Szegedi Tudományegyetem

Természettudományi és Informatikai Kar

Informatikai Intézet

Az AMBITUS rendszer alkalmazása patkányok magatartásának videóalapú elemzésére

Szakdolgozat

Készítette:
Korfanti Dániel Mihály
Programtervező informatika
szakos hallgató

Témavezető:
Dr. Kőrösi Gábor
Egyetemi docens

Szeged
2025

Tartalmi összefoglaló

- **Téma megnevezése:**
  Videóelemző szoftver fejlesztése az AMBITUS kísérleti rendszerhez, amely a hagyományos szenzoralapú méréseket vizuális adatgyűjtéssel egészíti ki a patkányok viselkedésvizsgálata során.

- **A megadott feladat megfogalmazása:**
  Egy Python alapú alkalmazás készítése, amely képes a kísérleti alanyok mozgásának, pozíciójának és jutalomfelvételének automatikus, valós idejű detektálására. A feladat része az interaktív régiókijelölés biztosítása, az adatok strukturált exportálása, valamint a rendszer pontosságának validálása manuális elemzéssel szemben.

- **Megoldási mód:**
  A szoftver adaptív háttérmodellezést és morfológiai szűrést alkalmaz az alany leválasztására. A pozíciókövetés kontúr alapú tömegközéppont számítással történik, amelyet hiszterézis alapú eseménydetektálás és a jelvesztést kezelő becslési algoritmus egészít ki.

- **Alkalmazott eszközök, módszerek:**
  A fejlesztés Python nyelven történt, elsősorban az OpenCV könyvtár képfeldolgozó algoritmusaira és a NumPy numerikus műveleteire építve, CSV alapú adattárolással.

- **Elért eredmények:**
  A rendszer megbízhatóan, közel maximális pontossággal követi az alanyokat és rögzíti a viselkedési eseményeket. A kinyert adatokból részletes vizualizációk készíthetők.

- **Kulcsszavak:**
  AMBITUS, videóanalízis, objektumkövetés, OpenCV, Python, képfeldolgozás, etológia, automatikus detekció.

Tartalomjegyzék

Feladatkiírás

A feladat az AMBITUS rendszer támogatása egy videóelemző szoftver fejlesztésével, aminek felhasználásával a szenzoros adatgyűjtést vizuális elemzéssel lehet kiegészíteni.

Ez egy többmodulos alkalmazás, azaz mind az interaktív régiókijelölést, mind a mozgáskövetést és jutalomdetekciót magába kell, hogy foglaljon. A mért adatok exportálása és a beállítások kezelése is a feladat része.

Ezek mellett egy kísérlet végzése arra vonatkozóan, hogy mennyire pontosak az automata rendszer által mért adatok a manuális elemzéssel kapott eredményekhez képest ugyanazon a felvételen.

Bevezetés

Az automatizált videóelemzés mára a modern viselkedéskutatás egyik alapvető eszközévé vált, mivel nagy mennyiségű vizuális adat gyors, objektív és megismételhető feldolgozását teszi lehetővé. Ez különösen fontos olyan kutatási területeken, ahol a viselkedési minták és az egyedek közötti interakciók pontos rögzítése és értelmezése kulcsfontosságú [8].

Jelen dolgozat középpontjában az AMBITUS kísérleti rendszerhez fejlesztett videóelemző szoftver áll. A program célja, hogy támogassa a kutatókat a rögzített felvételek hatékony feldolgozásában, a fontos események automatikus észlelésében, valamint az így gyűjtött adatok strukturált kiértékelésében. A fejlesztett megoldás a hagyományos, szenzoralapú méréseket kiegészítve egy olyan vizuális adatréteget biztosít, amely lehetővé teszi a viselkedési folyamatok részletesebb, finomabb szintű elemzését.

1. fejezet

Irodalmi áttekintés

1.1. Az állatmagatartás vizsgálat módszertani háttere

Az állatok viselkedésének kutatása az agyi folyamatok és a kognitív funkciók megértésének egyik alapvető eszköze. A rágcsálókon végzett kísérletek lehetőséget adnak olyan kérdések vizsgálatára, amelyek emberi alanyok esetén etikai vagy technikai okokból nem lennének kivitelezhetőek [12]. A tanulás és az emlékezés alapvető mechanizmusai az emlősök körében hasonlóak, ezért ezek az állatmodellek megbízható alapot nyújtanak az emberi folyamatok megértéséhez [13].

A rágcsálók kognitív képességeinek, térbeli tájékozódásának és információszerzési stratégiáinak vizsgálatára számos standardizált eljárás áll rendelkezésre. A klasszikus tesztek közé tartoznak többek között:

- **Lyukas tábla tesztek:** Elsősorban a kíváncsiságot, a felfedező viselkedést és a szorongásos reakciókat vizsgálják. A lyukak közötti mozgásminták és a felfedezés időzítése alapján részletes képet adnak az állat vizsgálódási stratégiáiról [14, 23].
- **Útvesztők:** Az Y-útvesztő, Barnes labirintus vagy a Morris vízi labirintus a térbeli memória és a navigáció legfontosabb mérőeszközei [16–18, 25]. Ezek a tesztek azt vizsgálják, hogy a patkány mennyire képes mentális térképet alkotni környezetéről, és hogyan hívja elő a tárolt információt például egy menekülőplatform vagy jutalom elérése érdekében [24].
- **Új tárgy felismerési tesztek:** Ezek az eljárások a felismerésen alapuló memória és a tudatos visszaemlékezés vizsgálatára szolgálnak, kihasználva a rágcsálók ösztö-

nös érdeklődését az újdonságok iránt. A jól ismert és új tárgyak közötti preferencia alapján következtethetünk a memória épségére és a tanulási folyamatokra [15].

A hagyományos, manuális adatgyűjtés korlátai mint az alacsony mintavételezési ráta, az emberi megfigyelő szubjektivitása és a hosszadalmas kiértékelési folyamat egyre kevésbé felelnek meg a modern kutatási igényeknek. A cél ma már olyan automatizált adatfeldolgozási láncok kialakítása, amelyek a viselkedési mintázatokat objektív, számszerűsíthető idősoros adatokká alakítják. Ehhez fejlett képfeldolgozó algoritmusokra van szükség, amelyek a finommotoros mozgások és a komplex útvonal stratégiák detektálását is lehetővé teszik folyamatos emberi jelenlét nélkül [9].

1.2. Az AMBITUS rendszer és a videóalapú elemzés előnyei

Az AMBITUS egy egyedi fejlesztésű, derékszögű körfolyosó elrendezésű kísérleti rendszer, amely 16 szimmetrikus fülkével rendelkezik, ezekben helyezik el a jutalomfalatokat a patkányoknak [11]. A berendezés konstrukciója tudatosan ötvözi a klasszikus modellek előnyeit. A lyukas tábla tesztek által nyújtott felfedezési lehetőségeket kombinálja az útvesztők térbeli navigációs kihívásaival. Ez a hibrid elrendezés lehetővé teszi, hogy a viselkedési mintázatokat, a memóriafolyamatokat és a jutalomfelvételi stratégiákat egyidejűleg és részletesen lehessen vizsgálni [22].

A rendszer infravörös LED-ek és fotocellák segítségével követi a patkányok mozgását. Ez a szenzoralapú adatgyűjtés milliszekundumos időfelbontással képes regisztrálni az áthaladásokat a folyosószakaszokon és a fülkék bejáratánál. Ugyanakkor ezek a szenzorok bináris jellegű információt biztosítanak, így megmutatják, hogy az állat mikor haladt át egy adott ponton, de nem adnak részletes képet a pontos testtartásáról, a hezitációkról vagy az alkalmazott megközelítési útvonalakról [21].

A videóalapú elemzés bevezetése ezt a hiányt hidalja át. A folyamatos videorögzítés lehetővé teszi a mozgás pályájának, sebességének, irányváltásainak és a fülkék körüli viselkedés részletes rekonstrukcióját, valamint a speciális események például a tényleges jutalomfelszedések vizuális verifikálását és automatikus detektálását. Ez a kiegészítő vizuális adatréteg különösen fontos a kognitív zavarokhoz kapcsolódó finom viselkedésbeli

eltérések azonosításában, ahol a pusztán szenzoralapú mérés önmagában nem elegendő [19].

1.3. Képfeldolgozás Pythonban OpenCV-vel

A videóalapú követés és viselkedéselemzés technikai alapját a Python programozási nyelv, a NumPy könyvtár és az OpenCV keretrendszer együttes használata adja [5, 6]. Az OpenCV optimalizált, valós idejű feldolgozásra is alkalmas algoritmusokat biztosít többek között az adaptív háttérkivonás, a morfológiai transzformációk és a kontúrdetektálás megvalósításához [26]. A Python magas szintű interfésze lehetővé teszi ezeknek a funkcióknak a gyors és rugalmas integrálását, ami különösen előnyös olyan kutatási környezetben, ahol a rendelkezésre álló hardver korlátozott és nincs mód nagy erőforrás igényű mélytanulási modellek futtatására [27].

A videók feldolgozása több, egymásra épülő lépésből áll. Elsőként egy adaptív háttérmodell, a MOG2 algoritmus segítségével a rendszer elkülöníti az állatot a statikus háttértől [1]. Az algoritmus folyamatosan alkalmazkodik a környezeti változásokhoz, ami különösen hasznos infravörös megvilágítás mellett, ahol a háttér fényviszonyai módosulhatnak [7]. A háttérkivonás eredményeként kapott bináris maszkot ezt követően morfológiai műveletek tisztítják meg az elszórt zajpontoktól [4].

A zajszűrt maszkon az OpenCV hatékony `findContours()` függvénye segítségével detektáljuk az állatot reprezentáló kontúrt [3]. A kontúr alapján meghatározható az állat képen belüli kiterjedése, és a kontúrhoz tartozó pixelek felhasználásával kiszámítható a pozíció tömegközéppontja is [2]. Az így nyert, időben folyamatos pozíciósorozat már alkalmas arra, hogy a viselkedés elemzése során különféle mozgásparamétereket, útvonal stratégiákat és döntési mintázatokat számszerűsítsen.

1.4. Az alkalmazott algoritmusok matematikai háttere

Az implementáció során felhasznált függvények komplex matematikai modellekre és hatékony tömbműveletekre épülnek. A rendszer megbízhatóságának és korlátainak megértéséhez szükséges a háttérkivonás, a zajszűrés és az objektum pozicionálásának elméleti hátterét röviden áttekinteni.

1.4.1. Adaptív Gauss-keverék modellek

A videófeldolgozás egyik alapvető feladata a mozgó objektumok elkülönítése a statikus háttértől. A szoftver ehhez a `BackgroundSubtractorMOG2` algoritmust alkalmazza, amely a Zivkovic által ismertetett, pixelenként adaptívan tanuló Gauss-keverék háttérmodellen alapul [1]. A modell célja, hogy egy adott pixel időben változó intenzitásértékeit statisztikailag leírja, és ennek alapján eldöntse, hogy az adott pillanatban a háttérhez vagy az előtérhez tartozik.

A módszer lényege, hogy minden pixel intenzitáseloszlását nem egyetlen értékkel, hanem M darab, izotróp Gauss-eloszlás súlyozott összegével közelíti [1]. Jelölje $\mathcal{X}_T = \{x^{(t)}, \ldots, x^{(t-T)}\}$ a pixel előző T képkockára vonatkozó történetét, továbbá legyen d a vektor $\vec{x}$ dimenziója. Ekkor a pixel aktuális értékének becsült valószínűsége a teljes Gauss-sűrűség felírásával:

$$\hat{p}(\vec{x} \mid \mathcal{X}_T) = \sum_{m=1}^{M} \hat{\pi}_m \frac{1}{(2\pi)^{d/2}\hat{\sigma}_m^d} \exp\left(-\frac{1}{2\hat{\sigma}_m^2}\left\|\vec{x} - \hat{\vec{\mu}}_m\right\|^2\right) \tag{1.1}$$

ahol $\hat{\pi}_m$ az m-edik keverékkomponens becsült súlya, $\hat{\vec{\mu}}_m$ a várható értéke, $\hat{\sigma}_m^2$ pedig az ehhez tartozó variancia. Az $\hat{\sigma}_m^2 I$ alakú kovarianciamátrix izotróp Gauss-eloszlást feltételez, összhangban a Gauss-eloszlás standard definíciójával. Az algoritmus minden új képkocka érkezésekor rekurzívan frissíti a keverékkomponensek súlyait és paramétereit, így a háttérmodell folyamatosan alkalmazkodik a fényviszonyok lassú változásaihoz és a jelenet dinamikájához [7]. Háttérnek azokat a komponenseket tekinti, amelyek tartósan nagy súllyal és kis szórással rendelkeznek. Ha egy új pixelérték egyik ilyen háttérkomponens Gauss-görbéjéhez sem illeszkedik, akkor előtérhez, vagyis mozgó objektumhoz sorolódik [1].

1.1. ábra. Háttérszubsztrakció

1.4.2. Matematikai morfológia és zajszűrés

A háttérkivonás eredményeként kapott bináris maszkon gyakran jelennek meg elszórt, zajként értelmezhető pixelek. Ezek eltávolítására a szoftver a matematikai morfológia alapműveleteit használja. A Gonzalez és Woods által ismertetett halmazelméleti keretben legyen A a bináris kép, B pedig a választott strukturáló elem [4]. A morfológiai műveletek A és B halmazokon értelmezett halmazoperációk, amelyek az objektum alakját és topológiáját módosítják.

Az A halmaz B-vel történő *eróziója* a következőképpen definiálható:

$$A \ominus B = \{z \mid (B)_z \subseteq A\}, \tag{1.2}$$

ahol $(B)_z$ a B halmaz z vektorral történő eltolását jelöli. Intuitívan az erózió minden olyan pontot eltávolít az objektum pereméről, ahol a strukturáló elem már nem illeszthető teljes egészében az A halmazba, így az objektumok mérete lecsökken, a kis, keskeny struktúrák pedig eltűnnek [4].

A *dilatáció* definíciója a B strukturáló elem $\hat{B}$ tükörképén alapul:

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}, \tag{1.3}$$

ahol $\hat{B} = \{w \mid w = -b, b \in B\}$. A dilatáció ennek megfelelően bővíti az objektumot a strukturáló elem alakjának megfelelő pixeleket az A halmazhoz, ezáltal az objektumok kitágulnak, és a kisebb hézagok, szakadások részben feltöltődnek [4].

A zajszűrésre alkalmazott *morfológiai nyitás* az A halmaz B-vel végzett eróziójából, majd az így kapott halmaz B-vel végzett dilatációjából áll:

$$A \circ B = (A \ominus B) \oplus B. \tag{1.4}$$

A nyitás egyenletesebbé teszi az objektumok szélét, és eltünteti a vékony összekapcsolódásokat, valamint eltávolítja a vékony, elszigetelt nyúlványokat, miközben a nagyobb, releváns struktúrák geometriáját nagyrészt megőrzi [4]. Ezzel szemben a *zárás* művelet az erózió és a dilatáció dualitására épül, és az A halmaz B-vel végzett dilatációjaként, majd az így kapott halmaz B-vel végzett eróziójaként definiálható:

$$A \bullet B = (A \oplus B) \ominus B. \tag{1.5}$$

A zárás is egyenletesebbé teszi az objektum szélét, de nem bontja szét a formákat, hanem inkább összezárja a repedéseket és kitölti a kisebb üregeket [4]. Ez a tulajdonság különösen hasznos a patkány testének összefüggő bináris reprezentációjának kialakításához, amely a további viselkedéselemzés kiindulópontja.

1.4.3. Topologikus struktúraelemzés

A geometriai jellemzők, így különösen a momentumok kiszámításához a szoftvernek először azonosítania kell a bináris maszkon megjelenő összefüggő objektumokat, és ki kell jelölnie ezek határvonalait. E célból a Suzuki és Abe által kidolgozott határkövető algoritmus kerül alkalmazásra a `cv2.findContours` függvényen keresztül, amely a bináris kép topológiai viszonyait használja fel a kontúrok hierarchikus feltérképezésére [3].

Az algoritmus a külső és belső kontúrokat fa struktúrába rendezi, amelyben a gyökércsomópontok a külső objektumhatárokat, a gyermekcsomópontok pedig az ezekhez tartozó belső lyukakat reprezentálják [3]. Ez a hierarchikus leírás egyértelműen áttekinthetővé teszi a kontúrok közötti viszonyokat, és lehetővé teszi, hogy a szoftver automatikusan kiválassza a legnagyobb területű, összefüggő objektumot, majd a további geometriai és topológiai jellemzők számítását kizárólag erre az objektumra korlátozza.

1.2. ábra. Kontúrdetektálás

1.4.4. Kép momentumok és tömegközéppont számítás

A háttérkivonás és morfológiai zajszűrés eredményeként kapott bináris maszkon a patkány pozíciójának meghatározása geometriai momentumok segítségével történik [2]. A

momentumok olyan integrált jellegű jellemzők, amelyek a pixelek intenzitásának súlyozott átlagaként írják le az objektum alakját és elhelyezkedését a képen.

Egy $I(x, y)$ intenzitású digitális kép vagy bináris maszk (i, j)-edik rendű nyers momentuma (M_{ij}) a következőképpen definiálható:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y), \tag{1.6}$$

ahol x és y a pixel koordinátái, bináris kép esetén pedig $I(x, y) = 1$, ha a pixel az objektumhoz tartozik, és 0, ha a háttérhez. A patkány tömegközéppontjának koordinátái (C_x, C_y) az elsőrendű momentumok és a nulladrendű momentum hányadosaként számíthatók [2]:

$$C_x = \frac{M_{10}}{M_{00}}, \quad C_y = \frac{M_{01}}{M_{00}}. \tag{1.7}$$

Ez a módszer jól viseli a kisebb kontúrzajokat és lokális egyenetlenségeket, mivel a centroid a teljes objektum pixeleinek eloszlását veszi figyelembe [4]. Ennek köszönhetően a patkány pozíciója stabilan és megbízhatóan becsülhető még enyhén zajos bináris maszk esetén is, ami alapvető a későbbi térbeli tájékozódás, memória és viselkedés vizsgálatokhoz [20].

2. fejezet

Felhasznált anyagok és eszközök

A kutatómunka alapját egy összetett mérőrendszer képezi, amely többféle hardveres komponens integrációjára épül, és amelyhez a dolgozat keretében kifejlesztett szoftver kapcsolódik. Jelen fejezet célja egyrészt az AMBITUS kísérleti berendezés fizikai felépítésének és szenzorarchitektúrájának bemutatása, másrészt azoknak a szoftveres technológiáknak az ismertetése, amelyekre az implementáció épül, továbbá ezek megválasztásának rövid indoklása.

2.1. Hardver komponensek

2.1.1. A berendezés specifikációi

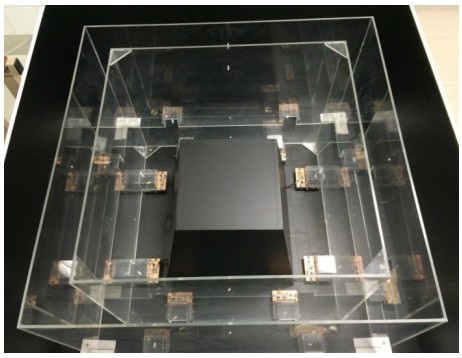
A konstrukció a klasszikus körfolyosó tesztek és labirintusok előnyeit ötvözi, így alkalmas a komplex térbeli tájékozódás, agyi aktivitás és a kíváncsiság vizsgálatára.

A rendszer főbb fizikai paraméterei a műszaki leírás alapján a következők [11, 29]:

- **Befoglaló méretek:** A berendezés külső mérete 800×800 mm, magassága 500 mm [11, 29].
- **Folyosórendszer:** A patkányok mozgását egy 80 mm széles folyosó határolja [11, 29]. A közlekedő térrész geometriáját úgy alakították ki, hogy az állatok számára egyértelmű, jól definiált útvonalakat biztosítson, miközben minimalizálja a beszorulás vagy a holtterek kialakulásának lehetőségét.
- **Fülkék:** A rendszer összesen 16 darab, egyenként $50 \times 50 \times 50$ mm méretű fülkével rendelkezik, amelyek közül 8 a belső, 8 pedig a külső íven helyezkedik el [11, 29].

Ezek a jutalomfalatok elhelyezésére és a jutalomfelvételi viselkedés vizsgálatára lettek kialakítva.

- **Anyaghasználat és környezet:** A folyosórendszer falai átlátszó plexiüvegből készültek, amelyet kívülről egy zárt fekete doboz, belülről pedig egy fekete tetraéder határol [11, 29]. Ez a kialakítás biztosítja, hogy a külső vizuális ingerek ne zavarják az állatot, miközben az infravörös kamerarendszer számára a belső tér jól megfigyelhető marad.



2.1. ábra. Az AMBITUS rendszer

2.1.2. Szenzorrendszer

Az automatikus adatgyűjtést egy nagy pontosságú infravörös szenzorrendszer támogatja. A műszaki leírásnak megfelelően minden fülkénél, valamint a négy folyosószakasz közepén infravörös LED-fotocella párok kerültek beépítésre [11, 29]. Ezek a szenzorok a rendszerhez kapcsolt számítógépen keresztül 1 ms-os időtartamon belül regisztrálják, ha a patkány megszakítja az infravörös fényt [11, 29]. Ez lehetővé teszi az áthaladások időpontjának és sorrendjének precíz rögzítését anélkül, hogy az állat viselkedését befolyásolnák.

2.1.3. Videórendszer - Infravörös kamera

A viselkedés részletes, képi dokumentálását egy infravörös videófelvételre alkalmas eszköz biztosítja [11, 29]. Az eszköz lehetővé teszi az állatok magatartásának megfigyelését infravörös tartományban, így a külső megvilágítás sem zavarja a kísérleti alanyokat. A rögzített videóanyag pixelsűrűsége a 80 cm átmérőjű körfolyosóhoz viszonyítva elegendő

felbontást biztosít a szoftveres kontúrdetektálás és a pontos útvonalkövetés megvalósításához.

2.2. Szoftver technológiák

A videóelemző szoftver fejlesztése során olyan technológiák kerültek alkalmazásra, amelyek egyaránt lehetővé teszik a gyors prototípus készítést, valamint egy robusztus és megbízható képfeldolgozási lánc kialakítását.

2.2.1. Programozási nyelv: Python

A szoftver implementációjára a Python programozási nyelv került kiválasztásra. A Python napjainkra a gépi látás, az adatfeldolgozás és a tudományos számítás területén széles körben elterjedt és szinte ipari szabványnak tekinthető. A választást elsősorban a gazdag könyvtár ökoszisztémája és a rugalmas integrálhatósága indokolta. [27].

A fejlesztés során a Python interpretált jellege biztosította a szükséges rugalmasságot, ami a videóelemzés paramétereinek beállításakor volt kritikus szempont. Az automatikus memóriakezelés és a dinamikus típusosság jelentősen egyszerűsítette a kódot, mivel nem volt szükség manuális memóriaműveletekre a komplex adatszerkezetek kezelésekor. Emellett a Python könnyű összekapcsolhatósága a C/C++ alapú OpenCV könyvtárakkal, amely lehetővé teszi a teljesítménykritikus részek hatékony kihasználását.

2.2.2. Képfeldolgozás: OpenCV

A projekt legfontosabb gépi látás könyvtára az OpenCV [26]. Ez a széles körben elterjedt könyvtár tartalmazza a háttérkivonáshoz használt `MOG2` algoritmus implementációját [1], valamint számos további képfeldolgozó és számítógépes látás algoritmust.

A szoftverben az OpenCV felel többek között:

- A videófolyam beolvasásáért és a *Region of Interest* területek kezeléséért.
- A háttérleválasztásért, amely a mozgásdetektálás és a patkány bináris maszkjának előállítása szempontjából kulcsfontosságú.
- A morfológiai nyitás és zárás műveleteinek végrehajtásáért, amelyek a bináris maszk zajszűrését és a kontúrok tisztítását szolgálják.

2.2.3. Numerikus számítások: NumPy

Az OpenCV a képeket Python oldalon NumPy tömbök formájában reprezentálja. Szürkeárnyalatos képkocka esetén kétdimenziós, színes kép esetén pedig háromdimenziós mátrixként jelenik meg, amelyen a NumPy segítségével nagy sebességű, vektorizált műveletek hajthatók végre [5, 28].

A szoftverben a NumPy könyvtár elsődleges feladata a hatékony adatstruktúrák biztosítása a képmanipulációhoz, például a maszkoláshoz szükséges nullmátrixok létrehozása vagy a poligonok csúcspontjainak formázása. Emellett kulcsszerepet játszik a geometriai számítások vektorizált végrehajtásában. A program ennek segítségével határozza meg valós időben az euklideszi távolságokat a patkány és a folyosók referenciapontjai között, valamint végzi a koordináták átlagolását a geometriai középpontok számításakor.

2.2.4. Adattárolás: CSV

Az elemzés eredményeinek tárolását a Python beépített `CSV` modulja biztosítja. A kimeneti adatok egyszerű vesszővel tagolt formátumban kerülnek mentésre. A választás azért esett erre a formátumra, mert a CSV használata szükségtelenné teszi adatbáziskezelő szoftverek telepítését, így minimalizálja a szoftver környezeti függőségeit, miközben fenntartja a platformfüggetlen hordozhatóságot.

A CSV fájlok közvetlenül importálhatók a legtöbb statisztikai és adatfeldolgozó szoftverbe, illetve táblázatkezelőkbe, így jól illeszkednek a kutatók meglévő munkafolyamataihoz. Ez a megoldás egyszerre nyújt technikai egyszerűséget és rugalmasságot az adatok további feldolgozása során.

3. fejezet

A szoftver architektúrája és implementációja

A fejlesztett szoftver architektúrája egy szekvenciális feldolgozási modellre épül, ami három fő szakaszra bontható. Ezek az inicializálás és interaktív ROI kijelölés, a fő videófeldolgozási ciklus, valamint az adatok exportálása. A rendszer modularitását a funkciók, a `RegionSelector` osztályba, és önálló függvényekbe szervezése biztosítja. Ebben a fejezetben részletesen bemutatom a kódbázis legfontosabb komponenseit és azok belső logikáját.

3.1. Inicializálás és felhasználói beállítások

A program indításakor a `main` függvény végzi el a környezet előkészítését. A szoftver konzolos interakció révén kéri be a futtatáshoz szükséges alapvető paramétereket:

- **Üzemmód kiválasztása:** A felhasználó dönthet arról, hogy a szoftver `debug` módban fusson. Bekapcsolt állapotban, amit az `y` gomb megnyomásával érünk el, a rendszer valós idejű vizuális visszacsatolást nyújt OpenCV ablakban, megjelenítve a detektált kontúrokat, a régiókat és a státuszüzeneteket. Kikapcsolt állapotban a feldolgozás a háttérben zajlik, ami jelentősen növeli a futási sebességet.
- **Jutalmak száma:** Mivel a kísérleti elrendezés változhat, a program dinamikusan kezeli a jutalomzónák számát, amelyet indításkor kell megadni.

A sikeres paraméterezés után a szoftver betölti a videófájlt, és kinyeri az első képkockát, amely a régiókijelölés alapjául szolgál.

3.2. Interaktív régiókijelölés

A videóelemzés kritikus előfeltétele a vizsgált területek, azaz a jutalmak, ablakok és folyosók precíz meghatározása. Ezt a feladatot a `RegionSelector` osztály látja el.

3.2.1. Eseményvezérelt működés

A kijelölési folyamat az OpenCV könyvtár `setMouseCallback` mechanizmusára épül [26]. A metódus egy eseményfigyelő, amely valós időben reagál az egérműveletekre. A rendszer három alapvető egéreseményt kezel:

- `EVENT_LBUTTONDOWN`: A kijelölés kezdetét jelzi. Folyosók esetén ez egy új csúcspont lerakását, ablakok és jutalmak esetén a téglalap sarkának rögzítését jelenti.
- `EVENT_MOUSEMOVE`: Ez felel a vizuális visszacsatolásért. Téglalap rajzolásakor a rendszer folyamatosan frissíti az alakzatot, hogy a felhasználó lássa a várható végeredményt a gomb felengedése előtt.
- `EVENT_LBUTTONUP`: A kijelölés véglegesítése és a koordináták memóriába mentése.

3.2.2. A kijelölési módok állapotai

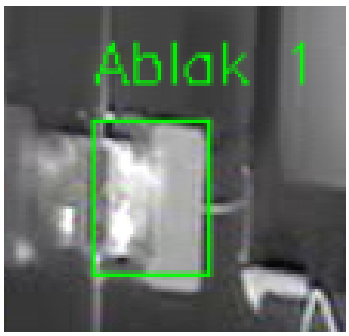
Az osztály belső logikáját a `modes` szótár és a `current_mode_idx` változó vezérli. A rendszer három, egymástól elkülönülő állapotban működhet, melyek között a felhasználó az ’n’, ’a’ és ’c’ billentyűk valamelyikével válthat az aktuális módtól függően:

1. **Folyosó:** A kód ebben a módban pontosan 4 csúcspontból álló négyszögeket vár. A felhasználó egyesével kattintva jelöli ki a sarkokat, amelyeket a rendszer kék színű vonallal köt össze. Ha a 4. pontot is megadtuk, a rendszer lezárja a poligont.

3.1. ábra. Sarokpontok kijelölése

3.2. ábra. Folyosó kijelölve

2. **Ablak:** Itt a rendszer egérhúzással, a bal gomb lenyomva tartása mellett kijelölhető téglalapokat kezel. A kijelölés zöld színnel jelenik meg, és a szoftver maximum 8 ilyen területet engedélyez.



3.3. ábra. Ablak kijelölés

3. **Jutalom:** Szintén téglalap alakú, egérhúzással történő kijelölés piros színnel. Logikailag elkülönül az ablakoktól. Míg a folyosók és ablakok koordinátái eltárolásra kerülnek, a jutalmakat minden futtatáskor újra meg kell adni, mivel pozíciójuk változhat.

3.4. ábra. Jutalom kijelölés

3.2.3. Perzisztencia és vezérlés

A felhasználói élmény javítása és a munkafolyamat gyorsítása érdekében a régiókat kezelő osztály támogatja a gyorsbillentyűk használatát és az adatok mentését:

- **Adatmentés:** A kijelölt folyosó és ablakrégiók koordinátáit a rendszer automatikusan elmenti a `regions.csv` fájlba a `save_regions_to_single_csv` függvény segítségével.
- **Billentyűparancsok:**
  - `'n'`: Váltás a következő kijelölési módra.
  - `'a'`: Visszalépés az ablak és folyosó kijelöléshez. Ez csak a jutalom kijelölés fázisban aktív.
  - `'z'`: Az utolsó hibás kijelölés visszavonása.
  - `'c'`: Ablak és folyosó kijelölés után váltás jutalom kijelölő módba, illetve jutalom kijelölés befejezése és az elemzés indítása.
  - `'q'` vagy `'Esc'`: Program bezárása

3.3. A videóelemző motor

1. Algoritmus A videóelemző szoftver fő vezérlési logikája

Bemenet: Videófájl, N db jutalom
Kimenet: Eseménynapló (CSV), Pozíció adatok (CSV)
$cap \leftarrow$ Videó megnyitása
$bg\_subtractor \leftarrow$ MOG2 inicializálása
if létezik *regions.csv* **then**
    Régiók betöltése fájlból
else
    $RegionSelector$ futtatása (Folyosók, Ablakok)
    Régiók mentése *regions.csv*-be
end if
Jutalmak kijelölése interaktívan
$ROI\_mask \leftarrow$ Régiók maszkjának létrehozása
while van következő képkocka **do**
    $frame \leftarrow$ képkocka beolvasása
    $raw\_mask \leftarrow bg\_subtractor$.apply($frame$)
    $mask \leftarrow raw\_mask$ **AND** ROI_mask ▷ Maszkolás
    $mask \leftarrow$ Morfológiai szűrés ($mask$) ▷ Zajszűrés
    $contours \leftarrow$ Kontúrkeresés($mask$)
    if van MIN_AREA-nál nagyobb kontúr **then**
        if nincs $last\_pos$ **then**
            $rat \leftarrow$ legnagyobb területű kontúr
        else
            $rat \leftarrow last\_pos$-hoz legközelebbi kontúr
        end if
        $pos \leftarrow$ MomentumSzámítás(rat) ▷ Centroid
        $last\_pos \leftarrow pos$
        $frames\_lost \leftarrow 0$
    else
        if $frames\_lost < MAX\_LOST$ **then**
            $pos \leftarrow last\_pos$ ▷ Pozícióbecslés
            $frames\_lost \leftarrow frames\_lost + 1$
        end if
    end if
    if pos érvényes **then**
        Vizsgálat: pos benne van-e a Folyosó/Ablak poligonokban?
        Vizsgálat: pos áthaladt-e a folyosó középpontján?
        if rat metszi a Jutalom ROI-t **és** még aktív **then**
            Jutalom státusz $\leftarrow$ BEGYŰJTVE
        end if
        Adatok írása a CSV fájlokba
    end if
end while
Erőforrások felszabadítása

3.3.1. Adaptív háttérmodellezés

A mozgó objektum és a statikus környezet szétválasztására a `MOG2` algoritmust alkalmaztam [1]. A `cv2.createBackgroundSubtractorMOG2` függvény inicializálása az alábbi paraméterekkel történik:

- `history = 1000`: Az algoritmus az utolsó 1000 képkocka információit veszi figyelembe a háttérmodell frissítésekor.
- `varThreshold = 60`: A pixelvariancia küszöbértéke, amely meghatározza az érzékenységet.
- `detectShadows = False`: Bár az algoritmus képes lenne árnyékok jelölésére, a feldolgozási sebesség növelése és a bináris maszk egyszerűbb kezelhetősége érdekében ezt a funkciót kikapcsoltam.

A kód tartalmaz egy úgynevezett bemelegítési fázist, ahol az első 100 képkockán a detektálási logika még inaktív, kizárólag a háttérmodell tanulása zajlik.

3.3.2. Zajszűrés és Morfológia

A nyers háttérmaszk feldolgozása több lépésben történik a zajok eltávolítása érdekében [4]:

1. **Biztonsági küszöbölés:** Bár az árnyékdetektálás ki van kapcsolva, a rendszer egy explicit binarizálási lépést alkalmaz, biztosítva, hogy a maszk kizárólag 0 és 255 értékeket tartalmazzon.
2. **Zárás:** A `cv2.morphologyEx` függvénnyel végzett művelet segít kitölteni a patkány testén belüli esetleges lyukakat, és összekapcsolja a közeli objektumtöredékeket.
3. **Nyitás:** Ezt követően egy nyitás művelet távolítja el a kisméretű, elszigetelt fehér zajpontokat a háttérből.
4. **Median Blur:** Végül egy 5x5-ös medián szűrő simítja el a maszk éleit.

3.3.3. Objektumkövetés

A tisztított maszkon a `cv2.findContours` függvény keresi meg az összefüggő komponenseket [3]. A szoftver az objektumazonosításhoz egy hibrid stratégiát alkalmaz. A detektálás kezdetekor a legnagyobb területű kontúrt tekinti a kísérleti alanynak. A folyamatos követés során azonban a zajok okozta tévesztések elkerülése érdekében a program azt a kontúrt választja ki az érvényes jelöltek közül, amelynek tömegközéppontja a legközelebb esik az előző képkockán rögzített pozícióhoz.

3.3.4. Pozícióbecslés

A szoftver robusztusságát növeli a követés elvesztésének kezelése. Amennyiben egy képkockán a detektálás sikertelen, a kód 1000 képkocka erejéig megőrzi az utolsó ismert érvényes pozíciót. Ebben a becslés fázisban a szoftver feltételezi, hogy a patkány a legutóbbi helyén tartózkodik.

3.4. Eseménydetektálás logikája

A szoftver kétféle módszert alkalmaz annak eldöntésére, hogy a patkány egy adott régióban tartózkodik-e az objektumdetektálás sikerességtől függően. Ez a hibrid megközelítés biztosítja a pontosságot aktív követés esetén, és a robusztusságot jelvesztés esetén.

3.4.1. Geometriai átfedés vizsgálat

Amikor a rendszer sikeresen detektálja a patkány kontúrját mozgáskövetés során, az OpenCV `intersectConvexConvex` függvényét használja [26].

- **Poligonok és Téglalapok:** A kód nem pusztán a középpontot vizsgálja, hanem ellenőrzi, hogy a patkány konvex burka metszi-e a régiók konvex burkainak valamelyikét. Ez a módszer rendkívül pontos, mivel akkor is jelez, ha a patkány teste csak részben lóg bele a területbe.

3.4.2. Koordináta alapú vizsgálat és becslés

Amennyiben a patkány kontúrja nem található, például mozdulatlanság miatt beleolvad a háttérbe, a rendszer a `last_position` pontkoordinátáit használja, feltételezve, hogy

az állat nem mozdult el jelentősen. Ebben az esetben a számításigényes konvex metszetvizsgálat helyett pont alapú ellenőrzés fut le:

- **Folyosók:** A `cv2.pointPolygonTest` függvénnyel vizsgálja, hogy az utolsó ismert pont a poligonon belül van-e.
- **Ablakok és Jutalmak:** Egyszerű koordináta összehasonlítással történik a vizsgálat.

3.5. ábra. Pozícióbecslés logikája

3.4.3. Sebességmérés és áthaladás detektálás

A program inicializáláskor kiszámítja minden folyosó geometriai középpontját. A futás során a kód figyeli a patkány távolságát ezektől a fix pontoktól.

- **Áthaladás:** Ha a távolság 30 pixel alá csökken, és az áthaladás még nem volt regisztrálva az adott körben, a rendszer eseményt rögzít.
- **Reset:** A rendszer akkor tekinti az áthaladást befejezettnek, ha a patkány távolsága a középponttól meghaladja a küszöbérték kétszeresét.

3.5. Vizualizáció és Adatexport

Amennyiben a debug mód aktív, a szoftver valós idejű vizuális visszajelzést biztosít a felhasználó számára az elemzés során. A még be nem gyűjtött jutalmakat a rendszer piros kerettel emeli ki, míg a már megszerzett jutalmakat szürke, áthúzott négyzettel jelöli.

Az elemzési folyamat lezárásakor a program a kinyert adatokat három strukturált CSV állományba exportálja, biztosítva ezzel a részletes utólagos feldolgozhatóságot:

- `{videó_név}_speed_recording.csv`: A folyosókon történő áthaladások diszkrét eseményeit rögzíti, amely elsődlegesen a sebességprofilok meghatározását szolgálja.

- `{videó_név}.csv`: Egy átfogó, bináris állapotokat tartalmazó mátrix, amely képkockánkénti felbontásban jelzi a kísérleti alany jelenlétét a definiált érdeklődési területeken, valamint a jutalmak aktuális státuszát.

- `{videó_név}_continuous_positions.csv`: Ez az állomány a kísérleti alany mozgáspályájának folyamatos rögzítését tartalmazza. A táblázat soronként tárolja a vizsgált képkocka sorszámát, az állat tömegközéppontjának `x` és `y` pixel koordinátáit, valamint egy `estimated` státuszjelzőt. Utóbbi logikai értéke arról tájékoztat, hogy az adott pozíció valós detektálás eredménye-e, vagy rövid jelvesztés esetén a szoftver által ismert utolsó pozíció.

3.6. A választott módszertan összehasonlítása a mélytanulási eljárásokkal

Napjainkban a számítógépes látás területén egyre dominánsabbá válnak a mélytanuláson alapuló megoldások, mint például a DeepLabCut [9]. Ezek a rendszerek képesek komplex környezetben testrészek precíz követésére is, a jelenlegi fejlesztés során tudatos döntés volt a klasszikus megközelítés alkalmazása.

Ezt a döntést az alábbi érvek indokolják:

- **Tanítóadatbázis szükségtelensége:** A mélytanulási modellek tanulást igényelnek, amelyhez több száz vagy ezer képkockát kell manuálisan annotálni a tanítás előtt. Ezzel szemben az implementált algoritmus nem igényel előzetes tanítást, a videó elindítása után azonnal képes alkalmazkodni a felhasználási környezethez és működni.

- **Hardverigény és hordozhatóság:** A modern neurális hálók futtatásához és különösen a tanításához nagy teljesítményű dedikált videókártya szükséges [10]. A szoftver ezzel szemben rendkívül alacsony erőforrásigényű, és egy átlagos irodai laptop processzorán is valós időben, nagy sebességgel futtatható.

- **Kompaktság és telepíthetőség:** Míg egy Deep Learning környezet telepítése TensorFlow, PyTorch függőségek, CUDA driverekkel együtt bonyolult és több gigabájt

tárhelyet is foglalhat, a jelenlegi program mindössze egyetlen Python fájlból és néhány alapvető könyvtárból áll. Ez jelentősen megkönnyíti a szoftver telepítését a kutatólaboratóriumi környezetben.

- **A feladat jellege:** Az AMBITUS berendezés kontrollált környezetet biztosít, ahol a patkány kontúrja jól elkülöníthető a háttértől. Ebben a speciális esetben a klasszikus algoritmusok is kiváló eredményt adnak. Nincs szükség a Deep Learning komplexitására a jelenlegi kutatási célokhoz, azaz a pozíció és a mozgás aktivitásának rögzítéséhez.

Összességében a választott architektúra talán a legideálisabb eszköz a feladatra. Igaz, technológiailag kevésbé komplex, mint a neurális hálók, a felhasználási célnak véleményem szerint tökéletesen megfelel, miközben fenntartja a gyorsaságot és a könnyű kezelhetőséget.

4. fejezet

Kiemelt implementációs részletek

Ebben a fejezetben a szoftver legkritikusabb algoritmusai kerülnek bemutatásra.

4.1. Képelőfeldolgozási futószalag

A videóelemzés magja az `analyze_video` függvény. Minden képkockán végrehajtásra kerül a háttérkivonás a MOG2 algoritmussal, amelyet a zajszűrési lépések követnek.

```
# 1. Hattermaszk generalasa MOG2-vel
fg_mask = bg_sub.apply(frame, learningRate=-1)

# 2. ROI Maszkolas: Csak a kijelolt teruletek vizsgalata
# A tobbi resz nullazasa a zajok elkerulese erdekeben
fg_mask = cv2.bitwise_and(fg_mask, fg_mask, mask=roi_mask)

# 3. Binaris kuszoboles
_, fg_mask = cv2.threshold(fg_mask, 200, 255, cv2.THRESH_BINARY)

# 4. Morfologiai muveletek
# Zaras: lyukak betomese a patkany testen belul
fg_mask = cv2.morphologyEx(fg_mask, cv2.MORPH_CLOSE,
                           kernel_close, iterations=MORPH_CLOSE)
# Nyitas: apro zajpontok eltavolitasa a hatterbol, median szures
fg_mask = cv2.morphologyEx(fg_mask, cv2.MORPH_OPEN,
                           kernel_open, iterations=MORPH_OPEN)
fg_mask = cv2.medianBlur(fg_mask, 5)
```

4.1. Listing. Háttérkivonás, ROI maszkolás és morfológiai szűrés

4.2. Objektumkövetés és centroid számítás

A megtisztított maszkon a `cv2.findContours` függvény keresi meg az összefüggő objektumokat. A hibrid követési stratégiát stabilabbá teszi zajos környezetben.

```
if contours:
    # Csak a relevans meretu konturok vizsgalata
    valid_contours = [c for c in contours if cv2.contourArea(c) > MIN_AREA]

    if valid_contours:
        # Ha meg nincs elozo pozicio, a legnagyobb konturt keressuk
        if last_position is None:
            rat_contour = max(valid_contours, key=cv2.contourArea)
        else:
            # Ha van, akkor a legutobbi poziciohoz legkozelebb esot
            rat_contour = min(valid_contours, key=lambda c: np.linalg.norm(
                np.array(calculate_polygon_center(c)) - np.array(last_position)))

        M = cv2.moments(rat_contour)

        # Tomegkozeppont szamitasa:
        if M["m00"] != 0:
            cx = int(M["m10"] / M["m00"])
            cy = int(M["m01"] / M["m00"])
            current_position = (cx, cy)

            # Pozicio mentese a kovetkezo kepkockakra
            last_position = current_position
            frames_lost = 0
```

4.2. Listing. A patkány pozíciójának meghatározása hibrid stratégiával

4.3. Interaktív régiókijelölés eseménykezelése

A `RegionSelector` osztály felel a felhasználó által kijelölt régiók valós idejű feldolgozásáért. Az alábbi kódrészlet demonstrálja, hogyan különbözteti meg a rendszer a pontonként definiált poligonokat és a húzással kijelölt téglalapokat.

```
def mouse_callback(self, event, x, y, flags, param):
    mode = self.get_current_mode()

    if mode == "folyoso":
        # Folyosok eseten pontonkenti kijeloles
        if event == cv2.EVENT_LBUTTONDOWN:
            self.temp_corridor_points.append((x, y))
            # Ha megvan a 4. pont, lezarjuk a poligont
            if len(self.temp_corridor_points) == 4:
                self.modes["folyoso"]["regions"].append(
                    self.temp_corridor_points.copy()
                )
                self.temp_corridor_points.clear()
    else:
        # Ablakok es jutalmak eseten teglalap huzasa
        if event == cv2.EVENT_LBUTTONDOWN:
            if len(self.modes[mode]["regions"]) < self.modes[mode]["max
"]:
                self.start_point = (x, y)
                self.drawing = True
        elif event == cv2.EVENT_MOUSEMOVE and self.drawing:
            # Vizualizacio frissitese huzas kozben
            self.current_rect = (self.start_point, (x, y))
        elif event == cv2.EVENT_LBUTTONUP and self.drawing:
            # Kijeloles veglegesitese
            self.drawing = False
            end_point = (x, y)
            self.modes[mode]["regions"].append((self.start_point,
end_point))
            self.current_rect = None
```

4.3. Listing. Egéresemények kezelése a RegionSelector osztályban

4.4. Sebességmérés és adatrögzítés

Az adatgyűjtés két módszerrel valósul meg. A rendszer regisztrálja a folyosók középpontján történő áthaladásokat, valamint ezzel párhuzamosan folyamatos helymeghatározást és rögzítést is végez.

4.4.1. Áthaladás detektálás hiszterézissel

A folyosókon történő áthaladás regisztrálása egy hiszterézisen alapuló logikára épül. Az esemény akkor aktiválódik, ha az alany távolsága a folyosó geometriai középpontjától egy küszöbérték alá csökken. A téves, többszörös detektálások elkerülése érdekében a rendszer csak akkor állítja alaphelyzetbe a figyelést, ha az állat jelentősen eltávolodik ettől a ponttól.

```
# Tavolsag szamitasa a folyoso kozeppontjatol
distance = np.sqrt((current_position[0] - midpoint[0])**2 +
                   (current_position[1] - midpoint[1])**2)

# 1. Esemeny aktivalasa: Szigoru kuszobertek
if distance <= MIDPOINT_THRESHOLD and not midpoint_crossed[i]:
    midpoint_crossed[i] = True
    # Esemeny mentese a sebesseg-szamitashoz
    speed_recordings.append({
        'corridor': i + 1,
        'frame': frame_idx,
        'time_sec': frame_idx / fps,
        'rat_x': current_position[0],
        'rat_y': current_position[1]
    })

# 2. Reset: Csak akkor, ha mar messzire (2x) tavolodott
elif distance > MIDPOINT_THRESHOLD * 2:
    midpoint_crossed[i] = False
```

4.4. Listing. Középponti áthaladás detektálása hiszterézissel

4.4.2. Pontos tartózkodási hely rögzítése

A diszkrét eseményeken túl a szoftver minden képkockán rögzíti az állat tömegközéppontjának (x, y) koordinátáit. A kód egyik kiemelt funkciója a pozícióbecslés. A becsült értékeket egy `estimated` jelzővel látja el a kimeneti fájlban. Ez lehetővé teszi az utólagos elemzés során a valós és a szoftver által interpolált adatok szétválasztását.

```
if rat_contour is not None:
    # 1. eset: Sikeres detektalas - Valos koordinatak
    pos_writer.writerow([frame_idx, current_position[0],
    current_position[1], False])

elif last_position is not None and frames_lost < MAX_LOST_FRAMES:
    # 2. eset: Jelvesztes - Becsles az utolso ismert pozicio alapjan
    # Az 'estimated' flag True erteke jelzi az adatbizonytalansagot
    pos_writer.writerow([frame_idx, last_position[0], last_position[1],
    True])

else:
    # 3. eset: Tartos jelvesztes - Nincs adat
    pos_writer.writerow([frame_idx, "", "", ""])
```

4.5. Listing. Folyamatos pozíciómentés és státuszjelzés

5. fejezet

Kísérleti eredmények

5.1. Tesztrendszer konfigurációja

A szoftver teljesítményét és megbízhatóságát a következő konfiguráción mértem:

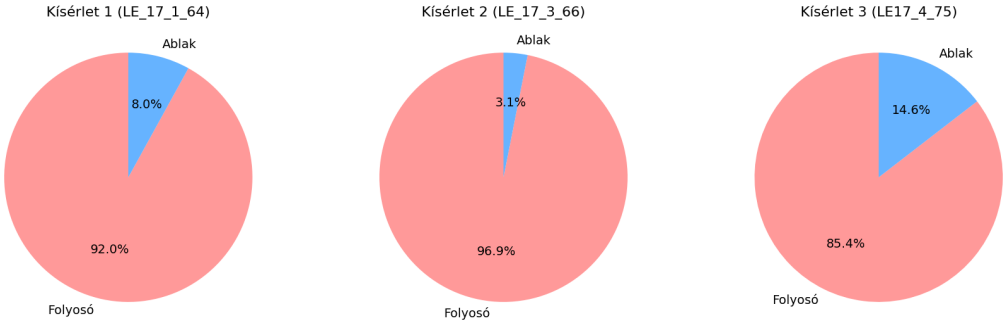
- CPU: AMD Ryzen 7 5000 series
- RAM: 32 GB
- Operációs rendszer: Windows 11
- Feldolgozási idő: Az 5 perces videók feldolgozása körülbelül 30 másodperc alatt lezajlott.
- Memóriahasználat: A futás során nem haladta meg a 100 MB-ot.

5.2. Kísérletek vizualizációja és elemzése

A kísérleti adatok értelmezéséhez a nyers adatok grafikusan kerültek ábrázolásra. Ennek érdekében egy külön adatfeldolgozó és vizualizációs szkript került implementálásra. Ez a program a szoftver által generált CSV-ket dolgozza fel, és ezek alapján állítja elő a diagramokat. Az alábbiakban ezen vizualizációk és a belőlük levonható megállapítások kerülnek ismertetésre.

5.2.1. Időbeli eloszlás vizsgálata

Az elemzés első lépése a különböző régiókban eltöltött idő vizsgálata. A szoftver által generált adatok segítségével számszerűsíthetővé vált a tartózkodási idő eloszlása a folyosók és ablakok között, amely lehetővé teszi a passzív közlekedés és az aktív környezetfelmérés elkülönítését.

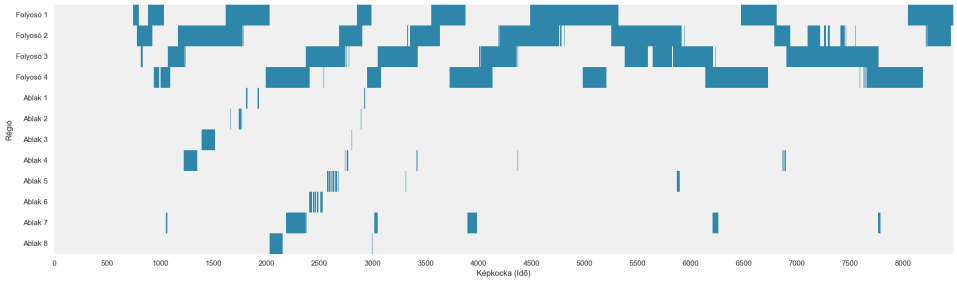


5.1. ábra. Összesített idő – Folyosó és ablak régiók kísérletenként

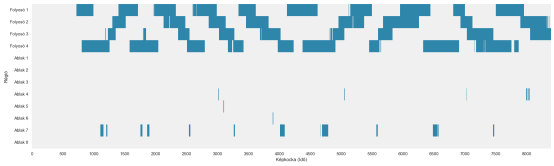
Elemzés: A fenti kördiagramok nemcsak az általános megoszlást szemléltetik, hanem remekül kirajzolják az egyedek közötti karakterbeli különbségeket is. Míg a 2. kísérlet alanya idejének mindössze 3,1%-át töltötte a felfedezés vizsgálatára szolgáló ablakokban, ami erős passzivitásra utal, addig a 3. kísérletben részt vevő állat közel ötször annyi időt szentelt a környezet aktív vizsgálatára. Ez az adatvizualizáció azonnal láthatóvá teszi a különbséget a félénkebb és a kíváncsibb egyedek stratégiája között, igazolva, hogy a rendszer pontosan detektálja a finom viselkedésbeli eltéréseket is.

5.2.2. Mozgásmintázatok feltérképezése

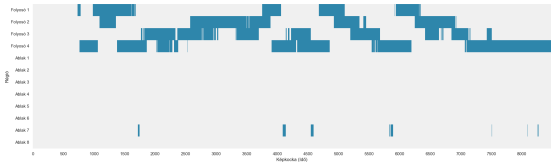
A részletesebb térbeli elemzéshez a patkányok pozíciója a kinyert koordináták alapján hőtérképeken kerültek ábrázolásra. Ez a vizualizáció megmutatja a térhasználat egyenletességét és a preferált tartózkodási helyeket.



5.2. ábra. LE17_4_75 – Magas aktivitású egyed



5.3. ábra. LE_17_1_64

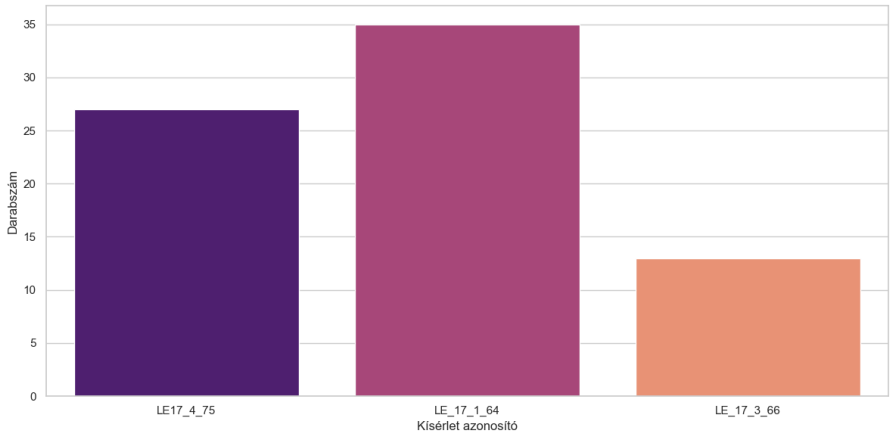


5.4. ábra. LE_17_3_66

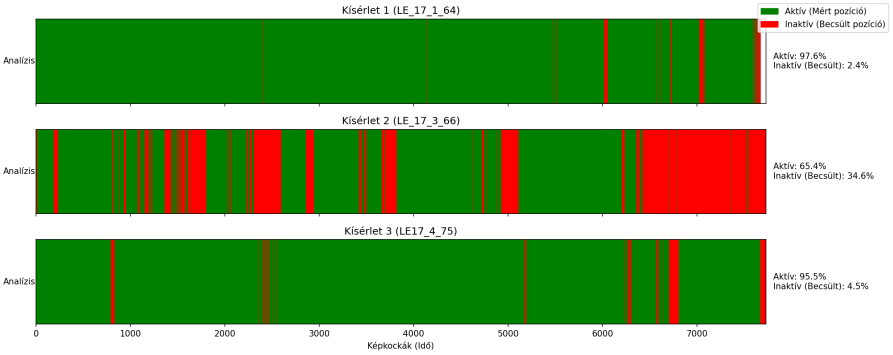
Elemzés: A hőtérképek összehasonlítása során szembetűnő különbségek fedezhetők fel az egyedek stratégiái között. Az első ábrán látható patkány szisztematikusan bejárta a teljes teret, ami magas szintű explorációs hajlandóságra utal. Ezzel szemben a másik két egyed mozgása főként a folyosókra, illetve pár ablakra korlátozódott, ami a térbeli memória hiányosságaira vagy csökkent motivációra enged következtetni. A szoftver által biztosított pozícióadatok lehetővé teszik a hely preferenciák és az útvonal stratégiák automatikus felismerését.

5.2.3. Aktivitási szint mérése

A mozgás folyamatosságának és az aktivitás mértékének jellemzésére a feldolgozó szkript a középponti áthaladások és becsült vagy mért pozíciók alapján vizualizálta a mozgást.



5.5. ábra. Középpont áthaladások száma kísérletenként

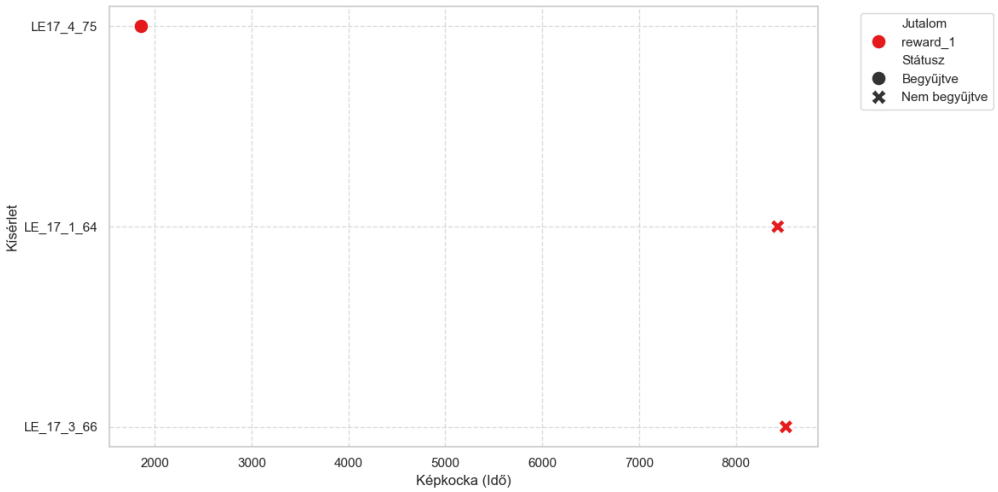


5.6. ábra. Aktivitás mérése becsült vagy mért pozíció alapján

Elemzés: A rendszer az áthaladási események gyakorisága és a pontos, mért pozíciók révén képes számszerűsíteni a lokomotoros aktivitást. Az első diagramon látható magas áthaladási szám a `LE17_4_75` kísérlet esetén korrelál a hőtérképen látott területi lefedettséggel. A második ábrán a folyamatos pozíciók alapján vizualizálja az aktivitást a program, feltételezve, hogy ha becsült értéket mentünk a CSV-be, akkor az állat egyhelyben tartózkodik, azaz pillanatnyilag nem aktív. Ezek a módszerek objektív alternatívát kínálhatnak a szubjektív, mozgékony vagy lusta aktivitási kategóriák helyett, lehetővé téve a viselkedési dinamika statisztikai összehasonlítását. Ezen mérések alapján a szoftver segítségével gyorsan kiszűrhetők azok az állatok, amelyek lefagytak vagy nem a megszokott módon viselkedtek.

5.2.4. Jutalomkeresési hatékonyság

A kísérlet végső célja a tanulási képesség vizsgálata volt a jutalomfalatok megtalálásán keresztül. A szoftver automatikusan rögzítette a jutalomzónába lépést.



5.7. ábra. Jutalom események

Elemzés: A jutalom események automatikus kimutatása támogatja a viselkedés alapú validációt. Az eredményekből kiderül, hogy a hőtérképeken legaktívabbnak bizonyuló patkány sikeresen megtalálta a jutalmat, míg a passzívabb egyedek nem. Ez az összefüggés megerősíti a kapcsolatot a környezetfelmérési aktivitás és a problémamegoldó képesség között. A szoftveres detektálás kiküszöböli a manuális jegyzetelésből adódó emberi hibákat, és pontos időbélyeget rendel a siker pillanatához, ami elengedhetetlen a tanulási görbék későbbi elemzéséhez.

5.3. Összefoglaló a rendszer megbízhatóságáról

Az elvégzett kísérletek és a fenti elemzések alapján az alábbi technikai és módszertani következtetések vonhatók le:

- **Stabilitás:** A videóelemző szoftver nagy biztonsággal képes detektálni a folyosókon és ablakokban való tartózkodást, illetve a középponti áthaladásokat is. Tévesen csak ritkán érzékel abban az esetben, ha az állat nagyon közel kerül egy vizsgált ablakhoz.

- **Adatkinyerés:** Jelentős mennyiségű adat érkezik áttekinthető CSV formátumban. A szoftver becslés során is pontos adatokat szolgáltat, amelyekből az adatfeldolgozó szkript segítségével valós időben strukturált és vizualizált eredmények kinyerése válik lehetségessé. Ez a kétlépcsős folyamat illeszkedik a modern adatfeldolgozási láncokhoz.

- **Robusztusság:** A detektálás hitelesítését kézi felülvizsgálattal végeztem el. A mozgásdetektálás pontossága kiemelkedő, a téves detekciók aránya alacsony, valamint sikerült kiküszöbölni a korábbi problémát, miszerint a becslés során volt, hogy a szoftver a vizsgált területeken kívül érzékelt, ezáltal a CSV-kben eltűnt az állat erre az időre, így értékes adatt veszett el. Másrészt a MOG2 algoritmus adaptivitásának is köszönhető a megbízhatóság, amely hatékonyan választja le a statikus hátteret a mozgó alanyról.

Az algoritmus strapabíró, jól tűri a kísérleti környezetben előforduló zajokat, így a kutatómunkához biztonsággal alkalmazható az AMBITUS rendszer kiegészítéseként.

6. fejezet

Jövőbeli fejlesztések

6.1. Mélytanulás alapú megközelítések

Jelenleg a háttérkivonást adaptív MOG2 módszerrel végezzük, ami jól működik, de korlátozott bizonyos környezeti zavarok és megvilágítás változások kezelésében. Fejlesztésként mélytanulási modellek bevezetése lenne a legmegfelelőbb, amelyek képesek hatékonyabban felismerni a mozgó objektumokat a változó fényviszonyok között is.

6.2. Viselkedés kategorizálás és automata felismerés

A patkány viselkedésének részletesebb megértése érdekében a mozgáselemek automatikus kategorizálására és mintázatfelismerésre is alkalmazható modellek hasznosak lehetnek. Ehhez konvolúciós neurális hálók integrációja lenne alkalmas, amelyek képesek lennének a viselkedés komplex dinamikáinak felismerésére, például ismétlődő mozgások, pihenő szakaszok vagy jutalomkeresési minták azonosítására.

6.3. GPU gyorsítás

Bár a jelenlegi CPU alapú feldolgozás sebessége a kutatáshoz használt videók esetén megfelelő, de a későbbiekben érdemes lehet a CUDA alapú GPU gyorsítás implementálása. Ez lehetővé tenné a valós idejű feldolgozást nagyobb felbontású videók esetén, illetve más kutásokban is könnyedén alkalmazható lenne, így jelentősen javítaná a szoftver teljesítményét.

6.4. Webes felhasználói felület

A szoftver egy felhőalapú webes platformhoz integrálható, amely a feldolgozott adatokat összegzi és vizualizálja a kutatók számára. Ez a platform lehetővé tenné az eredmények távoli megtekintését, az adatok összehasonlítását, a paraméterek módosítását, valamint az információk egyszerű megosztását, ami jelentősen növeli a kutatási folyamat hatékonyságát és rugalmasságát.

6.5. Paraméterfelderítés és automata kalibrálás

A szoftver a különböző berendezéseken és felvételi körülmények között való megbízható használatához a paraméterek automatikus kalibrálása hasznos fejlesztés lenne. Gépi tanulás alapú eljárások integrálásával a rendszer képes lenne alkalmazkodni a környezethez, optimalizálva az érzékenységet és az elemzés pontosságát anélkül, hogy manuális beavatkozásra lenne szükség.

Összefoglalás

Jelen szakdolgozat keretében egy olyan videóelemző szoftver került kifejlesztésre, amely hatékonyan és automatizáltan képes feldolgozni az AMBITUS kísérleti rendszerben rögzített infravörös felvételeket. A fejlesztés elsődleges célja egy olyan rugalmasan paraméterezhető eszköz létrehozása volt, amely minimális felhasználói beavatkozás mellett képes a kísérletek közti különbségekhez alkalmazkodni, biztosítva ezzel a kutatási eredmények konzisztenciáját és reprodukálhatóságát. A rendszer kialakítása támogatja a szinte teljesen automatikus működést, ugyanakkor a beépített, opcionális vizuális visszacsatolás révén lehetőséget ad a detektálási folyamat valós idejű ellenőrzésére és a paraméterek finomhangolására is.

A szoftver egyik kiemelt funkcionalitása a vizsgálati régiók intelligens kezelése. A rendszer maradandó adattárolást valósít meg, így a felhasználó által grafikusan definiált geometriai alakzatokat a `regions.csv` állományba menti, ezáltal azok a későbbi mérések során újrarajzolás nélkül, azonnal újrafelhasználhatóak, jelentősen gyorsítva ezzel a sorozatvizsgálatok kiértékelését. Az elemzés eredményeként a szoftver részletes, többszintű adatszolgáltatást nyújt. Három elkülönített CSV állományt generál az adatok széleskörű elemzéséhez. Az első a teljes időbeli lefutást rögzíti képkockánkénti felbontásban, dokumentálva a patkány vizsgált területeken lévő tartózkodási helyét és a jutalmak státuszát. A második kifejezetten a mozgásdinamikai elemzéseket támogatja a folyosókon történő áthaladások és bizonyos sebességmérési események rögzítésével. A harmadik pedig a folyamatos pozíció képkockánkénti rögzítését teszi lehetővé, így nem csak az átlagsebességet, de a maximum és minimum sebességet is mérhetjük. Ráadásul az aktivitás másként is megfigyelhetővé válik azáltal, hogy az `estimated` jelzőt figyeljük, mert az megmutatja mennyi ideig volt mozdulatlan az állat.

A rendszer validálása három valós kísérleti videóanyag feldolgozásával történt. Az eredmények azt mutatják, hogy az alkalmazott adaptív háttérmodellezés és a momentum

alapú pozícióbecslés biztos és pontos megoldást kínál a követési feladatokra, még a rövid idejű jelvesztések vagy a patkány mozdulatlansága esetén is. A szoftver megbízhatóan detektálja a komplex viselkedési mintázatokat, beleértve a jutalomfelvételt és a térbeli tájékozódást. Összességében megállapítható, hogy az elkészült alkalmazás sikeresen hidalja át a tisztán hardveres adatgyűjtés korlátait, és egy objektív vizuális alapú adatrögzítési réteggel egészíti ki a rendszer szenzoros méréseit, ezáltal átfogóbb képet adva a kísérleti állatok magatartásáról.

Irodalomjegyzék

[1] Zivkovic, Z. Improved adaptive Gaussian mixture model for background subtraction. *Proceedings of the International Conference on Pattern Recognition*, 2004.

[2] Mukundan, R., Ramakrishnan, K. R. *Moment functions in image analysis: theory and applications*. World Scientific, 1998.

[3] Suzuki, S., Abe, K. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 1985.

[4] Gonzalez, R. C., Woods, R. E. *Digital Image Processing*. Pearson, 2018.

[5] Harris, C. R. et al. Array programming with NumPy. *Nature*, 2020.

[6] Bradski, G., Kaehler, A. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

[7] Bouwmans, T. et al. Background modeling and subtraction algorithms. *Applied Sciences*, 2019.

[8] Dell, A. I. et al. Automated image-based tracking and its application in ecology. *Trends in Ecology & Evolution*, 2014.

[9] Mathis, A. et al. DeepLabCut: markerless pose estimation with deep learning. *Nature Neuroscience*, 2018.

[10] NVIDIA Corporation. *CUDA Programming Guide*. https://docs.nvidia.com/cuda/, 2024. Utolsó látogatás dátuma: 2025.12.12.

[11] Horvath, G. et al. Characterization of exploratory activity in the AMBITUS system. *Physiology and Behavior*, 2017.

[12] Vorhees, C. V., Williams, M. T. Assessing spatial learning and memory in rodents. *ILAR Journal*, 2014.

[13] Buzsáki, G., Moser, E. I. Memory, navigation and theta rhythm. *Nature Neuroscience*, 2013.

[14] Casarrubea, M. et al. Temporal patterns analysis of rat behavior in hole-board. *Behavioral Brain Research*, 2010.

[15] Chambon, C. et al. Automated assessment of rat recognition memory. *Behavioral Brain Research*, 2011.

[16] Kraeuter, A. K. et al. The Y-Maze for assessment of spatial memory in mice. *Methods in Molecular Biology*, 2019.

[17] Fitzgerald, R. E. et al. Maze patrolling by rats with and without food reward. *Animal Learning & Behavior*, 1985.

[18] Morris, R. Developments of a water-maze procedure for studying spatial learning. *Journal of Neuroscience Methods*, 1984.

[19] Kekesi, G. et al. Sex-specific behavioral alterations in an animal model of schizophrenia. *Behavioral Brain Research*, 2015.

[20] Monaco, J. D. et al. Attentive scanning behavior and hippocampal place fields. *Nature Neuroscience*, 2014.

[21] Kimchi, T., Terkel, J. Role of somatosensory stimuli in maze learning. *Behavioral Brain Research*, 2004.

[22] Petrovszki, Z. et al. Behavioral profiling of selectively bred rats. *Behavioral Brain Research*, 2013.

[23] van der Staay, F. J. et al. The cognitive hole-board task. *Neuroscience & Biobehavioral Reviews*, 2012.

[24] Wikenheiser, A. M., Redish, A. D. Hippocampal theta sequences and goal-directed behavior. *Nature Neuroscience*, 2015.

[25] Locklear, M. N., Kritzer, M. F. Spatial cognition in rats using the Barnes maze. *Hormones and Behavior*, 2014.

[26] OpenCV Contributors. *OpenCV Documentation*. https://docs.opencv.org, 2024. Utolsó látogatás dátuma: 2025.12.12.

[27] Python Software Foundation. *Python Documentation*. https://docs.python.org, 2024. Utolsó látogatás dátuma: 2025.12.12.

[28] NumPy Developers. *NumPy Documentation*. https://numpy.org/doc, 2024. Utolsó látogatás dátuma: 2025.12.12.

[29] Horváth, G. és mtsai. *Az AMBITUS rendszer ismertetése*. Szegedi Tudományegyetem, belső műszaki leírás, 2017.

Nyilatkozat

Alulírott Korfanti Dániel Mihály programtervező informatika szakos hallgató, kijelentem, hogy dolgozatomat a Szegedi Tudományegyetem Informatikai Intézet Képfeldolgozás és Számítógépes Grafika Tanszékén készítettem Programtervező Informatikus diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat használtam fel.

Tudomásul veszem, hogy szakdolgozatomat a Szegedi Tudományegyetem Diplomamunka Repozitóriumában tárolja.

2025. december 13. Korfanti Dániel .....

aláírás

Köszönetnyilvánítás

Szeretnék köszönetet mondani Dr. Kőrösi Gábornak a szakdolgozat témaválasztásáért, valamint az AMBITUS rendszer részletes ismertetéséért és folyamatos konzultációjáért. Továbbá köszönettel tartozom az Informatikai Intézetnek az infrastruktúra biztosításáért, az értékes konzultációkért és az OpenCV-vel való munkához szükséges támogatásért.