Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
TMIT Tanszék

Czurkó Dániel

PONTOS TÉRBELI OBJEKTUM LOKALIZÁCIÓ ARUCO KÓDOK SEGÍTSÉGÉVEL

KONZULENS
Dr. Fehér Gábor

BUDAPEST, 2022

Tartalomjegyzék

Absztrakt

Az objektumok számítógépes felismerése már napjaink részévé vált. Megtalálható ez a funkció az okostelefonunkon az okosotthonainkban számos IoT (Internet of Things) eszköz használja, valamint számtalan ipari és mezőgazdasági felhasználása is ismert. Az objektum felismeréséhez kapcsolódó, kevésbé elterjedt funkció a felismert objektum pontos térbeli elhelyezkedésének meghatározása. Ennek az információnak a segítségével a felismert objektumok elhelyezhetők egy térképen, valamint az objektumok automatizált megkeresése is támogatható. A felismert objektumok térbeli elhelyezése fontos szerepet játszhat pl. a mezőgazdaságban ahol a jelentős méretű területeken kell szabadtartású állatokat megkeresni.

A munkám során felhasználtam az aruco kód nevű jelölőket. Ezek olyan fekete-fehér QR (Quick Response) kódokhoz hasonló képek, melyet egy klasszikus kamera is egyszerűen fel tud ismerni. Ennek a felismert aruco kódnak a segítségével egy kis számítási kapacitású számítógép is ki tudja számolni a jelölő és a kamera pontos térbeli viszonyát. Ilyen jelölőket gyakran használnak ipari környezetben robotkarok pontosságának növelésére, pontos beltéri lokalizációra, valamint feltérképező (SLAM - Simultaneous localization and mapping) algoritmusok esetén.

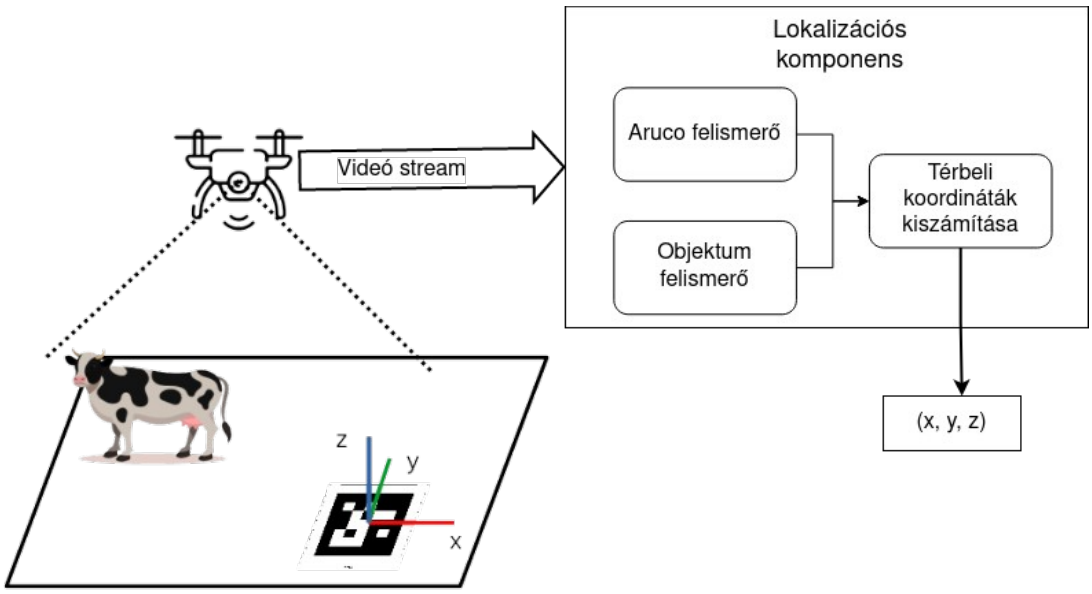
A kutatásom során egy olyan módszert fejlesztettem ki, valamint végeztem el ennek teljesítmény elemzését, mely az aruco kódok és az objektum felismerő algoritmusok segítségével képes meghatározni egy előre definiált objektum pontos térbeli pozícióját. Különböző mérésekkel megvizsgáltam milyen paraméterek befolyásolják a módszerem pontosságát és ezek alapján optimalizáltam a működését. Az eljárásom képes több kamerából, valamint egy mozgó kamerából érkező videofolyam kezelésére is.

Megvizsgáltam más olyan algoritmusokat amelyek több kamera (sztereó kamerák) segítségével vagy egy mozgó kamera (SFM - Structure from motion algoritmusok) segítségével rekonstruálják a háromdimenziós teret. Ezt követően összehasonlítottam ezeket a megoldásokat az általam fejlesztett eljárással, pontosság, gyorsaság és robusztussági szempontból.

1 Bevezetés

Napjainkban egyre elterjedtebbé válnak objektumok felismerését és térbeli elhelyezését felhasználó alkalmazások. A kiterjesztett valóságnál értelmezni kell a környezetet: milyen akadályok, objektumok vannak benne, ezek hol, milyen pozícióban helyezkednek el és ezt felhasználva kell megvalósítani a kívánt feladatot. Egy kiterjesztett valóságot használó szobatervező alkalmazásnál pontosan meg kell határozni, más objektumokat, bútorokat is figyelembe véve, hogy hova kerüljön a virtuális asztal vagy kanapé. Fontos szerepe van az objektum felismerésnek és az objektumok térbeli elhelyezésesének mezőgazdasági felhasználás során is, ahol hatalmas területeken kell szabadtartású állatokat megfigyelni.. Ilyenkor fel kell ismerni az állatokat, esetleg a nemüket, korukat és meg kell határozni a helyzetüket is, hogy követni lehessen őket a könnyebb megfigyelés érdekében Hatalmas építkezéseknél is hasznos lehet egy olyan rendszer, mely felismeri és egy térképen számon tartja az egyes építkezési eszközök pontos helyzetét. Így, a munkásoknak nem kell folyamatosan fejben tartani, mit hol tartanak, valamint segít optimalizálni ezek mozgatásának tervezését, hogy mindig a legoptimálisabb helyen legyenek.

Munkám során egy olyan módszert fogok bemutatni, mely képes egy tetszőleges objektum térbeli pozíciójának meghatározására egy kamera és egy különleges jelölő, aruco kód (további információk az aruco kódról a 2.2 alfejezet) segítségével. A módszer bármilyen kamerával képes működni mindaddig, amíg a kamera az objektumra és az aruco kódra is jól rálát. A módszer három fő komponensből épül fel. Az aruco kód felismerő komponensből, az objektum felismerő komponensből és a térbeli koordinátákat meghatározó komponensből. Ezek láthatók az alábbi 1 ábrán. Egy kamera, (a lenti példán ez egy drónra van felszerelve) készít egy videofelvételt az objektumról (a lenti példában ez a tehén) és az aruco kódról. Ezt a felvételt elküldi a lokalizációs komponensnek, mely három alkomponensből épül fel, az aruco felismerő, az objektum felismerő és a térbeli koordináták kiszámításáért felelős részekből. A videofelvételt az aruco felismerő és az objektum felismerő komponensek párhuzamosan feldolgozzák és a kiszámított adatokat átadják a térbeli koordináták kiszámításáért felelős komponensnek. Ez a komponens kiszámítja az objektum (a példában a tehén) térbeli koordinátáját az aruco kód koordináta-rendszerében. A TDK dolgozatom során a lokalizációs komponensnek a pontos működését fogom ismertetni, majd mérésekkel megvizsgálni a pontosságát.



1. ábra: A módszer működésének vázlatos leírása

1.1 Dolgozat felépítése

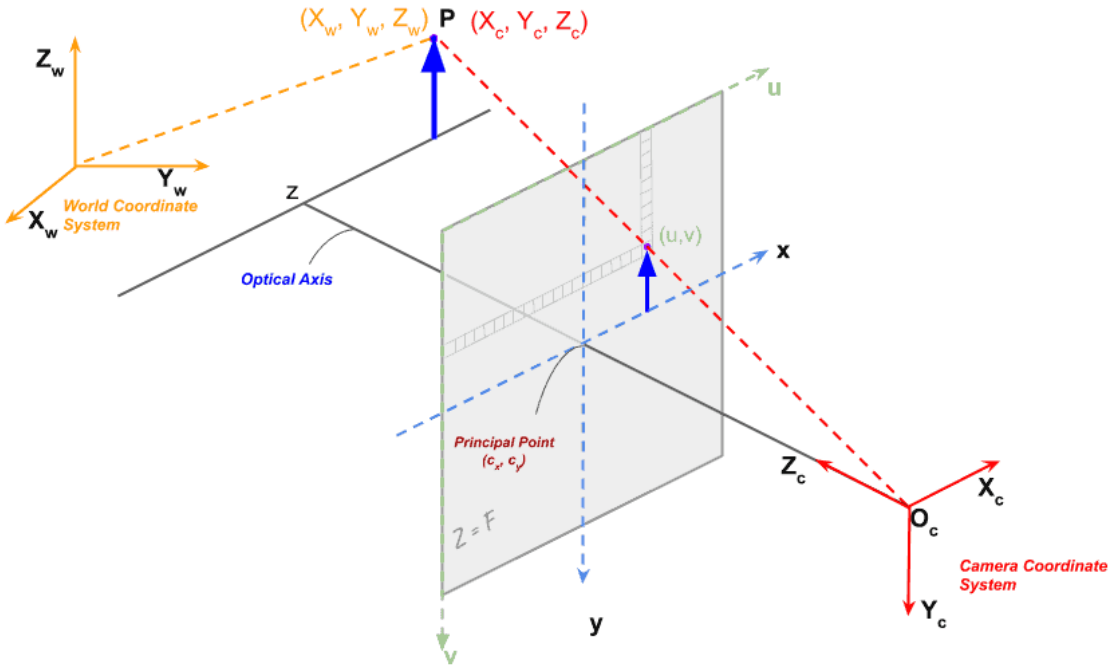
A TDK dolgozatom során először az elméleti hátteret, algoritmusokat és a használt eszközöket fogom bemutatni majd ismertetek más módszereket melyek objektumok térbeli lokalizációját teszik lehetővé. Ezt követően a tervezés szakaszban vázolom a módszer elméleti működését és a megvalósítás szakaszban leírom, hogy ezt, hogyan valósítottam meg a gyakorlatban. Végül a mérés fejezetben bemutatom méréseken keresztül a módszerem pontosságát és lezárásképpen a konklúzió szakaszban összegzem az eredményeket és megemlítek továbbfejlesztési lehetőségeket is.

2 Irodalomkutatás

2.1 Kamera modell bemutatása

A módszerem elkészítése során kiindulási alapnak a pin hole kamera modellt használtam fel, mely térbeli objektumok kameraképre történő leképzését írja le. Ez a modell segít abban, hogy meghatározhassuk egy térbeli objektum koordinátái mely kamerakép koordinátáknak fognak megfelelni, más szóval hogyan tudjuk a térbeli pontokat egy síkra a kameraképre vetíteni.

Az alábbi 2. ábrán látható, hogy lehet a háromdimenziós P pontot leképezni a kamerakép (u, v) pontjára. Ez a leképzési folyamat két nagy lépésből áll. Első lépésnek át kell alakítani a háromdimenziós P pont világ koordináta rendszerbeli (a képen World Coordinate System) koordinátáit (X_w', Y_w', Z_w') a kamera koordináta rendszerébe (X_c', Y_c', Z_c'). Ez a folyamat egy eltolás (t vektor) és egy térbeli forgatás (R mátrix) segítségével tehető meg. Ha az eltolást végző vektort és a forgatásért felelős 3x3-as mátrixot egy 3x4-es mátrixba (R|t) írjuk le, ahol az első 3 oszlop a forgatási mátrix az utolsó oszlop pedig az eltolási vektor, akkor megkapjuk az Extrinsic mátrixot (külső mátrix).



2. ábra: Háromdimenziós pont leképzése kameraképre [2]

Második lépésben a kamera koordináta rendszerében lévő objektumot kell leképezni a kamera képre. Ez két kisebb lépésből tehető meg. Először háromszög hasonlósággal és a fókusztávolság (f) segítségével meg határozható a háromdimenziós pont (x, y) hozzávetőleges helye. A 2 ábra jelöléseit használva az alábbi egyenletek (1) írhatóak fel a fókusztávolság és a háromszög hasonlóság segítségével. Innen egyszerű átrendezéssel kiszámítható x és y.

$$\frac{x}{f} = \frac{X_c'}{Z_c'}$$
$$\frac{y}{f} = \frac{Y_c'}{Z_c'}$$

1. egyenlet: Kamera kép koordináták meghatározása

A kamera szenzorában különböző kisebb hibák lehetnek, ezért a fókusztávolságot érdemes további paraméterekkel kiegészíteni. A pixelek a kamera szenzorán nem mindig teljesen négyzet alakúak így bevezettek két fókusztávolságot f_x és f_y néven. A kamera optikai közepe sem esik minden esetben egybe a kép koordináta rendszerével. Ennek kijavítására bevezették a c_x és c_y paramétereket. Végül, lehet egy kis ferdeség a szenzor koordinátaiban is ezt egy γ paraméterrel lehet korrigálni. A korábban felsorolt paraméterek is felírhatóak mátrix alakban ez látható az alábbi egyenletben:

$$K = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

2. egyenlet: Kameramátrix

Ez a mátrixot Intrinsic (belső) mátrixnak vagy kamera mátrixnak nevezik és K betűvel szokás jelölni.

A K mátrixot jobbról beszorozva a korábban bemutatott Extrinsic mátrixszal, megkapjuk a projekciós mátrixot. Ez a projekciós mátrix lesz az, ami megadja az összerendelést a P pont (X_w', Y_w', Z_w') világkoordinátái és a kamerakép (u, v) koordinátái között. Más szóval egy tetszőleges P pont világkoordinátáját homogén koordináta rendszerben balról beszorozva a projekciós mátrixszal, megkapjuk a P pont kamera képen lévő koordinátáját (Valójában itt még szükséges egy osztás a kamera koordináták meghatározásához).

$$Proj \cdot \begin{bmatrix} X_w' \\ Y_w' \\ Z_w' \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

3. egyenlet: A képi koordináták meghatározása a világkoordinátákból

A Proj a projekciós mátrix amit a korábban leírtak alapján az alábbi módon lehet meghatározni (4 egyenlet):

$$Proj = K \cdot (R|t)$$

4. egyenlet: A projekciós mátrix

A P pont kamera képen lévő (u, v) koordinátája a korábban (3 egyenletben) meghatározott (x, y, z) számokból az alábbi osztás segítségével határozható meg:

$$(u, v) = (x/z, y, z)$$

5. egyenlet: A kamera koordináták meghatározása

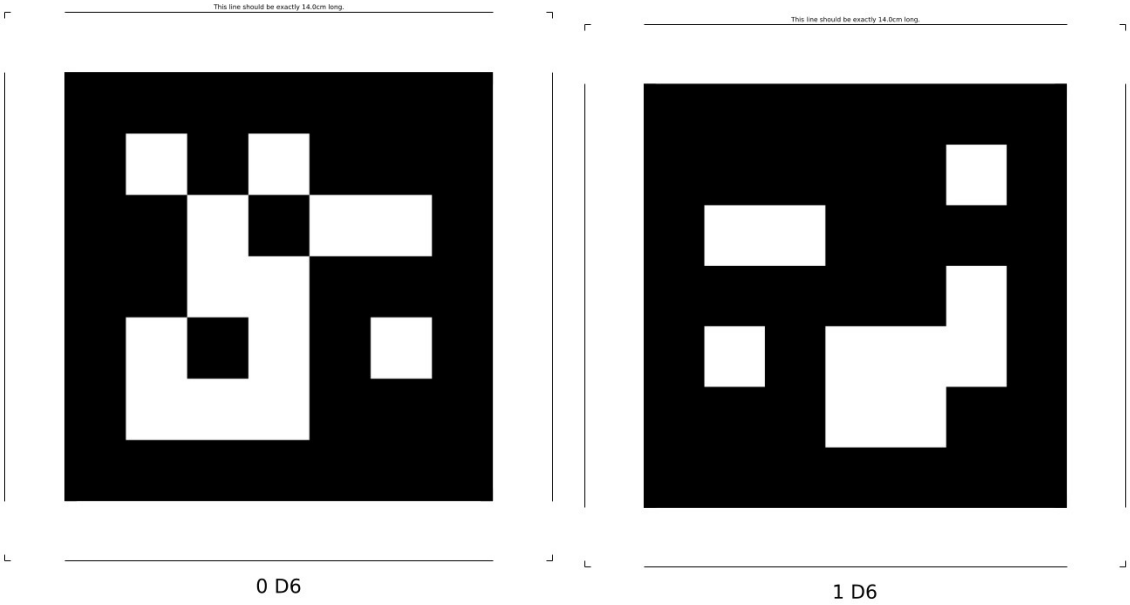
A tématerületről további információk a Learn OpenCV 3 könyv [1] 18-ik fejezetében és a learnopencv weboldalon[2] olvashatóak.

A módszerem segítségével a fentebb bemutatott folyamatnak pont a fordítottját valósítottam meg, vagyis a kamerakép pontjait alakítottam át térbeli pontokká. Azonban ez nem egy triviális feladat. Ha csak a kamerakép egy pontját visszavetítjük a térbe, egy egyenest kapunk nem egy pontot. Ahhoz, hogy meg tudjuk határozni, hogy a kamerakép egy pontja melyik pontnak felel meg az egyenesen, szükséges egy másik kamerakép is ugyanarról a környezetről, csak egy kicsit más pozícióból. Két kamerakép segítségével már ki lehet számolni a térbeli pontot. Ezt a módszert nevezik sztereó látásnak. A sztereó látás gyakorlati megvalósítása során további problémák lépnek fel. Meg kell határozni a két kamera relatív pozícióját és a kameraképeken a közös pontot, amelynek a térbeli megfelelőjét szeretném meghatározni. A módszeremnél ezeket a problémákat az obejektumfelismeréssel és az aruco kódokkal orvosoltam. A következő fejezetben az aruco kódokat fogom részletesen ismertetni.

2.2 Aruco kódok bemutatása

Az aruco (az Augmented Reality University of Cordoba szavakból áll össze) kódok [3] a fiducial [4] markerekhez (jelölőkhöz) tartozó, QR kódokhoz hasonló fehér négyzetek egy fekete háttéren és fekete szegéllyel (a fekete szegély a felismerést segíti). Az aruco kódok mérete a felhasználástól függően változhat. Ha több aruco kód szükséges az adott felhasználáshoz akkor ehhez nagyobb kódok szükségesek, míg ha kevesebb kód is elég akkor érdemes kisebb kódokat használni amiket robusztusabban fel lehet ismerni. Az aruco kódokat gyakran alkalmazzák a kiterjesztett valóságot használó projekteknél, valamint a robotok is igénybe veszik ezeket a kódokat a pontos térbeli pozicionáláshoz. Két ilyen aruco kód látható az alábbi 3 ábrán. Egy aruco kód

felismerő algoritmus pontosan fel tudja ismerni ezeket a kódokat egy kameraképen és meg tudja határozni a bele kódolt számot. Ezt követően ugyanez az algoritmus a felismert kód négy sarkának az ismeretében képes kiszámolni a kamera térbeli pozícióját az aruco kódhoz képest (ehhez a kamera előzetes kalibrálása is elengedhetetlen). Az algoritmus a kamera aruco kódhoz viszonyított térbeli pozícióját egy eltolási vektorral és egy fogatási mátrixszal írja le. Ezt az eltolási vektort és forgatási mátrixot használtam fel később a projekciós mátrix Extrinsic mátrixának elkészítéséhez (a 2.1 fejezet alapján).



3. ábra: A 6. dictionary-be tartozó aruco kódok

Az aruco kódok csoportokba, dictionary-kbe (szótárakba) rendezhetőek. Azok az aruco kódok kerülnek egy dictionary-be, amelyeknek a legnagyobb a Hamming-távolsága [5]. A Hamming-távolság azonos méretű bináris kód eltérő bitjeinek a számát jelenti. Az aruco kódok felismerése előtt mindig meg kell adni a felismerő algoritmusnak, hogy melyik dictionary-be tartozó aruco kódokat szeretnénk felismerni. Az aruco felismerő algoritmus csak ezekben a dictionery-kbe tartozó aruco kódokat fogja felismerni, másik dictionary-be tartozó aruco kódokat pedig nem. Ezek a dictionary-k nagyon hasznosak tudnak lenni olyan szituációkban, ahol több kamerát is használunk egyszerre és azt szeretnénk ha a kamerák csak bizonyos kódokat ismerjenek fel és a többit hagyják figyelmen kívül . Az aruco kód felismerő algoritmusnak a dictionary-n kívül meg lehet adni a kinyomtatott aruco kód szélességét, méterben. Ennek az információnak és a kamera fizikai paramétereinek (tulajdonképpen a K Intrinsic mátrix) a birtokában a felismerő algoritmus nagyon pontosan meg tudja határozni a kamera térbeli pozícióját az aruco kódhoz képest.

A munkám során aruco kódokat használtam a kamera pontos térbeli pozíciójának és a projekciós mátrixának a meghatározására.

2.3 Objektumfelismerő algoritmus

Az objektumok felismerése egy komplex feladat. Fel kell ismerni egy képen, hogy melyik pixelek tartoznak egy objektumhoz, majd ha felismertünk egy objektumot, el kell dönteni, hogy milyen objektumot ismertünk fel. Nehéz megérteni a probléma komplexitását, hiszen ez a feladat nekünk, embereknek triviális. A probléma bonyolultsága és sokszínűsége miatt érdemes gépi tanuláson alapuló mély tanuló algoritmusokat használni. A mély tanuló algoritmusok során egy neurális hálót használunk, hogy az algoritmus a bemenethez határozza meg a legvalószínűbb kimenetet.

A gépi tanuláson alapuló objektum felismerő algoritmusok körében három elterjedt megoldást ismerünk:

- R-CNN (Region proposal Convolutional Neural Network) [6]
- YOLO (You Only Look Once) [7]
- SSD (Single Shot multiBox Detector) [8]

Az YOLOv4-et [7] leíró cikkben található összehasonlítás alapján az R-CNN igen lassú, míg a YOLO, illetve az SSD gyorsak. Ha a pontosság értékeket vesszük figyelembe akkor viszont az SSD igencsak pontatlan míg a YOLO és a Faster R-CNN hasonlóan pontosak. Ezek alapján a YOLOv4 algorimus egy optimális megoldás, és ezt választottam a munkám során az objektumfelismerés megvalósításához.

A YOLO a „You Only Look Once”, mint „csak egyszer nézed meg”, szavak kezdőbetűiből áll össze. Ez az objektumfelismerő algoritmus egy egylépcsős objektumfelismerő. A YOLO első verzióját 2015-ben mutatták be Joseph Redmon, Santosh Divvala, Ross Girshick és Ali Farhadi [9]. A cikkben leírtak alapján GPU segítségével képes 45 képkockát feldolgozni másodpercenként. Ez nagyon gyorsnak számít. Később megjelentek újabb verziói a YOLO-nak. A YOLO9000 [10] (~YOLOv2), mely képes 9000 objektum kategóriában felismerni objektumokat bár nem igazán pontos, a 2018-ban bemutatott YOLOv3 [11] mely az előzőekhez képest egy nagyobb modellt használ, de pontosabb. A YOLOv3-at követően Joseph Redmon kilépett a YOLO fejlesztéséből, mivel a munkáját etikátlan célokra használták. A fejlesztést Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao vették át és 2020 áprilisában megjelent a YOLOv4 [12]. Ezt követően sorra jelentek meg újabb és újabb YOLO verziók. Jelenleg a legújabb verzió a 2022 júliusában megjelent YOLOv7[13, o. 7]. Megvizsgáltam a YOLOv7-et és minimális az eltérés a YOLOv4-hez képest így a módszerem kifejlesztéséhez a bevált YOLOv4-et használtam.

2.4 Használt eszközök

A munkám során egyszerű, kereskedelemben is kapható eszközöket használtam. A videofelvételt egy kis drón végezte és az objektumnak egy egyszerű játéktehenet választottam Az aruco kódokat kinyomtatás után

csempékre ragasztottam, hogy tartósabbak legyenek, míg a számításokat egy a tanszéken található szerver végezte. Ezekkel az eszközökkel szimuláltam egy valós környezetet, ahol egy drón valós teheneket ismerne fel és lokalizálná egy olyan környezetben, ahol aruco kódokat is elhelyeztek. A valós környezetben természetesen nagyobbak a tehenek, viszont a drón mind a valós, mind a szimulált környezetben majdnem ugyanakkorának fogja látni őket. A valós környezetben sokkal távolabbról készíti a drón a felvételt a nagyobb tehénről. A tehénhez hasonlóan az aruco kódnak is nagyobbnak kell lennie a valós környezetben, hogy távolról jól lássa a drón. A következő részben a használt drónt és a játéktehenet fogom röviden bemutatni.

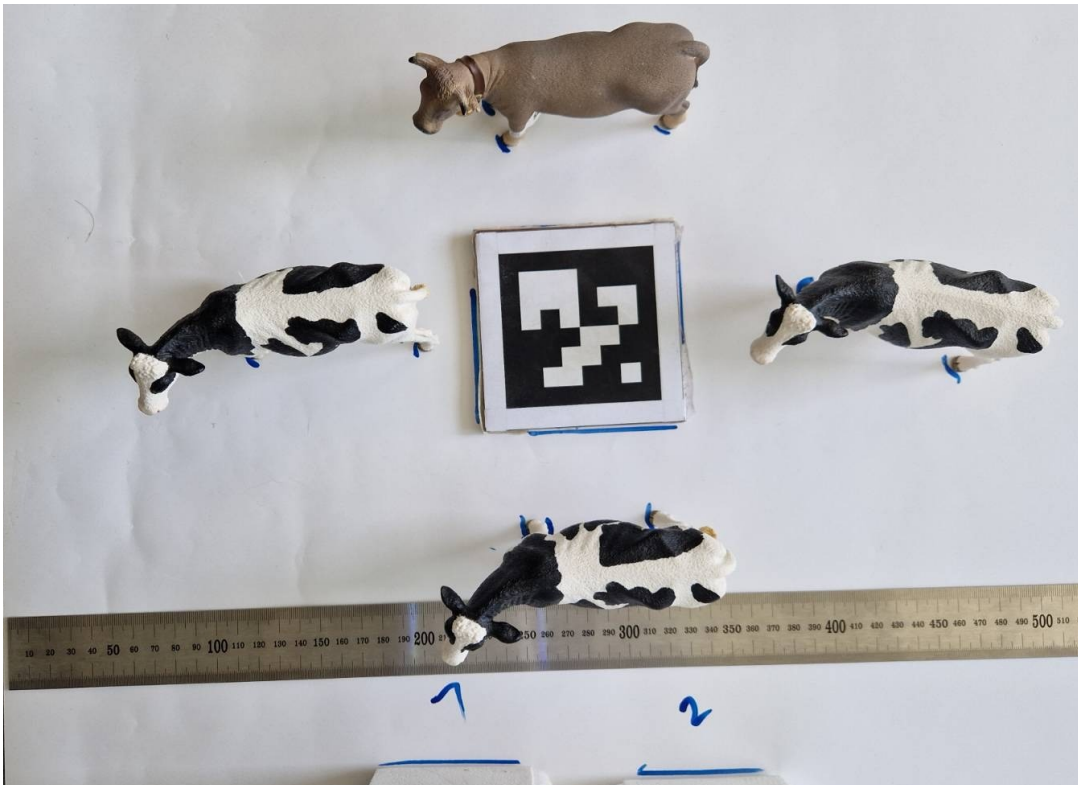
2.4.1 Drón

Az alábbi 4 ábrán látható Yuneec Mantis Q drónt használtam fel kutatásaim során a videofelvétel elkészítéséhez. Későbbiekben a dolgozatomban használt Yunnec Mantis Q drónra csak Mantis-ként fogok hivatkozni. Ez egy kereskedelemben is kapható drón, ami a földi irányító központtal (GCS, Ground Control Station-ek nevezik) egy nyílt forráskódú, Mavlink [14] nevű protokollon keresztül kommunikál. Ennek a nyílt Mavlink protokollnak a segítségével nemcsak a telefonos alkalmazással képes kommunikálni a drón, hanem egy számítógépről vagy akár a felhőből is lehetőségünk nyílik a drón irányítására. Mantis egy 1280x720 felbontású kamerával rendelkezik és az alacsony késleltetésű videofolyamot egy RTSP [15] protokollon keresztül biztosítja számunkra. Ezt a videofolyamot használtam az aruco kódok és objektum(ok) felismerésére, valamint a pozíció(k) kiszámítására.

4. ábra: Yunnec Mantis Q drón

2.4.2 Tehén

Felismerendő objektumnak egy kis játék tehenet választottam. Ebből a tehénből látható négy darab az alábbi 5 ábrán. Ezt a tehenet nagy pontossággal felismeri a YOLOv4 objektumfelismerő, valamint a tehén mint állat jól kapcsolódik az objektumfelismerés és lokalizáció esetleges mezőgazdasági felhasználásának témájához.



5. ábra: Az objektumfelismeréshez használt játéktehenek

A tehenek és a körülöttük lévő környezet értelmezésére számos technológia létezik, mindegyiknek meg vannak a sajátosságai és a jellemző felhasználási területei a következő fejezetben ezeket az eljárásokat és eszközöket fogom bemutatni.

2.5 Algoritmusok

2.5.1 SFM

Az SFM (Structure from motion) egy olyan technika, melynek segítségével rendezetlen képekből lehet háromdimenziós struktúrákat (objektumokat, városokat) rekonstruálni. Az algoritmus alapját az a jelenség adja,

hogy ha mozgunk a hozzánk közelebb lévő objektumok úgy tűnnek, mintha sokkal gyorsabban mozognának, mint a tőlünk távol lévők. Ezt a jelenséget motion parallax-nak (mozgási parallax) is szokták nevezni és minket, embereket is ez a jelenség segít két szemünk nyújtotta sztereó látás mellett, hogy jobban meg tudjuk határozni az egyes objektumok távolságát, mélységét.

Az SFM algoritmus működése során a képeken különböző „jellegzetességeket” (features angolul) határoz meg, majd ezeket követi a képeken és ezekből számolja ki a kamera térbeli pozícióiját, mozgását, valamint a képekből összeállított háromdimenziós felületet. A legelterjedtebb jellegzetesség kereső algoritmusok a SIFT (scale-invariant feature transform) [16] és a SURF (speeded-up robust features) [17] A jellegzetességek megtalálását követően össze kell őket párosítani. Az egyik képen elhelyezkedő jellegzetesség melyik jellegzetességnek felel meg a másik képen. Ennek a problémának a megoldására általában a KLT (Kanade–Lucas–Tomasi) feature tracker [18] (jellegzetesség követő) algoritmust szokták használni. Többfajta SFM algoritmust is megkülönböztetnek annak függvényében, hogy hány képen és hogyan végzi az összepárosítást. Az incremental SFM [19] esetén egyesével történik a képek feldolgozása, a global SFM [20] esetén az összes képet egyszerre dolgozza fel, míg a hierarchical SFM [21] esetén a képeket egy fába rendezi és hierarchikusan végzi el a háromdimenziós felület rekonstrukcióját. Az összepárosított jellegzetességek között sajnos lehetnek hibás párosítások is így ezeken szokás egy szűrést elvégezni. A szűréshez leggyakrabban a RANSAC [22] nevű algoritmust használják. Ezek alapján látható, hogy az SFM algoritmus több lépésből és számos algoritmus együttműködésével valósul meg.

Az SFM-et gyakran használják egy objektum háromdimenziós másának elkészítésére, nagy területek, valamint épületek digitális másának megalkotására. A tématerületről további információk a Learn OpenCV 3 könyv [1] 19-ik fejezetében olvashatóak

2.5.2 Sztereó kamera

Mélységi információ kinyeréséhez nemcsak a mozgást, hanem a kamerák számának növelését is fel tudjuk használni. Nem egy, hanem kettő, vagy több, egymástól fix és kis távolságra elhelyezett kamerával készítünk felvételt ugyanarról az objektumról vagy környezetről. Az így elkészített képeket egy algoritmus megvizsgálja, megkeresi az egyező részeket, meghatározza, hogy mennyivel vannak egymástól eltolva a képen az egyező részek (ez a távolság angolul a disparity) és ez alapján kiszámolja a kamerától mért távolságukat. Az emberi agy is hasonló folyamatok segítségével találja ki, hogy milyen távolságra van egy objektum, amit a két szemünkkel látunk További információk az emberi látás távolság érzékeléséről az alábbi Foundations of Vision című könyvben olvashatóak [23]. Az élőlényeknél Stereopsis néven is megjelenik az a folyamat, amely segíti az állatokat a mélységi információ kinyerésében.

A gépi látás során még további előkészítő lépések is szükségesek mielőtt el lehetne kezdeni a képek összehasonlítását és a mélységi információk kiszámítását. Első lépésben el kell távolítani a kamerakép torzításait (mind a hordó, mind a tangenciális torzításokat) melyeket a lencsék okoznak. A második lépésben át kell transzformálni a képeket, hogy egy közös síkon helyezkedjenek el (ez a folyamat angolul az image rectification). Ezen lépések után lehet csak a képeket összehasonlítani és a távolság információkat később pedig egy pontfelhőt kiszámítani. A sztereó kamerát az algoritmusok részhez tettem, mivel két egyszerű kamerával bárki építhet magának sztereó kamerát, de ezen kívül kész sztereó kamera szenzorokat is lehet venni amelyek mindent maguktól elvégeznek és csak a mélységi információt adják vissza nekünk.

Számos robotot segítenek sztereó kamerák is az objektumok detektálására és elkerülésére. Ilyenek a különböző AGV-k (Automated Guided Vehicle, automatikusan irányított jármű), melyeknek nagy gyárakban, raktárakban kell eligazodniuk, navigálniuk. A robotkarok melyeknek önállóan kell kikerülniük a mozgásukat esetleg akadályozó más objektumokat. Valamint a Marson mozgó Perseverance Rover [24] is stereo látást használ az ismeretlen környezetben történő a navigációra. A kutatásaim során vizsgált módszer is a sztereó látás elméletének alapját használja, vagyis, ahhoz hogy egy pont térbeli koordinátáját meghatározhassuk legalább két kamera szükséges.

A tématerületről további információk a Learn OpenCV 3 könyv [1] 19-ik fejezetében olvashatóak.

2.6 Szenzorok

Mélységi információ nemcsak kamerával és algoritmusokkal nyerhető ki hanem más szenzorokkal különleges kamerákkal is. Ilyenek például a ToF (time-of-flight) szenzorok, structured light kamerák. A következő fejezetekben ezt a két típust, illetve a ToF szenzorok legelterjedtebb megvalósítását a LIDAR-t fogom bemutatni.

2.6.1 ToF

A ToF (Time-of-flight) szenzorok működésük során kibocsátanak egy fénysugarat, majd megmérik, hogy ez a fénysugár mennyi idő alatt jut vissza az érzékelőbe és ebből az időből, valamint a fény ismert sebességéből tudnak a távolságra következtetni. A fénysugarat általában egy LED vagy pedig egy lézer bocsátja ki. A fény kibocsátó és az érzékelő eszköz legtöbbször egymás mellett egy eszközben helyezkednek el.

Megkülönböztetnek direct ToF (dToF) és indirect ToF (iToF) kamerákat. A dToF esetén egy fény pulzust használnak és megmérik, hogy a pulzus kiadásától mennyi idő telt el a pulzus visszaérkezéséig A iToF esetén pedig egy folyamatos modulált, kódolt fénysugarat bocsájt ki a szenzor, és a visszaérkező folyamatos fény fázisát vizsgálják meg, hogy mennyit változott a kibocsájtott fénysugár fázisához képest. A dToF

legelterjedtebb implementációja a LIDAR míg a iToF technológia a flash-based (felvillanás alapú) kameráknál (vagy Flash Lidar ToF kamerának is szokták nevezni) jelenik meg. A flash-based kamerák nem egy darab érzékelőt, hanem a kameráknál is alkalmazott érzékelő mátrixot használ a kibocsájtott, modulált fénynek az érzékelésére. Az érzékelő mátrix minden pixele külön megméri a fénye utazási idejét, így egy darab lépésben (egy „fényképezéssel”) képesek az egész környezetet letapogatására. A NASA is ilyen elven alapuló szenzort használ a leszálló egységeiben [25] a biztonságos leszállás biztosításra. A következő fejezetben a dToF legismertebb fajtáját a LIDAR-t fogom bemutatni.

2.6.2 Lidar

A LIDAR a LIght Dtection And Ranging angol szavakból áll össze (radar, RAdio Detection And Ranging mintájára) és magyarul lézer alapú távérzékelésnek lehetne lefordítani [26]. Működése nagyban hasonlít a radaréhoz csak míg a radar esetén elektromágneses hullámot bocsátunk ki és annak a visszaverődését érzékeljük és dolgozzuk fel addig a LIDAR esetén fényt, lézert bocsátunk ki és a lézerfény visszaverődését érzékeljük, majd dolgozzuk fel. Mivel fényt használunk így kisebb az a távolság, amit pontosan meg tudunk mérni. A LIDAR-oknak egy fontos tulajdonsága, hogy mekkora szögben képes megmérni az objektumok távolságát, illetve hogy csak síkban vagy térben is képesek mérni. A LIDAR-okat gyakran használják a robotok feltérképezésre lokalizálásra, valamint objektum elkerülésre. (Az újabb Apple [27] készülékek is rendelkeznek LIDAR-al így látható, hogy a LIDAR már mindennapjaink részévé kezd válni.)

2.6.3 Structured light.

A Structured light szenzorok hasonlóan működnek a ToF kamerákhoz, csak egyszerűbbek. Egy különleges mintát vetítenek a környezetükre, majd egy egyszerű kamera segítségével megvizsgálják hogyan változott meg a minta a környezetre vetítés után és ez az információ alapján határozzák meg a mélységet, rekonstruálják a helyszínt. A különleges minta lehet egy pontháló, csíkok vagy bármilyen más akár színes mintázat. Ezt a mintát a kamera mellett egy előre ismert pozícióban elhelyezett projektor biztosítja.

Gyakran használják ezt a Structured light technológiát objektumok, arcok [28], [29] háromdimenziós szkennelésére. Az Apple Iphone telefonjainak FaceID-ja [30] is a Structured light technológián alapul.

2.6.4 Összehasonlítás.

A korábban bemutatott technológiák közül a sztereó kamerát, a Structured light és a ToF technológiákon alapuló szenzorok egy konkrét megvalósítását fogom röviden összehasonlítani. Ez az összehasonlítás látható az alábbi 1 táblázatban.

1. Táblázat: Különböző térbeli látásra alkalmas szenzorok összehasonlítása [31]

Név	**Típus**	**Mélység hatótávolsága**	**3D felbontás**	**Képkockasebesség**	**Látómező (H: horizontális V: vertikális D: diagonális)**
Microsoft Azure Kinect	ToF	0.25 - 5.46 m	Narrow mód: 6540x576 Wide mód: 1024x1024	30 fps	Narrow mód - 75° H, 65° V; Wide mód - 120° H, 129° V
Orbbec Astra Mini	Structured Light	0.6 - 5.0 m	640 x 480	30 fps	73° D, 60° H, 49.5° V
MYNT EYE	Sztereó kamera	0.5 to 18 m	752x480	60 fps	146° D, 122° H, 76° V

A fenti 1 táblázatban lévő egyes konkrét termékek mellett más paraméterekkel rendelkező termékek is elérhetőek az adott típusból. A táblázatba olyan konkrét termékeket választottam, amelyek megfelelően reprezentálják az adott típust. A táblázatban látható, hogy fél méter alatti távolságban a ToF szenzorok működnek csak jól nagy, hat méter feletti távolságban csak a sztereó kamerák használhatóak. Térbeli felbontás tekintetében a Structured light és a sztereó kamerák hasonlóan viselkednek, a ToF szenzorok, viszont sokkal nagyobb térbeli felbontással rendelkeznek. Gyorsaság és a látómező tekintetében a sztereó kamerák rendelkeznek nagyobb teljesítménnyel, valamint nagyon megbízhatóan működnek a környezeti behatások esetén [32].

3 Tervezés

Az munkám során tervezett módszernek a célja, hogy képes legyen egy egyszerű kamera, esetleg kamerák és aruco kódok segítségével meghatározni egy előre kiválasztott objektum térbeli pozícióját. Az OpenCV biztosít számunkra egy triangulatePoints [33] nevű függvényt, mely háromszögeléssel képes a kameraképen elhelyezkedő pontok térbeli megfelelőjét meghatározni. Ehhez csak a kameraképek összetartozó pontjaira és a kameraképekhez tartozó projekciós mátrixra van szüksége.

A kameraképeken az összetartozó pontok megkeresését, a képeken közös objektumok felismerésével lehet megoldni. Az objektumokat a 2.3 fejezetben bemutatott YOLOv4 objektumfelismerő algoritmus fogja felismerni és meghatározni az objektum köré rajzolható határoló négyzetet (bounding box angolul). Ennek a határoló négyzetnek a közepe fogja az objektumot reprezentálni a számítások során. Az egyes felvételeken megtalált objektum középpontok lesznek az összetartozó pontok az OpenCV triangulatePoints függvényéhez. Abban az esetben lehet probléma, ha ugyanabból az objektumból több is megtalálható a képen. Ilyen esetben bonyolult képfeldolgozási feladat összekapcsolni az összetartozó objektumokat a kamerák tetszőleges és esetleg időben változó pozíciója miatt. E probléma megoldása nem része a dolgozatnak. A módszeremet úgy terveztem, hogy csak egy objektum lesz jelen az adott fajtából.

Tehát összességében azt a térbeli pontot szeretném megkeresni a triangulatePoints függvénnyel, amely a kameraképen az objektumfelismerő által felismert középpontba képződne le a kép készítése során.

Az objektum közepének megfelelő térbeli pont megtalálásához elengedhetetlenül fontos a kamerafelvétel projekciós mátrixának a meghatározása, mivel a projekciós mátrix egy fontos paramétere a triangulatePoints függvénynek. A projekciós mátrix az Extrinsic és az Intrinsic mátrixból áll össze. Az Intrinsic mátrix könnyen meghatározható kamera kalibrálással (Learn OpenCV 3 könyv [1] 18-ik fejezetében), az Extrinsic mátrixhoz szükséges eltolási vektort és forgatási mátrixot pedig az aruco felismerő algoritmus biztosítja. Ezek alapján az is látható, hogy elegendő lesz egy aruco kód a módszer megvalósításánál. Az Extrinsic és az Intrinsic mátrix ismeretében 2.1 alfejezet alapján már minden információ a rendelkezésünkre áll, hogy mátrix szorzással meg tudjuk határozni az adott kameraképhez tartozó projekciós mátrixot. A triangulatePoints függvény a kiszámított térbeli koordinátákat az aruco kód koordináta-rendszerében fogja megadni, mivel az aruco kód projekciós mátrixát használtam.

Következő fontos kérdés, hogy ezt a problémát egy kamerával is meg lehet e oldani vagy pedig több kamera szükséges a térbeli pont megkereséséhez. Egy kamera sajnos nem elég, nem biztosít elég információt a keresett pont pontos mélységének meghatározásához, valamint az triangulatePoints OpenCV függvényhez sem

elég egy felvételből kiszámított képkoordináták és projekciós mátrix. Ezek alapján indokolt kettő vagy több különböző kamerakép használata a számításokhoz.

A fenti bekezdésekben leírtaknak megfelelően mind az összetartozó pontok, mind a projekciós mátrixok rendelkezésre állnak az OpenCV triangulatePoints függvényéhez így minden készen áll egy előre meghatározott objektum térbeli elhelyezkedésének a kiszámítására.

A következő fejezetben bemutatom, hogyan valósítottam meg egy előre meghatározott objektum térbeli pozíciójának kiszámítását az aruco kód felismerő algoritmus, YOLOv4 neurális háló és az OpenCV triangulatePoints függvényének a segítségével. Az ezt követő fejezetekben pedig megvizsgálom a módszerem pontosságát.

4 Megvalósítás

A megvalósítás során egy olyan tesztkörnyezetet állítottam össze, ahol könnyen ki tudtam próbálni, szimulálni tudtam a módszerem működését. Az előző fejezetnek megfelelően az objektum, ebben az esetben a tehén, pontos térbeli pozíciójának meghatározásához legalább két kamera használata szükséges. Mindegyik kamerának egy olyan kameraképet kell biztosítania, természetesen különböző pozíciókból, amelyek egyszerre tartalmazzák az objektumokat, teheneket és az aruco kódot.

Felmerül a kérdés, hogy ezt hogyan lehet megvalósítani. Az egyik lehetőség, hogy két drónt használunk és ezeket elhelyezzük egy tetszőleges pozícióba, például egymás mellé úgy, hogy a tehenekre és az aruco kódra is jól rálássanak. Nehézséget jelenthet ennél a megvalósításnál a két kamerától érkező kameraképek szinkronizálása. Olyan esetekben különösen nagy probléma ez, ha hálózati hibák lépnek fel és az egyik képkocka sokkal később érkezik meg, mint a másik. Két drón használatának a valós életbeli megfelelője, mikor két drón repül egyszerre és párhuzamosan készítik a felvételeket. Egy másik gyakorlati megvalósítás, mikor nem két drón repül egyszerre hanem csak egy és a másik kamera szerepét a földön mozgó kamerával felszerelt AGV [34] (Automated Guided Vehicle, automatikusan irányított jármű) látja el. Ennek a megoldásnak egy fontos előnye, hogy az AGV és a drón teljesen más szemszögből látják az objektumot és így pontosabb becslést tudnak adni az objektum fizikai kiterjedésére.

A másik lehetőség, ha egy drónt használunk és ez az egy drón készít egy felvételt először az egyik tetszőleges pozícióból, majd egy másik tetszőleges pozícióból. Ebben az esetben egy fontos kritérium, hogy a két felvétel között lehetőleg ne változzon a környezet, a vizsgált objektumok. Ha egy viszonylag fix, nem mozgó környezetünk van akkor ez egy egyszerűbb megoldás. Egy drón használatának a valós életbeli megfelelője lenne egy olyan szituáció, mikor egy drón, egy kevéssé mozgó környezetet megfigyelve repül és készít felvételeket. A drón a felvételek készítésének helyszínei között mindig tart egy megfelelő távolságot, így biztosítva, hogy az objektumról mindig különböző felvételek készüljenek. Általában a drónok repülés közben sokat mozognak, így ez könnyem megvalósítható.

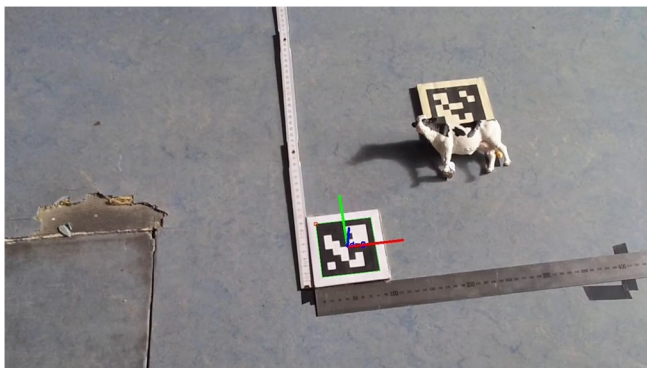
Mind a két kamerás, mind az egy kamerás megvalósítás esetén a kameráknak nemcsak az objektumot, hanem mindig legalább egy aruco kódot is látniuk kell. Ez egy nehézsége ennek a módszernek és adott esetben nagyon sok aruco kód elhelyezésével járhat. Ezt a problémát megkönnyíti, ha bevezetünk egy új entitást is, egy AGV-t például, ami együtt mozog a drónokkal a földön és a hátán visz egy aruco kódot. Ha a drónok mindig úgy repülnek, hogy rálássanak erre az AGV-re és a hátán lévő aruco kódra, akkor már minden követelmény adott az objektum lokalizációs módszeremhez és nem szükséges további aruco kódok elhelyezése a környezetben. A drónoknak egy ilyen jellegű repülését egy egyszerű követő algoritmussal könnyen meg lehet

valósítani. Ez az algoritmus a repülés során mindig úgy forgatja és állítja be a drón kamerájának a szögét (feltéve, ha mozgatható kamerával rendelkezik), hogy a drón kamerája minél jobban rálásson az aruco kódra. Az AGV és a drónok együttműködésének egy további előnye, hogy az objektum felismerő algoritmus az AGV szenzorait (mint LIDAR, GPS) is fel tudja használni az objektumok térbeli elhelyezéséhez.

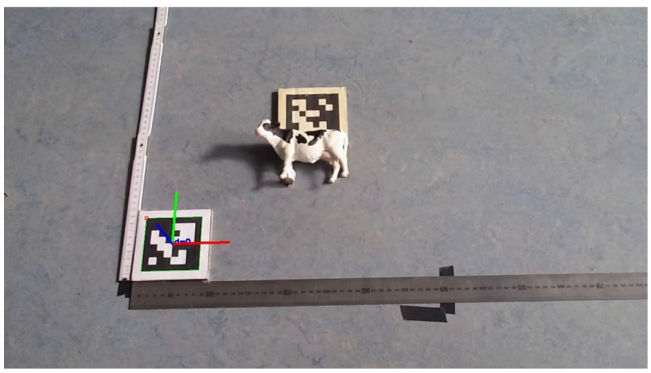
Az általam fejlesztett módszernek egy nagy előnye, hogy mind a két, mind az egy drónos esetben a készült kamerafelvételek egymáshoz viszonyított pozíciója tetszőleges lehet és változhat is a működés során. Nincsenek olyan jellegű megkötések mint a sztereó kameráknál, ahol a két kamera egymáshoz viszonyított pozíciójának fixnek kell lennie. A módszer működéséből adódik, hogy minden képkockánál ismerem mind a két kamera pozícióját egy fix ponthoz az aruco kódhoz képes (és így a két kamera egymáshoz viszonyított pozícióját is könnyen ki tudom számolni). Ebből adódóan mindegy, hogy a két kamera egymáshoz képest hogyan helyezkedik el.

A fentebb leírtak alapján egy kamerát és egy fix, nem változó környezetet használtam a megvalósítás során. A kamerát a 2.4.1 fejezetben bemutatott Mantis drón a környezetet pedig az aruco kódok és a 2.4.2 fejezetben bemutatott játék tehenek alkották. Összeállítottam egy teszt szituációt a Mantis drónnal, egy aruco kóddal és egy darab játék tehénnel, ahol ki tudtam próbálni a módszerem és vissza tudtam ellenőrizni a módszerem hozzávetőleges pontosságát, mennyire reális eredményt adott vissza.

Elhelyeztem a földön két darab aruco kódot (a kettő kódból csak egyet használt a módszerem a számításokhoz) és egy tehenet a drónt pedig beállítottam úgy az asztalon, hogy jól rálásson mind a tehénre mind az aruco kódokra. Készítettem egy félvételt majd arrébb tettem a drónt 30cm-el és készítettem még egy kamerafelvételt. Az alábbi 6. és 7. ábrán láthatóak a drón által készített és az aruco kód felismerő algoritmus által már feldolgozott képek. Az alsó aruco kódra az aruco kód felismerő algoritmus rajzolta rá a színes koordináta tengelyeket és az aruco kódot körbefogó zöld négyzetet.



6. ábra: Az első pozícióban készült és az aruco felismerő algoritmussal feldolgozott kép



7. ábra: A második pozícióban készült és az aruco felismerő algoritmussal feldolgozott kép

Az aruco kód felismerő algoritmus a tengelyek és a körbefogó négyzet rajzolása előtt az alábbi eltolási vektort és forgatási mátrixot számolta ki a két különböző pozícióban lévő kameraképre. Ezek az értékek láthatóak az alábbi 2-es és 3-es táblázatokban. Ezek az értékek a kamera koordináta rendszerében vannak.

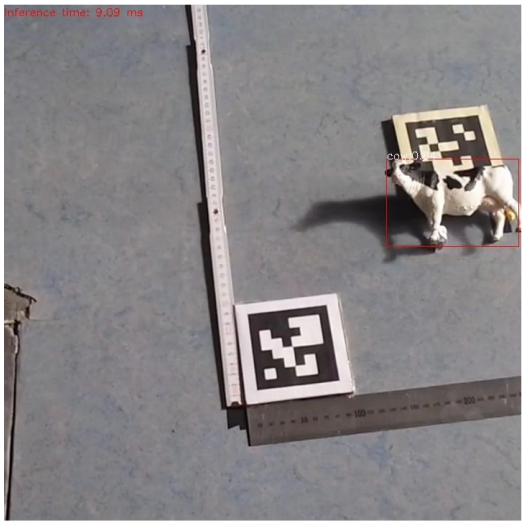
2. Táblázat: Az aruco kód felismerő algoritmusból származó eltolás vektor a két pozícióra

	Kamerakép 1	**Kamerakép 2**
x koordináta [méter]	0.02844	-0.21142
y koordináta [méter]	0.08379	0.0794029
z koordináta [méter]	0.842828	0.843313

3. Táblázat: Az aruco kód felismerő algoritmusból származó forgatási kvaternió

	Kamerakép 1	**Kamerakép 2**
1	0.97701	0.97887
i	-0.06096	-0.012208
j	0.01315	-0.006551
k	0.20386	0.203986

A 3-es táblázatban kvaternió értékek helyezkednek el, így első lépésben ezeket egy egyszerű beépített függvény segítségével átalakítottam forgatási mátrixszá mind a két kameraképnél. Ezt követően kiszámítottam a projekciós mátrixot a két kamerapozícióra a 2.1-es fejezetben bemutatottak szerint. Mielőtt használni tudnám az OpenCV triangulatePoints függvényét, szükséges a tehén kameraképen elhelyezkedő pozíciójának meghatározása a YOLO objektumfelismerő algoritmusnak a segítségével. Az alábbi 8. ábrán látható az algoritmus által bekeretezett tehén az egyik kameraállásra. A 8. ábrán a YOLO által feldolgozott képnek csak a fele látható. Az YOLO algoritmust futtató keretrendszer a feldolgozott képeket egy weboldalon jeleníti meg és hálózati késleltetés csökkentése érdekében a képnek csak a felét továbbítja erre a weboldalra. Később a mérést bemutató 5.2.1 fejezet 10. és 11. ábráján láthatóak példák a teljes képre.



8. ábra: Az objektum felismerő által felismert és bekeretezett játéktehén (ez egy levágott kép)

Az objektumfelismerő algoritmus az alábbi 4. táblázatban látható kamerakép koordinátákat határozta meg a felismert tehénnek az első és a második fotó esetén.

4. Táblázat: Tehén kameraképen elhelyezkedő pozíciója a két fotón.

	Kamerakép 1	**Kamerakép 2**
u [pixel]	910	589
v [pixel]	280	290

A 4 táblázat pixel koordinátáit és a korábban meghatározott projekciós mátrixot felhasználva lefuttattam az OpenCV triangulatePoints függvényét. Ez a függvény homogén koordinátaként adta vissza a tehén koordinátáját ezt visszaalakítva descartes koordinátává. Az alábbi 5 táblázatban látható eredményt kaptam.

5. Táblázat: A tehén térbeli elhelyezkedése

	Tehén koordinátája
x [méter]	0.1786
y [méter]	0.1168
z [méter]	0.0361

A 5 táblázat eredményeit az algoritmus az aruco kód közepétől mérte. Ezt figyelembe véve és az eredményeket összevetve az 6. és 7. ábrával láthatjuk, hogy a tehén pozícióját a módszer igencsak pontosan határozta meg. Ezek alapján látható, hogy a módszer működik és nagyon pontos. A következő fejezetben több mérésen keresztül is bemutatom, hogy pontosan mennyire pontos a módszer.

5 Mérés

A 4 fejezetben bemutatottak alapján az általam fejlesztett módszer képes egy aruco kód és két kamerafelvétel segítségével meghatározni egy tetszőleges objektum, játéktehén térbeli elhelyezkedését. A következő alfejezetekben azt fogom bemutatni több mérésen keresztül, hogy mennyire pontosan tud működni, mennyire precízen képes meghatározni az egyes koordinátákat.

5.1 Mérések leírása

A módszerem pontosságát három különböző mérés segítségével vizsgáltam meg. Ezek láthatóak az alábbi felsorolásban.

Mérés 1. Mennyire pontosan képes meghatározni a módszer a földön elhelyezett négy játéktehenet. A két kamerakép között a drónt csak egy irányba kicsit oldalra mozgattam.

Mérés 2. Mennyire pontosan képes a módszer meghatározni egy megemelt tehén pozícióját. Ebben az esetben is a két kamerakép között a drónt csak egy irányba kicsit oldalra mozgattam.

Mérés 3. Mennyire pontosan képes a módszer meghatározni a földön elhelyezett tehenet ha a két kameraképet a drón két tetszőleges pozícióból készítette.

A mérések elvégzése során először elkészítettem a kamerafelvételeket, majd megmértem kézzel, vonalzóval, hogy milyen messze helyezkedik el a tehén vagy helyezkednek el a tehenek az aruco kód középpontjától. Ezt követően kiszámoltattam az általam fejlesztett módszer segítségével is a tehenén térbeli koordinátáját és ezt összehasonlítottam a korábban kézzel lemért eredményekkel. Az összehasonlítás során csak kivontam a számolt és a mért értékeket, tulajdonképpen koordinátákat, egymásból és vettem az abszolút értéküket. Az így kiszámolt értékek, lettek az adott mérés során a módszer pontossága.

A mérés során a pontosságot több fontos tényező is befolyásolja. Egyrészt kézzel mértem meg a pozíciókat így csak milliméter pontosan tudtam meghatározni a pozíciókat. Másrészt egy játéktehénnek nem lehet olyan egyértelműen meghatározni a középpontját mint egy labdának vagy egy kockának így a tehén középpontját én határoztam meg szemmértékkel. Valamint az objektumfelismerő algoritmus csak két dimenzióban látja a tehenet és az általa meghatározott középpont nem feltétlenül fog egybe esni az általam meghatározottal. Ezeknek a figyelembevételével a mérések során meghatározott pontosság értékek nagyon

jónak tekinthetőek. A következő alfejezetben a korábban felsorolt méréseket fogom sorban bemutatni és kiértékelni.

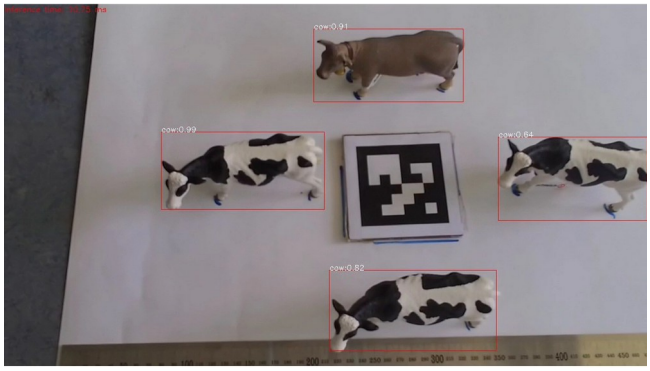
5.2 Mérések bemutatása és kiértékelése

5.2.1 Mérés 1

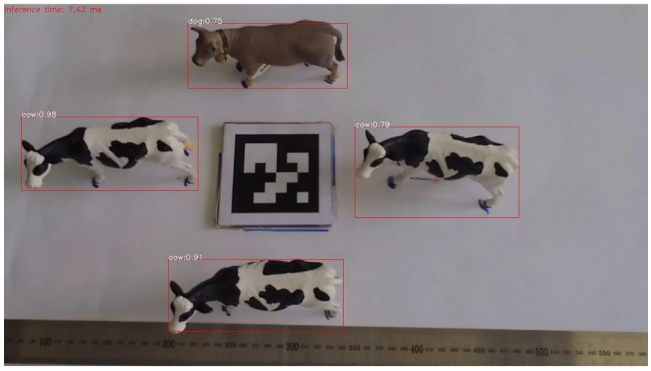
Az első mérés során megvizsgáltam mennyire pontosan képes meghatározni a módszerem a földön különböző pozíciókban elhelyezett játéktehéneket. A két kamerafelvétel között a drónt csak nagyjából 10 centiméterrel mozgattam el oldalra. Az alábbi 9. ábrán látható a mérés összeállítása a drónnal az aruco kóddal és a négy tehénnel. A drón ebben az összeállításba az egyes jelölésű hungarocell emelvényen készítette az első kamerafelvételt. Ezt követően áthelyeztem a jobb oldali kettes jelölésű hungarocellre és ott készítette a második felvételt.

9. ábra: Első mérés összeállítása

Az elkészült felvételeket az aruco kód felismerő algoritmus feldolgozta és kiszámolta a drón és az aruco kód egymáshoz képesti térbeli pozícióját, majd ennek az információnak a segítségével kiszámítottam a két projekciós mátrixot. Ezt követően a YOLO objektumfelismerő algoritmus segítségével meghatároztam a tehenek kamerafelvételen elhelyezkedő pozícióját. Az alábbi 10. és 11. ábrán láthatóak az objektumfelismerő algoritmus által felismert tehenek a két kameraképen. Ezek már a teljes képek amiket az algoritmus kiad magából és nem a korábban említett levágottak (A 8 ábrán csak a levágott kép volt látható)



10. ábra: Az objektumfelismerő által feldolgozott első kép



11. ábra: Az objektumfelismerő által feldolgozott második kép

Az objektumfelismerő által kiadott tehén koordináták nem tartalmazták, hogy a két képen melyek az összetartozóak ezeket nekem kézzel kellett összepárosítanom. Ez a probléma csak azért lépett fel mert ugyanaz a fajtájú objektum többször is megjelent a kamerafelvételen. Végül az objektumfelismerő által kiadott képi koordináták, a projekciós mátrix és az OpenCV függvények segítségével kiszámítottam a tehenek térbeli koordinátáját.

A pontosság kiszámításához szükséges még megmérni, hogy a tehenek milyen magasak és milyen messze helyezkednek el síkban az aruco kód közepétől. Miután ezt elvégeztem, már ki tudtam számolni a pontosságot vagyis a módszer által kiszámított és az általam lemért értékek különbségének az abszolútértékét. Ezek a pontosság értékek, eltérések láthatóak az alábbi 6 táblázatban. A táblázat oszlopai mutatják az egyes koordinátákat míg a sorok felelnek meg az egyes teheneknek. Látható, hogy a kettes tehén y koordinátája esetén 0.0078 méter vagyis 7.8 milliméter eltérés volt a számított és a mért érték között. A táblázat alapján a legtöbb esetben csak milliméter nagyságrendű hibák léptek fel és csak ritkán fordult elő egy centiméter nagyságrendű eltérés, valamint az értékek alapján a tehenek pozíciója nem befolyásolja a pontosságot.

6. Táblázat: Az első mérés során a pontosság (eltérés) az egyes tehenek esetén

	x eltérés [méter]	**y eltérés [méter]**	**z eltérés [méter]**
1	0.0064	0.0122	0.0078
2	0.0056	0.0078	0.0129
3	0.0080	0.0129	0.0009
4	0.0007	0.0166	0.0117

5.2.2 Mérés 2

A második mérés során a játéktehenet egy emelvényre tettem és az első mérésnél használt összeállítás segítségével mértem meg a módszer pontosságát. Lényegében az első mérés összeállításából elvettem a négy tehenet és kicseréltem egy tehénre amit egy 13 centiméter magas dobozra helyeztem. Elkészítettem a két felvételt, az első méréshez hasonlóan kiszámítottam a két projekciós mátrixot majd meghatároztam a tehenek kamerafelvétel koordinátáját. Az alábbi 12. és 13. ábrán látható az objektumfelismerő algoritmus által feldolgozott képek.

12. ábra: Az objektumfelismerő által feldolgozott első kép

13. ábra: Az objektumfelismerő által feldolgozott második kép

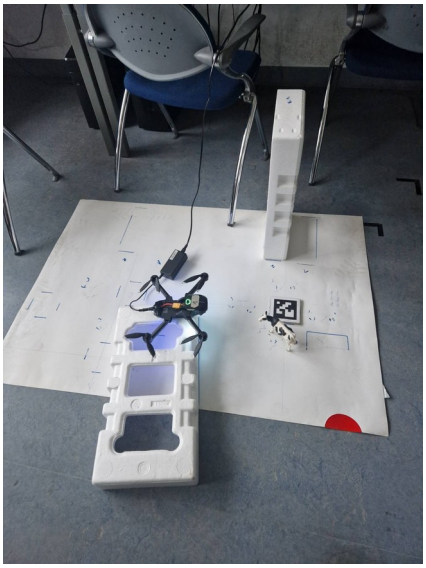
A projekciós mátrix és a tehenek kamerafelvétel koordinátájának a segítségével kiszámítottam a tehén valós térbeli koordinátáját, majd a módszerem pontosságát. Az alábbi 7. táblázatban láthatóak az eltérések más szóval a pontosság értékek. Ennél a mérésnél kicsit nagyobbak voltak az eltérések, mint az első mérés esetén, de még így is csak egy-két centiméter nagyságrendű a hiba. A kissé nagyobb értékeket mérési pontatlanság okozhatta vagy pedig, hogy a 13. ábrán is látható második felvételen az objektumfelismerő algoritmus nem látta az egész játék tehenet. Továbbfejlesztésképpen a jövőben érdemes lesz különválasztani a pontosság növelése érdekében azt az esetet, amikor az objektumfelismerő algoritmus úgy ismer fel egy objektumot, hogy csak egy részét látja.

7. Táblázat: A módszer pontossága a második mérés esetén

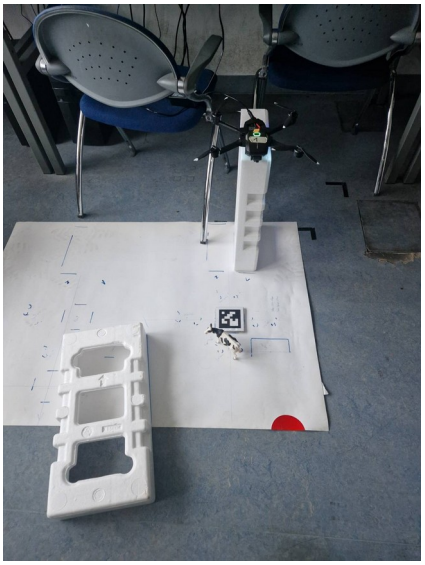
x eltérés [méter]	y eltérés [méter]	z eltérés [méter]
0,0206	0,0011	0,0172

5.2.3 Mérés 3

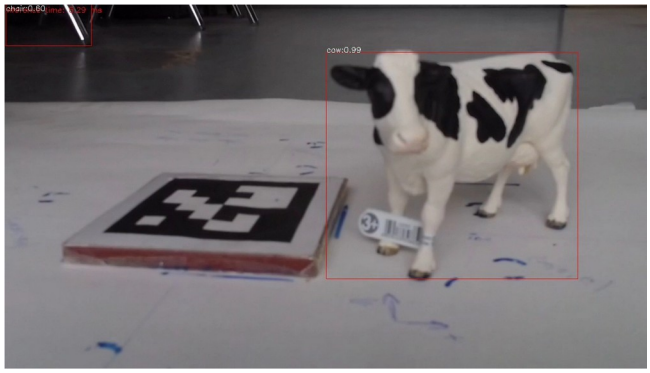
Végül a harmadik, utolsó mérés során a megvizsgáltam mennyire pontos a módszer abban az esetben, ha a játéktehén ugyanúgy a földön helyezkedik el, viszont a két kamerafelvétel tetszőleges pozícióban készült (természetesen úgy, hogy mind az aruco kód mind az objektum jól látszódjon a képen). Az alábbi 14. és 15. ábrán a mérés elrendezése az első és a második felvétel készítésének pillanatában a 16. és 17. ábrán pedig az objektumfelismerő által feldolgozott képek láthatóak. Megfigyelhető, hogy a két felvétel teljesen eltérő szögből készült. A drónt nem csak oldalra mozgattam el, a két felvétel között, mint a korábbi két mérés esetén, hanem egy teljesen tetszőleges pozícióba mozgattam el és még el is forgattam.



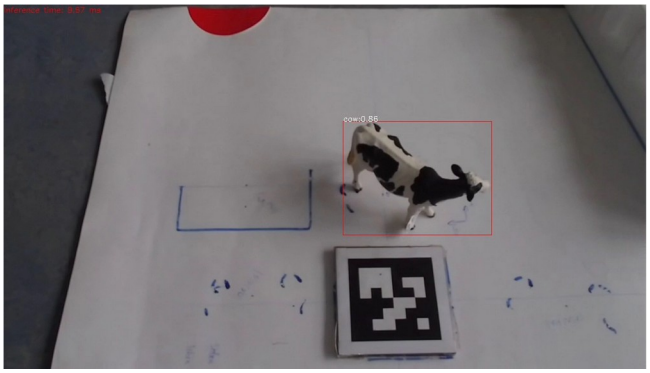
14. ábra: A harmadik mérés elrendezése az első felvétel készítésénél



15. ábra: A harmadik mérés elrendezése a második felvétel készítésénél



16. ábra: Az objektumfelismerő által feldolgozott első kép a harmadik mérés esetén



17. ábra: Az objektumfelismerő által feldolgozott második kép a harmadik mérés esetén

Ezekben a pozíciókban is jól működött mind az objektumfelismerő mind az aruco kód felismerő algoritmus és meghatározták a szükséges értékeket. A drón kamerafelvételen elhelyezkedő pozícióját, valamint az eltolási vektort és a forgatási kvaterniót. Ezek segítségével pedig meghatároztam a tehén térbeli elhelyezkedését és az eltéréseket, más szóval a módszer pontosságát. Az eltérések az alábbi 8. táblázatban láthatóak. A módszer a harmadik mérés esetén is nagyon pontos volt és csak két esetben volt centiméter nagyságrendű eltérés. Ennek az eltérésnek az oka lehet ugyancsak mérési hiba vagy pedig, hogy a két kamera teljesen más szemszögből látta az objektumot és máshogy határozták meg a középpontját.

8. Táblázat: A módszer pontossága a harmadik mérés esetén

x eltérés [méter]	**y eltérés [méter]**	**z eltérés [méter]**
0.0039	0.0159	0.0163

A harmadik mérés során bemutatott mérési összeállítás nagyban hasonlít a korábban a 2.5.1 alfejezetben említett SFM-hez. Az SFM-nél és az én módszeremnél is tetszőleges pozícióból készült képek segítségével szeretnénk térbeli pontokat rekonstruálni. Amíg az SFM-nél az összepárosított képekből kell a kamera pozíciót is kitalálni ami nem feltétlenül lesz pontos addig az én megoldásom esetében az aruco kód biztosítja ezt az információt.

A fenti mérések alapján látható, hogy sem az objektumok száma, térbeli pozíciója (síkbeli és magassága) sem a kamerák helyzete nem befolyásolja szignifikánsan a módszer pontosságát. A következő alfejezetben röviden összehasonlítom a módszerem más háromdimenziós látásra képes megoldásokkal, mint a sztereó kamerával, ToF és a Structured light szenzorokkal.

5.3 Összehasonlítás

A munkám során fejlesztett módszer nagyban hasonlít a sztereó kamerákhoz. A sztereó kameráknál fontos, hogy a két kamera fix legyen egymáshoz képest addig az én módszeremnél nincsenek ilyen megkötések, a két kamera pozíciója akár folyamatosan is változhat. Míg a sztereó kameránál egy fontos és komplex lépés, hogy meg kell találni a két képen az egyező részeket, addig az én módszeremnél ezt a lépest az objektumfelismerő algoritmus helyettesíti. További előnye a módszeremnek a sztereó kamerával szemben, hogy a tetszőleges kamera állás miatt a keresett objektum fizikai kiterjedésével kapcsolatban is nyújt információt. A sztereó kamera az objektumok térbeli kiterjedéséről semmilyen információt nem tud adni. Az általam fejlesztett módszernek egy hátránya, hogy el kell helyezni legalább egy aruco kódot a kamerák környezetében és a módszer csak akkor működik ha legalább egy aruco kódot lát. Ez, egy ipari környezetben könnyen megtehető, egyszerűen csak fel kell ragasztani a megfelelő helyekre az aruco kódokat. Mezőgazdasági környezetben sem nehéz ekkor például úgy kell repülni a drónoknak, hogy mindig lássanak egy olyan autót (ez lehet akár teljesen automata, AGV) aminek a tetején egy aruco kód található.

A korábbi 1. összehasonlító táblázatot kiegészítve az új módszerrel az alábbi 9 táblázatot kaptam.

9. Táblázat: Különböző térbeli látásra alkalmas szenzorok összehasonlítása kiegészítve az általam fejlesztett módszerrel

Név	**Típus**	**Mélység hatótávolsága**	**3D felbontás**	**Képkockasebesség**	**Látómező (H: horizontális V: vertikális D: diagonális)**
Microsoft Azure Kinect	ToF	0.25 - 5.46 m	Narrow mód: 6540x576 Wide mód: 1024x1024	30 fps	Narrow mód - 75° H, 65° V; Wide mód - 120° H, 129° V
Orbbec Astra Mini	Structured Light	0.6 - 5.0 m	640 x 480	30 fps	73° D, 60° H, 49.5° V
MYNT EYE	Sztereó kamera	0.5 to 18 m	752x480	60 fps	146° D, 122° H, 76° V
Sztereó aruco kóddal	Sztereó aruco kóddal	aruco kód és objektum méretétől függ	1280x720	30fps	117° H

A módszeremet a táblázatban sztereó aruco kóddal néven írtam be. Mélységi távolságot sajnos nem tudtam meghatározni mivel ez az objektum és az aruco kód méretétől függ. Lesz egy alsó és egy felső korlát, de ezt mindig az adott aruco kód és objektum konkrét mérete fogja meghatározni. A többi cellát a Mantis drón által készített videofolyam paraméterivel töltöttem ki, mivel a módszernek ez a videofolyam a bemenete és ennek a paramétereitől függ az objektum meghatározásának a pontossága.

A 9 táblázat alapján látható, hogy a sztereó aruco kóddal módszer felbontás tekintetében majdnem a legjobb gyorsaságban átlagos míg látómező tekintetében is majdnem a legjobb. Továbbá egy fontos előnye még a sztereó aruco kóddal módszernek, hogy nagyon egyszerű és olcsón meg lehet valósítani.

6 Konklúzió

A munkám során egy olyan módszert fejlesztettem ki, mely nagy pontossággal képes meghatározni egy tetszőleges objektum térbeli pozícióját kamera és egy aruco kód segítségével. Több mérésen keresztül bemutattam, hogy a környezeti tényezők, mint az objektumok száma, helyzete a kamerák pozíciójának változtatása esetén is képes pontosan és robusztusan működni.

Számos továbbfejlesztési lehetősége létezik a témának. Meg lehet vizsgálni, hogy az objektumfelismerő által kiadott befoglaló négyzet, kamerakép koordinátái hova képződnek le a térben és meg lehet-e határozni a térbeli objektum köré rajzolható bennfoglaló téglatestet. Másik továbbfejlesztési irány az AGV és a drón együttműködéséhez kapcsolódik. Fel lehet-e használni az AGV Lidar szenzorát a pontosabb objektum lokalizációhoz.

7 Irodalomjegyzék

[1] A. Kaehler és G. Bradski, *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*, 1st edition. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly Media, 2016.

[2] „Geometry of Image Formation | LearnOpenCV #", 2020. február 20. https://learnopencv.com/geometry-of-image-formation/ (elérés 2022. október 15.).

[3] F. J. Romero-Ramirez, R. Muñoz-Salinas, és R. Medina-Carnicer, „Speeded up detection of squared fiducial markers", *Image Vis. Comput.*, köt. 76, o. 38–47, aug. 2018, doi: 10.1016/j.imavis.2018.05.004.

[4] D. Jurado-Rodriguez, R. Munoz-Salinas, S. Garrido-Jurado, és R. Medina-Carnicer, „Design, Detection, and Tracking of Customized Fiducial Markers", *IEEE Access*, köt. 9, o. 140066–140078, 2021, doi: 10.1109/ACCESS.2021.3118049.

[5] R. W. Hamming, „Error detecting and error correcting codes", *Bell Syst. Tech. J.*, köt. 29, sz. 2, o. 147–160, ápr. 1950, doi: 10.1002/j.1538-7305.1950.tb00463.x.

[6] R. Girshick, J. Donahue, T. Darrell, és J. Malik, „Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, jún. 2014, o. 580–587. doi: 10.1109/CVPR.2014.81.

[7] A. Bochkovskiy, C.-Y. Wang, és H. Liao, *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020.

[8] W. Liu *és mtsai.*, „SSD: Single Shot MultiBox Detector", in *ECCV*, 2016. doi: 10.1007/978-3-319-46448-0_2.

[9] J. Redmon, S. Divvala, R. Girshick, és A. Farhadi, „You Only Look Once: Unified, Real-Time Object Detection", in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jún. 2016, o. 779–788. doi: 10.1109/CVPR.2016.91.

[10] J. Redmon és A. Farhadi, „YOLO9000: Better, Faster, Stronger", in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, júl. 2017, o. 6517–6525. doi: 10.1109/CVPR.2017.690.

[11] J. Redmon és A. Farhadi, „YOLOv3: An Incremental Improvement", *ArXiv*, 2018.

[12] A. Bochkovskiy, C.-Y. Wang, és H. Liao, „YOLOv4: Optimal Speed and Accuracy of Object Detection", *ArXiv*, 2020.

[13] C.-Y. Wang, A. Bochkovskiy, és H.-Y. M. Liao, „YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors". arXiv, 2022. július 6. doi: 10.48550/arXiv.2207.02696.

[14] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, és M. Khalgui, „Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey", *IEEE Access*, köt. 7, o. 87658–87680, 2019, doi: 10.1109/ACCESS.2019.2924410.

[15] A. Rao és R. Lanphier, „Real Time Streaming Protocol(RTSP)", Internet Engineering Task Force, Internet Draft draft-rao-rtsp-00, 1996. Elérés: 2022. október 24. [Online]. Elérhető: https://datatracker.ietf.org/doc/draft-rao-rtsp-00

[16] D. G. Lowe, „Object recognition from local scale-invariant features", in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, szept. 1999, köt. 2, o. 1150–1157 köt.2. doi: 10.1109/ICCV.1999.790410.

[17] H. Bay, T. Tuytelaars, és L. Van Gool, „SURF: Speeded Up Robust Features", in *Computer Vision – ECCV 2006*, Berlin, Heidelberg, 2006, o. 404–417. doi: 10.1007/11744023_32.

[18] B. D. Lucas és T. Kanade, „An Iterative Image Registration Technique with an Application to Stereo Vision", o. 11.

[19] J. L. Schonberger és J.-M. Frahm, „Structure-from-Motion Revisited", in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, jún. 2016, o. 4104–4113. doi: 10.1109/CVPR.2016.445.

[20] V. M. Govindu, „Combining two-view constraints for motion estimation”, in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001, köt. 2, o. II–II. doi: 10.1109/CVPR.2001.990963.
[21] R. Gherardi, M. Farenzena, és A. Fusiello, „Improving the efficiency of hierarchical structure-and-motion”, in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, jún. 2010, o. 1594–1600. doi: 10.1109/CVPR.2010.5539782.
[22] M. A. Fischler és R. C. Bolles, „Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”, *Commun. ACM*, köt. 24, sz. 6, o. 381–395, Június 1981, doi: 10.1145/358669.358692.
[23] © Brian Wandell és S. U. | T. of U. | Login, „Foundations of Vision » Chapter 6: The Cortical Representation”. https://foundationsofvision.stanford.edu/chapter-6-the-cortical-representation/ (elérés 2022. november 1.).
[24] mars.nasa.gov, „Rover Cameras”. https://mars.nasa.gov/mars2020/spacecraft/rover/cameras/ (elérés 2022. október 22.).
[25] V. Roback *és mtsai.*, „Helicopter Flight Test of 3-D Imaging Flash LIDAR Technology for Safe, Autonomous, and Precise Planetary Landing”, előadás SPIE Defense, Security, and Sensing 2013, ápr. 2013. Elérés: 2022. október 23. [Online]. Elérhető: https://ntrs.nasa.gov/citations/20130013472
[26] Wojtaszek V., „Fotointerpretáció és távérzékelés 3.”, o. 27.
[27] „Apple unveils new iPad Pro with LiDAR Scanner and trackpad support in iPadOS”, *Apple Newsroom*. https://www.apple.com/newsroom/2020/03/apple-unveils-new-ipad-pro-with-lidar-scanner-and-trackpad-support-in-ipados/ (elérés 2022. október 23.).
[28] P. Fechteler, P. Eisert, és J. Rurainsky, „Fast and High Resolution 3D Face Scanning”, in *2007 IEEE International Conference on Image Processing*, San Antonio, TX, USA, szept. 2007, o. III-81-III–84. doi: 10.1109/ICIP.2007.4379251.
[29] P. Fechteler és P. Eisert, „Adaptive color classification for structured light systems”, in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, Anchorage, AK, USA, jún. 2008, o. 1–7. doi: 10.1109/CVPRW.2008.4563048.
[30] „About Face ID advanced technology”, *Apple Support*. https://support.apple.com/en-us/HT208108 (elérés 2022. október 23.).
[31] „3D Camera Survey”, *ROS-Industrial*. https://rosindustrial.org/3d-camera-survey (elérés 2022. október 31.).
[32] V. Gutta, E. Lemaire, N. Baddour, és P. Fallavollita, *A Comparison of Depth Sensors for 3D Object Surface Reconstruction*. 2019.
[33] „OpenCV: Triangulation”. https://docs.opencv.org/4.x/d0/dbd/group__triangulation.html#ga211c855276b3084f3bbd8b2d9161dc74 (elérés 2022. október 25.).
[34] J. Sankari és R. Imtiaz, „Automated guided vehicle(AGV) for industrial sector”, in *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, 2016, o. 1–5. doi: 10.1109/ISCO.2016.7726962.