M Ű E G Y E T E M 1 7 8 2

Budapesti Műszaki és Gazdaságtudományi Egyetem
Közlekedésmérnöki és Járműmérnöki Kar
Közlekedés- és Járműirányítási Tanszék

Országos Tudományos Diákköri Konferencia
2024

Alacsony számításigényű, moduláris track-to-track szenzorfúzió autonóm járművek környezetérzékelésének támogatására

Készítette:

Czibere Balázs, járműmérnöki szak (BSc)

Témavezető:

Lindenmaier László

Tartalomjegyzék

1 Bevezetés **1**

1.1 Kutatás motivációja 1

1.2 Kutatás fókusza 2

2 Elméleti háttér **3**

2.1 Járműipari szenzorok 3

2.2 Szenzorfúzió 4

2.3 Track-to-Track fúzió 6

2.3.1 Adatasszociáció 8

2.3.2 Állapotbecslés 11

2.3.3 Objektum menedzsment 13

3 Saját fejlesztés **18**

3.1 Adatasszociáció 18

3.2 Fúzió 22

3.3 Track menedzsment 25

4 Kiértékelés **27**

4.1 A kiértékelési környezet 27

4.2 Kiértékelési módszertan 28

4.3 Eredmények 29

5 Konklúzió **31**

1 Bevezetés

1.1 Kutatás motivációja

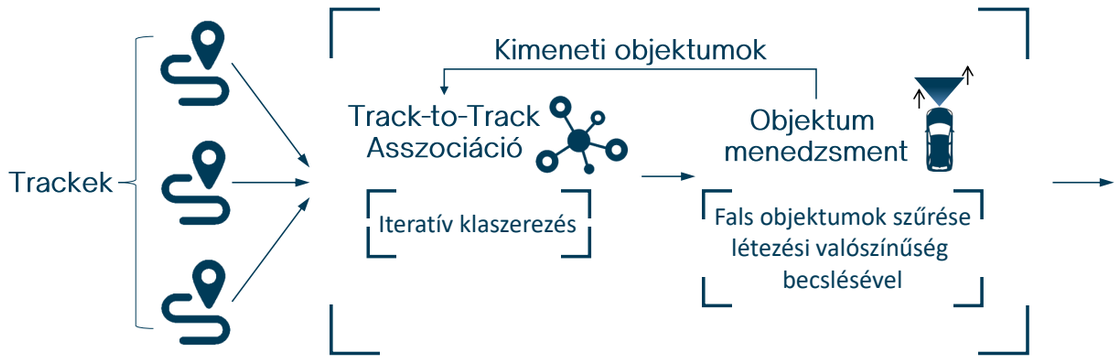
Napjainkban a vezetéstámogató rendszerek széleskörben elterjedtek [1]. Az önvezető autók társadalomra, környezetre és a gazdaságra gyakorolt hatásait folyamatosan fejlődő kutatási témák övezik [2][3]. Azoban a közlekedésbiztonság növelése kiemelkedő cél a közlekedés- és járműautomatizálás során. Ma már 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]. A járművekbe telepítve ezen új, szükségszerű, aktív biztonsági funkciók a járművezetőnek és egyéb közlekedésben résztvevőknek biztonságot nyújthatnak, így mérsékelve az emberi hibákat, mely az európai utakon bekövetkező balesetek 90%-át teszi ki. Ezzel együtt az automatizáltsági szint növelésével a teljes közlekedés gazdaságosabbá és fenntarthatóbbá tehető. Egyes funkciók akár önmagukban, vagy hálózatba kapcsolva az önvezető járművek tüzelőanyag fogyasztásának kiemelkedő csökkenését képesek elérni, ezzel akár 60%-kal is csökkentve az üvegházhatású és a környezetre káros egyéb anyagok kibocsátását. Másrészt pedig növelhető a gépjárművek kihasználtsága, ezáltal csökkentve a közlekedésben résztvevő járművek számát. Mindez normalizálhatja az általános energiafelhasználást is, így jelentős pozitív hatást gyakorolva a világ gazdaságára és a környezetre, hiszen a fejlett országokban a GDP jelentős hányadát a közlekedés érinti. További gazdasági előnyöket jelenthet a magas automatizáltság a haszonjárművek esetén, ugyanis a szállítmányozással és fuvarozással foglalkozó vállalatok jelentős mértékben csökkenthetik a kiadásaikat, akár részleges járműautomatizálással is.

A magasabb automatizáltsági szintek esetén a jármű dinamikus környezetének a felügyelete már nem az emberi vezető feladata, hanem az autonóm rendszer felelőssége, amely komoly kihívást jelent. Ezért a fejlett vezetéstámogató és önvezető rendszerekkel felszerelt járművek biztonságos üzemeltetéséhez egy megbízható és robusztus környezetérzékelést támogató algoritmus fejlesztése a kulcs. Az automatizáltsági szint növekedésével továbbá, a környezetérzékelés szerepe egyre fokozódik. Azonban, a rohamosan fejlődő szenzortechnológia ellenére napjainkban egyetlen járműipari szenzor sem képes kielégíteni a magas automatizáltság által a környezetérzékelés felé támasztott fokozódó igényeket [5], [6]. Az egyes szenzortípusok különböző előnyökkel és hátrányokkal bírnak. A járműipari radar jó távolság és sebesség becsléssel bír, míg az okos kamerák képesek az objektumok klasszifikációjára, illetve laterális pozíciók kellően jó becslésére. A megbízható környezetérzékeléshez ezért elengedhetetlen a különböző szenzoroktól származó adatok együttes felhasználása, fuzionálása, így egy megbízhatóbb eredményt elérve mintha bármelyik szenzort önmagában alkalmaztuk volna. A kutatás motivációja egy olyan moduláris architektúrával bíró szenzorfúziós algoritmus fejlesztése, amely rugalmasan adaptálható kölünböző szenzorklaszterekre. A fejlesztés során a valós-idejű alkalmazhatóságot figyelembe véve fontos szempont volt az algoritmusok számításkapacitás igénye. Ezzel elősegítve az újabb vezetéstámogató és autonóm rendszerek elterjedését.

1.2 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 (nyomon követett objektumok), melyek a szenzorfúziós logika bemeneti interfészén definiált, szenzorfüggetlen bemenetek. Az objektumfúzió egy előnyös megvalósítása a szenzorfúziónak, mely magas absztrakciós szintű, objektum információkat fuzionál. Ennek köszönhetően egy moduláris algoritmust kapunk, mely különböző szenzorokkal rendelkező rendszerekre könnyen adaptálható.

Kutatásom középpontjában a track-to-track architektúrájú objektum fúzió állt. Fő célom az volt, hogy a valós-idejű alkalmazhatóságot figyelembe véve egy alacsony számításigényű módszert fejlesszek, melynek teljesítménye megközelíti a state-of-the-art objektum fúziós módszerekét. Ezért a kutatás fókuszában a track-to-track asszociáció illetve az architektúra kimeneteként előálló fuzionált objektumok posztprocesszálása áll. Az 1.Ábra szemlélteti az elvárt rendszer architektúrát, bekeretezve az imént felsorolt fókuszterületeket.



1.1. Ábra: A Track-to-Track fúzió lépései

A track-to-track asszociációnál a feladat a kölönböző szenzoroktól érkező, ugyanahhoz az objektumhoz tartozó trackek párosítása. Ehhez egy iteratív klaszterező eljárást fejlesztettem, mely a különböző szenzorok által szolgáltatott, összetartozó trackeket rendeli össze, úgy hogy minden track párosításra kiszámításra kerül egy specifikus küszöbérték. Ezen érték alapján eldönthető, hogy a vizsgált trackek összetartoznak-e vagy sem. Ezen felül előfordulhat, hogy egy adott szenzor egy adott objektumról duplikált tracket szolgáltat. Hipotézisem, hogy egy másik, az előzőnél szigorúbb küszöbérték beállítható specifikusan minden track párosításnál, melynek segítségével kiszűrésre kerülnek ezen duplikátumok.

A számításigény mellett fontos szempont a kimenet megbízhatósága. Ebben a legfontosabb szerepe a szenzoroktól származó fals objektumok szűrésének volt. Ehhez kidolgoztam egy új eljárást a fuzionált objektumok létezési valószínűségének becslésére, mely hatékonyan támogatja a fals objektumok szűrését és a valódi objektumok megerősítését.

A hagyományos track-to-track asszociációhoz képest az általam fejlesztett eljárásban a fuzionált trackek állapotai multi-object-tracking architektúrát követve visszacsatolásra kerülnek az asszociációs blokkba, ezáltal megvalósítva egy magas teljesítménnyel, illetve kedvező komplexitással bíró algoritmust.

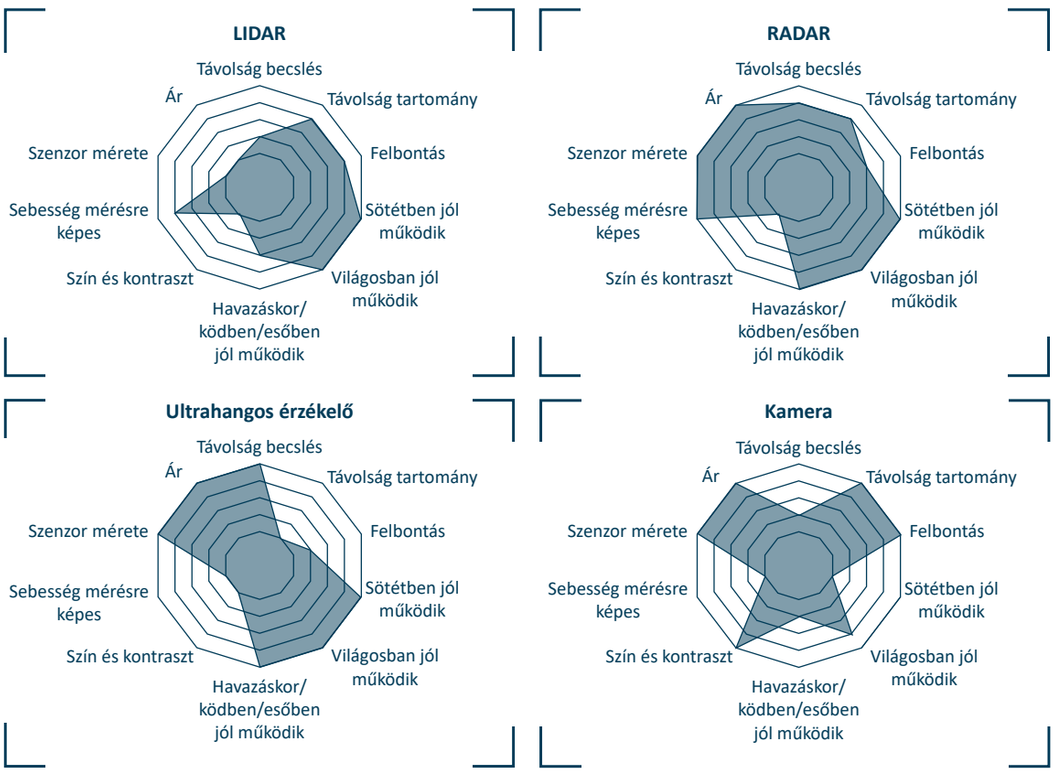
2 Elméleti háttér

2.1 Járműipari szenzorok

Magasan automatizált szintek eléréséhez különböző szenzorok kombinációjára van szükség, melyek megbízhatóan észlelik a jármű környezetét. Ezek közé tartozik például a RADAR, LIDAR, ultrahangos érzékelők, illetve kamerák. Azonban minden szenzortípusnak meg van az előnye, illetve hátránya [7].

A RADAR (Radio Detection and Ranging, magyarul rádióérzékelés és távmérés) rádióhullámok segítségével végzi az objektum detektálást. Amikor a kibocsátott hullámok a terjedési útvonaluk mentén egy tárgyat elérnek, akkor annak felületéről visszaverődnek. Ekkor a RADAR-antenna a visszaszórt jelet (visszhangot) a látómezejében (FOV) összegyűjti, így meghatározva az adott objektum pozícióját, illetve sebességét. Az intelligens járművek fejlődése szükségessé tette ezen szenzor használatát, a biztonság javítása érdekében. Adaptív sebességtartó automatika (ACC), ütközéselkerülő rendszerek (CAS), holttér-érzékelő (BSD) és sávváltás-asszisztens (LCA) csupán néhány példa.

A LIDAR (Light Detection and Ranging, magyarul lézer alapú távérzékelés) rendszer működése, egy lézerdiódából kibocsátott infravörös (vagy közel infravörös) tartományban levő fényimpulzus adott távolság megtételéhez szükséges időmérésén alapul (Time-of-Flight, ToF). A ToF-technológiában a LiDAR egy meghatározott időtartamú (τ) fényimpulzust generál, amely, a kibocsátás pillanatában egy időzítő áramkör belső óráját indítja el. Az objektumról visszaverődő fényimpulzust egy fotodetektor érzékeli, amely elektromos kimenetet előállítva kikapcsolja az órát.



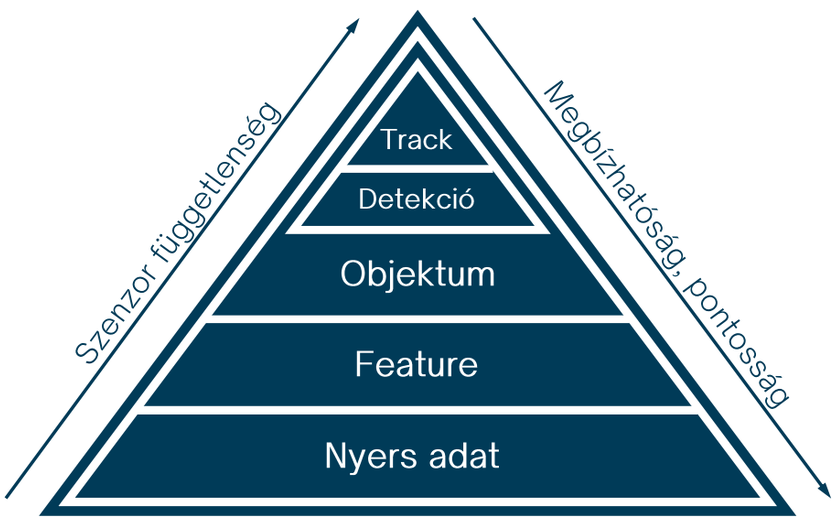
2.1. Ábra: Különböző szenzortípusok erősségei

Az ultrahangos érzékelők számos érzékelési feladatra alkalmasak az ipari alkalmazásokban.

Hangjeladókra támaszkodnak, amelyek $40 - 70kHz$-ig terjedő frekvencia tartományban továbbítják a hanghullámokat.
Az önvezető járművek a környezet érzékelésében nagymértékben támaszkodhatnak a kamerákra. Leginkább alacsony költségük, nagy felbontásuk és szín megkülönböztető képességük miatt használják őket. Azonban a távolság becslési pontosságuk alacsonyabb, mint a RADAR vagy LIDAR esetében.

2.2 Szenzorfúzió

A járműipari szenzorfúzió célja, hogy az elérhető összes szenzoradat felhasználásával, és annak ismeretében, hogy az egyes szenzortípusok milyen erősséggel, illetve hátránnyal bírnak, a fúzió által felépített egységes környezetmodell megbízható és robusztusabb legyen mintha bármelyik szenzort önmagában alkalmaztuk volna. Kikompenzálva, olyan gyakorta előforduló szenzorhibákat, mint a kimaradó detekciók és a fals érzékelések és csökkentve a kitettséget a különböző környezeti viszontagságoknak, mint például a fényviszonyok illetve az időjárási viszonyok. Továbbá szükséges a zajjal terhelt szenzormérések zajszűrése is, mely szintén szenzortípus függő.
A szenzorfúzió architektúráját és a kimenet megbízhatóságát jelentősen befolyásolja a szenzorfúzió szintje, amely bemeneti adatoknak az absztrakcióját írja le. A járműipari szenzorfúzióban 3 fő szintet különböztetünk meg általában, a nyers adat, a feature, illetve az objektum szintű szenzorfúziót. Az objektum fúzió a környezetérzékelésben alkalmazott szenzor adatfúziónak egy olyan speciális szintű fúziója, melyben objektumnak feltételezett, numerikus paraméterekkel jellemzett absztrakt méréseket fuzionálunk, így megvalósítva egy egységes környezeti modellt. Ezen belül két architektúrát, a detekció szintű és a track-to-track fúziót különböztethetünk meg. 2.2.Ábra

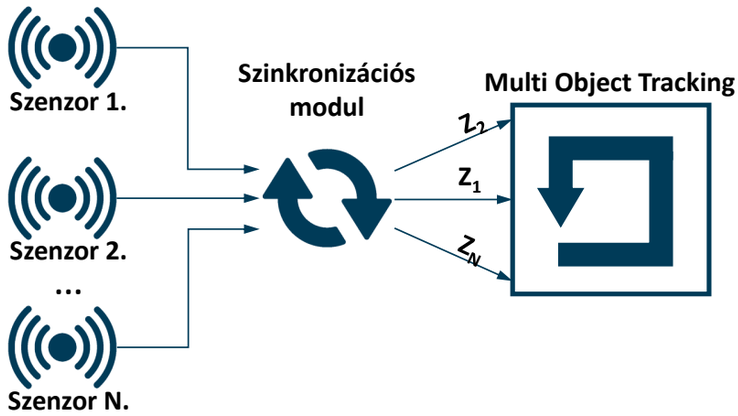


2.2. Ábra: Szenzorfúziós szintek

Minél magasabb szinten történik meg a szenzofúzió, annál modulárisabb, szenzorfüggetlenebb algoritmust kapunk. Ezzel szemben az alacsonyabb absztrakciós szintű szenzorfúziónak a megbízhatósága és pontossága kedvezőbb mint a magasabb szintű szenzorfúzióé, de általában neurális hálókkal valósítják meg őket ezért ezeknek az algoritmusoknak a tanító adatigénye magas. További hátrányuk még, hogy az algoritmus kizárólag azon szenzorokra fog működni, melyekre a

háló tanítva lett, így, mint az ábra is mutatja nem szenzorfüggetlenek, ellentétben az objektum szintű fúzióval. Feature szinten nem a nyers adatok szintjén történik a fúzió, hanem előbb úgynevezett *feature*-öket nyerünk ki a szenzoradatokból, mint például kamera esetén a BEV (Bird's-eye View) és ezen *feature*-ök kerülnek fuzionálásra, szintén neurális hálók alkalmazásával.
Azonban az architektúra választás sokszor nem választás kérdése ugyanis napjainkban az elérhető járműipari szenzorok többsége erősen előfeldolgozott, magas szintű adatokat, mint például objektum detekciókat, vagy akár trackelt objektumokat biztosítanak.

Okos kamerák az objektumokról pozíciót, míg a járműipari radarok sebességeket is képesek becsülni. Emiatt a két szenzor által szolgáltatott adatok nem hasonlíthatóak össze. Ennek a problémának a feloldását a detekció szintű architektúrában egy rekurzív modul oldja meg. Ezt a 2.3.Ábra is szemlélteti.



2.3. Ábra: Központi szintű architektúra

A nyers szenzor adatok bekerülnek a szinkronizációs modulba, mely meghatározott logika alapján időzítve továbbítja az adatokat, egymás után egy MOT algoritmusnak. Az algoritmus minden egyes szenzoradattal frissíti a közös objektumlistát. Miután lefutott egy teljes ciklus, akkor kimenetként megjelenik a fuzionált objektumlista. Ezen objektumlista minden trackje ugyanazzal az állapotvektorral rendelkezik, melyet (2.1) egyenlet is reprezentál.

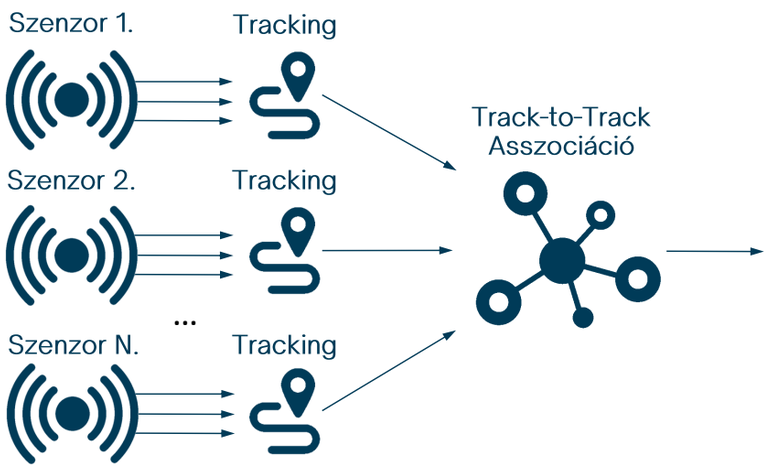
$$x = \begin{bmatrix} d_x & v_x & d_y & v_y \end{bmatrix}^\top, \tag{2.1}$$

Alacsony szintű fúzió relevanciáját a nyers szenzoradatok előfeldolgozatlansága adja, mivel minden egyes előfeldolgozás információ vesztést eredményez. A nyers adatszintű fúzió azonban 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. Így a hagyományos, analitikus szenzorfúziós algoritmusok többsége esetén, a bemenetek fúziója, detekció szinten történik. Ez viszont olyan nehézségekhez vezethet, mint például az objektumszám bizonytalansága (cardinality problem). Azonban egy ilyen magasabb absztrakció kedvező jellemzője lehet, ha a nyers szenzor adatok előfeldolgozásakor objektumokat nyerünk ki a környezetből. Ezzel a lépéssel a szenzoradatok összehasonlíthatóvá válnak, melynek köszönhetően az algoritmusunk flexibilis lehet. A track-to-track fúzió egy jó példa erre, hisz ott minden bemenetet trackelünk, így minden egyes tracknek ugyanaz az $\vec{x}$ állapottere van,

melyet (2.1) egyenlet is mutat. Így az egyszerűbb és moduláris jellege miatt válhatott a track-to-track fúzió egyre elterjedtebbé.

2.3 Track-to-Track fúzió

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 2.4.Ábra is mutat.

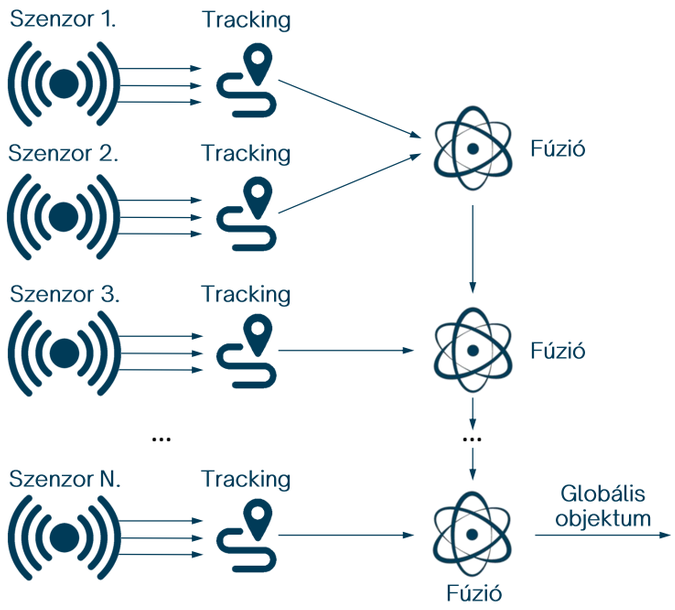


2.4. Ábra: Track-to-Track szenzorfúzió

Azon szenzoroknál, melyek nem adnak közvetlen információt bizonyos állapotokról (például kamera esetében a sebességek), ott ezen hiányzó adatok becslés eredményeként adódnak. Ennek következtében a kovariancia mátrixban szereplő értékek magasabbak lesznek, mint azon szenzorok esetén, melyek viszont szolgáltatnak közvetlen információt ezen állapotokról. Így az összes szenzor összes adata beküldhető egy közös halmazba. Ez az architektúra, a detekció szintűnél egyszerűbb, hisz nem igényel adatszinkronizációt, kisebb a számítási igénye és minden track egy állapottéren van. Azonban a track ID folytonosság megőrzése érdekében a fúziós architektúra végére szükséges egy tracking algoritmus kapcsolása, mint ahogy az a 2.3.Ábrán is látszik.

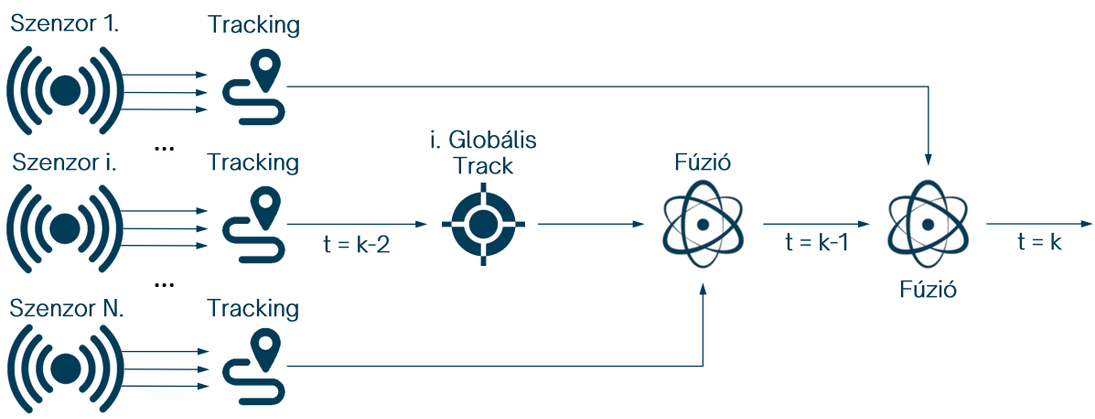
A Track-to-Track asszociációs eljárások alapvetően két nagy csoportra bonthatók [8]. Egyik a Sensor-to-Sensor architektúra, melyben szenzor szintű objektumok kerülnek fuzionálásra. Azonban a szenzorok nem szinkronizáltak, így igen csekély az esély arra vonatkozóan, hogy A és B szenzor ugyanabban az időpillanatban szolgáltasson tracket egy adott objektumról, hogy aztán megtörténhessen a fúzió. Ezért, Amint az egyik szenzor, például A küld egy tracket a $k-1$. időpillanatban, viszont B nem, akkor A szenzor trackjét prediktáljuk a következő, k. időpillanatra. Abban az esetben, ha B szenzor k. időpillanatban már szolgáltat adatot, akkor megtörténhet a fúzió A szenzor prediktált trackje és a B szenzor friss trackje közt, így létrehozva az úgynevezett globális objektumot. Abban az esetben, ha több mint két szenzor van a konfigurációban, akkor a fuzionálás szekvenciálisan történik. Először A és B szenzor trackek közt, majd ezen fuzionált track a következő, C szenzor trackjével, és így tovább. Érdemes megjegyezni, hogy ezen fúziós stratégiánál, a kimeneten megjelenő globális objektumok a k. időpillanatban függetlenek az összes

előző időpillanatban szereplő globális objektumoktól. Tehát minden egyes iterációban, teljesen új globális objektumok jelenek meg a fúzió kimenetén.



2.5. Ábra: Sensor-to-Sensor fúziós architektúra

Egy másik eljárás a Sensor-to-Global fúzió. Lényege, hogy a globális trackek minden iterációban frissítésre kerülnek, valamely szenzor által küldött trackel. $k-1$. időpillanatban legenerálásra kerül egy globális track, melynek állapota aztán prediktálásra kerül a következő, k. időpillanatra. Ebben az időpillanatban, az egyik szenzor (legyen ez A vagy B), ha az adott globális objektumhoz tartozó tracket küld, frissíti ezen prediktált, globális objektum állapotát. Több előnye is van ennek az eljárásnak. Először, hogy a szenzor adatok feldolgozásra kerülnek abban a pillanatban, hogy megjelennek, így megvalósítva a fúziót. Másodszor, hogy egy új szenzoradat megjelenésével a korábbi időpillanatokhoz tartozó globális objektum állapotok nem vesznek el. Harmadszor pedig, hogy moduláris, így különböző szenzor konfigurációkra is alkalmazható.



2.6. Ábra: Sensor-to-Global fúziós architektúra

Kutatásom során egy olyan magas track szintű fúzió megvalósítását tűztem ki célul, melynek megbízhatósága megközelítheti egy feature, vagy esetleg egy nyers adat szintű szenzorfúzióét, miközben figyelembe vettem az algoritmus futási idejét a valós idejű alkalmazhatóság érdekében.

2.3.1 Adatasszociáció

A Track-to-Track asszociáció feladata, hogy valamilyen szabály szerint összerendelje azon szenzor-szintű trackeket, melyek ugyanattól a valós objektumtól származnak. Ez kulcsfontosságú eleme a szenzorfúziónak. Ugyanakkor bonyolult is, ha figyelembe vesszük a különböző szenzorhibákat. Ilyenek lehetnek például a kimaradó detekciók vagy éppen a fals érzékelések. Az adatasszociáció jelenti a legfőbb különbséget a detekció szintű és a track-to-track fúzió közt.
A hagyományos track-to-track asszociációs probléma visszavezethető egy vegyes-értékű lineáris programozási feladatra, aminek a komplexitása NP osztályba tartozik, ami jelentősen bekorlátozza a valós idejű alkalmazhatóságot. Közkedvelt megközelítése a Track-to-Track asszociációnak a likelihood értékek számítása. Egy ilyen architektúra fejlesztésével foglalkozott Dimitri J. Papageorgiou is [9].
Az asszociációt számos tényező befolyásolhatja. Először, a különböző elven működő szenzortípusok eltérő módon követhetik az objektumokat. Így előfordulhat, hogy egy adott valós objektumot az egyik szenzor trackel, míg a másik nem. Vagy épp az egyik szenzor több fals-pozitív objektumot küld, mint egy másik. Másodszor, egymáshoz közel elhelyezkedő objektumok követését nehéz megoldani az egyes szenzorokra jellemző mérési bizonytalanságok miatt. Továbbá harmadszor, egyéb szenzorhibákból származó torzítások is előfordulhatnak, melyeket a mérési tengelyek illetve szenzorpozíciók pontatlan beállítása okozhatja.

Legyen két különböző szenzor A és B, melyek mindegyike ismeretlen számú objektumot követ a térben. N_A és N_B jelölje A és B szenzorok trackekjeinek a halmazát. Továbbá legyen $\mathbf{x}_i^A$ és $\mathbf{P}_i$, ahol $i \in N_A$, az A szenzor i track állapotbecslésének a vektora, illetve kovariancia mátrixa. Hasonlóan a B szenzor j trackjére, legyen $\mathbf{x}_j^B$, illetve $\mathbf{Q}_j$, ahol $i \in N_A$. A szenzorok egymáshoz viszonyított kalibrációs hibáját (jelölje $\mathbf{b}$) modellezze egy Gauss-féle eloszlásgörbe, mely $\mathbf{0}$ várható értékkel, és $\mathbf{R}$ kovarianciával rendelkezik.
Az asszociációt N_A i trackje és N_B j trackje közt jelölje (i, j_i). Továbbá jelölje a Track-to-Track asszociációt $\mathbf{j} = \{(1, j_1), (2, j_3), ..., (n_A, j_{nA}), \}$ vektor. Előfordulhat, hogy például az A szenzor i trackje nem tartozik egyetlen B szenzor trackhez sem, akkor (i, j_i) jelölés továbbra is használható, de $j_i = 0$.
Jelölje a következő egyenlet a Negative Log Likelihood (NLL) függvény egy módosított verzióját:

$$NLL(\mathbf{b}, \mathbf{j}) = \mathbf{b}^T \mathbf{R}^{-1} \mathbf{b} + \sum_{i=1}^{n_A} \begin{cases} {d_{ij_i}}^2 + log(|\mathbf{S}_{ij_i}|), & \text{ha } j_i > 0 \\ g, & \text{ha } j_i = 0 \end{cases}, \tag{2.2}$$

ahol a küszöbérték

$$g = -2log\left(\frac{\beta_{NTA}\beta_{NTB}(2\pi)^{D/2}}{\beta_T P_{AB}}\right), \tag{2.3}$$

illetve $\mathbf{S}_{ij} = P_i + Q_j$; és ${d_{ij}}^2(\mathbf{b}) = \left(\mathbf{x}_i^A - \mathbf{x}_j^B - \mathbf{b}\right)^T \mathbf{S}_{ij}^{-1} \left(\mathbf{x}_i^A - \mathbf{x}_j^B - \mathbf{b}\right)$ a négyzetes Mahalanobis távolság az i és j track között, $\mathbf{b}$ kalibrációs hiba becsléssel paraméterezve. Továbbá $\beta_{NTA} = \beta_T P_{\bar{A}B} + \beta_{FA}$ a nem létező objektumok sűrűsége A szenzorra nézve, ahol β_T az objektumok sűrűsége $P_{\bar{A}B}$ annak a valószínűsége, hogy egy adott objektumot követ B szenzor, de A nem, β_{FA} pedig A szenzor fals trackjeinek sűrűségét reprezentálja. Ugyanezek elmondhatóak B szenzorra is,

ahol $\beta_{NTB} = \beta_T P_{A\overline{B}} + \beta_{FB}$. P_{AB} azt a valószínűséget reprezentálja, hogy egy adott track mindkét szenzorra létezik, D pedig a vonatkozási rendszer dimenziója.

Természetesen már léteznek heurisztikus, iteratív megoldások is. Adam Houenou [10] is egy ilyennel foglalkozott. A fúzió a következő lépéseket követi: Először, leimplementálásra kerül a fogadott adatok szoftveres szinkronizációja, továbbá a fúzió mintavételezési frekvenciája definiálásra kerül, így minden egyes ciklusban az összes szenzor szolgáltat adatot a fúziós szintnek. A fúzió feladata, hogy ugyanarról a valós objektumról szolgáltatott trackeket asszociálja. Így a klaszterek száma megegyezik az objektumszámmal, hisz egy klaszterben levő trackek egyesítésre kerülnek. Azon trackek, melyek nem kerülnek hozzárendelésre egyetlen fuzionált trackhez sem, új fuzionált trackeket realizálnak. Azon fuzionált objektumok, melyekhez nem kerül hozzárendelésre egyetlen track sem, prediktálva lesznek minden iterációban, egészen addig, amíg egy track le nem frissíti az állapotukat. Ha ez bizonyos ciklusszámon belül nem történik meg, akkor ezen fuzionált trackek törlésre kerülnek a globális listáról. A fuzionált objektumok állapota a hozzájuk rendelt trackek állapotával frissülnek.
A Track-to-Track asszociáció a szenzoroktól származó összes track klaszterekbe rendezésével történik meg. Így a különböző klaszterekben szereplő trackek ugyanahhoz az adott objektumhoz tartozó állapotbecslések. Ezen Track-to-Track asszociációs algoritmus esetén két fontos kitétel szükséges:

1. egy klaszterben szenzoronként maximum egy track szerepelhet
2. egy adott track nem szerepelhet egynél több klaszterben.

Ezzel azt feltételezzük, hogy egy szenzor nem küldhet egy adott objektumról egynél több tracket, illetve, hogy kettő vagy több track nem tartozhat egyetlen objektumhoz. Ez egy ideális eset, hisz gyakorlatban előfordulhatnak duplikált trackek, szenzor, illetve filterezési hibákból kifolyólag.

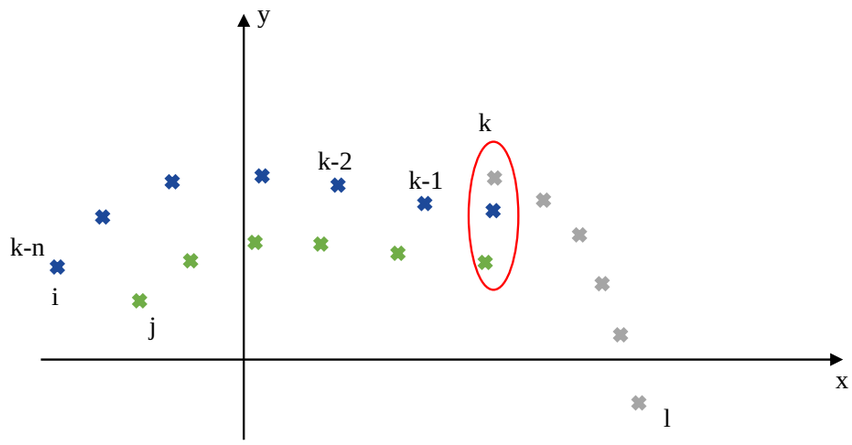
A trackek páronkénti távolság szamítása alapján határozhatóak meg a klaszterek. Ez önmagában csupán nem elég, tekintsünk a 2.7.Ábrára. Az ábrán látható i, j és l trackek trajektóriái, a $k-n$. időpillanattól a k. időpillanatig. Az is látható, hogy a k. időpillanatban az l track közelebb helyezkedik az i trackhez, mint j. Ha csupán a páronkénti kiszámított távolságok alapján rendelnénk össze a trackeket, akkor az l track kerülne összerendelésre i-vel. Azonban ha megvizsgáljuk a k előtti időpillanatokat, akkor tisztán látható, hogy j track sokkal hasonlóbb i-hez, mint l. Ezen probléma feloldásához felhasználásra kerülnek az egyes trackek korábbi időpillanatokhoz tarozó állapotai az asszociáció során. Így a "távolság" két track között:

$$D_k^{(i,j)} = \frac{1}{n} \sum_{m=0}^{n-1} d_{k-m}^{(i,j)}, \tag{2.4}$$

ahol n a korábbi állapotok száma, és

$$d_k^{(i,j)} = (x_k^i - x_k^j)^T \cdot (P_k^i + P_k^j)^{-1} \cdot (x_k^i - x_k^j) + ln(|P_k^i + P_k^j|), \tag{2.5}$$

ahol i és j két track, x_k és P_k a hozzájuk tartozó állapotvektor és kovariancia mátrix, a k. időpillanatban. n értéke az azon eltelt időpillanatok száma, melyekben a vizsgált trackek léteznek.



2.7. Ábra: Korábbi időpillanatokhoz tartozó track állapotok felhasználása Track-to-Track asszociációnál

Ennek az összefüggésnek köszönhetően a távolság l illetve i track között nagyobb lesz, mint j és i közt. Így az asszociációs probléma elhárult a k. időpillanatban.

Az, hogy a minimális távolság értékek iteratív módon kerülnek megtalálásra garantálja, hogy minden egyes iterációban a legjobb asszociáció valósuljon meg. Minél pontosabbak a szenzorok, annál közelebb helyezkednek el a különböző szenzor trackek a valós objektumhoz. Így az előre definiált küszöbérték lehet minimális. Azonban ha a szenzorok pontatlanok, akkor a trackek is távol helyezkedhetnek el egymástól. Ez esetben a küszöbérték növelésére van szükség. Az n előzmény puffer a második paramétere az algoritmusnak, melyet hangolni kell. Ha értéke nagy, akkor csökken az asszociációs hibák esélye, hiszen több múltbeli track állapot áll rendelkezésre. Azonban ez esetben a számításkapacitási igény nem kívánt mértékben megnövekedhet.

2.3.2 Állapotbecslés

Track-To-Track fúzió esetén azon objektumok kerülhetnek fúzionálásra, melyek állapotai ugyanabban az időpillanatban értelmezettek. Azonban a szenzoradatok nem szinkronizálva érkeznek a fúziós modulba. Különböző szinkronizációs eljárások azonban már léteznek [11]. Az adatok szinkronizálását követően azok fúzionálására kerül sor a kereszt-kovariancia vagy kovariancia metszet vagy kovariancia unió megközelítés alapján.

Kereszt-kovariancia

Legyen két állapotbecslés $\hat{x}_i$ illetve $\hat{x}_j$ és a hozzájuk tartozó kovariancia mátrixok P_i és P_j. Így a fúzionált állapot kereszt-kovarianciát használva:

$$\hat{x}_F = \hat{x}_i + P[\hat{x}_j - \hat{x}_i], \tag{2.6}$$

ahol

$$P = [P_i - P_j]U_{ij}^{-1} \tag{2.7}$$

$$U_{ij} = P_i + P_j - P_{ij} - P_{ij}^T \tag{2.8}$$

A kovariancia mátrix továbbá:

$$P_F = P_i - [P_i - P_{ij}]U_{ij}^{-1}[P_i - P_ij]^T \tag{2.9}$$

A kereszt-kovariancia mátrix P_{ij} kezdeti értéke $P_{ij}(0|0)$ és utólagosan frissítésre kerül

$$P_{ij}(k|k) = \alpha_i(k)\beta(k-1)\alpha_j^T(k), \tag{2.10}$$

ahol

$$\alpha_i(k) = I - K_i(k)H_i(k), \alpha_j(k) = I - K_j(k)H_j(k)$$

$$\beta(k-1) = F_iP_{ij}(k-1|k-1)F_j^T + Q(k-1)$$

Azonban (2.10) egyenletben szereplő α_i és α_j ismert, hisz a szenzorok Kálmán-szűrőjére vonatkozó H, K, F és Q is ismert. Azonban ez nem mindig van így, különösen ipari szenzorok esetén, melyek beépített tracking algoritmussal rendelkeznek. Ekkor egy alternatív megközelítés szükséges a P_{ij} számítására, ahol a kereszt-kovariancia mátrix a bemenő kovariancia mátrixok Hadamard-szorzataként áll elő.

$$P_{ij} = \rho\sqrt{P_i \circ P_j}, \tag{2.11}$$

ahol ρ az effektív korrelációs együttható, melynek $\rho \approx 0,4$ értéke egy jó megközelítés általánosan használt tracking algoritmusoknál.

Kovariancia metszet

Legyenek $I = \{\hat{x}_i, P_i\}$ illetve $J = \{\hat{x}_j, P_j\}$ állapotok, melyekkel meghatározható a F fúzionált állapot.

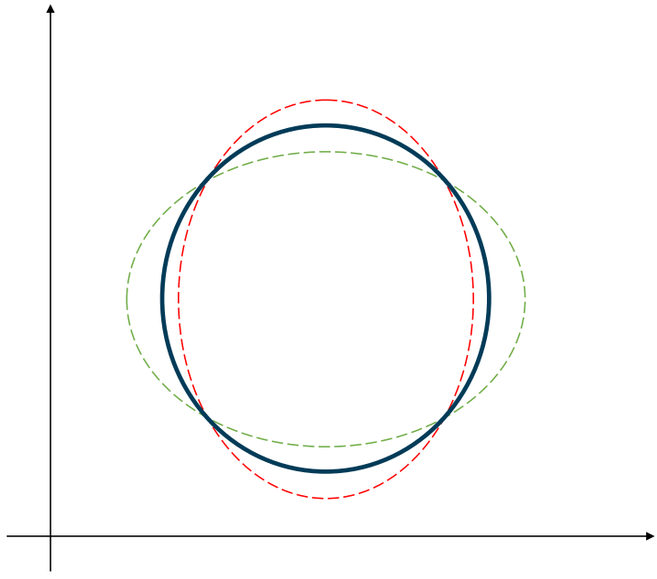
$$P_F^{-1} = \omega P_i^{-1} + (1-\omega)P_j^{-1} \tag{2.12}$$

$$\hat{x}_F = P_F \left(\omega P_i^{-1}\hat{x}_i + (1-\omega)P_j^{-1}\hat{x}_j\right), \tag{2.13}$$

ahol

$$\omega = \arg\min(\det(P_F)) \tag{2.14}$$

ω számítását [12] mutatja be. Minden $\omega \in [0,1]$-re a becslés konzisztens marad. Ezen becslés optimális, ha a Kereszt-kovariancia ismeretlen.



2.8. Ábra: *A* track (piros), *B* track (zöld) és a fuzionált track (kék) kovarianciája

Kovariancia Unió

A Kovariancia Unió eljárást [13] mutatja be. Megengedi két track egyesítését, akkor is, ha az állapotbecslések várható értéke nagy mértékű különbsége alapján a valóságban összetartozó két track inkonzisztenciát mutat. Például, ha két pozíció becslés várható értéke $1km$ eltérést mutat, de közben az összetartozó kovarianciák feltételezik, hogy egyes várható értékek pontosak $1m$-en belül, akkor egyértelmű, hogy valamilyen probléma van. Ahhoz, hogy meghatározzuk az egyes becslések közötti statisztikailag szignifikáns eltéréseket a Mahalanobis távolságszámítás több mint alkalmas. Legyenek a korábbi példákból kiindulva $I = \{\hat{x}_i, P_i\}$ illetve $J = \{\hat{x}_j, P_j\}$ állapotok, így a Mahalanobis távolság a következő:

$$d(x_i, x_j; Q) = \sqrt{(\hat{x}_i - \hat{x}_j)^T (P_i + P_j)(\hat{x}_i - \hat{x}_j)}, \tag{2.15}$$

ahol $Q \in \mathbb{R}^2$ valószínűségi eloszlás adott. Ha a kovarianciák nagyok, akkor nem meglepő, ha x_i és x_j vektorok között nagy a különbség. Ekkor a Mahalanobis távolság alacsony. Azonban, ha a kovarianciák alacsonyak, akkor a várható értékek közti kis távolság ellenére is a Mahalanobis

távolság nagy lehet. A magas Mahalanobis távolság azt mutathatja meg, hogy az állapot becslések nem konzisztensek egymással. Ekkor, alkalmazástól függően beállítható egy küszöbérték, aminek segítségével eldönthető, hogy mekkora Mahalanobis távolságnál mondható biztosra az, hogy a két becslés valóban inkonzisztens. Inkonzisztenciák feloldására [14] mutat megoldást. A kovariancia metszet esetén a konzisztencia garantált, egészen addig, amíg az egyes állapot becslések, melyek fuzionálásra kerülnek, konzisztensek egymással. Az úgynevezett ”konfliktusmentesítés” esetén csupán az tudott, hogy vagy $I = \{\hat{x}_i, P_i\}$ vagy pedig $J = \{\hat{x}_j, P_j\}$ konzisztens állapot becslése a vizsgált objektumnak. Mivel ismeretlen, hogy melyik becslés hamis, így a becslések kombinálására csupán egy módszer van. Ehhez létre kell hozni, egy egyesített állapot becslést $U = \{\hat{x}_u, P_u\}$, mely konzisztens mindkét becsléssel.

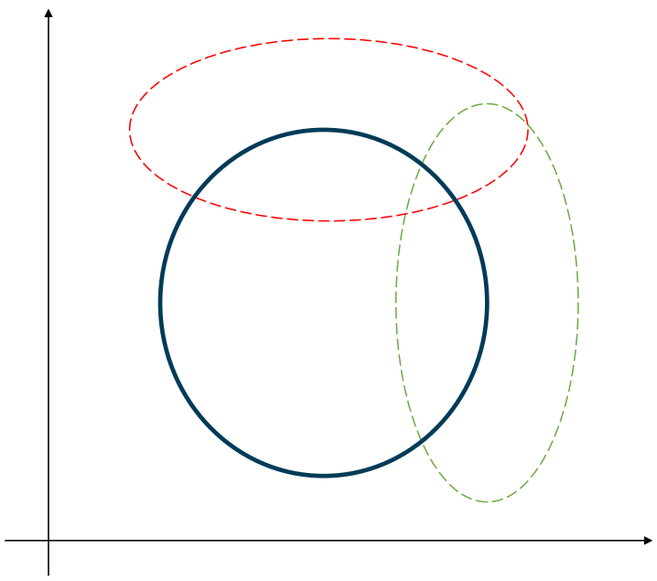
$$P_u^i \geq P_i + (\hat{x}_u - \hat{x}_i)(\hat{x}_u - \hat{x}_i)^T \tag{2.16}$$

$$P_u^j \geq P_j + (\hat{x}_u - \hat{x}_j)(\hat{x}_u - \hat{x}_j)^T \tag{2.17}$$

$$P_{ij} = \max(P_u^i, P_u^j) \tag{2.18}$$

$$\hat{x}_u = \arg\min(\det(P_{ij})) \tag{2.19}$$

Így meghatározásra kerül a $\hat{x}_u$ várható érték vektor és a hozzá tartozó legkisebb P_u kovariancia érték, mely elég nagy ahhoz, hogy garantálja a konzisztenciát attól függetlenül, hogy melyik állapotbecslés tartozik az objektumhoz.



2.9. Ábra: *A* track (piros), *B* track (zöld) és a fuzionált track (kék) kovarianciája

2.3.3 Objektum menedzsment

Az objektum menedzsment a Track-to-Track fúzió legnehezebb része. feladatai a következők:

1. új, a látómezőbe belépő objektumok inicializálása (objektum születés)
2. látómezőből kilépő objektumok törlése az objektum listáról (objetum törlés)

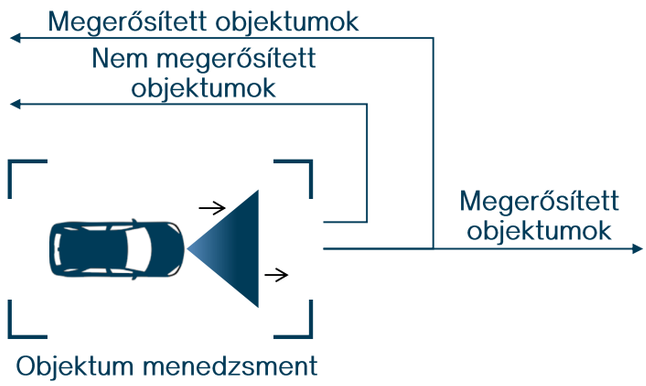
3. fals-pozitív objektumok szűrése

4. megerősített (confirmed) és a még meg nem erősített (tentative) objektumok elkülönítése és kezelése.

Vannak, akik ezen fontos feladattal nem foglalkoznak, hanem egyszerűen minden fuzionált és nem fuzionált tracket megerősítettnek tekintenek [10]. Azonban ez nem célravezető, hisz ilyen módon gyakoriak lehetnek a fals-pozitív realizálódások. Továbbá a globális objektumlistán szerepelhetnek olyan objektumok, melyek lehetséges, hogy valósak de már huzamosabb ideje nem tartózkodnak a szenzorok látómezejében, így ezen objektumok állapotfrissítése minden iterációban megtörténik, mely nem kívánt számítási kapacitás igényhez vezethet. Ezért szükséges az objektumok szelektálása. Van aki track páronkénti likelihood értékeket számít a trackek kétértelműségének kezelésére [9].

$$L_{ij} = \frac{1}{L_N} \sum_{k=1:(i,j)\in T_k}^{r} L(\mathbf{j}^k) \tag{2.20}$$

Két főbb objektum menedzsment megközelítés létezik. Egyik a korábbi időpillanatokhoz tartozó objektum állapotokat használja (history-based) [15]. A másik pedig valószínűségi értékeken alapul (probabilistic) [8]. 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.



2.10. Ábra: Objektum menedzsment

History-based objektum menedzsment

A history-based objektum menedzsment esetén, egy tentative objektumot akkor és csak akkor tekintünk megerősítettnek, ha N darab ciklusból legalább M-szer detektált, ahol N és M előre beállított értékek. Ez a módszer elősegíti, hogy kiszűrjünk néhány fals objektumot, melyek pár ciklus erejéig léteznek. Minél magasabb N értéke annál hatékonyabban szűrjük a fals-pozitívokat, azonban nem lehet nagyon magas sem, hisz azzal késleltetjük az objektumok megerősítését. Ha M-et a maximumra állítjuk, 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 egyes valós objektumok nem kerülnek megerősítésre. Az objektumok törlése hasonló

módon történik, mint az objektumok megerősítése. Ha egy objektumot nem detektálunk P-szer R darab ciklusból, akkor az az objektum törlésre kerül.

Létezési valószínűség alapú objektum menedzsment

Léteznek olyan adatasszociációs módszerek, melyek nem egy az egybe kölcsönösen egyértelmű hozzárendelést valósítanak meg, hanem probabilisztikus alapúak. Ilyenek például a Probabilistic Data Association (PDA) [16, 17], Joint Probabilistic Data Association (JPDA) [18] algoritmusok, melyek egy jó becslést nyújtanak számunkra az ismert számú objektumok állapotáról. Valós alkalmazáskor az objektumszám ismeretlen és időben változó, ezért annak becslésére is szükség van. Erre kínál megoldást az Integrated Probabilistic Data Association (IPDA) [19] és a Joint Integrated Probabilistic Data Association (JIPDA) [20] filterek. Ezen megközelítések képesek becslést adni az objektumok létezési valószínűségeire (Probability of Existence) [21].

Integrated Probabilistic Data Association [19]

Minden időpillanatban, minden track leírható az állapotbecslésével $\hat{x}_{k|l}$, a kovariancia mátrixával $\hat{P}_{k|l}$ és a létezési valószínűségével $P\{x_k|Z^l\}$, ahol Z^l a detekciók halmaza az l. időpillanatig bezárólag ($l \leq k$). Az Egyes típusú Markov Lánc modell azzal a feltételezéssel él az objektum létezési valószínűségére vonatkozóan, hogy az objektum mindig detektálható egy P_D detektálási valószínűséggel, abban az esetben ha létezik.

$$P\{x_k|Z^{k-1}\} = \triangle_{11}^1 P\{x_{k-1}|Z^{k-1}\} + \triangle_{21}^1 P\{\overline{x}_{k-1}|Z^{k-1}\}, \tag{2.21}$$

ahol $P\{x_k|Z^{k-1}\}$ az objektum *a priori* létezési valószínűsége k. időpillanatban. $P\{x_{k-1}|Z^{k-1}\}$ pedig a $k-1$. időpillanathoz tartozó *a posteriori* létezési valószínűség. Az átmeneti valószínűségek a következőképpen vannak definiálva:

$$\begin{aligned} \triangle_{11}^1 &\triangleq P\{x_k|x_{k-1}\} \\ \triangle_{21}^1 &\triangleq P\{x_k|\overline{x}_{k-1}\} \end{aligned} \tag{2.22}$$

A felsőindexben szereplő [1] az Egyes típusú Markov Lánc modellt jelöli.

$$P_W = \int_{V_k} f(z|Z^{k-1})dV, \tag{2.23}$$

ahol $f(z|Z^{k-1})$ *a priori* valószínűségi sűrűség függvénye a prediktált detekció pozíciójára $\hat{z}_{k|k-1}$, P_W pedig annak a valószínűsége, hogy a detekció beleesik a prediktált állapot köré definiált területbe. Továbbá

$$p(z|Z^{k-1}) = \frac{1}{P_W} f(z|Z^{k-1}), \tag{2.24}$$

mely a *a priori* valószínűség sűrűség érték arra vonatkozóan, hogy a becsült detekció pozíció beleesik az előre definiált területbe. Ha feltételezzük, hogy a fals detekciók Poisson eloszlást követnek, akkor

a becsült számuk:

$$\hat{m}_k = \begin{cases} 0, & \text{ha } m_k = 0 \\ m_k - P_D P_W P\{x_k|Z^{k-1}\}, & \text{ha } m_k > 0, \end{cases} \tag{2.25}$$

továbbá

$$\delta_k = \begin{cases} P_D P_W, & \text{ha } m_k = 0 \\ P_D P_W - P_D P_W \dfrac{V_k}{\hat{m}_k} \sum_{i=1}^{m_k} p(z_{k,i}|Z^{k-1}), & \text{ha } m_k > 0, \end{cases} \tag{2.26}$$

így a létezési valószínűség a következő képp áll elő:

$$P\{x_k|Z^k\} = \frac{(1-\delta_k) \cdot P\{x_k|Z^{k-1}\}}{(1-\delta_k) \cdot P\{x_k|Z^{k-1}\} + (1 - P\{x_k|Z^{k-1}\})} \tag{2.27}$$

Dempster-Shafer elmélet

A Dempster-Shafer elmélet [22, 23, 24] meghatározza az egymást kizáró eseményeket egy adott rendszerben. A két legegyszerűbb, egymást kizáró eset a létezés $\exists$ és a nem létezés $\nexists$, így:

$$\theta = \{\exists, \nexists\}. \tag{2.28}$$

Továbbá

$$2^\theta = \{\varnothing, \{\exists\}, \{\nexists\}, \{\exists, \nexists\}\}, \tag{2.29}$$

ahol $\varnothing$ az üres halmaz, illetve $\{\exists, \nexists\}$ arra vonatokozó részhalmaz, hogy nincs információ az adott objektum létezésére vonatkozóan, tehát létezése ismeretlen. Aeberhard [25] ezen megközelítéssel élt az egyes objektumok létezési valószínűségeinek fuzionálására vonatkozóan. $m_A(\{\exists\})$ és $m_A(\{\nexists\})$ súlyok arra vonatkozóan, hogy az A szenzorra létezik, vagy nem létezik egy adott objektum. Továbbá $m_A(\{\exists, \nexists\})$ az objektum létezésére vonatkozó ismeretlen feltevés. Ezek ismeretében A és B szenzoroktól származó objektum létezési valószínűségének fúziója a Dempster-Shafer szabály értelmében:

$$m_F(\{\exists\}) = \frac{m^A(\{\exists\})m^B(\{\exists\}) + m^A(\{\exists\})m^B(\{\exists, \nexists\}) + m^A(\{\exists, \nexists\})m^B(\{\exists\})}{1 - [m^A(\{\exists\})m^B(\{\nexists\}) + m^A(\{\nexists\})m^B(\{\exists\})]}. \tag{2.30}$$

Ezen összefüggés használható Sensor-to-Sensor architektúrára. Azonban Aeberhard [8] alkalmazta ezt a megközelítést a Sensor-to-Global módszerre is:

$$m^G_{k|k}(\{\exists\}) = \frac{m^G_{k|k-1}(\{\exists\})m^A_k(\{\exists\}) + m^G_{k|k-1}(\{\exists\})m^A_k(\{\exists, \nexists\}) + m^G_{k|k-1}(\{\exists, \nexists\})m^A_k(\{\exists\})}{1 - [m^G_{k|k-1}(\{\exists\})m^A_k(\{\nexists\}) + m^G_{k|k-1}(\{\nexists\})m^A_k(\{\exists\})]}, \tag{2.31}$$

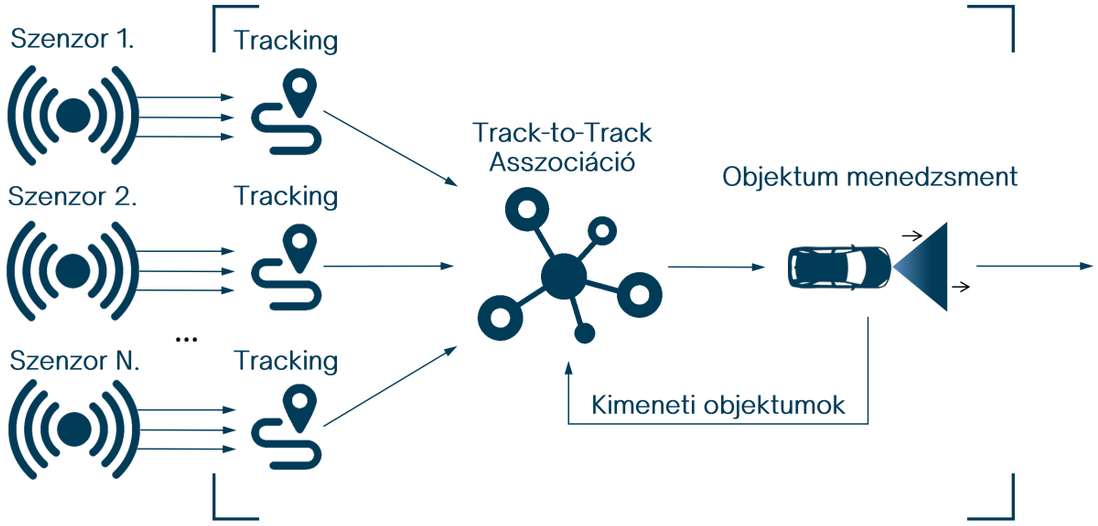
ahol $m^G_{k|k-1}(\{\exists\})$ a globális objektum $k-1$. időpillanatról, a k. időpillanatra prediktált létezési valószínűségi súlya. Hasonló módon $m^G_{k|k-1}(\{\nexists\})$ pedig a nem létezés, illetve $m^G_{k|k-1}(\{\exists, \nexists\})$ az ismeretlen állapot valószínűségi súlya.

Ezen probabilisztikus megközelítéseknél az objektumok megerősítése, illetve törlése két előre beállított érték alapján történik [8, 26]. Egyik ilyen érték a g_c (megerősítési küszöbérték), másik pedig a g_d (törlési küszöbérték). Ezen értékek beállítása heurisztikus. Egy adott objektum, csupán akkor kerül leinicializálásra, ha a létezési valószínűsége magasabb, mint az előre beállított törlési

küszöbérték. Abban az esetben, ha az objektum létezési valószínűsége eléri a g_c-t, akkor megerősítetté válik. Egyszer már megerősített objektum csupán akkor kerül törlésre az objektumlistáról, ha a létezési valószínűsége g_d alá csökken.

3 Saját fejlesztés

A korábban 2.3 fejezetben részletezett Track-to-Track architektúrájú objektumfúzió állt a kutatásom középpontjában.



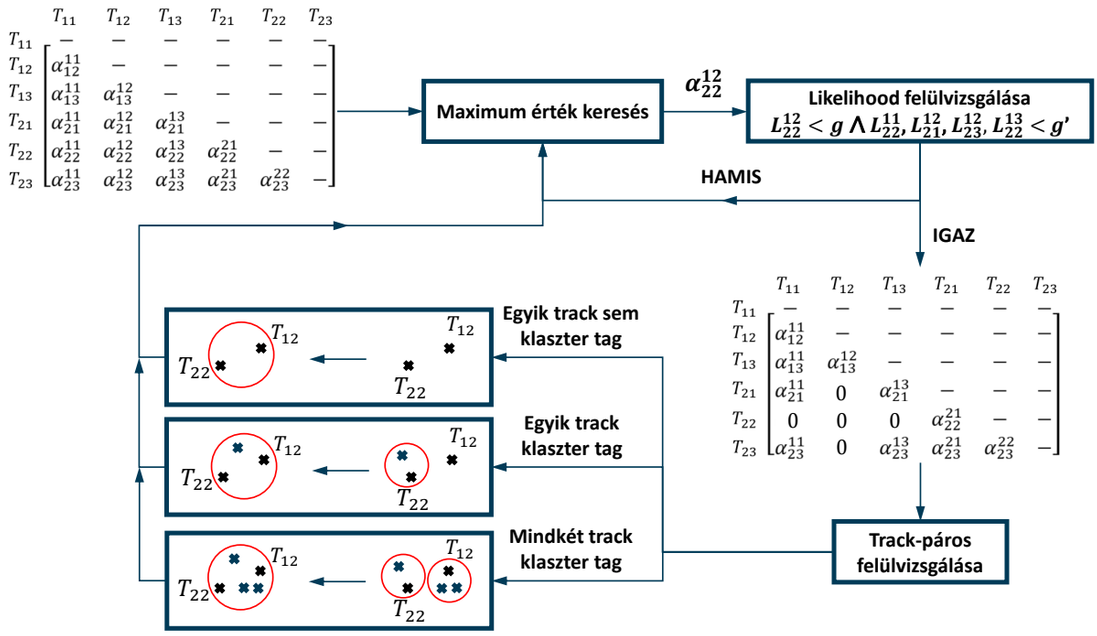
3.1. Ábra: Hibrid Track-to-Track architektúra visszacsatolással

Léteznek már olyan fúziós eljárások, melyeknél felhasználásra kerülnek a korábbi időpillanatokhoz tartozó globális trackek, ezáltal megvalósítva egy pontosabb és folytonosabb környezetérzékelést [27]. Mint azt a 3.1.Ábra is szemlélteti, az általam alkalmazott objektumfúziós architektúra az előző időpillanathoz tartozó kimeneti objektumok visszacsatolásra kerülnek a Track-to-Track asszociációs blokkba. Így, Multi Object Tracking architektúrát követve, ezen fuzionált trackek és a friss bemeneti adatok összerendelésre kerülnek. Ahhoz, hogy ez megtörténjen előbb a bemenetek összerendelésére van szükség, mely Track-to-Track asszociációval történik. Így összességében ezen visszacsatolás alkalmazásával egy hibrid Track-to-Track architektúrát kapunk eredményül, hisz a track asszociációnál a korábban (2.3 fejezetben) részletezett Sensor-to-Sensor eljárást alkalmazom, azonban a globális trackek is felhasználásra kerülnek, mely Sensor-to-Global módszert követ.

3.1 Adatasszociáció

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 korábban (2.3.1 fejezetben) részletezett Adam Houenou iteratív megközelítését [10] veszi alapjául. Azonban Ő erős feltételeket szab: Nevezetesen egy adott szenzor maximum egyetlen tracket realizálhat, továbbá egy adott T_{ij} track (ahol i a szenzor, j pedig a track ID) nem szerepelhet két különböző klaszterben. Így azt feltételezi, hogy egy adott szenzor nem generálhat egy adott objektumról egynél több tracket, mely egy teljesen ideális eset. Valóságban azonban könnyen előfordulhat, hogy több detekciót küld egy szenzor ugyanarról az objektumról. Így a kutatásom egyik fő célja az volt, hogy az általa fejlesztett klaszterező algoritmust úgy paraméterezhessem, hogy az esetleges track duplikátumokat is kezelni tudjam.

Mint ahogy azt az ábra is mutatja először létrehozunk egy $N \times N$-es mátrixot, ahol N az összes



3.2. Ábra: A klaszterezés folyamata, duplikátumok kezelésével

szenzortól beérkezett trackek száma. A mátrix elemei egyértelműen meghatározzák, hogy az egyes track-párosításoknak mekkora az asszociációs valószínűsége. Ezen asszociációs értékek számítását a következő összefüggés szemlélteti:

$$\alpha_{ij} = 1 - \frac{L_{ij}}{g_{ij}} > 0, \tag{3.1}$$

ahol L_{ij} a párosítás likelihoodja, g_{ij} pedig az i és j trackekre értelmezett specifikus küszöbérték. Ezen értékek számolását a 3.2 és 3.3 egyenlet szemlélteti.

$$L_{ij} = (x_i^A - x_j^B)^T (P_i^A + P_j^B)^{-1} (x_i^A - x_j^B) + ln(|P_i^A + P_j^B|) \tag{3.2}$$

$$g_{ij} = 2log\left(\frac{\beta_{FT} P_{TA} P_{TB}}{2\pi^{D/2} \beta_{NTA} \beta_{NTB}}\right) \tag{3.3}$$

β_{NTA} az A szenzor objektum sűrűség értéke arra nézve, hogy a B szenzor j trackjére nem létezik az A szenzornak trackje, továbbá megmutatja, hogy ezen track hogyan illeszkedik az X_{FT} (kimeneti objektumok) környezetre. β_{NTA} számítását a 3.4 egyenlet szemlélteti. β_{NTB} a B szenzor objektum sűrűség értéke arra nézve, hogy az A szenzor i trackjére nem létezik a B szenzornak trackje és mely β_{NTA}-hoz hasonlóan megmutatja, hogy ezen track hogyan illeszkedik az X_{FT} környezetre. β_{NTB} számítását a 3.5 egyenlet szemlélteti. β_{FT} a végső (final) tracking objektum sűrűség értéke, melynél előre fuzionálom az i illetve j trackeket a Kovariancia Unió 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 3.6, az i és j track elő fuzionálását pedig 3.7 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.

$$\beta_{NTA} = \frac{\beta_{BA}}{p(x_j^A|X_{FT})}\beta_{FT} + \beta_{FA} \tag{3.4}$$

$$\beta_{NTB} = \frac{\beta_{BB}}{p(x_j^B|X_{FT})}\beta_{FT} + \beta_{FB} \tag{3.5}$$

$$\beta_{FT} = p(x_{ij}^F|X_{FT}) \tag{3.6}$$

$$x_{ij}^F = P_j^B(P_i^A + P_j^B)^{-1}x_i^A + P_i^A(P_i^A + P_j^B)^{-1}x_j^B \tag{3.7}$$

Ahol β_{FA} és β_{FB} az A illetve B szenzor fals objektumainak sűrűsége. $p(x_i^A|X_{FT})$ az A szenzor i track likelihoodja a környezet (X_{FT}) függvényében, $p(x_j^B|X_{FT})$ pedig a B szenzor j track likelihoodja a környezet függvényében.

$$p(x|X_{FT}) = \beta_{BF} + \sum_{k=1}^{N} \mathcal{N}(x, \hat{x}_k, \hat{P}_k)p(\exists x_k), \tag{3.8}$$

ahol $\hat{x}_k$ a k. kimeneti objektum állapot vektora, $\hat{P}_k$ pedig a hozzá tarrtozó kovariancia mátrix. Ezen objektumokból van N darab, melyeken végigiterálva kiértékelem a Gauss-eloszlás függvényt az x helyen, így meghatározva x track likelihoodját. Mindezt pedig megszorozva a k. kimeneti objektum létezési valószínűségével ($p(\exists x_k)$) és hozzáadva a β_{BF} (birth density) értéket, mely az új objektumok sűrűségét reprezentálja.

Duplikátumok kezelése

Trackek duplikálásának a kiszűrésére bevezetek egy új, szigorúbb küszöbértéket, mely az eredeti küszöbértékhez hasonlóan specifikusan track-párokra értelmezett. Ezen g'_{ij} érték számítását a 3.9 egyenlet ír le.

$$g'_{ij} = 2log\left(\frac{\beta_{FT}P_{TA}P_{TB}(P_{DA} + P_{DB})}{2\pi^{D/2}\beta_{NDA}\beta_{NDB}}\right), \tag{3.9}$$

Ahol β_{NDA} és β_{NDB} a szenzorok specifikus density értéke arra vonatkozóan, hogy a szenzorok nem duplikálnak. Melyeket a következő 3.10 és 3.11 egyenletek írnak le.

$$\beta_{NDA} = p(x_i^A|X_{FT})P_{TA}(1 - P_{DA}) \tag{3.10}$$

$$\beta_{NDB} = p(x_j^B|X_{FT})P_{TB}(1 - P_{DB}) \tag{3.11}$$

Ha egy track-páros likelihood értéke kisebb mint g'_{ij}, akkor duplikáció áll fenn. Ezen g' egy szigorúbb, track páronként értelmezett küszöbérték, mint g.

Látható, hogy a mátrixnak csupán az egyik, a főátló alatti elemei vannak használva. 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 főátló már a mátrix generálásánál feltöltésre kerül nulla értékekkel. Így egy adott T_{ij} nem kerül összerendelésre

önmagával.

A teljes mátrix legenerálása után kiválasztjuk a maximális asszociációs értéket a mátrixból, majd megvizsgáljuk, hogy a hozzá tartozó L_{ij} likelihood érték, a kiszámított g_{ij} értéknél 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 3.2. 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 mindez 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 hozzájuk tartozó track egy duplikáció. A példára visszatérve, ha α_{22}^{13} asszociációs értékhez tartozó L_{22}^{13} likelihood érték kisebb, mint g'^{13}_{22}, akkor ebben az esetben az 1-es 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. Ekkor kerülhet bele abba a klaszterbe, ahova T_{12} és T_{22} is.

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:

1. A két track egyike sem klaszter tag.
2. A két track egyike már egy létező klaszter tagja.
3. 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 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 $\mathbf{A}$ klaszter tagja, míg T_{22} a $\mathbf{B}$ klaszteré. Megvizsgáljuk,

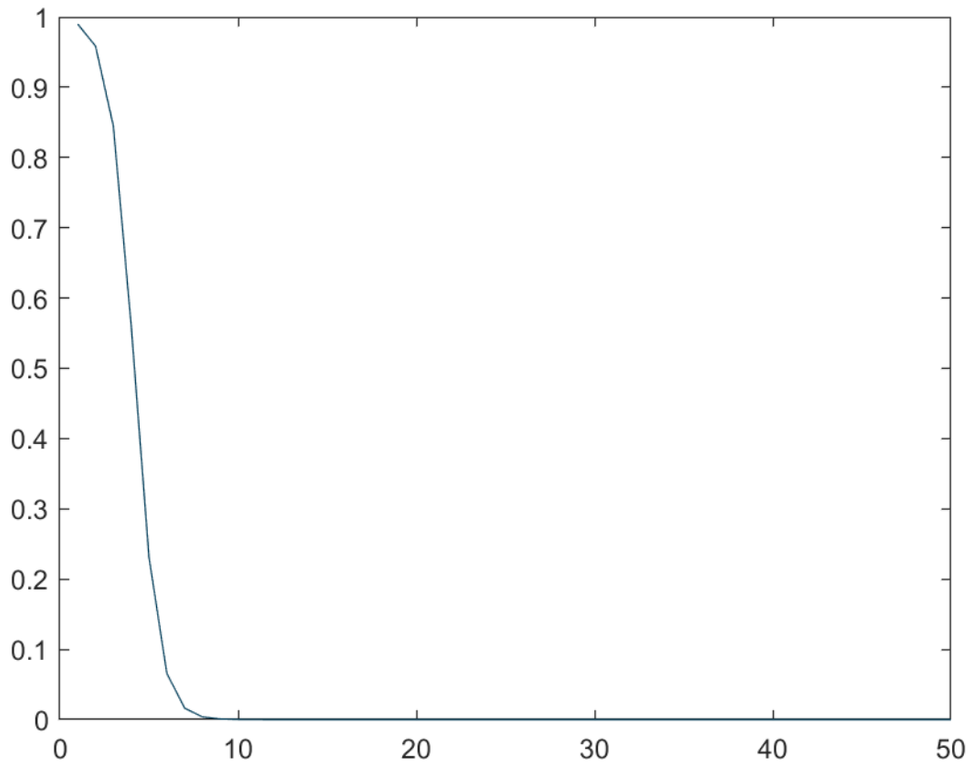
hogy T_{22} milyen átlagos asszociációs likelihood értékkel rendelkezik az $\mathbf{A}$ klaszter tagjaival, majd ugyanezt lefuttatjuk egy, T_{22}-től különböző, másik $\mathbf{B}$ klaszter-beli trackre. És ezt egészen az utolsó $\mathbf{B}$ klaszter-beli elemig folytatva. Abban az esetben, ha mindegyik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

3.2 Fúzió

A klaszterek előállítása után, a bennük szereplő trackek fuzionálására kerül sor. Ehhez először megvizsgálom, hogy az egyes klaszterek mennyi tracket tartalmaznak. Abban az esetben, ha ez a szám egy, akkor a kimeneti fuzionált track egy az egybe azon egy track állapotával fog megegyezni. Ha ez a szám több mint egy akkor a kovariancia unió alapján fuzionálásra kerülnek ezen trackek. A fúzió leglényegesebb pontja a trackek létezési valószínűség értékeinek a fuzionálása. Minden egyes track szennzor szinten kap egy létezési valószínűség értéket, melyek fuzionálására több korábban ismertetett (2.3.3.fejezet) módszer alkalmas. Ezen módszerek egyike a Dempster-Shafer szabály, mely alkalmazható Sensor-to-Sensor, illetve Sensor-to-Global modellekre is.

Sensor-to-Global probléma

Legyen két szenzor A, illetve B, továbbá legyen, egy track, mely $k-n$. időpillanatban mindkét szenzor által detektált, így egy magas létezési valószínűséggel legenerálhatott egy globális objektumot. Azonban tételezzük fel, hogy a $k-(n-1)$. időpillanattól kezdve egészen a k. időpillanatig ezen objektumot csupán az egyik szenzor trackeli. Ebben az esetben a Dempster-Shafer szabályt Sensor-to-Global modellre alkalmazva, 2.31. képlettel számolva minden egyes iterációban a következő függvényt kapjuk eredményül:



3.3. Ábra: Sensor-to-Global problémája

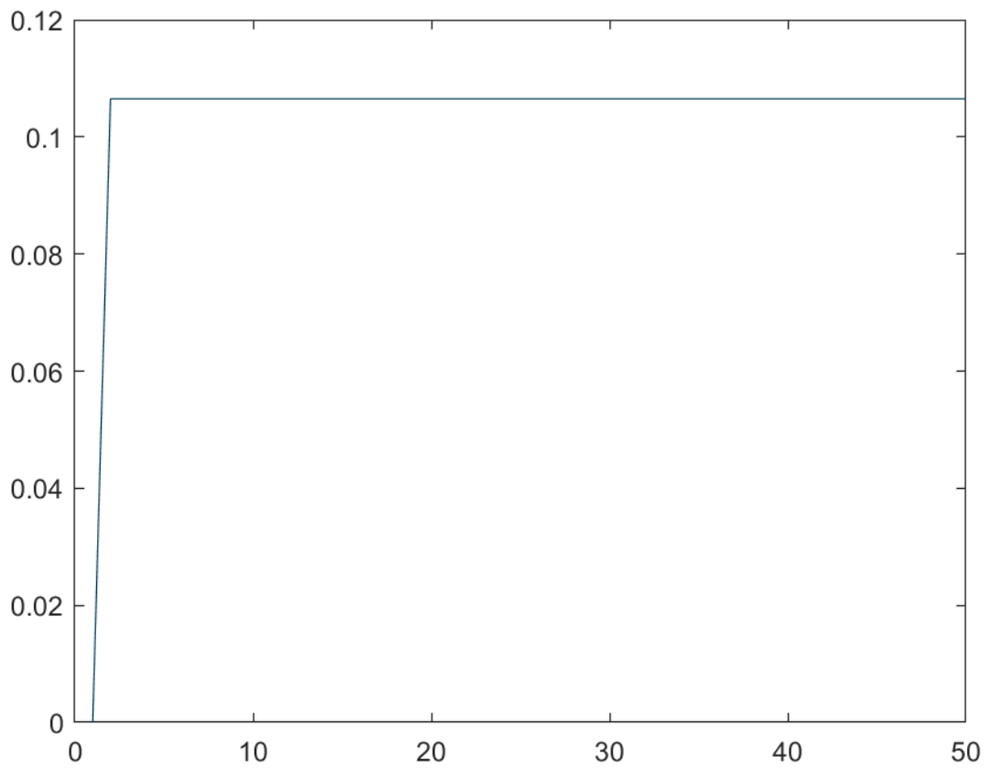
Látható, hogy amint az egyik szenzor elveszíti az objektumot, annak létezési valószínűsége nagyon gyorsan 0-ra csökken. Ez nem ideális, hisz a törlési küszöbértéket (g_d) bármekkorára is állítom, amint az objektumot az egyik szenzor elveszíti, néhány iteráción belül azon objektum törlődik a globális listáról. Tehát hiába kerül egy objektum megerősítésre, csupán addig létezik, amíg mindegyik szenzor észleli. Audun Jøsang [28] részletezi ezen megközelítés problémáit. Mivel az egyik szenzor (legyen A) nem detektálja az objektumot, így arra a szenzorra nézve az adott track létezési valószínűsége $m_k^A(\{\exists\}) = 0$, így 2.31 egyenlet a következőre módosul A szenzor esetében:

$$m_{k|k}^G(\{\exists\}) = \frac{m_{k|k-1}^G(\{\exists\})m_k^A(\{\exists, \nexists\})}{1 - [m_{k|k-1}^G(\{\exists\})m_k^A(\{\nexists\})]}. \tag{3.12}$$

Látható, hogy a prediktált globális létezési valószínűség továbbra is jelen van a számlálóban, ezáltal mindig az előző iterációban kiszámolt alacsony érték felhasználásra kerül a következő iterációban levő számításhoz. Viszont, amikor B szenzor (mely szolgáltat információt, $m_k^B(\{\exists\}) \neq 0$) is lefrissíti a globális track állapotát, 2.31 egyenletet alkalmazva, a számlálóban ugyanúgy megjelenik a korábbi iterációban kiszámolt, A szenzor által lefrissített, alacsony globális létezési valószínűség. Így, B szenzor általi állapotfrissítés is alacsony értéket vesz fel.

Sensor-to-Sensor probléma

A Sensor-to-Global megközelítés problémáit látva merül fel a kérdés, hogy a Sensor-to-Sensor alkalmasabb-e az általam kívánt cél eléréséhez. A 3.4.Ábra szemlélteti a fuzionált valószínűség értékét, abban az esetben, ha az objektumot csupán a B szenzor követi.



3.4. Ábra: Sensor-to-Sensor problémája

Látható, hogy a fuzionált létezési valószínűség megközelítőleg $0,106$ értéket veszi fel az első

iterációban, majd azt tartja a vizsgált időintervallum maradó részén. Ezen érték egy alacsony P^B_{trust} érték mellett került kiszámításra. P^B_{trust} érték azt reprezentálja, hogy az adott szenzor által küldött adatokban mennyire bízunk meg. Például a radar sok fals-pozitív tracket küld, így a rá vonatkozó P_{trust} alacsonyabb mint mondjuk kamera esetében. Ezen $0,106$ hozzávetőlegesen kedvező érték, alacsony P_{trust} mellett, viszont Sensor-to-Sensor modell minden iterációban újraszámolja a létezési valószínűség értékeket, figyelmen kívül hagyva a korábbi időpillanatokhoz tartozóakat.
Legyen egy objektum, mely a $k-1.$ időpillanatban mindkét szenzor által detektált, így legenerálásra kerül egy magas, mondjuk $0,9$ értékű létezési valószínűséggel. Azonban, ha a következő, $k.$ időpillanatban az egyik szenzor elveszíti, akkor ezen magas érték lecsökken olyan alacsonyra, melyet a 3.4.ábra is szemléltet. Abban az esetben, ha a g_d magasabb, mint azon érték, melyre lecsökken az objektum létezési valószínűsége, az objektum törlésre kerül. Az objektum korai törlésének megelőzése érdekében egy olyan karakterisztika lenne megfelelő, melyet a 3.3.ábra is mutat. Ez esetben ugyanis a létezési valószínűség iterációról-iterációra fokozatosan cseng le, hisz itt a Sensor-to-Sensor eljárással ellentétben van visszacsatolás a korábbi időpillanatokhoz tartozó állapotokhoz.

Hibrid modell

A különböző megközelítések hibáit már ismerve adódik a motiváció, hogy a Sensor-to-Sensor és a Sensor-to-Global eljárást ötvözzem. Ezáltal egy olyan hibrid architektúrát alkotva, mely esetén, egyrészt van visszacsatolás a korábbi állapotokhoz, így fokozatos a létezési valószínűség lecsengése, ha az egyik szenzor nem küld adatot. Másrészt a függvény egy szenzor adatszolgáltatása esetén nem 0 értékhez konvergál, hanem egy annál jóval magasabbhoz. A Hibrid modell kimenete egy darab szenzor adatküldése esetén a 3.5.ábrán látható. Ezen karakterisztika eléréséhez a Dempster-Shafer szabály alkalmazása helyett a PDA szabályainak megfelelően [26] becsülöm a létezési valószínűséget. Így a 2.27 egyenlet a következőképpen módosul:

$$m^G_{k|k}(\{\exists\}) = \frac{m^S_k(\{\exists\}) + m^G_{k|k-1}(\{\exists\})m^S_k(\{\exists,\nexists\}) + m^G_{k|k-1}(\{\exists,\nexists\})m^S_k(\{\exists\})}{[m^S_k(\{\exists\}) + m^G_{k|k-1}(\{\exists\})m^S_k(\{\exists,\nexists\}) + m^G_{k|k-1}(\{\exists,\nexists\})m^S_k(\{\exists\})] + [1 - m^G_{k|k-1}(\{\exists\})]}, \tag{3.13}$$

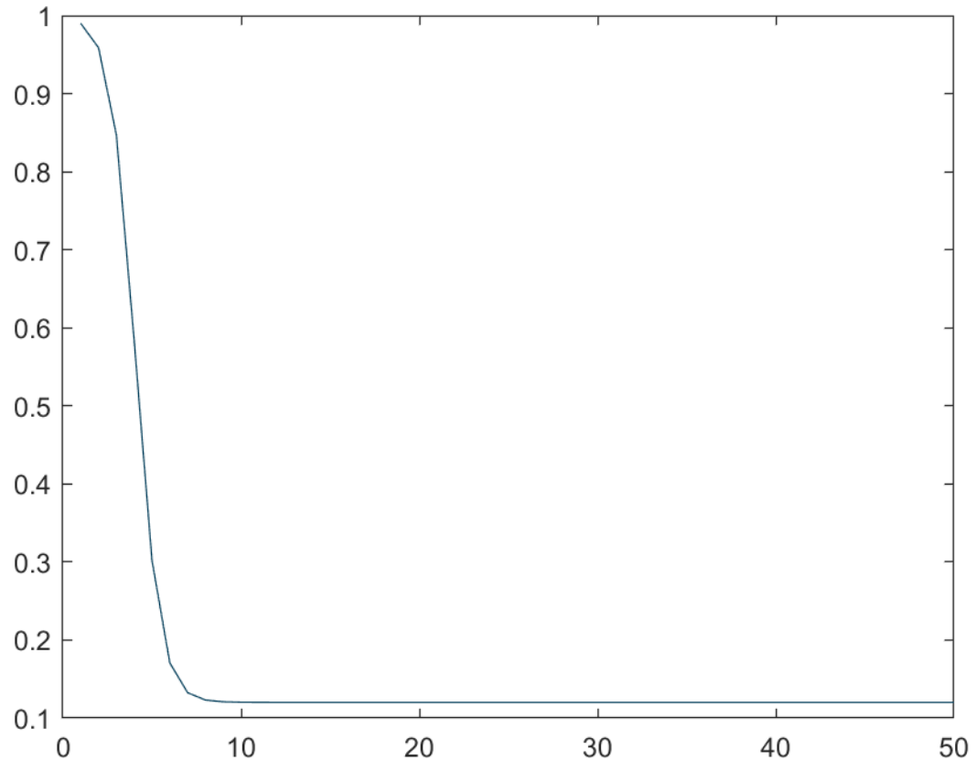
ahol ^{S} index a szenzort reprezentálja. Egy iterációban a következő, kétlépcsős módszerrel áll elő az új, globális létezési valószínűség: Először, felhasználva 3.13 egyenletet fuzionálom a $k-1.$ időpillanathoz tartozó globális objektum $k.$ időpillanatra becsült létezési valószínűségét és az A szenzortól származó track létezési valószínűségét, melyek a következőképp állnak elő:

$$m^G_{k|k-1}(\{\exists\}) = P_{Survive} \cdot m^G_{k-1|k-1}(\{\exists\}), \tag{3.14}$$

$$m^G_{k|k-1}(\{\exists,\nexists\}) = 1 - P_{survive}, \tag{3.15}$$

$$m^A_{k|k-1}(\{\exists\}) = P^A_D \cdot P^A\{\exists\}, \tag{3.16}$$

$$m^A_{k|k-1}(\{\exists,\nexists\}) = 1 - P^A_D, \tag{3.17}$$



3.5. Ábra: Hibrid fúziós modell

ahol P_D a detektációs valószínűség, mely szenzorspecifikus érték, $P^A\{\exists\}$ a vizsgált track szenzor szinten számított létezési valószínűség értéke, továbbá $P_{survive}$ túlélési valószínűség, mely az én implementációmban egy magas, konstans érték. Így a 3.13 egyenlet öltse a következő alakot:

$$m_{k|k}^{G_A}(\{\exists\}) = \frac{m_k^A(\{\exists\}) + m_{k|k-1}^G(\{\exists\})m_k^A(\{\exists,\nexists\}) + m_{k|k-1}^G(\{\exists,\nexists\})m_k^A(\{\exists\})}{[m_k^A(\{\exists\}) + m_{k|k-1}^G(\{\exists\})m_k^A(\{\exists,\nexists\}) + m_{k|k-1}^G(\{\exists,\nexists\})m_k^A(\{\exists\})] + [1 - m_{k|k-1}^G(\{\exists\})]}. \tag{3.18}$$

Tehát $m_{k|k}^{G_A}(\{\exists\})$ az A szenzortól származó track létezési valószínűsége által lefrissített, globális objektum létezési valószínűsége. Azonban szükség van a B szenzor által szolgáltatott track létezési valószínűségének is a fuzionálására a globális objektuméval. Így a 3.19 egyenlet eredményét felhasználva 3.13 tovább módosul:

$$m_{k|k}^{G}(\{\exists\}) = \frac{m_k^B(\{\exists\}) + m_{k|k}^{G_A}(\{\exists\})m_k^B(\{\exists,\nexists\}) + m_{k|k}^{G_A}(\{\exists,\nexists\})m_k^B(\{\exists\})}{[m_k^B(\{\exists\}) + m_{k|k}^{G_A}(\{\exists\})m_k^B(\{\exists,\nexists\}) + m_{k|k}^{G_A}(\{\exists,\nexists\})m_k^B(\{\exists\})] + [1 - m_{k|k}^{G_A}(\{\exists\}))]}, \tag{3.19}$$

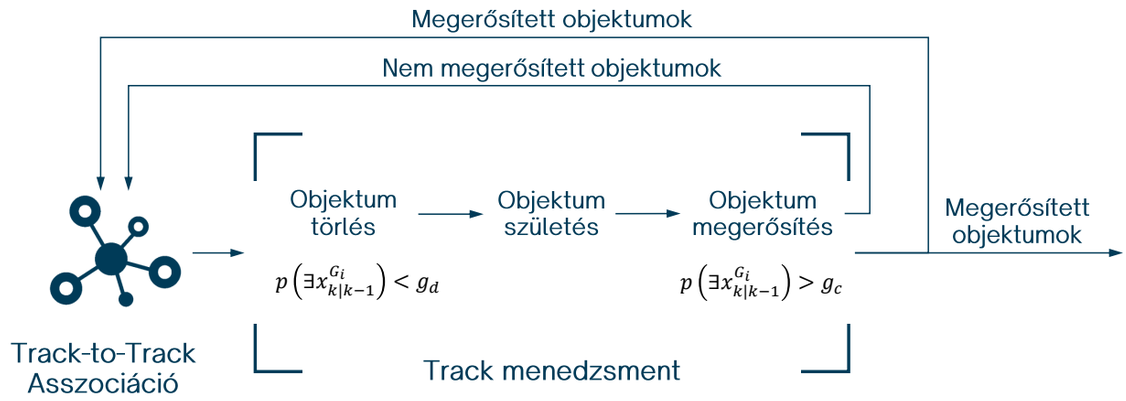
ahol $m_{k|k}^{G_A}(\{\exists,\nexists\})$ 3.15 egyenlet alapján számítható.
Így előállt a fuzionált létezési valószínűség.

3.3 Track menedzsment

Miután megtörtént minden egyes klaszterben a trackek fuzionálása, majd azok asszociálása a globális objektumokhoz, a fúzió egyik legfontosabb eleme, a track menedzsment következik. Mint

ahogy a 3.6.Ábra is mutatja az objektum menedzsment több, elkülönülő blokkból áll. Ezek rendre az objektum törlés, objektum születés és objektum megerősítés.

Először megvizsgálom, hogy melyek azok az objektumok, melyek törölhetők a globális objektum



3.6. Ábra: Track menedzsment

listáról. Ez az előző alfejezetben részletezett fuzionált létezési valószínűség érték alapján határozható meg. Tehát legyen $p(\exists x_{k|k-1}^{G_i})$, mely i. kimeneti (globális) objektum prediktált létezési valószínűsége a $k-1$. időpillanatról a k.-ra. Ha ezen érték egy előre deklarált g_d törlési küszöbértéknél alacsonyabb, akkor az objektum törlésre kerül a globális listáról.

Objektumszületés esetén megvizsgálom, hogy melyek azok a szenzor szinten fuzionált trackek, melyek az asszociáció során nem kerültek hozzárendelésre egyetlen globális objektumhoz sem. Ezen trackek inicializálásra kerülnek, mint új, nem megerősített (tentative) globális objektumok, melyek a track menedzsment végén visszacsatolásra kerülnek az asszociációs blokkba.

A trackek megerősítése hasonló módon működik a korábban ismertetett törlési eljáráshoz. Megvizsgálom, hogy a $p(\exists x_{k|k-1}^{G_i})$ érték magasabb-e mint egy előre beállított g_c megerősítési küszöbérték. Ha ez teljesül, akkor az adott globális objektum megerősítetté (confirmed) válik. Fontos megjegyezni, hogy az újonnan megszületett objektumok is felülvizsgálásra kerülnek. A track menedzsment végén, hasonlóan a nem megerősített objektumokhoz, ezen megerősített objektumok is visszacsatolásra kerülnek az asszociációs blokkba.

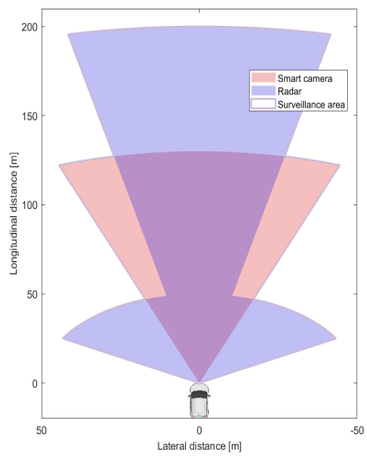
Miután megtörtént az objektum menedzsment, a kimeneti megerősített, illetve nem megerősített objektumok visszacsatolásra kerülnek az asszociációs blokkba, mint ahogy azt a 3.1. és 3.6.ábra is szemléltet. Ezen visszacsatolt objektumok és a következő iterációban szenzor szinteken előálló bemeneti trackek közt lefuttatok egy Multi-Object Tracking algoritmust, mely egy korábbi fejlesztésem [29].

4 Kiértékelés

4.1 A kiértékelési környezet

A fejlesztett módszert sikeresen adaptáltam egy valós szenzorokon álló mérési klaszterre, mely egy járműipari radarból, illetve egy okos kamerából áll. Ezen szenzorok trackelt objektumokat biztosítanak. A kamera $130m$-es látótávolsággal és $40°$-os látószöggel rendelkezik. A radar kettő

	Continental ARS408	Mobileye EyeQ2
Szenzor típus	Okos radar	Okos kamera
Frekvencia	12 Hz	9 Hz
Mérés	• Pozíció (long., lat.) • Sebesség (long., lat.)	• Pozíció (long., lat.) • Sebesség (long.) • Gyorsulás (long.)
Detekciók maximum száma	50	4
Klasszifikáció	-	+
FoV szög	120°, 20°	40°
Maximális távolság	50 m, 200 m	130 m



4.1. Ábra: Mérési klaszter

látómezővel rendelkezik, melyek közül az egyik $60m$-es látótáv $120°$-os látószög, míg a másik pedig $200m$ látótáv és $20°$-os látószög. A radar frekvenciája $12Hz$ körül van, míg a kamerának ezen értéke megközelítőleg $9Hz$. A szenzor konfiguráció tartalmaz továbbá egy másik kamerát is, mely a valós objektumok legenerálásában és vizualizációjában nyújt segítséget. Ezen kamera a szenzorokkal szinkronizáltan rögzített. A mérés az M1/M7 kivezető szakaszán történt. A keletkezett videó 7000 képkockából áll.

A szenzorok nem minden esetben detektálják az összes objektumot, ezért létrehoztam 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 a következő táblázat mutatja.

Paraméterek	Jelölés	Kamera	Radar
Poisson-eloszlás	λ	$0,001$	10
Született objektumok becsült száma	n_B	1	1
Született objektumok sűrűsége	β_B	$\frac{n_B}{V_C}$	$\frac{n_B}{V_R}$
Detekciós valószínűség	P_D	$0,85$	$0,9$
Duplikációs valószínűség	$P_{duplicate}$	$0,001$	$0,04$
Szenzorba vetett bizalom	P_{trust}	$0,9$	$0,5$
Frekvencia $[Hz]$	f	9	12
Maximum detekciószám	N_{max}	4	50

4.1. Táblázat: Szenzor paraméterek

Paraméterek	Jelölés	Érték
Megerősítési küszöbérték	g_c	$0,9$
Törlési küszöbérték	g_d	0.07
Maximum objektumszám	O_{max}	30

4.2. Táblázat: Fúzió paraméterek

4.2 Kiértékelési módszertan

A megerősített objektumok, illetve a manuálisan előállított valós (Ground truth) objektumok közt megvalósítok egy Munkres asszociációt [30]. A lineáris hozzárendelési feladat esetén a mátrix sorok, illetve oszlopok közti összerendelést valósítja meg oly módon, hogy minden sor egy oszlophoz tartozzon és az összerendelések összköltsége minimum (vagy maximum) legyen. Minden sor-oszlop összerendelés költsége eltárolásra kerül a költségmátrixba.

$$TC = \sum_{i=1}^{p} Cost(M(i,1), M(i,2)) + costUnmatched \cdot (m + n - 2p), \tag{4.1}$$

ahol TC a teljes költsége az M megoldásnak. $Cost(i,j)$, függvény mely meghatározza, hogy mekkora i sor és j oszlop összerendelésének a költsége. Ha egy sor, vagy egy oszlop nem kerül hozzárendelésre, akkor a teljes költség megnő $costUnmatched \cdot (m + n - 2p)$ értékkel, ahol m és n a költségmátrix sorainak, illetve oszlopainak száma, továbbá p, mely az M mátrix sorainak a száma. A költségmátrix számítása a kimeneti és a valós objektumok közti Mahalanobis távolság alapján számítódik. A laterális szórás értéke $1,5m$, míg a longitudinális pedig $10m$, továbbá annak a költsége, hogy a hozzárendelés nem történik meg $1m$.

Miután meghatározásra kerültek a párok, kiszámításra kerülnek a fals-negatív, illetve fals-pozitív objektumok. Egy valós objektum jelen esetben akkor fals-negatív, ha nem kerül hozzárendelésre egyetlen kimeneti objektum sem. Továbbá egy kimeneti objektum abban az esetben fals-pozitív, ha nem kerül hozzárendelésre egyetlen valós objektum sem. Ezen fals-negatív (FN), fals-pozitív (FP), illetve a megtalált és valós (true-positive, TP) objektumok számával már számítható a P_r (*precision*), illetve R_c (*recall*) értékek, melyek rendre:

$$P_r = \frac{TP}{TP + FP} \tag{4.2}$$

$$R_c = \frac{TP}{TP + FN}. \tag{4.3}$$

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. 4.2, illetve 4.3 egyenleteket felhasználva, a *precision* és *recall* harmonikus közepeként előáll az F_1, mely:

$$F_1 = \frac{2P_r \cdot R_c}{P_r + R_c}. \tag{4.4}$$

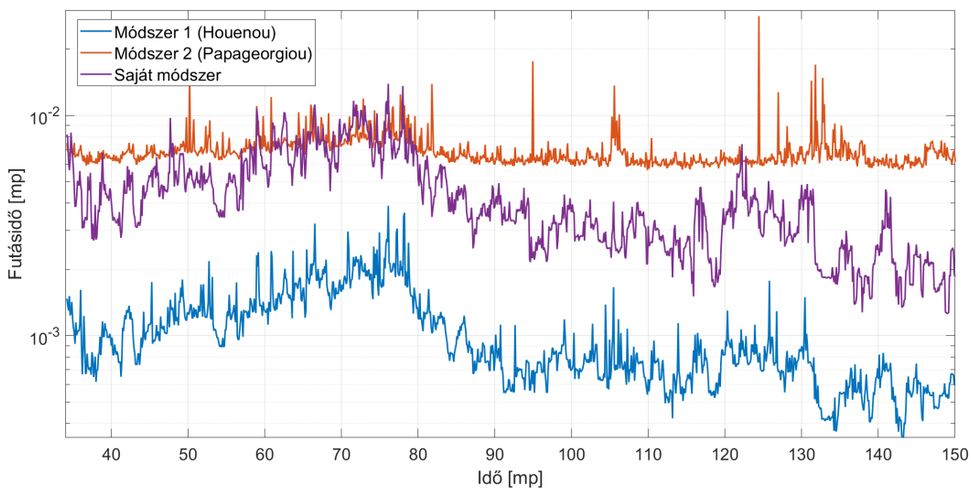
4.3 Eredmények

A fejlesztett fúziós módszer 2.3.1. fejezetben részletezett asszociációs algoritmusokkal, illetve a Dempster–Shafer Sensor-to-Sensor [25] és Sensor-to-Global [8] módszerekkel lett összevetve. Mind Dimitri J Papageorgiou [9] és Adam Houenou [10] által fejlesztett módszereknél két esetet vizsgáltam. Először a fúzió kimenetén megjelenő összes objektumra vizsgáltam az algoritmusokat, míg másodszor csupán azon objektumokra, melyek a két szenzor fúziójának eredményeként adódtak, tehát figyelmen kívül hagyva azon klasztereket, melyek csupán 1 tracket tartalmaznak. Az eredmények a 4.3-es táblázatban láthatók.

	Precision	Recall	F1 score
Radar	$48,75\%$	$87,06\%$	$62,50\%$
Kamera	$97,43\%$	$73,40\%$	$83,73\%$
Módszer 1 [10]: Összes objektumra	$50,80\%$	$94,79\%$	$66,15\%$
Módszer 1 [10]: Fuzionált objektumokra	$99,29\%$	$65,80\%$	$79,15\%$
Módszer 2 [9]: Összes objektumra	$51,46\%$	$94,75\%$	$66,69\%$
Módszer 2 [9]: Fuzionált objektumokra	$99,32\%$	$68,46\%$	$81,05\%$
Dempster–Shafer Sensor-to-Sensor	$97,79\%$	$73,88\%$	$84,17\%$
Dempster–Shafer Sensor-to-Global	$95,94\%$	$76,54\%$	$85,15\%$
Hibrid modell	$95,76\%$	$85,56\%$	$90,46\%$

4.3. Táblázat: Kiértékelési eredmények

A Sensor-to-Sensor, illetve Sensor-to-Global módszerek esetén a saját asszociációs algoritmusomat használtam. Ezáltal a kiértékelés fúzió fokuszáltabbá válhatott. Mint ahogy a táblázat is szemlélteti, az általam fejlesztett Hibrid modell magasan túlteljesíti a referencia algoritmusokat. $F1score$ esetében látható, hogy az 1. illetve 2. módszer összes objektum vizsgálatakor ront, míg az általam javasolt megoldás határozottan javít a szenzorok teljesítményén.



4.2. Ábra: Futási idők

Az általam fejlesztett módszer kedvező futási idővel bír, mint ahogy azt a 4.2.ábra is szemlélteti. Mind az 1. módszer, mind pedig a 2. módszer a likelihood mátrix felépítése során azon track-párokra nem kerül kiszámításra likelihood érték, melyek ugyanattól a szenzortól származnak,

így kizárva a track duplikátumok létezését. Az általam javasolt módszer futási ideje megközelítőleg az $57.$ másodperctől a $80.$ másodpercig a $2.$ módszer futási idejéhez hasonló értékeket vesz fel. Ennek oka a likelihood mátrix felállítása, hisz ezen két módszerrel ellentétben az általam alkalmazott adatasszociációnál, minden track párra kiszámításra kerül egy likelihood érték. Ezáltal egy pontosabb asszociációt megvalósítva, mely a duplikátumok kiszűrésére is alkalmas.

5 Konklúzió

Az eredmények alapján egyértelműen kijelenthető, hogy egy olyan szenzorfúziós architektúrát sikerült fejleszteni, melynek futásideje kedvező miközben a teljesítménye a hagyományos algoritmusokénál magasabb.

A track-to-track fúziós logika felépítésénél fontos kérdés, hogy hogyan lehet a különböző szenzorokból érkező trackeket oly módon összerendelni, hogy számtáskapacitási igényt spóroljunk a teljesítmény növelése mellett. A fuzionált trackek visszacsatolása az asszociációs blokkba egy olyan hibrid track-to-track architektúrát eredményez, mely megfelel az imént felsorolt követelményeknek. Megfelelően kiszámított, trackpár specifikus küszöbértékekkel pontosabban végezhető a különböző szenzoroktól származó trackek asszociációja a klaszterező algoritmusnak köszönhetően. Emellett egy előzőtől különböző, szintén track-pár specifikus küszöbérték meghatározásával képes vagyok megmondani a szenzoroktól érkező esetleges track-duplikációk megjelenését, ezzel is pontosítva az asszociációt. A kutatásom során a számításigény mellett fontos szempont volt a kimenet megbízhatósága. Ebben a legfőbb szerepet a szenzoroktól származó fals objektumok szűrése játszotta. A fejlesztett létezési valószínűség becslési eljárás hatékonyan támogatja a fals objektumok szűrését és a valódi objektumok megerősítését. Mindennek köszönhetően, felgyorsítható a jármű automatizáltság növekedése a költséghatékonyság megőrzése mellett. Továbbá ezen fúziós logika nem érzékeny a szenzorklaszterek változására, mely így összességében egy moduláris architektúrát eredményez.

A szenzoradatokon végzett, továbbfejlesztett fúziós algoritmusnak köszönhetően magas konfidenciával bíró környezetérzékelést sikerült megvalósítani, mely megközelíti a state-of-the-art objektum fúziós módszerek teljesítményét, miközben a számításkapacitási igény alacsony maradhatott, így támogatva a valós idejű alkalmazhatóságot.

Irodalomjegyzék

[1] J. Van Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 384–406, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X18302134

[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, p. 244–250, 2016. [Online]. Available: https://pp.bme.hu/tr/article/view/9464

[3] W. Gruel and J. M. Stanford, "Assessing the long-term effects of autonomous vehicles: A speculative approach," *Transportation Research Procedia*, vol. 13, pp. 18–29, 2016, towards future innovative transport: visions, trends and methods 43rd European Transport Conference Selected Proceedings. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352146516300035

[4] A. A. Association, "Advanced driver assistance technology names aaa's recommendation for common naming of advanced safety systems," 2019. [Online]. Available: https://www.sbdautomotive.com

[5] K. Rana and P. Kaur, "Comparative study of automotive sensor technologies used for unmanned driving," 01 2021, pp. 346–350.

[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*, vol. 21, no. 16, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/16/5397

[7] ——, "An overview of autonomous vehicles sensors and their vulnerability to weather conditions," *Sensors*, vol. 21, no. 16, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/16/5397

[8] M. Aeberhard, "Object-level fusion for surround environment perception in automated driving applications," 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:115639617

[9] D. Papageorgiou and M. Holender, "Track-to-track association and ambiguity management in the presence of sensor bias," vol. 6, 08 2009, pp. 2012 – 2019.

[10] A. Houenou, P. Bonnifait, and V. Cherfaoui, "A Track-To-Track Association Method for Automotive Perception Systems," in *IEEE Intelligent Vehicle Symposium (IV 2012)*, Alcala de Henares, Spain, Jun. 2012, pp. 704–710. [Online]. Available: https://hal.science/hal-00740787

[11] Y. Bar-Shalom and R. Xiao, "Multitarget-multisensor tracking: Principles and techniques," 1995. [Online]. Available: https://api.semanticscholar.org/CorpusID:110757951

[12] S. J. Julier and J. K. Uhlmann, "Using covariance intersection for slam," *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 3–20, 2007, simultaneous Localisation

and Map Building. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889006001436

[13] J. K. Uhlmann, "Covariance consistency methods for fault-tolerant distributed data fusion," *Information Fusion*, vol. 4, no. 3, pp. 201–215, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253503000368

[14] R. Mittu and F. Segaria, "Common operational picture (cop) and common tactical picture (ctp) management via a consistent networked information stream (cnis)," 2000. [Online]. Available: https://api.semanticscholar.org/CorpusID:59018463

[15] A. Berg, , and M. Käll, "Track-to-track fusion for multi-target tracking using asynchronous and delayed data," 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:59128929

[16] Y. Bar-Shalom and E. Tse, "Tracking in a cluttered environment with probabilistic data association," *Automatica*, vol. 11, no. 5, pp. 451–460, 1975. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0005109875900217

[17] Y. Bar-Shalom, "Tracking methods in a multitarget environment," *IEEE Transactions on Automatic Control*, vol. 23, no. 4, pp. 618–626, 1978.

[18] F. García, A. de la Escalera, and J. M. Armingol, "Joint probabilistic data association fusion approach for pedestrian detection," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 1344–1349.

[19] D. Musicki, R. Evans, and S. Stankovic, "Integrated probabilistic data association," *IEEE Transactions on Automatic Control*, vol. 39, no. 6, pp. 1237–1241, 1994.

[20] D. Musicki and R. Evans, "Joint integrated probabilistic data association: Jipda," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 3, pp. 1093–1099, 2004.

[21] D. Musicki, "Track score and target existence," in *2006 9th International Conference on Information Fusion*, 2006, pp. 1–7.

[22] G. Shafer, "A mathematical theory of evidence," *Princeton University Press,*, 1976.

[23] U. RAKOWSKY, "Fundamentals of the dempster-shafer theory and its applications to reliability modeling," *International Journal of Reliability, Quality and Safety Engineering - IJRQSE*, vol. 14, 12 2007.

[24] D. Koks and S. Challa, "An introduction to bayesian and dempster-shafer data fusion," 2003. [Online]. Available: https://api.semanticscholar.org/CorpusID:908081

[25] M. Aeberhard, S. Paul, N. Kaempchen, and T. Bertram, "Object existence probability fusion using dempster-shafer theory in a high-level sensor data fusion architecture," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 770–775.

[26] L. Lindenmaier, S. Aradi, T. Bécsi, O. Törő, and P. Gáspár, "Gm-phd filter based sensor data fusion for automotive frontal perception system," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 7, pp. 7215–7229, 2022.

[27] C.-Y. Chong, S. Mori, W. Barker, and K.-C. Chang, "Architectures and algorithms for track association and fusion," *IEEE Aerospace and Electronic Systems Magazine*, vol. 15, no. 1, pp. 5–13, 2000.

[28] A. Jøsang, J. Diaz, and M. Rifqi, "Cumulative and averaging fusion of beliefs," *Information Fusion*, vol. 11, no. 2, pp. 192–200, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S156625350900044X

[29] L. Lindenmaier, B. Czibere, S. Aradi, and T. Bécsi, "A runtime-efficient multi-object tracking approach for automotive perception systems," in *2023 IEEE 17th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, 2023, pp. 000 785–000 792.

[30] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957. [Online]. Available: http://www.jstor.org/stable/2098689

Ábrajegyzék

1.1 A Track-to-Track fúzió lépései 2
2.1 Különböző szenzortípusok erősségei 3
2.2 Szenzorfúziós szintek 4
2.3 Központi szintű architektúra 5
2.4 Track-to-Track szenzorfúzió 6
2.5 Sensor-to-Sensor fúziós architektúra 7
2.6 Sensor-to-Global fúziós architektúra 7
2.7 Korábbi időpillanatokhoz tartozó track állapotok felhasználása Track-to-Track asszociációnál 10
2.8 *A* track (piros), *B* track (zöld) és a fuzionált track (kék) kovarianciája 12
2.9 *A* track (piros), *B* track (zöld) és a fuzionált track (kék) kovarianciája 13
2.10 Objektum menedzsment 14
3.1 Hibrid Track-to-Track architektúra visszacsatolással 18
3.2 A klaszterezés folyamata, duplikátumok kezelésével 19
3.3 Sensor-to-Global problémája 22
3.4 Sensor-to-Sensor problémája 23
3.5 Hibrid fúziós modell 25
3.6 Track menedzsment 26
4.1 Mérési klaszter 27
4.2 Futási idők 29

Táblázatjegyzék

4.1 Szenzor paraméterek 27
4.2 Fúzió paraméterek 28
4.3 Kiértékelési eredmények 29

Lezárás dátuma: 2023.11.07.