hossza 109,83 mm a fókusztávolság 10 és 40 mm között állítható, a diagonális pixelméret pedig 2,62 µm.

F6 3D nyomtatás eredménye

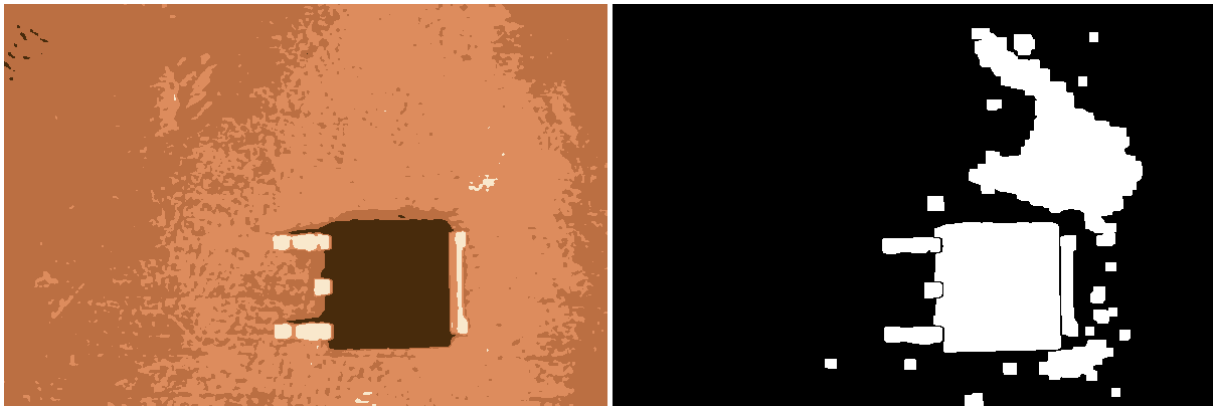


79. ábra: Kinyomtatott kameratartó alkalmazás közben

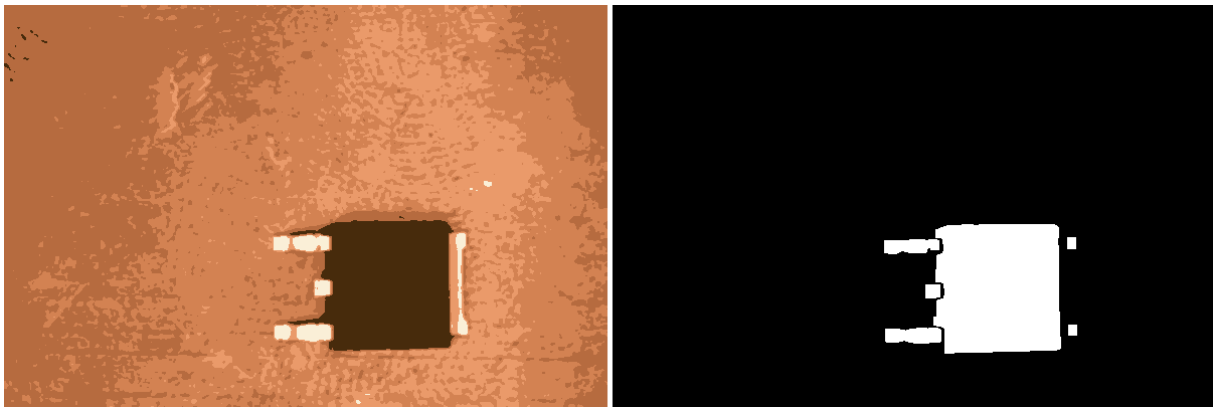
F7 A K-means algoritmus és az ezzel megvalósított szemantikus szegmentáció eredményei különböző klaszterszámok esetén



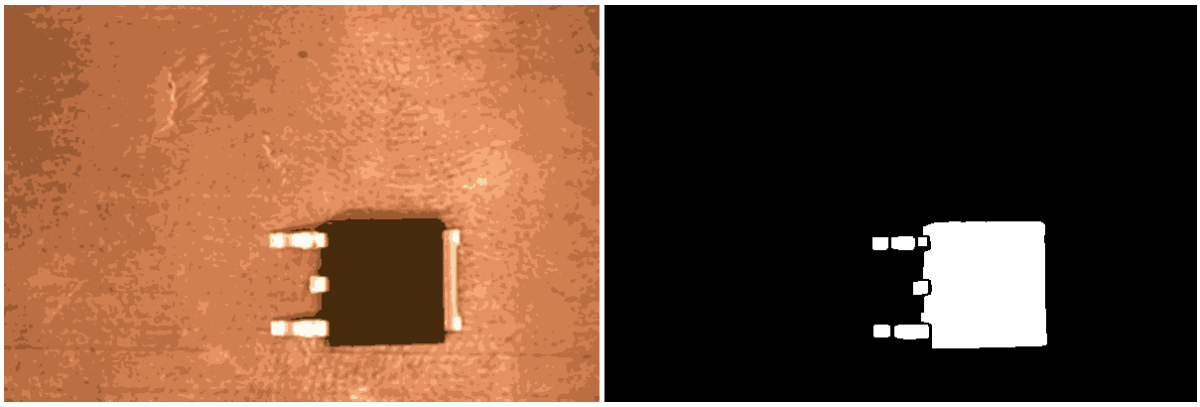
80. ábra: Klaszterezés és szegmentáció K = 3 esetén



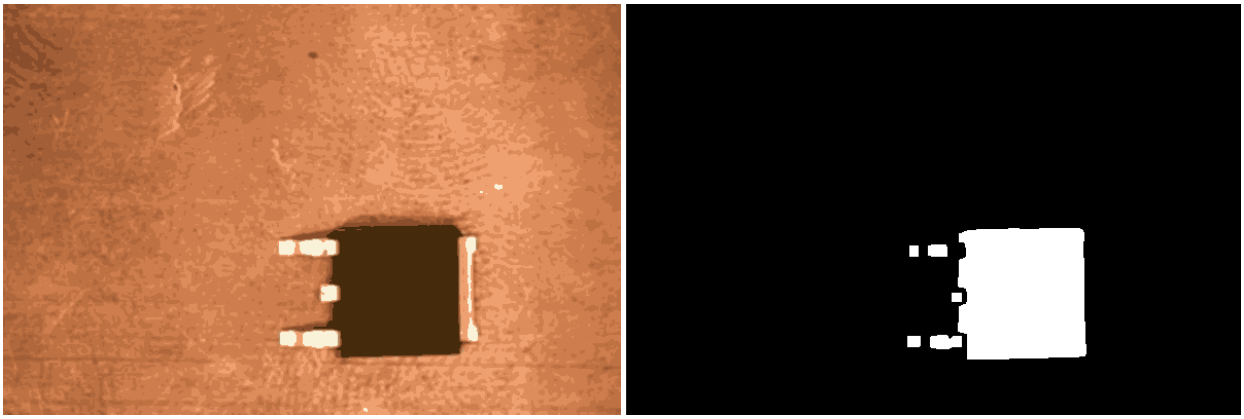
81. ábra: Klaszterezés és szegmentáció K = 4 esetén



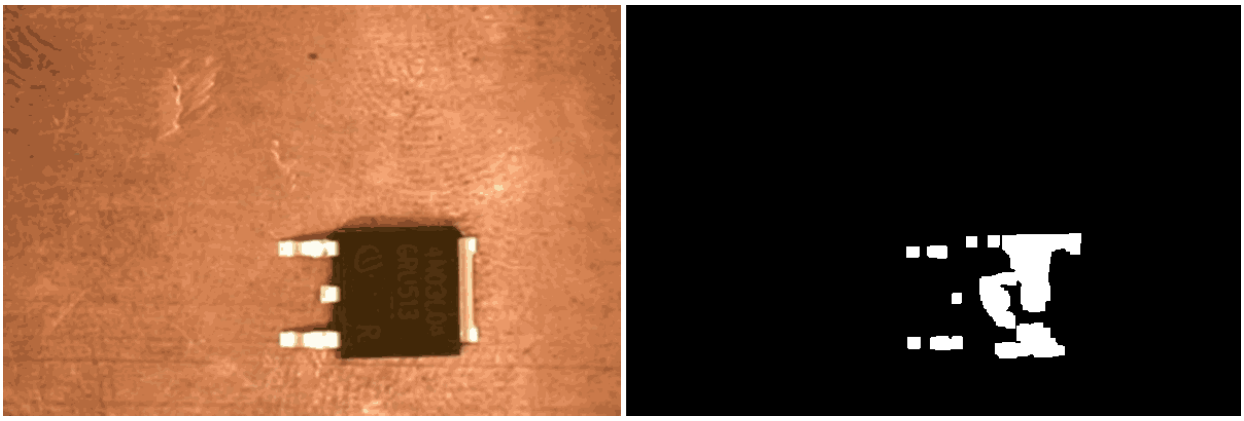
82. ábra: Klaszterezés és szegmentáció K = 6 esetén



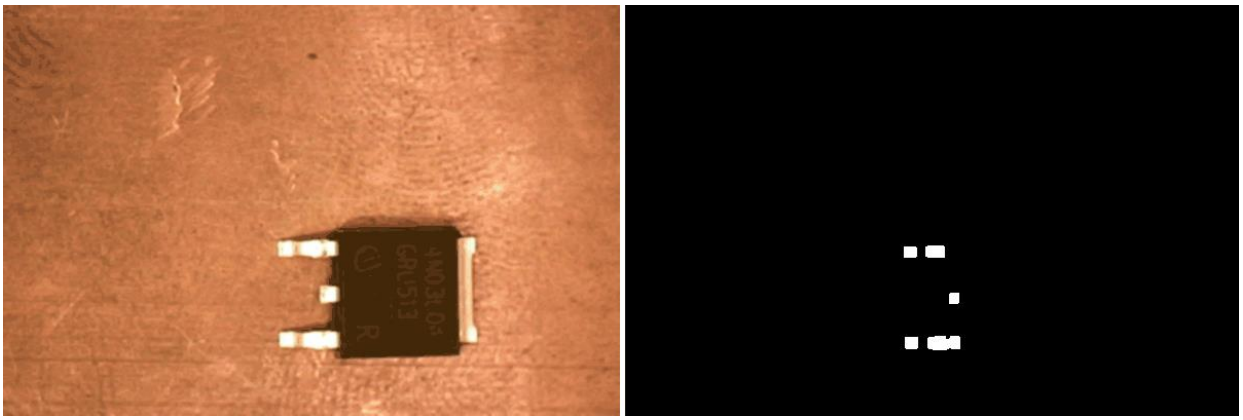
83. ábra: Klaszterezés és szegmentáció K = 8 esetén



84. ábra: Klaszterezés és szegmentáció K = 10 esetén

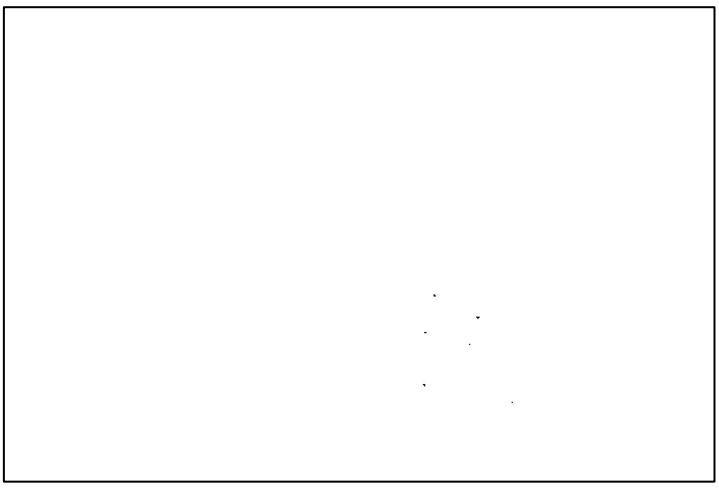


85. ábra: Klaszterezés és szegmentáció K = 30 esetén



86. ábra: Klaszterezés és szegmentáció K = 100 esetén

F8 A K-means algoritmust követő fényesség alapú szegmentáció javításának lépései

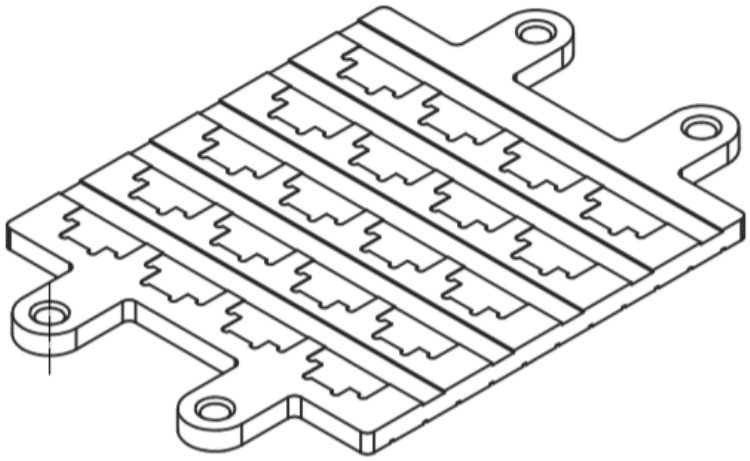


87. ábra: A *floodFill* függvény transzformációjának eredménye

88. ábra: Az invertálás eredménye

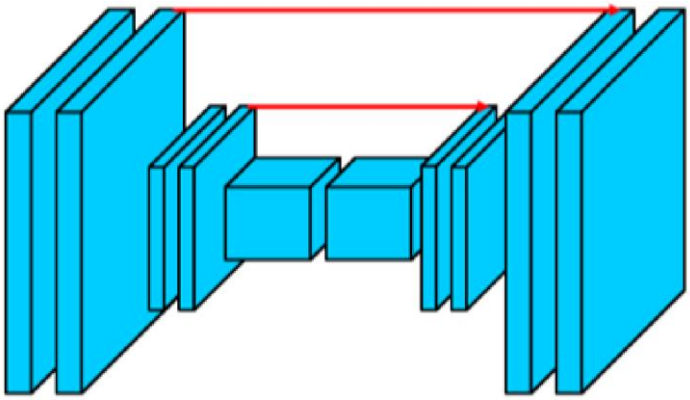
89. ábra: A logikai *vagy* művelet eredménye

F9 Speciális kialakítású hűtő rézlap különböző komponensekhez



90. ábra: Hűtést szolgáló rézlap különböző komponensek részére

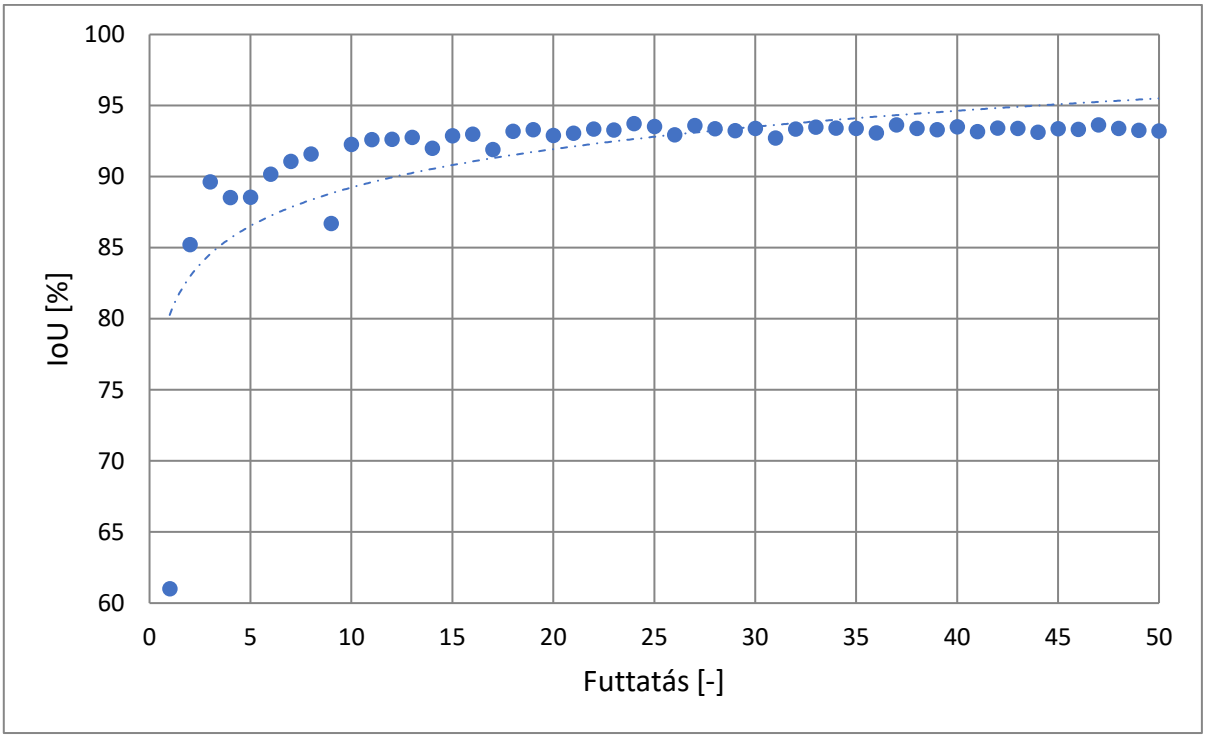
F10 Neurális háló szemléltetése



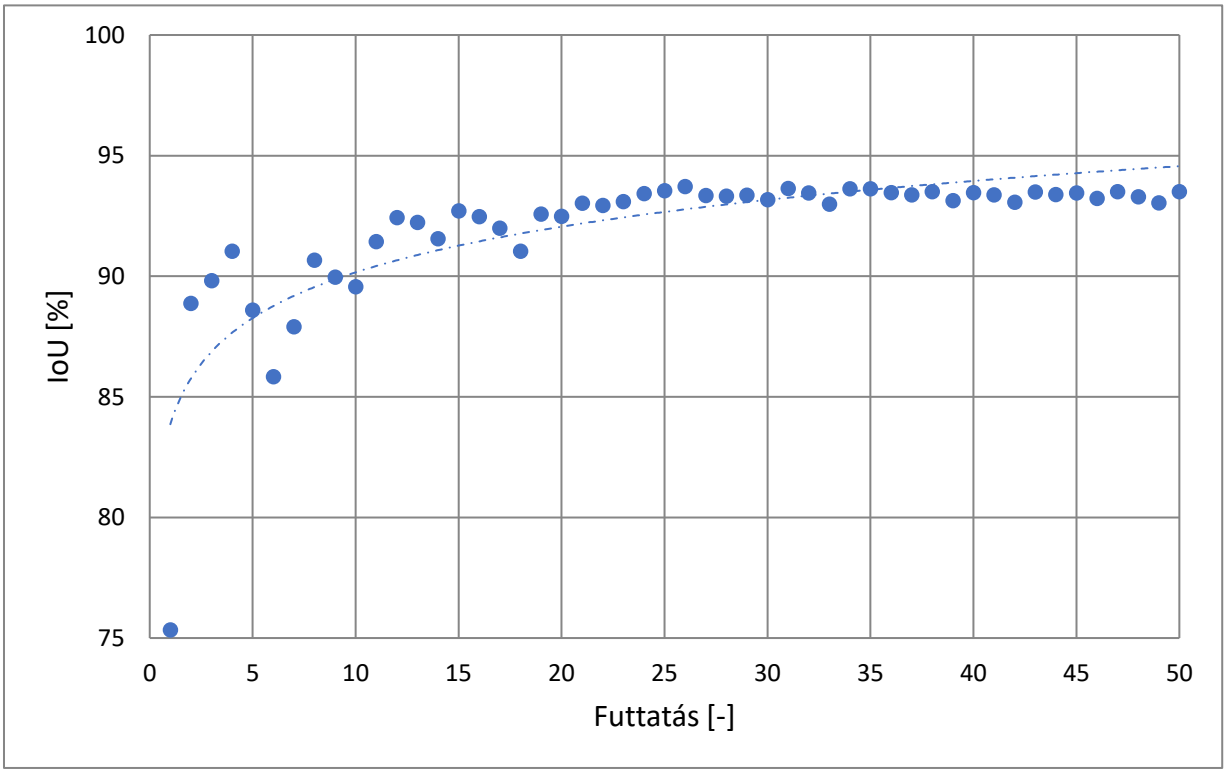
91. ábra: Neurális háló szemléltetése *m = 2* esetén [104]

F11 Legjobban teljesítő neurális hálók IoU eredményei

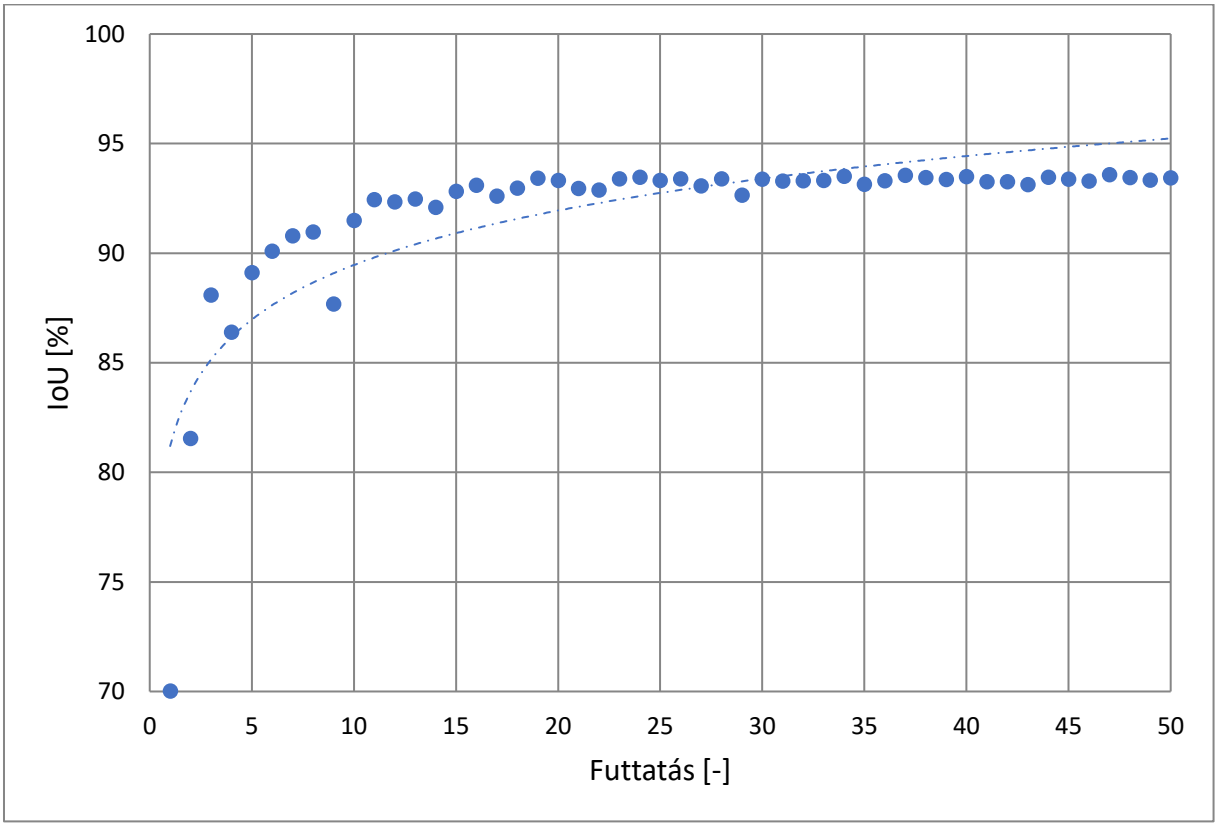
A legjobban teljesítő neurális háló IoU eredményeit az 5.4.2.3 fejezetben a 30. ábra mutatja be. A további, 93,5 % felett teljesítő hálók a 3. táblázat sorrendjében alább ábrázoltak.



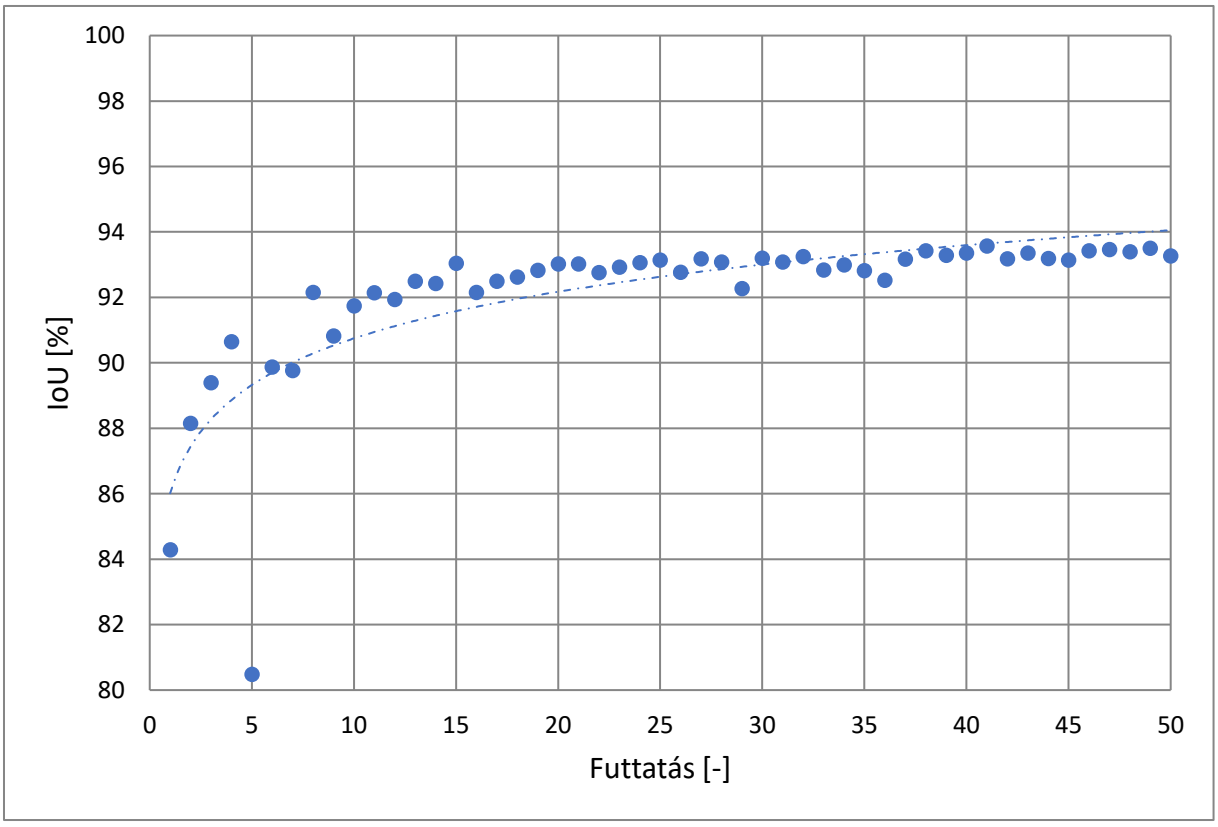
92. ábra: Második legjobban teljesítő háló IoU eredményei epochonként



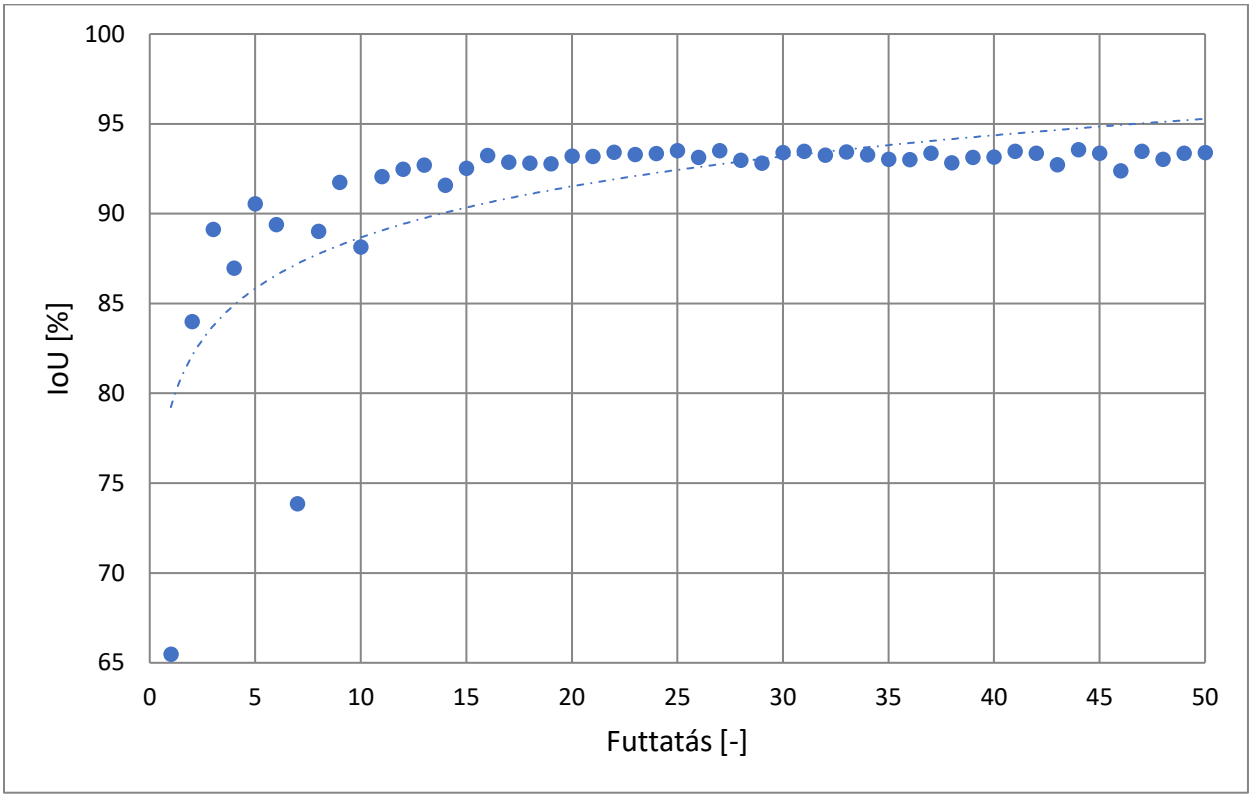
93. ábra: Harmadik legjobban teljesítő háló IoU eredményei epochonként



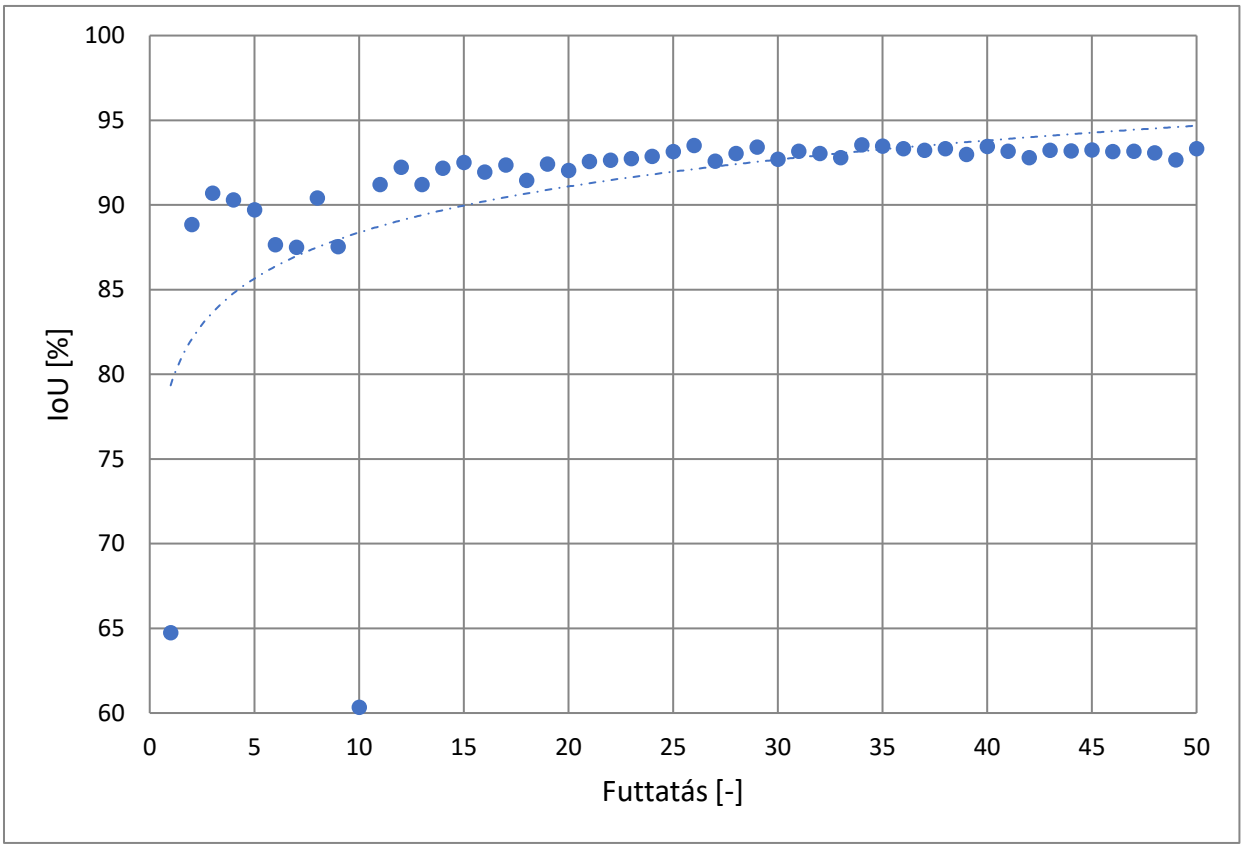
94. ábra: Negyedik legjobban teljesítő háló IoU eredményei epochonként



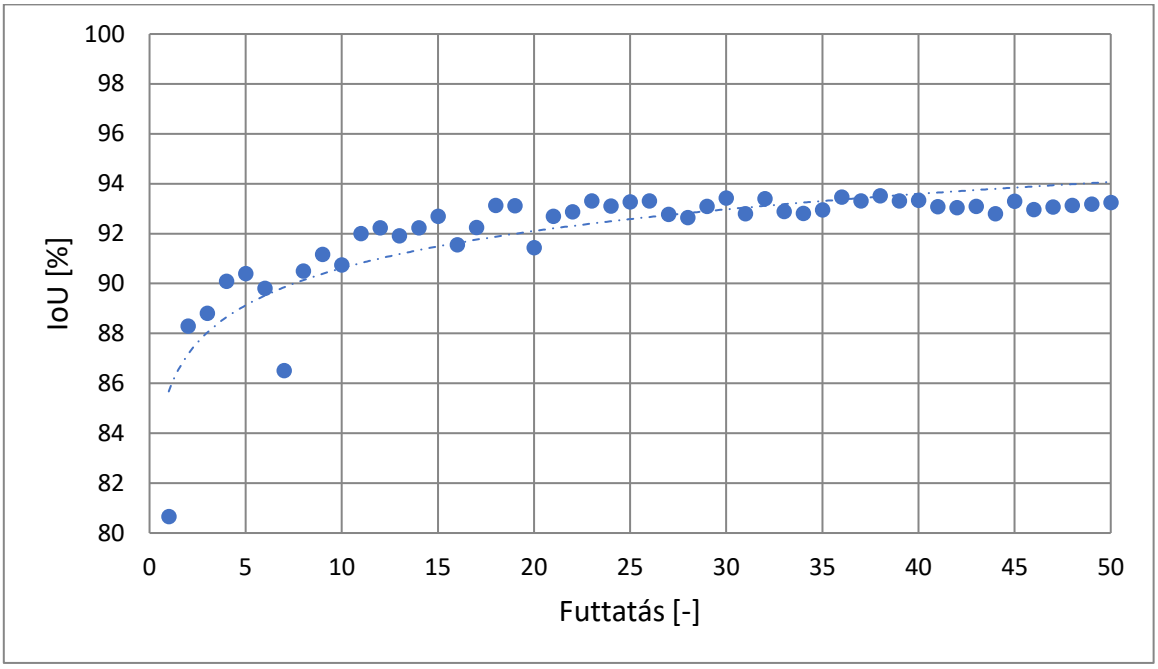
95. ábra: Ötödik legjobban teljesítő háló IoU eredményei epochonként



96. ábra: Hatodik legjobban teljesítő háló IoU eredményei epochonként



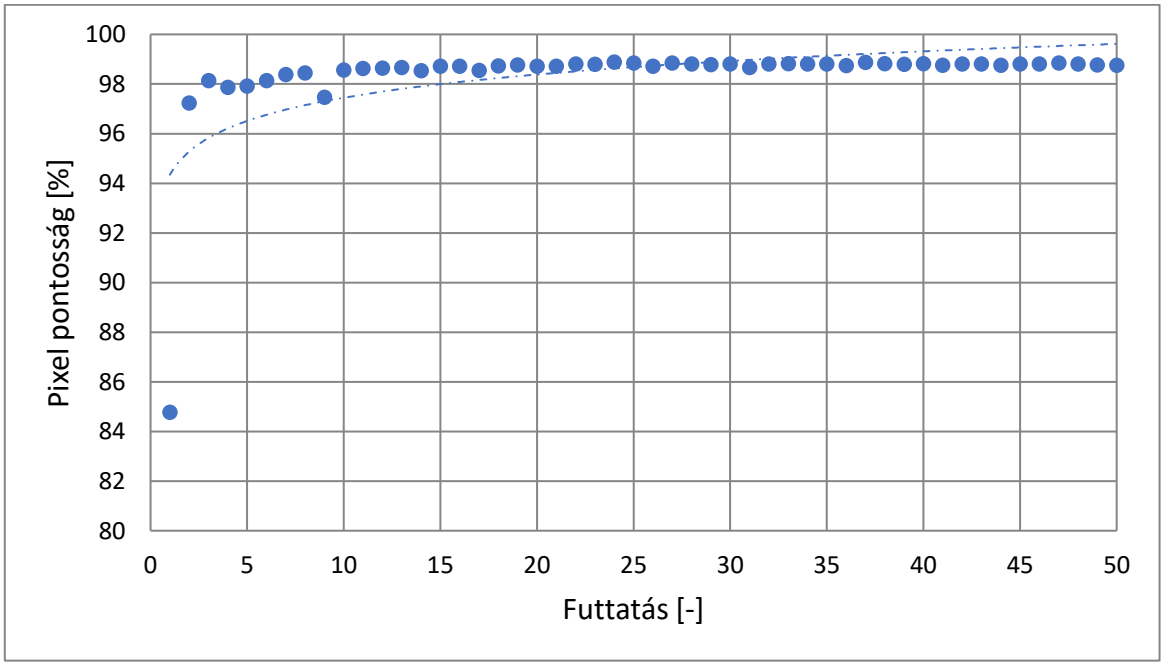
97. ábra: Hetedik legjobban teljesítő háló IoU eredményei epochonként



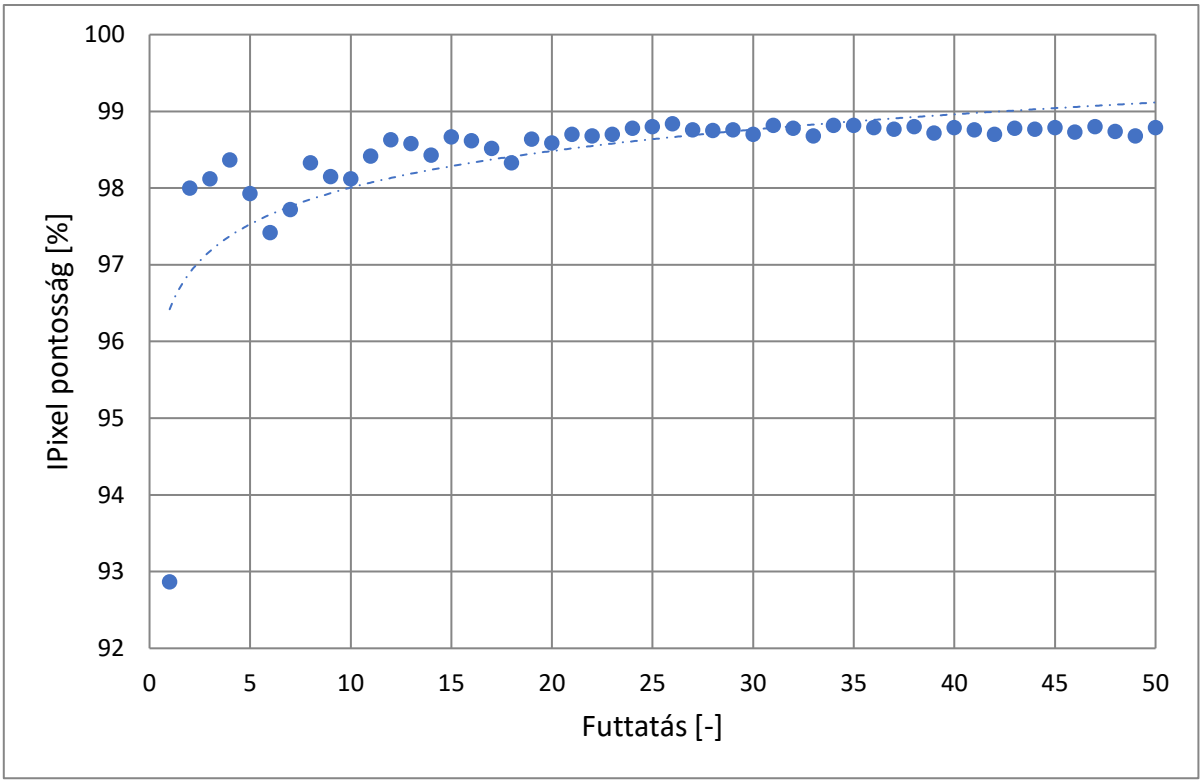
98. ábra: Nyolcadik legjobban teljesítő háló IoU eredményei epochonként

F12 Legjobban teljesítő neurális hálók pixel pontosságai

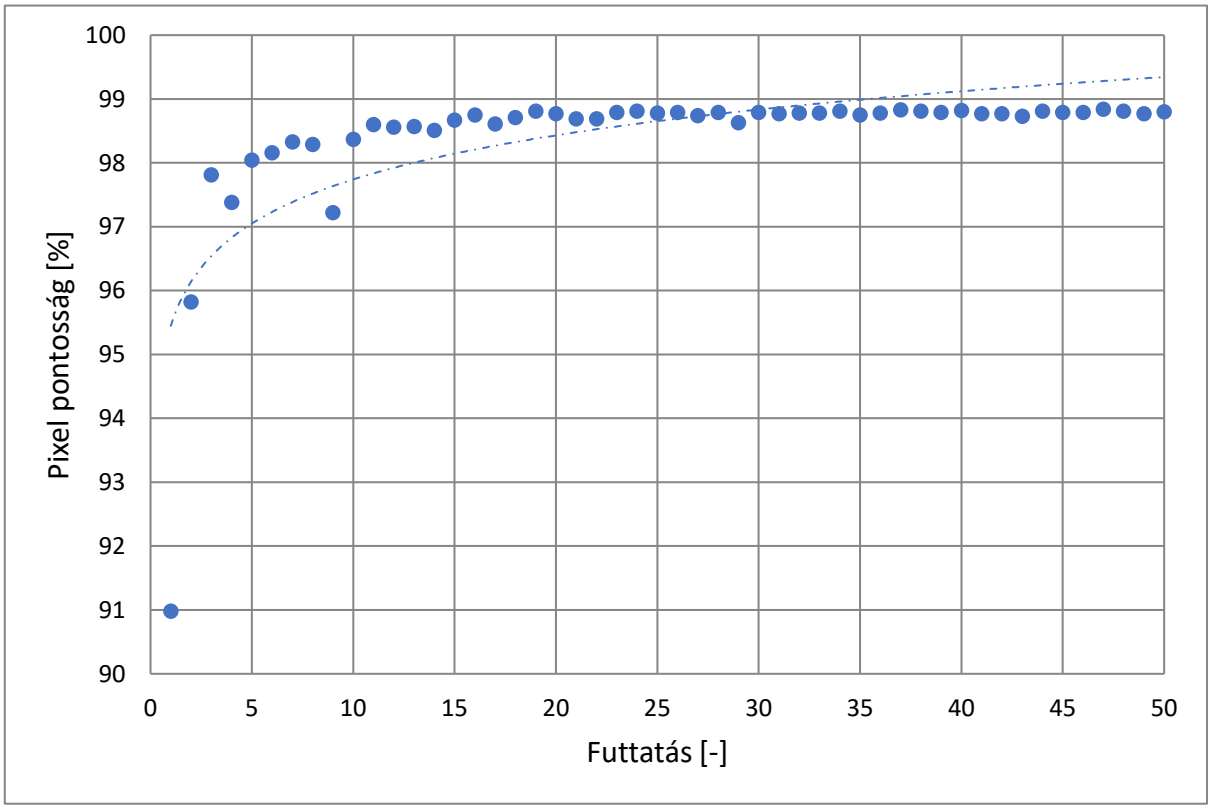
A 4.4.2.3 fejezetben a 2. táblázat első sorában tárgyalt neurális hálót a 31. ábra mutatja be. A további hálók eredményei sorrendben alább találhatók.



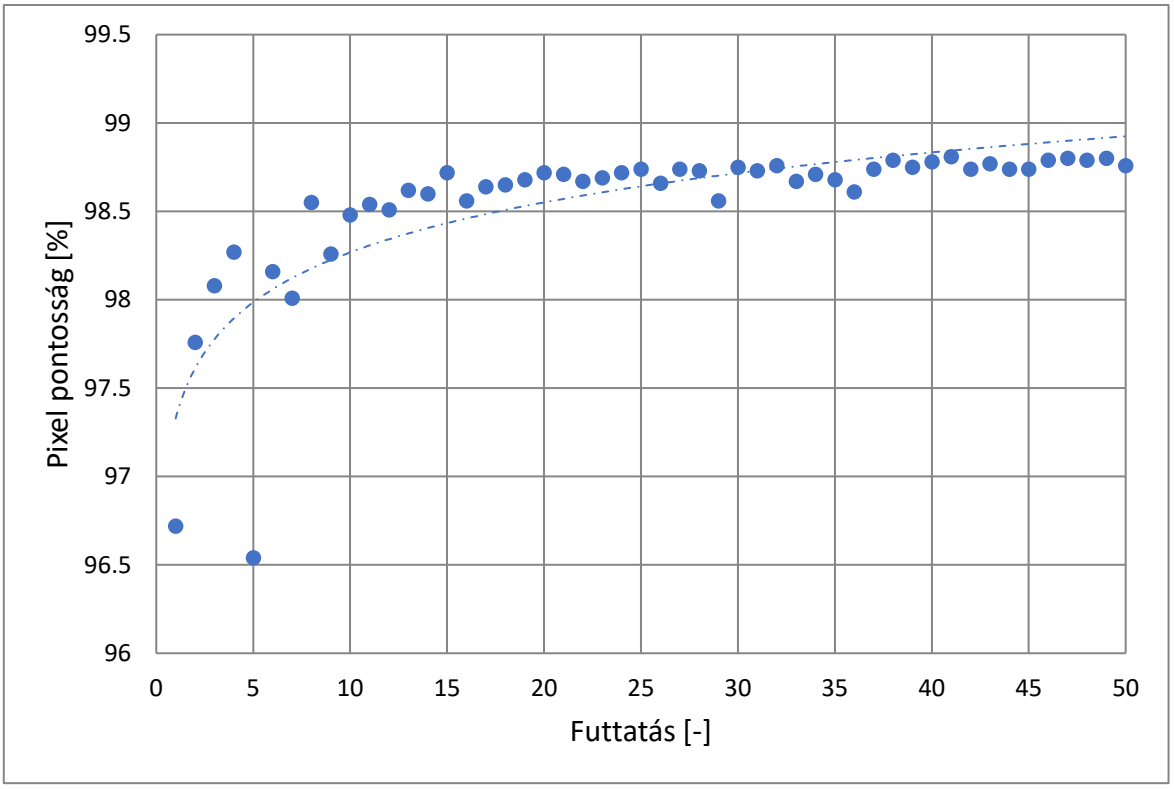
99. ábra: Második háló pixel pontossága



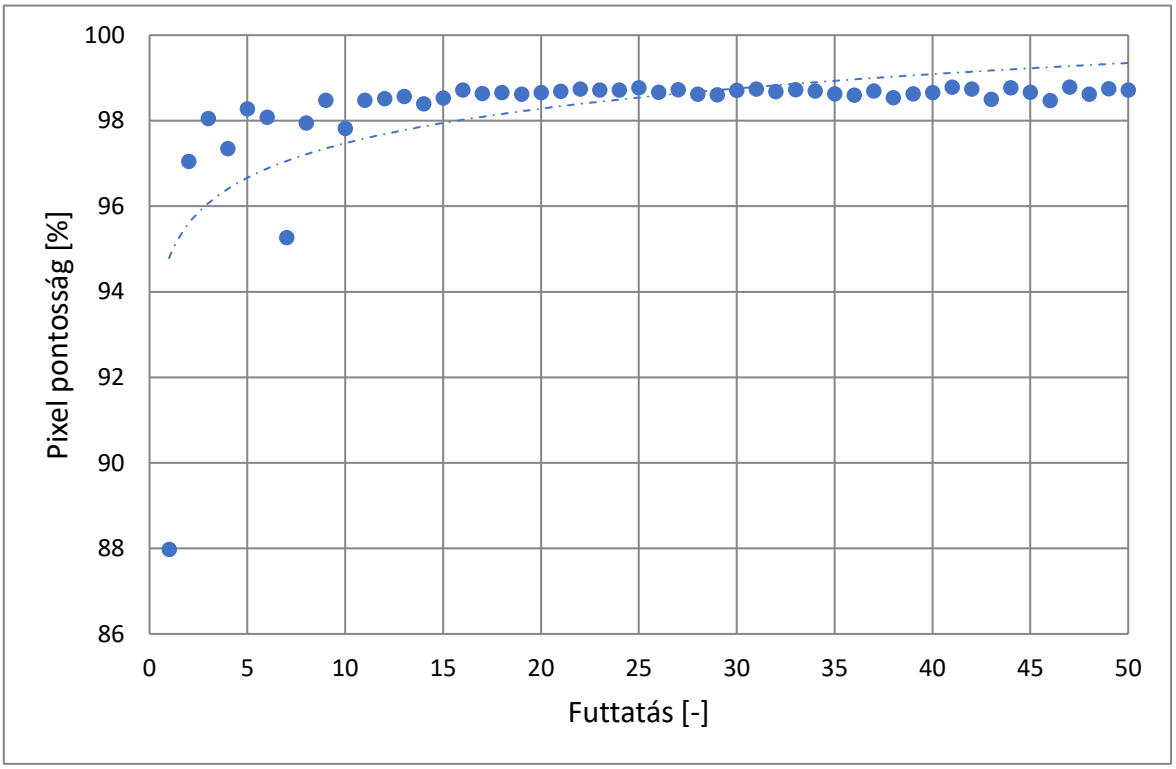
100. ábra: Harmadik háló pixel pontossága



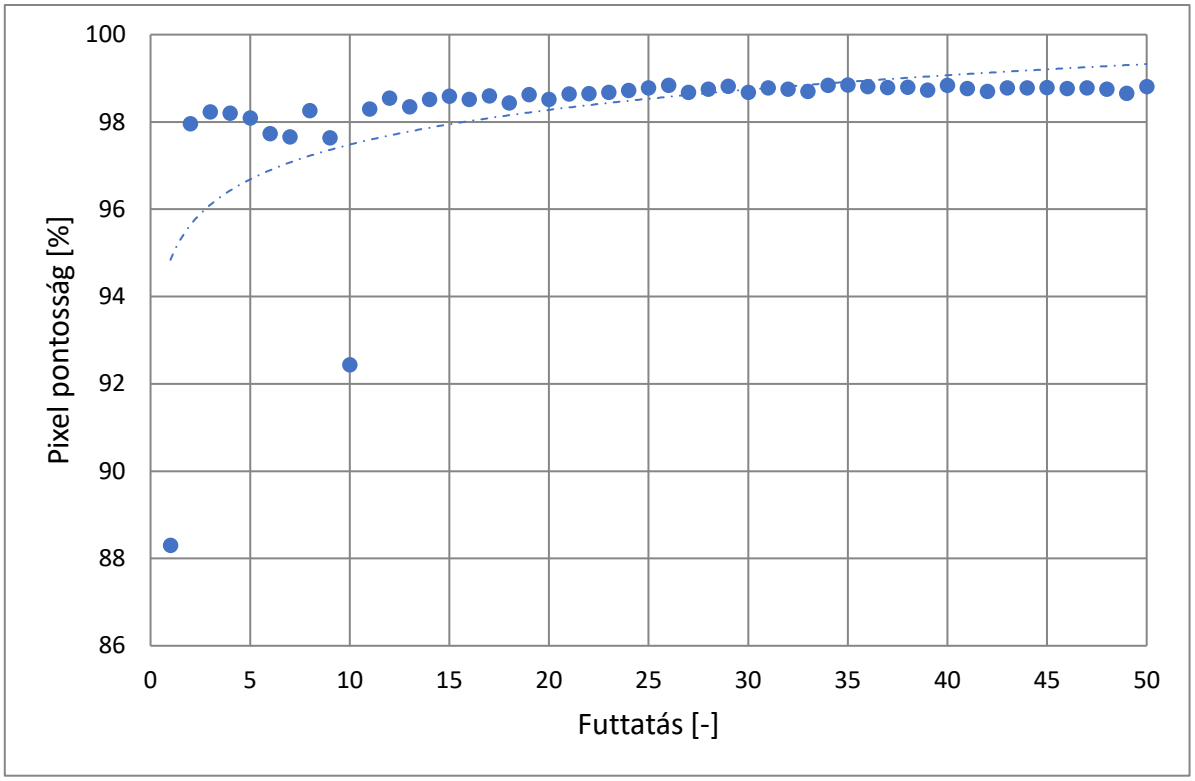
101. ábra: Negyedik háló pixel pontossága



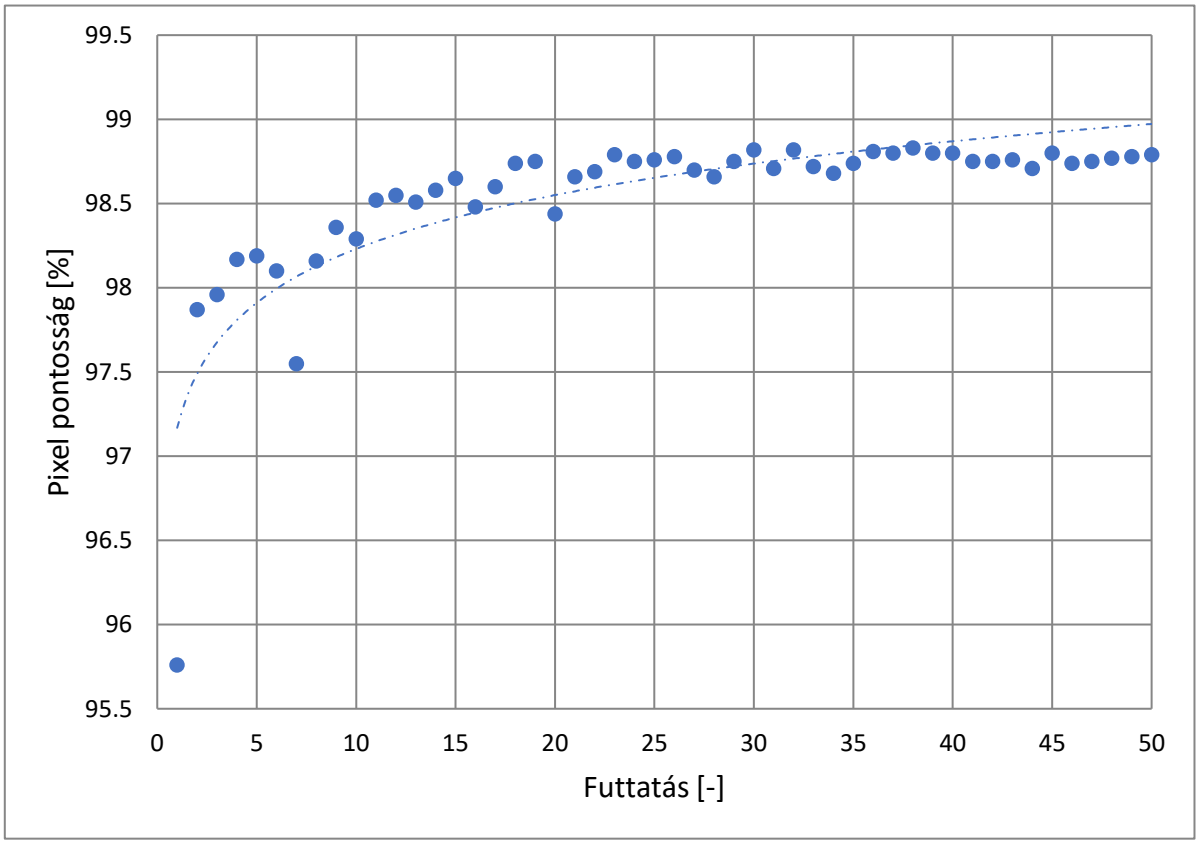
102. ábra: Ötödik háló pixel pontossága



103. ábra: Hatodik háló pixel pontossága

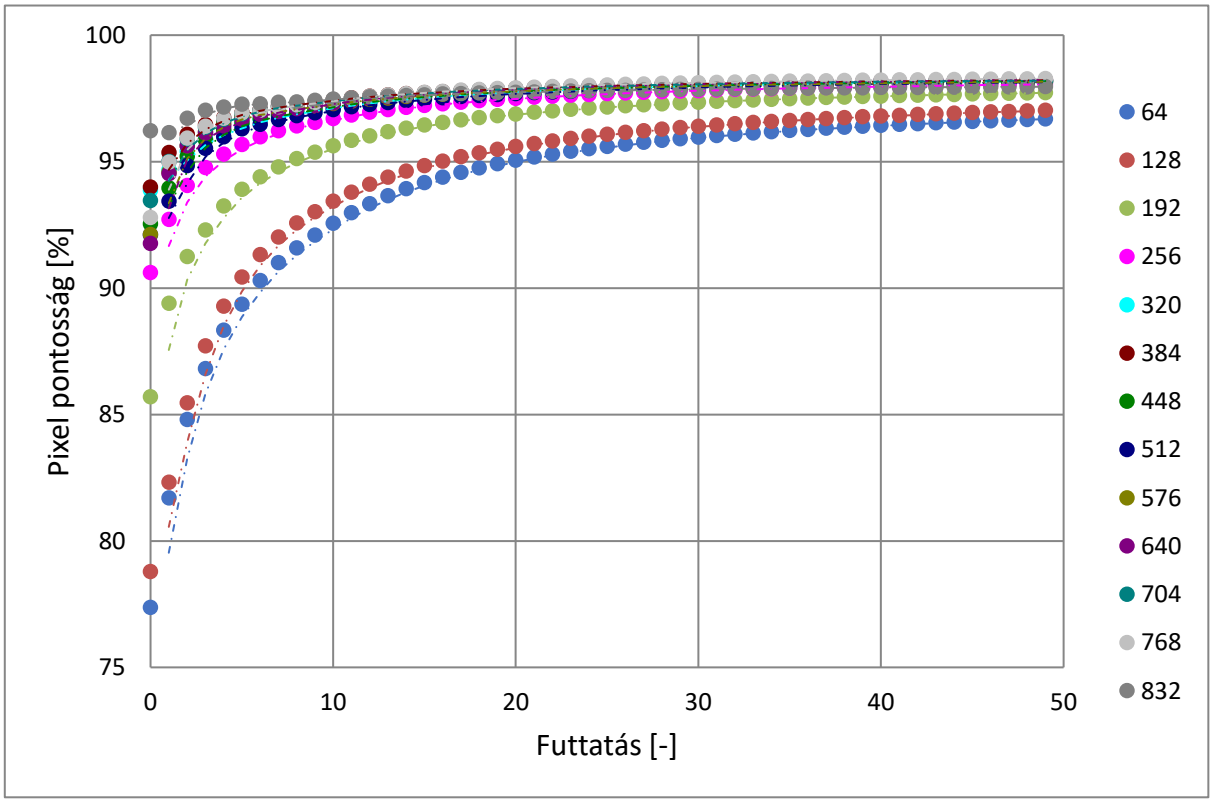


104. ábra: Hetedik háló pixel pontossága

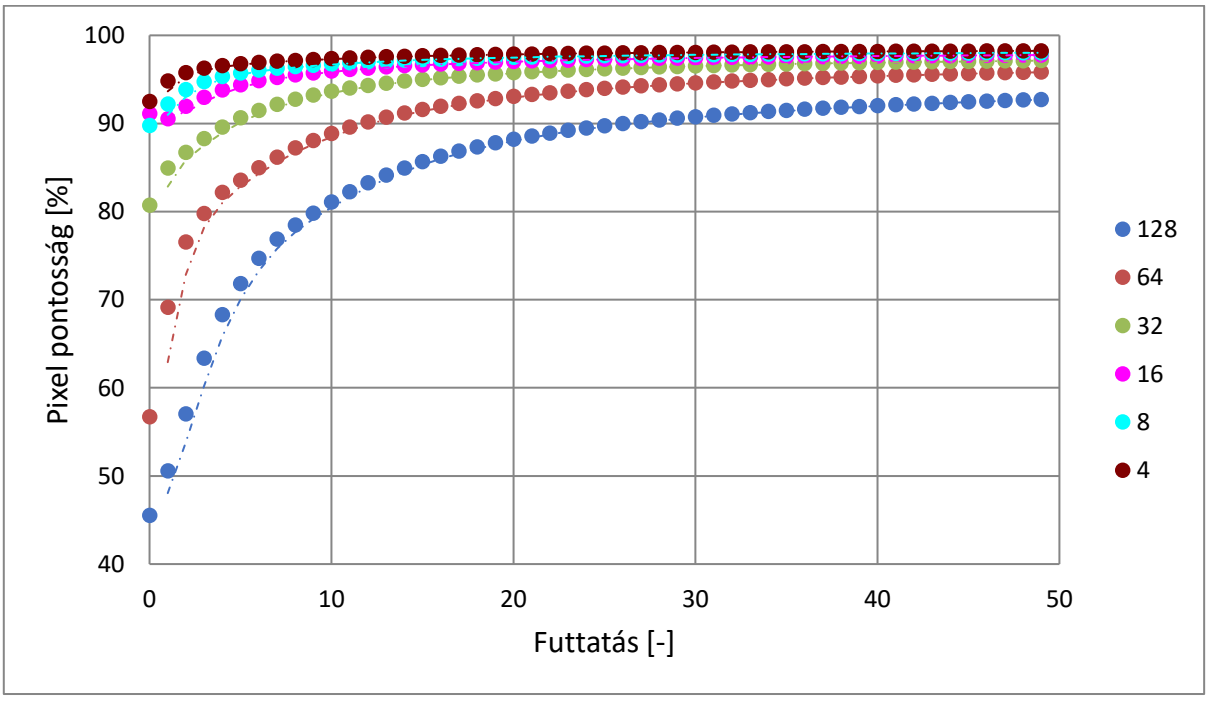


105. ábra: Nyolcadik háló pixel pontossága

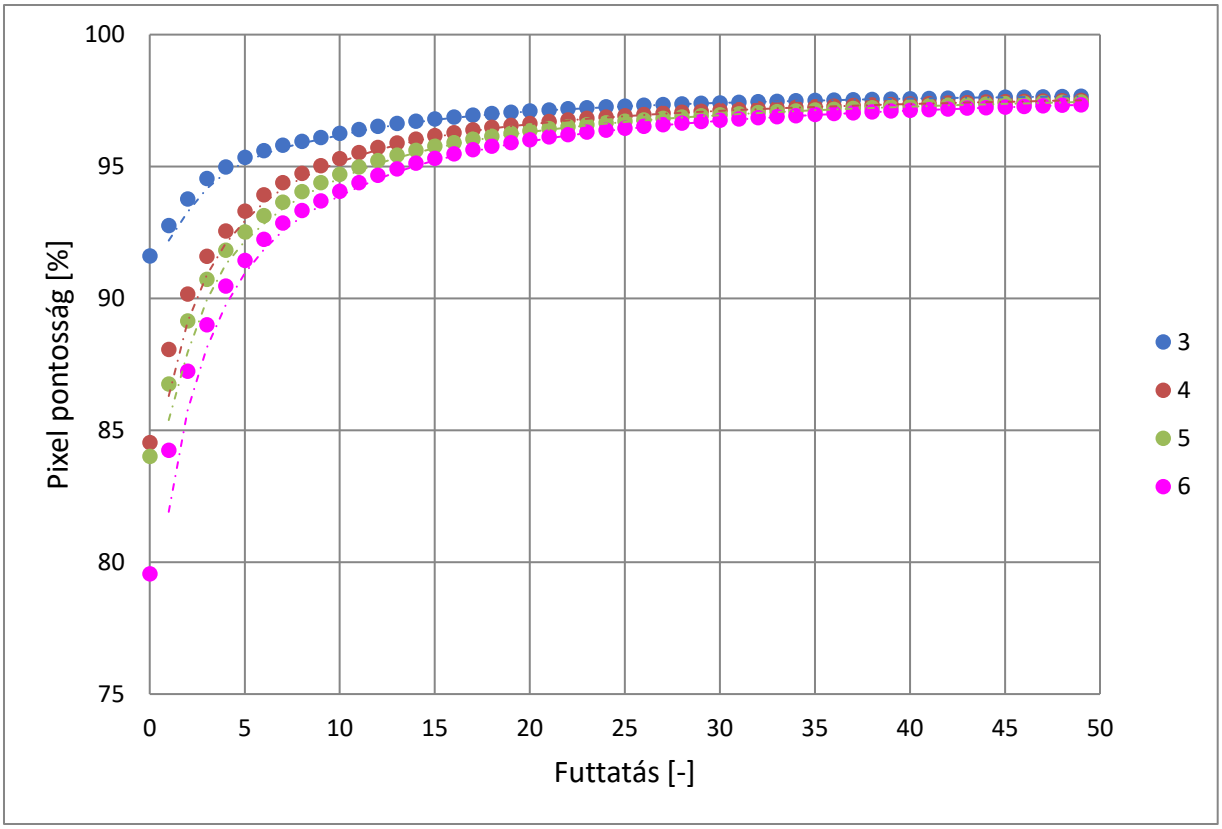
F13 Futtatások átlagos pixel pontosság, felbontás, batch méret, háló mélység, és bemeneti csatornaszám szempontjából



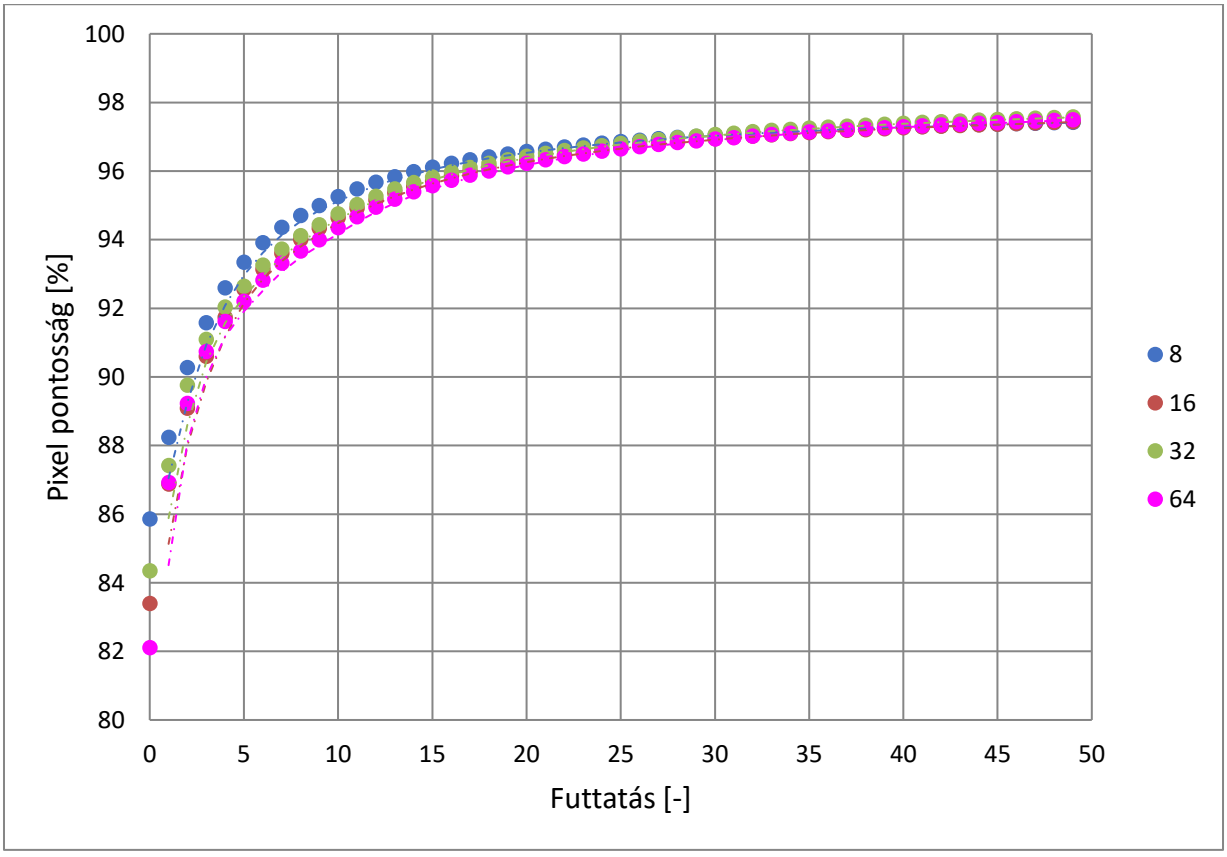
106. ábra: Különböző felbontású képeket használó hálók pixel pontosságai



107. ábra: Különböző batch mérettel dolgozó hálók pixel pontosságai

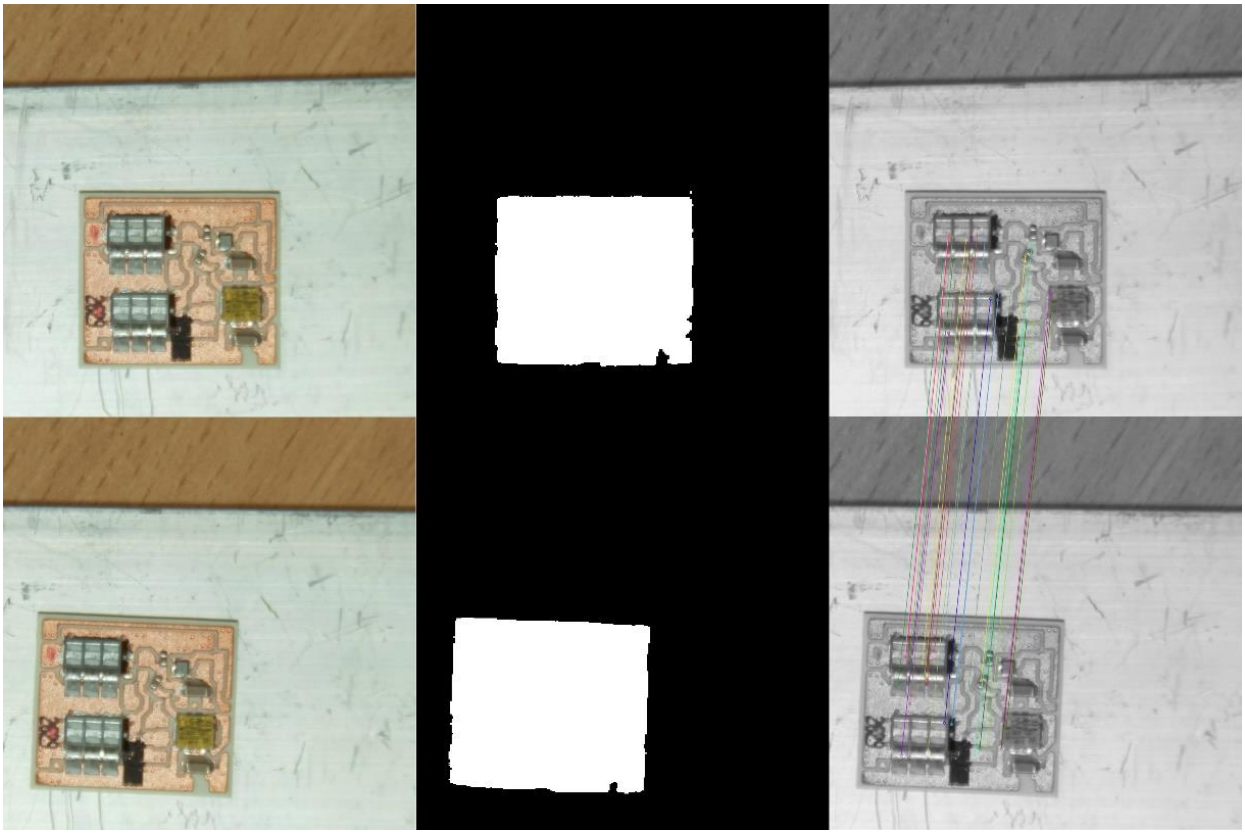


108. ábra: Különböző mélységű hálók pixel pontosságai

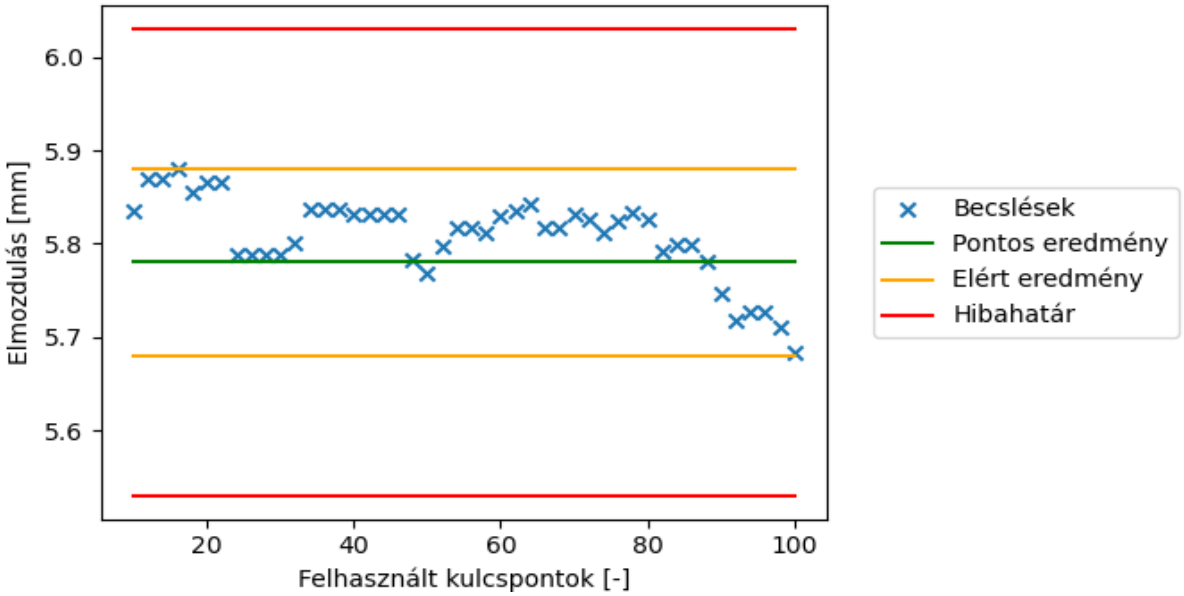


109. ábra: Különböző bemeneti csatornaszámú hálók eredményei

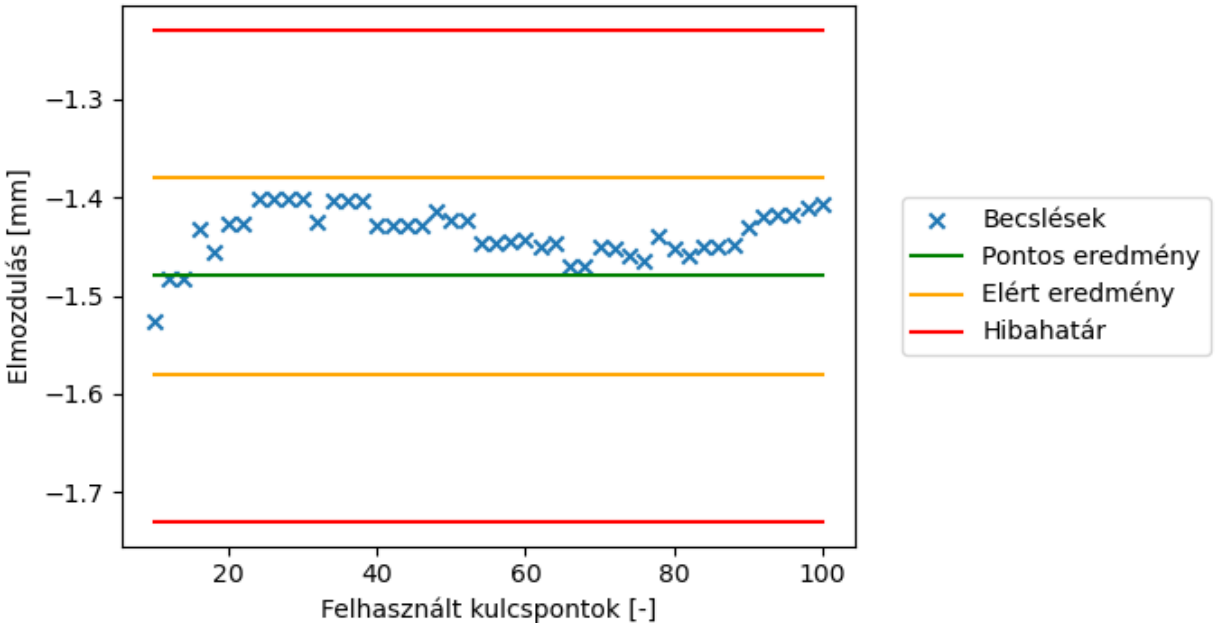
F14 DBC szemantikus szegmentációk és korrekciók példái



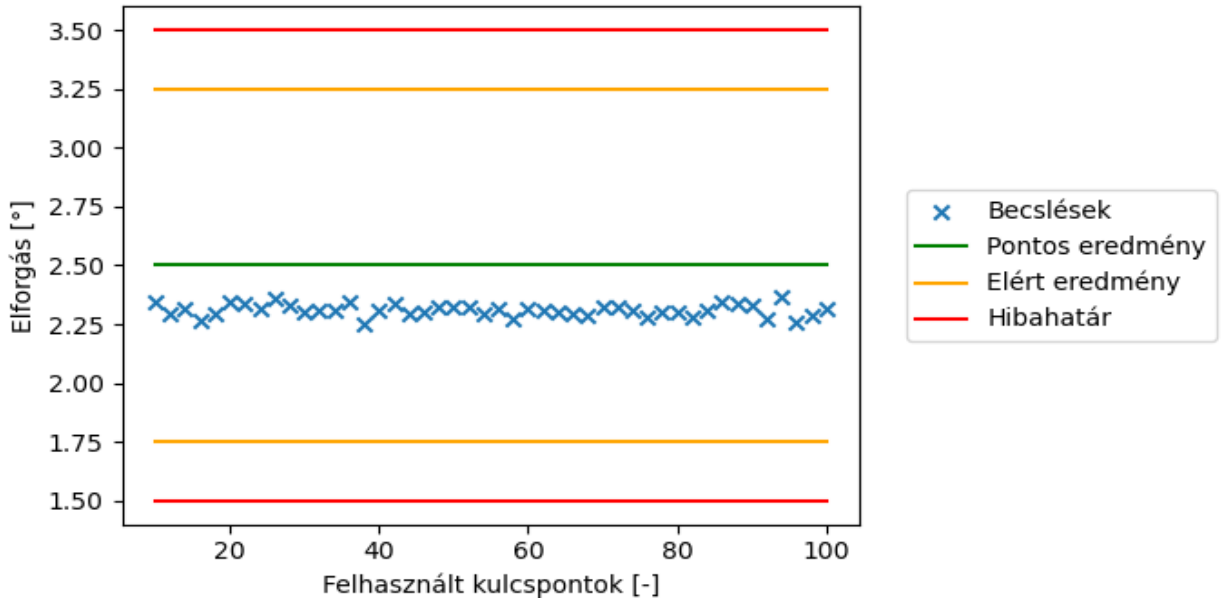
110. ábra: A DBC szemantikus szegmentáció és a SIFT algoritmus futásának eredménye 1.



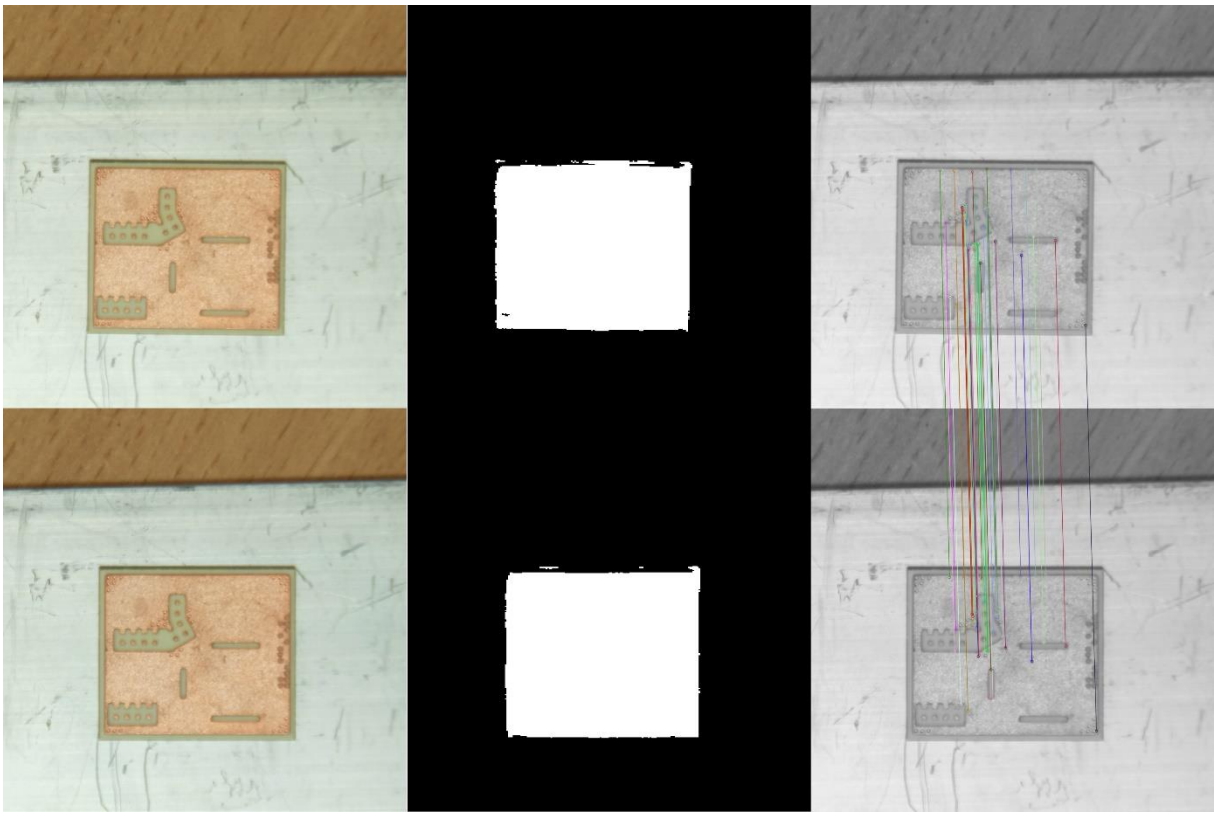
111. ábra: DBC X tengely mentén számított elmozdulás 1.



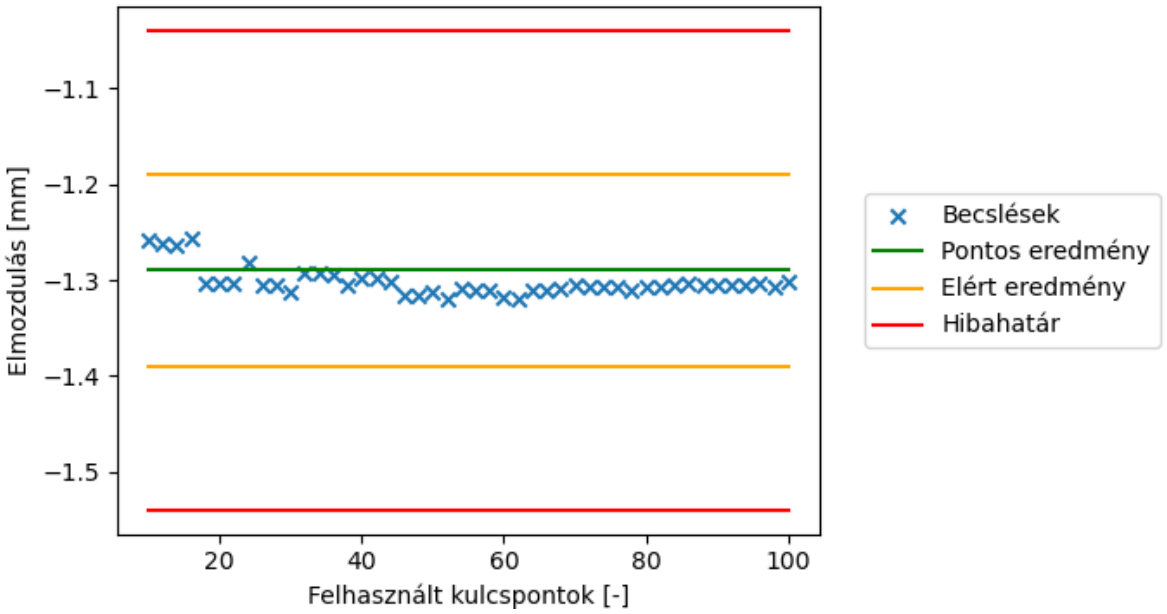
112. ábra: Y tengely mentén számított elmozdulás 1.



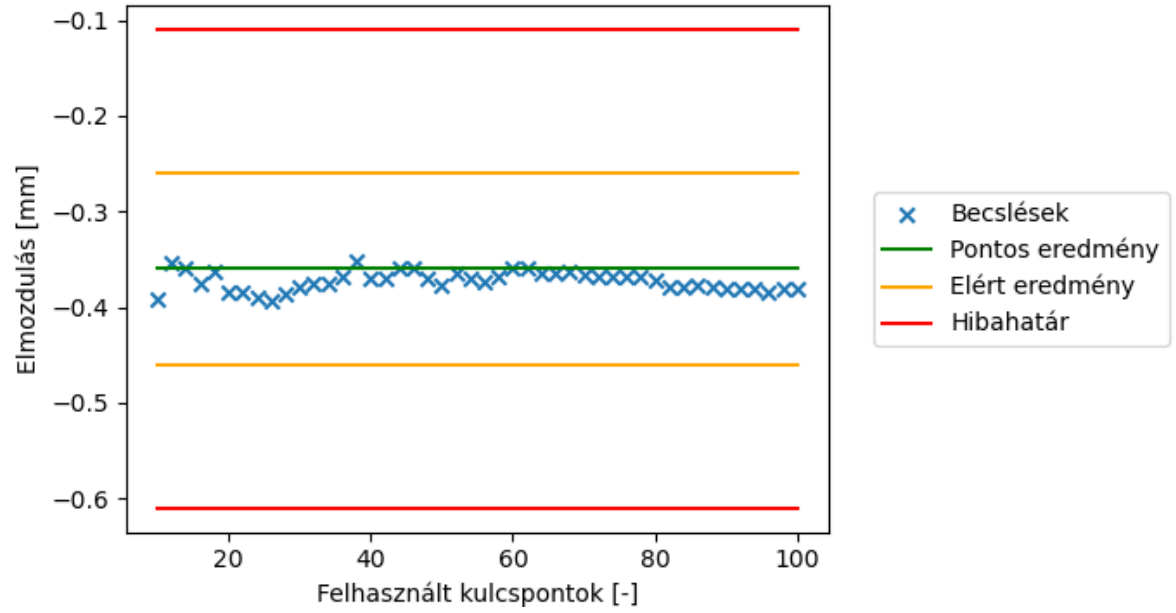
113. ábra: Z tengely körül számított elforgás 1.



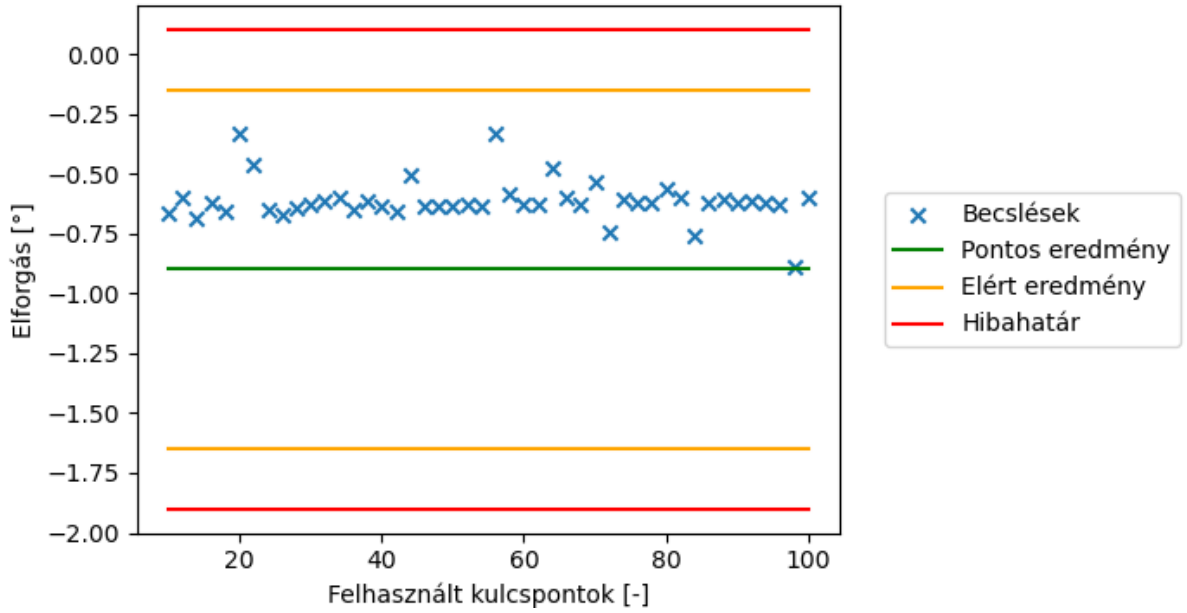
114. ábra: A szemantikus szegmentáció és a SIFT algoritmus futásának eredménye 2.



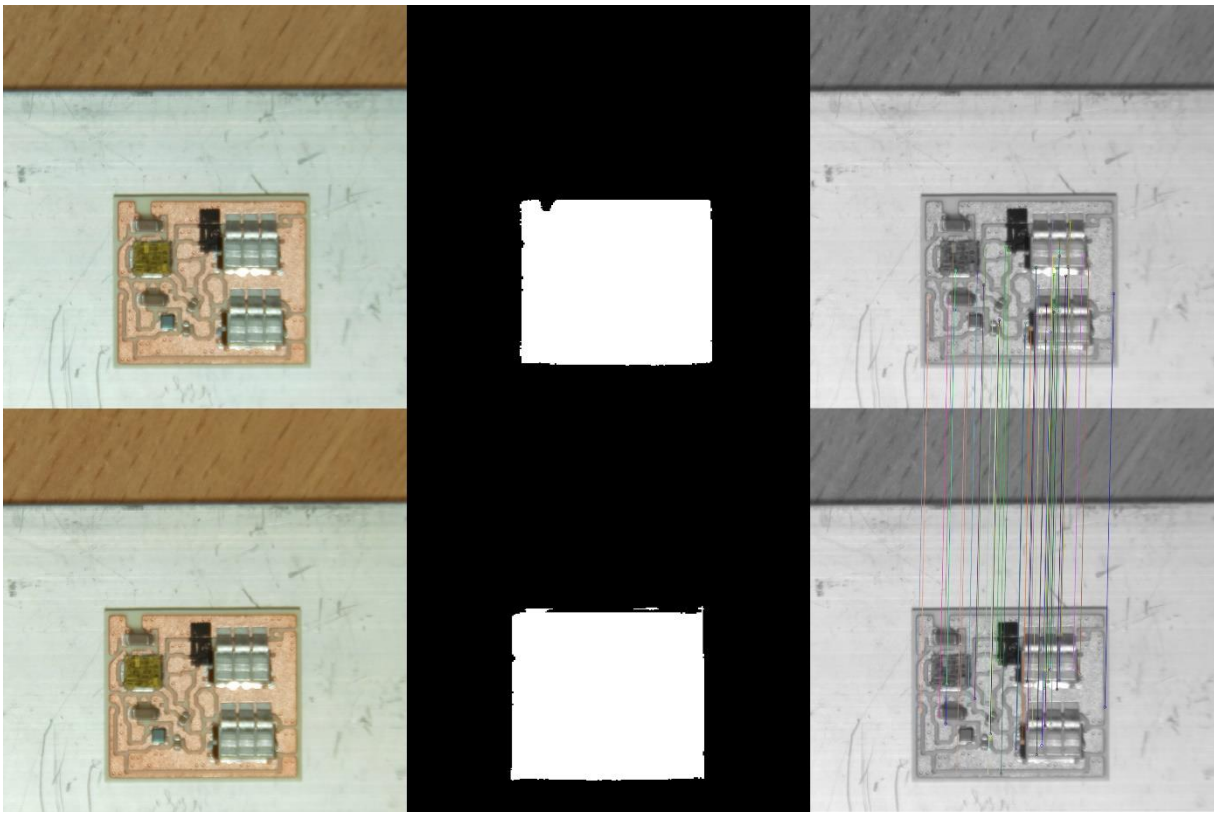
115. ábra: X tengely mentén számított elmozdulás 2.



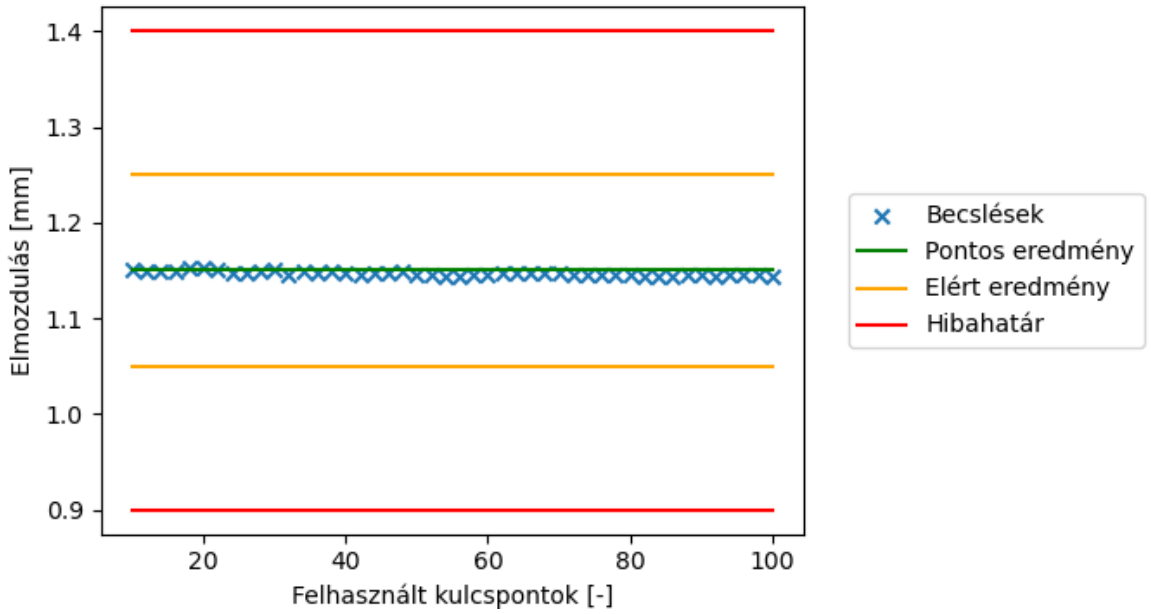
116. ábra: Y tengely mentén számított elmozdulás 2.



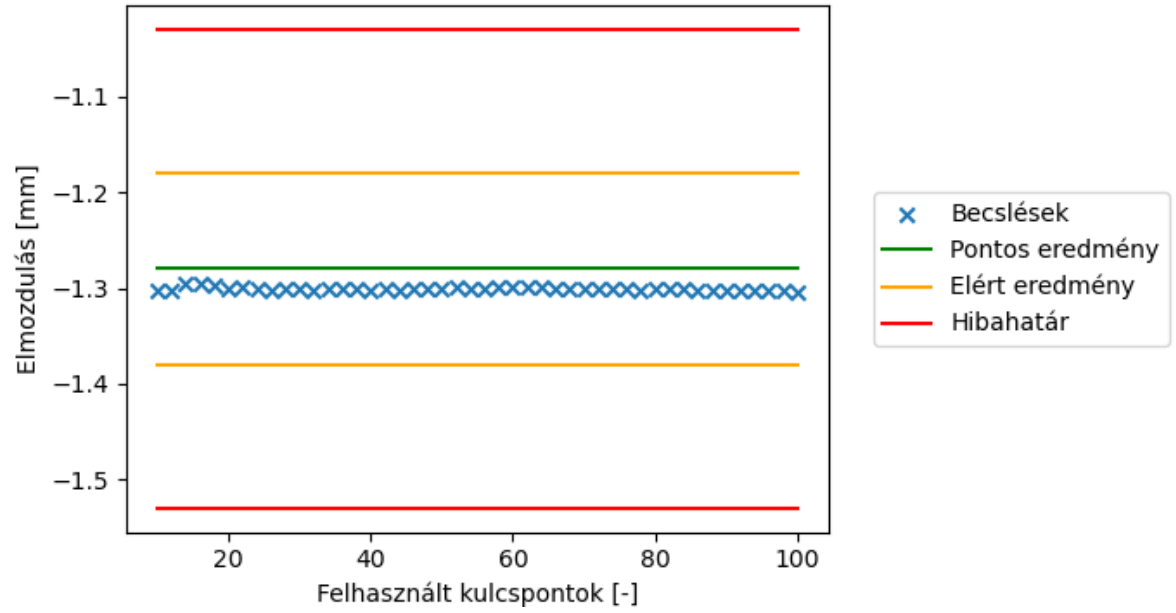
117. ábra: Z tengely körül számított elforgás 2.



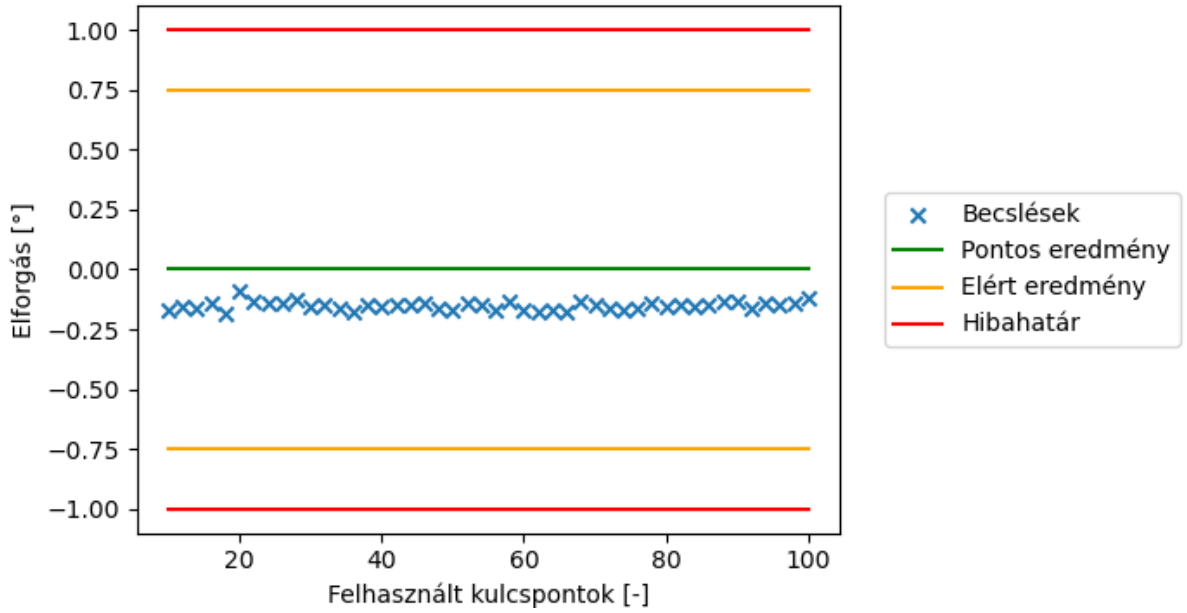
118. ábra: A szemantikus szegmentáció és a SIFT algoritmus futásának eredménye 3.



119. ábra: X tengely mentén számított elmozdulás 3.



120. ábra: Y tengely mentén számított elmozdulás 3.



121. ábra: Z tengely körül számított elforgás 3.