



TDK DOLGOZAT

Takács Patrik, Tóth Patrik

Gépi látáson alapuló automata robotprogram generálás inhomogén munkadarabok szortírozására

Konzulens:

Szántó Norbert

Monek Gergő Dávid

Győr

2023. január 09.

Széchenyi István Egyetem
Gépész, Informatikai és Villamosmérnöki Kar

TDK dolgozat

Gépi látáson alapuló automata
robotprogram generálás inhomogén
munkadarabok szortírozására

konzulens:

Szántó Norbert

egyetemi tanársegéd
GIVK Automatizálási
és Mechatronika
Tanszék

Monek Gergő Dávid

egyetemi tanársegéd
GIVK Automatizálási
és Mechatronika
Tanszék

Takács Patrik

Mechatronikai mérnök
alapszakos hallgató, 7. félév

Tóth Patrik

Mechatronikai mérnök
alapszakos hallgató, 7. félév

dolgozat lezárva:

2023. január 09.

Tartalomjegyzék

1. Bevezetés.....	1
2. Szakirodalmi áttekintés	2
2.1. Ipari robotok.....	2
2.2. Kollaboratív robot	3
2.3. Bin picking	4
2.4. Gépi látás és képfeldolgozás	6
2.4.1. Tárgy detektálása, súlypontjának és színjellemzőinek kiszámolása	8
2.4.2. Képfeldolgozás eszközei	9
2.5. Gyakorlati alkalmazási példák	12
3. Szortírozási folyamat fejlesztése	14
3.1. Munkakörnyezet ismertetése.....	14
3.1.1. Robot.....	15
3.1.2. Megfogó	16
3.1.3. Kamera	17
3.1.4. Szoftver	17
3.2. Fejlesztési folyamat lépéseinek bemutatása	19
3.2.1. Főprogram	21
3.2.2. Mozcásérzékelő.....	23
3.2.3. Kontúr kereső függvény	24
3.2.4. Alakzat ellenőrző program.....	29
3.2.5. Tárgyosztályozás	31
3.2.6. Lista ellenőrző program	38
3.2.7. Szortírozás.....	39
3.3. A rendszer korlátai	43
4. Összefoglalás.....	45
5. Irodalomjegyzék.....	46

Ábra- és táblázatjegyzék

1. ábra Mozgásgeneráló architektúra felépítése [56]	5
2. ábra Laboratóriumban rendelkezésre álló munkaasztal	14
3. ábra UR5 robot, vezérlő és TP	15
4. ábra RobotIQ 2F-85 megfogó [59]	16
5. ábra Intel RealSense D435 kamera	17
6. ábra Folyamatábra I.	19
7. ábra Folyamatábra II.	20
8. ábra Munkakörnyezet.....	21
9. ábra Alulcsordult kép	22
10. ábra Abszolútértékes kivonás.....	23
11. ábra Mozgásérzékelő diagramja.....	24
12. ábra Szűrés alapján detektált elem	25
13. ábra Dilatáció és erózió szemléltetése [64].....	27
14. ábra Erózió dilatáció előtti és utáni kép	27
15. ábra Minimális területű körülrajzolt négyzet	29
16. ábra Folyamatábra III.....	30
17. ábra A végső folyamatban résztvevő elemek: henger(a), hosszúkás téglatest(b), rózsaszín téglatest(c), kék téglatest(d)	35
18. ábra Henger maszkolása	36
19. ábra Henger kontúrja és a feature vectorok értékei	37
20. ábra A program jóslata arra, milyen tárgyak jelennek meg a képen	38
21. ábra Robot koordináta rendszer	39
22. ábra Területmeghatározás.....	41
23. ábra Bepakolás	44
24. ábra Munkadarab megfogása	44
25. ábra Tároló konténerbe helyezés.....	44
1. táblázat A RobotIQ 2F-85 megfogó technikai adatai [58]	16
2. táblázat RealSense D435 technikai adatai	17

1. Bevezetés

A kollaboratív robotok az iparban számos módon alkalmazhatók, technológiájukból adódóan alkalmasak automatizálási megoldásokra, együttműködő jellegükből fakadóan pedig élő munkát igénylő környezetben is egyszerűen telepíthetők. Alkalmazásukkal csökkenthetők a gyártási költségek és növelhető a termelékenység. Az ömlesztett áru kommissiózás (bin picking) egy vizuális alapú anyagmozgatás (pick and place), amely napjaink két legjellemzőbb automatizálási technológiájára épül, a gépi látásra és a robotizációra. A gépi látás robotokkal való kombinációja új automatizálási lehetőségeket nyitott meg, a szenzorokkal vagy kamerával felszerelt robotok a véletlenszerűen elhelyezett tárgyak felismerésére és mozgatására lettek képesek. A TMDK dolgozat célja egy kollaboratív robotos, digitális képfeldolgozás alapú demonstrációs szortírozó eszköz fejlesztése. A folyamat során színben és alakban is különböző munkadarabok felismerése majd szortírozása valósul meg, az előre definiált paraméterek függvényében. A munkadarabok munkaterületen való elhelyezése során videófelvétel készül, amely a digitális képfeldolgozás segítségével képkockaként feldolgozásra kerül így meghatározva a munkadarab beérkezési sorrendet és a geometriát. A következő lépésben az egyes munkadarabok középponti koordinátái és irányultsága kerül meghatározásra, amely információk alapján megtörténik az automata robotprogram generálás. Ennek eredményeképpen a különböző geometriákhoz előzőleg definiált tárolókba a munkadarabok beadási sorrendjében (FIFO) a kollaboratív robot végrehajtja a szortírozást. A demonstráció az előre meghatározott maximális munkadarab szám elérésekor vagy kevesebb darabszám esetében felhasználói gombnyomásra indul.

2. Szakirodalmi áttekintés

Napjainkban a kollaboratív robotok nagy szerepet játszanak az emberi élet minden területén, mert az igény alacsony költségű, magas színvonalú és gyorsan előállított termékek iránt egyre növekszik a kereslet. Ez a munkavállalók számára is új helyzetet teremt, mivel ezek a robotok különösen előnyösek lehetnek, veszélyes munkakörnyezetekben, vagy nehéz környezeti feltételek mellett az ember közelében alkalmazva. Annak érdekében, hogy megértsük a fogalmát és a jelentőségét a kollaboratív robotoknak, fontos megérteni az ipari robotok és környezetük folyamatainak működését. [1, 2, 3, 4, 5]

2.1. Ipari robotok

Manapság a legtöbb szervezet egyetért az ipari robotok definíciójával, melyet a Nemzetközi Szabványügyi Szervezet fogalmazott meg: „Az ipari robot egy automatikusan vezérelt, újraprogramozható, többfunkciós, több szabadságfokkal rendelkező, manipulatív gép, amely lehet helyhez kötött vagy mobil az ipari alkalmazásokban való használata szerint”. A robotok megkülönböztetése valahol a programozhatóság kifinomultságában rejlik. Ha egy mechanikus eszköz sokféle feladatra programozható akkor az valószínűleg egy ipari robot. Példának okáért egy numerikus vezérlésű (NC) marógép nem ipari robot. A lényeges különbség egy ipari robot és egy NC gép között a robot sokoldalúsága. Az NC gép egy speciális feladatra van kitalálva, míg a robot sokoldalúsága biztosítva van a különböző típusú megfogók által. [6,7]

Az ipari robotok tudományának területe és irányítása nem új tudomány, inkább a meglévő klasszikus területek keveréke. A gépészetből a gép statikus és dinamikus helyzeteit, a matematikából a térbeli mozgások leírását tudjuk megkapni. Irányításelméletből különböző eszközök állnak rendelkezésre az algoritmusok tervezéséhez és kiértékeléséhez, a kívánt mozgás eléréséhez. Az elektrotechnika hasznos az érzékelők tervezésénél. A számítástechnika pedig biztosítja a megfelelő eszközöket a programozáshoz. [8]

Az első robotokat olyan egyszerű feladatokra használták, mint a pick and place. Az embereket helyettesítették a monoton, ismétlődő, nehéz és veszélyes feladatokban. Amikor a robotok már képesek voltak bonyolultabb mozgásokat is végezni, akkor összetettebb alkalmazások következtek, mint a hegesztés, csiszolás, összeszerelés. Az ipari robotok napjainkban nagyjából a következőkre oszthatók fel: anyagmozgatás, technológiai műveletek és összeszerelés. Az ipari robotok feltalálása 1954-re nyúlik vissza, amikor is megalapították az elsőrobotcéget. 1961-ben helyezték üzembe az első robotot a General Motors üzemében, amit ponthegesztéshez és munkadarab kezeléshez használtak. Ezután sok más cég is elkezdte

az ipari robotok fejlesztését és gyártását, ezzel egy innováció vezérelt iparág született. [9, 10]

1969-ben megalkották a 6 szabadságfokú robotkart. Ez azért volt fontos mivel 6 szabadságfokkal kell rendelkeznie egy manipulátornak, hogy szabadon tudjon manipulálni egy tárgyat a háromdimenziós térben. A technológia fejlődésével a robotkarok egyre több feladat elvégzésére voltak képesek. A robot sebességével, pontosságával kapcsolatos követelmények ezzel együtt nőttek tovább. A megfelelő struktúrák utáni fejlesztések az 1980-as évek óta folynak a párhuzamos kinematikai gépek kifejlesztésével. A párhuzamos, illetve soros robotok osztályozása a robotok szerkezeti topológiája szerint történik. Egy robot soros (nyílt hurkú) robot - a következő ábra első képén látható – ha a kinematikai felépítése nyitott hurokláncú. Ha a robot párhuzamos akkor az zárt hurokláncú. Ezek az úgynevezett párhuzamos robotok különösen alkalmasak a rövid ciklusidők elérésére (pl. anyageltávolításra, kommissiózásra). A robot képessége, hogy képes a biztonságos ember-robot kollaborációra lett az újabb gyártói elképzelés, amivel bővíteni akarták a robotok képességeit és termelékenységét a munkahelyeken. 1995-ben a General Motors alapítvány által vezetett kutatás hozott létre elsőként kollaboratív, vagyis együttműködő robotot. [11, 12, 13, 14]

2.2. Kollaboratív robot

Az ember-robot együttműködés lehetővé teszi a kombinációját a robotok és az emberek tipikus erősségeinek. Az ipari robotok erősségei közé sorolhatjuk a nagy állóképességüket, nagy teherbírásukat, pontosságukat és ismétlődőképességüket. Az emberi munkások erősségei, melyekhez egyetlen gép sem párosul az a rugalmasság, a kreatív problémamegoldási készségek és az előre nem látható helyzetekre való reagálás képessége. Az ipari robotok azonban jelentős károkat okozhatnak az emberekben, feladatuk elvégzését a programjuk alapján végzik és nem foglalkoznak azzal, ha valami az útjukba kerül, környezetük megváltozik. Emiatt általában az ipari robotoknak külön munkacellája van, ahol dolgozik. Itt elkerítve egy védőfalon túl a robotkar és a perifériái – megfogók, szerszámok, biztonsági berendezések, szenzorok, anyagátviteli eszközök az anyag mozgatásához – biztonságosan működhetnek úgy, hogy az emberre nem jelentenek veszélyt. Ezzel ellentétben a kollaboratív robotoknál pedig a biztonságot a számos szenzoron túl a magasabb szintű biztonsági előírásokkal biztosítják a gyártók. 1990 és 2000 között az információ beszerzés és számítástechnika tömeges adaptációjának köszönhetően megjelentek az automata rendszerek melyek visszacsatolással rendelkeztek és ezzel fellendítették az ipari robotok fejlődését. Ebben az időben jelentek meg a kulcsfontosságú technológiák az erő- és nyomatér-szenzor, látórendszer, Ethernet, beágyazott és valós idejű rendszerek. Mindezek következtében a kollaboratív robot fejlődésnek indult és az

ember közvetlen környezetében dolgozva azt az olyan feladatoknál melyek teljesen nem automatizálhatók kiegészíti és emellett helyet és pénzt spórol a gyár mivel nem kell munkacellába zárni a robotot. [15]

Összehasonlítva a hagyományos ipari robotkarokkal, a kollaboratív robotkarok nagyobb szintű rugalmassággal rendelkeznek. Az integrált hajtásrendszer a kollaboratív robotok tervezése és fejlesztése során vált elterjedtté. Az integrált hajtás figyelembe veszi a mechanikai alkatrészeket, így a motort, a hajtóművet, a fékeket, jeladókat és az elektromos áramköröket együtt és az összeszerelése ezeknek a megfelelően módon következik be. [16]

A szenzorok a megoldás arra, hogy a robotok és a gépek képesek legyenek érzékelni az őket körülvevő környezetet. Általánosságban elmondható, hogy az érzékelők típusaiknak és számának növelésével növelni lehet az alkalmazkodóképességét a robotnak. A több forrású szenzoros jelek biztosítják a lehetőséget, hogy javuljon a mozgási-, észlelési-, és kognitív intelligencia. Más technológiák, mint például az IoT (Internet of Things), kiber-fizikai rendszerek, big data analízis, nem működhetnek a valós idejű szenzor információk nélkül. A szenzorok melyek a kollaboratív és intelligens robotok szerves részét képezik a következők: erő/nyomaték szenzor, tapintásérzékelő, látásérzékelő, ütközés érzékelő szenzor, biztonsági szenzorok, közelségérzékelő, tárgyérzékelő szenzorok. Szenzorfüziónal megoldható az, hogy az egyes érzékelők funkcióit összehangolva, azoknak a pozitív tulajdonságait ötvözzük és ezzel nagyobb kihívást jelentő bonyolultabb feladatokat is meg tudunk oldani velük. [16]

Általában a kollaboratív robotok a legtöbb kommunikációs protokollal kompatibilisek, beleértve a következőkkel: Ethernet/IP, TCP/IP, Modbus, CAN bus, Profinet, és MQTT. Valójában az Ethernet és más ipari hálózatok gyorsan cserélődnek a vezetékes robotalkalmazásokban. A vezetékezési költségek megtakarítása önmagában gyakran indokolja az ipari hálózatok használatát, míg a további előnyök, mint a diagnosztika biztosítása, a vezetékes és vezeték nélküli megoldások együttélése, valamint a csatlakozás egyszerűsége a vállalathoz és a távoli kapcsolódás lehetősége a felhasználók számára mind további motivációt jelent a robotikai alkalmazások fejlesztői számára, hogy adaptálják az Ethernet alapú ipari protokollokat. [16]

2.3. Bin picking

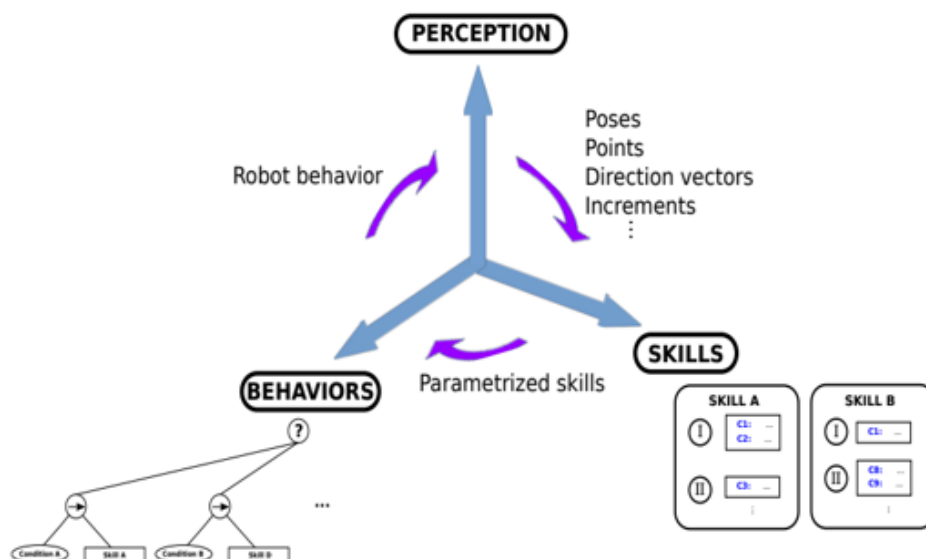
A robotok bevezetése óta a gyártás területén és az automatizált termelési eszközök fejlődésében mindig ott volt a vágy a teljes automatizálásra és az intelligencia bevezetésére a gyártási eljárások többségébe. A negyedik ipari forradalom tovább fokozta ezt a vágyat az intelligencia iránt és a tanulási képességek fejlesztésére a kollaboratív robotokban. A

kollaboratív robotok intelligenciájának a fő bizonyítéka és az egyik legnehezebb probléma, ami megoldásra vár az ipari automatizáció területén az a véletlenszerűen elhelyezett, struktúrátlan alkatrészeknek a pontos észlelése és manipulálása. Különösképp az olyan esetekben mikor a különböző részek változatosak és többféle megfogási technikával kell megoldanunk a feladatot, ami még nagyobb kihívást jelent. [17]

Nagy mennyiségű kutatás és kísérlet történt az iparban történő alkatrész bin picking területén. A Tuan-Tang Le és Chyi Yeu Lin által kifejlesztett rendszerek több mint 99%-os sikerarányt érnek el azáltal, hogy szemantikai szegmentálást és mélységi adatokat használnak fel. Azonban ők csak pneumatikus fejeket használnak a feladat végrehajtására, ami korlátozza a rendszer képességeit. [18, 19]

Egy másik, sokoldalúbb rendszert a 2017-es Amazon Robotics Challenge során lett kifejlesztve Andy Zeng által. Ez a rendszer egy multifunkcionális grippert használ, ami lehetővé teszi, hogy a robotkar pontosabb megfogást hajtson végre. Ráadásul ez egy eléggé gyors rendszer, $0.05 * n$ másodperc feldolgozási sebességgel rendelkezik, ahol n az a lehetséges megfogási szögek számát jelöli. A rendszer nagy sebességének kárára azonban kisebb, 89%-os pontosságot tudtak csak elérni. A rendszer deep learning használatával elemzi ki azt, hogy milyen megfogási módot válasszon a siker érdekében. [20]

Amint azt az 1. ábrán láthatjuk a teljes mozgásgeneráló architektúra három részből áll. Ezek a következők: érzékelés, készségek és viselkedés. [56]



1. ábra Mozgásgeneráló architektúra felépítése [56]

Az emberi kéz hatékonyságának és rugalmasságának reprodukálása a tárgymanipulációs feladatoknál fontos kihívás. Ehhez alapvetően újra kell gondolni, hogy hogyan lehet kihasználni a számos szenzorból érkező adatainkat és a rendelkezésre álló robotkarnak a mechanikus kezűgyességét. Az emberhez vagy főemlőshöz képest a mai robotmegfogók és kezek kezűgyessége rendkívül korlátozott. Az érzékelést négy kategóriába sorolják, mint az érzékelő nélküli manipuláció, a képérzékelés, az aktív érzékelés és az interaktív érzékelés. Az érzékelő nélküli manipuláció bár egyszerűbb feladatoknál működhet, azonban összetetteknél már sokszor nem elegendő. [21, 22]

A vizuális észlelési megközelítésekben statikus képeket használnak arra, hogy a manipulációs feladatokhoz szükséges érzékelő visszajelzéseket létrehozzák, képeket használnak a fogástípusok és tárgytipusok közötti kapcsolat megértéséhez. [23]

Az aktív észlelési megközelítésekben az érzékelő visszacsatolását úgy oldják meg, hogy az emberi figyelmet és tekintetet utánozzák. Ebben a környezetben egy RGB kamera van használva. Egyes esetekben úgy használják ezt a módszert, hogy a robot mozgásával a kamerát a jelenet körül mozgatják és így szegmentálják a felvételt és találják meg a tárgyakat a képen. Összehasonlítva a statikus kamerabeállítással, ez a módszer lehetővé teszi az objektumok jobb modellezését a több nézőpontból származó adatok által. [24]

Az interaktív érzékelési megközelítéseket a fizikai interakció és a hagyományos érzékelési módszerek kombinálására fejlesztették ki. E kettő kombinációja szélesebb körű alkalmazásokat tesz lehetővé, például ismeretlen tárgyak manipulálásának megtanulását tárgytulajdonságok tanulását. Ha az érzékelés által a szükséges paraméterekhez hozzájutunk akkor a következő lépés jön. [56]

2.4. Gépi látás és képfeldolgozás

Az Automated Imaging Association (AIA) a világ legnagyobb gépi látással foglalkozó kereskedelmi egyesülete, amely több, mint 350 taggal rendelkezik, köztük rendszerintegrátorokkal, gépi látás komponens gyártókkal, szoftverfejlesztőkkel és forgalmazókkal. Az AIA szerint a gépi látás magában foglal minden olyan ipari és nem ipari alkalmazást, amelyben az alkalmazott hardver és szoftver kombinációja alkotta rendszer képek rögzítésén és feldolgozásán alapuló funkciókat hajt végre. A gépi látás rendszerek fejlesztése több tudomány- és szakterületet érintő ismereteket és a potenciális alkalmazási területek azonosítását követeli meg. Bár a gépi látás rendszerek különféle technológiákat alkalmazhatnak, jellemzően hasonló lépéseket követnek a képrögzítéstől a beavatkozásig. Ezek a lépések tipikusan az optikai adatgyűjtés a jelenet (angolul „scene”) korlátozásainak

figyelembevételével, az előfeldolgozás, a szegmentálás, a vizsgált jellemző, mint képi információ kinyerése, az osztályozás és/vagy képelemzés, majd végül a döntéshozatal és beavatkozás. A kamerák és feldolgozóegységek legújabb technológiai fejlesztései, valamint a hardverköltések csökkenése felgyorsíthatják a gépi látás alkalmazását az iparban, sok vállalatnak azonban továbbra is korlátozott hozzáférése van a gépi látással kapcsolatos ismeretekhez. Az oktatásban a témakör elméleti alapismereteinek összefoglalása mellett nagy hangsúlyt szükséges fektetni a gépi látás alkalmazások megvalósításának gyakorlati oldalára, a megfelelő megvilágítási technikával támogatott képi adatgyűjtéstől az irányítástechnikai rendszer által végrehajtott beavatkozási műveletig. A gépi látás rendszerek a képrögzítéshez az ipari kamerákon belül védett digitális szenzorokra támaszkodnak, speciális optikával, így a számítógépes hardver és szoftver képes feldolgozni, elemezni és mérni a különböző jellemzőket a döntéshozatalhoz. A gépi látás egyre széleskörűbb felhasználása figyelhető meg mind az iparban, mind az oktatási, kormányzati vagy katonai alkalmazásokban. A negyedik ipari forradalomban az intelligens gyártási folyamatok megjelenésével egyre növekszik az igény a gyártási folyamatok professzionális vizsgálatára és a minőségi adatok iránt, ezért válik egyre fontosabbá a gépi látás szerepe az ipar számos területén. A gépi látáson alapuló alkalmazások köre jelentősen megnőtt az elmúlt években, erre egy példa a 2D képfeldolgozástól a háromdimenziós rendszerekre történő fejlődés. A gépjárműgyártásban a 3D rendszerek összetett geometriákat, például öntvényeket vizsgálnak. Az optikai rendszerek nemcsak a legkisebb hibákat észlelik, hanem értékes adatokat szolgáltatnak, amelyek a gyártási folyamat optimalizálására használhatók fel. A gépi látás rendszerek egyre gyakoribb alkalmazása tapasztalható a gyártásban, különösképpen a minőségellenőrzés területén, annak megbízhatósága, gyorsasága, pontossága és ismételhetősége végett. A negyedik ipari forradalomban támasztott irányelvek szerint a gyártás hatékonysága az adatok következetes gyűjtése- és feldolgozása által növelhető, melyben fontos szerepet kapnak a korszerű automatizálási technológiákkal támogatott intelligens gyártási folyamatok. [25, 26, 27]

Az automatizált látórendszereket széles körben alkalmazzák különböző területeken, mint például a mezőgazdaság, az ipar, a földrajzi térképek stb. A robotizált látásrendszer egyre fontosabbá válik azokban az automatizált iparágakban, amelyeknek rugalmas módszerekre van szüksége a képfeldolgozáshoz és a tárgyak megkülönböztetéséhez. A robotizált látásrendszer azonosítja az objektumok helyét és rögzíti a képeket, tehát a robotot egy 3D látóképességgel ruházza fel, amely ezáltal megoldást tud nyújtani a bin-picking problémákra, melyet számos objektumra vonatkozóan ellenőriztek. [28, 29]

A tanulási algoritmus az objektumról egy 3D-s modellt próbál előállítani és megkísérel azonosítani néhány pontot két vagy több képen, annak érdekében, hogy képes legyen a tárgyat jól megfogni. A robotos látásrendszer nagyon széles alkalmazási területekkel rendelkezik, amelyek célirányos érzékelést, objektum- és helyszínmodellezést, robot lokalizációt, térképezést, navigációt, útvonaltervezést, feltárást, felderítést, megfigyelést, nyomkövetést, keresést, felismerést, ellenőrzést, robotmanipulációt, összeszerelést és szétszerelést képesek elérni. Egy látórendszer lekéri és elemzi a képeket, hogy észlelje a tárgyat, és elküldje annak koordinátáit a robotkar-vezérlésnek. A látórendszer mélységinformációk kinyerését, a kívánt objektumok szegmentálását, tárgyfelismerést, nyomkövetést és pózbecslést biztosít számos algoritmus, egy folyamatba történő integrálásával. [30, 31, 32, 33]

2.4.1. Tárgy detektálása, súlypontjának és színjellemzőinek kiszámolása

A tárgy alakjának megkülönböztetésére kidolgozott megközelítés a tárgy alakjának digitális képből történő meghatározásán alapul, amely során a pozíciójának megtalálása a cél. A későbbiekben a tárgy alakja alapján a robot akár azonosíthat egy megfelelő szállítószalagot is, ahova a terméket helyezni szükséges. A természetes objektumok kültéri környezetben történő felismerésének három fontos része van, melyek a kültéri fényviszonyok figyelembevétele, a színszegmentálást és a karakterizálási javaslatok. Az objektumok észlelése és azonosítása a megjelenésük és elhelyezkedésük alapján történik. Objektumkövetés történhet videó alapú megfigyeléssel is, ahol a kamera feladata olyan nyomkövetési stratégiákon alapul, mint a régió alapú és az aktív kontúr alapú. Az objektum alapú képelemzést olyan földrajzi információs rendszerekben alkalmazzák, amelyek képesek pixeleket úgy elemezni, hogy a képekből könnyen használható objektumok kerülnek letárolásra. Fényintenzitás-elemzés felületdetektálásnál alkalmazható. Az objektuminformációk automatikusan az objektumról készített képből vannak származtatva, például az objektum lokalizációjának felismeréséhez, míg a megfogáshoz az objektum súlypontja kerül felhasználásra. [34, 35, 36, 37, 38, 39, 40]

Az alakzatfelismerés éldetektáláson alapul, amely egy kétváltozós exponenciális függvény deriválása által kapható meg, melyet eredetileg Ben-Arie és Rao kettős vezetett le. Az elméletben történő alakfelismerés során egy, az adott alakzat kontúrját befoglaló úgynevezett határoló doboz (R) keresése történik meg. Ahol az R alakja:

$$R = (u_{\min} * v_{\max} * v_{\min} * u_{\max}) \quad (1)$$

ahol 'u' a pixelek sora és 'v' a pixelek oszlopa. A határolódoboz a látható alakra korlátozódik, majd később ezáltal könnyen megkereshető az adott objektum súlypontja, ami a

következőképpen számítható:

$$C = (\bar{x}, \bar{y}) \quad (2)$$

ahol C az alakzat súlypontja, míg $\bar{x}, \bar{y}$ a következőképpen számolható:

$$\left. \begin{aligned} \bar{x} &= \frac{1}{|R|} \sum_{(u,v) \in R} u \\ \bar{y} &= \frac{1}{|R|} \sum_{(u,v) \in R} v \end{aligned} \right\} \quad (3)$$

A képfeldolgozás tudományában a képfeldolgozás a jelfeldolgozás olyasfajta formája, amelynél a bemenet egy kép, például fénykép vagy videóból vett képkocka; a képfeldolgozás kimenete lehet egy kép vagy a képhez kapcsolódó jellemzők vagy paraméterek halmaza. A legtöbb képfeldolgozási technika a képet kétdimenziós jelként kezeli, és szabványos jelfeldolgozási technikákat alkalmazza rajta. A képfeldolgozás a 2D kép számítógépes feldolgozását jelenti. A képet a „valós világban” kétváltozós függvénynek tekintjük, amely megadja bármely objektum valós koordinátaezredékeit (x, y) . Az RGB színmodell a számítógépes grafikus rendszerek standard kialakítása, de nem minden alkalmazásra ideális. Emiatt az elkészült képet RGB színrendszerről a következő módszerrel néhány esetben Gray-re szükséges átkonvertálni:

$$Y = \text{Avg}(R, G, B) = \frac{R + G + B}{3} \quad (4)$$

A tárgyak szín szerinti megkülönböztetése a HSV (színárnyalat, telítettség, érték) módban, míg a szürke binárisban a pozíciókoordináták megtalálása megy egyszerűbben. Színszegmentáció olyan módszer, amelyet olyan digitális képek elemzésére használnak, amelyek a kép egy pixelének például vörösségi értékét jelzik, összehasonlítva a szükséges, előírt vörösségi értékkel. A színeskorrekció többnézetű kép- és videó esetén kilenc színeskorrekciós algoritmus használatával automatikusan történik. A digitális kép régióinak azonosítására és osztályozására szolgáló módszerek olyan azonosítási információkon alapulnak, amelyek színes és nem színes régiókat különítenek el, továbbá az RGB színteret 3D polárkoordinátákban árnyalat, telítettség, érték (fényerő) szerint osztályozzák. A szín szerinti rendezés hasonló az objektum alakja szerint történő rendezési művelethez. [41, 42, 43, 44, 45]

2.4.2. Képfeldolgozás eszközei

A gépi látáson alapuló rendszerek tervezési módszertana már számos tanulmányban összefoglalásra került. Egy általunk vizsgált forrásban a gépi látás rendszer első tervezési lépése a jelenet vizsgálata és az alapján a megfelelő megvilágítási technika és optikai rendszer kiválasztása. A második lépésben az optikai adatgyűjtés történik a (jellemzően CCD vagy

CMOS) képérzékelőkkel a megvilágított területről, a harmadik lépés a digitalizálás, mely során a képrögzítés és -megjelenítés kerül elvégzésre. Az utolsó lépés a képfeldolgozás, amely során olyan módosításokat hajtodnak végre a digitális kép pixelértékein, hogy megfelelő forma álljon rendelkezésre a későbbi műveletekhez. A tervezési módszerünk első lépése a jelenet vizsgálata és a rendszer környezetének előkészítése, azonban a megvilágítás tervezését csupán az optikai paraméterek meghatározása és a képrögzítő eszköz kiválasztása után tartjuk célszerűnek. Ennek az oka, hogy a képrögzítés módja és az optikai rendszer fizikai kialakítása nagymértékben befolyásolhatja az alkalmazott megvilágítási technika hatékonyságát. Az adatgyűjtés és megvilágítás tervezése után fontos szempont, hogy az optikai adatok digitalizációja és továbbítása hogyan, milyen interfészen keresztül történik a feldolgozóegység felé. Az utolsó lépés a képfeldolgozásért és egyéb irányítástechnikai feladatokért felelős szoftver tervezése és az ezt végrehajtó vezérlőegység kiválasztása, figyelembe véve a kiválasztott képrögzítő eszköz és a rendszert felépítő további eszközök jellemzőit és a vezérléssel szemben támasztott követelményeket. A következő alfejezetekben a gépi látás rendszerek tervezéséhez szükséges alapismereteket foglaljuk össze. A rendszerünk meghatározó eleme a kamera, melynek kiválasztásánál számos szempontot kell figyelembe vennünk. Elsőként az optikai rendszer paramétereit sorra véve meghatározzuk a szükséges szenzorméretet és felbontást, a munkadarab távolságának függvényében a megfelelő fókusztávolságot, majd ezután az alkalmazáshoz mérten a képrögzítés módját, végül pedig megvizsgálva a lehetséges kamera típusokat és interfészeket, kiválasztjuk az alkalmazásunkhoz legjobban illeszkedő eszköztípust. A kameraszensorok kétdimenziós pixeltömböket tartalmaznak, melyek a fény intenzitását érzékelik, ezek a mérési adatok pedig sorosan vagy párhuzamosan egy kimeneti regiszteren keresztül továbbíthatók a feldolgozóegység felé. A szenzor felbontását a vizsgált objektum legkisebb jellemzőjének mérete és a látómező segítségével határozhatjuk meg. A fókusztávolság (f) az objektív jellemző paramétere, a nagyítás mértékének meghatározására használt mérőszám, melyre a munkatávolság, a látómező és a szenzorméret is hatással van. Mivel a lencsék csak bizonyos szabványos fókusztávolságokkal elérhetőek (pl. 6 mm, 9 mm, 12,5 mm), ezért az eszköz kiválasztását követően a munkatávolságot úgy kell megválasztani, hogy a vizsgált munkadarab fókuszbán legyen. [47, 48]

Alapvetően két technológiát különböztethetünk meg, az *area scan* és *line scan* kamerákat. Az *area scan* kamerák egy időben egy pixelterületet rögzítenek, a *line scan* kamerák esetén a képrögzítés sorról sorra történik, ezáltal egy gyorsabb adatgyűjtést biztosítva. Ez a módszer további feldolgozási feladatokat követel meg a rögzített képpontsorok egymáshoz illesztéséhez, a kép felépítéséhez. A *line scan* kamerák olyan alkalmazásokban előnyösek, ahol a vizsgált

objektum mozgást végez, például egy szállítószalagon. A képrögzítési módszert tehát az adott alkalmazás és a jelenet tulajdonságai szerint választjuk ki. Azért célszerű a tervezés első lépéseiben meghatározni, mert a világítási technika tervezésénél a kép rögzítési módszer egy alapvető szempont. *Area scan* kamerák megvilágításánál a világítóegység geometriai kialakítása is fontos szempont, *line scan* kameráknál a fényforrás fotometriai tulajdonságaira kerül nagyobb hangsúly. A képrögzítő eszköz kiválasztása után alaposan meg kell fontolnunk, hogy milyen fényforrást és megvilágítási technikát alkalmazzunk a rendszerben. A megfelelő minőségű világítás biztosítása érdekében figyelembe kell venni a fény hullámhosszát, terjedési szögét, intenzitását, meg kell terveznünk a világítóegység kialakítását és elhelyezkedését a kamerához képest. A megfelelő világítás a kamerás vizsgálatok egyik nélkülözhetetlen eleme, inhomogenitása nagy mértékben befolyásolja a gépi látást alkalmazó vizsgálati módszer pontosságát és robusztusságát. Egyre több tudományos vizsgálat és alkalmazás fókuszában áll a megvilágítás, amellyel speciális mérési feladatok is elvégezhetők. Például világítás kompenzációs technikákat használnak a talaj felületi érdességének meghatározására. A gépi látás alkalmazásokban leggyakrabban fluoreszcens, fémhalogén, LED, kvarc halogén és xenon fényforrásokat használnak. Kisebb méreteknél jellemzően fluoreszcens, LED és kvarchalogén fényeket, nagyméretű alkalmazásoknál, vagy ahol nagyon erős fényforrásra van szükség, xenont és fémhalogént használnak. Kamerás vizsgálatoknál a fény spektrális összetételét is figyelembe kell vennünk a tervezés során, össze kell hangolnunk a fényforrás kimeneti spektrumát a kamera spektrális érzékenységeivel. A mikroszkópos vizsgálatoknál gyakran egy teljes spektrumú kvarc halogén vagy xenon fényforrást használnak, különösen színes képeken. A monokróm LED fényforrás megfelelő fekete fehér CCD kameránál és színes alkalmazásokban is. Ahhoz, hogy optimális világítást alakítsunk ki, figyelembe kell venni a vizsgálat közvetlen környezetét, majd a vizsgált minta és a fény közötti kölcsönhatásokat. A környezeti fény befolyásolhatja a vizsgálat eredményét, ezért ki kell küszöbölnünk villanófénnyel, szűrővel vagy burkolattal. A minta és a fény kölcsönhatását a minta felületének alakja, a fényforrás elhelyezkedése, a minta és a fény színe is befolyásolja. A különböző anyagok a fény különböző hullámhosszait más-más módon tükrözik vissza vagy nyelik el. Míg a felület színével azonos színű fények a felületet világosabbá teszik, addig az eltérő színek elnyelődnek és sötétebbé teszik azt. A megfelelő színkeveréssel tehát kontrasztot tudunk képezni a háttér és a vizsgálni kívánt felület között. Egy tanulmányban ismertetett vizsgálatok szerint a felületi érdesség, a fényesség és a felület színe jelentősen befolyásolják a színállandóságot. A fényes és sima felületek nagyobb színállandósággal rendelkeznek, mint az érdes és matt felületek. A megvilágítás változása vagy az objektum elmozdítása a kamerához

képest befolyásolni fogja az objektum színeit és következtetésképp annak felismerését. A megvilágítás és a kamera felbontása tehát igazán kritikus tényező, hiszen ezektől jelentős mértékben függ az alkatrészek valódi határainak megtalálása. Abban az esetben, ha a fényforrás nem a megfelelő helyen van, szóval ha éppen pont olyan magasan vagy alacsonyan, akkor az, egynél több koordináta hozzáadását eredményezheti - a fény hulláma miatt, amely visszaverődik a tárgyról. Viszont kizárólag egy koordináta kell, hogy a robotkar az alkatrész megfelelő helyéhez mozogjon. Ez a probléma kiküszöbölhető nagy felbontású kamera használatával, illetve alacsony frekvenciájú megvilágítással is. [49, 50, 51, 52]

De van egy másik probléma is a valós idejű működésnél, ami nem más mint, hogy a robotkart úgy kell programozni, hogy az a megfelelő időben válassza ki az objektumot, amikor az a megfelelő pozíciótartományba kerül. Erre több számítás is elvégzésre kerülhet a jobb eredmények elérése érdekében. Továbbá, a kamera elhelyezését interaktívan kell kialakítani a tárgy és a robotkéz közötti távolság leküzdéséhez szükséges idő alapján. Ezáltal válik elérhetővé, hogy ne történjen megszakítás a képfeldolgozás és szortírozási művelet között. A rendszer interaktívan operál a rendezés során az objektumok típusa alapján (szín vagy forma). Érdekes fejlesztési lépésként megközelítések hatékony tényezőit is megvizsgálni, hogy a folyamat időtartama minimális szintre legyen csökkenthető. [53]

2.5. Gyakorlati alkalmazási példák

A negyedik ipari forradalomban egyre jelentősebb szerepet kap a digitalizált gyártáshoz szükséges kompetenciák oktatása. Ehhez megfelelő környezetet biztosítanak a mintagyárak, amelyek a jövőbeli munkaerő számára, a digitalizált termelési környezethez és a kiber-fizikai rendszerekhez kapcsolódó technológiák alkalmazásához és integrálásához szükséges készségek fejlesztését teszik lehetővé. Sőt, ennek a koncepciónak számtalan valós alkalmazása létezik. Ez a tudományos közösség számára népszerű kutatási téma, ennek köszönhetően sok információ áll rendelkezésre bárki számára. Ezen szemlélet alapján néhány gyakorlati példa ismertetése történik meg: [46, 54]

- a) Egyik ilyen alkalmazási terület például: a valós idejű automatizált gyümölcsválogatás, melyet látásrendszer segítségével valósít meg a robot. Kifejlesztésre került egy robotizált látórendszer a gyümölcsök automatikus, roncsolásmentes ellenőrzésére és átadására, a foltok észlelésére és a gyümölcs minőségének értékelésére. A látórendszerrel rendelkező gyümölcsszedő robot a gyümölcs méretének és helyzetének becslésére a színpixelek statisztikai osztályozását alkalmazza. Az automatikus narancsbetakarítást megvalósító robot infravörös lézeres távolságmérő érzékelőt is használ; a látórendszer kimenete adja a

3D pozíciót, a sugár által pedig az egyes gömb alakú objektumok felületi reflektivitásáról lehet információt kapni. Egy robotizált látórendszert lehetőség van úgy is beprogramozni, hogy a megkülönböztetéshez különféle fényviszonyok között is felismerje az objektumokat és statisztikai adatok alapján kiszámítsa azok méretét és helyzetét. A robot látórendszert egy nagy teljesítményű neurális multi-osztályozó segítségével fejlesztették ki az egyes objektumok pontos helyzetének és orientációjának azonosítására. A tárgymegfogás az érzékelésen és vezérlésen alapul, annak érdekében, hogy képes legyen észlelni, figyelni és manipulálni a környezetében lévő tárgyakat. A rendszerkamera a fő szállítószalag keretére van rögzítve, míg a válogató robot a képkészítés utáni pozícióban található. Ez a távolság elég időt biztosít a képfeldolgozáshoz, és ahhoz, hogy azáltal a robot a megfelelő döntést meg tudja hozni. Amikor a termék megérkezik a megfogási pozícióba, a szállítószalag hevederei közé rögzített ultrahangos érzékelő jelzi a gépnek, hogy végrehajtható a megfogási művelet. A robot ekkor felveszi a tárgyat, majd a szállítószalagra helyezi. A szín szerinti rendezés hasonló az objektum alakja szerint történő rendezési művelethez, ez a szortírozási folyamat valós időben ment végbe, az objektum fő szállítószalagra történő mozgatásával. [53]

- b, A gépi látáson alapuló megfogási módszert széles körben alkalmazzák az iparban, és referenciaként szolgálhat mezőgazdasági termékek automatikus válogatására. Egy mezőgazdasági célra készült automatikus válogató berendezés kerül bemutatásra. Az automata válogató berendezés egyszerű felépítésű, megbízható teljesítményű és alacsony költséggel bír, és meg tudja valósítani a pozicionálást, osztályozást és a céltárgyak megfogását. Jelenleg Kínában a mezőgazdasági termékek válogatása elsősorban a kézi munkára támaszkodik, amely alacsony hatásfokot eredményez, a megfelelő automata berendezések fejlesztése pedig elmarad. Épp emiatt került kifejlesztésre egy, a mezőgazdasági termékeket automata módon szortírozó, vizuális szervót használó tesztplatform. Célosztályozási, pozicionálási és válogatási tesztek kerültek futtatásra, melyek során paradicsom és narancs volt használva vizsgálati tárgyként. A folyamat célja az volt, hogy a rendszer képes legyen többféle gyümölcs előre meghatározott helyre történő mozgatására. A vizsgálati eredmények azt mutatják, hogy a sikerességi arány: a célpozicionálásé közel 98%, a célbesorolásé közel 98%, a megfogásé közel 95%. Továbbá egyetlen célobjektum rendezési ideje akár 1 másodperc is lehet, ami képes megfelelni a közönséges mezőgazdasági termékek automatikus válogatási követelményeinek. A szerkezet és a folyamatban javasolt algoritmusok egyszerűek, megbízhatóak és rendkívül hatékonyak, így könnyen megvalósítható a technológiai átültetésük. [55]

3. Szortírozási folyamat fejlesztése

A fejlesztés célja egy olyan automata folyamat létrehozása kollaboratív munkakörnyezetben, ahol a demonstrációs eszköz inhomogén munkadarabok felismerését és szortírozását végzi. A felhasználó tetszőleges sorrendben bepakolja az előre betanított formájú munkadarabokat a munkaterületre. A rendszer a digitális képfeldolgozás segítségével elvégzi a munkadarabok azonosítását, kategorizálását és az előzetesen definiált, de dinamikusan változtatható paraméterek (munkadarab kontúrhoz és színhez rendelt tároló) függvényében robot programot generál. Majd a munkadarabok bepakolásának sorrendjében elvégzi a szortírozást.

3.1. Munkakörnyezet ismertetése

A fejlesztés megvalósítási helyszíne a Széchenyi István Egyetem Kiber-Fizikai Gyártórendszerek Laboratóriuma, ami „Az „Ipar 4.0 kutatási és innovációs kiválósági központ” projekt keretein belül jött létre. A laboratórium fő tevékenységi köre az alábbi feladatcsoportokba sorolható:

- Szituáció-tudatos, erőforrás-hatékony és robusztus termelésstervezés és irányítás
- I4.0 megoldásokat használó, demonstrációs mini gyártó- és logisztikai rendszer
- Kooperatív és adaptív termelési és logisztikai hálózatok tervezése és működtetése
- Robusztus kooperatív kontrol kiber-fizikai rendszerekben
- I4.0 megoldások az energia-hatékony és fenntartható gyártás támogatására

A kutatások célja a hasznosítható eredmények elérése az ipar és az egyetemi oktatás számára.[57]



2. ábra Laboratóriumban rendelkezésre álló munkaasztal

A felhasznált mintakörnyezet, ami a laboratóriumon belül az egyik asztalon van a 2. ábrán látható, az előtérben a robottal. Alatta a fekete munkatér az, ami háttérként szolgál a képvételezéshez. Rajta pedig láthatóak azok a munkadarabok, amiket a kamera képe alapján felismerve a robotkar bepakolást követően szortíroz a munkatérből egy adott pozícióba.

3.1.1. Robot

Az alkalmazásra kerülő, 3.ábrán látható eszköz a dán Universal Robots cég UR sorozatába tartozik. Ebben a kategóriában 4 típus lelhető fel, melyek terhelhetőségük és munkatérük méretében térnek el jelentősen. Az UR5 egy kisméretű együttműködő asztali robot, mely tökéletesen alkalmazható könnyebb összeszerelési feladatokra és automatizált munkapadi műveletekhez. A kompakt asztali kobot mindössze 18,4 kg-ot nyom, de 5 kg-os terhelhetőséggel bír. Minden csuklós csatlakozásán 360 fokos, az utolsó csatlakozásnál pedig végtelen forgástartománnyal rendelkeznek. A termék 850 mm hatósugarú működésre képes. Mindezeknek az egyedi funkcióknak köszönhetően ez a legrugalmasabb könnyűsúlyú eszköz a piacon, mely az emberi munkaerő közvetlen közelében dolgozhat. [58]



3. ábra UR5 robot, vezérlő és TP

3.1.2. Megfogó

A robotkarra egy, a 4. ábrán látható Robotiq 2F-85 típusú kétujjas megfogó van rögzítve. Fontosabb technikai adatai az 1. táblázatban olvashatóak. Sajátos kialakításával lehetőséget biztosít eltérő formájú és méretű elemek megfogására. [59]

Jellemző	Érték
Fogási tartomány	0 - 85 mm
Fogóerő	20 – 235 N
Hasznos teher	5 kg
Pozíció felbontás	0,4 mm
Tömeg	0,9 kg

1. táblázat A RobotIQ 2F-85 megfogó technikai adatai [58]

„Plug + play funkcióval” is rendelkezik, amely rendkívül egyszerűvé teszi a robotra való felhelyezését, nincs szükség átalakítókra. Ennek köszönhetően az esetleges későbbi fejlesztések során, olyan alkatrészek is bevonhatók a műveletbe, melyek alacsony méreteik miatt csak egy másik, például vákuumos megfogóval mozgathatóak.



4. ábra RobotIQ 2F-85 megfogó [59]

3.1.3. Kamera

A folyamat egyik legérzékenyebb része a megfelelő felvétel készítése az adott munkaterületről, ahol az alkatrészek véletlenszerűen elszórva helyezkedhetnek el. A 2. táblázatban megtekinthető milyen tulajdonságokkal rendelkezik a választott Intel RealSense D435 kamera [60].

Jellemzők	RealSense D435
Max. felbontás (RGB)	1920 x 1080
Pixelméret	1,4 μm
Képfrissítés (max. felbontáson)	30 fps
Látószög (H x V)	69° x 42°
Interfész	USB 3.1

2. táblázat RealSense D435 technikai adatai

Az 5.ábrán látható eszköz mélységi fotók készítésére is alkalmas, de a dolgozathoz csak RGB képek készültek vele. Módszeres kísérletezések arra vezettek, hogy a fényerősség és a kontraszt értékének csökkentésével érhető el a képfeldolgozáshoz legalkalmasabb felvétel. Ezen beállításokkal a képen jelentősen csökkenthető a zaj, és a háttér gyenge vagy nem teljesen egyenletes megvilágításból származó zavar.



5. ábra Intel RealSense D435 kamera

3.1.4. Szoftver

A fő szoftver amit használtunk a munka során az a Python. Maga a Python egy általános célú magas szintű programozási nyelv, ami nyílt forráskódú (open source) fájl így bárki le tudja tölteni a saját gépére. A Python egy *interpreter* nyelv, ami annyit tesz, hogy a program futása előtt egy gyors előfeldolgozás van és a program futása közben értelmezi folyamatosan az utasításokat. Pontosan emiatt rugalmasabb, ámde lassabb egy *compiler* nyelvnél, ami futtatás

előtt az egész kódot beolvassa és gépi kódra átalakítva futás közben már a lefordított kódot olvassa. Ennek kiküszöbölésére szolgál, hogy a képeket beolvasásnál nem több ciklusban olvassuk be, soronként és oszloponként hanem *NumPy* tömbkezelést használva vektorizálást hajtunk végre logikai operátorok segítségével. Több felület is létezik, ahol tudunk Pythonban programozni, néhány a legismertebbek közül: PyCharm, Jupyter Notebook, IDLE. Mi a program futtatására az IDLE-t használtuk, ami egy integrált fejlesztői és tanulási környezet. Lehetővé teszi a kódok egyszerű megírását és futtatását, ezenkívül támogatja a Python szkriptek létrehozását, módosítását és végrehajtását is.

A Pythonban a beépített függvényeken túl különböző könyvtárak vannak melyekben előre megírt programokat találunk. Ezek a modulgyűjtemények tartalmazhatnak különböző osztályokat, utasításokat vagy éppen függvényeket, amiket meghívva (beimportálva) szabadon használhatunk. Az általam képfeldolgozáshoz használt könyvtárak: [61]

- Numpy: Többdimenziós, nagy mátrixoknak a kezeléséhez nyújt segítséget magasszintű matematikai funkciókkal támogatva azok használatát. A feldolgozási sebességet nagyban segíti, hiszen vektorizált tömb műveleteket tesz elérhetővé.
- cv2: Opencv-python modul, a számítógépes látásban, vagyis a képfeldolgozásban van segítségünkre. Rengeteg hasznos függvény van benne, képek beolvasásától a mentésükig, ablakok deklarálására is használjuk. Ezenkívül szűrések, kontúrkeresés és meghatározás esetén is ehhez a modulhoz fordulunk segítségért.
- os: Biztosítja számunkra a kapcsolatot az operációs rendszerrel. Gépünkön lévő könyvtárak, fájlok használatában segít nekünk.
- socket: Alacsony szintű hálózati felület. Ezzel adatokat küldeni és fogadni bármikor képesek vagyunk.
- ur-rtde: Egy Python felület ahhoz, hogy tudjunk adni és lekérdezni adatokat a robot RTDE (Real Time Data Exchange) felületének segítségével.
- matplotlib pyplot: Képek megjelenítésében és grafikai feladatokban van segítségünkre ez a modul.
- time: Ez a könyvtár alacsony szintű időkezelésre ad eszközöket, pl. időtartammérés vagy várakozás.
- math: Ez a könyvtár segítséget ad matematikai műveletek elvégzésében. Különböző függvényeket, konstansokat tartalmaz.

Több más általunk definiált ciklust hívunk még meg a program közben, ezekre az általunk meghatározott név alapján tudunk hivatkozni, amin elmentettük azokat.

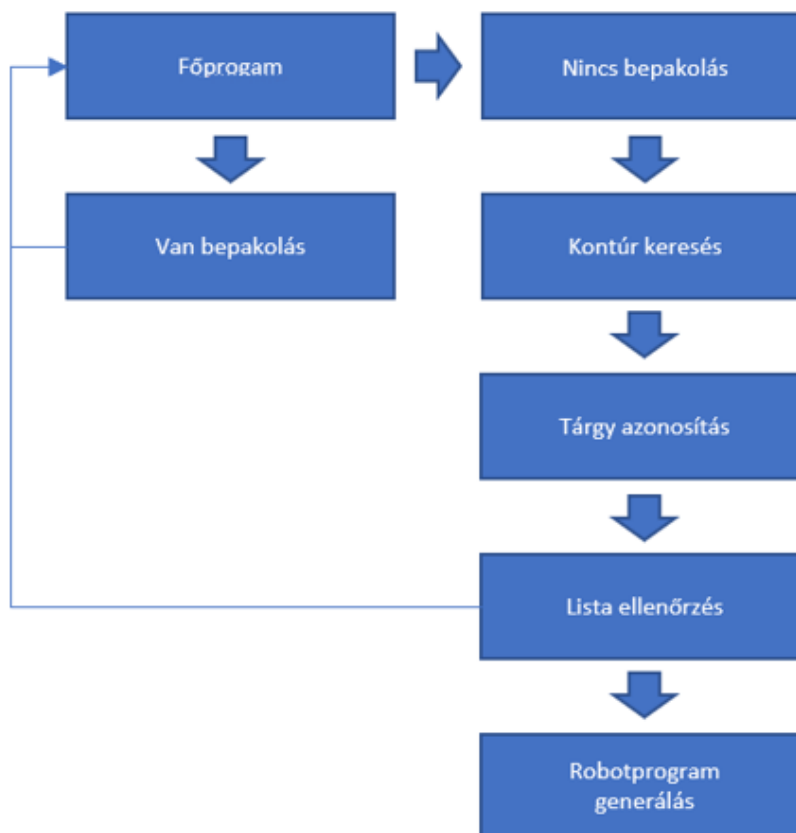
3.2. Fejlesztési folyamat lépéseinek bemutatása

A program fejlesztését a 6. ábrán látható lépések alapján végeztük. A folyamat a robot, és a számítógép bekapcsolásával és a kamera aktiválásával kezdődik. A programot elindítva a felhasználó párosíthatja az elemeket a kívánt tárolókonténerekkel és elkezdheti a bepakolást. Egy más után tetszőleges sorrendben történhet meg az alkatrészek munkaterületen való elhelyezése. Eközben a rendszer videófelvétel által képes észlelni meddig tart a bepakolás, ez idő alatt még nem kezd meg a tárgyak konkrét azonosítását, csak a sorrendjüket menti el. Amint a felhasználó végzett az utolsó elem elhelyezésével is, tehát a program már nem észlel mozgást, elkezdődik a videó utolsó képkockájának feldolgozása. A rendszer a munkaterületen található összes tárgyat képes egyszerre észlelni, azonosítani őket, kiszámolni a megfogáshoz szükséges középponti koordinátaikat és irányultságukat. Ezen információk alapján automata robotprogram generálása történik meg, amely lehetővé teszi, hogy a robotkar a megfogó segítségével FIFO elv szerint a kívánt helyre juttassa a munkadarabokat.



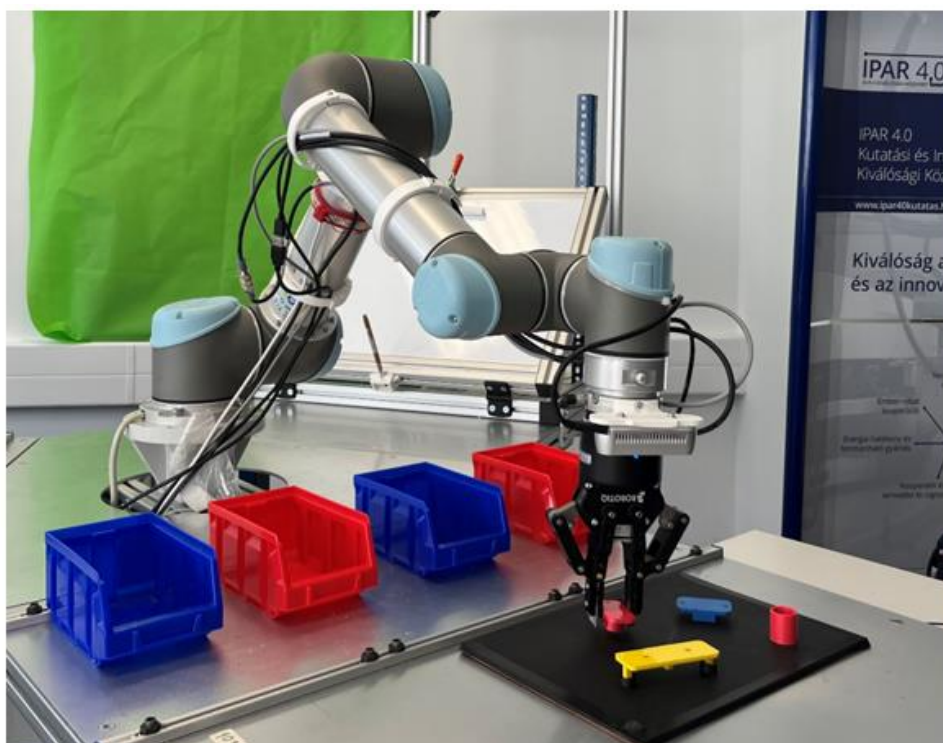
6. ábra Folyamatábra I.

A folyamat részletesebb ábrája a 7. képen látható. A feladat során a robotkarra felszerelt kamerától kapja a program a videófelvételt a munkaterületről.



7. ábra Folyamatábra II.

Ha nincsen bepakolás – mozgás - a munkaterületen, akkor megkezdődhet az aktuális képkocka kielemezése. Digitális képfeldolgozással megkeresésre kerülnek a kontúrok, majd ezeknek középponti koordinátáit és a szögelfordulásait egy listában eltároljuk a sorrend megjegyzéséhez. Ezt követően a kontúr alapján számolt különböző geometriai jellemzők alapján tárgyzonosítás történik. A rendszer képes a munkaterületen elszórtan elhelyezkedő összes tárgyat felismerni és pontosan besorolni a megfelelő kategóriába. Ahogy az a 8. ábrán látható, mindegyik csoporthoz rendelve van egy tárolópozíció, ahova az adott elemeket szállítani szükséges. Ezen információk átadásra kerülnek a robotnak, ami azok alapján a sorrend szerint következő alkatrészt biztonságos magassági szinten megközelíti. Ezután a kinyert adatok által kiszámítja milyen szögben kell elforgatnia a 3.csuklóját, hogy a rajta található 2 ujjas gripperrel meg tudja fogni és a megfelelő helyre mozgatni a tárgyat.



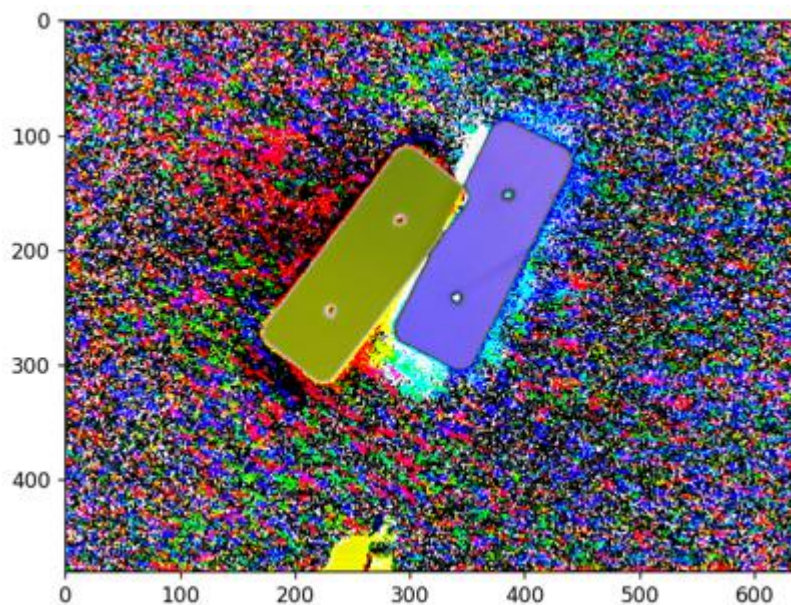
8. ábra Munkakörnyezet

3.2.1. Főprogram

A digitális képfeldolgozás megkezdése előtt fel kellett mérni, hogy vajon a feladathoz szükséges-e valós idejű folytonos videó felvétel elemzése, vagy pedig elégséges egy adott időközönként vételezett kép is. Mint tudjuk a videó is képek halmaza, amit többféle módon is tudunk jellemezni, az egyik leginkább elterjedt jellemzője az az fps. Vagyis *frame per second*, magyarul a másodpercek alatt vételezett képkockák száma. Alapos megfontolás után arra jutottunk, hogy feladathoz valós idejű videó feldolgozást fogunk használni. Ez amiatt kellett mivel, ha adott időközönként vennénk mintát akkor lehetséges, hogy éppen a kezünk benne lesz a mintavételben, mivel abban a pillanatban bepakoltunk egy munkadarabot a kamera képébe. Ezt kiküszöbölve videófelvétel mellett döntöttünk és készítettünk egy, a mozgást érzékelni tudó programot. Ezt használva már megállapítható mikor van mozgás, ilyenkor nem elemezzük ki a képkockát.

Először a robotkart el kell küldeni a mintavételezési ponthoz, ezt a program elején meghívott függvény teszi meg. Mivel a videót egy fix pontból készíti így ezeket a paramétereket fix értéként elmentve tudjuk tárolni és meghívni mikor szükségünk van rá. Miután a parancsot kiadtuk a felvételt már elkezdti ilyenkor a kamera, késleltetésre nincsen szükségünk hiszen amíg mozgásban van addig a program elején lévő mozgást érzékelő rutin nem engedi, hogy a fő programból tovább lépjen a feldolgozási részre a program.

Ezt a mozgást egy *move* parancs segítségével listában stringként (szöveggként) összefűzve adjuk meg a robotnak. A robotnak a koordinátákon kívül fontos megadni a sebesség és gyorsulás adatokat is, amiket a mozgáshoz használ, vagy pedig az időt, ami alatt végezze el az adott műveletet. A feladat során a sebesség értéknek 0.1 m/s míg a gyorsulásnak 1 m/s^2 értéket határoztunk meg. A videókapcsolat létrehozása után egyből veszünk egy frame mintát, amit *lastframe*-nek nevezünk el (előző képkockának). Erre azért volt szükség, hogy a ciklusba való belépés után azonnal tudjunk venni egy aktuális képkockát és ezek különbségét képezni. Ez úgy néz ki, hogy az előző és a mostani képkockának az abszolút értékét egymásból kivonjuk és így megkapjuk a különbségüket. Azért fontos, hogy az abszolútértéküket vonjuk ki egymásból mivel, ha csak simán kivonnánk őket akkor könnyen alulcsordulhatnak a pixelek, ahogy az 9. ábrán is látható.

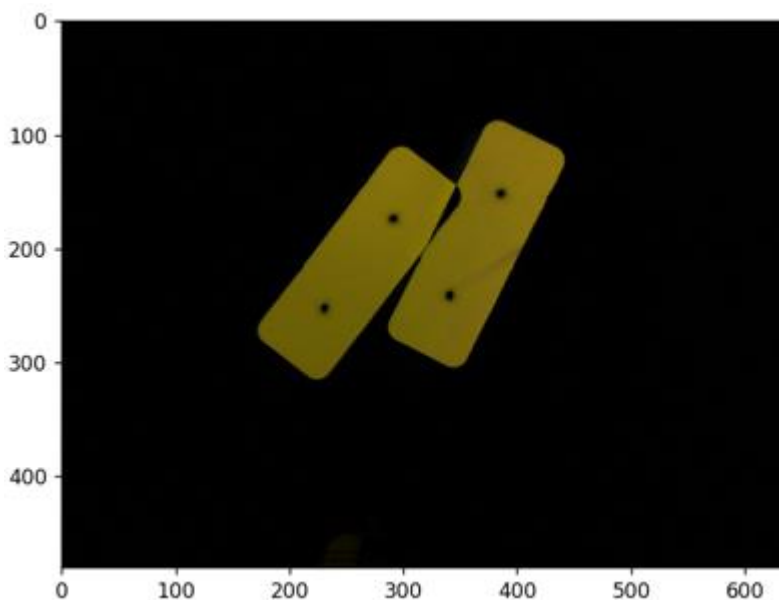


9. ábra Alulcsordult kép

A képen látható sok kék, zöld, piros stb. pöttyök mind azt mutatják, hogy az adott pixelek azok alulcsordultak. A színes képek -az esetek döntő többségében- három színcsatornával rendelkeznek. Ez a három színcsatorna a tiszta vörös (Red), a zöld (Green) és a kék (Blue), egyben RGB. Az RGB színrendszer az *opencv* modulban fordított sorrendben van elmentve, programozásnál a BGR szerint kell hivatkoznunk rá. Ezen színcsatornák adatainak mentésére a képfeldolgozásban egyenként 1 bájt áll rendelkezésre. Vagyis egy pixelen belül 8 vörös, 8 zöld és 8 kék bit lesz, ami tudja menteni az adott komponens értékét. Ez minimálisan a 0, míg maximálisan a 255 érték felvételét biztosítja. [64]

Ha az előbb tárgyaltak szerint vesszük a példát, miszerint kivonunk két képet egymásból és választunk egy pixelt, aminek az egymásból kivont értékei nullánál kisebb negatív számot

adnak nekünk vissza, akkor ezt hívhatjuk alulcsordulásnak. Ilyenkor a számláló csökkenés közben átlépve a nullát visszaugrik 255-re, majd onnan csökken tovább. Ezt úgy tudjuk kiküszöbölni, hogy abszolútértékét vesszük ennek a kivonásnak és ilyen módon a pixelünk egyes színcsatornáit nem csordulnak alul. Ellenkező esetben beszélhetünk túlcsordulásról is. Alulcsordulás nélkül, abszolútértékkel a differencia a 10. ábrán látható.

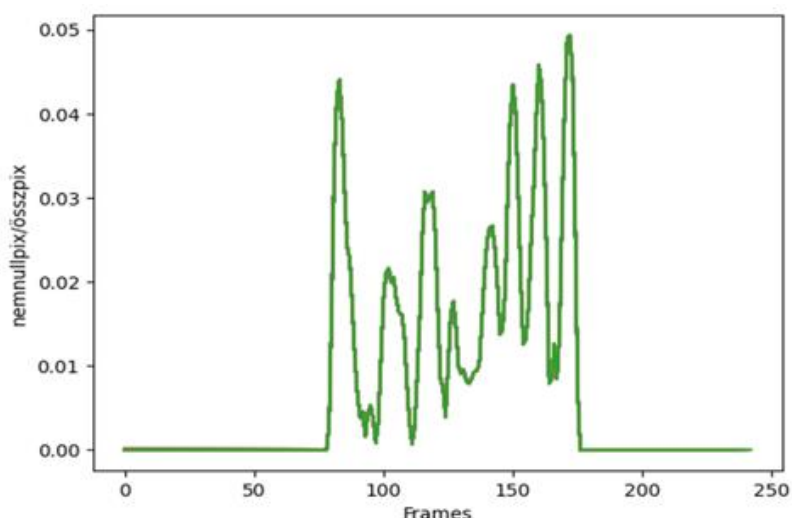


10. ábra Abszolútértékes kivonás

Ezek után a képkockák differenciáját szürkeárnyalatossá konvertáltuk át. Ez annyit tesz minthogy, az alapból három színcsatornával rendelkező valós idejű képsorozatot egy színcsatornássá tesszük, másképpen úgy fogalmazhatnánk meg, hogy csak a fényesség paramétert hagytuk meg míg a szín paramétereket elhagytuk.

3.2.2. Mozgásérzékelő

A főprogramban ezután egy változót definiálunk, melyben a számláló helyén a küszöbölt képkockák nemnulla elemeinek száma van, a nevezőben pedig a kép összes pixele. A kép összes pixelét úgy számítottuk ki, hogy beszoroztuk az aktuális frame alakját meghatározó oszlopok és sorok számát, míg a képen lévő nemnulla (vagyis nem fekete, mivel a teljesen fekete pixel az 0 értékű) értékeket parancs segítségével ugyancsak az éppen aktuális frame-ből számítottuk. Ezek hányadosa egy olyan értéket ad vissza számunkra, mely akkor változik mikor valami mozgást érzékel a kamera. Nem nehéz belátnunk, hogy a kamera az adott fényességű tárgyakat bejelzi az osztással, hiszen, ha a képeken mozgás van és így a fényesség változik, akkor a nemnull pixelek/összes pixelek arány az nőni fog.



11. ábra Mozgásérzékelő diagramja

Amint a mozgás abbamarad akkor az arányunk változása is lecseng, ahogyan azt a 11. ábrán is láthatjuk. Ennek az érzékenységet változtathatjuk a küszöbérték változtatásával. A programot úgy írtuk, hogy ameddig mozgást érzékel a program addig a fő program hurkában maradunk, nem hívjuk meg a függvényt, ami a képeket feldolgozná. Mozgásküszöb paraméter néven változtathatóvá tettük a limitet a határ megválasztáshoz, amivel az érzékenységen tudunk állítani. Ez esetben a végső küszöbérték az 0,0001 lett. Amint a mozgásnak vége, a jel lecseng, és a főprogramon belüli ágba beágyazott függvény, amit ilyenkor meghívunk kiértékeli a képet, ennek a függvénynek a neve kontúr kereső függvény.

3.2.3. Kontúr kereső függvény

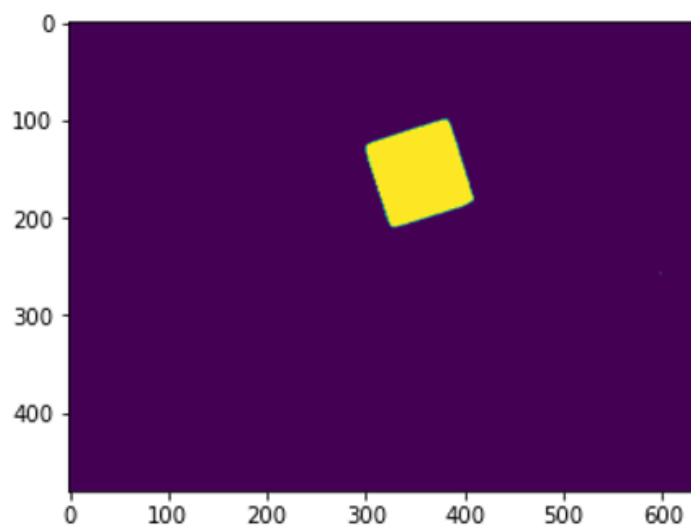
A függvény az adott képen elvégez egy maszkolást. A küszöbölés a HSV rendszerben történik, *S*-re és *V*-re szűrjük a különböző munkadarabokat, megvizsgálásra kerül az összes pixel *S*~Saturation és *V*~Value értéke, amelyek a telítettségről és fényességről szolgáltatnak információkat. Ennek megvalósításához több módszer is használható. Az egyik ezek közül a logikai műveletek alkalmazása. Ennél az adott kép pixeleinek valamelyik kiválasztott értékei kerülnek összehasonlításra egy általunk meghatározott számmal. A kritériumnak megfelelő képpontok logikai igaz, tehát 1-es értéket kapnak, a többiek pedig 0-át. A következő lehetőség a *threshold* parancs használata: működésében hasonló az előzőhöz, csak még lehetőség van az argumentumában megadni azt is, hogy kapásból mennyivel szorozódjanak meg a pixelértékek. Általában ez úgy van megválasztva, hogy 0 és 255 értékű képpontok álljanak elő, amely annyi előnyt jelent, hogy már a művelet közben láthatóvá válik a maszkolás eredménye. Ennél az

utasításnál *Otsu*-módszer is alkalmazható, amelynél nem kell nekünk egy küszöbértéket választani, hanem a program megpróbálja saját maga kitalálni (és az esetek nagyrésztében elfogadható módon ez sikerül is.) A felsorolt 3 eljárás mind fix értékekkel dolgozik, amelyek sokszor nem működnek jól pl. egyenetlen háttérvilágítás komoly zavart okoz náluk, viszont adaptív értékekkel ez jól kiküszöbölhető. Pythonban található parancs erre: *cv2.adaptiveThreshold()* néven, 6 állítható paraméterrel az argumentumában:

- simító kernel mérete, fajtája, iránya
- színcsatorna kiválasztása
- egy küszöbérték megadása
- egy érték, amellyel a pixelek megszorzásra kerülnek

Ez képes arra, hogy a bejövő képen egyből elvégez a megadott beállításokkal egy simítást, amellyel a homogenitásra próbál törekedni, majd irányadóként figyelembe veszi a beírt küszöbszámot. Ezzel az értékkel megvizsgálja a fotó minden részét és ha szükséges módosítja azt, annak alapján, amit észlel. Viszonylag meglepő módon a feladatnál nem sikerült elérni vele még a kamera beállításainak módosításával sem, hogy a tárgyakat és a háttérrel mindig megfelelően elválassza. Emiatt visszatértünk az első módszerhez, és egy konkrét érték megválasztásával próbálkoztunk. Az alkatrészek egy sötét háttér előtt helyezkednek el, ezáltal a helyes küszöbölési szintet alkalmazva megfelelően kivehetőek. A kamera beállításainak változtatásával, illetve módszeres tesztelések után sikerült megtalálni azt az értéket, melynél már csak és kizárólag a tényleges tárgyak kerülnek detektálásra.[64]

Ezzel megalkottuk a végleges maszkot, amivel szűrjük a képeket. Ha megvan a szűrt kép akkor a 12. ábrán látható képet kapja vissza a program.



12. ábra Szűrés alapján detektált elem

A program végén még szükség van arra, hogy meghatározzuk a szöget, melyet a munkadarabok az elhelyezésükkor bezárnak a vízszintes tengellyel, illetve a munkadarabok középponti koordinátáit. Erre azért van szükség, hogy a megfogó pofái a megfelelő – a munkadarabbal párhuzamos – állásba kerüljenek és így fel tudja azokat venni. Emellett fontos még, hogy a mozgatni kívánt alakzatoknak, ha van hosszabb oldaluk (nem kockák) akkor azoknál fogjuk meg a gripperrel.

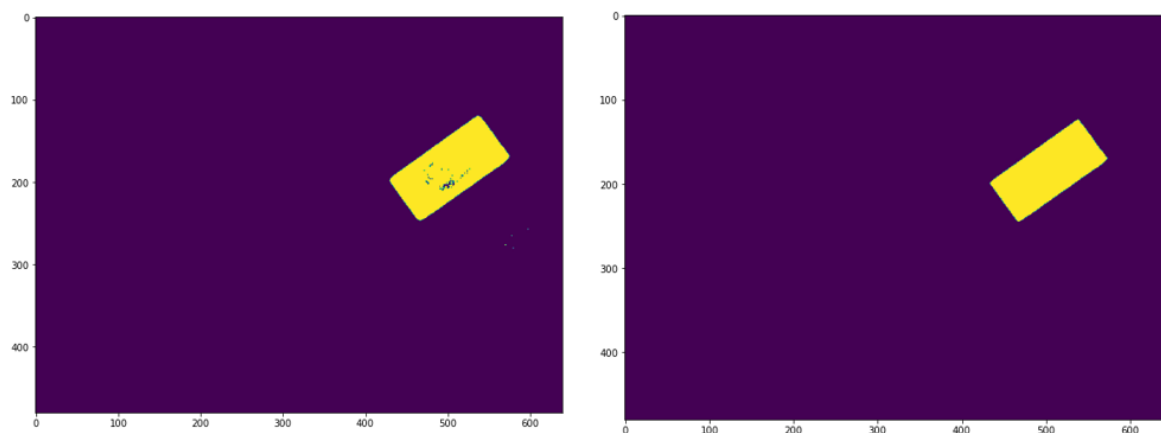
Ezután jöhet a kontúrkeresés. OpenCV-nek van *findcontour* funkciója, mely segít a kontúrok kiemelésében a képről. Ez a legjobban bináris képeken működik, ezért érdemes előtte elvégezni a küszöbölést, szűrést. A kontúrkeresésnél a talált kontúrokat kigyűjti a program egy változóban, amit utána vissza tudunk rajzoltatni az eredeti képre ellenőrzésképpen. Azzal a problémával szembesültünk, hogy sok nulla pixeles hiba van a képeken és a program ezeket találta meg és rajzolta körbe, majd értelmezte kontúrként. Ezután pedig a kiértékelésnél az adott pixeleknél a szögét, illetve középponti koordinátáját kaptuk eredményként. Ezt a problémát úgy oldottuk meg, hogy a kontúrok alapján kiszámítottuk a területet, amit behatárol. Ha ez megvan akkor feltételhez kötjük, hogy egy előre megadott méretnél nagyobbnak kell lennie a területnek vagy nem megfelelő kontúrról van szó. Ez úgy történik, hogy a kontúrokon végig iterálva azokat a kontúrokat melyek a megadott méret feltételnek nem adnak eleget átugorja a program. Emellett a kép javítása érdekében ennél a pontnál használtunk morfológiai műveleteket, melyek segítségével kiszűrtük a zajos régiókat. Ilyen egyszerű morfológiai műveletek például az erózió illetve a dilatació. Az erózió során a struktúráló elem tagjainak a minimumát számoljuk ki, míg dilataciónál a struktúráló elemek maximumát. Ezeknek kombinációiból felépítve létezik még rengeteg más művelet. A dilatació struktúrálóelem tagjainak a maximumának vételével egyenlő. Ha elképzelünk egy egyszerű struktúráló elemet, mondjuk egy 3x3-as négyzetet csupa egyesekkel. Ha ez egy morfológiai elemnek a struktúráló eleme akkor azt írja le, hogy saját maga és a szomszédjai – esetünkben az öt körülvevő 9 elem közül – a maximummal helyettesítjük az adott pixel értékét. Ezt úgy is megfogalmazhatjuk, hogy a „fehér” területeket növeljük általa. Ha például mi egy tiszta fekete és fehér képről beszélünk akkor ez azt jelenti, hogy ha a pixelnek a környezetében találunk egy fehér pixelt akkor a pixel fehér lesz a transzformáció után. Az erózió ugyan ezt jelenti, csak abban az esetben a minimumot ellenőrzi. Példánknál maradván egy teljesen fekete és fehér kép esetében ez a módszer a „sötét” területeket fogja növelni.

Ha adott pixel környezetében van egy fekete pixel akkor azzal lesz helyettesítve. A 13. ábrán mindezekre láthatunk példát, balról az első kép az eredeti kép, középen a dilatáció képe, majd a jobb oldalon az erózió által létrehozott képet láthatjuk. Megfigyelhetjük, hogy az eredeti képen el lett helyezve néhány fekete pixel, többek között az égre, illetve néhány fehér pixel a fák lombjai közé. Ezeket a dilatáció és az erózió a maga módján kiemeli, illetve eltünteti. [64]



13. ábra Dilatáció és erózió szemléltetése [64]

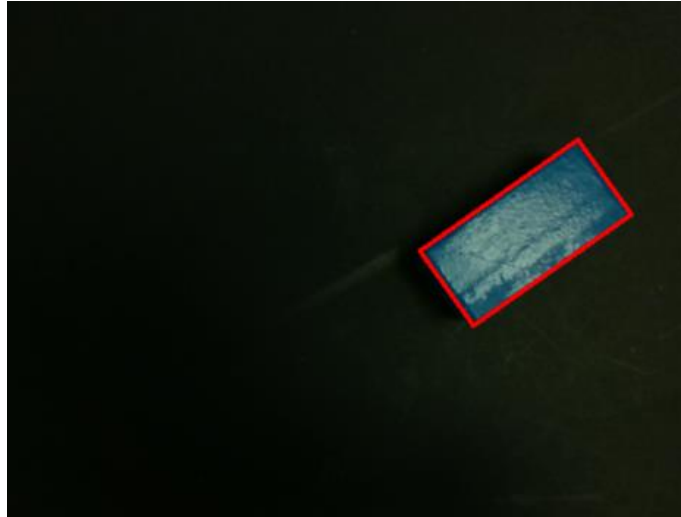
Az előbbieket tudatában kijelenthetjük, hogy az erózió, illetve dilatáció használatával a maszkolt képünkön a kisebb zajos régiók eltávolítása sikeresen megtörtént, melyet láthatunk is a 14. ábrán. Emellett a méretbeli szűrést alkalmazva a program már megfelelően képes érzékelni a kontúrokat.



14. ábra Erózió dilatáció előtti és utáni kép

Ha a kontúrjaink meghatároztuk utána tudjuk meghívni a *minarearect* függvényt. Ezzel a függvénnyel képesek vagyunk a kontúrok felhasználásával meghatározni a legkisebb köréírható négyzetet. Ezután a négy sarokpontot a parancs meghatározza és megszámozza őket. Az alapján, hogy melyiknek van a legnagyobb „y” koordinátája, az lesz a nulladik pont, abban az esetben, ha két pont van ugyan olyan „y” értékkel akkor a jobb oldali lesz a nulladik számú. Az „y” koordináta a képfeldolgozásban úgy helyezkedik el, hogy bal felső sarokból növekszik lefelé, míg az „x” koordináta ugyan onnan csak jobbra növekszik. Ha megvan a legnagyobb „y” akkor ezután a sarokpontok számozása az óramutató járásával megegyezően történik. Ennek segítségével pedig képes a függvény kiszámítani az objektum által bezárt szöget. Azonban a szög, amit visszakaptunk az mindig a nulla és egyes pontok közötti volt. Viszont nem volt megfelelő minden esetben ez a szög, hiszen, ha a munkadarabnak a rövidebb oldalainál lévő szöget mérte meg a program akkor a rövidebb oldal mentén fogta volna meg a gripper, ami nem minden esetben lenne biztonságos. Ezért a programot kiegészítve beleírtunk egy oldalhossz mérő függvényt, ami a meghatározott nulla-egy és nulla-három pontok között számítja ki a távolságot. Ezt használva pedig eldönthető, hogy egyes esetekben melyik oldal hol található. Miután ez megvolt feltételeket szabtuk és aszerint, hogy melyik oldal a hosszabb kikötöttük milyen korrekció kell, hogy megfelelő oldalak mentén fogja meg a munkadarabokat. Ezt 90 fok kivonásával értük el. Emellett beleírtuk a feltételek közé, ha a kamera éppen 0 vagy 90 fokot detektál akkor szintén az oldalhosszúságok alapján maradjon annyi vagy forduljon el.

Ezeket az értékeket átváltva radiánba a robotkar értelmezni tudja és be tudja fogadni azokat. A fejlesztésnek ennek a szakaszában abba a problémába ütköztünk, hogy a robotkar mozgatni kívánt hármas csuklója az nem abban a pozícióban volt nulla fokra kalibrálva, amit használtunk. Emiatt kiegészítésre került korrekciós tényezővel, amivel ki lett küszöbölve a szögeltérési probléma. A tényező mértéke 17.79 fok volt negatív oldalra. Mindezeket megoldva a robotkarnak már csak be kellett táplálni az adatokat. A 15. ábrán láthatjuk egy, a program által meghatározott minimális területű négyzetet.



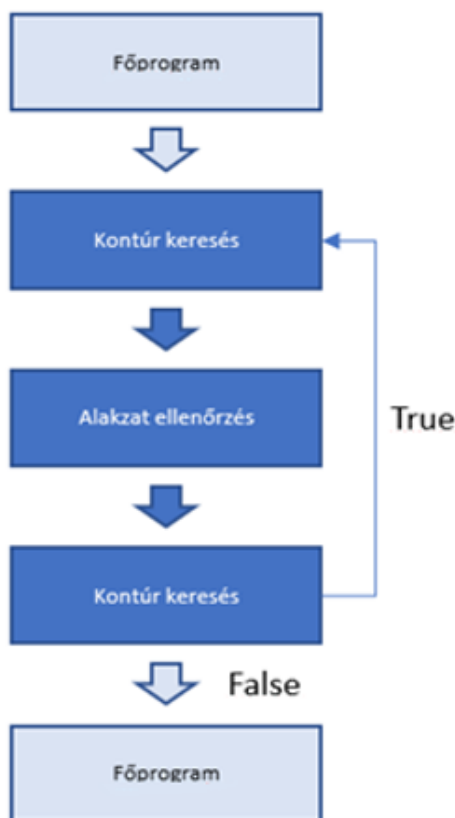
15. ábra Minimális területű körülrajzolt négyzet

A *minarearect* függvényből még ki tudjuk venni a középponti koordinátákat is, amikkel a megfelelő pozícióba tudjuk mozgatni a robotkart. Ehhez szükség volt egy pontra, amihez viszonyítva hozzászámoljuk a távolságot, amit a fotóról kijelez számunkra a program. Ez a kitüntetett pont a kép sarkában lett meghatározva. A képen mérhető pixel értékeket, amit a program x és y koordinátákban megad átváltottunk méterbe, hiszen a robotkarnak abban kell megadni a távolságot. Ezután pedig hozzáadtuk az értékeket a fix koordinátaponthoz. Olyan módon oldottuk meg ezt, hogy a robot tengelye szerinti x koordinátához adtuk a kép szerinti y tengely átváltott pixel értékeit. A másik robot szerinti y tengelyhez pedig a kép szerinti x értékeket adtuk hozzá. Amiatt kellett keresztbe hozzáadni mivel a kamera abban a helyzetben ahogyan használtuk az ellenkező koordinátatengelyei mentén volt párhuzamos a robotéval.

3.2.4. Alakzat ellenőrző program

A feladat ezen részénél a tesztek alatt előjött az a probléma, hogy nem érzékel több kontúrt a program, mivel nem tudja mi az, amit már megtalált és mi az, amit még nem. A program úgy működik, hogy a kamera „y” tengelye mentén halad lefelé az aktuális képnek a kiértékelésekor. Ennél fogva, ha ezen tengely mentén növekvő irányba helyeztük be a munkadaraboknak már most is észlelte a kontúrjait, azonban nekünk az a célunk, hogy bármilyen helyre történő egymás utáni bepakolásnál is megtalálja az új kontúrjainkat.

Írtunk egy programot, melyet a kontúr kereső programba hívunk meg. Ennek a folyamatábrája a 16. ábrán látható.



16. ábra Folyamatábra III.

Ez az alakzat ellenőrző program azt csinálja, hogy megvizsgálja szerepelt-e már az adott kontúr előzőleg, ezt olyan módon teszi, hogy elmenti a talált kontúrnak a középponti koordinátáit és elfordulásának szögét és mindig az aktuálisan kapott új érték alapján megvizsgálja (adott tűréssel), hogy a már elmentett értékekkel egyezik-e az. A függvényben határértékek között kell kezelni a méreteket, hisz mind a képfeldolgozással történő mérésben, mind az élethelyzetekből adódóan – odébb lökünk bepakolásnál egy munkadarabot – lehetnek olyan hibák melyekre nem számítunk. A kisebb elmozdulásokat melyek a mérésből és kisebb lökésből adódnak tudja kezelni a program, de a nagyobb elmozdulásokat új munkadarabnak veszi és újként emeli be a listába.

Mindezek után a program első felének a felépítése úgy néz ki, hogy a főprogramba meghívjuk a kontúr kereső függvényt mikor nincsen mozgás. Ebben a függvényben kontúrokat keresünk a képen és azok szögelfordulásait. Ebbe a függvénybe, ha talált kontúrt, meghívjuk az alakzat ellenőrző függvényt, amivel azt figyeljük volt-e már az adott kontúr az előzőekben, és ha igen akkor kerestetünk újabb kontúrt. Ha talál új kontúrt akkor annak középponti koordinátaival és szögelfordulás értékével megy tovább a program, ha nem, akkor üres listával

tér vissza. Üres lista visszatérése esetében a főprogramba visszatérve kihagyja a lista ellenőrzést és a *while* hurok elejére ugrik, ezzel elkezdve a következő képkocka feldolgozását. Ha viszont koordinátákkal és elfordulási szöggel tér vissza akkor azt a sorrendi listába menti, hogy azt később a szortírozási szakaszban fel lehessen használni. Ezután tér csak vissza a főprogramba és kezdi újra a mozgásnak a vizsgálatát majd elemzi ki az elkövetkező képkockákat.

A programnak ennél a részénél a tesztek közben abba a problémába ütköztünk, hogy mikor teszteltük a folyamatot akkor a kontúr kereső random téglalapokat talált nekünk - még mikor nem került a kamera alá munkadarab – a fényesebb részeknél. Próbáltuk a határértékeket állítani, de ezzel nem sikerült elérni, hogy ezt kiküszöböljük. Ami feltűnt az az volt, hogy mindig csak egyszer csinálja ezt meg a programom az indítás utáni első képkockák egyike volt állandóan, ami bejelezte a „fantom” téglalapot. Miután ezt az értéket engedték továbbvinni a programnak az minden probléma nélkül futott. Ekkor jutottunk arra a következtetésre, hogy lehet a kamerának kell egy adott idő, ami alatt a konfigurációs beállításával végez. Ezen gondolatnál fogvaadtunk egy 60 képkockás (két másodperces) késleltetést a program indításának elejére, ami után az elvártaknak megfelelően működött már.

3.2.5. Tárgyosztályozás

Ahhoz, hogy a folyamat során ne csak egy, hanem több különböző alkatrész konkrét azonosítása és áthelyezése is megvalósulhasson, részben módosításra került a képfeldolgozási rész, amelynél már a mesterséges intelligencia kezdeti szintű alkalmazása történik meg. A művelethez az alábbi elemek állnak rendelkezésre: rózsaszín, illetve kék színben azonos formájú hasáb, továbbá sárga színben hosszúkás téglatest, és rózsaszínben henger geometriával rendelkező. A szoftveres rész szempontjából amiatt bír nagy jelentőséggel az, hogy ezen elemek kerülnek felhasználásra, mert így már nem lehetséges kizárólag szín alapú szűrő által szétválogatni a tárgyakat, nagy hangsúlyt kell fektetni a geometriai sajátosságok detektálására is. Hiszen ezek által sorolhatóak be az alkatrészek kellő pontossággal az adott osztályokba. A programrészek létrehozása előtt szükséges néhány kérdés tisztázása: az első és legfontosabb, hogy milyen tárgyak kerülnek felhasználásra. Egy-egy mintadarabbal teljesen megegyezők, tehát pontos egyezés keresésére lesz szükség? Vagy esetleg kis mértékben eltérők, akkor viszont az alapján fog történni a kategóriába sorolás, hogy melyikekhez hasonlít leginkább. Olyan szituáció is előfordulhat természetesen, hogy előre nincsenek definiálva az alkatrészcsoportok, és a programnak az összes elem áttekintése után saját magának kell osztályokat létrehozni.

Manapság rendkívül felkapott és rohamosan fejlődő terület az Artificial Intelligence=AI. A matematika és a logika használatával a számítógépes rendszer képes szimulálni azt az érvelési eljárást, amelyet az emberek is használnak, amikor új információk alapján tanulnak és döntéseket hoznak. A mesterséges intelligenciát használó számítógépes rendszerek előrejelzéseket nyújtanak vagy a meglévő adatok mintázatán alapuló műveleteket hajtanak végre, és tanulni képesek a hibáikból, így növelve a pontosságukat. Egy kifejlett AI az új információkat rendkívül gyorsan és pontosan dolgozza fel, ami lehetővé teszi, hogy bonyolult helyzetekben is használjuk, például az önvezető autókban, a képfelismerő programokban vagy a virtuális asszisztensekben. [62]

Amikor már olyan képfeldolgozási feladatot kell megoldani, ahol nem elég egy bizonyos tárgyat megtalálni a többi között, hanem többet is tudni kell pontosan azonosítani, akkor már egy jóval összetettebb, 3 nagy részből álló folyamat és azon belül több művelet végrehajtása szükséges:

- a) Feature-vectorok ismertetése: amelyek lehetővé teszik az elemek pontos azonosítását
- b) Maszkolási metódus: a tárgyak kontúrjainak előállítására
- c) Momentumok bemutatása: amelyek az egyik legfontosabb jellemzőértékeket tartogatják
- d) Színszűrő leírása: amiatt, hogy ne csak geometria alapján történjen az azonosítás
- e) Betanítás: ahhoz, hogy a gép tudja milyen elemek vesznek részt a folyamatban
- f) Tárgyak csoportba sorolását végző algoritmus prezentálása: a szétválogatás elvégzése
- g) Tesztelés: a felmerülő hibák detektálása miatt

a) Feature-vectorok ismertetése

A működés lényege, hogy valami szabályosságot találjunk, ami alapján a gép olyan esetben is jó választ ad, amivel még nem találkozott pl. mutatunk pár képet a gépnek, amelyeken kutya látható és be is adjuk neki ezt az információt, illetve pár olyan tanítóképet is, amiken nem található. Ezeket a gép összefésüli és olyan sajátosságokat keres, ami alapján majd ismeretlen képekről is el tudja dönteni melyik csoportba tartoznak. Viszont azáltal, hogy az összes fotó minden pixelét vizsgálnia kell, felmerül egy komoly probléma. Ugyanis még egy 100 x 100-as méretű szürkeárnyaltos kép is rengeteg adatot tartalmaz. Ugyanez a kép minimálisan módosítva például elforgatva, kissé eltolva már teljesen más számtömbnek felel meg. Az adatcsökkentésre a megoldást az jelentené, ha valami által meg lehet mondani a gépnek, hogy nem két teljesen különböző képről van szó, hiszen ekkor már nem lenne szükséges olyan sok adatot eltárolnia. Ekkor jönnek képbe a *feature vector*-ok. A jellemzővektor a megfigyelt

jelenségek numerikus tulajdonságainak rendezett listája. Az emberek minőségi adatokat elemezhetnek a döntés meghozatalához. Például látjuk a felhős eget, érezzük a nyirkos szellőt, és úgy döntünk, veszünk egy esernyőt, amikor kimegyünk a szabadba. Öt érzékszervünk képes átalakítani a külső ingereket neurális tevékenységgé az agyunkban.

A gépi tanulási modellek azonban csak kvantitatív adatokkal tudnak foglalkozni. Ennek megfelelően a megfigyelt jelenségek jellemzőit mindig számértékekké kell konvertálnunk, és ugyanabban a sorrendben be kell táplálnunk egy gépi tanulási modellbe. Röviden, a jellemzőket jellemzővektorokban kell ábrázolnunk. A jellemzők egy képpontot vagy egy egész objektumot képviselhetnek a képen. Ezen tulajdonságok lehetnek például a színösszetevők, a hossz, a terület, a körszerűség, a színátmenet nagysága, a szürkeárnyaltos kép intenzitásértéke és ezeken kívül még rengeteg dolog. Tehát ezek kiszámítása által lehetőség van egy kép összes pixelértékének eltárolása helyett jóval kisebb méretű adattömbökkel jellemezni egy fotót a további feldolgozáshoz. Jó *feature vector*-okra általánosságban a következők mondhatóak el: egy bizonyos tárgy különböző képeken közel azonos jellemző-vektort eredményez, illetve különböző tárgyak *feature vectorai* jelentősen eltérnek. Továbbá ami a jelenlegi feladtnál is igazán lényeges, hogy az összetartozó tárgycsoportok elemei eléggé hasonlóak. [63, 64]

Kétféle módszer létezik a konkrét megalkotásukra. Az egyik *feature-engineering* néven ismerhető, amely során az ember választja ki az adott feladathoz illeszkedő jellemzőket. Míg a másik módszer az automatikus *feature*-keresés, amikor az ember csak a jellemzők osztályát adja meg és a gépi keresésre bízza a legjobb ismertetőjegyek megtalálását. Jelen feladat során az előbbi módszer került alkalmazásra.

b) Maszkolási metódus (ezt és a kontúrkereső fv kéne összegyűrni egy helyre...)

A korábban leírásra került kontúrkereső függvény meghívása történik ebben a szakaszban, mely által előáll az összes alkatrész körvonala. Így ezután következhet ezen kontúrok alapján a jellemzővektorok kiszámítása.

c) Momentumok bemutatása

Az adott programrészben több jellemzővektor is létrehozásra kerül, majd néhány szimuláció után azok adódnak hozzá a végső *feature*-listához, amelyek a legjobb eredményt szolgáltatják a mostani elemeknél. Az egyik például a cirkularitás értéket biztosító, tehát megadja mennyire körszerű az adott forma. A másik, amelyik arra ad vissza egy egész számot, hogy az adott alakzat hány oldalú sokszöggel közelíthető. Ezenkívül a *Hu* momentumokat számoló is alkalmazásra kerül. A momentum értékeket a gép a háttérben bonyolult matematikai műveletek által kalkulálja ki az adott képről. Jelölésük: $m_{j,i}$. Minden egyes alakzatnak

megvannak a sajátos értékei. Ha bármi módosítás is történik az adott tárggyal, ezek is változnak vele. Viszont léteznek belőlük invariáns mennyiségek is:

- Centrális momentumok: ez a tömegközéppontot az origóba helyezi át és ezáltal nem érzékeny az eltolásra.
- Normalizált centrális momentumok: ez is eltolja a tömegközéppontot az origóba és ezenkívül még az alakzatot egységnyi területűre skálázza, ezek által érzéketlen eltolásra és átméretezésre is.
- Hu momentumok: az eddigieken kívül ez elforgatásra sem változtatja az értékét minél inkább aszimmetrikus az adott alakzat, annál több értéke fog 0-tól különbözni, gyakran elég eltérő értékeket adnak vissza, emiatt összehasonlításnál a konkrét számok helyett a nagyságrendet érdemes figyelni

Példa az alkalmazásukra: tanítás során megadunk egy mintát, kiszámoljuk a momentumait és eltároljuk. A felismerés során a talált alakzatok értékeit összevetjük a korábban tárolttal. Ezen összevetési módszert alkalmazva a hasonló alakzatok (még ha át is vannak méretezve vagy el vannak forgatva) hasonló momentumokkal fognak rendelkezni és könnyedén megállapítható az egyezőségük. Ebből adódóan jelen feladatnál akár csak a *Hu* momentumok figyelembevételével is megállapítható lenne elméletileg, hogy egy alakzat melyik csoporthoz tartozik. Azonban a rengeteg befolyásoló tényező miatt érdemes több tulajdonság összehasonlítása a megbízhatóbb működés érdekében. [64]

d) Színszűrő alkalmazása

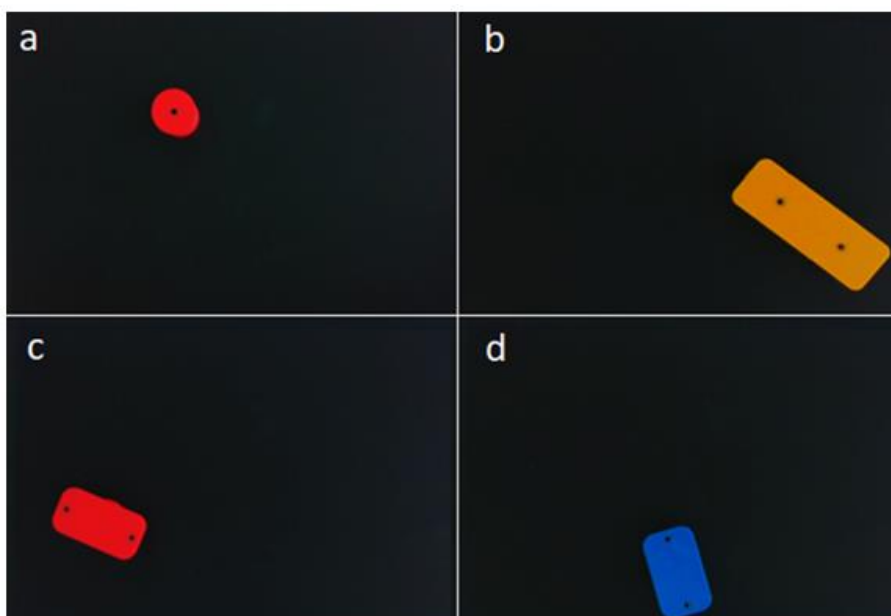
Természetesen a geometriai jellemzőkön kívül szükség van a színek figyelésére is, hiszen a rendszernek képesnek kell lennie teljesen egyforma, viszont eltérő színű alakok megkülönböztetésére. Alapvetően a kamerák szenzorjai minden pixelről egy-egy intenzitásértéket adnak vissza, miszerint az adott tárgy egyes képpontjai mennyi vörös, zöld és kék összetevőt tartalmaznak. Bizonyos feladatoknál viszont gyakran kerülnek alkalmazásra a pixelek konkrét Red, Green és Blue értékei helyett a színezet koordináták:

$$r = \frac{R}{R+G+B} \quad g = \frac{G}{R+G+B} \quad b = \frac{B}{R+G+B} \quad (5)$$

Ezek értékei nem tartalmaznak információt a fényességről, ami rendkívül nagy pozitívum lehet olyan folyamatok esetén, mint például a jelenlegi feladat, melyeknél nem biztosítható végig egyenletes megvilágítás. Ezeken kívül találhatóak illesztett ellipszis/téglalap oldalarányt számoló, tömörséget figyelő függvények is. Természetesen, ha más elemek is alkalmazásra kerülnének később, akkor a lista egyszerűen módosítható, a legjobb szétválogatás érdekében.

e) Betanítás

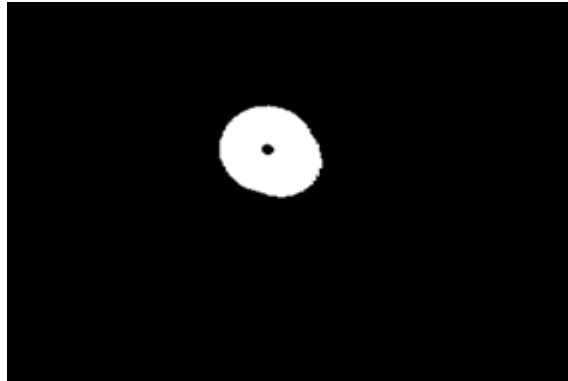
A második rész a betanítási procedúrából áll. Ahhoz, hogy a program később majd ismeretlen képeken is megfelelően meg tudja határozni az alakzatok csoportját, be kell tanítani neki milyen osztályok állnak rendelkezésére. A mostani folyamatban előforduló 4 különféle tárgynak létrehoztunk egy-egy osztályt. Mindegyikről 10-10 úgynevezett tanítóképet készítettünk, ezek közül látható egy-egy a 17. ábrán. Érdekes a minél széleskörűbb betanuláshoz a lehető legeltérőbb pozíciókban elhelyezni a tárgyat a munkaterületen. Ezáltal bármilyen elhelyezkedésnél lesznek már az összehasonlításhoz egy az egyben használható értékek korábról, szóval csökkenthető a besorolási bizonytalanság. Ezt a faktort jelentősen redukálja az is, ha minél több mintaképek kerül felvételre.



*17. ábra A végső folyamatban résztvevő elemek: henger(a),
hosszú téglatest(b), rózsaszín téglatest(c), kék téglatest(d)*

Az adott programrész oly módon került létrehozásra, hogyha a felhasználó új alkatrészt is be akar vonni a folyamatba, akkor csak egy mappát kell létrehoznia, amely legalább 10 png kiterjesztésű képet tartalmaz róla. Majd ennek a nevét hozzátennie a tanítólistán szereplőkhöz. Ezáltal automatikusan képes a program egyesével beolvasni a fotókat, majd feldolgozni azokat, például úgy, ahogyan az a 18. ábrán is látható.

Minden egyes tanítóképen a korábban leírásra került maszkolási módszer megy végig. A program majd csoportonként, a 10 képen megtalált kontúrokat összefésülve számolja a kiválasztott *feature-vektorokat*, és próbál meg „elfogadási tartományokat” felállítani a kategóriákhoz. Tehát például egy adott alakzatot csak akkor fog a hengerekhez sorolni, ha elég nagy körszerűségi értékkel fog rendelkezni, ahogyan észlelte az adott csoport összes mintaképen.



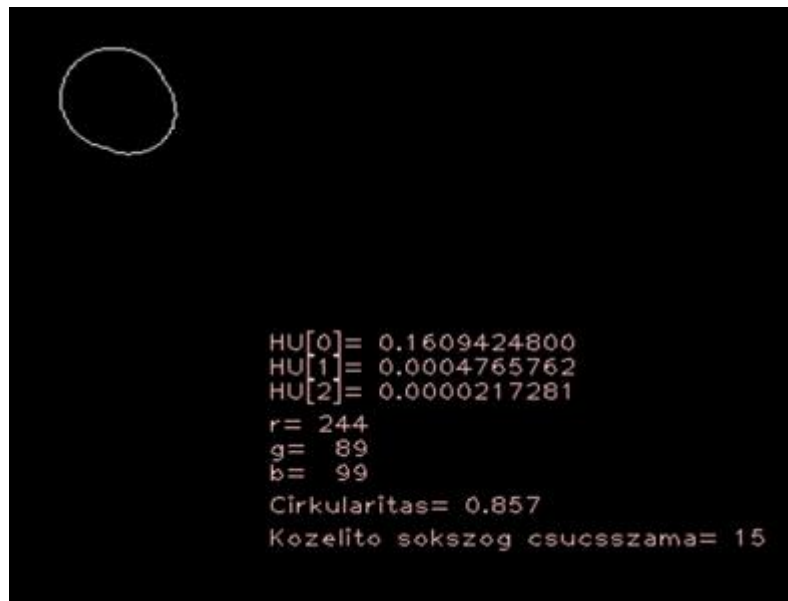
18. ábra Henger maszkolása

f) Tárgyak csoportba sorolását végző algoritmusok prezentálása

Ezen műveletek elvégzése után következik az, hogy a program valamilyen séma szerint az előállított információk alapján csoportosítson. Ez többféleképpen is történhet, rengeteg algoritmus elérhető: az egyik módszer a KNN-algoritmus. Működésének alapja a legközelebbi szomszédok keresése. Tehát miután elvégezte az új, ismeretlen tárgy *featureinek* kiszámolását, megnézi, hogy az értékei a tanítóhalmaz melyik eleméhez esnek legközelebb. Ezáltal abba a csoportba fogja majd besorolni, amelyikben az a tárgy is van. Egyszerű, viszont vannak hátrányai. Ugyanis nem igazi modellel dolgozik, ha egy bizonyos típusú alak teljesen eltérő, nem igazán tehető egyik osztályba sem, akkor teljesen véletlenszerű minek esik a közelébe. Ezenkívül csak akkor pontos, ha a tanítóhalmaz nagy, de ekkor a kiértékelés sok időt és nagy tárhelyet vesz igénybe.

Emiatt nem erre, hanem a Support-Vector Machines algoritmusra esett a választás. Ez összetettebb lefutású, a számolt vektorértékek alapján egy *feature*-teret hoz létre és 2D-ben elválasztó vonalak felállításával, 3D-ben pedig síkok által határoz meg zónákat. Ezáltal csak azt kell megjegyeznie, hogy az egyes osztályok elemei melyik részt fedik le, és az új alakzat hova esik. Ennél már jóval több paraméter állítható a minél jobb működés érdekében. Például megadható az algoritmusban használandó kerneltípus: rendelkezésre áll lineáris, *rbf* (~radial basis function), *poli*, *sigmoid* verzió. Ezek mind-mind eltérő működésűek, máshogy kalkulálják ki a háttérben, hogy az adott alkatrészt pontosan hova kategorizálják. Ezenkívül a *C*, úgynevezett büntetési paraméter/hibatagot is lehet állítani. Ha kicsinek van véve, akkor a

rendszer engedékenyebb, vagyis még bizonyos jellemzőknél (példaként szolgál a 19.ábra), nagyobb eltérés észlelésekor sem dob hibát, mindenképpen besorolja oda az adott elemet. Viszont nagyobb értéknél már megeshet, hogy nem fogja sehova sem beosztani, és ismeretlen tárgynak titulálja. Emellett lehetőség van nemlineáris SVM alkalmazására is, ami bizonyos függvényosztályú görbékkel, görbe felületekkel működtethető. További előnye, hogy sokszor jól funkcionál olyan tartományon is, ahol nincs tanítóhalmazbeli pont. A két bemutatott típuson kívül még rengeteg elérhető, azonban azok már jóval bonyolultabbak és ezen feladat összetettsége nem indokolja azok használatát. [64]



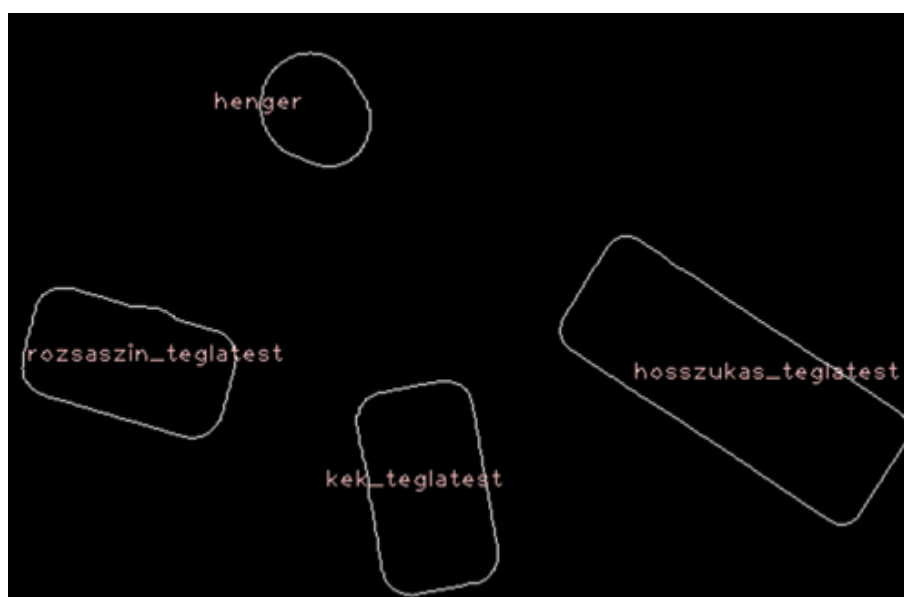
19. ábra Henger kontúrja és a feature vectorok értékei

g) Tesztelés

A harmadik rész pedig a tesztelésből áll, hogy időben fény derüljön a felmerülő hibákra. Ugyanis a teljes szimulációnál már pontosan láthatóvá válik, a program melyik része szorul esetleges módosításra. Ennél már egy olyan kép kerül beimportálásra, mint amilyen a folyamat tényleges futtatásakor is készülni fog. Ez azt jelenti, hogy egy bizonyos magasságú pozícióból létrejövő fotót alkalmazunk, amelyen már a műveletben résztvevő összes alkatrész megtalálható egyszerre. Maga a maszkolási folyamat ugyanúgy zajlik, mint a tanítóképeknél. Viszont itt már az összes elem maszkja előáll. Tehát egyidőben több hasznos kontúr is elmentésre kerül. Ezeket a program képes külön kezelni és egymás után a kért *feature*-öket számolni belőlük. Azonban a színezet koordináták számításánál mindig csak olyan fotóval tud operálni a rendszer, amelyen egyszerre egy tárgy szerepel. Hiszen, ha az összes, különböző színű alkatrész rajta lenne, akkor hamis RGB eredmények jönnének létre. Ez úgy tud megvalósulni, hogy létrehozásra kerül a kép méreteivel megegyező, teljesen fekete fotó. Majd

egy *for* ciklusban történik meg egymás után sorban a kontúrok meghívása. A cikluson belül az adott kontúr átmásolásra kerül arra, és az általa határolt részbe eső pixelek összes színcsatornája egy `cv2.fillPoly(out_img,[cont],[1,1,1])` paranccsal csupa 1-es értékkel kerül feltöltésre, míg a többi rész megtartja 0-ás értékét. Ezáltal gyakorlatilag egy speciális maszk képződik, amellyel „megszorzásra” kerül az eredeti kép. Ez azt eredményezi, hogy az az egy alakzat marad látható, a többi pedig elsötétítésre kerül.

Így már megvalósulhat a színjellemzők kiszámítása is a többi mellett. Aztán jöhet a következő kontúr, és ugyanez a procedúra. Miután megtörtént a léptetés az összesen, a debugolás miatt mindegyik kontúr kirajzolásra kerül egy közös kimeneti képre (20. ábra), illetve az is ki lesz rá írva, hogy a program melyik csoportba sorolta őket.



20. ábra A program jóslata arra, milyen tárgyak jelennek meg a képen

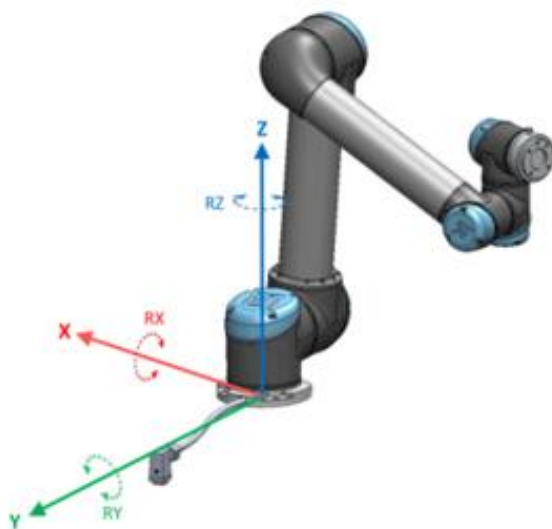
3.2.6. Lista ellenőrző program

Az adatokat, amiket kapunk a kontúr kereső függvénytől elmentjük egy listába, olyan sorrendben, hogy a két középponti koordinátát követi az elfordulási szög. Ezeket az adatokat egy újabb listába mentjük el a főprogramban, ahol így sorrendben lesznek az információk, hogy kipakolásnál ezeket tudjuk majd használni. Szükséges azonban egy lista ellenőrzés, ami azt figyelni, hogy az új, beérkezett lista által tartalmazott adatok listája vajon már egyszer benne volt-e a sorrendi listában. Ezt ellátja az alakzat ellenőrző program a kontúrkeresés programon belül.

A programunkban a munkadarabok felső határát 4 elembe állapítottuk meg. További elemeket is tudna érzékelni és a pozíciójukat elmenteni azonban a feladatunk megoldásában ez csak 4 munkadarabra lett megírva, ennyit kényelmesen el lehet helyezni a munkatérben. A program a végén visszatér a sorrendi listával. Ha plusz munkadarabokat pakolunk be jelzi számunkra, hogy megtelt a kapacitása. Paraméterként megadhatjuk, ha mi több vagy kevesebb munkadarabbal szeretnénk a műveletet megcsinálni, a *default* (alapértelmezett) érték azonban az 4. Mindezzel elértjük azt, hogy ne mindig nekünk kelljen foglalkozni a paraméterek hozzárendelésével, azonban, ha egyedi célra kell felhasználni abban az esetben lehetőségünk van egyénileg megadni.

3.2.7. Szortírozás

A sorrendi listából az adatok kivételét egy egyszerű iteráló művelettel oldjuk meg. Ezzel megkapva a középponti koordinátákat és a szöghelyzetet. Az elmozdulási folyamatokkal nem volt probléma, *moveL* paranccsal került megvalósításra. A nehézséget az jelentette, hogy az a *moveL* parancs paramétereit *base* koordinátarendszerben kell megadni. Ez viszont nem megfelelő ahhoz, hogy létrehozzunk egy megfogó általi forgatást adott szögben, ami viszont fontos mivel a szögelfordulás kell a munkadarabok megfogásához. A *moveL* parancsban megadható a TCP térbeli elhelyezkedése három ponttal $[x;y;z]$ koordináták alapján méterben, és a három forgási vektort $[R_x;R_y;R_z]$ koordináták alapján. Ahogy azt a 21. ábrán is látható a forgások itt az egyes tengelyek körüli forgatást jelentik.



21. ábra Robot koordináta rendszer

Ez viszont egy olyan megfogó általi forgást eredményez, aminél a gripper forgása a szerszámközepptől van nézve. Az így kialakuló forgási vektoroknak magasszintű fizikai és matematikai számításai vannak. Ennél az oknál fogva, mivel ezzel a paranccsal nem lehet

közvetlen irányítani a megfogó általi körülfordulást - az utolsó csuklón – így egy másik megoldást kellett választani.

A *move joint* jelentette megoldást a problémára. Ezzel a mozgási paranccsal azt lehet meghatározni, hogy a hat forgási pontjában a robot milyen szögben legyen. Vagyis, hogy milyen szögekben álljanak az egyes csuklói. Ennek alkalmazásával már képes a robotnak beadni olyan értéket, hogy a megfogóhoz tartozó hármas csukló forduljon csak el.

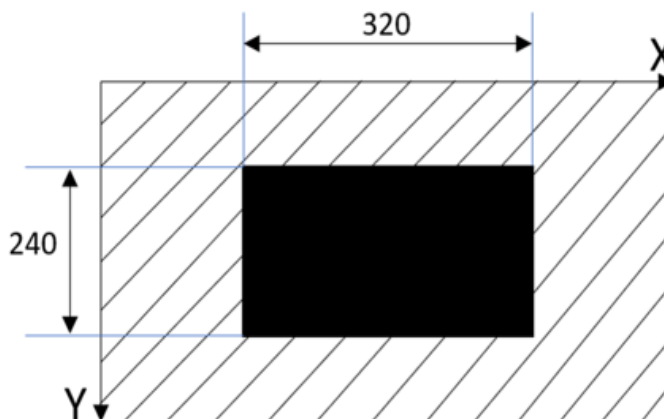
Azért, hogy ne kelljen mindig ugyan arról a kitüntetett pontról kezdeni a mozgást és így egyszerűbben lehessen importálni más alkalmazásokba ezáltal, a programomban használunk egy TCP és egy Joint (csukló) állást lekérdező ciklust is. Ez egy külön függvény, amit meghívunk a saját ciklusban és általa lekérdezzük mind a TCP mind a Joint állások értékeit, majd ezeket átszámítottam radiánba. A program alapja készen állt rendelkezésünkre, mi pedig ezt továbbfejlesztve a számunkra érdekes információk lekérdezésére használunk. Ez a RTDE (Real Time Data Exchange) kommunikáció. Ez lehetővé teszi számunkra azt, hogy egyénileg beállított adatokat kaphassunk a robottól valós időben. A kontroller mindig biztosít adatokat, amik reprezentálják a robot állapotát, ki és bemeneti adatait, mi ebből a számunkra fontosakat kiválasztva egy minta alapján beállítottuk egy külső .xml fájlban. Ez mindig a felhasználó feladata, hogy megválassza az általa szinkronizálni kívánt adatokat. A program során többszöri alkalommal megvalósítunk RTDE, vagyis valós idejű adatkommunikációt a robot, a számítógép és a kamera között. Enélkül nem tudható, hogy a jelenlegi robotkar állás alapján milyen értéket vesznek fel a csuklók, mekkora szögekben vannak. Emiatt pedig nem tudjuk azt sem, hogy mihez képest forgatjuk el a robotkar csuklóit a rendelkezésünkre álló szöggel.

Ez a lekérdezés egy valós idejű információ, melyet úgynevezett dictionary vagyis 'szótár' típusú változóba mentünk el. A dictionary egy olyan halmaz mely kulcs-érték párokat tartalmaz kettősponttal elválasztva ezeket ebben a sorrendben. Ez azért kedvező, mert így egyszerűen tudunk hivatkozni a megadott kulcsokra és cserében megkapjuk az értéket, ami hozzá párosul. Mivel a szótárba minden csak egyszer szerepelhet, így, ha újra meghívjuk a programom csak egyszerűen felülírja az előző értékeket. Miután a meghívott ciklus lefut visszalép a program az előző rutinba magától és ott folytatja ahonnan kilépett. A TCP-ben, illetve a Joint-ban a pozíciók szerinti lekérdezést a kulcs helyében a megnevezésekkel, értéként pedig a koordinátákkal méterben és szögelfordulásokkal fokban tartalmazzák.

A változót kiolvasni a programomból többféleképpen lehetséges. Beillesztett ciklusból úgy, hogy a ciklus nevét beimportáljuk a program elején majd a változókat külön utána írjuk, hogy miket importáljon be a jelenlegi programomba. Lehetőség van még arra is, hogy minden változóját meghívjuk és tudjuk használni, ekkor viszont nagyobb odafigyelést igényel az

esetleges megegyező változónevek miatt. Mi azt a megoldást választottuk, hogy a függvény visszatérő értékeként kapjuk vissza a változókat.

A program második fele azzal kezdődik, hogy meghívjuk a pakolás függvényt. Ez történhet gombnyomással a kezelő által, illetve a program által is a sorrendi lista hosszára való paramétert megadva. Ennek a függvénynek a bemeneteként csak a sorrendi listát adjuk meg, ami tartalmazza a koordinátákat és a szögelfordulásokat. A lista egy iterálás során kiolvasásra kerül és az elmentett adatokat egy külön változóba menti. Ezután a koordináta pixel értékeit átszámítva méterbe majd azokat hozzáadva a viszonyítási ponthoz a robotkar odamozog a munkadarab fölé. Ezután lekérdezzük az éppen aktuális *Joint* állásokat majd ehhez hozzászámítva a szögelfordulást, elforgatjuk a hármas csuklóját a robotnak. Újra lekérve a helyzetét a robotkarnak, most TCP szerint már lehet vezérelni a grippert. Ezt követően a megfogó zárása történik meg, ami után elviszi a kar a kért pozícióba a munkadarabot. Itt a megfogó kinyílik és a tárgyat leteszi. Ha az első tárgyat kipakoltuk akkor utána jön sorban a többi. Mindaddig míg az összeset ki nem rakjuk a munkatérre. A program végső tesztelésénél derült ki, hogy túl közelről készítjük a felvételt így bezavar a munkadarabok magassága a koordináták bemérésénél. Mivel nem egy fix kép van, hanem videó, ezért a legjobb megoldásnak az ígérkezett, ha magasabb pontból vételezzünk, mivel így kisebb szög alatt látszódnak a munkadarabok és kevésbé fog bezavarni a magasságuk. A videónak a 640*480 felbontási képéből csak a középső 320*240 pixeles részét vételezzük a kiértékelésnél ahogyan a 22. ábrán is láthatjuk (fekete rész), így a környezet (sraffozott rész) nincsen már benne a felvételben, ami a munkaasztalon kívül esik és ezzel továbbra is ugyan azt a területet figyelhető meg. Ilyen módon, ezt a területet a megfelelő magassággal megválasztva vételeztük ezután a videót. Ettől természetesen a bázis pont megmarad annak, ami előzőleg volt, annyiban változik a program, hogy a pixelértékek átváltásánál kettővel való szorzásra van szükség.



22. ábra Területmeghatározás

Ez azért van mert a mostani, új vételezett területen az x irányban 320, míg y irányban 240 pixel szélességű. Ez azt jelenti, hogy ami eddig 640 pixelre volt a képen az most 320 pixel értéket kap, vagyis a helyes megoldásokhoz kettővel szorzandó. Szükség volt még arra is, hogy a kontúr ellenőrzőben átállítsuk a méret szűrést kisebbre, hiszen a távolság megválasztásával a képkockán lévő kontúrok által határolt terület is kisebb lett. Ezzel a megoldással immár pontosabb a helyzet megállapítás, aminél fogva pontos lesz a kiértékelt kép oldalán lévő tárgy helyzete.

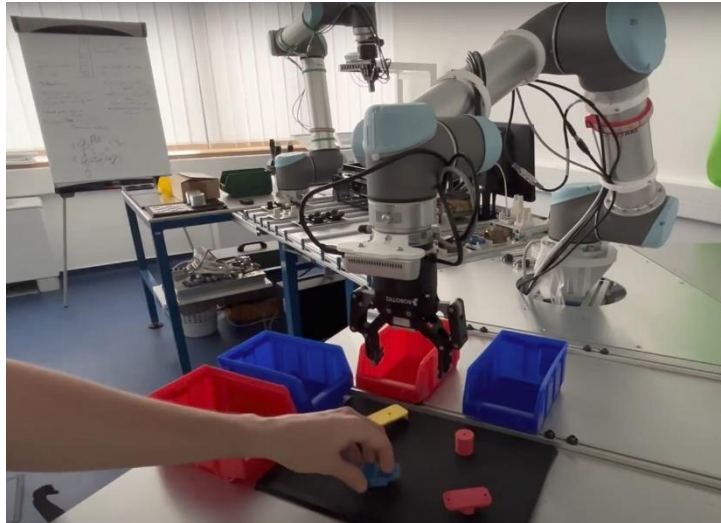
3.3. A rendszer korlátai

A tesztelések során a laboratóriumi megvilágítás mellett tökéletes működést tudtunk elérni a rendszerrel. A HSV színrendszerrel történő szűrés alatt H-ra, és S-re szűrjük a különböző színű munkadarabokat. Azért színre és telítettségre mivel ezek segítségével függetlenek lesznek a fényviszonyoktól és így fényesebb illetve rosszabb fényviszonyok között sem zavarják meg a rendszert.

Jelen feladat tesztelése során a bemutatott 4 munkadarab állt rendelkezésünkre, emiatt lett a bepakolható elemek maximális száma ilyen mennyiségűre korlátozva. Természetesen másik folyamat esetén lehetőség van ezen limit módosítására, illetve a felhasználó képes új, tetszőleges tárgyakat is bevonni a műveletbe. Nincs különösebb alak és szín szerinti megkötés sem. A program oly módon került megírásra, hogy korábban nem használt alkatrészek esetén egy előzetes betanítást szükséges elvégezni. Ehhez elegendő (legalább 10) tanítóképet kell az adott elemről készíteni, és egy nevet adni neki. Ezután a rendszer automatikusan beintegrálja a műveletbe.

Számbeli korlátot csak az szabhat a folyamat során, hogy a munkadarabokat megfelelő távolságra helyezzük el egymástól, mivel a szorosan lévő munkadarabok felvételi stratégiájának a kidolgozása még nincsen teljesen kidolgozva.

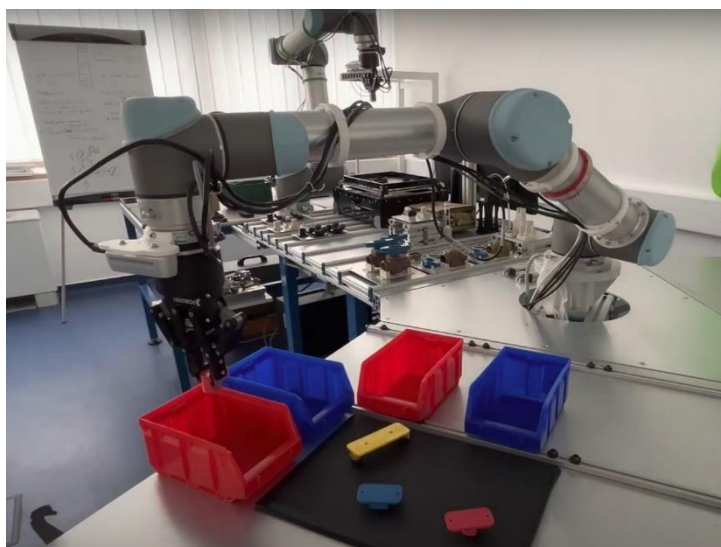
A rendszer működtetése során készített szemléltetőképek a 23-25. ábrákon láthatóak:



23. ábra Bepakolás



24. ábra Munkadarab megfogása



25. ábra Tároló konténerbe helyezés

4. Összefoglalás

Dolgozatunk célkitűzése egy olyan gépi látáson alapuló automata demonstrációs eszköz megalkotása volt, mely egy szortírozási feladatot képes megvalósítani inhomogén munkadarabokon. A demonstrációs eszköz fejlesztése sikeresen megvalósult. Munkánk során a célunkat egy teljes mértékben saját elképzelés szerint, az általunk eddig ismert képfeldolgozási módszerekkel összehangolva oldottuk meg.

Első lépésében a betanítás alapján a munkaterületen véletlenszerűen elhelyezett inhomogén alkatrészek kontúr és szín azonosítása majd osztályba sorolása történt meg. Ebben az esetben az előre betanított négyféle színt és kontúrt sikerült 100%-os hatékonysággal azonosítani. Második lépésben az egymás után a munkatérbe kerülő munkadarabok sorrendiségének azonosítása és rögzítése következett, ami a szortírozás FIFO alapelv szerinti végrehajtásának feltétele. Következő lépésben az automatikus robotprogram generálás valósult meg. Ennek feltétele az előző két pont teljesülése, valamint az egyes munkadarabok középponti koordinátáinak és irányultságának meghatározása, ami a robot megfogási pozícióját fogja meghatározni. A szortírozás közbeni valós idejű (RTDE) adatkommunikációt is végeztünk a robotkarhoz tartozó control box és a program között, az onnan származó adatokat pedig felhasználtuk a robotprogram generálásához. A szortírozás során a szín és kontúr alapján megkülönböztetett munkadarabokat előre meghatározott tárolókba szortírozzuk, ezen paraméterek dinamikusan változtathatók a felhasználó által a programkódba való beavatkozás nélkül, ezzel a rendszer rugalmas alkalmazását elősegítve. Mindezeket összefoglalva sikerült egy olyan demonstrációt fejlesztenünk, ami kollaboratív munkatérben automatikus munkadarab szortírozást végez és a FIFO elvet is követi.

5. Irodalomjegyzék

- [1] E. H. Ostergaard (2017): The role of cobots in Industry 4.0, Universal Robots, White paper
- [2] Varun Gopinath (2016): Risk Assessment Process for Collaborative Assembly – A Job Safety Analysis Approach, 6th CIRP Conference on Assembly Technologies and Systems, pp. 199 – 203
- [3] Varun Gopinath (2017): Safe Assembly Cell Layout through risk assessment – An Application with Hand Guided Industrial Robot, The 50th CIRP Conference on Manufacturing Systems, pp. 430 – 435
- [4] George Michalos (2014): ROBO-PARTNER: Seamless HumanRobot Cooperation for Intelligent, Flexible and Safe Operations in the Assembly Factories of the Future, Conference on Assembly Systems and Technologies and systems, pp. 71 – 76
- [5] F. Sherwani; Muhammed Mujtaba Asad (2020): B.S.K.K. Ibrahim: Collaborative Robots and Industrial Revolution 4.0
- [6] Gunnar S. Bolmsjö (1992): Industriell robotteknik. Studentlitteratur, Lund, 2 edition, In Swedish.
- [7] Lorenzo Sciavicco and Bruno Siciliano (2000): Modelling and Control of Robot Manipulators. Springer Verlag, 2 edition
- [8] John J. Craig (1989): Introduction to robotics mechanics and control. Addison-Wesley Publishing Company, Inc, 2 edition
- [9] R. Hollmann, A. Rost, M. Hägele, A. Verl (2010): A HMMbased approach to learning probability models of programming strategies for industrial robots, 2010 IEEE Int. Conf. Robotics Autom., Anchorage
- [10] D. Hunt (1983): Industrial Robotics Handbook
- [11] V.D. Scheinman (1969): Design of a Computer Controlled Manipulator, Ph.D. Thesis (Stanford Univ., Stanford)
- [12] Martin Hägele, Klas Nilsson, J. Norberto Pires, Rainer Bischof (2016): Handbook of Robotics
- [13] X.-J. Liu, J. Wang (2014): Performance evaluation of parallel mechanisms. In: Parallel Kinematics, Springer Tracts in Mechanical Engineering, (Springer, Berlin, Heidelberg, pp. 185–238
- [14] S. Kock, T. Vitor, M. Björn, H. Jerregard, M. Källmann, I. Lundberg (2011): Robot concept for scalable, flexible assembly automation, Proc. IEEE Int. Symp. Assem. Manuf. (ISAM), Tampere

- [15] P. Gorle, A. Clive (2013): Positive Impact of Industrial Impacts on Employment (International Federation of Robotics, Frankfurt)
- [16] Z. Gao, T. Wanyama, I. Singh., A. Gadhri, R. Sshmidt (2019): From Industry to Robotics 4.0
- [17] Albert S. Olesen, B. B. Gergaly, E. A. Ryberg, M. R. Thomsen, D. Chrysostomou (2021): A Collaborative Robot Cell for Random Bin-picking based on Deep Learning Policies and a Multi-gripper Switching Strategy
- [18] M. Alonso, A. Izaguirre, M. Grana (2018): Current research trends in robot grasping and bin picking, in: The 13th Int. Conf. on Soft Computing Models in Industrial and Environmental Applications, Springer, 2018, pp. 367–376
- [19] T.-T. Le, C.-Y. Lin (2019): Bin-picking for planar objects based on a deep learning network: A case study of usb packs, Sensors
- [20] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu (2018): Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching, in: 2018 IEEE Int. Conf. on Robotics and Automation (ICRA), IEEE, 2018, pp. 1–8
- [21] Thomas Feix, Javier Romero, Carl Henrik Ek, Heinz-Bodo Schmiedmayer, and Danica Kragic (2013): A metric for comparing the anthropomorphic motion capability of artificial hands. IEEE Trans. Rob.
- [22] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S. Sukhatme (2017): Interactive perception: Leveraging action in perception and perception in action. IEEE Trans. Rob.
- [23] Minjie Cai, Kris M. Kitani, and Yoichi Sato (2016): Understanding hand-object manipulation with grasp types and object attributes. In Robotics: Science and Systems
- [24] Lazaros Nalpantidis, Marten ° Bjorkman, and Danica Kragic (2012): Object segmentation: Exploiting camera movement. In Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., pp. 2116–2121
- [25] (2016) Introduction to machine vision – A guide to automating process & quality improvements, Cognex Corporation, USA
- [26] Zancul, E., Martins, H. O., Lopes, F. P., da Silva Neto, F. A. T. V. (2020): Machine vision applications in a learning factory, Procedia Manufacturing, pp.45:516-521.
- [27] Alonso, V., Dacal-Nieto, A., Barreto, L., Amaral, A., Rivero, E. (2019): Industry 4.0 implications in machine vision metrology: an overview, Procedia Manufacturing, pp. 41:359-366

- [28] Nayar, S.K. (January 9. 1990.): Robotic vision system. U.S. Patent 4,893,183,
- [29] Pochyly, A., Kubela, T., Kozak, M., Cihak, P. (2010): Robotic vision for bin-picking applications of various objects. In: Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK), pp. 1–5. VDE
- [30] Saxena, A., Driemeyer, J., Ng, A.Y. (2008): Robotic grasping of novel objects using vision. *Int. J. Robot. Res.* 27(2), pp. 157–173
- [31] Chen, S., Li, Y., Kwok, N.M. (2011): Active vision in robotic systems: a survey of recent developments. *Int. J. Robot. Res.* 30(11), pp. 1343–1377
- [32] Blasco, J., Aleixos, N., Roger, J.M., Rabatel, G., Molto, E. (2002): AE—automation and emerging technologies: robotic weed control using machine vision. *Biosyst. Eng.* 83(2), pp. 149–157
- [33] Kragic, D., Björkman, M., Christensen, H.I., Eklundh, J.-O. (2005): Vision for robotic object manipulation in domestic settings. *Robot. Auton. Syst.* 52(1), pp. 85–100
- [34] Batlle, J., Casals, A., Freixenet, J., Martí, J. (2000): A review on strategies for recognizing natural objects in colour images of outdoor scenes. *Image Vis. Comput.* 18(6–7), pp. 515–530
- [35] Boncyk, W.C., Cohen, R.H. (December 4. 2012.): Image capture and identification system and process., U.S. Patent 8,326,031,
- [36] Ojha, S., Sakhare, S. (2015): Image processing techniques for object tracking in video surveillance—A survey. In: 2015 International Conference on Pervasive Computing (ICPC), pp. 1–6. IEEE
- [37] Lang, S. (2008): Object-based image analysis for remote sensing applications: modeling reality—dealing with complexity, Blaschke, T., Lang, S., Hay, G. (eds.) *Object-Based Image Analysis*, Springer, Berlin, pp. 3–27.
- [38] Nacy, S.M., Abbood, W.T. (2013): Automated surface defect detection using area scan camera. *Innov. Syst. Des. Eng.* 4(8), 1–10
- [39] Boncyk, W.C., Cohen, R.H. (November 19. 2013.): Object information derived from object images. U.S. Patent 8,588,527,
- [40] van Assen, H.C., Egmont-Petersen, M., Reiber, J.H.C. (2002): Accurate object localization in gray level images using the center of gravity measure: accuracy versus precision. *IEEE Trans. Image Process.* 11(12), pp. 1379–1384.
- [41] Burger, W., Burge, M.J. (2008): *Digital image processing: an algorithmic introduction using JAVA*. Springer Science, New York, NY
- [42] Petrescu, S., Gangea, M., Bigioi, P., Pososin, A., Drimbarean, A. (November 8. 2011.):

- Color segmentation. U.S. Patent 8,055,067,
- [43] Xu, W., Mulligan, J. (2010): Performance evaluation of color correction approaches for automatic multi-view image and video stitching, 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 263–270. IEEE
 - [44] Cho, S., Bither, W. (August 20. 2013.): System and method for identifying and classifying color regions from a digital image. U.S. Patent 8,515,163
 - [45] Hanbury, A., Serra, J. (2002): A 3d-Polar Coordinate Colour Representation Suitable for Image Analysis, PRIP. TU Wien, Tech. Rep. PRIP-TR-77
 - [46] Sai Yeshwanth Chaganti, Ipseeta Nanda, Koteswara Rao, Pandi Tavva, GNRSN Prudhvith Niraj Kumar (March 2020): Image Classification using SVM and CNN
 - [47] Golnabi, H., Asadpour, A. (2007): Design and application of industrial machine vision systems, Robotics and Computer-Integrated Manufacturing, pp. 23:630–637
 - [48] LabVIEW Machine Vision and Image Processing Online Course, National Instruments Inc., Austin, Texas
 - [49] Imaging Electronics 101: Camera Types and Interfaces for Machine Vision Applications, Edmund Optics Inc.
 - [50] Jibin, G. J., Arunachalam, N. (2019): Illumination compensated images for surface roughness evaluation using machine vision in grinding process, Procedia Manufacturing, pp. 34:969–977
 - [51] A Practical Guide to Machine Vision Lighting, (2019), National Instruments Inc.,
 - [52] Kumar, V., Kumar, C. P. S. (2020): Investigation of the influence of coloured illumination on surface texture features: A Machine vision approach, Measurement, pp. 152.
 - [53] Wisam T. Abbood, Oday I. Abdullah, Enas A. Khalid, (1 November 2019): A real-time automated sorting of robotic vision system based on the interactive design approach
 - [54] Forgács Zsófia (2020): Gépi látáson alapuló demonstrációs alkalmazások fejlesztése, Multidiszciplináris tudományok, 10. kötet. 4 sz., pp. 56-65
 - [55] Kai Zhou, Zhiwei Meng, Mingyang He, Jialin Hou, Tianhua Li :Design and Test of a Sorting Device Based on Machine Vision, IEEE Access, 2018.
 - [56] Danica Kragic, Joakim Gustafdo, Hakan Karaoguz, Patric Jensfelt, Robert Krug: Interactive, Collaborative Robots: Challenges and Opportunities, IJCAI-18, 2018.
 - [57] Járműgyártási Tanszék honlapja 2022.10.04.
<https://jt.sze.hu/kiber-fizikai-gyartorendszer-laboratorium>
 - [58] Universal Robots honlapja 2022.10.04
www.universal-robots.com

- [59] Robotiq 2F-85 Technical Datasheet 2022.10.04.
<https://robotiq.com/products/2f85-140-adaptive-robot-gripper>
- [60] Intel RealSense D435 honlapja 2022.10.04.
<https://www.intelrealsense.com/depth-camera-d435/>
- [61] Python honlapja 2022.10.04.
python.org
- [62] Mesterséges Intelligencia 2022.10.04.
<https://azure.microsoft.com/hu/resources/cloud-computing-dictionary/what-is-artificial-intelligence/#how>
- [63] Feature vektor 2022.10.04.
<https://www.iguazio.com/glossary/feature-vector/>
- [64] Dr. habil. Horváth András: Digitális képfeldolgozás, egyetemi előadásvázlat, 2022