Budapesti Műszaki és Gazdaságtudományi Egyetem

Közlekedésmérnöki és Járműmérnöki Kar

Közlekedés- és Járműirányítási Tanszék

Tudományos Diákköri Konferencia 2022

Track-to-track architektúrájú objektumfúzió autonóm járművek környezetérzékelésének támogatására

Készítette:

Czibere Balázs, járműmérnöki szak (BSc)

Témavezető:

Lindenmaier László

Tartalomjegyzék

1. Bevezetés

1.1. A kutatás motivációja

Napjainkban mind a vezetéstámogató, mind pedig az autonóm járművek térhódítása zajlik a hétköznapi és a tudományos életben egyaránt [1]. Az önvezető autók társadalomra, környezetre és a gazdaságra gyakorolt hatásait folyamatosan fejlődő kutatási témák övezik [2][3]. A közlekedés- és járműautomatizálás céljai közül mégis kiemelendő a közlekedésbiztonság növelés. Ma már számos vezetéstámogató rendszert kötelezően előírnak személy- és haszongépjárművekre egyaránt. Ilyen például a sávtartó, tolatás segítés, vezető álmosságát vagy figyelemelterelését figyelő, melyeket mind az Európai Unió közlekedésbiztonságra vonatkozó General Safety Regulation (GSR) direktívái indítványoznak [4]. Ezen új, szükségszerű, aktív biztonsági funkciók az európai járművekbe való telepítésével, védelmet nyújthatunk a járművezetőknek, a gyalogosoknak és kerékpárosoknak, így mérsékelve az emberi hibákat, mely az európai utakon bekövetkező balesetek 90%-át teszi ki. Ezzel együtt a teljes közlekedés gazdaságosabbá és fenntarthatóbbá tehető, az automatizáltsági szint növekedésével. Egyes funkciók önmagukban is, de a hálózatban kapcsolt, önvezető járművek kiemelkedő tüzelőanyag fogyasztás csökkenést képesek elérni, ezzel akár 60%-kal is csökkentve az üvegházhatású és a környezetre káros egyéb anyagok légkörbe jutását. Másrészt pedig növelhető a gépjárművek kihasználtsága, ezzel a járművek számával tovább csökkentve a környezeti terhelést. Mindez jelentős pozitív hatást gyakorolna világ gazdaságára és a környezetre egyaránt, hiszen a fejlett országokban a GDP nagy hányadát a közlekedés érinti. További gazdasági előnyöket jelenthet a magas automatizáltság a haszonjárművek esetén, ugyanis a szállítmányozással és fuvarozással foglalkozó vállalatok jelentős költséget spórolhatnak akárcsak a részleges járműautomatizálással is. Mindezzel az általános energiafelhasználás is normalizálható.

A fejlett vezetéstámogató és önvezető rendszerekkel felszerelt járművek biztonságos üzemeltetéséhez egy megbízható, környezetérzékelést támogató algoritmus fejlesztése a kulcs. Az automatizáltsági szint növekedésével továbbá, a környezetérzékelés szerepe egyre fokozódik. Azonban, napjainkban egyetlen járműipari szenzor sem képes kielégíteni a magas automatizáltság által a környezetérzékelés felé támasztott fokozódó igényeket [5], [6]. Ezért, a megbízható környezetérzékeléshez elengedhetetlen, különböző szenzorokból származó adatok együttes felhasználása, fuzionálása. A kutatás motivációja egy olyan moduláris architektúrával bíró szenzorfúziós algoritmus fejlesztése, amely rugalmasan kezeli az adatok forrását, ezzel könnyen adaptálva különböző szenzorokkal rendelkező járművekre. A fejlesztés során fontos szempont volt az algoritmusok számításkapacitás igénye, figyelembe véve a valós-idejű alkalmazhatóságot, ezzel járulva hozzá a megbízható környezetérzékeléshez és a fejlett vezetéstámogató rendszerek terjedéséhez.

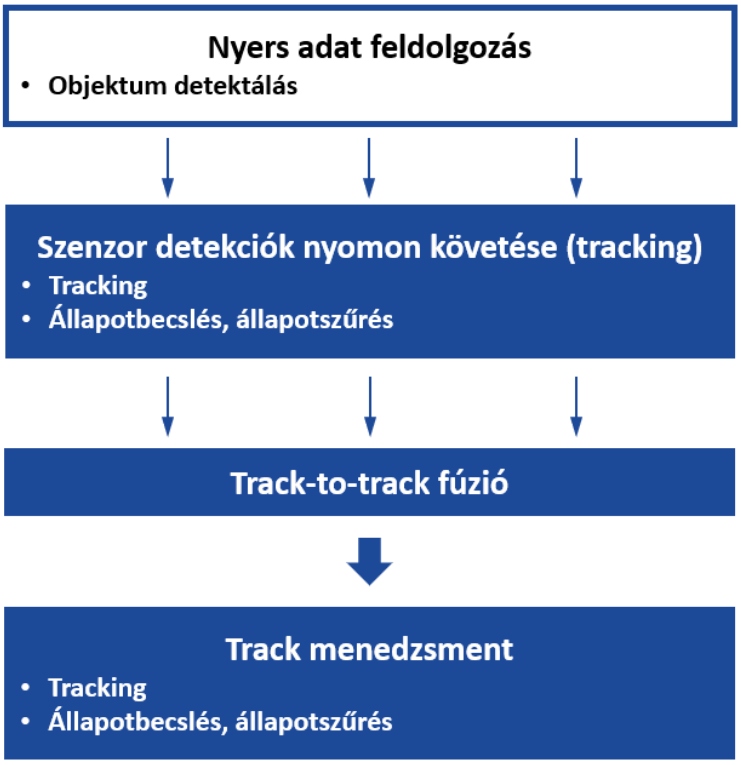
1.2. A kutatás fókusza

Szenzoradatok fúziója az utolsó rétege a környezetérzékelésnek, melynek fő feladata az egységes környezeti modell felépítése, mindezt oly módon, hogy az egyes szenzortípusokra jellemző kedvező tulajdonságokat előtérbe, míg a kedvezőtleneket a háttérbe helyezi. Ehhez szükségesek a trackek, melyek a szenzorfúziós logika bemeneti interfészén definiált, szenzor-

független bemenetek. A trackek lehetővé tehetik számunkra egy moduláris algoritmus fejlesztését, mint a track-to-track fúzió.

A szenzorfúziós és objektumkövető algoritmusok a valós-idejű alkalmazhatósága és szamításkapacitási igényük csökkentése fontos feladat a kutatás szempontjából. Fő célom az volt, hogy a fentebb részletezett motivációkat elérjem, a state-of-the-art megoldásokból kiindulva, egy olyan objektumkövető algoritmuson alapuló szenzorfúziós architektúra kifejlesztésével, melynek futásideje alacsonyabb, mint a klasszikus algoritmusoké, miközben azok kedvező tulajdonságait nem nyomja el.

Ezért a kutatás fókuszában a továbbfejlesztett objektumkövető algoritmus, a fúziós logika és annak bemeneti adatainak, a kifejlesztett tracking algoritmus segítségével való előkészítése, majd a logika kimeneteként előálló fuzionált objektumok posztprocesszálása áll. Az 1.1. ábra szemlélteti az elvárt rendszer architektúrát, teli téglalappal jelölve az imént felsorolt fókuszterületeket.



1.1. ÁBRA: *A kutatás fókusza, a környezetérzékelés folyamatábrája*

A szenzor adatok előkészítésénél arra a kérdésre kerestem a választ, hogyan lehet a klasszikus objektumkövető algoritmusok futás idejét jelentős mértékben redukálni, úgy, hogy hatásfokuk ne romoljon. Kutatásom hipotézise, hogy mindez elérhető egy objektum vagy detekció priorizálási struktúra alkalmazásával. Így az objektum-detekció összerendelések egy előre meghatározott logika alapján történnek és ezáltal jelentős számítás kapacitási igényt megspórolunk. Ezt a tracking algoritmust minden szenzoron külön-külön futtatom, az így keletkező kimeneteket pedig egy track-to-track fúziós logikával feldolgozom.

A track-to-track fúziós logikánál kérdés az, hogy hogyan lehet a különböző szenzorokból érkező, de ugyanahhoz az objektumhoz tartozó trackeket úgy összerendelni, hogy azt a kisebb számítási igényt, melyet a kifejlesztett tracking algoritmus alkalmazásával

nyertünk, itt ne veszítsük el. Ehhez több klasszikus track-to-track eljárásba belemerültem. Azt vizsgáltam, hogy hogyan lehet egy olyan threshold értéket kiszámítani minden egyes *track* párosításnál, hogy egyértelműen eldönthető legyen, hogy az adott két *track* valóban ugyanahhoz az egy objektumhoz tartozik-e. Ezen felül kérdés, hogy mi alapján tudjuk megmondani azt, hogy egy adott szenzor egy adott objektumot duplikál, vagy az a két track amit küld, valóban két különböző objektumhoz tartozik-e. Hipotézisem, hogy egy másik, az előzőnél szigorúbb threshold érték kiszámítható, beállítható specifikusan minden track párosításnál, melynek segítségével megválaszolhatjuk ezt a kérdést.

Azt feltételeztem, hogy így egy olyan robosztus környezetérzékelés valósítható meg, mely megközelíti az általánosan elfogadott, klasszikus objektumfúziós eljárások teljesítményét, miközben a számítás kapacitási igény jelentős mértékben redukálódik. Mindezt úgy, hogy a fúziós logika szenzor-független maradhat.

2. Irodalomkutatás összefoglalása

A szenzorfúziós szintek megismerése elengedhetetlen ahhoz, hogy megismerjük a szenzorfúziós algoritmusok előnyeit és hátrányait. Az irodalomkutatás során ezen szintek feltérképezése volt az egyik fő célom.

A fúzió különböző szinteken történik, melyek mindegyikének megvannak az előnyei, hátrányai. Különböző elképzelések vannak arra vonatkozóan, hogy hány szintet lehet meghatározni[7][8][9][10]. Az objektum fúzió nagy szerepet játszik az ADAS funkciók területén és mint kutatásom egyik fő eleme, egy symbol szintű fúziót valósít meg [11][12][13]. Elterjedté vált, hisz az autóiparban is használatos szenzorok magas szintű előfeldolgozott adatokat küldenek, melyek lehetnek objektum detekciók, vagy akár trackek is. Klaszterekbe rendezésére Adam Houenou kitalált egy iteratív módszert [11], melynek köszönhetően nem szükséges a Global Nearest Pattern Matching megközelítés alkalmazása, melyet Dimitri J. Papageorgiou is ajánl a Track-to-Track Association and Ambiguity Management in the Presence of Sensor Bias [12] című cikkében. Objektum detekciók fuzionálására Multi-Object Tracking (MOT) algoritmusokat használnak, mint például a Nearest Neighbor megközelítések [14]. Ezek explicit adatasszociációt valósítanak meg az objektum, és a hozzá legközelebb álló detekció közt. Irodalomkutatás során a Global Nearest Neighbor (GNN) és a Local Nearest Neighbor megközelítéssel foglalkoztam, melyek gyakorlati alkalmazására néhány irodalom részletes leírást nyújt [15][16], ahol ezen algoritmusok segítségével asszociálják az objektum detekciókat a nyomon követett trackekkel. A GNN pontosabb eredményt nyújt, mint az LNN, ezért elterjedtebb a használata is, bár futási ideje az objektumok, és a detekciók számának növekedésével rohamosan növekszik [17][18]. Az LNN közel sem ad ennyire pontos eredményeket viszont számításkapacitási igénye sok objektum és detekció kezelésekor is alacsony marad, mely egy kedvező tulajdonság. Kihasználva és megtartva ezt, egy olyan algoritmus fejlesztésével, mely pontosíthatja az LNN megközelítést, egy jó hatásfokú, alacsony futási idejű Multi-Object Trackinget valósíthatunk meg.

Léteznek olyan adatasszociációs módszerek, melyek nem egy az egybe kölcsönösen egyértelmű hozzárendelést valósítanak meg, hanem probabilisztikus alapúak. Ilyenek például a Probabilistic Data Association (PDA) [19], Joint Probabilistic Data Association (JPDA)[20] algoritmusok, melyek egy jó becslést nyújtanak számunkra az ismert számú objektumok állapotáról. Valós alkalmazáskor az objektumszám ismeretlen és időben változó, ezért az objektumszám becslésére is szükség van. Erre kínál megoldást az Integrated Probabilistic Data

Association (IPDA) [21] és a Joint Integrated Probabilistic Data Association (JIPDA) [22] filterek, melyek az előző megközelítésektől annyiban különböznek, hogy képesek becslést adni az objektumok létezési valószínűségeire, alapként szolgálva az objektum menedzsmentnek. Bár ezen valószínűségi alapú megoldások kombinatorikus problémák, ezért a számításkapacitásuk jóval magasabb, mint a Nearest Neighbor alkalmazások esetén, hisz a futási idő az objektum és detekció szám növekedésével exponenciálisan növekszik.

Ezért fókuszál a kutatás arra, hogy hogyan lehet egy már meglévő, alacsony számításkapacitású, track-to-track algoritmust úgy továbbfejleszteni, hogy teljesítménye megközelíthesse az eleve pontosabb, alacsonyabb absztrakciós szintű fúziót, a szenzor függetlenség megtartása mellett. Ehhez a szenzorfúziós algoritmusokat [10][23] kezdtem el tanulmányozni.

3. Elméleti háttér

3.1. Objektumkövető algoritmusok

Objektum követő algoritmusok alapvetően arra a célra lettek fejlesztve, hogy a jármű környezetében található objektumokon alapuló detekciókat egy szenzor lekövetni tudja a következő problémákkal megbirkózva:

- Asszociációs probléma: adott detekciók valós objektumok által lettek-e legenerálva és melyek ezek, az ismeretlen,
- Objektum állapot meghatározásának problémája: a szenzor mérések zajjal terheltek és nem teljesek (pl.: kamera nem tud távolságot mérni direkt módon),
- Objektumok számossága időben változó és ismeretlen.

A Single-Object és a Multi-Object Tracking algoritmusokat különböztethetünk meg aszerint, hogy mi csupán egyetlen ismert objektumot vagy változó és ismeretlen számú objektumokat szeretnénk lekövetni.

Az általános Multi-Object Tracking algoritmus egy rekurzív architektúra, melyet az alábbi ábra is szemléltet:

3.1. ÁBRA: Multi-Object Tracking architektúra

A következő három főbb modul kezeli a fentebb felsorolt problémákat: Adatasszociáció, állapotmeghatározás, és objektum menedzsment. Single-Object Tracking algoritmusok esetén ugyanúgy szükséges az adatasszociáció és az állapotbecslés mint Multi-Object Tracking-nél de az objektum menedzsment elmarad hisz ismert, egyetlen objektumot követünk. Az objektumkövető algoritmusok a 3.1-es fejezetben esik szó, ahol részletezésre kerül az adatasszociáció és az állapotbecslés, a Kálmán-szűrő segítségével. A 3.2-es fejezetben kerül bemutatásra az objektum fúzió. 3.3-as fejezet pedig az objektum menedzsmentet írja le.

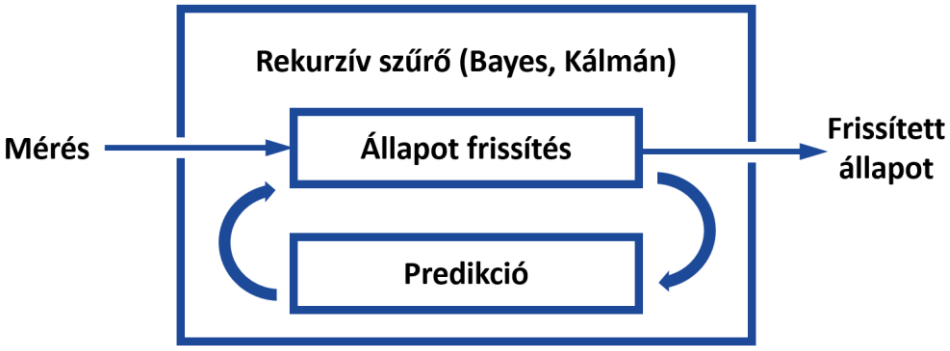
Objektum-detekció folytonosságát ez a rekurzív architektúra teszi lehetővé, az adatasszociáció segítségével melynek részletezésére később kerül sor. Ennek köszönhetően létrejöhetnek az objektumok új absztrakciója, az úgynevezett track-ek, melyekben nem csak objektumdetektálást végzünk, hanem azok mozgásának nyomon követését is. Tehát egy új objektumot ellátunk egy egyedi azonosítóval, nevezetesen egy ID-val, mely egy olyan ismertető szám, melynek folytonossága jelzi számunkra azt, hogy a nyomon követés sikeres, tehát frame-ről frame-re látható, hogy az adott objektum már szerepelt korábban a detektált objektumok között.

Multi-Object Tracking algoritmusok általánosan heterogén adatok fúziójára használatos, de mivel alapvetően arra lettek fejlesztve, hogy egy szenzor detekcióit kövesse, több, különböző szenzor adatot kell biztosítani egy megfelelő sorrendben annak érdekében, hogy fuzionálni tudjuk őket egy közös objektum listává.

3.1.1. Állapot becslés

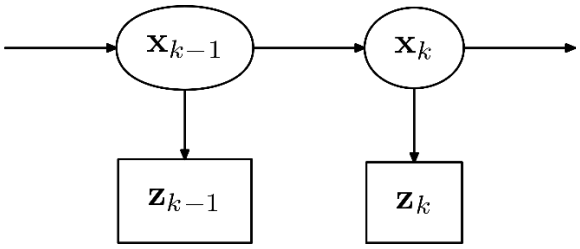
A szenzortechnológia folyamatos fejlődése mellett a detektáló algoritmusok is egyre jobbá válnak. Azonban a detektált objektumok állapotainak mérése zajjal terheltek, ezért létfontosságú ezen objektumok állapotainak szűrése a szenzorfúzió során.

Multi-Object-Tracking esetén az állapot becslés a rekurzív Bayes-féle rekurzív becslés alapján történt, a becslést (prediction) és az állapot frissítést (state update) figyelembe véve.



3.2. ÁBRA: Rekurzív szűrés

A rekurzív szűrők alapját a Bayes-szűrés adja. Az itt alkalmazott modell a Markov-lánc, melynek lényege, hogy az aktuális állapot kizárólag az előző állapottól függ. Ezen túlmenően, a modellt kiegészítjük azzal az információval, hogy léteznek nem olyan állapotváltozók, melyek nem mérhetők, ezek a rejtett állapotok. Így kaphatjuk a rejtett Markov-láncot, melyet a következő ábra is szemléltet:



3.3. ÁBRA: Rejtett Markov modell

Az ábrán szereplő y_T a T időponthoz tartozó mérést, x_T pedig a becsült állapotot hivatott jelképezni.

Az általános Bayes-szűrő használható bármilyen valószínűségi eloszlásra, de magas a számítási kapacitási igénye. Erre megoldást kínálnak egyszerűsített algoritmusok például a Kálmán filter, felhasználható valós-idejű alkalmazásokban, feltéve, hogy a szenzor mérések zaja Gauss féle eloszlásgörbe, melynek várható értéke nulla és a modell lineáris.

Kérdés tehát az, hogy hogyan lehetséges meghatározni az aktuális állapot, $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ feltételes eloszlását, az aktuális időpontig rendelkezésre álló összes mérés feltételi esemény függvényében. Ezt a 3.1. egyenlet írja le.

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) = \frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})}{p(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})} \propto p(\boldsymbol{z}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}) \qquad \textbf{(3.1.)}$$

Ahol a pediktált állapotot a Chapman-Kolmogorov egyenlet írja le.

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1})d\boldsymbol{x}_{k-1} \qquad \textbf{(3.2.)}$$

Mivel a tracking algoritmus során az előző objektum listát és az aktuális időpillanathoz tartozó méréseket (objektumhipotéziseket) hasonlítjuk össze, fontos, hogy az előző objektum listát milyen formában reprezentáljuk. A legjobb asszociációs eredményt az biztosítja, ha az előző objektum listát, a lehető legpontosabban transzformáljuk arra az időpillanatra, amelyről az objektumhipotézisek információt adnak. A jármű állapotának predikcióját egy diszkrét, lineáris állapot dinamikai modell segítségével végezzük:

$$\boldsymbol{x}_t = \boldsymbol{F}\boldsymbol{x}_{t-1} + \boldsymbol{B}\boldsymbol{u}_t \qquad \textbf{(3.3.)}$$

Ahol $\boldsymbol{x}_t$ a t. időpillanatban vett állapot, $\boldsymbol{F}$ az átmeneti mátrix, $\boldsymbol{B}$ az irányító mátrix, míg $\boldsymbol{u}_t$ a t. időpillanatban értelmezett irányítójelet jelöli. Jelen alkalmazásban az irányítójel nem definiálható, ezért csak az átmenet mátrix leírásával prediktáljuk az objektum állapotát. Így az objektum állapotvektora:

$$\boldsymbol{x}_t = [x_p \quad v_x \quad y_p \quad v_y]^T \qquad \textbf{(3.4.)}$$

Ahol x_p és y_p az objektum longitudinális és laterális pozícióját, v_x és v_y a sebességeket jelöli. Természetesen az objektumnak lehetnek egyéb állapotai is, ám ezeken a modell alapú predikció nem módosít. Amennyiben tehát az objektum állapotát így definiáljuk, két modellt különböztetünk meg, amelyekkel az átmenet mátrixot leírhatjuk.

A CV (Constant Velocity) modell alkalmazásánál, azt feltételezzük, hogy két mintavételezési idő közt a sebesség konstans. Így az egyenletes mozgást leíró dinamikai egyenletekkel meghatározható a pozíció változása. Az átmeneti mátrix tehát:

$$\boldsymbol{F}_{CV} = \begin{bmatrix} 1 & 0 & T & 0 & 0 & 0 \\ 0 & 1 & 0 & T & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.5.)}$$

Ahol T a diszkrét modell mintavételezési idejét jelöli.

A CA (Constant Acceleration) modellben, azzal a feltételezéssel élünk, hogy a gyorsulás állandó a mintavételezési idők között. A pozícióváltozást így a négyzetes úttörvény, míg a sebességet az állandó gyorsulásból származó sebességváltozás segítségével számíthatjuk. Ezt a következő összefüggés írja le:

$$\boldsymbol{F}_{CA} = \begin{bmatrix} 1 & 0 & T & 0 & T^2/2 & 0 \\ 0 & 1 & 0 & T & 0 & T^2/2 \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.6.)}$$

A Kálmán-szűrő egy olyan egyszerűsített rekurzív, Bayes-szűrő, amelyben, mind a mozgás, mind pedig a mérési modell lineáris. Továbbá, a folyamat és a mérési zaj modelljére egyéb kikötések is vonatkoznak, amelyeket a 3.9. és 3.10. összefüggések írnak le. A folyamatmodellt a 3.7., a mérési modellt pedig a 3.8. összefüggés szerint:

$$x_k = \boldsymbol{F}_k x_{k-1} + v_k \quad \textbf{(3.7.)}$$

$$y_k = \boldsymbol{H}_k x_k + \eta_k \quad \textbf{(3.8.)}$$

Ahol $\boldsymbol{F}_k$ az átmeneti mátrixot, $\boldsymbol{H}_k$ pedig a mérési modell mátrixot jelöli. Továbbá v_k és η_k a folyamat és a mérési zaj vektorok, melyek jellemzői:

$$v_k \sim N(\cdot, \mathbf{0}, \boldsymbol{Q}_k) \quad \textbf{(3.9.)}$$

$$\eta_k \sim N(\cdot, \mathbf{0}, \boldsymbol{R}_k) \quad \textbf{(3.10.)}$$

Ezen összefüggések a folyamat, illetve a mérési zaj azon tulajdonságát írja le, mely szerint ezek zérus várható értékű, Gauss-eloszlással bíró valószínűségi vektorváltozók, melyekhez tartozó kovariancia mátrixok $\boldsymbol{Q}_k$ és $\boldsymbol{R}_k$.

A predikciós lépés, a 3.7. összefüggésből kiindulva, az állapot pontbecslésének és az állapotbizonytalanságot leíró kovariancia mátrixnak a predikciójából áll, melyet a következő összefüggések írnak le:

$$\hat{x}_{k|k-1} = \boldsymbol{F}_k \hat{x}_{k-1|k-1} \quad \textbf{(3.11.)}$$

$$\widehat{\boldsymbol{P}}_{k|k-1} = \boldsymbol{F}_k \widehat{\boldsymbol{P}}_{k-1|k-1} \boldsymbol{F}_k^T + \boldsymbol{Q}_k \quad \textbf{(3.12.)}$$

Az állapotfrissítéshez előbb alkalmazzuk a prediktált állapoton a mérési modellt:

$$\hat{z}_{k|k-1} = \boldsymbol{H}_k \hat{x}_{k|k-1} \qquad \textbf{(3.13.)}$$

Ez alapján megkaphatjuk a modell alapján előre becsült állapot azon állapotváltozóit, melyek a szenzor által is közvetlen módon mérhetőek. Ez a lépés teszi lehetővé, azt, hogy a modell és a valódi mérés összehasonlíthatóvá váljon. Az állapotfrissítést a 3.14. írja le.

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \boldsymbol{K}_k(z_k - \hat{z}_{k|k-1}) \qquad \textbf{(3.14.)}$$

Ahol $z_k - \hat{z}_{k|k-1}$ fejezi ki az úgynevezett innovációt, vagyis az aktuális mérés és a modell közti eltérést, $\boldsymbol{K}_k$ pedig a Kálmán-nyereség, amely az inováció szorzótényezője. Kálmán-nyereséget a következőképp adjuk meg:

$$\boldsymbol{K}_k = \widehat{\boldsymbol{P}}_{k|k-1}\boldsymbol{H}_k{}^T(\boldsymbol{H}_k\widehat{\boldsymbol{P}}_{k|k-1}\boldsymbol{H}_k{}^T + \boldsymbol{R}_k)^{-1} \qquad \textbf{(3.15.)}$$

Továbbá az állapotbecslés bizonytalanságát leíró kovariancia mátrix frissítésének elve:

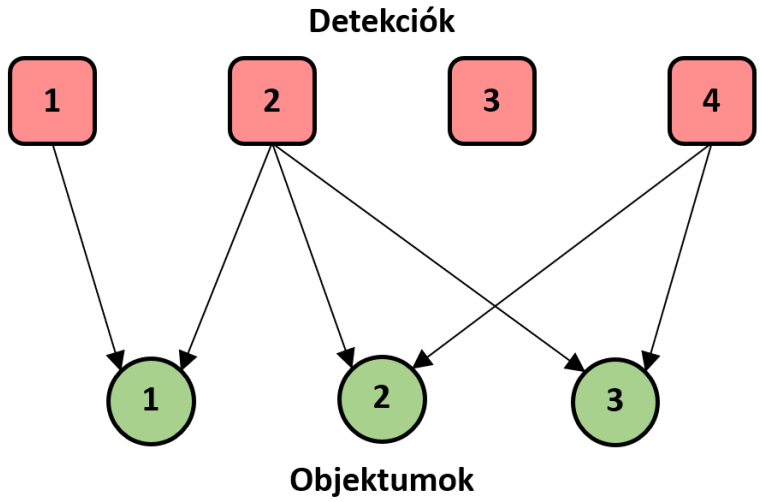
$$\widehat{\boldsymbol{P}}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\widehat{\boldsymbol{P}}_{k|k-1} \qquad \textbf{(3.16.)}$$

Azonban a frissített kovariancia mátrix számításának megadható egy numerikusan stabilabb változata is, mely számítógépes implementáció során ajánlott:

$$\widehat{\boldsymbol{P}}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\widehat{\boldsymbol{P}}_{k|k-1}(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)^T + \boldsymbol{K}_k\boldsymbol{R}_k\boldsymbol{K}_k{}^T \qquad \textbf{(3.17.)}$$

3.1.2. Adatasszociáció

Az adatasszociáció, mely a fúzió fő magjának tekinthető. Feladata az adatok valamilyen szabály szerinti egymáshoz rendelése. Az adatasszociáció jelenti a legfőbb különbséget a detekció szintű és a track-to-track fúzió közt. A detekció szintű algoritmus esetén a fúzió bemenetén megjelenő detektált objektumok és a végső objektumlista közt zajlik a hozzárendelés. Ezt mutatja a 3.4. ábra is.



3.4. ÁBRA: Detekció szintű adatasszociáció

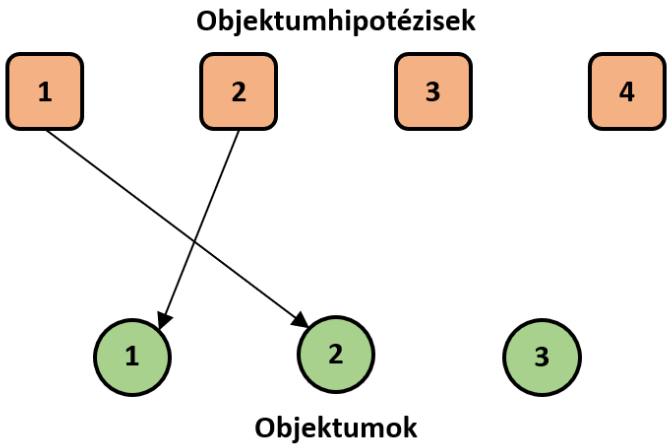
Mivel a nyers detekciókat rendeljük a végső objektumlista elemeihez, így a fals pozitívok szűrése és a különböző forrásokból érkező adatok fúziója is itt történik. Egy objektumhoz több

detekció is tartozhat és az is fennállhat, hogy egy detekció egyik objektumhoz sem rendelhető hozzá, mint ahogy azt az ábra is szemlélteti. Emiatt nem kölcsönösen egyértelmű a hozzárendelés. Így tehát az sem egyértelmű a bemenet alapján, hogy hány kimeneti objektum szerepel a környezeti modellben (cardinality). Ebből fakadóan, az új objektumok megjelenése és eltűnése, különböző eloszlás függvényekkel modellezhető, mely az algoritmus bonyolultságát és számításkapacitás igényét növeli.

Az adatasszociáció feladata az, hogy bizonyos objektumokat vagy detekciókat egymáshoz rendelje valamilyen szabály szerint. Első lépésként fel kell építenünk a távolságmátrixot a kiszámított távolságokkal. Két főbb adatasszociációs eljárás létezik:

- Bináris adatasszociáció
- Valószínűség alapú asszociáció

A korábban részletezett állapotbecslő és szűrő algoritmusok kizárólag egy objektumra való működést biztosítanak. A Kálmán-szűrő esetében például, szükséges megadnunk azt az információt, hogy a beérkező mérések közül, melyik mérés, melyik objektum állapotát frissítse. A 3.5. ábra szemlélteti a kölcsönösen egyértelmű hozzárendelést.



3.5. ÁBRA: Kölcsönösen egyértelmű hozzárendelés

Látható, hogy minden objektumhipotézishez legfeljebb egy objektumot rendelünk hozzá. Az is látszik, hogy mindkét csoportnál szerepelhet olyan elem, melyek nem kerülnek hozzárendelésre.

A bináris adatasszociáció egymást kizáró eseményeken alapuló feladat a detekciók $\{z_j\}_{j=1}^{m}$ és az előzú objektumok $\{x_i\}_{i=1}^{n}$, ahol m és n a detekciók és objektumok számát jelölik. Ez azt mutatja, hogy egy detekció hozzárendelhető legfeljebb egy objektumhoz és egy objektum legfeljebb egy detekcióhoz. Az eredménye az adatasszociációnak egy A bináris asszociációs mátrix, a következő definíció alapján:

$$a_{ij} \in A = \begin{cases} 1, & ha\ z_j\ hozzá\ van\ rendelve\ x_i - hez \\ 0, & egyébként \end{cases}$$

A kölcsönösen egyértelmű hozzárendelést két elv alapján lehet végrehajtani, az úgynevezett LNN (Local Nearest Neighbor) és a GNN (Global Nearest Neighbor) szerint.

A hozzárendelési probléma megoldható egy mohó módszerrel, mely a következő: Az LNN (Local Nearest Neighbour) módszer minden egyes objektumhipotézishez a legközelebb eső objektumot rendeli. Tehát a távolságminimalizálás elve lokális.

$$a_{ij} \in \boldsymbol{A}, a_{ij} = 1 \qquad \textbf{(3.18.)}$$

$$j = \arg\min(\boldsymbol{D}_i) \qquad \textbf{(3.19.)}$$

Ahol $\boldsymbol{A}$ az előállítandó bináris asszociációs mátrix, a_{ij}, $\boldsymbol{A}$ asszociációs mátrix i. sorának és j. oszlopának eleme, míg $\boldsymbol{D}_i$, az előzetesen felépített távolság mátrix i. sora. Ez a módszer egy szuboptimális megoldás, ugyanis csak lokális minimumot érünk el a távolságok összegének tekintetében. Továbbá csak, akkor tekintjük elfogadottnak a hozzárendelést, ha az objektumhipotézis, a prediktált objektum előre definiált kis „környezetén" belül van.

A GNN (Global Nearest Neighbour) egy olyan módszer, mely egy globálisan optimális megoldást keres. A cél ugyanis, hogy a teljes párosító algoritmusra nézve minimalizáljuk a távolságösszeget, nem pedig objektumhipotézisenként tegyük ezt meg.

$$\boldsymbol{A} = \arg\min(\boldsymbol{A} \cdot \boldsymbol{D}) \qquad \textbf{(3.20.)}$$

Kuhn-Munkres algoritmus megoldja a GNN problémát a költségfüggvény minimalizálásával:

$$c(A) = \sum_{i=0}^{n} \sum_{j=0}^{m} a_{ij} d_{ij} \qquad \textbf{(3.21.)}$$

Érdemes megjegyezni, hogy ebben a költségfüggvényben az objektum és detekció index i, j 0-ról indul azt az eseményt jelölve, amikor az objektum x_i-nek vagy a detekció z_j-nek nincs párja, tehát nincsenek benne az előre meghatározott d_0 küszöbtávolságban.

Azonban a módszer alkalmazása nem minden esetben ad jobb eredményt, mint az LNN típusú megoldások, bonyolultságából adódóan viszont, sokkal nagyobb számításkapacitás igénnyel rendelkezik, mint az egyszerűbb LNN típusú megoldás.

Általában az asszociáció a D távolságmátrixon alapul, mely tartalmazza a d_{ij} távolságokat az i-dik objektum és a j-dik detekció közt. Ezen távolságok számítására két fő lehetőség áll rendelkezésre, Euklideszi- vagy Mahalanobis-távolság.

A jól ismert Euklideszi-távolság kizárólag az összehasonlítandó objektumok relatív pozícióját veszi figyelembe. Azonban az állapotbecslés bizonytalansága nem kerül beszámításra, ami hátrány.

$$d_E(o_i, o_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \qquad \textbf{(3.22.)}$$

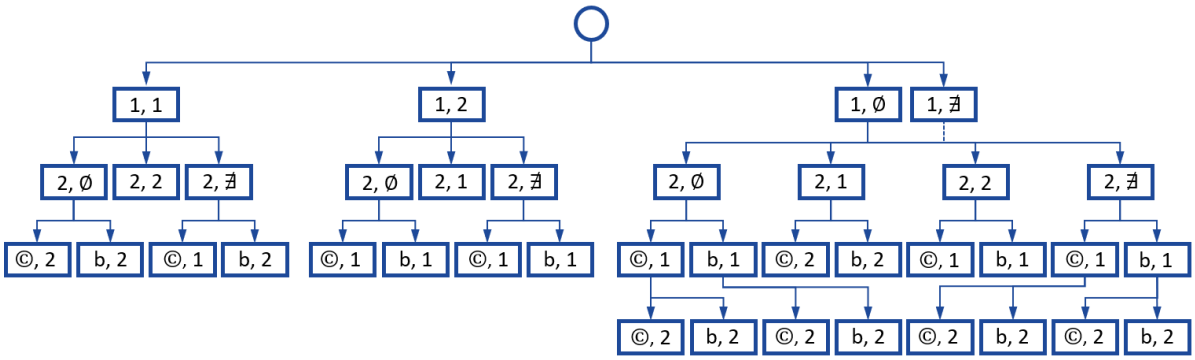
Ahol x_i az o_i objektum longitudinális, míg y_i az objektum laterális pozíciója. Amennyiben minden lehetséges objektum párosra kiszámítjuk, felépíthetünk egy távolság mátrixot, melynek minden $\{i, j\}$ eleme az o_i és az o_j objektumok közötti távolságot reprezentálja.

A Mahalanobis-távolság, egy valószínűségi változókra értelmezett, statisztikai távolság. A 3.23. összefüggés mutatja számításának menetét.

$$d_M(o_i, o_j) = \sqrt{(\vec{x}_i - \vec{x}_j)\boldsymbol{S}^{-1}(\vec{x}_i - \vec{x}_j)} \qquad \textbf{(3.23.)}$$

Ahol $\vec{x}_i$ az o_i objektum állapotvektorát fejezi ki, míg $\boldsymbol{S}$ az állapotvektorok, mint valószínűségi vektorváltozók kovariancia mátrixát fejezi ki.

A nem kölcsönösen egyértelmű hozzárendelés esetén, valószínűség alapú hozzárendelést valósítunk meg. Összevetve a bináris adatasszociációval, a valószínűség alapú megközelítés létrehoz egy P valószínűségi asszociációs mátrixot, ahol $p_{ij} \in P$ jelöli annak az eseménynek a valószínűségét, hogy a z_j detekció a x_i előző objektum által lett létrehozva. Ezek a valószínűségi értékek a különböző asszociációs esetek leklihoodjai alapján számolandók, figyelembe véve minden lehetséges párosítási esetet a detekciók $\{z_j\}_{j=1}^{m}$ és az objektumok $\{x_i\}_{i=1}^{n}$ közt, mely a következő ábrán is látható, egy hipotézisfába rendezve:



3.6. ÁBRA: Valószínűség alapú adatasszociáció hipotézisfája

A csúcsok a fában reprezentálnak párosításokat objektum és egy detekció közt, ahol $\emptyset, \nexists, ©, b$ speciális elemek reprezentálják a nem detektált és false (nem létező) objektumokat, a false (clutter) és az új (born) detekciókat. Az ábrán egy asszociációs eset egy csomópontokból álló lánc a fa tetejétől az aljáig.

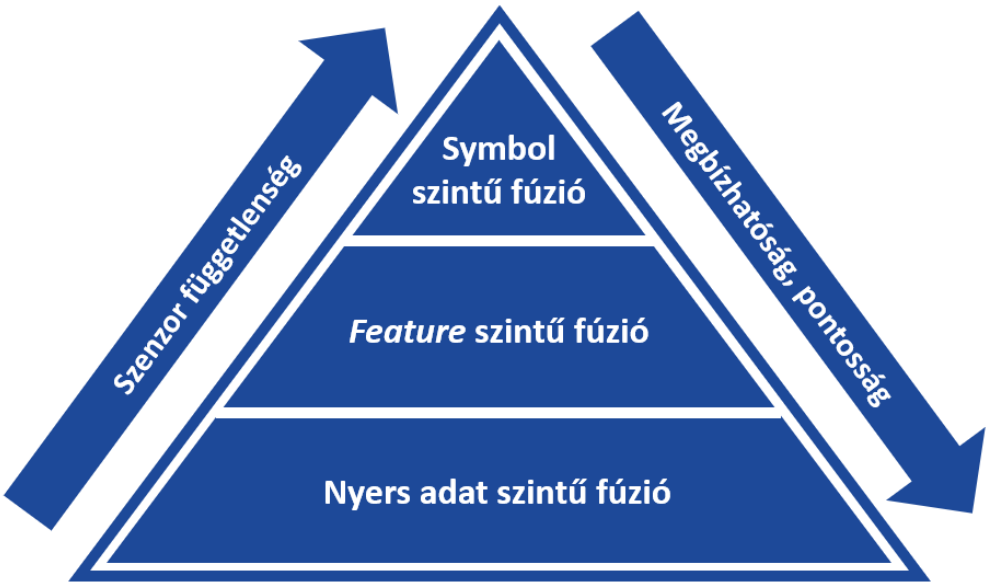
A valószínűségi megközelítés általánosan egy robusztusabb asszociáció, mint a bináris. A leggyakrabban alkalmazott ilyen algoritmus, az úgynevezett JIPDA (Joint Integrated Probability Data Association). Fontos megemlíteni még, hogy az asszociációs hipotézis fa mérete exponenciálisan növekszik a detekciók és az objektumok számától függően, emiatt a valószínűség alapú megközelítés sokkal komplexebb és a számítási kapacitási igénye is nagyobb, emiatt a kutatás során a bináris adatasszociáció képezte a fő fókuszt.

3.2. Objektum Fúzió

A szenzorfúzió motivációja, hogy egy általános, közös környezeti modellt építsen fel, valamilyen stratégia mentén, melyben minden szenzor jelen van. Továbbá ez a környezeti modell legyen minél robusztusabb, tehát minél kevesebb fals érzékelés realizálódjon, melyek lehetnek fals negatívok vagy éppen fals pozitívok. Ezen kívül fontos feladat még, hogy a felépített modell kellően pontos legyen, azáltal, hogy az egyes szenzortípusok erősségeit előtérbe hozva és a negatív tulajdonságokat kiküszöbölve fuzionáljuk a szenzor adatokat. És végül, de nem utolsó sorban, szükséges a zajjal terhelt szenzormérések zajszűrése is, mely szintén szenzortípus függő.

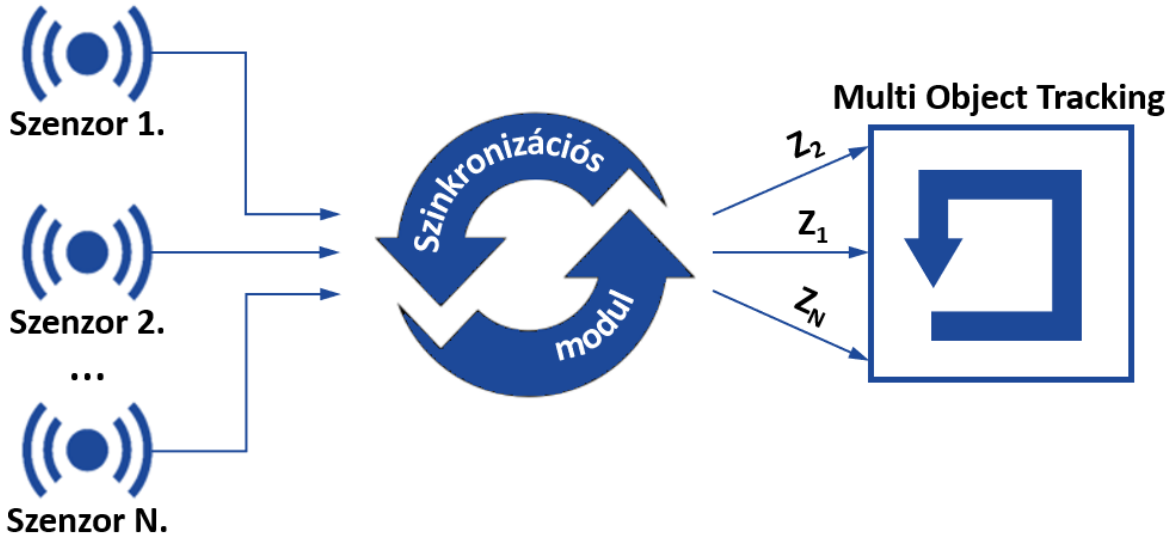
Az objektum fúzió a környezetérzékelésben alkalmazott szenzor adatfúziónak egy olyan speciális, symbol szintű fúziója, amelyben objektumnak feltételezett, numerikus

paraméterekkel jellemzett absztrakt méréseket fuzionálunk, így készítve egy egységes környezeti modellt. Ezen, a környezetből kinyert, symbol-ok mozgásképes objektumokat detektálnak, vagy akár track-elve elküldik. Két fő architektúrát, a detekció szintű és a track-to-track fúziót különböztethetünk meg.



3.7. ÁBRA: Szenzorfúziós szintek

Kameránál két, x és y pozíciót küld, míg a radar az x és y sebességeket is, melyek az állapotvektor részét képezik, emiatt nem hasonlíthatóak össze. Így a szenzoradatok összehasonlításának problémáját, detekció szintű architektúrában egy rekurzív modullal oldjuk meg. Ezt a 3.8. ábra is szemlélteti.



3.8. ÁBRA: Detekció szintű architektúra

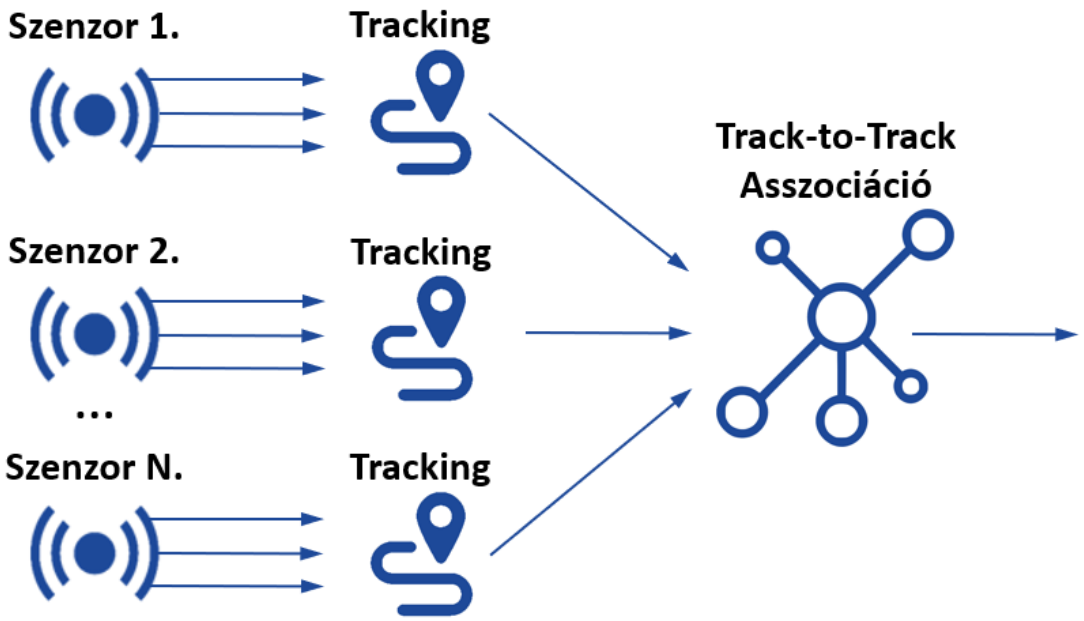
A nyers szenzor adatokat beküldjük ebbe a szinkronizációs modulba, mely időzítve továbbítja az adatokat, egymás után egy MOT algoritmusnak, egy meghatározott logika alapján. Az algoritmus minden egyes szenzor adatfeldolgozással frissíti a közös objektumlistát. Miután lefutott egy teljes ciklus, akkor a végén már az összes szenzor adata bele van fuzionálva az

objektumlistába. Ez az az aspektus, ami miatt fúzió ez az architektúra. A végén egy trackelt objektumlistát kapunk, melyek mindegyike 3.4. egyenletben leírt $\boldsymbol{x}$ állapotvektorral rendelkezik. Alacsony szintű fúzió relevanciáját a nyers szenzoradatok előfeldolgozatlansága adja, mivel minden egyes előfeldolgozás információ vesztést eredményez, mely pontatlansághoz vezethet.

A hagyományos, analitikus szenzorfúziós algoritmusok többsége esetén, a bemenetek fúziója, detekció szinten történik. Ez viszont olyan nehézségekhez vezet, mint például az objektumszám bizonytalansága (*cardinality problem*). Így az egyszerűbb és moduláris jellege miatt válhatott a track-to-track fúzió egyre elterjedtebbé.

Nyers adatszintű fúzió egy szenzorklaszterre alkalmas csupán, nem moduláris, ezért adódik a motiváció, ahhoz, hogy magasabb szintű, egyre modulárisabb algoritmusokat alkossunk, mellyel a környezet egy magasabb absztrakcióval való reprezentálását értjük. Egy ilyen magasabb absztrakció egyik jellemzője, ha már a nyers szenzor adatokat előfeldolgozás következtében objektumokat nyerünk ki a környezetből. Ezzel a lépéssel a szenzoradatok összehasonlíthatóvá válhatnak, melynek köszönhetően az algoritmusunk flexibilissé válhat. A track-to-track fúzió egy jó példa erre, hisz ott minden bemenetet trackelünk, így minden egyes track-nek ugyanaz az $\boldsymbol{x}$ állapottere van, melyet 3.4. egyenlet is mutat.

Track-to-track fúzió esetében, a már trackelt bementi objektumok között zajlik az asszociáció. Majd ezt követően a fuzionált adatokból kialakult, úgynevezett objektumhipotézisek és a végső objektum lista között valósul meg egy kölcsönösen egyértelmű hozzárendelés, ez a fuzionált adatokon végzett tracking, melyet a 3.8. ábra is mutat.



3.9. ÁBRA: *Track-to-Track szenzorfúzió*

Azon szenzoroknál, melyek nem adnak közvetlen információt bizonyos állapotokról, például kamera esetében a sebességek, ott ezen hiányzó adatok becslés eredményeként adódnak és ennek következtében az értékek a kovariancia mátrixban magasabbak lesznek, mint azon szenzorok esetén, melyek viszont szolgáltatnak közvetlen információt. Emiatt is küldhettem be

minden szenzor adatot egy közös halmazba. Ez az architektúra, a detekció szintűnél egyszerűbb. Nem igényel adatszinkronizációt, kisebb a számítási igénye és minden track egy állapottéren van. Viszont semmi sem biztosítja azt, hogy a fúzió végén track-eket kapunk, ezért szükséges egy utolsó tracking-et a végére kapcsolnunk, mint ahogy az a 3.8. ábrán is látszik. Ezzel a lépéssel biztosíthatjuk a track ID folytonosságot.

Összegezve tehát a kutatás egyik fő célja volt, hogy kihasználjuk a magasabb szintű fúzió adottságait, hogy egy flexibilis algoritmust kapjunk, emellett próbáltunk mindent megtenni annak érdekében, hogy a velejáró hátrányok kisebbek legyenek, miközben megközelítjük az alacsony szintű szenzorfúziónak a működését és hatékonyságát, mellyel megvalósítjuk az objektum fúziót.

3.3. Objektum menedzsment

Az objektum menedzsment feladatai Multi-Object-Tracking esetén:

- új, a látómezőbe belépő objektumok inicializálása (object birth)
- látómezőből kilépő objektumok törlése az objektum listáról (object death)
- false detekciók kiszűrése

Két főbb objektum menedzsment megközelítés létezik. Egyik a history-based, másik pedig a valószínűség alapú (probabilistic). A history-based megközelítés általában bináris adatasszociáció esetén, míg a valószínűség alapú objektum menedzsment a valószínűség alapú adatasszociációnál használatos. Mindkét alkalmazásnál, ha egy detekció nem tartozik egyik objektumhoz sem (valószínűség alapúnál, ha a valószínűségi értékek kisebbek, mint a küszöbérték), akkor a detekció inicializál egy új objektumot. A kutatás során a hisory-based modellt alkalmaztam.

Az adatasszociáció során előállított bináris mátrix segítségével megállapítható egy objektum születése vagy épp halála. Ez alapján o_i detekció megjelenő objektum, ha nem rendelhető egyetlen másik objektumhoz hozzá. Tehát:

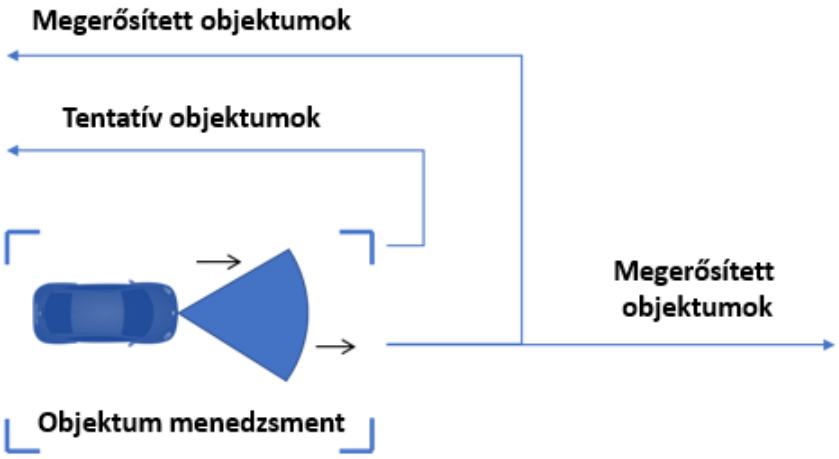
$$\sum_{j=1}^{m} a_{ij} = 0 \qquad \textbf{(3.24.)}$$

Abban az esetben, ha o_j eltűnő objektum, akkor:

$$\sum_{i=1}^{n} a_{ij} = 0 \qquad \textbf{(3.25.)}$$

Ahol a_{ij} az asszociációs mátrix i. sorának és j. oszlopának eleme, míg n a mátrix sorainak, m pedig az oszlopainak száma.

Annak érdekében, hogy elkerüljük a fals-pozitív objektumok létrejöttét, meg kell különböztetnünk két objektumtípust, a megerősítettet (confirmed) és a nem megerősítettet (tentative), mint ahogy azt a következő ábra is mutatja.



3.10. ÁBRA: Multi-Object Tracking objektum menedzsment

Minden új objektum, melyhez nem tartozik egyik detekció sem, az inicializálva van, mint tentative.

A history-based objektum menedzsment esetén, egy tentativ objektumot akkor és csak akkor tekintünk confirmednek, ha *N* darab ciklusból legalább *M*-szer detektált, ahol *N* és *M* felhasználó által előre beállított értékek. Ez a módszer elősegíti, hogy kiszűrjünk néhány fals objektumot, melyek csak néhány ciklus erejéig léteznek. Minél magasabb *N* annál hatékonyabban szűrjük a fals-pozitívokat, de nem lehet nagyon magas értéke sem, hisz azzal késleltetjük az objektumok megerősítését. Ha *M*-et a maximumra állítjuk ($N \cdot n_{szenzorok}$), akkor a szenzoroknál feltételezzük, hogy a detektálási valószínűségük 100%. Minél magasabb *M* értéke, annál jobban szűrjük a fals-pozitívokat, de fennállhat annak a veszélye, hogy nem erősítünk meg egyes objektumokat, amik valósak. Objektum törlése az objektumlistáról hasonló módon történik, mint az objektum megerősítés. Ha egy objektumot nem detektálunk *P*-szer *R* darab ciklusból, akkor az az objektum törlésre kerül.

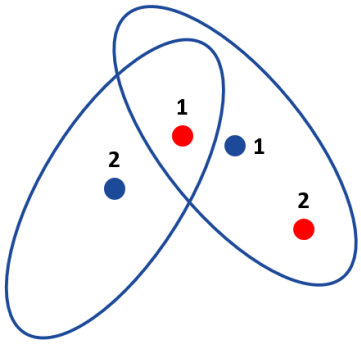
Valószínűség alapú objektum menedzsmentnél, annak a valószínűsége, hogy x_i létezik (probability of existence vagy PoE), rekurzív módon számítandók. Ha egy objektum PoE-je egy adott időpillanatban meghalad egy adott küszöbértéket, akkor az objektum megerősített, feltételezve, hogy az egy valós objektum. Ha egy objektum létezési valószínűsége kisebb, mint a törlési küszöbérték, akkor eltávolításra kerül az objektumlistáról.

4. Javasolt módszertan

Fentebb tárgyalt módszerek (Global Nearest Neighbor, Probabilistic Data Association) nagy hátránya, hogy alkalmazásuk nagy számításkapacitási igénnyel jár. Célunk egy olyan architektúra fejlesztése melynek futásideje alacsonyabb, mint a klasszikus algoritmusoké, miközben azok kedvező tulajdonságait megtartjuk.

Az architektúra alapját a Local Nearest Neighbor megközelítés adja. Ennek az eljárásnak a főbb hátránya, nevezetesen, hogy nem ad kellően pontos eredményt. Egy távolság alapú súlyozási eljárással pontosítottunk a LNN algoritmuson, melynek alapvetően két verzióját is leteszteltük.

LNN használatakor a következő problémát tapasztaltuk, melynek megoldása volt az egyik fő motiváció egy priorizáló algoritmus fejlesztése. A problémát a 4.1. ábra szemlélteti.

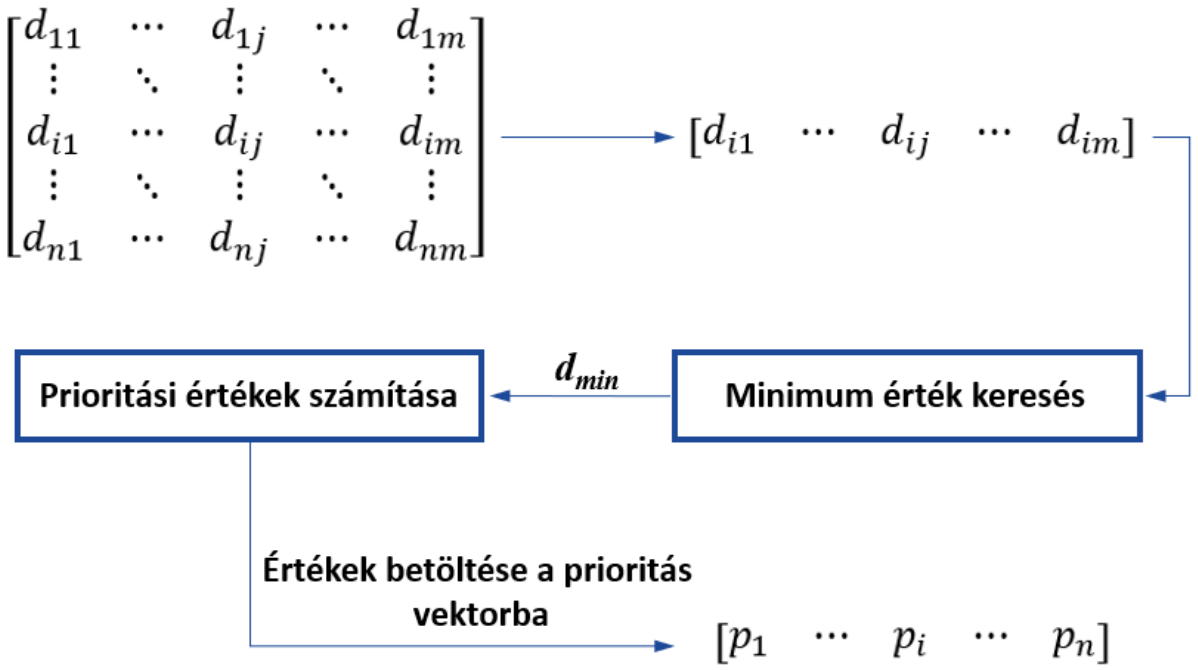


4.1. ÁBRA: *Saját fejlesztés motivációja*

Mint ahogy az ábra is szemlélteti, abban az esetben, ha két objektum (kék pontok), a hozzájuk tartozó gating size-ok (kék ellipszisek) és két detekció (piros pontok) a következőképpen helyezkednek el, akkor az LNN algoritmus az 1-es objektumhoz, a legközelebbi, 1-es detekciót rendeli, így a 2-es objektumnak nem hagyva detekciót. Ehelyett sokkal jobb lenne, hogy ha az 1-es objektumhoz a 2-es detekciót és a 2-es objektumhoz az 1-es detekciót rendeljük. Tehát kell egy olyan algoritmus, mely eszerint rendeli össze az objektumokat a detekciókkal. A cél az, hogy az egyértelmű párosításokat kapásból megtaláljuk és elvégezzük rájuk az asszociációt. A példánkban akkor érhetnénk el az ideális állapotot, azt, hogy mindkét objektum detekcióhoz jusson, hogy ha a 2-es objektum, 1-es detekció pár, az összerendelés során, magasabb prioritást kap, tehát előbbre vesszük, hisz ott egyértelmű, hogy ők összetartoznak, viszont ugyanez nem mondható el az 1-es objektum és a két detekció esetén, melyek mindegyike beleesik ezen objektumnak a környezetébe.

4.1. Local Nearest Neighbor algoritmus továbbfejlesztése

A Local Nearest Neighbor, futási teljesítményre vonatkozóan, egy jobb megoldás, mint a Global Nearest Neighbor, viszont nagy hátránya, hogy nem ad kellően pontos eredményt. A kutatás során ezen, utóbbi rossz tulajdonságát javítottam, a kis számításkapacitási igénye megőrzése mellett.



4.2. ÁBRA: A továbbfejlesztett Local Nearest Neighbor folyamatábrája

Első ötlet az volt, hogy objektumokhoz rendeljünk detekciókat, oly módon, hogy annak az az objektum kap magasabb prioritást, amelyhez egyértelműbben meghatározható, hogy mely detekció tartozik. Mindezt egy olyan logika alapján, miszerint megvizsgáljuk azt, hogy az adott objektumhoz közel álló detekciók mennyire is közeliek valójában. A viszonyítási alapot az objektumhoz legközelebb álló detekció szolgáltatja.

Tehát mint azt a 4.2. ábra is mutatja, először, a már korábban felépített távolságmátrixunkból kiválasztjuk azt az i-dik sort mely a pillanatnyilag vizsgált o_i objektumhoz tartozik. Ebből az m elemű vektorból, ahol m a detekciók számát jelöli, meghatározhatjuk a minimális értéket, és annak a vektorban való elhelyezkedését is, mely által deklarálásra kerül az adott objektumunkhoz legközelebb álló detekció.

Az objektum prioritásának számolásánál a legközelebb álló detekción kívül minden másikat figyelembe vesszük, így egy olyan számértéket kapva, miáltal deklarálható az, hogy adott objektumunkhoz tartozó, legközelebb álló detekció valójában mennyire biztosan tartozik hozzá. Azon a detekción kívül, amely a legközelebb áll az objektumhoz, ha a többi detekció nagyon távol esik, akkor a prioritás magas, ha pedig ezen detekciók a legközelebbihez hasonlóan közel helyezkednek az objektumhoz, akkor bizonytalanabbak vagyunk abban, hogy mely detekció is tartozik pontosan az objektumhoz, ezért a prioritás alacsony. Így ezen prioritás értékek alapján rangsoroljuk az objektumokat. Minél nagyobb az adott objektum prioritása, annál korábban rendelünk hozzá detekciót, ezáltal pontosítva a Local Nearest Neighbor algoritmust. Vizsgált detekciók mindegyike az objektum küszöbtávolságán γ_g belül helyezkednek el, mely kedvező, hisz ez az érték leírja az objektum bizonytalanságát. Az i-dik objektum prioritása:

$$P^o{}_i = \begin{cases} \frac{1}{\gamma_g}\left(\sum_{z_j \in \boldsymbol{Z}_i} w_{ij} \cdot d_{ij}\right), & \text{ha } n_{\boldsymbol{Z}_i} > 1 \\ 1 - \frac{d_{ij_i}}{\gamma_g}, & \text{ha } n_{\boldsymbol{Z}_i} = 1 \\ 0, & ha\ n_{\boldsymbol{Z}_i} = 0 \end{cases} \quad \textbf{(4.1.)}$$

és

$$\boldsymbol{Z}_i \subseteq \boldsymbol{Z} = \forall z_j \in \boldsymbol{Z}: d_{ij} < \gamma_g \backslash z_{j_i} \quad \textbf{(4.2.)}$$

ahol d_{ij} az i-dik objektum és a j-dik detekció közti távolság, $n_{\boldsymbol{Z}_i}$ azon objektumok száma, melyek küszöbtávolságon belül helyezkednek el, $\boldsymbol{Z}$ a detekciók halmaza, $\boldsymbol{Z}_i$ az a részhalmaz, mely a küszöbtávolságon belül eső detekciókat tartalmazza, z_{j_i} a legközelebb eső detekció, d_{ij_i} az egyetlen detekció (mely küszöbtávolságon belül van) és az objektum távolsága.

Mint ahogy a 4.1. egyenlet is mutatja három esetet különböztetünk meg. Az objektum prioritása zérus, tehát legutoljára rendelünk hozzá detekciót az összes objektum közül, ha nincs a küszöbtávolságán belül egyetlen detekció sem. Prioritása $1 - \frac{d_{ij_i}}{\gamma_g}$, ha egyetlen egy detekció esik a küszöbtávolságba. Míg a harmadik prioritás szamításra akkor kerül sor, amikor több detekció is a környezetében van. Ekkor mindegyik detekciónak adunk egy, az objektumtól mért, Mahalanobis távolságok alapján számított súlyt, mely az objektumhoz való hozzátartozási mértékét hivatott reprezentálni. Ezt a 4.3. egyenlet szemlélteti.

$$w_{ij} = \frac{1}{d_{ij}}\left(\sum_{z_j \in \boldsymbol{Z}_i} \frac{1}{d_{ij}}\right)^{-1} \quad \textbf{(4.3.)}$$

History-based objektum menedzsmentet használunk. Az imént tárgyalt objektumpriorizálási eljárást külön alkalmazzuk megerősített (*confirmed*) és nem megerősített (*tentative*) objektumok esetében, melyek közül minden esetben a megerősített objektumokra számolunk először prioritást, ezáltal a detekciók is előbb itt kerülnek hozzárendelésre. Utána a nem megerősítettekre is ugyanígy számolunk prioritást és a megmaradt detekciók pedig itt kerülnek hozzárendelésre, ezáltal is pontosítva az adatasszociációt. Az objektum priorizált LNN eljárást az Algoritmus 1 reprezentálja.

Algoritmus 1 Objektum-orientált LNN priorizálással

1: adott $D = \{d_{ij}\}\ i = 1, \dots, n\ \ j = 1, \dots, m$ távolság (költség) mátrix,
2: $\forall a_{ij} \in A = 0$

Objektum priorizálás:

3: D szétbontása $D_c = \{d^c_{i_c j}\}\ i_c = 1, \dots, n_c$ és $D_t = \{d^c_{i_t j}\}\ i_t = 1, \dots, n_t$ mátrixokra
4: **for** $i = 1$–**től** $n_c + n_t$ –**ig**
5: **if** $i \leq n_c$
6: $P_i^{o_c}$ prioritás kiszámítása $\{d^c_{ij}\}_{j=1}^m$ alapján (4.1.) szerint
7: **else**
8: $P_{i-i_c+1}^{o_t}$ prioritás kiszámítása $\{d^t_{(i-i_c+1)j}\}_{j=1}^m$ alapján (4.1.) szerint
9: **end if**
10: **end for**
11: D_c és D_t sorainak permutálása $\left\{P_{i_c}^{o_c}\right\}_{i_c=1}^{n_c}, \left\{P_{i_t}^{o_t}\right\}_{i_t=1}^{n_t}$ szerinti csökkenő sorrendbe

Hozzárendelési feladat megoldása:

12: **for** $i = 1$–**től** $n_c + n_t$ –**ig**
13: **if** $i \leq n_c$
14: $d_{ij_i} = \min_j \{d^c_{ij}\}_{j=1}^m$
15: **else**
16: $d_{ij_i} = \min_j \{d^t_{(i-i_c+1)j}\}_{j=1}^m$
17: **end if**
18: **if** $d_{ij_i} < \gamma_g$
19: $a_{ij_i} = 1$
20: $\forall d^c_{i_c j_i} \in D_c = \gamma_g \quad (i_c = 1, \dots, n_c)$
21: $\forall d^t_{i_t j_i} \in D_t = \gamma_g \quad (i_t = 1, \dots, n_t)$
22: **end if**
23: **end for**
24: **kimenet:** $A = \{a_{ij}\}\ i = 1, \dots, n\ \ j = 1, \dots, m$ asszociációs mátrix

Második ötlet az volt, hogy az egyes detekciókhoz rendeljük hozzá az objektumokat. Ez esetben a priorizálási eljárás eltérő az imént tárgyalt objektum priorizálástól. Az *j*-dik detekcióhoz tartozó prioritást az alapján számoljuk, hogy a hozzá legközelebb elhelyezkedő objektum a többi objektumhoz képest mennyire is számít közelinek:

$$\mathrm{P}^{d}{}_j = 1 - d_{i_j j} \cdot \sum_{i=1}^{n} \frac{1}{w_{ij} \cdot d_{i_j j}} \qquad \textbf{(4.4.)}$$

ahol $d_{i_j j}$ a *j*-dik detekció és az *i*-dik objektum közti távolság.

A súly hasonlóképpen a korábbi eljáráshoz:

$$w_{ij} = \frac{1}{d_{ij}} \left(\sum_{i=1}^{n} \frac{1}{d_{i_j j}} \right)^{-1} \qquad \textbf{(4.5.)}$$

Ezen eljárás során nem alkalmazunk egy γ_g küszöbtávolságot az adatok plusz előfeldolgozása esetén, így a detekció priorizálás során az egész mérési téren történnek az összehasonlítások, nem pedig egy-egy szűkebb környezeten belül. Érdemes azt a tényt figyelembe venni, hogy így nem dobunk el hasznos információkat, melyeket távolabb eső detekciók szolgáltathatnak a priorizálási folyamat során. A detekció priorizált LNN megközelítést az Algoritmus 2 ír le.

Algoritmus 2 Detekció-orientált LNN priorizálással

adott $D = \{d_{ij}\}\ i = 1, \dots, n\ \ j = 1, \dots, m$ távolság (költség) mátrix,
$\forall a_{ij} \in A = 0$

Detekció priorizálás:

D szétbontása $D_c = \{d^c_{i_c j}\}\, i_c = 1, \dots, n_c$ és $D_t = \{d^c_{i_t j}\}\, i_t = 1, \dots, n_t$ mátrixokra
for $j = 1$–**től** m –**ig**
    $P_j^{d_c}$ és $P_j^{d_t}$ prioritások kiszámítása $\{d^c_{i_c j}\}_{i_c=1}^{n_c}$ és $\{d^t_{i_t j}\}_{i_t=1}^{n_t}$ alapján (4.4.) szerint
    end if
end for
D_c és D_t oszlopainak permutálása $\left\{P_j^{d_c}\right\}_{j=1}^{m}$, $\left\{P_j^{d_t}\right\}_{j=1}^{m}$ szerinti csökkenő sorrendbe

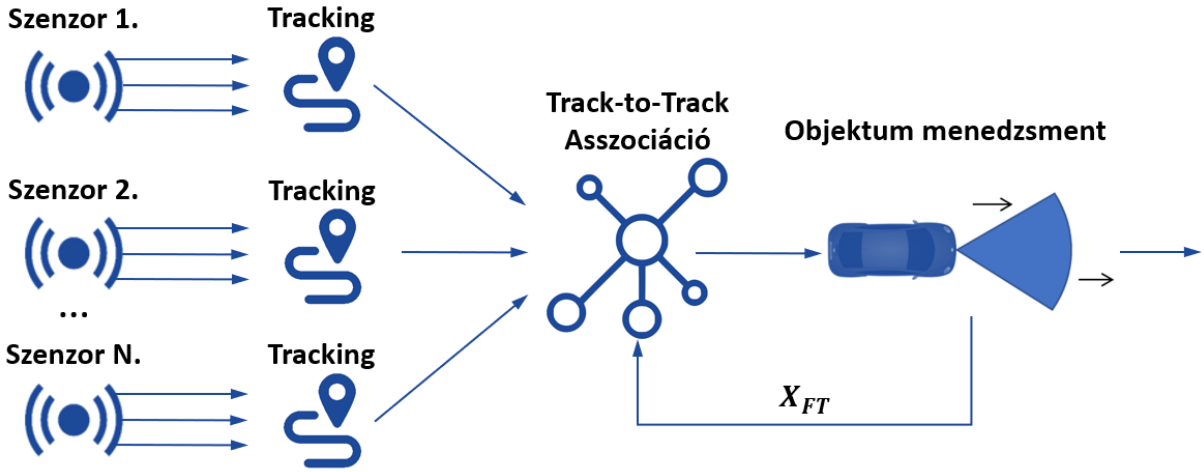
Hozzárendelési feladat megoldása:

for $j = 1$–**től** m –**ig**
$d_{i_j j} = \min_{i_c} \{d^c_{i_c j}\}_{i_c=1}^{n_c}$
    if $d_{i_j j} \geq \gamma_g$
        $d_{i_j j} = \min_{i_t} \{d^t_{i_t j}\}_{i_t=1}^{n_t}$
    end if
    if $d_{i_j j} < \gamma_g$
        $a_{i_j j} = 1$
        $\forall d^c_{i_j j} \in D_c = \gamma_g \qquad (j = 1, \dots, m)$
        $\forall d^t_{i_j j} \in D_t = \gamma_g \qquad (j = 1, \dots, m)$
    end if
end for
kimenet: $A = \{a_{ij}\}\ i = 1, \dots, n\ \ j = 1, \dots, m$ asszociációs mátrix

Az objektum priorizált LNN algoritmushoz hasonlóan, itt is History-based objektum menedzsmentet használunk, ahol ugyanúgy a megerősített, majd pedig a nem megerősített objektumokhoz rendelünk előbb detekciókat.

4.2. Track-to-Track fúzió továbbfejlesztése

Korábban, a 3.2-es fejezetben már részletezett objektumfúziós architektúra fejlesztése volt a kutatás egyik fő pilére, hisz hiába nyerünk jelentős számításkapacitás igényt a tracking-nél, ha a fúzió realizálásának továbbra is nagy a futásideje.

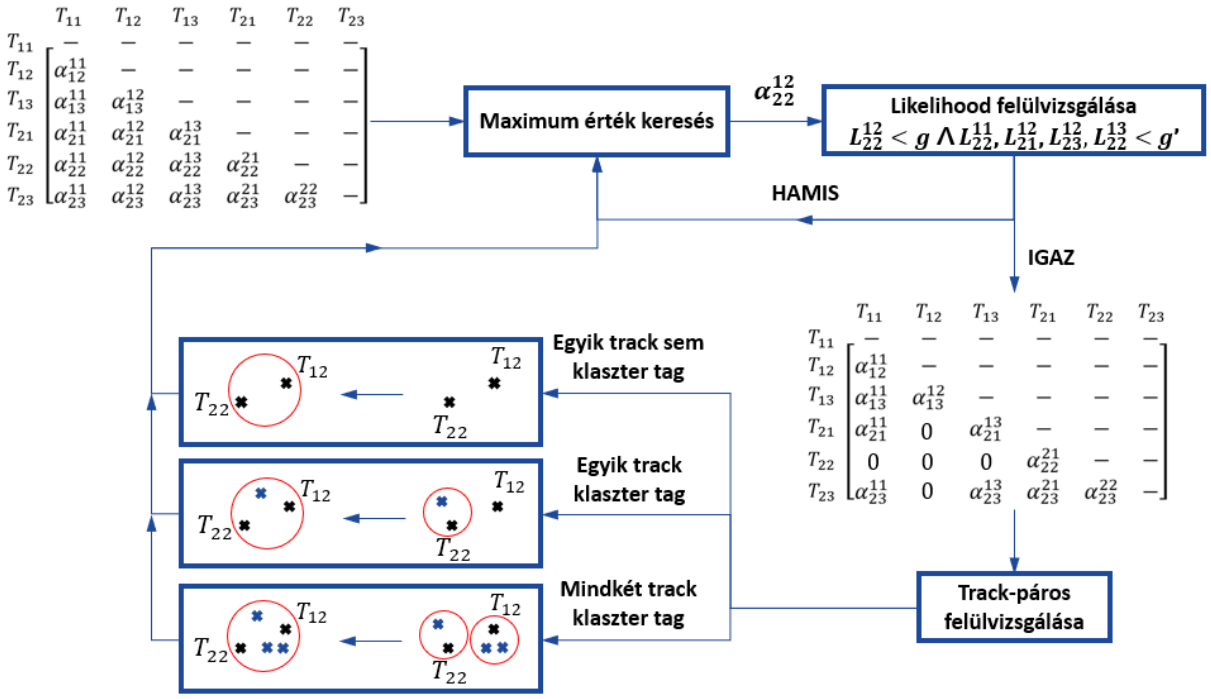


4.3. ÁBRA: Hibrid track-to-track architektúra visszacsatolással

Mint azt a 4.3. ábra is szemlélteti, az általunk alkalmazott objektumfúziós architektúra esetén, az objektum menedzsment után, a fuzionált trackek (X_{FT}) állapotát felhasználjuk az asszociációs blokkban, mellyel így egy hibrid track-to-track architektúrát kapunk, ugyanis a bemenetek összerendelése track-to-track asszociációval történik, de a visszacsatolás Multi-Object Tracking architektúrát követ.

A track-to-track architektúránál a fúziós blokkba már trackelt objektumok érkeznek, melyek mindegyike ugyanazon az állapottéren értelmezett, ezáltal összehasonlíthatóságukat kihasználva egy csoportosító (*clustering*) algoritmust fejlesztettem, mely ezen trackelt objektumokat (trackek) megfelelteti egymással, oly módon, hogy $S_1, S_2, S_3, \dots, S_n$ szenzoroktól érkező ugyanazon trackek összerendelésre kerüljenek. Általánosan használt clustering eljárásoknál egy klaszterbe, egy adott S_i szenzor maximum egy track-et realizálhat és emellett egy adott T_{ij} track, ahol i a szenzor ID, j pedig a track ID, nem lehet két különböző klaszterben. Ezzel azt feltételezzük, hogy egy adott szenzor nem generálhat egy adott objektumról egynél több track-et, mely egy teljesen ideális eset. Valóságban könnyen előfordulhat, hogy több detekciót küld egy szenzor ugyanarról az objektumról, így a kutatásunk ezen részének egyik fő célja az volt, hogy egy hagyományos clustering algoritmust úgy paraméterezhessünk, hogy ezt a duplikációs problémát is kezelni tudja, természetesen a futási időre ügyelve.

A clustering lépéseit a következő ábra szemlélteti:



4.4. ÁBRA: A clustering folyamata

Mint ahogy azt az ábra is mutatja először létrehozunk egy $N \times N$-es mátrixot, ahol N az összes szenzortól beérkezett trackek száma. A mátrix elemei egyértelműen meghatározzák, hogy egyes track-párosításoknak mekkora az asszociációs valószínűsége. Ezen asszociációs értékek számítását a következő összefüggés szemlélteti:

$$\alpha_{ij} = 1 - \frac{L_{ij}}{g_{ij}} > 0 \qquad \textbf{(4.6.)}$$

Ahol L_{ij} a párosítás likelihoodja, g_{ij} pedig az i és j trackre értelmezett specifikus gate-value. Ezen értékek számolását a 4.7. és 4.8. egyenletek szemléltetik.

$$L_{ij} = \left(x_i^A - x_j^B\right)^T \left(P_i^A + P_j^B\right)^{-1} \left(x_i^A - x_j^B\right) + \ln\left(\left|P_i^A + P_j^B\right|\right) \qquad \textbf{(4.7.)}$$

$$g_{ij} = 2\log\left(\frac{\beta_{FT} P_{TA} P_{TB}}{2\pi^{\frac{D}{2}} \beta_{NTA} \beta_{NTB}}\right) \qquad \textbf{(4.8.)}$$

Ahol β_{NTA} az A szenzor target density értéke arra nézve, hogy a B szenzor j trackjére nem létezik az A szenzornak trackje és mely megmutatja, hogy ezen track hogyan illeszkedik az X_{FT} (kimeneti objektumok) környezetre. Ezt a 4.9. ír le. β_{NTB} a B szenzor target density értéke arra nézve, hogy a A szenzor i trackjére nem létezik a B szenzornak trackje és mely az előzőhöz hasonlóan megmutatja, hogy ezen track hogyan illeszkedik az X_{FT} környezetre. Ezt pedig 4.10. ír le. β_{FT} a final tracking target density értéke, melynél elő fuzionáljuk az i illetve j trackeket a Covariance intersection algoritmus segítségével, így megmondható, hogy ezen két track fuzionált eredménye hogyan illeszkedik a X_{FT} (kimeneti objektumok) környezetre. Így

megfogalmazható, hogy g_{ij} azon arány, melynél a fuzionált track jobban illeszkedik a kimeneti objektumokra, mint önmagukban a két track külön. β_{FT}-t a 4.11., az i és j track elő fuzionálását pedig 4.12. egyenlet írja le. P_{TA} és P_{TB} pedig a szenzorok tracking probability értéke, tehát annak a valószínűsége, hogy egy adott tracket A vagy éppen B szenzor követ.

$$\beta_{NTA} = p\left(x_j^B \middle| X_{FT}\right) P_{TB}(1 - P_{TA}) + \beta_{FB} \quad \textbf{(4.9.)}$$

$$\beta_{NTB} = p\left(x_i^A \middle| X_{FT}\right) P_{TA}(1 - P_{TB}) + \beta_{FA} \quad \textbf{(4.10.)}$$

$$\beta_{FT} = p\left(x_{ij}^F \middle| X_{FT}\right) \quad \textbf{(4.11.)}$$

$$x_{ij}^F = P_j^B \left(P_i^A + P_j^B\right)^{-1} x_i^A + P_i^A \left(P_i^A + P_j^B\right)^{-1} x_j^B \quad \textbf{(4.12.)}$$

Ahol β_{FA} és β_{FB} az A illetve B szenzor fals track density értéke. $p\left(x_i^A \middle| X_{FT}\right)$ az A szenzor i track likelihoodja a környezet (X_{FT}) függvényében, $p\left(x_B^j \middle| X_{FT}\right)$ pedig a B szenzor j track likelihoodja a környezet függvényében és hasonlóképpen β_{FT} esetén, melyek számítását a 4.13. egyenlet mutatja be.

$$p(x|X_{FT}) = \beta_b + \sum_{k=1}^{N} \mathcal{N}\left(x,\ \hat{x}_k, \hat{P}_k\right) p(\exists x_k) \quad \textbf{(4.13.)}$$

Ahol $\hat{x}_k$ az állapot vektora a k-dik kimeneti objektumnak, $\hat{P}_k$ pedig a hozzá tartozó kovariancia érték. Ezen objektumokból van N darab, melyeken végigmenve kiértékeljük a Gauss-eloszlást az x helyen, hisz a x track likelihoodját szeretnénk megkapni. Mindezt pedig megszorozzuk a k-dik kimeneti objektum létezési valószínűségével ($p(\exists x_k)$) és hozzáadjuk a birth density-t (β_b), mely az új objektumok sűrűségét reprezentálja.

Trackek duplikálásának a kiszűrésére bevezettünk egy új, szigorúbb küszöbértéket, mely az eredeti gate-value-hoz hasonlóan specifikusan track-párokra értelmezett. Ezen g'_{ij} érték számítását a 4.14. egyenlet ír le.

$$g'_{ij} = 2 \log\left(\frac{\beta_{FT} P_{TA} P_{TB} (P_{DA} + P_{DB})}{2\pi^{\frac{D}{2}} \beta_{NDA} \beta_{NDB}}\right) \quad \textbf{(4.14.)}$$

Ahol β_{NDA} és β_{NDB} a szenzorok specifikus density értéke arra vonatkozóan, hogy a szenzorok nem duplikálnak. Melyeket a következő 4.15. és 4.16. egyenletek írnak le.

$$\beta_{NDA} = p\left(x_i^A \middle| X_{FT}\right) P_{TA}(1 - P_{DA}) \quad \textbf{(4.15.)}$$

$$\beta_{NDB} = p\left(x_j^B \middle| X_{FT}\right) P_{TB}(1 - P_{DB}) \quad \textbf{(4.16.)}$$

Ha egy track-páros likelihood értéke kisebb mint g'_{ij}, akkor duplikáció áll fenn. Ezen g' kitalálásakor onnan indultam, hogy kell egy szigorúbb threshold érték track páronként mint g.

Látható, hogy a mátrixnak csupán az egyik, a főátló alatti elemeit használjuk. Ez érthető, hisz azon feletti értékek megegyeznének az átló alattiakkal, így nincs értelme annak, hogy minden egyes értékbetöltésnél a mátrix két különböző cellájába is berakjuk ugyanazt a likelihoodot. A főátlót már a mátrix generálásánál feltöltjük nulla értékekkel, hisz egy adott T_{ij} tracket semmiképp sem szeretnék összerendelni önmagával.

A teljes mátrix legenerálása után kiválasztjuk a maximális asszociációs értéket a mátrixból, majd megvizsgáljuk, hogy a hozzá tartozó likelihood L_{ij} érték, a kiszámított g_{ij} értékünknél kisebb-e, akkor az α_{ij}-hez tartozó, a mátrixon belüli helye alapján meghatározott i és j trackek összerendelhetők. Ez után ezen trackekhez tartozó sorban azokat az oszlop elemeket nullázzuk, melyek ugyanattól a szenzortól származnak, és a hozzá tartozó oszlopból pedig azon sor elemeket nullázzuk, melyek a szenzorpár másik eleméhez tartoznak. Tehát, mint ahogy a 4.4. ábra is szemlélteti, a maximum elemünk az α_{22}^{12}, így a hozzá tartozó sor elemei közül azokat nullázzuk, melyek az 1-es szenzortól származnak, míg a hozzá tartozó oszlopban pedig azon elemek kerülnek kinullázásra, melyek a 2-es számú szenzortól származnak. De mind ez csak akkor teljesül, ha $L_{22}^{12} < g_{22}^{12}$ reláció igaz. Abban az esetben, ha ezen sor és oszlop elemek közül, a példához hűen, nevezetesen L_{22}^{11}, L_{21}^{12}, L_{23}^{12} és L_{22}^{13} elemek esetében felülvizsgáljuk, hogy a rájuk számított, specifikus g' értéknél kisebbek-e. Azon elemek melyek ezt a feltételt teljesítik nem kerülnek kinullázásra, mivel feltételezzük, hogy akkor azon track egy duplikáció. Példánkra visszatérve, ha α_{22}^{13} asszociációs értékhez tartozó L_{22}^{13} likelihood érték kisebb, mint g'^{13}_{22}, akkor ebben az esetben az egyes szenzor duplikálva küldi az egyik trackjét, nevezetesen T_{12} duplikációja a T_{13}. Későbbi iterációk során megtaláljuk ezen track párosítást újra, hisz értékét nem nulláztuk, így megengedve a duplikálást és ekkor kerülhet bele abba a klaszterbe, ahova T_{12} és T_{22}-t is tettük.

Az összerendelhető trackek ID-ját és a hozzájuk tartozó cluster ID-ját eltároljuk. A már felhasznált asszociációs értékeket kinullázzuk a mátrixon belül, így egy következő iterációban az algoritmus nem találja meg ugyanazt a track párost. Ez után lefuttatjuk a ciklust újra, majd újra, egészen addig, míg olyan maximum értékeket találunk az asszociációs mátrixon belül, melyek még küszöbértéken belül esnek.

Minden ciklus kezdetekor felülvizsgáljuk, hogy az adott két track, melyek összetartozásához az adott pillanatban kiválasztott asszociációs érték tartozik, benne van-e már egy létező klaszterben vagy sem. Ekkor három esetet különböztethetünk meg:

- A két track egyike sem klaszter tag
- A két track egyike már egy létező klaszter tagja
- Mindkét track már egy létező klaszter tagja

Az első eset a legegyszerűbb. Ez esetben a két track, ha összerendelésre kerül, akkor a hozzájuk tartozó klaszter ID-t $c + 1$ értékre módosítjuk, ahol c a már létező klaszterek számát jelöli.

A második eset már bonyolultabb. Tételezzük fel, hogy T_{12}, mely a jelölés alapján az egyes szenzortól származó kettes track, és a T_{22} track vizsgálatakor T_{22} már egy klaszter tagja, melyben több másik szenzortól érkezett track is van. Ekkor T_{12}-t egy szeparált, külön klaszterként értelmezzük és felülvizsgáljuk azt, hogy a T_{22} klaszterében szereplő többi trackhez milyen átlagos asszociációs értékkel rendelkezik. Ennek függvényében, ha ez az átlag érték, hasonlóképp a T_{22}-vel képezett likelihood-hoz, küszöbértéken belül esik, akkor T_{12} a T_{22} klaszterébe tartozhat.

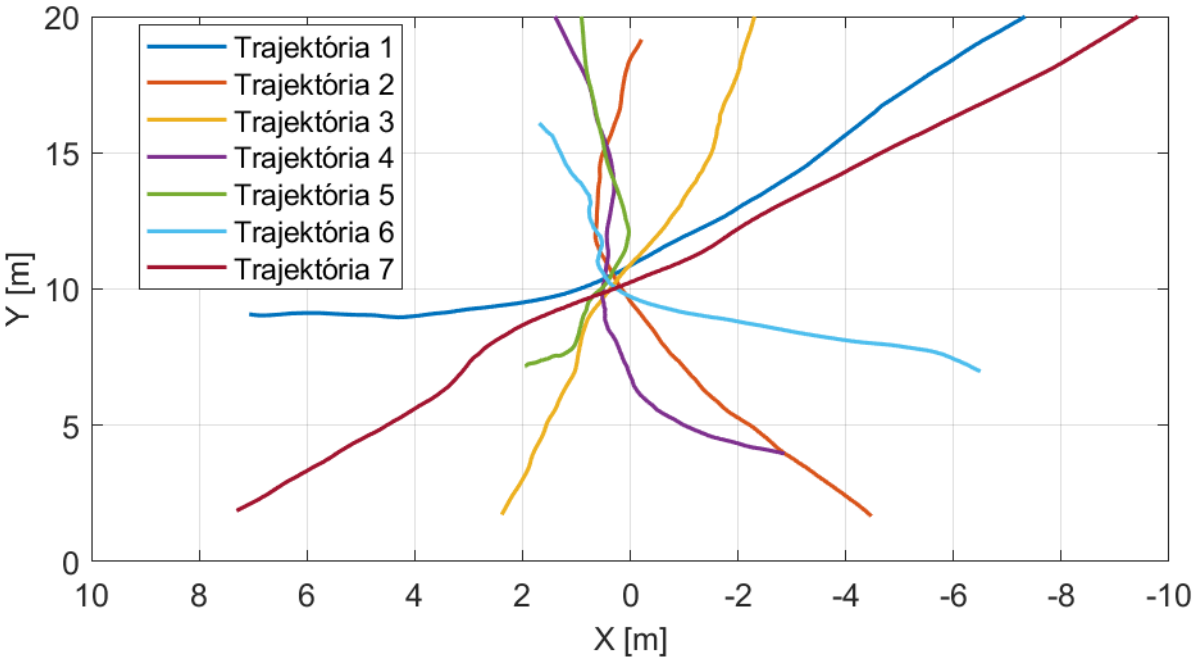
A harmadik eset az, amikor mindkét vizsgált track klaszter tag. Ekkor, mint az előző esetnél felülvizsgáljuk a klaszterek tagjait. Tehát előző példából kiindulva, legyen T_{12} az $\boldsymbol{A}$ klaszter tagja, míg T_{22} a $\boldsymbol{B}$ klaszteré. Megvizsgáljuk, hogy T_{22} milyen átlagos asszociációs likelihood

értékkel rendelkezik az ***A*** klaszter tagjaival, majd ugyanezt lefuttatjuk egy, T_{22}-től különböző, másik ***B*** klaszter-beli trackre. És ezt egészen az utolsó ***B*** klaszter-beli elemig folytatva és ha mindegyik track átlagos asszociációs értéke küszöbértéken belül esik, akkor a két klasztert egybe olvasztjuk.

5. Kiértékelés

5.1. A kiértékelési környezet

Az algoritmusainkat egy előre meghatározott objektum számú, állapotú, és trajektóriájú szcenárión teszteltük, melynél a legenerált 7 darab objektum, ugyanabban az időpillanatban, mi esetünkben a teljes futási idő felénél, és ugyanabban a pontban, nevezetesen a mérési tér közepén találkozzon. Az objektumokhoz véletlenszerűen választott konstans sebességekre additív zaj is került. Célunk ezen a szcenárión való teszteléssel az volt, hogy megnézzük az egyes algoritmusok, az objektumok keresztezése után milyen pontos becslést adnak azok állapotára vonatkozóan. A trajektóriák az 5.1. ábrán láthatóak:



5.1. ÁBRA: A kiértékelési szcenárió

Egy szimulációs környezetet hoztunk létre, ahol szimuláljuk a szenzorokat, azok paramétereit, detekcióit és a már fentebb tárgyalt objektumokat is. Mindehhez konstans sebességmodellt alkalmazva. Az állapotvektort az 5.1. egyenlet írja le.

$$\boldsymbol{x}_t = [x_p \quad v_x \quad y_p \quad v_y]^T \qquad \textbf{(5.1.)}$$

A szenzorok nem minden esetben detektálják az összes objektumot, ezért létrehoztunk egy tracking valószínűségi változót, mely azt hivatott reprezentálni, hogy mekkora annak a valószínűsége, hogy egy adott időpillanatban egy adott objektumot képes detektálni a

szenzorunk. Szenzorspecifikus értékeket az alábbi 5.1. táblázat mutatja. Ezen felül a szenzorok csupán x és y irányú pozíciót mérnek, tehát a sebességek (v_x, v_y) becsült értékek.

Paraméterek	**Jelölés**	***A* szenzor**	***B* szenzor**
Tracking valószínűség	P_T	0,85	0,9
Duplikációs valószínűség	P_D	0,2	0,15
Poisson-eloszlás	λ	2	2
x irányú gyorsulási zaj $[m/s^2]$	a_{n_x}	1	1
y irányú gyorsulási zaj $[m/s^2]$	a_{n_y}	1	1
x irányú mérési zaj $[m]$	μ_x	0,2	0,2
y irányú mérési zaj $[m]$	μ_y	0,2	0,2

5.1. Táblázat: *Szenzor paraméterek*

Az objektumok nem detektálása mellett van, hogy a szenzor hamis detekciókat küld. Ezen eseteket is szimuláltuk egy előre beállított értékkel, hogy milyen sűrűn vannak jelen ezen fals detekciók a mérési térben. Ezt a sűrűség értéket a Poisson-eloszlás mértéke és a mérési tér mérete alapján számítottuk:

$$\varrho = \frac{\lambda}{A} = \frac{2}{20 \cdot 20} = 0{,}005\ detekció/m^2 \qquad \textbf{(5.2.)}$$

ahol λ a Poisson-eloszlást jellemző mérték, A pedig a mérési tér területnagysága.

A szenzorok mérési bizonytalanságát a következő diagonális mátrix írja le:

$$R = \begin{bmatrix} \mu_x^2 & 0 \\ 0 & \mu_y^2 \end{bmatrix} \qquad \textbf{(5.3.)}$$

Paraméterek	**Jelölés**	**Érték**
Szimulációs tér x irányban $[m]$	A_x	0-tól 20-ig
Szimulációs tér y irányban $[m]$	A_y	-10-től 10-ig
Valós objektumok alap sebessége $[m/s]$	v_b	-2
Sebesség zaj $[m/s]$	v_n	4
Mérési mintavételezés ideje $[s]$	t	10
Mérési mintavételezés gyakorisága $[Hz]$	f	10

5.2. Táblázat: *Szcenárió paraméterek*

Az i-dik objektum sebességét az 5.4. egyenlet írja le.

$$v_i = v_b + [v_{n_x}, v_{n_y}] \qquad \textbf{(5.4.)}$$

ahol v_{n_x} és v_{n_y} az objektum x és y irányú sebesség zaja, melyeket 5.5. és 5.6. írnak le.

$$v_{n_x} = v_n \cdot \kappa_1 \qquad \textbf{(5.5.)}$$

$$v_{n_y} = v_n \cdot \kappa_2 \quad \textbf{(5.6.)}$$

$$0 < \kappa_1, \kappa_2 < 1 \quad \textbf{(5.7.)}$$

A folyamatzaj kovariancia mátrixát az 5.8. egyenlet szemlélteti.

$$\boldsymbol{Q} = \boldsymbol{Q_x} + \boldsymbol{Q_y} \quad \textbf{(5.8.)}$$

$$\boldsymbol{Q_x} = {a_{n_x}}^2 \cdot \begin{bmatrix} \Delta T^4/4 & \Delta T^3/2 & 0 & 0 \\ \Delta T^3/2 & \Delta T^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \textbf{(5.9.)}$$

$$\boldsymbol{Q_y} = {a_{n_y}}^2 \cdot \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \Delta T^4/4 & \Delta T^3/2 \\ 0 & 0 & \Delta T^3/2 & \Delta T^2 \end{bmatrix} \quad \textbf{(5.10.)}$$

Az objektumok állapotát a helyzetük a mérési téren és a becsült x és y irányú sebességeik írják le.

Paraméterek	**Jelölés**	**Érték**
Küszöbtávolság [m]	γ_g	9
Objektum megerősítéshez szükséges minimális detektálási szám	M	3
Objektum megerősítés ciklusszáma	N	4
Objektum törléséhez szükséges minimum kihagyott detektálás száma	P	5
Objektum törlés ciklusszáma	R	5

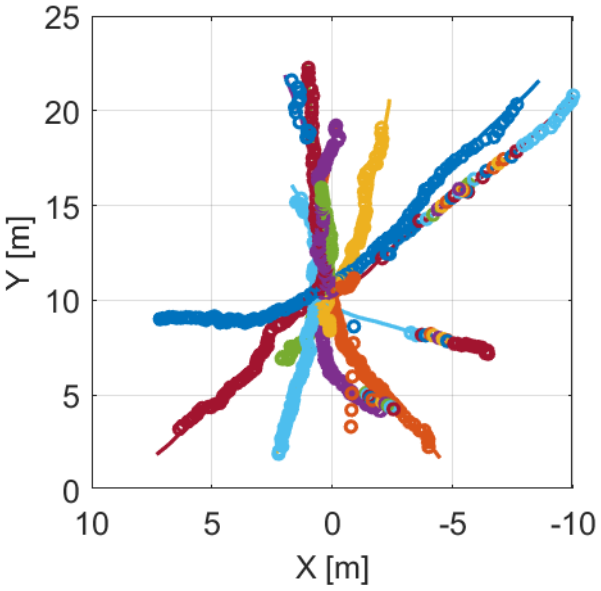
5.3. Táblázat: *Filter paraméterek*

Fentebb, a history-based objektum menedzsmentnél tárgyalt N és M, melyek objektumok megerősítéshez szükséges, előre beállított értékek.

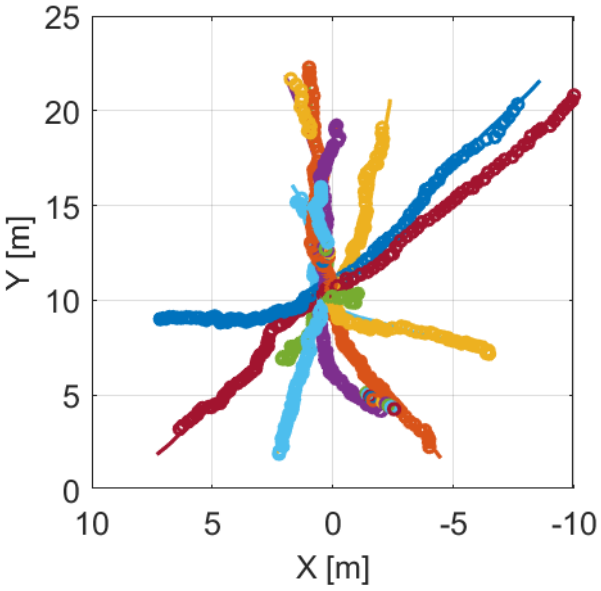
Tehát 4 ciklusból 3-szor kell egy objektumot detektálni, ahhoz, hogy az megerősítetté váljon. Hasonlóan az objektum törlésére vonatkozó P és R esetében, 5 ciklusból 5-ször ha nem detektálunk egy objektumot, akkor az törlésre kerül az objektumlistáról.

5.2. Eredmények

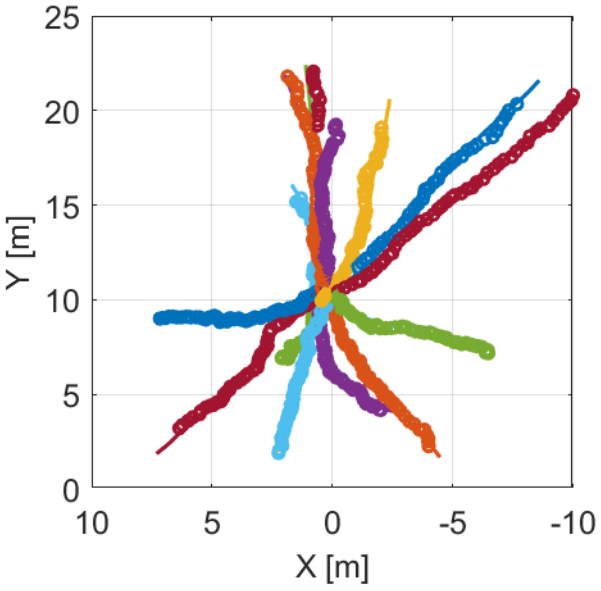
5.2.1. Tracking algoritmusok teljesítménye



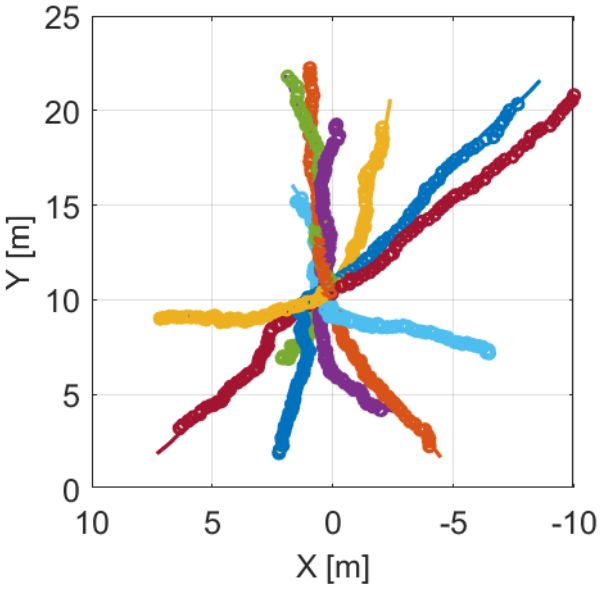
5.2. ÁBRA: LNN által nyomon követett trajektóriák



5.3. ÁBRA: GNN által nyomon követett trajektóriák



5.4. ÁBRA: Detekció priorizált LNN által nyomon követett trajektóriák



5.5. ÁBRA: Objektum priorizált LNN által nyomon követett trajektóriák

A szimulációs környezetben az egyes algoritmusok teljesítményét GOSPA (Generalized Optimal Sub-pattern Assignment) [24] rendszer alapján vizsgáltuk, mely figyelembe veszi az objektumok lokalizációs és számossági hibáit is, mely utóbbiba beletartoznak a nem detektált objektumok (fals-negatív) és a fals-pozitívok is.

t_k időpillanatban a valós objektumok:

$$Y = [y_1, y_2, \dots, y_m] \tag{5.11.}$$

t_k időpillanatban a trackek:

$$X = [x_1, x_2, \dots, x_n] \tag{5.12.}$$

Tehát a GOSPA a következőképpen néz kis, ha feltételezzük, hogy $m \leq n$:

$$GOSPA = \left[\sum_{i=1}^{m} d_c^P\left(y_i,\ x_{\pi(i)}\right) + \frac{c^P}{\alpha}(n-m) \right]^{\frac{1}{P}} \tag{5.13.}$$

Ahol, d_c a cutoff távolság, melynek értékét $3m$-re állítottuk. $x_{\pi(i)}$ reprezentálja azt a track-et, melyet a valós y_i-hez rendeltünk. c a cutoff távolság threshold értéke, α pedig az alfa paraméter, mely egy pozitív skalár a [0, 2] intervallumon.

A környezetérzékelés teljesítményét reprezentálja a P_r precision, R_c recall, F_1 pedig a precision és a recall harmonikus közepeként adódó eredő teljesítmény:

$$P_r = \frac{TP}{TP + FP} \tag{5.14.}$$

$$R_c = \frac{TP}{TP + FN} \tag{5.15.}$$

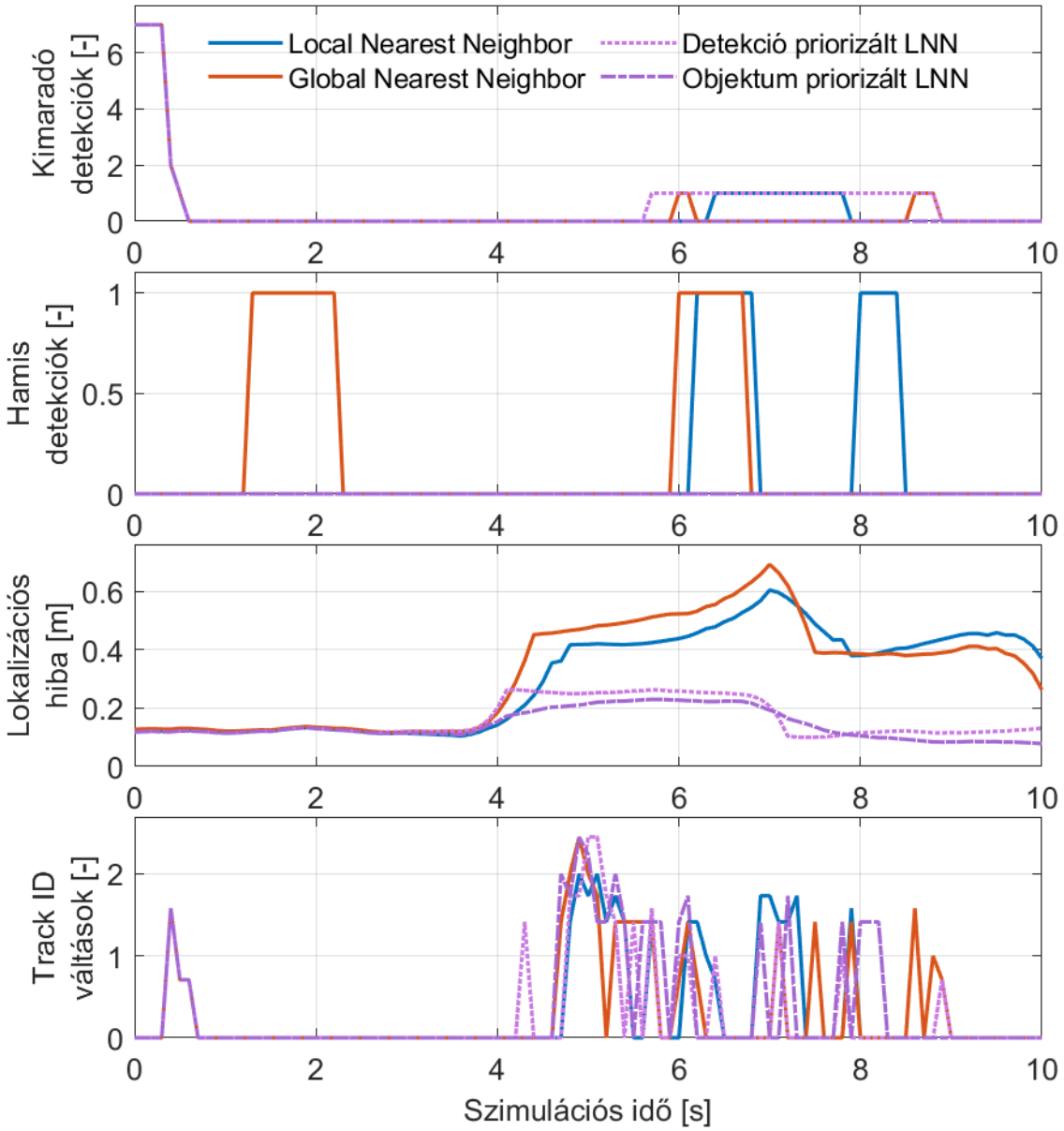
$$F_1 = \frac{2P_r \cdot R_c}{P_r + R_c} \tag{5.16.}$$

Melyeknél a TP a megtalált, valós objektumok száma (true-positive), FP a fals-pozitívok száma és FN pedig a nem detektáltaké. Tehát a precision azt mutatja meg, hogy a kiadott objektumok hány százaléka valós objektum, míg a recall azt mutatja, hogy az összes valós objektumból mennyit találtunk meg.

Algoritmusainkat összevetve a Local Nearest Neighbor (LNN) és a Global Nearest Neighbor (GNN) módszerekkel a GOSPA alapján:

	Precision	**Recall**	**Localization**	**GOSPA**	**Labeled GOSPA**	**F1 Score**
LNN + detekció priorizálás	100 %	91,09 %	0,161	1,0715	1,2275	95,33727563
LNN + objektum priorizálás	100 %	95,62 %	0,1422	0,4123	0,6882	97,76096514
LNN	96,76 %	93,07 %	0,2982	1,2582	1,4482	94,87913607
GNN	97,39 %	94,91 %	0,3147	1,0113	1,2002	96,13400832

5.4. Táblázat: *Tracking algoritmusok kiértékelésének eredménye*



5.6. Ábra: Tracking algoritmusok teljesítménye az idő függvényében

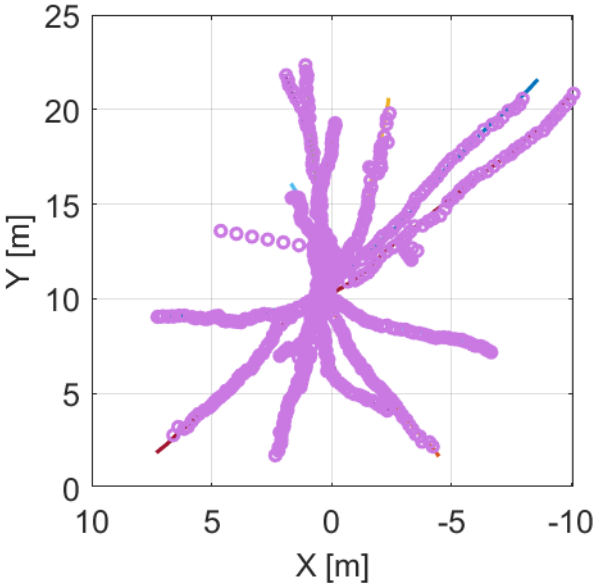
Az 5.2. ábrán jól látható, hogy a detekció priorizált LNN sokáig összevon két objektumot egyé. Ez [0, 18] körül látszik, hogy ott már szétválik az az egy track kettővé. Mivel a recall azt mutatja meg, hogy a valós objektumok hány százalékát találja meg az adott algoritmus, így ez az érték rosszabb, mint a többi három eljárás esetén. Ezt jól mutatja az 5.4. táblázat is. Az is látszik, hogy ez objektum priorizált LNN esetében viszont nem fordul elő, így a recall értéke is sokkal magasabb. Az is jól látható, hogy egyik általunk javasolt módszerben sincsen fals pozitív észlelés, hisz mindkét priorizálási eljárásnál a precision értéke 100%. Ez az érték LNN és GNN esetében számottevően alacsonyabb. GNN esetében az algoritmus egy globális optimumot ad, minden detekciót és objektumot egységesen figyelembe véve, így nagyobb annak a valószínűsége, hogy több fals pozitívot küld, mint a priorizáló megoldások. Ha a GOSPA értékeket figyeljük ott is egyértelműen látszik, hogy a detekció priorizált LNN közel olyan jól teljesít, mint a GNN és az objektum priorizált LNN pedig messze a legjobb eredményt mutatja, így átlagosan a legkisebb lokalizációs és számossági hibákat produkálva. Labeled GOSPA esetén azt vizsgáljuk, hogy egyes algoritmusok esetén mennyire folytonos a

track-elés. Amennyivel nagyobbak az egyes Labeled GOSPA értékek, annyival hajlamosabb az adott algoritmus cserélgetni a track-ekhez tartozó ID-kat. Itt is látszik az objektum priorizált LNN jobb teljesítménye. Végül pedig az F1 score reprezentálja az egyes algoritmusok redukált teljesítményét. Ezen értékeket a precision és recall értékek harmonikus közepeként kaphatjuk meg.

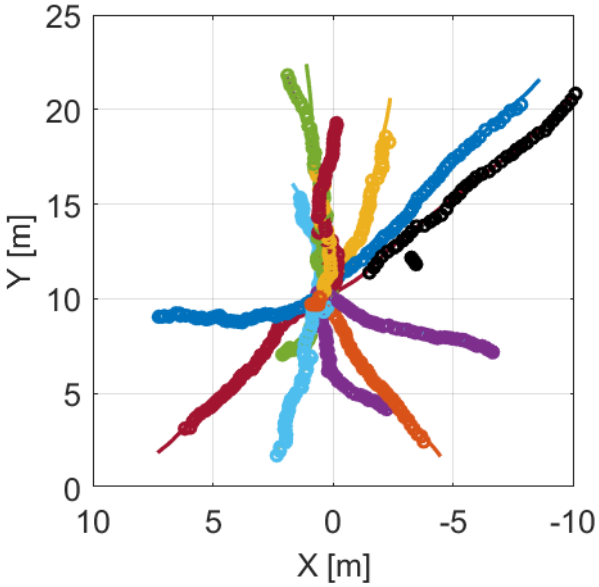
Összegezve az eredményeket, levonható az a következtetés, hogy az általunk fejlesztett két módszer közül az objektum priorizált LNN magasan a legjobb az összes algoritmus közül. A detekció priorizált LNN pedig megközelíti a GNN teljesítményét, mindezt úgy, hogy a lokalizációs hibái sokkal alacsonyabbak. Emellett a futási idő a GNN esetén n^3, míg a priorizáló megoldásoké csupán n^2, ahol n az objektumszámot jelöli.

5.2.2. Track-to-track fúziós algoritmusok teljesítménye

A saját fejlesztésű hibrid track-to-track algoritmusunkat Adam Houenou A Track-To-Track Association Method for Automotive Perception Systems [11] című cikkében tárgyalt track-to-track fúziós eljárással vetettük össze, hisz ez egy, a sajátunkhoz hasonló módon iteratív megközelítés, így megvizsgálva, hogy az általunk alkalmazott fuzionált trackek ($\boldsymbol{X_{FT}}$) visszacsatolása milyen hatása van a track-to-trak fúzióra.



5.7. ÁBRA: Track-to-track fúzió trajektóriái



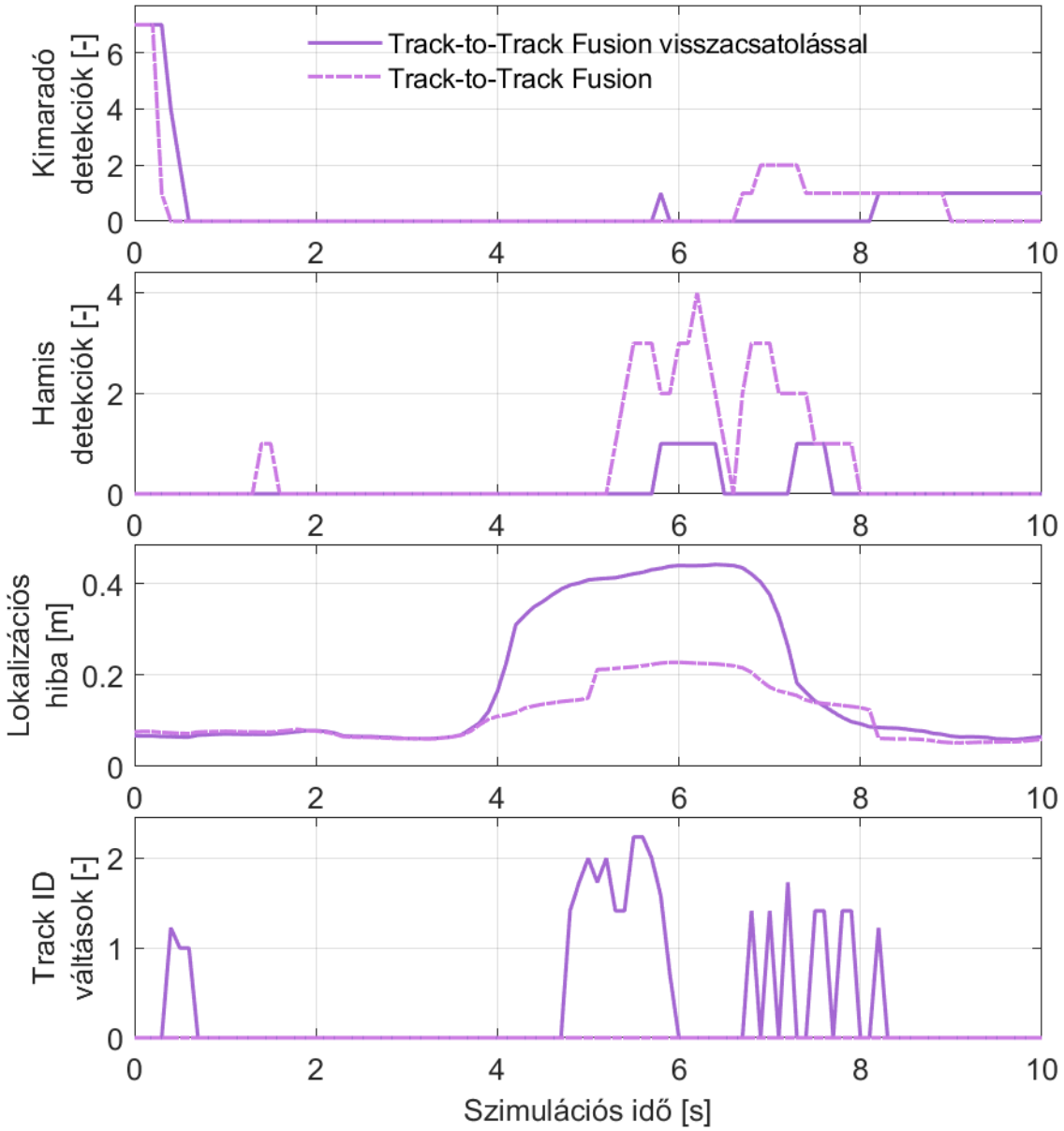
5.8. ÁBRA: Track-to-track fúzió + visszacsatolás trajektóriái

Dimitri J. Papageorgiou Track-to-track association and ambiguity management in the presence of sensor bias [12] című cikkében tárgyalt Global Nearest Pattern Matching (GNPM) algoritmus valós idejű alkalmazhatóság tekintetében nem alkalmas összehasonlítási alapnak, hisz az eredeti track-to-track fúziós probléma visszavezethető egy vegyes értékű lineáris programozási feladatra [12], [13][13], melyek megoldása általánosan N-P komplexitással bír. Az 5.5. táblázat jól szemlélteti a kiértékelési eredményeket.

	Precision	Recall	Localization	GOSPA	F1 Score
Track-to-track fúzió	91,89 %	92,93 %	0,1122	1,4097	92,407074
Track-to-track fúzió + visszacsatolás	98,34 %	92,36 %	0,1788	1,0656	95,25623912
A szenzor	94,86 %	93,92 %	0,1979	1,2014	94,38766
B szenzor	99,22 %	90,38 %	0,1773	1,3029	94,59392

5.5. Táblázat: *Track-to-track fúziós algoritmusok kiértékelésének eredménye*

A táblázat jól mutatja, hogy a recall érték valamivel alacsonyabb az általunk használt fúziós eljárásnál, mint a hagyományos track-to-track algoritmusnál. Ez az objektum menedzsment által, az objektumok megerősítésének késleltetése végett van. Viszont az is jól látszik, hogy a visszacsatolás és az objektum menedzsment miatt a precision sokkal magasabb, így a track-to-track fúzió hibás asszociációit kijavítva. Az F1 score esetében a hagyományos track-to-track fúzió ront a szenzorok teljesítményén, míg az általunk javasolt megoldás viszont javít rajtuk. Labeled GOSPA ezen összehasonlítás esetében nem került kiértékelésre, hisz az összehasonlítás alapját képező fúziós algoritmus esetén nincs Multi-Object Tracking, ezáltal az egyes track-ekhez nincs rendelve egyedi ID, így azok folytonosságát is értelmetlen lenne vizsgálni. Ez az oka annak, hogy az 5.7. ábrán mindegyik track ugyanolyan színű, míg a visszacsatolást tartalmazó track-to-track fúzió esetében (5.8. ábra) minden egyes track-hez egyedi szín tartozik. Ezen ábrák még jól szemléltetik, hogy a hagyományos track-to-track fúzió esetén több a fals érzékelés, mint a visszacsatolást tartalmazó track-to-track fúzió-nál. Nevezetesen [-3, 13] környékén, mely utóbbi algoritmusnál is megjelenik, ezen kívül $\boldsymbol{x}$ dimenzió szerinti 0-tól 5ig, $\boldsymbol{y}$ szerint pedig 13 környékén szintén van egy fals megerősítés. Azonban az is látszik, hogy [1, 20] környékén ezen megoldás leköveti az egymáshoz nagyon közeli track-eket külön-külön, míg az általunk javasolt eljárás egybeolvasztja ezen (4-es, lila színű trajektória és 5-ös, zöld trajektória) track-eket. Ennek oka a duplikáció kezelés, hisz ez a két track annyira közel helyezkedik el egymáshoz, hogy az algoritmus duplikációnak érzékeli.



5.9. Ábra: Fúziós algoritmusok teljesítménye az idő függvényében

Az 5.9. ábrán jól látható, hogy a track-to-track fúzió esetében jóval több a fals detekció, míg a visszacsatolást tartalmazó track-to-track fúziónál a lokalizációs hiba jelentősebb, melynek oka maga a visszacsatolás, hisz ez késlelteti az állapotmeghatározást, de ez a hiba kezelhető utólagos Delay compensation-nel. A track ID váltások esetén az általunk alkalmazott módszer értékelése releváns csupán, hisz a track-to-track fúzió esetén nincsenek track ID-k, így érthető módon azok cserélgetésére sem kerül sor.

6. Konklúzió

Az eredmények alapján egyértelműen kijelenthető, hogy egy olyan objektumkövető algoritmuson alapuló szenzorfúziós architektúrát sikerült fejleszteni, melynek futásideje alacsonyabb, mint a hagyományos algoritmusoké.

A klasszikus objektumkövető (tracking) algoritmusokból kiindulva, egy priorizálási struktúra felépítésével ezen megközelítések futási idejét redukáltuk, miközben hatásfokuk javult a klasszikus eljárásokhoz képest. Ebből az ötletből kiindulva két különböző, egy detekció priorizált és egy objektum priorizált Local Nearest Neighbor algoritmust fejlesztettünk, melyek közül utóbbi sokkal hatékonyabbnak bizonyult. Ezen fejlesztésnek köszönhetően javult az objektum lokalizációs képesség, így egy pontosabb trajektória képet kaphatunk a nyomon követett track-ekről. Emellett fals detekciók jelentős szűrésére is sor került, melynek köszönhetően a tracking folyamatosabb, pontosabb lehet. Továbbá javult a track ID folytonosság is, így azon problémákra megoldást kínálva, melyek esetén több objektum azonos időben való keresztezése miatt felcserélődhetnek a track ID-k és emiatt egy kevésbé biztonságos környezetérzékelés valósulhat meg.

A track-to-track fúziós logika felépítésénél fontos kérdés, hogy hogyan lehet a különböző szenzorokból érkező track-eket oly módon összerendelni, hogy számtáskapacitási igényt spóroljunk a teljesítmény növelése mellett. A hagyományos track-to-track fúziónak egy, a fuzionált, kimeneti trackek, az asszociációs blokkba való visszacsatolása és azok állapotát felhasználva egy olyan hibrid track-to-track architektúrát kaphatunk, mely megfelel az imént felsorolt követelményeknek. Ennek köszönhetően, felgyorsítható a jármű automatizáltság növekedése a költséghatékonyság megőrzése mellett. Továbbá megfelelően kiszámított, track-pár specifikus threshold értékeknek köszönhetően pontosabban végezhető a különböző szenzoroktól származó trackek asszociációja egy clustering algoritmusnak köszönhetően. Emellett egy előzőtől különböző, szintén track-pár specifikus threshold érték meghatározásával képesek vagyunk megmondani a szenzoroktól érkező esetleges track-duplikációk létezését, ezzel is pontosítva az asszociációt és ezáltal megbízhatóbb környezetérzékelést kapunk eredményül. Továbbá ezen fúziós logika nem érzékeny a szenzorklaszterek változására, így összességében egy moduláris architektúrát kaphatunk eredményül.

A szenzor- és a fuzionált adatokon végzett, továbbfejlesztett tracking és fúziós algoritmusoknak köszönhetően a teljes környezetérzékelésre nézve magas konfidenciával bíró objektumfelismerést sikerült megvalósítani, amely megközelíti az alacsony szintű szenzorfúzió teljesítményét, miközben a számításkapacitási igény jelentős mértékben csökkentve lett, így a módszer valós idejűleg alkalmazható maradhat.

Irodalomjegyzék

[1] J. van Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transp Res Part C Emerg Technol*, vol. 89, pp. 384–406, Apr. 2018, doi: 10.1016/J.TRC.2018.02.012.

[2] T. Tettamanti, I. Varga, and Z. Szalay, "Impacts of Autonomous Cars from a Traffic Engineering Perspective," *Periodica Polytechnica Transportation Engineering*, vol. 44, no. 4, pp. 244–250, Oct. 2016, doi: 10.3311/PPTR.9464.

[3] W. Gruel and J. M. Stanford, "Assessing the Long-term Effects of Autonomous Vehicles: A Speculative Approach," in *Transportation Research Procedia*, 2016, vol. 13, pp. 18–29. doi: 10.1016/j.trpro.2016.05.003.

[4] American Automobile Association, "ADVANCED DRIVER ASSISTANCE TECHNOLOGY NAMES AAA's recommendation for common naming of advanced safety systems," 2019, Accessed: May 02, 2022. [Online]. Available: www.sbdautomotive.com

[5] K. Rana and P. Kaur, "Comparative study of automotive sensor technologies used for unmanned driving," *Proceedings of 2nd International Conference on Computation, Automation and Knowledge Management, ICCAKM 2021*, pp. 346–350, Jan. 2021, doi: 10.1109/ICCAKM50778.2021.9357731.

[6] J. Vargas, S. Alsweiss, O. Toker, R. Razdan, and J. Santos, "An Overview of Autonomous Vehicles Sensors and Their Vulnerability to Weather Conditions," *Sensors 2021, Vol. 21, Page 5397*, vol. 21, no. 16, p. 5397, Aug. 2021, doi: 10.3390/S21165397.

[7] W. Elmenreich, "An Introduction to Sensor Fusion," 2002, Accessed: May 02, 2022. [Online]. Available: https://www.researchgate.net/publication/267771481

[8] E. F. Nakamura, A. A. F. Loureiro, and A. C. Frery, "Information fusion for wireless sensor networks," *ACM Computing Surveys (CSUR)*, vol. 39, no. 3, p. 55, Sep. 2007, doi: 10.1145/1267070.1267073.

[9] D. L. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 6–23, 1997, doi: 10.1109/5.554205.

[10] C. Otto, "Fusion of Data from Heterogeneous Sensors with Distributed Fields of View and Situation Evaluation for Advanced Driver Assistance Systems," *Fusion of Data from Heterogeneous Sensors with Distributed Fields of View and Situation Evaluation for Advanced Driver Assistance Systems*, pp. 1–238, Sep. 2013, doi: 10.5445/KSP/1000035932.

[11] A. Houenou, P. Bonnifait, V. Cherfaoui, V. A. Cherfaoui, V. Cherfaoui, and J.-F. Boissou, "A Track-To-Track Association Method for Automotive Perception Systems," pp. 704–710, 2012, doi: 10.1109/IVS.2012.6232261ï.

[12] Dimitri J. Papageorgiou and Michael Holender, *Track-to-track association and ambiguity management in the presence of sensor bias*. [IEEE], 2009.

[13] Y. Bar-Shalom and H. Chen, "Multisensor track-to-track association for tracks with dependent errors," in *Proceedings of the IEEE Conference on Decision and Control*, 2004, vol. 3, pp. 2674–2679. doi: 10.1109/cdc.2004.1428864.

[14] N. Bhatia, “Survey of Nearest Neighbor Techniques,” *IJCSIS) International Journal of Computer Science and Information Security*, vol. 8, no. 2, 2010, Accessed: Oct. 14, 2022. [Online]. Available: http://sites.google.com/site/ijcsis/

[15] P. Konstantinova, A. Udvarev, and T. Semerdjiev, “A study of a target tracking algorithm using global nearest neighbor approach,” pp. 290–295, 2003, doi: 10.1145/973620.973668.

[16] V. Bäck, “Tracking Vehicles using Multiple Detections from a Monocular Camera”, Accessed: Oct. 14, 2022. [Online]. Available: http://www.teknat.uu.se/student

[17] J. Munkres, “Algorithms for the Assignment and Transportation Problems,” *Journal of the Society for Industrial and Applied Mathematics*.

[18] Jack Edmonds and Richard M. Karp, “Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems,” 1972.

[19] Y. Bar-Shalom and E. Tse, “Tracking in a cluttered environment with probabilistic data association,” *Automatica*, vol. 11, no. 5, pp. 451–460, Sep. 1975, doi: 10.1016/0005-1098(75)90021-7.

[20] F. Garcia, A. de La Escalera, and J. M. Armingol, “Joint Probabilistic Data Association fusion approach for pedestrian detection,” *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 1344–1349, 2013, doi: 10.1109/IVS.2013.6629653.

[21] D. Musicki, R. Evans, and S. Stankovic, “Integrated probabilistic data association,” *IEEE Trans Automat Contr*, vol. 39, no. 6, pp. 1237–1241, Jun. 1994, doi: 10.1109/9.293185.

[22] D. Mušicki and R. Evans, “Joint Integrated Probabilistic Data Association: JIPDA,” *IEEE Trans Aerosp Electron Syst*, vol. 40, no. 3, pp. 1093–1099, Jul. 2004, doi: 10.1109/TAES.2004.1337482.

[23] S. Matzka and R. Altendorfer, “A comparison of track-to-track fusion algorithms for automotive sensor fusion,” in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2008, pp. 189–194. doi: 10.1109/MFI.2008.4648063.

[24] A. S. Rahmathullah, A. F. Garcia-Fernandez, and L. Svensson, “Generalized optimal sub-pattern assignment metric,” *20th International Conference on Information Fusion, Fusion 2017 - Proceedings*, Aug. 2017, doi: 10.23919/ICIF.2017.8009645.

Ábrajegyzék