Pozícióbecslés hatékonyságának növelése beltérben

Tudományos Diákköri Dolgozat

Daubner Tibor Tamás

Villamosmérnöki és Informatikai kar

Konzulens: Dr. Huszák Árpád

Budapest, 2012

Tartalomjegyzék

Ábrajegyzék

1 Bevezetés

Napjainkban egy átlagember életébe számos, valamilyen hálózathoz csatlakoztatott eszköz lép be. Ezek az eszközök a nap nagy részében velünk vannak, könnyen hozzájuk férhetünk és számos funkció ellátására képesek. Közülük az egyik a GPS, melynek segítségével műholdakhoz kapcsolódhatunk, azaz meghatározhatjuk helyzetünket, megtervezhetjük útvonalunkat. Azonban dacára a mai kor fejlett és a hétköznapi ember számára is elérhető technológiájának, még mindig kérdéses az említett funkció elérése belső térben. Az árnyékolás miatt ezeken a helyeken műholdak nem használhatók, viszont számos esetben rendkívül hasznos lehetne a szolgáltatás elérése. Az egyik ilyen eset a parkolóházakban való navigáció. Felmerülhet a kérdés, miért fontos egy parkolóházban útvonalat tervezni? A válasz azonban a vártnál kézenfekvőbb, hiszen ha a parkolóház átlagosan telített, akkor egy autós, amíg magától megtalálja a számára kedvező szabad helyet (pl. közel a mozi bejáratához), addig jelentősen több káros anyagot bocsájt ki, mintha azt szoftveres segítséggel tette volna, nem is beszélve arról, hogy azzal saját költségeit és a felesleges stresszt is csökkenthette volna.

A GPS helyett számos IPS (Indoor Positioning System) – beltéri helymeghatározó rendszer létezik (GSM, IrDa, RFID, Bluetooth, UWB, Wi-Fi), melyek közül a legolcsóbb és legelterjedtebb a Wi-Fi alapú pozicionálás, mely technológia a legtöbb mobil készülék számára szintén elérhető és előfizetést nem igényel, ezért is választottuk ezt a megoldást. Wi-Fi technológiát alkalmazva, valamint megfelelően optimalizált algoritmust választva, beltéri helymeghatározó rendszerünk jól kiegészíti a szabadtéri GPS műholdak alkalmazásának előnyeit.

Dolgozatomban az Alle bevásárló központ parkolóházának Wi-Fi-vel támogatott iParking rendszerén végzett méréseket felhasználva végeztem szimulációkat és futtattam különböző pozícionáló algoritmusokat. Célom az volt, hogy az egyes algoritmusok becsléseit pontosítani tudjam. Célkitűzésem az volt, hogy mérőszámokkal igazoltan egyre hasznosabb eredményt érjen el az adott algoritmus.

A témában fellelhető helymeghatározási módszercsoportok közül az ujjlenyomat alapút választottam kiindulópontnak. Szimulációs környezetnek a Matlab programrendszert alkalmaztam, amelynek kihasználva pozitív adottságait, egy beépített K-legközelebbi szomszéd kereső algoritmusból (KNN-Search) indultam ki, majd ezt finomítottam lépésről lépésre. Az algoritmus a minta-illesztés módszerét alkalmazza, az adott pozícióban mért jelszintek alapján. A jelszint minták adatbázisát (ujjlenyomat) helyszíni mérések alapján töltöttem fel, melyek referenciapontként szolgáltak a pozícióbecslés során. Dolgozatomban olyan becslésjavító módszereket dolgoztam ki és mutattam be, melyek az alap KNN keresést kiegészítve hatékonyabbá teszik a helymeghatározást. Múltbéli átlagolás, súlyozás, útvonalra illesztés módszerekkel sikerült a döntések helyességén javítani. További ötletem volt továbbá még a memória használata, valamint a döntési lehetőségek korlátozása távolság alapján. A kidolgozott becslésjavító módszerek hatékonyságának elemzéséhez létrehoztam egy mérőszámot, mely az egyes eredmények közötti számszerű eltérést, eredményességet hivatott megmutatni.

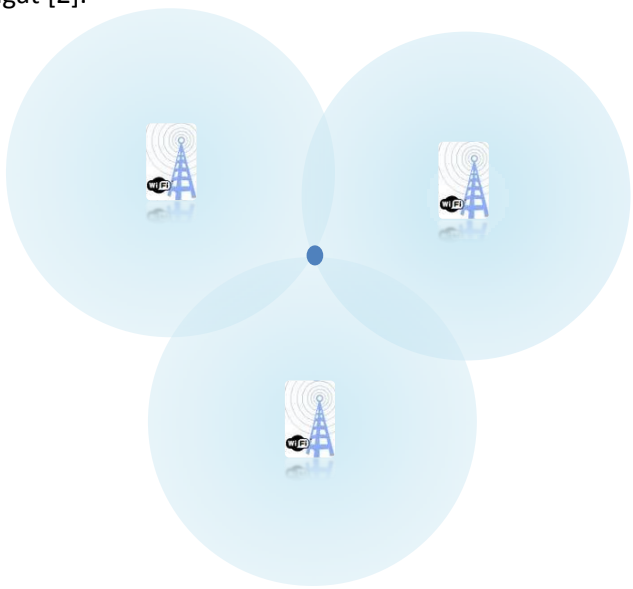
A fenti becslésjavító módszereket alkalmazva lehetőségem volt szimulálni és lemérni az egyes paraméter beállítások, mérési elrendezések eredményeit, melyeket összevetve értékes következtetéseket vontam le a beltéri helymeghatározással kapcsolatosan.

2 Beltéri helymeghatározás módszerei

Ahhoz, hogy belső térben meg tudjuk határozni egy jármű helyzetét, szükség van egy referencia koordináta rendszerre, melyben a jármű mozog, valamint referencia pontokra, melyekhez viszonyítjuk a felhasználók pozícióját. Ezáltal már tudjuk modellezni az előforduló eseteket, útvonalakat. A pozícionálás ebben a koordináta rendszerben mérésekből származó adatok feldolgozásával valósulhat meg. Az ezen a területen használatos technikák közül az egyik csoportba azok tartoznak, ahol a fizika törvényszerűségeit felhasználva, matematikai számításokat végzünk, különböző jelterjedési modellek alkalmazása mellett. Ebbe a csoportba tartoznak a különböző háromszögelési technikák, melyeket a 2.1 fejezetben ismertetek. A másik csoport ahol az Access Point-ok (továbbiakban AP) helyzetének ismerete nélkül a referencia koordinátarendszerben tárolt méréseket felhasználva, úgynevezett ujjlenyomat szerűen pozícionálunk. Mindkét esetben szűkségünk van az AP-ok által sugárzott jel erősségére, melyet a Wi-Fi eszközök jelentős fejlődésének köszönhetően már komolyabb számítás nélkül, könnyedén megkaphatunk, adó és vevő oldalon is [1].

2.1 Háromszögeléses technikák

A háromszögeléses módszereket két részre oszthatjuk. Az iránymérésen alapuló technikák [6] az eszköz adott pontokhoz képesti iránya alapján számolnak, míg a távolságmérésen alapulóak a tárgy távolságát határozzák meg rögzített pontoktól, majd az alapján végzik a pozícionálást. Az irányméréshez irányított antennákra van szükség, melyhez nagyon pontos irányítás kell és rendkívül drága, szabványos Wi-Fi eszközök használata esetén ezért ez a módszer nem jöhet szóba. A távolságmérésen alapuló megoldások több meghatározott helyen lévő adó és a vevő közti távolságok mérésével számolják az eszköz helyét. A helymeghatározás a jel erősségének, fázisának vagy késleltetésének mérésével történhet. Rádióhullámok terjedési sebessége ismeretében, ha pontosan le tudjuk mérni az időt, amíg a jelek az adótól a vevőbe elértek (Time Of Arrival, TOA) akkor megkaphatunk az adó körül egy *r* sugarú kört (vagy gömböt) és ezen a körön helyezkedik el az eszköz. Több ilyen kör (vagy adott esetben gömb) metszéspontja jelöli ki a pozíciót. Ahogy az 1. ábraán is látható, ennél a módszernél kulcsfontosságú a távolság mérés. Annak pontossága határozza meg a pozícionálás hatékonyságát [2].



1. ábra - Példa a háromszögeléses módszerre

Lehetőségünk van az észlelt időkülönbség mérése alapján is meghatározni a pozíciót, ezt a módszert OTD (Observed Time Difference) rövidítéssel jelöli a szakirodalom. Ehhez az AP-k óráinak teljesen szinkronban kell üzemelniük. A módszer működése azon alapszik, hogy a felhasználó által használt eszköz vesz egy az AP-k által egyszerre elküldött jelsorozatot, mely a távolságtól függő késleltetéssel rendelkezik. Az egyes AP-któl érkezett jelsorozatok érkezési idejét a nem szinkronban lévő, de pontos órájával az eszköz regisztrálja, majd veszi az egymáshoz mért különbségüket. Az azonos távolságkülönbségek egy-egy hiperbola mentén helyezkednek el. Az eszköz a hiperbolák metszéspontjában van [5].

2.2 Ujjlenyomat alapú módszerek

Az ujjlenyomat alapú módszer hatékonysága abban rejlik, hogy a pozícionálási problémát leegyszerűsíti egy minta-illesztési feladatra, melynek komplexitása lényegesen alacsonyabb. Főként emiatt az előny miatt, ezt a technikát alkalmazzák a leggyakrabban Wi-Fi környezetben történő helymeghatározás esetén.

A módszer két fázisból tevődik össze, az első az úgynevezett betanítási fázis, majd ezt követi pozícionálási szakasz. Az elsőben egy adatbázist hozunk létre, melyben a már említett referencia koordinátarendszer pontjaihoz tartozó (RSS – Recieved Signal Strength) értékeket tároljuk, esetünkben decibelben. Ezek az értékek az AP-ok által sugárzott jelerősség értékei, célunk, hogy minél több ponton lemérjük egy mozgó eszközzel az RSS - értékeket. Ha sűrűn vesszük fel ezeket a pontokat, akkor sajnos az első fázis időigényes feladat, hiszen jelentősen sok mérést kell elvégezni. A módszer hátránya még, hogy amennyiben megváltoztatjuk az AP-k számát vagy elhelyezésüket, úgy a teljes területen újra kell futtatnunk a méréseket, azaz az adatbázist újra fel kell töltenünk. A hátrányokhoz sorolnám még azt is, hogy a módszer érzékeny a felhasználói eszköz típusára (sőt még arra is, hogy hogyan tartja a kezében a felhasználó), hiszen ha a referenciamérést egy rendkívül érzékeny eszközzel végeztük és a felhasználó készüléke gyenge Wi-Fi képességekkel rendelkezik, akkor az jelentős mértékben hatással lehet a pozícionálás sikerességére. Sajnos ezen eszközök adóteljesítménye, antennájuk nyeresége sok esetben eltérő. A második, pozícionálási fázis már csak használja az elkészült adatbázist és a felhasználó eszközével megtett mérési eredményt és egy illesztési folyamatot hajt végre, melynek megvalósítására több osztályozó algoritmus is létezik, ilyen például a KNN (K-Nearest-Neighbor) módszer [7], amely *K* darab referenciapontot vesz figyelembe az értékek meghatározásakor, majd ezek átlagolásával ad becslést a keresett pozícióra [4][8][9].

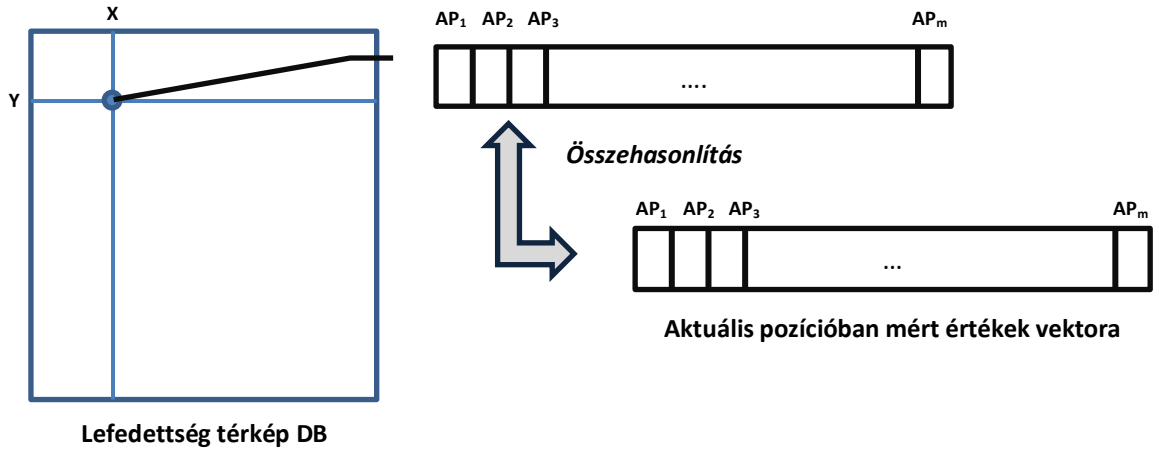
Az egyes pozíciókban vett jel erőssége valamelyest eltérő, valamint ezt az eltérést megerősíti, hogy minden pozíciónál több AP által sugárzott jelről beszélünk, így az eltérés még meghatározóbb. Az is előfordulhat, hogy a térkép egyes részein nem érzékelünk minden AP-ot. Ezáltal a pozíciók jellegzetesek és eltekintve a visszaverődésektől és az esetleges zavaró tényezőktől (pl. nagyobb számú ember elhaladása) ezek ujjlenyomatként tudnak funkcionálni, innen ered a fingerprint elnevezés, hiszen anélkül, hogy ismernünk kellene az AP-ok fizikai elhelyezését, ujjlenyomatszerűen minta-illesztéssel azonosítani tudjuk az adott pozícióban lévő járművet [1].

A pozíció becslés pontosságának növelése érdekében a fent taglalt módszert lehetőségeinkhez mérten kiegészíthetjük, finomíthatjuk. Amennyiben bármilyen kiegészítő információnk van a környezetről és a helyi szabályokról (pl. a járművek csak előre meghatározott útvonalakon közlekedhetnek), akkor ezeket beépítve a rendszerbe nagyobb pontosságot érhetünk el. Ilyen kiegészítő információ lehet, hogy a parkolóban a közlekedés szabályai érvényesek és hogy nem

szabad a parkoló helyeken áthajtani, átvágni a sávokon, túl nagy sebességgel közlekedni. Ezek az információk jól felhasználva, mind hatékonyabbá tehetik a fingerprint alapú módszert.

2.2.1 KNN – K-Nearest-Neighbours

A módszer elnevezése abból ered, hogy a keresett pozíció „szomszédjait" keressük és választjuk ki közülük a legközelebb esőt. Tehát az osztályozó algoritmus kiválasztja azt a *K* darab referenciapontot az összes közül, mely a legközelebb esik az aktuálisan mért RSS vektor osztályához. Vagyis az algoritmus kiszámolja az adott pozícióban mért jelerősség-vektor és az összes, az adatbázisban lévő adatsor értékei közti távolságot. A 2. ábra szemlélteti az összehasonlítást:



2. ábra - KNN – összehasonlítás menete

A távolság meresere alkalmazhatunk szamos modszert, melyek kozul lehetosegunk van meghatározni, hogy az osztályozó algoritmus melyiket használja. Ezek közül a legelterjedtebb az Euklideszi távolság, melyet a következő képen számolunk:

Jelölje s_{ij} a *j*-edik jelszintet az *i*-edik AP-tól, S_i az aktuálisan mért jelszintet az *i*-edik AP-tól. Ahol *i* = 1,2, …, *m*, *j* = 1,2, …,*n*; *m* az AP-ok száma, *n* pedig a minták száma. A távolság S_i és s_{ij} között minden mintára:

$$d_j = \sqrt{\sum_{i=1}^{m}(S_i - s_{ij})^2}, j = 1,2,\ldots,n \quad (1)$$

A mérések során ezt a módszert választottam. Alkalmazható még a Manhattan távolság is mely a Minkowski távolság egy speciális esete:

Manhattan:
$$d_j = \sum_{i=1}^{m}\left|S_i - s_{ij}\right|, j = 1,2,\ldots,n \quad (2)$$

Minkowski:
$$d_j = \left(\sum_{i=1}^{m}\left|S_i - s_{ij}\right|^p\right)^{\frac{1}{p}}, j = 1,2,\ldots,n \quad (3)$$

A számolás eredménye egy olyan adatbázis, ahol a távolságok jelennek meg, azaz a keresett pozíció szomszédjainak távolságai. Ezután kiválasztja távolság szerint növekvő sorrendben a *k* darab legközelebbi szomszédod, melyek tehát a pozíciónkhoz lévő legközelebb eső referenciapontok.

Opcionális lépésként végezhetünk átlagolást így egy átlagosan jó eredményt kaphatunk a *k* darab találat alapján [3].

2.2.2 Valószínűség alapú módszer

A módszer a valószínűség számítás alapjaira helyezi a hangsúlyt a témakörben elterjedt tételek felhasználásával.

Képzeljük el, hogy van *n* pozíció jelöltünk ($\omega_1, \omega_2, \omega_3, \ldots, \omega_n$), ami azt jelenti, hogy van n darab osztályunk. Közülük a legmegfelelőbb kiválasztása a posteriori valószínűség alapján történik. A kezdeti fázisban a közeli AP-k RSS értékeinek méréseit ebben az *n* mintavételezési pozícióban végezzük. *S* vektorban tároljuk az RSS értékeket a valós mérések a második, éles fázisban. A pozíció becslés eredménye az a *w* lesz, amelyiknek a posteriori valószínűsége a legmagasabb.

A döntéshez használt képlet:

$$Döntünk\ \omega_i\ mellett, ha\ P(\omega_i|S) > P(\omega_j|S)\ , ahol\ i, j = 1, 2, 3, \ldots, n. és\ j \neq i \tag{4}$$

Felhasználva a Bayes tételt:

$$P(\omega_i|S) = \frac{P(S|\omega_i)\ P(\omega_i)}{P(S)} \tag{5}$$

Feltételezzük, hogy a környezetünkben lévő AP-k függetlenek, így a teljes valószínűsége egy pozíció jelöltnek kiszámolható egyszerűen az egyes AP-k valószínűségének összeszorzásával.

$$P(S|\omega_i) = P(S_1|\omega_i) \times P(S_2|\omega_i) \times \ldots \times P(S_m|\omega_i)\ , \tag{6}$$

ahol *m* az AP-k számát jelöli, az S_j pedig a vett RSS a *j*-edik AP-tól.

Az eredmények javítása érdekében, számolhatjuk a becsült pozíciót (x,y) a helyi jelöltek koordinátájának átlagaként a posteriori valószínűségüket behozva súlyként.

$$x, y = \sum_{i=1}^{n} (P(\omega_i|S) \cdot (x_{\omega 1}, y_{\omega 1})) \tag{7}$$

Tehát a valószínűség alapú módszer a matematikai lehetőségekre helyezi a hangsúlyt, kihasználva a valószínűség számítás adta lehetőségeket [3] [12].

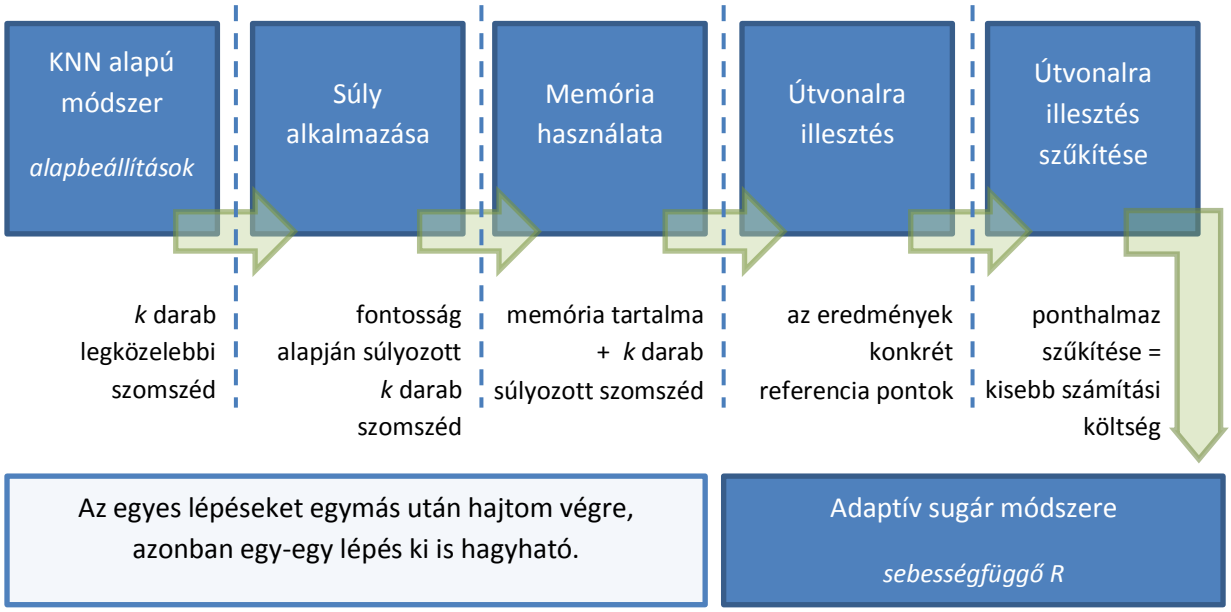
2.2.3 Neurális hálózatok

A neuronokból álló hálózatokat nevezzük neurális hálózatoknak. A neurális hálózatokat többek között azért is fejlesztik, hogy lehetőséget nyújtsanak a nem lineáris ki-bemenettel rendelkező rendszerek tervezésére és számos alkalmazásban hasznosítják előnyeit, például az osztályozó algoritmusoknál vagy becslések végzésénél [11].

A módszer a többrétegű perceptron hálózatok (MLP) használatát kezdeményezi, hiszen ez az egyik legkiemelkedőbb neurális hálózat, amely sikereket ért el az ellenőrzött tanulás algoritmusában. Az ilyen fajta hálózatok kiválóan alkalmazhatóak osztályozási feladatok elvégzésére, bár bonyolultságuk nagy odafigyelést igényel [3] [13].

3 Hatékonyság növelő módszerek

A fent leírt módszerek közül a KNN finomítását választottam, melyet finomítottam néhány lépésen keresztül a pontosabb pozícióbecslés és hatékony működés érdekében. Kezdetben a KNN alap algoritmust alkalmaztam az alapbeállítások mellett, majd a becslés pontosítása érdekében bevezettem a súlyozás módszerét, mely a KNN eredményeire hajt végre egy súlyozást, ennek segítségével az eredményt közelebb kerülhetett az ideális megoldás felé. Majd, hogy ne csak az aktuális helyzet jelszint értékeinek mintaillesztése segítse a pozícionálást, bevezettem a memóriát a rendszerbe, melynek segítségével a becslés hatékonyabbá vált, hiszen a számításba belekerültek a múltbéli helyesnek vélt döntéseink is. Majd a finomítást útra illesztéssel folytattam, hogy a pontosságot növeljem, így egy újabb keresés futtatása mellett lehetőség nyílt arra, hogy a becslés végeredménye egy referencia pontra mutasson, mely a legközelebb áll az algoritmus által helyesnek vélt pozícióhoz. Az útra illesztés módszere finomítható, hiszen az aktuális pozíció illesztésénél nincs szűkség az összes referencia pontra, így azok halmazát szűkíthetjük egy *R* sugáron belülre. A sugár középpontja az előző döntés, ez a megoldás néhány esetben tévedhet, például, ha a jármű hirtelen helyzetet vált, ezért a módszert hatékonyabbá tettem az adaptív sugár bevezetésével, melynek mérete a jármű sebességtől függ, valamint futási időben változik. Az említett lépéseket az alábbi, 3. ábraán szemléltetem, majd alább részletesen kifejtem.

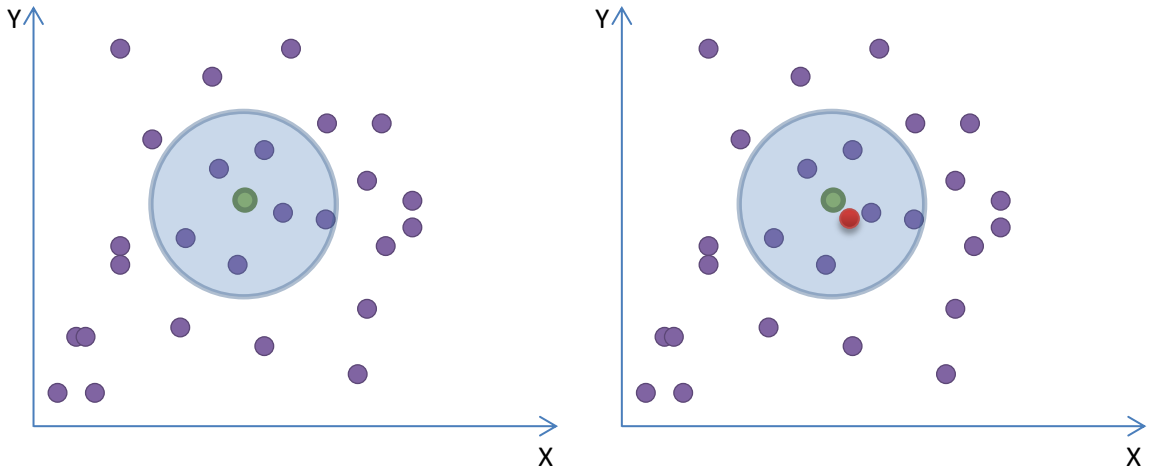


3. ábra - Hatékonyságnövelő módszerek alkalmazásának folyamata

3.1 KNN alapú helymeghatározás

A módszer alapja kiválóan alkalmazható kisszámú referencia adatbázis használata során. Azonban, ha a minta illesztés során egy nagy méretű adatbázisból kell kiválasztani a k-legközelebbi szomszédot, akkor az eredmény nem túl látványos, hiszen így a pontok sűrűbben helyezkednek el, azaz a szomszédos pontok jobban hasonlítanak egymásra. A módszer használata során kiszámoljuk a mért RSS vektor (*M*) és az ujjlenyomat adatbázis *i*. sora (F_i) közti euklideszi távolságot (i=1…L), majd visszaadjuk az első K=10 legkisebb távolságú pozíciót:

$$Dist\ \ M, F_1\ \ = \sqrt{|\alpha_1 - \rho_1|^2 + |\alpha_2 - \rho_2|^2 + \ldots + |\alpha_N - \rho_N|^2} \tag{8}$$



4. ábra - KNN módszer bemutatása, K=6 (bal oldalon: átlagolás nélkül, jobb oldalon: átlagolással)

A 4. ábraán jól látható, hogy önmagában a szomszédok kiválasztása még nem elég a pozícionáláshoz, ezért alkalmazható az átlagolás alapmódszere, mely az egyes eredmények átlagát veszi végeredményként.

A fenti módszert érdemes finomítani. Első ötletem az volt, hogy a kimenetet, azaz a legközelebbi szomszéd keresés eredményét az átlagolás előtt valamilyen logika szerint súlyozni kell. Az így kapott súlyozott átlagtól azt vártam el, hogy pontosabb eredményt adjon.

3.2 Súlyozás

Az átlagolás módszere az egyszerű eseteket kivéve nem hoz megfelelően jó eredményt. Az átlagszámításnál a keresés szerint legtávolabb eső pont is ugyanolyan arányban vesz részt az átlagolásban, mint a keresés alapján legközelebbinek vélt pozíció. Ezek a „rosszabb” eredmények maguk irányába elvonzhatják az átlagot, így a végső eredményt negatív irányba, egy rossz pont felé vezetik.
Az algoritmus könnyen finomítható egy súly bevezetésével, melyet a közelebbinek vélt pontokra alkalmazunk. A keresés felállít egy rangsort a résztalálatok között, mely rangsorra alkalmazhatjuk a súlyozást.

A listában szereplő értékek súlyait a következő képlet adja meg:

$$w_i = 1 - \frac{s^i}{\sum_{j=1}^{K} s^j}, \tag{9}$$

ahol *s* a súly alapja, *i* pedig a KNN táblázat (1. táblázat) megfelelő sorát jelöli.

1. táblázat - KNN táblázat

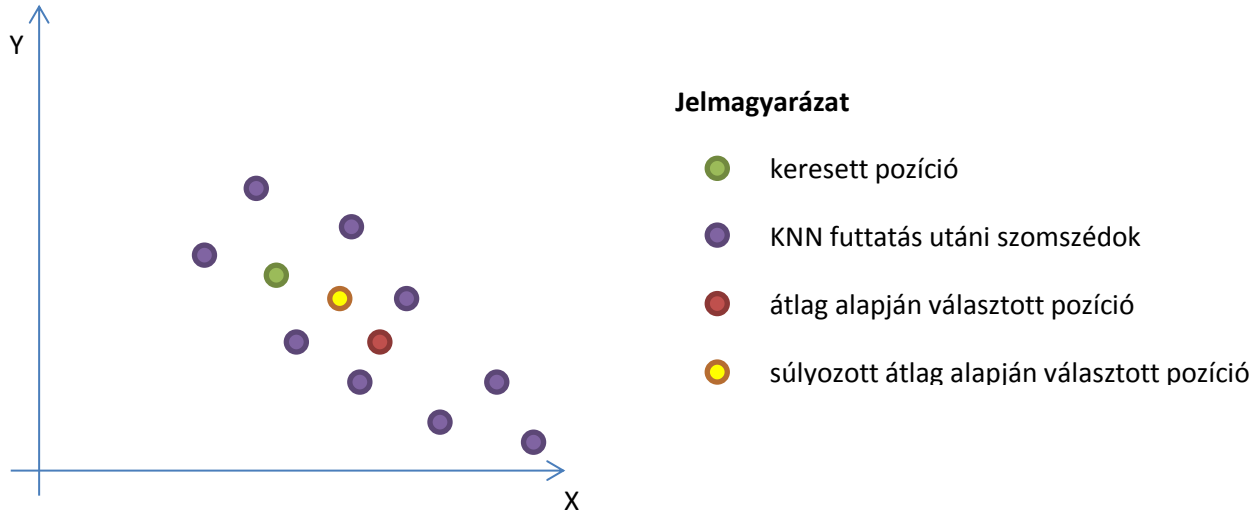
sorszám	koordináta	súly
1.	1. KNN (x,y)	w_1
2.	2. KNN (x,y)	w_2
3.	3. KNN (x,y)	w_3
…		
10.	10. KNN (x,y)	w_{10}

Az 1. táblázatban látható 3. oszlop az egyes szomszédokhoz tartozó súly értékek. Ezek a súly bevezetésével kerültek a táblázatba.

A becsült pozíció (x',y') a következő képlettel számolható:

$$x', y' = \left(\frac{\sum_{i=1}^{K} w_i \cdot x_i}{K}, \frac{\sum_{i=1}^{K} w_i \cdot y_i}{K} \right) \quad (10)$$

A súly alapjának kezdetben a kettőt választottam, majd később ezt választható paraméterré tettem, hiszen bizonyos esetekben a súly alapjának növelése kedvező hatást váltott ki. Az 5. ábraán látható a súlyozás alkalmazása *k*=9 paraméter esetén, jól látható, hogy a lila körök, melyek a KNN listában lévő szomszédok, hogyan szóródnak a keresett cél pozíció körül, az átlag (piros kör) jó eredményt hoz a lilákhoz képest, viszont a súlyozott átlag még közelebbi pozícióra mutat, így láthatóan hatékonyabbá teszi a becslést.



5. ábra - KNN súlyozás alkalmazásával (k=9 esetén)

3.3 Memória

A pozícionálás során végső célunk az, hogy egy útvonalon tudjuk követni a jármű haladását. Az előbb a súlyozás alkalmazásánál csak egy pontra nézve tekintettünk a feladatra. Azonban ha a KNN algoritmus egymás utáni lefutásait tekintem, akkor egymástól messze eltérő, hibás eredmények is születhetnek. Valamint figyelembe kell vennünk, hogy az egyes mérések jelentősen eltérhetnek az adott pozícióhoz tartozó adatbázisban tárolt RSS értékétől a nagy ingadozás miatt. Az volt az ötletem, hogy az egymás utáni pozícióbecslési lépéseket valami logika alapján, együttesen értelmezzük és az egyes lépések egymásutánjával javítsuk a pontosságot, hiszen feltételezhetjük, hogy a jármű nem változtat hirtelen helyzetet, így a következő becsülendő pozíció az előző döntésünk közelében keresendő.

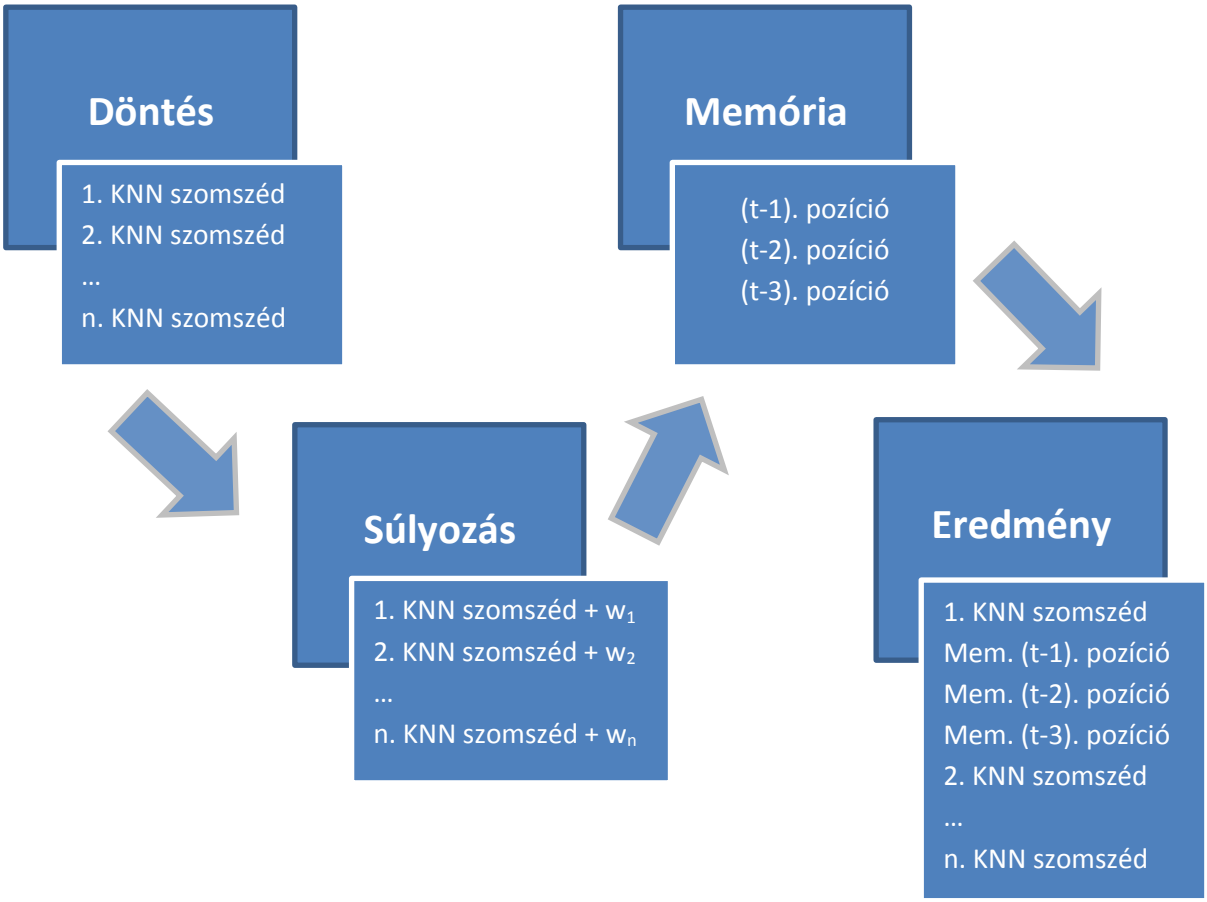
Ez memória alkalmazásával jöhet létre. A memória egy tároló, mely megjegyzi a korábbi döntéseket és azok figyelembe vételével hozza meg aktuális döntését. Így amikor egy a sorban új pozíciót becslünk, nem csak a közeli pontokat vesszük figyelembe, hanem a jármű eddigi pozícióit is. Azaz a súlyozás során nagy súlyt adtam a memóriában lévő pozícióknak és rendezetten csökkenő súlyt a maradék szomszédoknak távolság alapján. Így azt az eredményt várom, hogy az aktuális döntés mindig a már meglévő útvonalhoz közelítsen inkább, mint egy távolabbi ponthoz.

A 2. táblázatban láthatjuk a KNN módszer memóriával történő kiegészítésének eredményét. A táblázat első helyén továbbra is a legközelebbi szomszédok közül az első található, majd a memória méretétől függően a következő elemek, jelen példánkban a 2. és a 3. elem, a memória legfelső elemei, majd ezeket követik a KNN lista további elemei egészen az utolsóig, így a táblázat mérete *k*+memória méret méretű lett.

2. táblázat - Memóriával kiegészítette KNN táblázat

sorszám	koordináta	súly
1.	1. KNN (x,y)	w_1
2.	(t-1) pozíció	w_2
3.	(t-2) pozíció	w_3
4.	2. KNN (x,y)	w_4
5.	3. KNN (x,y)	w_5
…		
12.	10. KNN (x,y)	w_{12}

A 7. ábraán láthatjuk, hogy a súlyozás végrehajtása után a memória, hogyan kapcsolódik be a folyamatba. A memória mérete fontos paraméter, mely befolyásolja a működés hatékonyságát, jelen szemléltető példában a memória méretét 3-ra választottam.



7. ábra - Memória használatának szemléltetése

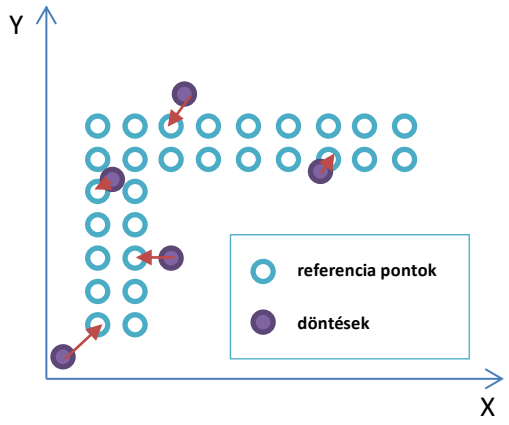
A memória használata valóban közelebb vihet a pontosabb megoldáshoz, azonban ha figyelembe vesszük a helyszíni adottságokat is, akkor még tovább finomítható a becslés. A parkolóházban a megjelölt területen a járművek csak a közlekedés szabályait betartva mozoghatnak, azaz egy előre meghatározott térrészen mozoghatnak így a lehetséges útvonalak száma jól meghatározott.

3.4 Útvonalra illesztés

A parkolóház adottságaihoz mérten a jármű a kijelölt útvonalakon haladhat, ezért arra az útra szűkítjük a döntési tartományt. Ezt segíti, hogy a referencia pontjaink is az útvonalon, elegendően sűrűn helyezkednek el. Ha tehát egy referencia pontra döntünk a végén, azzal útvonalillesztést is végzünk.

Az illesztés ötlete lényegében egy újabb KNN futtatását jelenti *k=1* paraméterrel, hiszen a feladat megkeresni a döntésünkhöz legközelebb eső referenciapontot az euklideszi algoritmus használatával.

Az illesztés elvégzése a referenciapontok segítségével történik, minél sűrűbben vannak elhelyezve annál nagyobb adatbázison kell a keresést futtatni. Amint az eredeti KNN keresés lefutott és megkaptuk a becsült pozíciót, a pozíció koordinátáit összehasonlítjuk az osztályozó algoritmus segítségével a referencia koordinátákkal és keressük a legkisebb távolságú párokat, azaz a legközelebbi szomszédokat. Ennél a fázisnál már nem keresünk több szomszédot és nem használunk átlagolást, csupán egyszerűen *K=1* paraméterrel futtatjuk az osztályozó algoritmust, amely így a legközelebbi referencia ponttal tér vissza.



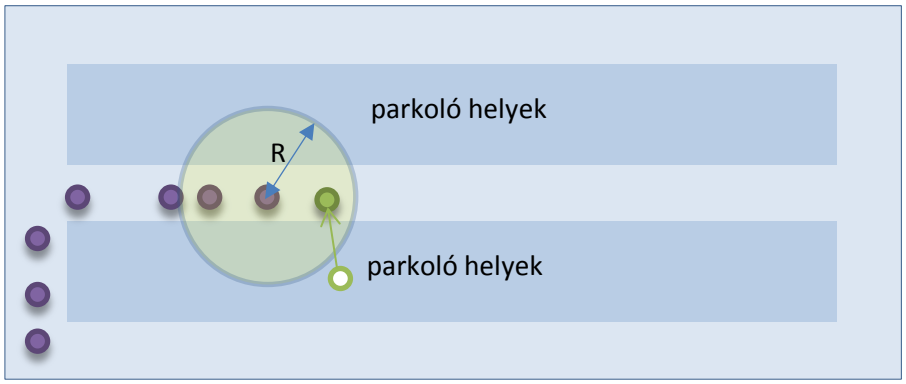
8. ábra - Útvonalra illesztés szemléltetése

A 8. ábraán kék körök jelzik a referencia pontokat, lila körök pedig a döntéseket az egyes pozíciókra. Az útvonalillesztés módszere az illesztést úgy végzi el, hogy az aktuális lila ponthoz éppen legközelebb lévő referencia pontra illeszti a döntést, így a végeredményben minden döntés referencia pontra kerül.

3.4.1 Illesztés szűkítése (R sugarú körben)

Az útvonalra illesztés bizonyos helyzetekben felesleges pontokat is figyelembe vesz, ezzel egy felesleges „zavarást” viszünk a rendszerbe, melyet egy könnyű módszerrel kiszűrhetünk. Például a valóságban nem lehetséges, hogy egy jármű 1-2 másodperc alatt átkerüljön a térkép egy távoli pontjába vagy egy másik párhuzamos sávba, ezt korlátozni tudjuk a sugár bevezetésével.

Ötletem az volt, hogy az éppen aktuális keresett pont útra illesztése előtt a lehetséges referencia pontok halmazát szűkítsük le a következő módon. Az előző döntés köré rajzoljunk egy R sugarú kört, majd válasszuk ki a referencia pontok közül azokat, amelyek ebben a körben helyezkednek el. A becsült pozíció útra illesztésénél csak ezeket a referencia pontokat vesszük figyelembe és a KNN keresést is ezen a szűkített halmazon futtatjuk. Így azt várom, hogy az algoritmus futási ideje kedvezőbb legyen és az előre biztosan hamisnak vélt megoldásokat kiszűrjük. Ötletemet a 9. ábra szemlélteti:



9. ábra - Illesztés szűkítése R sugarú körben

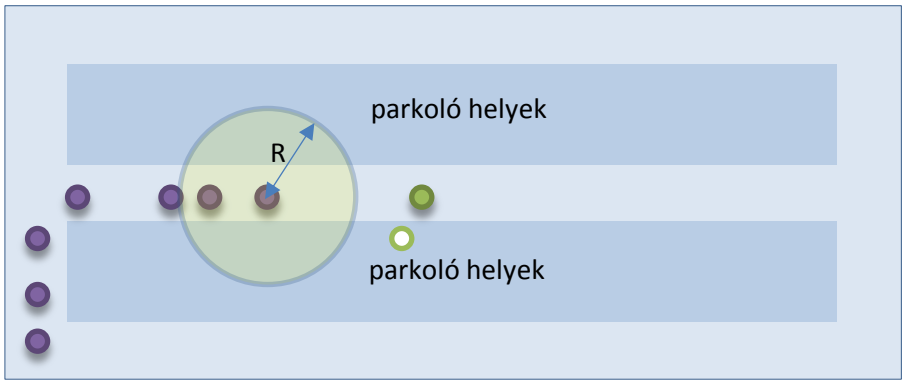
A fenti működést akkor tudjuk biztosítani, amennyiben a felhasználó betartja a közlekedési szabályokat, ha ezeket áthágja, akkor a módszer nem hoz jó eredményt, hiszen nem tudja követni a jármű mozgását, hiszen az átmegy egyik oldalról a másikra, eközben nem várhatóan nagy távolságot

megtéve. Ennek kiküszöbölésére megoldás lehet a szűkítés időnkénti feloldása valamekkora valószínűséggel, így a jármű nem várt helyváltoztatását korrigálhatjuk.

A sugár méretének megválasztása lényeges lépés, hiszen ha túl nagyra választjuk, nem érjük el vele a kívánt hatást, hiszen továbbra is megengedjük, hogy a becsült pozíció akár egy lehetetlen messze lévő pontra mutasson. Viszont ha túl kicsire választjuk, akkor pedig túlságosan visszafogja a haladást, így amíg a jármű már előrébb haladt, mi még hátrébb érzékeljük.

3.4.2 Adaptív sugár módszere

Abban az esetben, ha a jármű az egyenes szakaszon felgyorsít előfordulhat, hogy az előre meghatározott sugárméret kicsinek bizonyul és a jármű túlszalad az előző helyzet köré rajzolt körön. Így az előző módszer hátrébb becsüli a pozíciót, mint a valós helyzet, amiatt, hogy leszűkítettük az értékkészletét. Ahogy a 10. ábraán is látható, a sárga pontot nem tudjuk az útvonal jó helyére illeszteni, csak a jármű helyzetétől visszább.



10. ábra - Fix sugár problémája

Erre a problémára az adaptív sugár ötletét találtam ki. Az ötlet a sebesség függő sugár méreten alapszik. Mint ahogy azt a GSM navigációk egy részénél láthatjuk, amikor nagy sebességgel közlekedünk az autónk körüli kör, melyben pozícionál minket a rendszer elég nagy, viszont amikor lelassítunk a kör egyre kisebb. Az adaptív sugár módszerénél is hasonlóan oldom meg a sugár méretének változtatását. Amikor a jármű nagy sebességgel közlekedik, azaz a két legutolsó pozíció távolsága nagyobb, akkor a következő pozíciót nagyobb sugárral becslem. Alapvetően a mintavételi időt 1 másodpercben határoztam meg. Így a sebesség a következőképpen adódik:

$$v = c \cdot \frac{s}{T} \quad ,ahol\, c = konstans, T = mintavételi\, idő \tag{11}$$

A konstans megválasztása a környezet elrendezésétől függ, hiszen ha a parkolóhelyek nagyméretűek és az utak egymástól nagyobb távolságra vannak, akkor a konstans értékét növelnünk kell. A jelenlegi helyzetben a c = 2 beállítást választottam.

4 Szimulációs vizsgálatok

Ahhoz, hogy a fent taglalt módszerek hatékonyságát ellenőrizzem létre kellett hoznom egy szimulációs környezetet, ahol lehetőség nyílik a különböző technikák implementálására, összehasonlítására. Majd ezeket a módszereket más-más paraméterekkel és beállításokkal is le kell mérni, hiszen az eredmény csak akkor mondható reprezentatívnak.

4.1 Szimulációs környezet

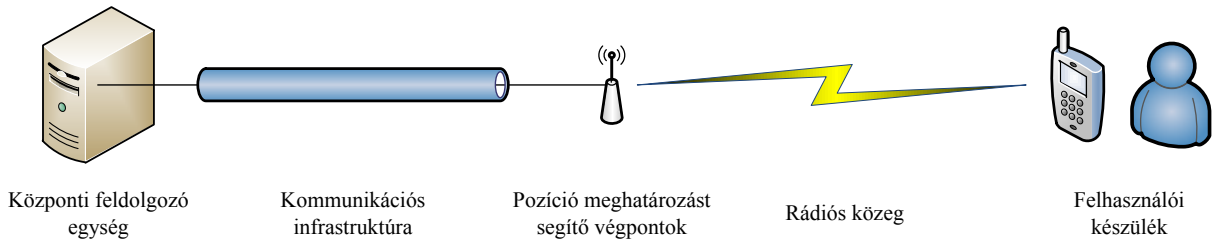
A környezet megválasztásánál figyelembe kellett vennem, hogy a fenti módszereket könnyen meg lehessen valósítani és a munka nagy részét ne az implementálás vegye el, hanem a technikák hatékonyságának növelése, összehasonlítása.

Választásom a Matlab [10] szoftverre esett, mert a nagyméretű lefedettség térkép adatbázis és a segéd mátrixok kezelésében egy hatékony lehetőséget kínál. A Matlab környezet rendelkezik egy knnsearch() beépített függvénnyel, mely pont a korábban említett KNN algoritmust alkalmazza a paraméterek megfelelő paraméterezettsége mellett.

Ugyan szimulációs környezetben dolgoztam, de mégis valós adatokat használtam fel, melyeket a helyszínen vettek fel több mérés alkalmával, tehát az adatbázisok valós helyszíni mérések alapján készültek, ezeket töltöm be a Matlab rendszerbe mátrixok formájában.

4.1.1 Szolgáltatási környezet bemutatása

A projekt, melyen a pozícióbecslés hatékonyságát szeretném növelni, az Alle bevásárlóközpont parkolóházában helyezkedik el. A projekt célja egy parkolást segítő rendszer megvalósítása, melynek felépítésében a három legfontosabb elem a központi feldolgozó egység, a kommunikációs infrastruktúra és a pozíció meghatározást segítő végpontok. Ezeken túl van még egy elem, mely elengedhetetlen a rendszer működése szempontjából, ez a felhasználók mobil készüléke (vagy bármely WLAN interfésszel rendelkező eszköz). A rendszer minden egyes eleme jól meghatározott feladatot lát el. A 11. ábraán látható az általános rendszerterv felépítése, az egyes elemek helye a rendszerben, valamint az egymással való kapcsolatuk, kommunikációjuk működése.



11. ábra - Általános rendszerterv

A következőkben röviden ismertetem a különböző funkcionális eszközöket és feladatukat.

4.1.1.1 Funkcionális egységek rövid bemutatása

A központi feldolgozó egység egy speciális számítógép, melyen futtat egy alkalmas operációs rendszert, melynek mérete kicsi és a jól meghatározott feladatára specializálódott valamint az eszközök kezelését el tudja látni.

A központi feldolgozó egység feladata, hogy a beérkező mérések eredményét feldolgozva az algoritmust lefuttassa az adatbázisán és meghatározza a jármű pontos helyzetét az adatok alapján,

majd az eredményeket feldolgozza és a pozícionálásnak megfelelően valamilyen irányba navigálja a felhasználót a cél elérése érdekében. Ehhez nem csak az AP-k által küldött adatok feldolgozása a feladata, hanem a háttér adatbázis, valamint a térkép információk feldolgozása is.

Feladata még a memória kezelése és az AP-k konfigurációjának segítése.

A kommunikációs infrastruktúra feladata, hogy összeköttetést biztosítson a központi feldolgozó egység, és a pozícionálást segítő végpontok között, valamint biztosítania kell a felhasználók és a központi egység közötti kommunikációt is. Elképzelhető, hogy a kiépített rendszert nem csak a helymeghatározásra szeretnénk használni, hanem a pozicionáló rendszerrel párhuzamosan vezeték nélküli internet hozzáférést is biztosítunk a felhasználóknak, ebben az esetben további feladatként az adatforgalom továbbítása, jelzésüzenetek továbbítása is a feladata lesz. Ez a jelenlegi demó rendszernek nem képezi részét.

A pozíció meghatározást segítő végpontok, jelenleg a már többször említett Acces Point (vezeték nélküli hozzáférési pont)-ok. Feladatuk a jelszintek mérése és továbbítása a központi feldolgozó egység felé.

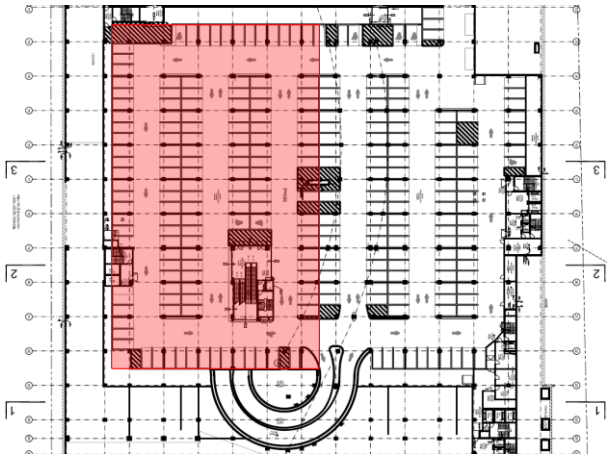
A felhasználó készülékek feladata, hogy a felhasználók kapcsolódhassanak a pozícionáló rendszerhez, valamint, hogy a navigációs lépéseket, információkat megjelenítse. A feladat bonyolultságának függvényében szükséges lehet kliens oldali szoftverek telepítésére (pl. térkép megjelenítése miatt).

4.1.1.2 Demó szektor bemutatása

Az iParking projekt keretén belül a demó szektor az Alle bevásárló központ P1-es parkoló szintjén található. A demó szektor kiválasztásánál fontos szempont volt, hogy a kijelölt területen minél több döntési helyzetet lehessen megvizsgálni, valamint, hogy elég nagy legyen ahhoz, hogy több AP megléte legyen szükséges a terület lefedéséhez, hiszen jelszintek változása így válik vizsgálhatóvá.

Három különálló sort fed le a demó szektor, melyek elején, valamint végén döntési pontok vannak definiálva, ugyanis itt kell megfelelően navigálnunk az autóst, hogy befordulјon vagy haladjon tovább a következő kereszteződésig. Az 12. ábraán látható útvonalak (ellentétben az ott látható nyilakkal) egyirányúak, ezzel megkönnyítve a navigálást, hiszen így tudjuk, hogy a jármű mindig csak előre haladhat, amennyiben betartja a szabályokat, így például könnyedén mérhetjük a sebességét a mérések eredményei alapján.

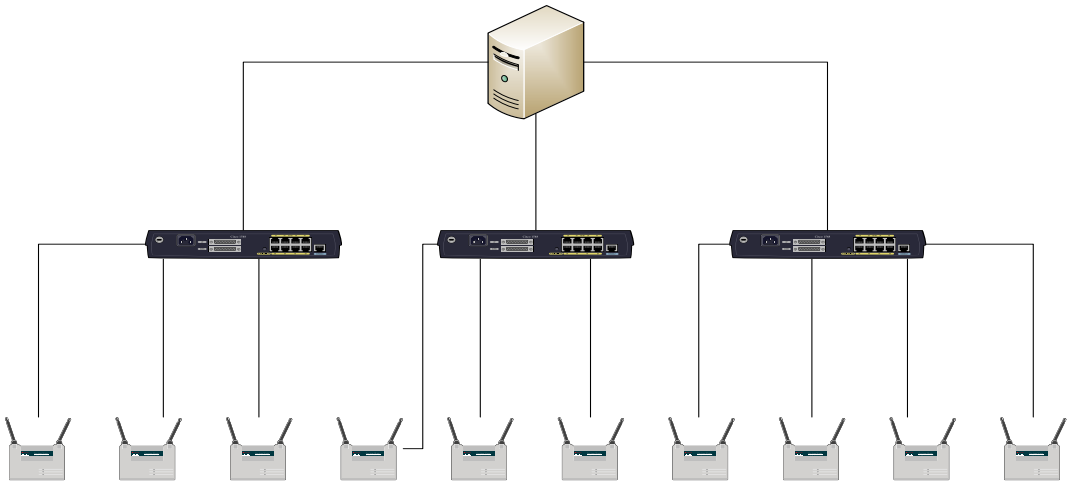
A demó szektornak csak egyetlen bejárata van, mely szempont is volt a tervezés során, ez azért fontos, mert a felhasználó itt fog felkapcsolódni a pozícionáló rendszerre és ezt a pozícióját vesszük kiindulópontnak.



12. ábra - Demó szektor szemléltetése

4.1.1.3 A rendszer logikai felépítése

A rendszer logikai felépítését a 13. ábra szemlélteti. A központi feldolgozó egység három 8 portos switch-en keresztül kapcsolódik a pozíció meghatározást végző eszközökhöz. Ez a központosított felépítés megkönnyíti az adatok feldolgozását, a pozíció meghatározását. Nincs szükség arra, hogy az AP-k nagy teljesítményű eszközök legyenek, mivel nekik csak a jelszintek mérése a feladatuk, komoly számításokat nem végeznek. További előnye, hogy egy pozíció meghatározást végző eszközök meghibásodása esetén nem a teljes rendszer válik működésképtelenné, hanem csak a pozicionálás pontossága csökkenhet. Természetesen a hátrány is ebből adódik, még pedig, ha a szerver hibásodik meg, a teljes rendszer működésképtelenné válik. Ezt megfelelő biztonsági óvintézkedések, esetlegesen redundáns központi feldolgozó egység beiktatásával lehet kiküszöbölni. További előny, hogy a rendszer egy helyen (a szerveren) konfigurálható, nem szükséges az egyes eszközök külön-külön történő beállítása, paraméterezése, ez könnyen megtehető a szerveren megfelelően kialakított scriptek, folyamatok alkalmazásával.

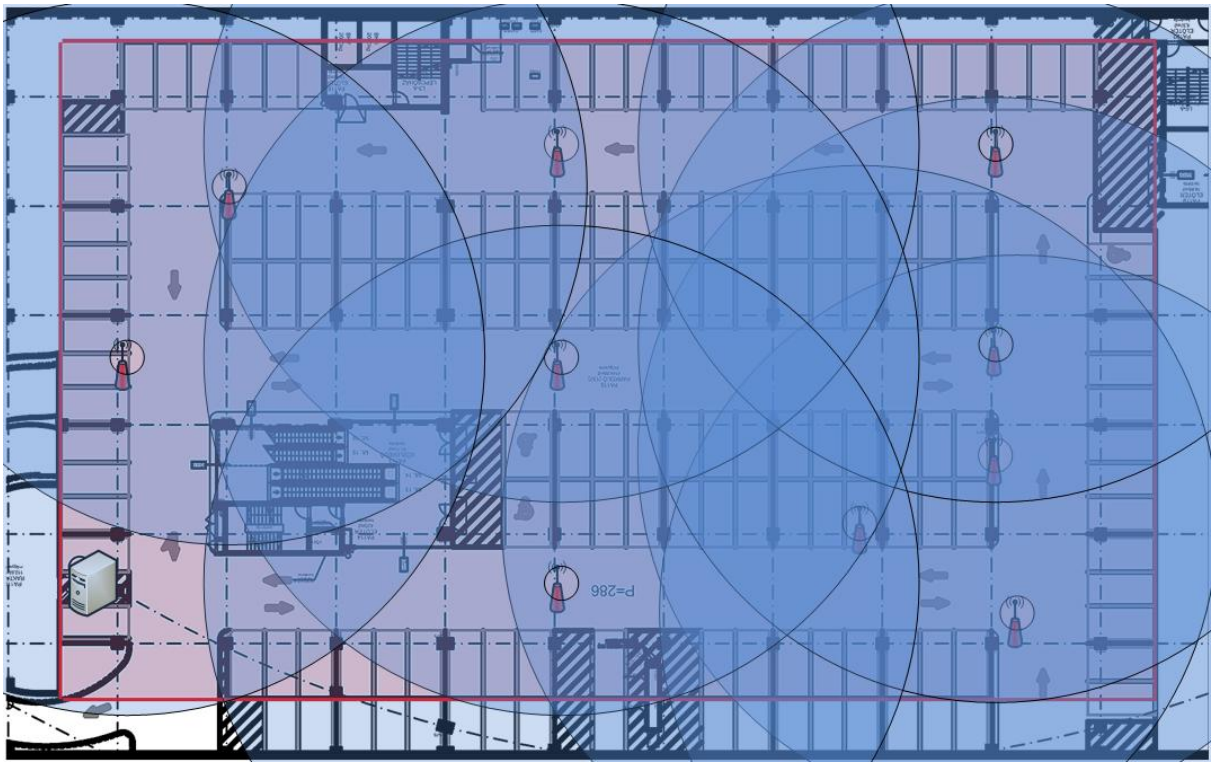


13. ábra - Rendszer logikai felépítése

4.1.1.4 Access Pointok fizikai elhelyezése

Az elhelyezés kapcsán az volt a cél, hogy a terület bármely pontján a lehető legtöbb eszköz jele vehető legyen. Ideális körülmények mellett már három eszköz jeléből meg lehet határozni az eszköz pozícióját, de a demó szektort nem mondhatjuk ideálisnak. A parkolóházban jelen lévő beton oszlopok és falak nagymértékben befolyásolják a jelek visszaverődését és ezzel a pozícionálás hatékonyságát, pontosságát. Valamint a térben helyet foglaló parkoló autók és helyüket változtató emberek, ember csoportok is nagyban változtathatják az egyes pozíciókban mérhető jelek erősségét. Például egy 2,4 GHz-es jel emberen történő áthaladása nagyjából 3dB-es csillapítást eredményez, mely távolságban mérve körülbelül 10-15 méteres ugrást eredményez. Ezek miatt kialakítási kritériumként szerepel, hogy a demó szektor minden pontjában legalább 5 AP jelét kell tudnunk venni.

A fentiek és egyéb paraméterek figyelembe vételével egy szimulációs szoftver segítségével meghatározták, hogy a kijelölt demó szektor optimális lefedéséhez 10 db. vezeték nélküli hozzáférési pont szükséges, melyek elhelyezését és a tér lefedettségét 14. ábraán láthatjuk.



14. ábra - Demó szektor, AP-k lefedettség térképe

4.1.2 Helyszíni mérések felhasználása

Kezdetben egy szoftverrel generált lefedettség térképet használtam, majd hamar átálltam a helyszínen készített valós mérésekből álló térkép adatbázisra. Rendelkezésemre állt néhány gyalogos és autós mérési eredmény is Excel táblázat formájában. Például a 2012.04.12-én rögzített gyalogos adatbázis 172 referenciapontot tartalmaz. Az *l* pozícióhoz tartozó ujjlenyomat értékek:

$$F_l = \ \alpha_1, \alpha_2, \dots, \alpha_N \ ^T, \tag{12}$$

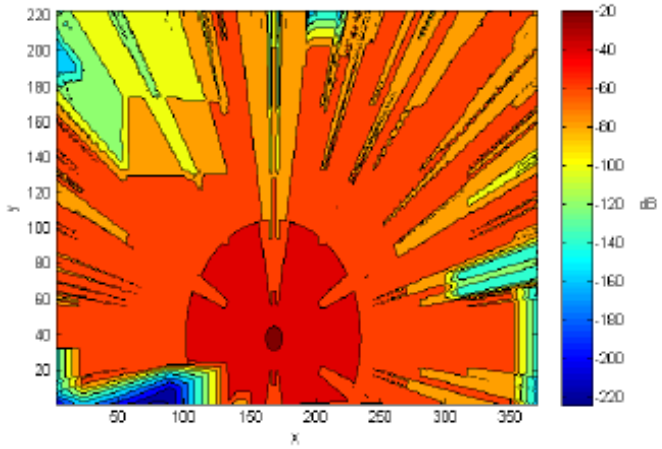
ahol α_i az *i.* AP-hoz tartozó RSS érték az ujjlenyomat adatbázisban és *N* az AP-k számát jelöli. Az ujjlenyomat adatbázis felépítése a 3. táblázatban látható:

3. táblázat - Ujjlenyomat adatbázis szemléltetése

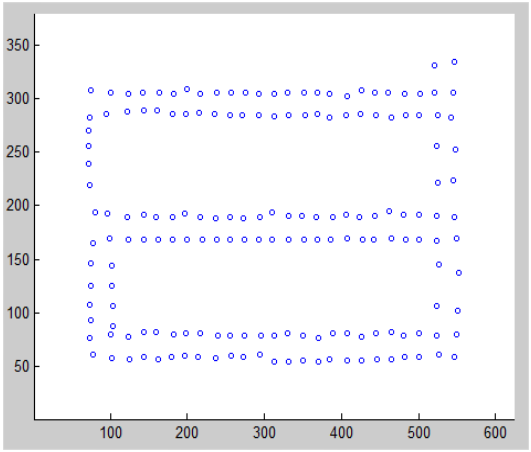
koordináta (x,y)		α_1	α_2	…	α_N
1	1	-45 dB	-55 dB	…	-66 dB
1	2	-50 dB	-62 dB	…	-72 dB
…					

A 3. táblázat első két oszlopa az *x*,*y* koordinátákat tartalmazza, ezek azonosítják az adott pozíciót, melyen a mérési eredményt rögzítettük. A további oszlopok (összesen *N* darab) pedig az egyes AP-ok által vett jelerősséget tartalmazzák decibel értékben rögzítve.

A referencia pontok lefedik a parkoló-rész járművek által használható teljes területét. Az utak mentén előrehaladva két-két referenciapont helyezkedik el, egymástól néhány méterre. Melyet a 12. ábra jobb oldalán látható, Matlab szimuláció futtatása során generált grafikon szemléltet:



15. ábra – Lefedettség térkép az első AP-ra nézve



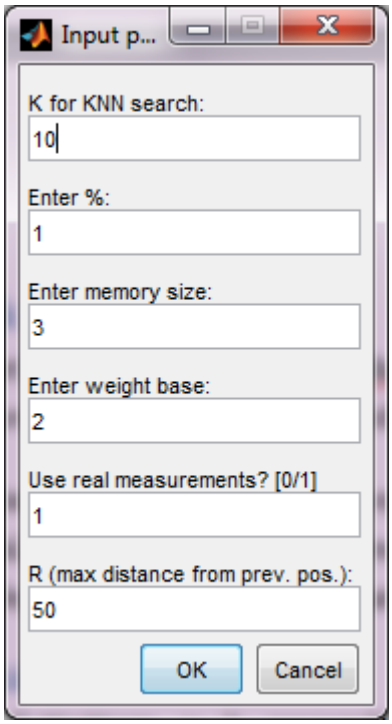
16. ábra - Referenciapontok elhelyezkedése

A 16. ábraán látható lefedettség térkép egy AP-ra nézve bemutatja, hogy a 15. ábraán látható területet az egyik AP hogyan fedi le, az egyes területeken milyen jelerősségi érték vehető. A lefedettség térkép elkészítésénél, segítségemre volt egy szimulációs Matlab program, mely jelterjedési modellekre támaszkodva elkészíti egy tér lefedettség térképét.

4.2 Eredmények

A valós eredményeket felhasználva megírt Matlab programomban számos futtatási paraméter állítható be, melyek konfigurálásával a fenti módszereket mértem le.

A paraméterek közül a 17. ábraán az első a *k*, melynek változtatásával a KNN keresés szomszéd listájának méretét adhatjuk meg. Az alapbeállítás *k=10.* A következő paraméter a torzítás százalékban, ez adja meg, hogy a valós méréseket milyen mértékben torzítsuk, ezzel behozva a rendszerbe véletlen hibát, amely a valós környezetben is előfordul az esetleges reflexiók miatt. A harmadik paraméter a memória mérete, alapbeállításként a három szerepel. Majd megadhatjuk a súlyozott átlag képzésnél használt súly alapját mely alapbeállításban 2, de akár 8-ra is állíthatjuk. A „Use real measurements? [0/1]" kérdésre válaszolva megadhatjuk, hogy valós vagy szimulált méréssel számoljon a program. Kezdetben a szimulációs program segítségével legenerált lefedettség térképet használtam, majd áttértem a valós mérési eredményekre. Végül az útvonalra illesztés módszerének paraméterét állíthatjuk be, amely az *R* sugár, ezt növelve egyre nagyobb mértékben tágítjuk a referenciapont halmazban való keresés határait.



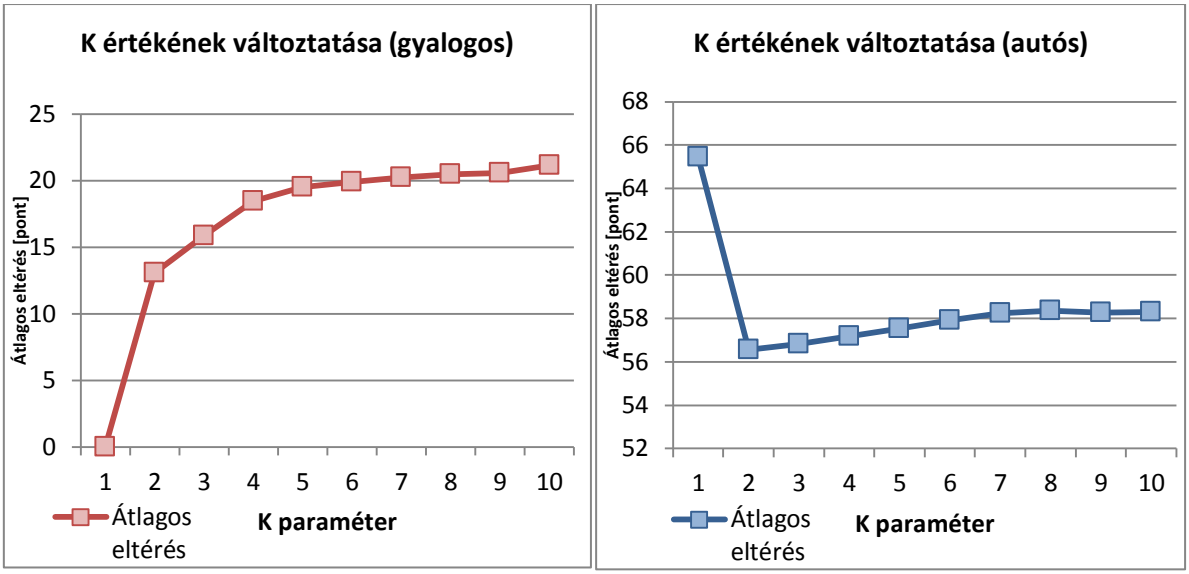
17. ábra - A Matlab program paraméter ablaka

Az eredmények összehasonlítására létrehoztam egy mérőszámot, mely az átlagos eltérést számolja az eredeti pozíciótól, a különböző paraméterek mellett. Ez a mérőszám hivatott arra, hogy konkrét számokkal követhető legyen az egyes lépések hatékonysága, az algoritmus finomítása. A mérőszám az eltéréseket számolja ki az egyes döntések és a keresett pozíciók között, majd ezeket átlagolja az úthossz függvényében. A különböző verziókra (memória, memória nélkül, illesztés, illesztés nélkül) külön mérőszám van a programban. Általános számítás az eltérés osztva az úthosszal.

4.2.1 KNN, finomítás nélkül

A KNN módszer önmagában futtatva egyedül a *k* paraméter beállítását teszi lehetővé. Ez alap metódus, ezt fogom a későbbiekben lépésről lépésre finomítani. Tehát a KNN algoritmust futtattam a

k paraméter változtatása mellett. A 18. ábraán a pozícióbecslések átlagos eltérését láthatjuk a keresett pozíciótól, a *k* paraméter 1-től 10-ig történő növelése mellett.



18. ábra - Átlagos eltérés K növelése mellett (bal oldalon gyalogos, jobb oldalon autós mérést alkalmazva)

A fenti grafikonokra tekintve rögtön szemet szúr az az érdekes jelenség, hogy *k*=1 esetén az átlagos hiba 0 vagy kiemelkedően nagy értéket vesz fel. A jelenségre a mérést elvégezve lettem figyelmes. A jelenség hátterében az van, hogy a mérési beállítások elvégzésekor a referencia pontok jelszintjeit ugyanabból a fájlból töltöttem be, mint amely fájlt használok arra, hogy szimuláljam a jármű mozgását és pozícióját. A jármű pozícióit a program 1%-ban torzítja, azonban ha nincs csak egy szomszéd, akkor az átlagolás során az első és egyetlen szomszéd jön ki győztesnek, ezért a *k*=1 paraméter esetén a rendszer teljes pontossággal eltalálja a pozíciót.

Megfigyelhetjük, hogy a *K* paraméter változtatásával, 2 és 10 közötti értékei mellett nincs kiugró változás.

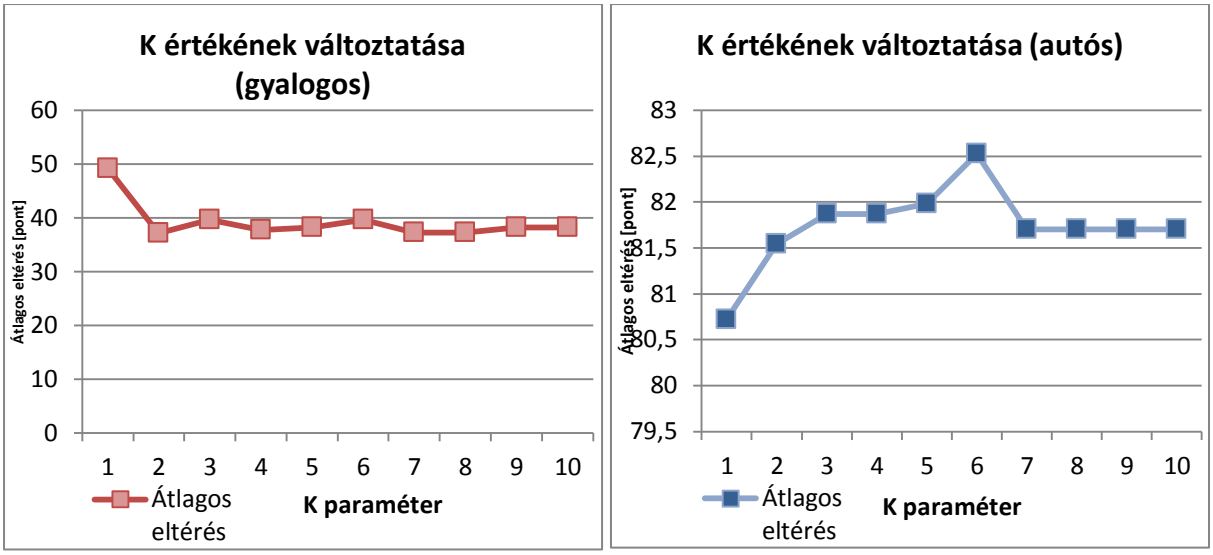
Elmondható az eredmények alapján, hogy a *K* paraméter értékének növelésével az átlagos eltérés is növekszik, ez azzal magyarázható, hogy a KNN listában egyre több vélt szomszéd kerül bele és a távolabbak magukhoz húzzák az átlagot.

A gyalogos mérést tekintve *K=2* mellett az átlagos eltérés 13,1 pont (≅3 m), *K=4* esetén 18,4 (≅3,5 m), *K=10* esetén pedig 21,1 (≅4,2 m). Látható, hogy *K* 2 és 10 értékei között az átlagos eltérés több mint 1 métert változik.

Az autós esetben *K=2* mellett az átlagos eltérés értéke 56,5 (≅11 m), *K=6* mellett 57,9 (≅12 m), valamint *K=10* esetén 58,2 (≅11,6 m). Tehát ebben az esetben a *K* paraméter változása az átlagos eltérést maximum 1 méterben befolyásolja.

A fent írt távolságok méterben a helyszín adottságai alapján kerültek megállapításra. A referencia térkép is a valós demó szektor arányos másolata, a referencia pontok távolsága kb. 20 képpontnyi, ami megközelítőleg 4 méter távolság. A fenti becsléseket méterben tehát ezek szerint adtam meg.

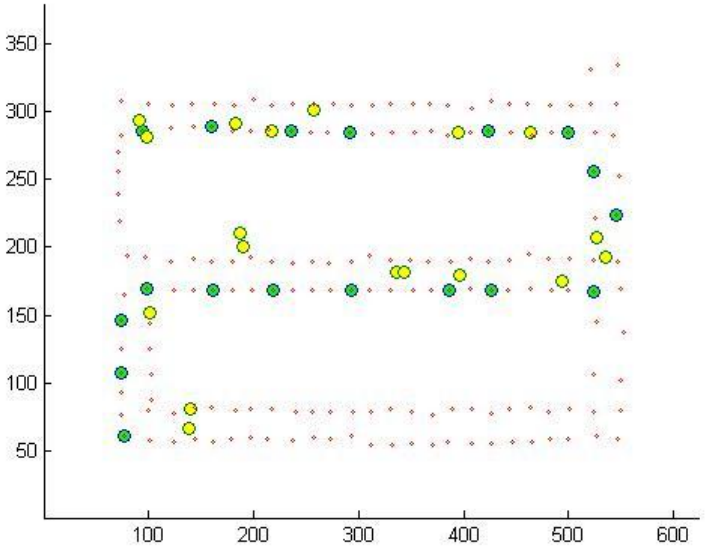
Megismételve a mérést helyes beállításokkal, azaz olyan referencia térképet választva, amely különböző a jármű pozícióinak adatbázisától, a következő eredményt kaptam.



19. ábra - Átlagos eltérés K értékének változtatása mellett

A 19. ábraán jól látható, hogy az átlagos hiba a *k* értékének növelése mellett nem mutat tipikus változást, tehát egyértelműen nem növekszik vagy csökken. Leolvashatjuk a grafikonról, hogy gyalogos mérésnél *k*=1 esetén 50 körül van az átlagos eltérés, mely kb. 10 méter, majd a paraméter értékét növelve ez javul körülbelül 2 métert. A paraméter 2 és 10 értékei között az átlagos hiba 37 és 40 pont között mozog, a különbség az egyes esetek között kevesebb, mint 1 méter.

A Matlab program futása végén, az alábbi térképet rajzoltatom ki, melyen a zöld pontok jelentik a keresett, eredeti pozíciókat, a sárga pontok a KNN algoritmus *k*=5 paraméter futtatásával számolt döntések sorozata.

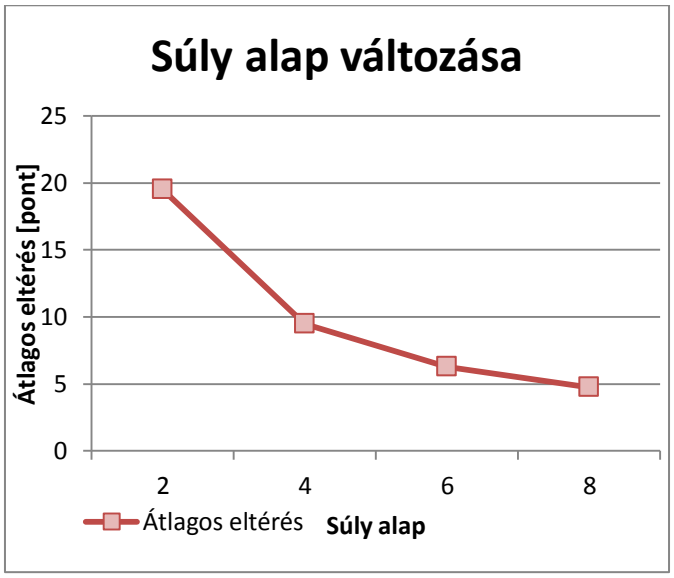


20. ábra - KNN módszer eredménye K=5 esetén (zöld-eredeti, sárga-becsült pontok)

Jól látható a 20. ábraán, hogy sok esetben elég jó becslést adott a módszer, azonban látható az is, hogy az útvonal egyes részein nagyobb hibát vétett, ez azonban a valós helyzetben csupán kis tévedéshez vezetett volna.

4.2.2 Súlyozás

A súly alapjának változása komoly hatással van a pozíció-becslés hatékonyságára, hiszen a súlyozással határozzuk meg, hogy hányadik szomszéd milyen erővel hasson a végleges döntésben. A k paraméter értékét 5-re választottam a súly alapját pedig az alapbeállítástól egészen 8-ig növeltem. Az alábbi ábrán látható a mérés eredménye.

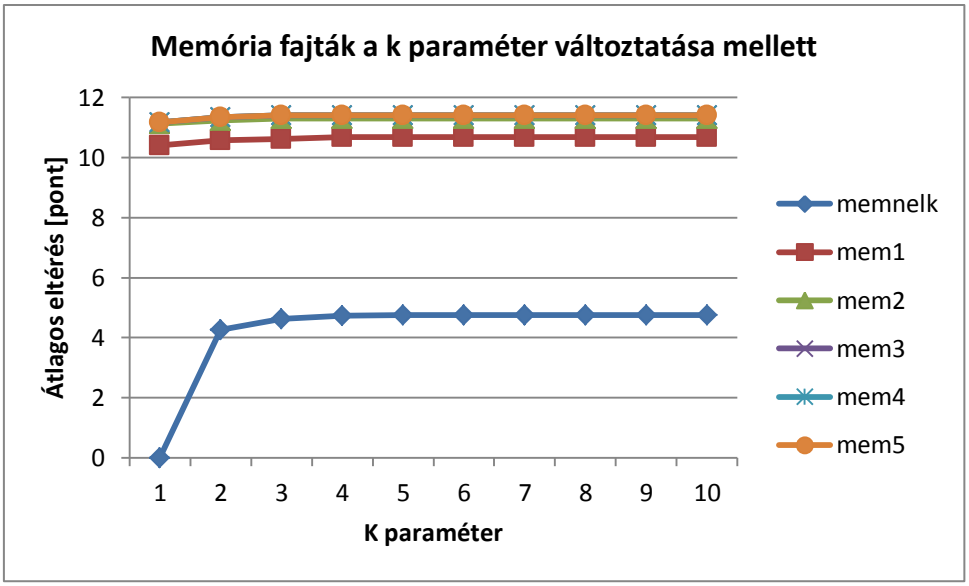


21. ábra - Súly alapjának változása k=5 mellett

Megfigyelhető, hogy az áltagos távolság az eredeti pozíciótól folyamatosan csökken a súly alapjának növelésével. Az alapot 8ra választva már 5 pont alá csökken az átlagos eltérés, ami a kezdeti 19,5-höz képest jelentős eltérés. Tehát látható, hogy a súly használatának megjelenése és annak finomítása jó hatással van a pozícióbecslés hatékonyságára nézve.

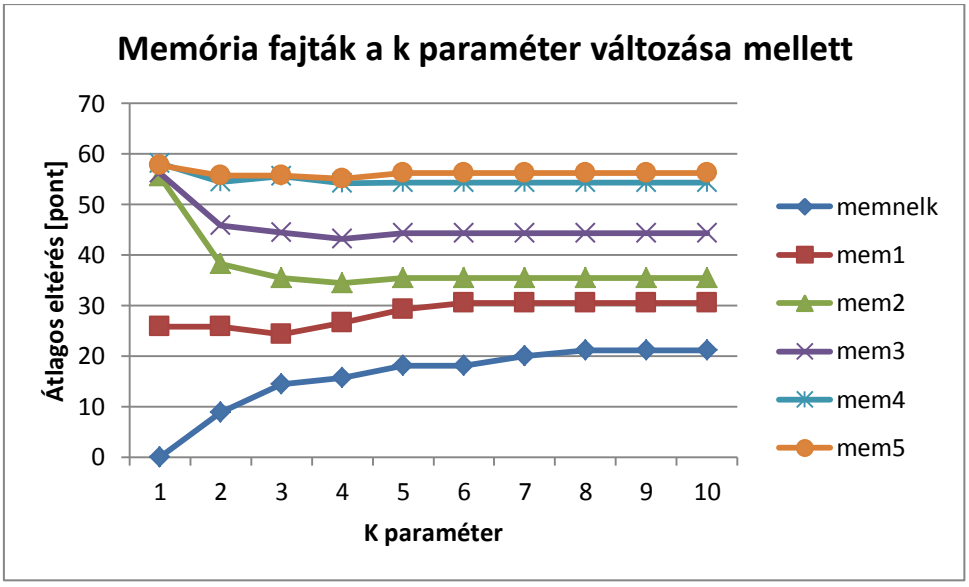
4.2.3 Memória

Az algoritmus finomításaként memóriát adunk a rendszerhez, mely a memória méret paraméter segítségével finom hangolható. A memória célja, hogy pontosítsa a következő pozíció becslését, azaz ne engedje, hogy az algoritmus kizárólag az éppen aktuális körülményeket vegye figyelembe, hanem számításba vegye a jármű múltbéli pozícióját és annak segítségével jó becslést tudjon adni a jelenlegi helyzetére. Mérésem során a k értékét 1..10 skálán változtattam és megfigyeltem az átlagos eltérés alakulását a memória méretének növelése mellett. Az eredmény az alábbi ábrán látható.



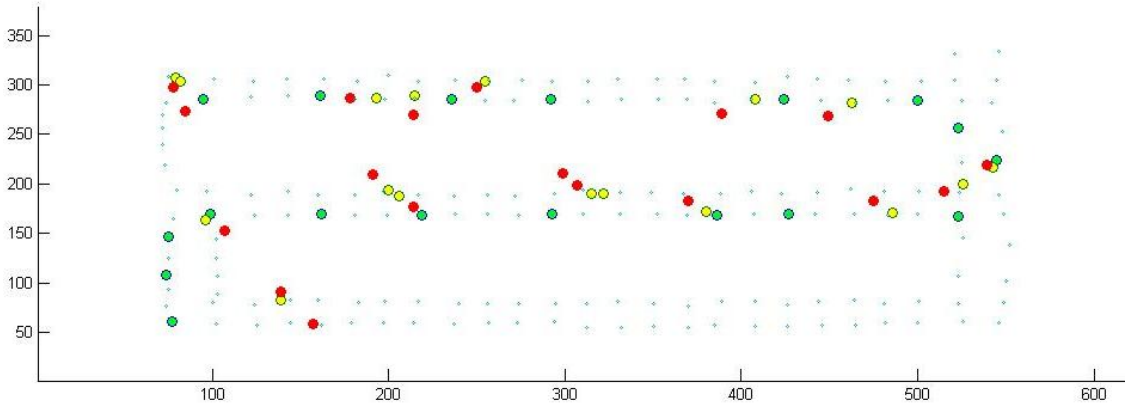
22. ábra - Memória méretek és átlagos eltérés kapcsolata a K paraméter változtatása mellett (súlyalap = 8)

A kék vonal jelzi, hogy memória alkalmazása nélkül az átlagos eltérés nem jelentős és a K paramétertől szinte függetlenül 4-5 értékek között helyezkedik el. Ábrázoltam azt az esetet is amikor a memória paraméterét a méretet állítjuk, láthatóan a nagyobb memória a jelen mérési helyzetben rosszabb eredményt produkál. A túl nagy súly a fenti mérést elronthatja, feleslegesen nagy prioritást adva a lista első elemének, ami pontatlan koordinátát tartalmazhat. A fenti mérésben a korábban, memória alkalmazása nélkül legjobbnak vélt súly alapot használtam (s=8).



23. ábra – Memória méretek és átlagos eltérés kapcsolata a K paraméter változtatása mellett (súlyalap = 2)

Kisebb súlyalapot választva szemléletesebb grafikont kaphatunk. Ebben az esetben a memória mérete érzékenyebb a K paraméter változtatására, viszont az átlagos eltérés maximuma már majdnem eléri a 60 pontot (12 m). Itt is azt lehet mondani, hogy a memória nélküli eset hozta a legjobb eredményt (kék vonal) hiszen növekvő K esetén növekvő hibával az átlagos eltérést 21 pontban (≅4 m) maximalizálta. Ha a memória méretet 1-re választjuk az átlagos hiba 20 és 30 közé esik. Majd a memória növelésével egyre nagyobb az átlagos eltérés értéke.



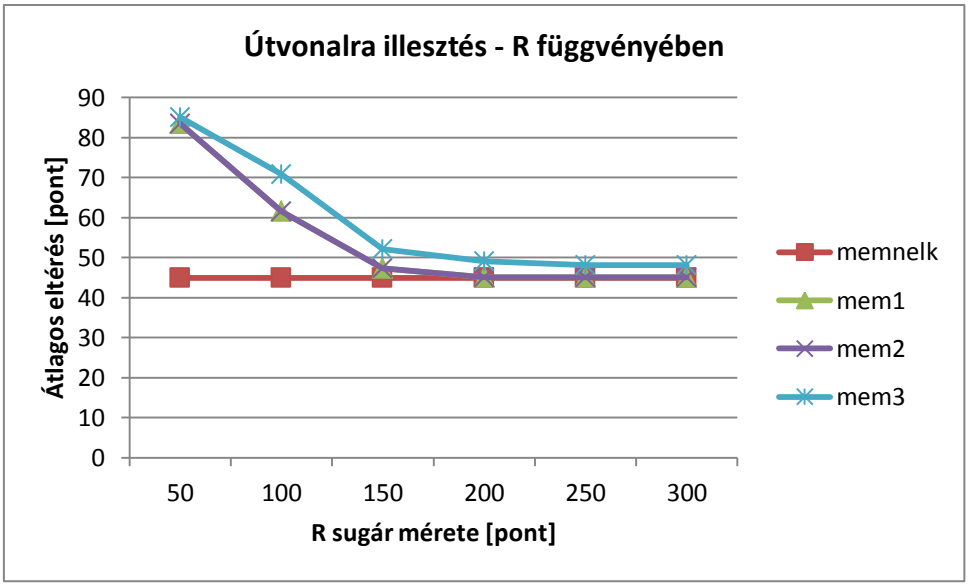
24. ábra - Pozícióbecslés eredménye - K=5, s=8

A fenti grafikonos eredményből kiválasztottam egyet, melyen a *k*=5, s=8 paramétereket állítottam be. Az eredményt a 24. ábra mutatja be, zölddel jelöltem az eredeti pontokat, amelyeket keresünk, sárgával a memória nélküli és pirossal a három méretű memóriával rendelkező esetet. Látható, hogy a memóriával kissé közelebb kerülünk az eredeti útvonalhoz és a becslés az útvonal mentén folytonosabbnak látszik.

4.2.4 Útvonalra illesztés

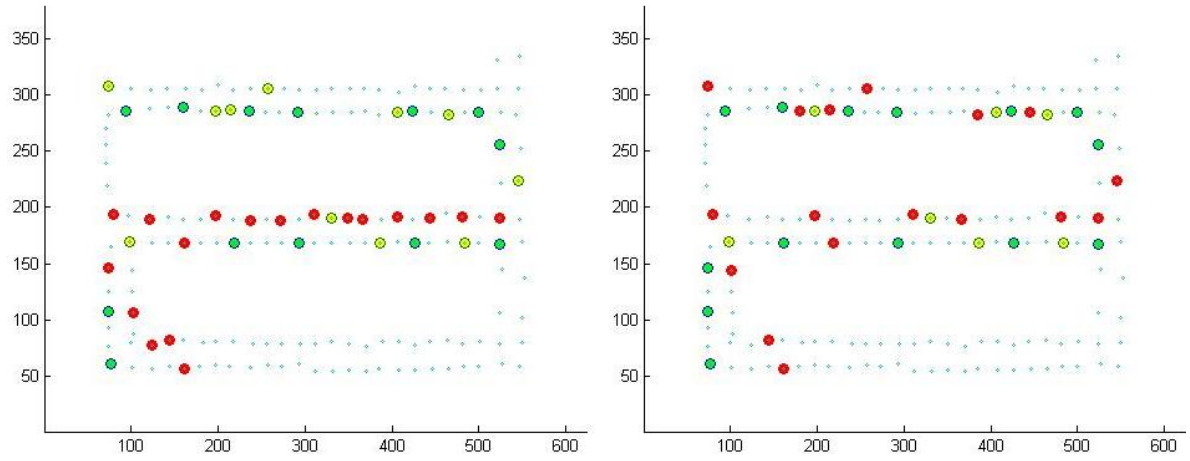
Az illesztés hatékonyságának kulcsa az ötlet magja, mely a szűrő sugár mérete. A méret változtatásával az illesztés is pontosabbá válhat, hiszen előfordulhat az az eset, hogy a sugár méretének alul becslése a jó döntések lehetőségét szűri ki. Persze ennek ellentettje is kialakulhat, hiszen ha túl nagy sugarat választunk, akkor a módszer lényegét, a szűrést, szüntetjük meg, mert hasonlóan a szűrés nélküli verzióhoz majdnem az összes referencia ponton végezzük a keresést.

Ennél a mérésnél is a különböző memória mérettel futtatott algoritmus kimenetét láthatjuk a sugár növelése mellett. A súly alapját a korábban legjobbnak talált alapon (salap=8) hagytam, a k paramétert középső állásba 5-re állítottam. A 25. ábraán a memória nélküli esetet látjuk a grafikon alsó részén, azaz kis átlagos eltérést mutatva. Majd felfelé haladva láthatjuk a memória növelésével kapott eredményeket. Az ábra alapján a kis sugár nem hoz kedvező kimenetelt, ez azzal magyarázható, hogy túlságosan lefogja a keresési tartományt és így a módszer csak kis mértékű hatékonyságra képes. A nagy sugár szintén nem jó, mert feleslegesen nagy tartományban hagyja keresni az algoritmust, ahogy ez az ábrán is látható. Ugyanis a 150 pontos sugár után már érdemi változás az eredményeket illetően nem látszik, azaz megállapítható, hogy a jelen mérési környezet mellett a 150es sugárnál nem érdemes nagyobbat választanunk, viszont azzal jó eredményt érhetünk el a többihez képest.



25. ábra - Útvonalra illesztés R változása mellett

A fenti eredményeket a Matlabban ábrázoltam először *R*=50, majd *R*=150 esetén, útvonalra illesztéssel és anélkül. A kimenet, amit kaptam a 26. ábraán látható.



26. ábra - Pozícióbecslés útvonalra illesztéssel R=50 (balra) és R=150 (jobbra) mellett

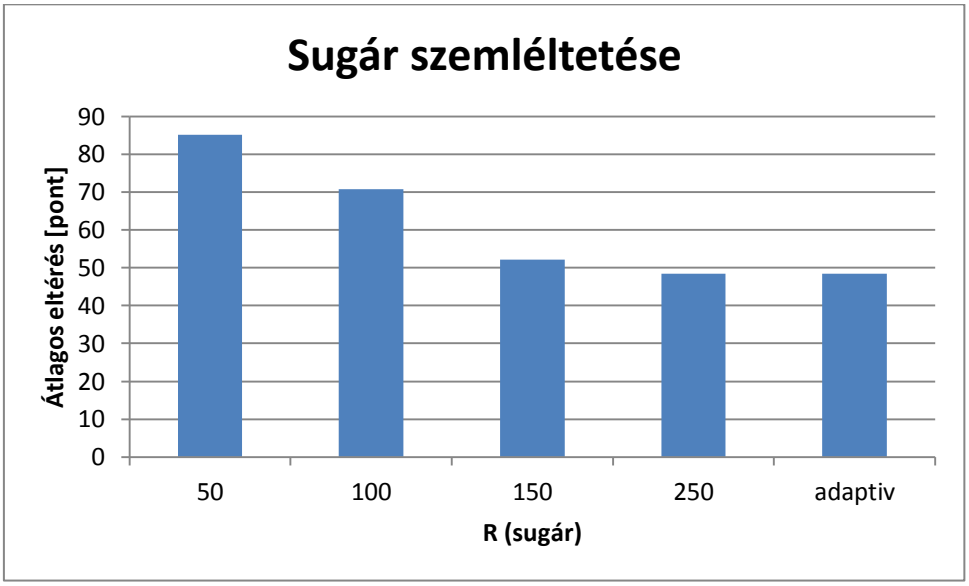
A bal oldali képen láthatjuk az 50 pontnyi sugárral készült pozícióbecslés eredményét. A sárga pontok a memória nélküli, a piros pontok a memória használatával készített becslést mutatják be, a zöld pontok az eredeti pozíciók. A 25. ábraáról mondottakat rendkívül alátámasztja, amit láthatunk. Az 50es kisméretű sugár visszafogja az algoritmust így az nem tud hatékonyan működni, míg a jól beállított 150es sugár éppen annyira nagy, hogy jó irányba vigye a becslést és a navigációt, emellett hatékony megoldás, mert nem hagy szükségtelenül nagy területet a keresés futtatására.

4.2.5 Adaptív sugár

Az előbbi mérésnél a sugár mérete egy algoritmus lefutása alatt nem változott, a futás elején beállított fix *R* sugárral finomítottuk a pozícionálást. Adaptív sugárról akkor beszélhetünk, ha a sugár

mérete futási időben rugalmasan változik. Jelen esetben a sebességtől függ a sugár változása. Ha a jármű növeli a sebességét a sugár is megnövekszik, így lehetőséget nyújtva arra, hogy egy mintavételi idő alatt nagyobb távolságra juthasson és ezt a pozícióbecslésünk is követni tudja.

A mérés során a *k* paramétert középállásba 5-re állítottam, a súly alapja a bevált 8, a sugár mérete pedig sebességfüggő.



27. ábra - Sugár szemléltetése, összehasonlítás az adaptív sugárral

A 27. ábraán nagyon világosan látható, a túl kicsi sugár választása nem jó választás. Hiszen rossz eredményt szül, a közepes sugár már jobb, de még nem éri el az ideális lehetőséget, tehát finomítható. A jól beállított 150-es sugár már hatékonynak mondható és az átlagos hiba minimumát is elérte, az ennél nagyobb sugár választása szinten rossz döntés, hiszen látható, hogy a 250-es sugár ugyan 100-al nagyobb, mint a 150-es sugár, így feleslegesen számoltat az algoritmussal, de mégsem tudja az átlagos hibát lejjebb vinni. Míg az adaptív sugár módszerénél a sugár mérete nem állandó, így az algoritmus sohasem számol feleslegesen, mert az aktuális sebességtől függően nagyítjuk vagy szűkítjük a számítási teret. Emellett eléri a lehető legjobb eredményt, javítva a 150es sugár által produkált eredményen. A mérés eredménye végső soron nem a legkedvezőbb 9,6 méter pontosságú, azonban a módszer finomítása, hatékonyságának növelése az adaptív sugár kapcsán egyértelműen kimondható.

5 Összefoglaló

A parkolóházak, bevásárlóközpontok és a környezet tudatos életvitel elterjedésével egyre nagyobb jelentőséggel, kihívással bír a pozícióbecslés hatékonyságának növelése belső térben. A Wi-Fi technológia alacsony használati költségének köszönhetően és az iParking rendszer felhasználó központúságának jóvoltából a beltéri navigáció, így az általam vizsgált rendszer használatának népszerűsége vélhetően növekedni fog. Célom az volt, hogy az Alle bevásárló központ parkoló házában kialakított iParking demó szektorban lévő pozícionáló rendszer kapcsán a beltéri pozícióbecslés hatékonyságát növeljem ötleteim és a kezdeti módszerek lépésről lépésre történő finomítása által. Az alapvető kiinduló algoritmus a KNN osztályozó függvény volt, melyet az ujjlenyomat alapú helymeghatározó módszer melletti döntés miatt választottam. A KNN egyszerű minta-illesztéssel dolgozik, mely kedvező számunkra és jól finomítható.

A bemutatott módszereket, melyek célja a becslés folyamatos finomítása, hatékonyságának növelése, valós mérési eredményeken alkalmaztam. A használt jelszínt adatbázist az iParking helyszínén az Alle P1 szintjén a demó szektorban történt gyalogos és autós mérések alapján építettük fel. Ezekre az adatokra alapozva létrehoztam egy Matlab programot, mely a KNN alap algoritmusból indul ki, mely a Matlab egy beépített függvényén (knnsearch) alapszik. Majd a programot folyamatosan fejlesztve, paraméterezhetőséget biztosítva, az alap elgondolást folyamatosan bővítettem, finomítottam a projekt céljához kapcsolódó ötleteim megvalósításával. A programot képessé tettem az egyes verziók futtatására, így összehasonlíthatóvá váltak az általam létrehozott módszerek. Az összehasonlítás láthatóságára a referencia pontok alapján létrehoztam egy referencia térképet, melyet a Matlab programmal grafikusan megjeleníthető módon egy koordinátarendszerben rajzoltattam ki. Így láthatóak a pozícionálás lépései és az eredmények alakulása a különböző esetek mellett. A számszerű eredményesség mérésére létrehoztam egy mérőszámot, mely az átlagos eltérést jeleníti meg a vélt pozíciók és az eredeti pozíciók között, ezzel lemérve az egyes technikák hatékonyságát egymáshoz viszonyítva.

Az eredményeket elemezve és összehasonlítva, arra jutottam, hogy a finomítás lépései minden alkalommal valamilyen szempontból mérhetően jobb eredményt hoztak. Az első módszer esetén az átlagos hiba még 3 és 11 méter között volt, majd a finomítás és hatékonyságnövelési lépések előre haladtával ez az érték egyre csökkent, míg végül már csak 1-2 méter között ingadozott. A valós helyszínen, a parkolóházban egy parkolóhely kb. 2-3 méter széles. Amennyiben a fenti 3-11 méter közti hibával számolunk, a rendszer még működőképes, hiszen a hiba folytán maximum 3-4 parkolóhellyel előrébb vagy hátrébb pozícionál a rendszer. Ezt a felhasználó korrigálni tudja, hiszen már láthatja a számára kijelölt szabad helyet maga előtt. Azonban a kanyarodásnál ez a hiba problémát okozhat. Az utóbb említett 1-2 méteres hiba viszont igen jónak mondható, hiszen így egy parkolóhelyen belüli a hibatávolság, azaz alig észrevehető és a kanyarokat is biztonsággal kezeli, az esetleges kiugró hibáktól, visszaverődésektől eltekintve.

Az eredmények elemzése után az a véleményem, hogy a pozícionáló rendszeren futó algoritmust sikerült hatékonyabbá tenni a finomítások által, olyannyira, hogy az a felhasználók számára is szembetűnő lehessen. A hatékonyság számszerűen mérhető és jól tükrözi a kezdetben leírtakat. Az algoritmus a fentieken túl is tovább finomítható, de véleményem szerint, már csak apró lépéseket tehetünk, például a valamilyen valószínűség melletti illesztési terület szűkítés kikapcsolásával.

Irodalomjegyzék

[1] Reza Zekavat, R. Michael Buehrer: “Handbook of Position Location: Theory, Practice and Advances”, John Wiley & Sons, p. 1192, August, 2011

[2] Németh László Harri, Kis Zoltán Lajos, Szabó Róbert: WLANpos: Wi-Fi alapú beltéri helymeghatározó rendszer, Budapesti Mûszaki és Gazdaságtudományi Egyetem, Távközlési és Médiainformatikai Tanszék, Híradástechnika - LXII. ÉVFOLYAM 2007/8

[3] Tsung-Nan Lin, Po-Chiang Lin: Performance Comparison of Indoor Positioning Techniques based on Location Fingerprinting in Wireless Networks, Graduate Institute of Communication Engineering; National Taiwan University, Taipei, Taiwan, 2005 International Conference on Wireless Networks

[4] Binghao Li, James Salter, Andrew G. Dempster, Chris Rizos: Indoor Positioning Techniques Based on Wireless LAN, in Proc. of the First IEEE International Conference on Wireless Broadband and Ultra Wideband Communications, Syndey, Australia, 2006

[5] Takács György: Helymeghatározás mobiltelefonnal és mobil hálózattal, Pázmány Péter Katolikus Egyetem, Információs Technológiai Kar, 2008

[6] Hui Liu, Houshang Darabi, Pat Banerjee, Jing Liu: “Survey of Wireless Indoor Positioning Techniques and Systems”, IEEE Transactions On Systems, Man, And Cybernetics—Part C: Applications and Reviews, Vol. 37, No. 6, November 2007

[7] Belur V. Dasarathy: “Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques”, IEEE Computer Society, ISBN 0-8186-8930-7, p. 550, 1991

[8] Yongxiang Zhao; Huaibei Zhou; Meifang Li; Ruoshan Kong: "Implementation of Indoor Positioning System Based on Location Fingerprinting in Wireless Networks," Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on , vol., no., pp.1-4, 12-14 Oct. 2008

[9] Khodayari, S.; Maleki, M.; Hamedi, E.: "A RSS-based fingerprinting method for positioning based on historical data," Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2010 International Symposium on , vol., no., pp.306-310, 11-14 July 2010

[10] MATLAB and Simulink for Technical Computing, www.mathworks.com

[11] S. Haykin: "Neural Networks - A Comprehensive Foundation," Second Edition, Prentice-Hall, 1999.

[12] R. 0. Duda, P. E. Hart, D. G.: "Pattern Classification," Second Edition, John Wiley, 2000

[13] Horváth Gábor (szerkesztő): Neurális hálózatok és műszaki alkalmazásaik, Műegyetemi Kiadó, Budapest, 1998. 6-112. oldal.