# MRF alapú előtérdetekció és objektumkövetés Lidar pontfelhő szekvenciákon

Horváth Csaba, Molnár Dömötör, Benedek Csaba, Szirányi Tamás

MTA SZTAKI Elosztott Események Elemzése Kutatólaboratórium
{vezeteknev.keresztnev}@sztaki.mta.hu

**Absztrakt.** Cikkünkben előtérkiemelést és célpontkövetést megvalósító módszert mutatunk be, fix pozícióban álló forgó Lidar eszközzel készült pontfelhő szekvenciák elemzéséhez. Azért, hogy valós idejű működést tegyünk lehetővé, henger felületre vetítjük a Lidar-pontfelhőt, elkészítve egy $360^0$-os, két-dimenziós mélységképet. A feldolgozást nehezítő tényezők, mint például a diszkretizált pixelrácsból adódó kvantálási hibák, a szenzor kalibrációjából származó pozíció torzítások, és a növényzet periodikus mozgása miatt létrejövő háttér fluktuáció, szignifikáns csökkentésére javasolunk egy dinamikus Markov véletlen mezős (MRF) modellt, amely térbeli és időbeli leírókat egyaránt használ. A kiértékelés valós Lidar adatokon történt, video felügyeleti és forgalom figyelés alkalmazásokhoz kapcsolódó mérési sorozatokon. Végül a kapott módszer alkalmazhatóságát is bemutatjuk többcélpontú követés és dinamikus helyszínrekonstrukció feladataira.

## 1. Bevezetés[1]

Az automatikus videó-felügyeleti rendszerek egyik fontos alapfeladata az előtér elkülönítése a háttértől, mivel az előtér területek gyakran tartalmazzák a feldolgozás szempontjából fontos régiókat. A pontos objektum-maszk hasznos információt nyújt a megfigyelt területről, amely hatékonyan felhasználható többek között emberek vagy járművek detekciójához, követéshez, esetleg biometrikai azonosításhoz vagy esemény-elemzéshez.

Szegmentációs feladatokban a mélységkép sorozatok geometriai információtartalmuk miatt előnyösebb jellemzőket hordoznak, mint a hagyományos videó szekvenciák intenzitás, szín vagy textúra leírói. A Time of Flight (ToF) kamerák vagy Lidar szenzorok használatával a külső fényviszonyoktól függetlenül tudunk mélységképeket rögzíteni, és ezzel elkerülhetjük a sztereó képalkotás bizonyos esetekben felmerülő hibáit is. Az eszközökből érkező adatokból a távolság-információ reguláris két-dimenziós pixelrácsra vetül, amelyen különböző kiforrott képfeldolgozási algoritmusok alkalmazhatók, mint például a Markov véletlen mezős (MRF) szegmentációs módszerek.

A forgó többszenzoros Lidar rendszerek (Rotating Multi-beam Lidar System, RBM-Lidar), mind például a Velodyne HDL 64-E, $360^0$-os látószöggel képesek

[1] A cikk részben a WDIA konferencián került publikálásra [2].

információt szolgáltatni a környezetről, a hatékonyabb adat-feldolgozáshoz pedig a 3D Lidar pontokat ezután gyakran vetítik egy henger alakú mélységképre [3] [4]. Ebben az esetben a kapott mélységképek függőleges felbontása megegyezik a szenzorok számával, míg vízszintes felbontása a forgás sebességétől függ. Továbbá, ez a megfeleltetés gyakran nem egyértelmű: esetenként több lézer visszaverődés tartozhat ugyanahhoz a pixelhez. Ez a probléma részben javítható ha minden pixelnél a megfigyelt háttér-értékekre több módusú eloszlást alkalmazunk [3], de sűrű háttérmozgások esetén a hibák gyorsan összegződnek, amit okozhat például a szeles időben periodikusan mozgó növényzet.

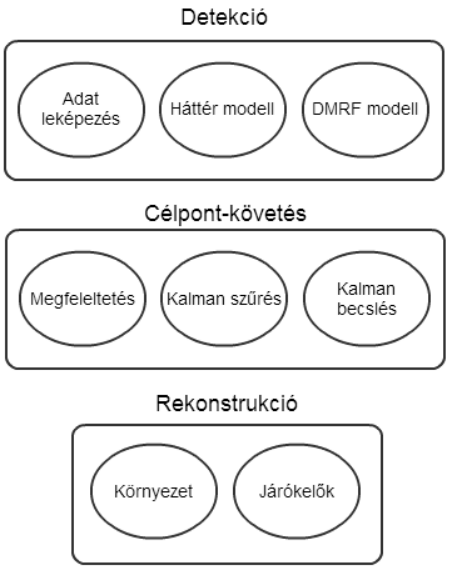
A fent említett nehézségeken túl azért, hogy megkapjuk az Euklideszi koordinátákat [5], az RMB-Lidarból kapott nyers távolság, forgási szög és szenzor index értékeknek egy erősen nemlineáris kalibrációs lépésen kell átesniük, és ezért a kapott pontok a vetítés után erősen inhomogén sűrűségű pixeltérképet eredményezhetnek. A fenti hibák kiküszöbölésére, [4] közvetlenül a mélységképből emelte ki az előtér objektumokat mean-shift szegmentációval és folt-detekcióval. Azonban tesztjeink során több esetben is megfigyeltük, hogy olyan környezetekben, ahol több mozgó és statikus objektum van, a járókelők többször kerülnek egy szegmentált régióba valamilyen szomszédos háttér-elemmel.

A pontfelhő mélységképre vetítése helyett egy másik lehetséges megoldás, hogy az előtér detekciót a három-dimenziós térben végezzük el. Az ilyen elven működő objektum szintű technikák általános célja, hogy meghatározzuk a járókelőket befoglaló téglatesteket [6], ahelyett, hogy minden előtér pontot felcímkéznénk a felhőben, utóbbi lépés azonban fontos lehet különböző események elemzésénél, például csontváz modell illesztésnél. MRF jellegű technikákat gyakran alkalmaznak 3D térbeli pont szomszédságok esetében [7], de ezek pontossága rossz, ha kis sugarú szomszédságokat használunk, egyébként pedig a számítási komplexitás növekszik gyorsan.

Cikkünkben egy hibrid technikát mutatunk be Lidar pontok előtér-háttér klasszifikációjára egy pontfelhőn, melyet rögzített pozíciójú RMB-Lidar rendszerből nyerünk. A bemutatott módszer a számítási szempontból kritikus térbeli szűrés problémáját egy, a két dimenziós mélységképen értelmezett MRF modellel oldja meg, továbbá a diszkretizációs hibákat kiküszöböli a 3D pozíciók és a 2D címkék együttes figyelembevételével. Térbeli előtér modellt használva szignifikánsan csökkentjük a háttér mozgásából adódó ponthibákat, melyeket többnyi-re fakoronák mozgása eredményez. Előtérkiemelő módszerünket összehasonlítjuk három referencia megoldással, és kiértékeljük az általunk készített 3D pontfelhő Ground Truth (GT) annotációs eszközzel. Végül az előtérdetekciós megoldásunk alkalmazhatóságát bemutatjuk többcélpontú követés és dinamikus helyszínrekonstrukció alkalmazásokra. A megvalósított elemek áttekintése az 1. ábrán látható.

## 2. Adat leképezés

Tegyük fel, hogy az RMB-Lidar rendszer $R$ vertikálisan pozicionált szenzort tartalmaz, és egy fix tengely körül forog kis sebesség fluktuációval. A rendszer




**1. ábra:** A megvalósított elemek áttekintő ábrája

kimenete egy adott $t$ idő pillanatban az $\mathcal{L}^t = \{p_1^t, \ldots, p_{l^t}^t\}$ pontfelhő, $l^t = R \cdot c^t$ darab ponttal, ahol $c^t$ a $t$ alatt kapott pont oszlopok számát jelöli, melyek az adott mérés $R$ szenzor által adott értékeit tartalmazzák, vagyis $c^t$ függ a forgás sebességétől. Minden $p \in \mathcal{L}^t$ pontot a szenzortól mért távolság $d(p) \in [0, D_{\max}]$, a szenzor index $\hat{\vartheta}(p) \in \{1, \ldots, R\}$, és a forgás szöge $\varphi(p) \in [0, 360°]$ ír le, melyek közül $d(p)$ és $\hat{\vartheta}(p)$ paraméterek közvetlenül az eszközből érkező adatfolyamból nyerhetők ki. A forgás szögét a $p$ pontot a föld síkjára levetített Euklideszi koordinátákból számolhatjuk.

A bemutatott módszer célja, hogy minden $t$ időpillanatban valamennyi beérkező $p \in \mathcal{L}^t$ ponthoz rendeljünk egy $\omega(p) \in \{\text{fg}, \text{bg}\}$ címkét aszerint, hogy mozgó (előtérhez tartozik), vagy háttér osztályhoz tartozik-e.

A hatékony adatfeldolgozás érdekében származtatunk egy mélységképet a három dimenziós adathalmaz leképezéseként. A pontfelhőt egy henger felületére vetítjük, melynek középpontja az RMB-Lidar pozíciója, tengelye merőleges a föld síkjára. Megjegyezzük, hogy ez a konfiguráció abban az esetben is alkalmazható, amikor az eszközt döntött pozícióba helyezzük a föld síkjához képest, annak érdekében, hogy növeljük a látószöget. Ezután $S_H \times S_W$ méretű $S$ kétdimenziós pixelhálót feszítünk a hengerre, melynek $S_H$ magassága megegyezik a szenzorok $R$ számával, és $S_W$ szélessége a forgási szög felosztásaival. Jelölje a pixelrács egy adott pixelét $s$, $[y_s, x_s]$ koordinátákkal. Definiáljuk a $\mathcal{P} : \mathcal{L}^t \to S$ leképezést, hogy $y_s$ megegyezik a szenzor-indexszel, $x_s$ pedig a $[0, 360°]$ intervallum $S_W$ részre történő felosztásával:

$$s \stackrel{\text{def}}{=} \mathcal{P}(p) \text{ iff } y_s = \hat{\vartheta}(p),\ x_s = \text{round}\left(\varphi(p) \cdot \frac{S_W}{360^\circ}\right) \tag{1}$$

Jelöljük az egy $s$ pixelre vetülő pontokat $P_s \subset \mathcal{L}^t$-vel. Feltesszük, hogy az egy adott irányban lévő előtér pontok közelebb vannak a szenzorhoz, mint a kiszámított háttér pontok távolságának átlaga. így minden $s$ pixelhez kiválasztjuk a legközelebbi pontot $p_s^t = \arg\min_{p \in P_s} d(p)$, és a pixel értékhez rendeljük a ponthoz tartozó $d_s^t = d(p_s^t)$ mélység értéket. Azon pixelek, melyeknek nincs mélység információjuk a felhő levetítése után sem ($P_s = \emptyset$), interpoláljuk a $d_s^t$ távolságokat a szomszédos pixelek értékeiből. A térbeli szűréshez a nyolc legközelebbi szomszéd pixel értékét vesszük figyelembe, és $N_s \subset S$-el jelöljük az $s$ pixel szomszédait.

## 3. Háttér modell

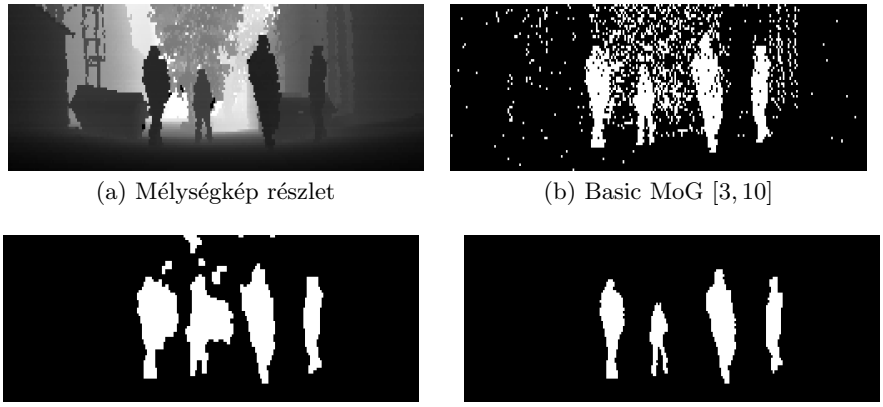
A háttérmodell létrehozásához minden $p \in \mathcal{L}^t$ ponthoz hozzárendelünk egy $f_{\text{bg}}(p)$ fitnesz függvényt, mely meghatározza, hogy $p$ milyen mértékben tartozik a háttérhez. A folyamat a hengerre történő leképezéssel kezdődik (1) szerint, ahol $S^{\text{bg}}$ pixelrácsot használunk. Hasonlóan [3]-hoz, minden $S^{\text{bg}}$-beli $s$ cellára meghatározunk egy többszörös Gaussok keveréke modellt (Mixture of Gaussians, MoG [8] [9]) a $d(p)$ jellemző mélység hisztogramja alapján, mely az $s$-re vetülő $p$ pontok $d(p)$ értékeiből áll. [10] alapján, mi is fix $K$ számú komponenst használunk ($K = 5$), $w_s^i$ súlyokkal, $\mu_s^i$ várható értékkel, és $\sigma_s^i$ szórással, $i = 1 \dots K$. A következő lépésben csökkenő sorrendbe rendezzük a súlyokat, és megkeressük a legkisebb $k_s$ egész számot, mely kielégíti az $\sum_{i=1}^{k_s} w_s^i > T_{\text{bg}}$ egyenlőtlenséget ($T_{\text{bg}} = 0.89$). Ezután a $k_s$ legnagyobb súlyú komponenst tekintjük a háttér elemének. $\eta()$ Gauss sűrűségfüggvény esetén, $\mathcal{P}^{\text{bg}}$-vel jelölve a leképezést $S^{\text{bg}}$-re, az $f_{\text{bg}}(p)$ fitnesz függvény a következőképpen áll elő:

$$f_{\text{bg}}(p) = \sum_{i=1}^{k_s} w_s^i \cdot \eta\left(d(p), \mu_s^i, \sigma_s^i\right),\ \text{ahol } s = \mathcal{P}^{\text{bg}}(p). \tag{2}$$

A gaussi paraméterek beállítása, és frissítése [10] alapján történik, esetünkben az $S_W^{\text{bg}} = 2000$ forgás-szög felbontás adta a leghatékonyabb detekciós eredményeket. A fitnesz értékre egy egyszerű küszöbértéket (threshold) alkalmazva, sűrű előtér/ háttér klasszifikációt kapunk a pontfelhőre [3] [10] (*Basic MoG*), de amint az a 2. ábrán látható, az eredmény rendkívül zajos lehet.

## 4. DMRF modell előtér szegmentációra

Ebben a fejezetben bemutatunk egy dinamikus véletlen Markov mező (DMRF) modellt, előtér-szegmentációra a pontfelhő szekvencián. Mivel az MRF optimalizáció számítási kapacitás szempontjából erőforrás igényes [11], ezért a DMRF



(a) Mélységkép részlet (b) Basic MoG [3, 10]

(c) uniMRF [12] (d) A javasolt DMRF szegmentáció

**2. ábra:** Előtér szegmentáció mélységkép részleten három különböző módszerrel

modellt a mélységképek pixelrácsán definiáljuk, majd a kapott címkéket visszavetítjük a pontfelhők 3D terébe, javítva közben a kvantálásból adódó hibákat. Ahogy a 2. fejezetben említettük az (1) által definiált hengerre vetítést alkalmazzuk a mélységkép létrehozásához, melyben a rács szélessége $S_W = \min(\hat{c}, S_W^{\mathrm{bg}}/2)$ ahol $\hat{c}$ az oszlopok számát jelöli.

A következőkben minden $s \in S$ pixelhez hozzárendelünk előtér és háttér energiákat, melyek leírják az osztályba tartozás mértékét a $d(s)$ mélység értékekből következtetve. A háttér energiát leíró függvény közvetlenül a paraméteres MoG valószínűségekből számolható (2):

$$\varepsilon_{\mathrm{bg}}^t(s) = -\log\left(f_{\mathrm{bg}}(p_s^t)\right).$$

Az előtér leírására a konstans $\varepsilon_{\mathrm{fg}}$ küszöbtényező kézenfekvő választásnak tűnik [12](uniMRF), de így a modell által adott eredményben több hibás előtérpixel keletkezik a háttér mozgásából, és a kvantálási hibákból adódóan. Ezen hibák kiküszöböléseként az időbeli statisztikai módszer helyett térbeli mélységkülönbségeket figyelünk a következő feltevéssel: amikor $s$ egy előtérpixel, akkor kell lennie hasonló mélységű előtérpixelnek $s$ környezetében is. Ezért egy kernel sűrűség függvényt használunk az előtér osztályhoz:

$$\varepsilon_{\mathrm{fg}}^t(s) = \sum_{r \in N_s} \zeta(\varepsilon_{\mathrm{bg}}^t(r), \tau_{\mathrm{fg}}, m_\star) \cdot k\left(\frac{d_s^t - d_r^t}{h}\right),$$

ahol $h$ a kernel sávszélessége, és $\zeta : \mathbb{R} \to [0, 1]$ szigmoid függvény:

$$\zeta(x, \tau, m) = \frac{1}{1 + \exp(-m \cdot (x - \tau))}.$$

Uniform kernelt használva: $k(x) = \mathbf{1}\{|x| \leq 1\}$, ahol $\mathbf{1}\{.\} \in \{0, 1\}$ a bináris tagság-indikátor függvény.

A mélységkép szegmentációt formálisan definiálva minden $s \in S$ pixelhez egy $\omega_s^t \in \{\text{fg}, \text{bg}\}$ címkét rendelünk, majd a következő energia függvényt szeretnénk minimalizálni:

$$E = \sum_{s \in S} V_D(d_s^t | \omega_s^t) + \sum_{s \in S} \underbrace{\sum_{r \in N_s} \alpha \cdot \mathbf{1}\{\omega_s^t \neq \omega_r^{t-1}\}}_{\xi_s^t} + \sum_{s \in S} \underbrace{\sum_{r \in N_s} \beta \cdot \mathbf{1}\{\omega_s^t \neq \omega_r^t\}}_{\chi_s^t}, \tag{3}$$

ahol $V_D(d_s^t|\omega_s^t)$ jelöli az adat tagot, míg $\xi_s^t$ és $\chi_s^t$ az időbeli, és a térbeli elemeket, $\alpha > 0$ és $\beta > 0$ konstansokkal. Esetünkben bár a modell dinamikus a különböző időpontokhoz tartozó szegmentált képek közötti kölcsönhatások miatt, a valós idejű működés érdekében egy kauzális rendszerrel dolgozunk, vagyis a múltban kiosztott címkéket nem változtatjuk később az új adatok függvényében.

Az adattagok az adatfüggő energiatényezőkből származnak szigmoid leképezéssel:

$$V_D(d_s^t|\omega_s^t = \text{bg}) = \zeta(\varepsilon_{\text{bg}}^t(s), \tau_{\text{bg}}, m_{\text{bg}})$$
$$V_D(d_s^t|\omega_s^t = \text{fg}) = \begin{cases} 1, & \text{if } \; d_s^t > \max_{\{i=1\ldots k_s\}} \mu_s^{i,t} + \epsilon \\ \zeta(\varepsilon_{\text{fg}}^t(s), \tau_{\text{fg}}, m_{\text{fg}}), & \text{egyébként.} \end{cases}$$

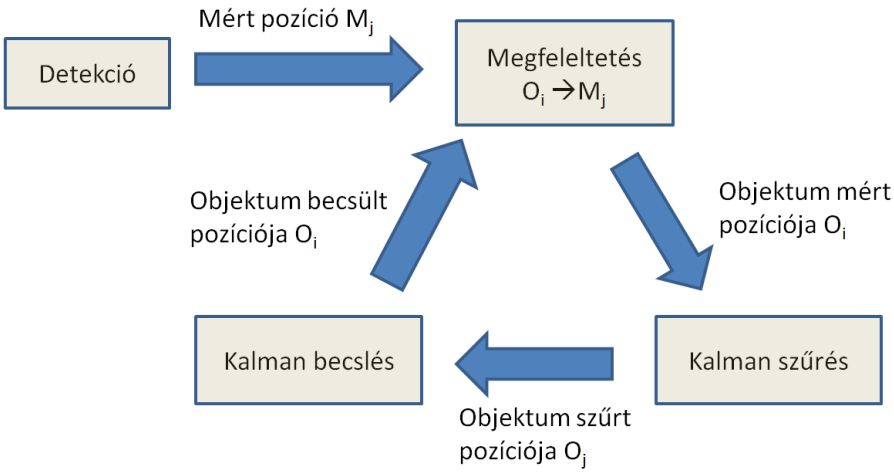
A szigmoid paraméterek $\tau_{\text{fg}}$, $\tau_{\text{bg}}$, $m_{\text{fg}}$, $m_{\text{bg}}$ és $m_\star$ meghatározhatók manuálisan annotált tanuló képek alapján, Maximum Likelihood módszerekkel. Esetünkben $\alpha = 0.2$ és $\beta = 1.0$ konstansokat használunk, a kernel sávszélessége pedig $h = 30$cm. Az MRF energia (3) minimalizálása a gyors gráf-vágáson alapuló algoritmussal [11] történik.

A DMRF optimalizáció eredményeként egy bináris előtér maszkot kapunk az $S$ pixelrácson. Az utolsó lépése a módszernek az eredeti $\mathcal{L}$ pontfelhő pontjainak klasszifikációja, figyelembe véve, hogy több különböző osztályba tartozó pont vetülhet ugyanarra a pixelre. Jelölje a $t$ időpillanatban az $s$ pixelre vetülő pontokat $\mathcal{P}(p)$. Az osztályozás a következőképpen történik:

- $\omega(p) = \text{fg}$, ha valamelyik feltétel a következő kettőből teljesül:
  (a) $\omega_s^t = \text{fg} \quad \text{és} \quad d(p) < d_s^t + 2 \cdot h$
  (b) $\omega_s^t = \text{bg} \quad \text{és} \quad \exists r \in N_r : \{\omega_r^t = \text{fg}, |d_r^t - d(p)| < h\}$
- $\omega(p) = \text{bg}$: egyébként.

A fenti feltételek több hamis pozitív (a) és hamis negatív (b) előtér pontot távolítanak el, ahol az objektumok széleinek környezetéből vetülnek a pontok egy-egy pixelre.

Az eljárás kimenete egy olyan pontfelhő, melyben valamennyi pontot előtér vagy háttér osztályba soroltunk be, ahogy az a 3. ábra képein is megfigyelhető.




**3. ábra:** A követő algoritmus folyamatábrája

## 5. Célpont-követés

Ebben a fejezetben a korábban ismertetett előtérszegmentációs modell alkalmazását mutatjuk be célpontkövetés feladatára. Az itt ismertetett módszer része az MTA SZTAKI EEE laboratóriumában valós időben bemutatható demonstrációs környezetnek, amely a Kutatók éjszakája 2012 rendezvénysorozatra készült el, az i4D belső támogatású projekt keretein belül.

A célpontkövetés első lépése a különböző objektumokhoz tartozó előtérpontok elkülönítése a pontfelhőben. Ezt az előtérpontok talajsíkra vetítésével és ezután a 2D síkon való összefüggő komponensek kinyerésével végeztük el. Egy-egy pillanatfelvételen előfordulhat, hogy a szorosan egymás mellett álló alakzatok összeolvadnak így, azonban az időbeli követéssel ezeket a hibákat később kiszűrjük. Az objektumok helyét ezután a talappontjuk koordinátáival reprezentáljuk. Következő cél, hogy minden aktuális pillanatfelvételre a $k$ detektált objektum jelöltet - vagyis a kétdimenziós talppontokat - lehetőség szerint hozzá kell rendelnünk az eddig nyilvántartott $n$ trajektóriához. A feldolgozás során a következő eseteket kell számításba vennünk:

- Ha az adott detektált pont folytatása egy már megkezdett trajektóriának, akkor meg kell találnunk a hozzá megfelelőt.
- A detektált pont lehet hibás detekció, ami nem tartozik objektum trajektóriához
- Az aktuális pontfelhőben detektált pont lehet egy új trajektóriának a kezdőpontja.
- Egy meglévő trajektória befejeződhetett az előző időpillanatban (például az objektum elhagyja a megfigyelt területet), ilyenkor nem tartoznak új mért pozíciók hozzá.

– A detektor kihagyhat pár objektumjelöltet, viszont ez nem okozhat több részre szakadt, vagy újraindított trajektóriákat, vagyis az időbeli folytonosság érdekében ki kell tölteni a hiányzó részeket becsült pozíciókkal.

A kidolgozott algoritmus folyamatábrája a 3. ábrán látható. Három lépést iterálunk minden pontfelhőre: (A) Megfeleltetés, (B) Kalman szűrő és (C) Kalman predikció.

### 5.1. Megfeleltetés

A megfeleltetés központi része az algoritmusnak. Elsőként normalizáljuk a mért pozíciókat, hogy a $[0, 1]$ tartományba kerüljenek minden dimenzióra. Jelölje $O_j$ $(j = 1, \ldots, k)$ a detektált, normalizált pozíciókat és $M_i$ $(i = 1, \ldots, n)$ az $i$. trajektóriához tartozó becsült pozíciót. Definiáljuk az $O_j$ és $M_i$ távolságot az Euklideszi távolság alapján, melyből a távolság mátrix $D$ számolható különböző $i$ és $j$ értékekre:

$$D_{ij} = d(O_i, M_j) \quad \text{for } i \leq n, j \leq k$$

Itt a $D_{ij}$ mátrix elem mutatja, hogy mennyire jól illik a $j$. mérés az $i$. trajektóriára.

A trajektóriákat és az aktuális méréseket a magyar módszer segítségével kapcsoltuk össze, $D$ távolság mátrix alapján [13].

Meg kell jegyeznünk, hogy a magyar módszer algoritmus mindig négyzetes mátrixokat fogad bemenetként, vagyis ha ugyanannyi trajektória van mint mérés. Ez a feltétel gyakran nem teljesül, ezért ha $n > k$, akkor generálunk $n - k$ darab mérést, melyek minden trajektóriától a lehető legtávolabb helyezkednek el:

$$d(O_i, M_j) = 2 \quad \text{for } i \leq n, n \geq j > k$$

Itt 2-t választottuk az *default* értéknek, mert ez nagyobb minden távolságnál a normalizált kockán belül.

A másik esetben, mikor $k > n$, generálunk $k - n$ *jelképes* trajektóriát a $D$ mátrix kiegészítéseként:

$$d(O_i, M_j) = 2 \quad \text{for } k \geq i > n, j \leq k$$

A magyar módszer kimenete egyértelmű megfeleltetés $j \to A(j)$ a mérések, és a trajektóriák között, ahol $j/A(j)$ index tartozhat valós, vagy *jelképes* méréshez/ trajektóriához is.

Legyen $t_{\text{dist}}$ a távolság küszöb. A $j \to A(j)$ leképezés a következő módon működik:

– ha $A(j) \leq n$, $j \leq k$ és $d(O_{A(j)}, M_j) < t_{\text{dist}}$: **DO** $j$. mérés az $A(j)$. trajektóriához tartozik (`matched`)
– egyébként
  • ha $A(j) \leq n$, $j \leq k$, de $d(O_{A(j)}, M_j) \geq t_{\text{dist}}$: mind a $j$. mérés és az $A(j)$. trajektória nem tartozik sehova (`unmatched`).

- ha $A(j) \leq n$ és $n \geq j > k$ az $A(j)$. trajektória nem tartozik sehova (`unmatched`).
- ha $k \geq A(j) > n$ és $j \leq k$ a $j$. mérés nem tartozik sehova (`unmatched`).

Ha az $M_j$ mérés az $O_i$ trajektóriához tartozó (`matched`), akkor úgy vesszük, hogy az $M_j$ trajektória következő pozíciója az $i$. jelölt, vagyis ezt használjuk a trajektória frissítésére.

Sehova sem tartozó mérések (`unmatched`) új trajektóriák kezdőpontjai, vagy a detektor hibás pozitív találatai lehetnek. Az aktuális pontfelhőben nem tudjuk megkülönböztetni a kettőt, ezért minden `unmatched` méréshez indítunk egy új trajektóriát, melyet a következő iterációk során értékelünk ki. Azt várjuk, hogy a folyamat végére a hamis jelöltek rövid trajektóriákat eredményeznek, melyeket utólagosan eltávolíthatunk. Ha egy trajektóriához nem találunk folytatást az aktuális időpillanatban, akkor két eset lehetséges: (i) véget ért, vagy (ii) a detektor nem találta meg a hozzá tartozó jelöltet. Emiatt az `unmatched` trajektóriákat nem zárjuk le azonnal, hanem egy INAKTÍV címkével látjuk el. Amennyiben a pálya inaktív marad egy küszöb érték $t_{\text{time}}$ után is, akkor TÖRÖLT-nek jelöljük, és kivonjuk a további vizsgálatok alól.

### 5.2. Kalman szűrés

Minden trajektória az egyes pontfelhőkben detektált pozíciók szekvenciájával írható le. Az adott felvételi sebességnél az objektumok mozgása folyamatosnak mondható, ezért becslést tehetünk a trajektória következő pontjára. Továbbá mivel a mérések zajosak lehetnek, ezért az objektum valós pozícióját mind a szenzor kimenete, mind a már ismert trajektória alapján kell meghatároznunk, a zaj minimális szintre történő csökkentése érdekében.

Ezért minden trajektóriához egy Kalman szűrőt rendelünk, melyet minden új pontfelhő feldolgozásakor, az aktuális mérésekkel frissítünk. Az INAKTÍV, de nem TÖRÖLT pályák esetén, a frissítés az aktuális pozícióra adott utolsó becsléssel történik. Mindkét esetben az új pont a szűrő korrigált állapota lesz. Mivel a Kalman szűrő csak egy bizonyos számú iteráció után érvényes, ezért bevezetjük a $t_{\text{Kalman}}$ küszöb értéket, és ha a trajektória rövidebb mint ez az érték, akkor a változtatott érték helyett közvetlenül a mért értéket használjuk következő pontként. Ebből az következik, hogy a pálya sokkal zajosabb lesz a kezdeti fázisban, de nagyban csökkentjük a valószínűségét, hogy teljesen elvesszenek trajektóriák.

Az INAKTÍV pályák további pontjai először egy ideiglenes tárolóba kerülnek. Ha később újra aktiválódik a trajektória, akkor hozzáadjuk a pontokat az eredeti pályához. Ha TÖRÖLT elemmé válik, akkor töröljük az ideiglenes tároló tartalmát.

### 5.3. Kalman predikció

Az utolsó lépése a trajektória javításnak az, hogy becsüljük a következő pozíciót minden pályához ($O_i$-vel jelölve), melyet a mérések megfeleltetésére használunk

a következő pontfelhőn. Itt megint a Kalman becslést csak egy adott $t_{\text{Kalman}}$ idő után kezdjük. A pálya kezdeti szakaszában a legutóbb ismert pozíciót használjuk, mint becslést a következő pontfelhőn.

### 5.4. Utófeldolgozás és szűrés

A fenti követő algoritmus kimenetében lehetnek hibás pályák, melyeket a mérési zaj miatt észlelünk. Az iteratív folyamat végén az ilyen trajektóriákat el kell távolítanunk. Megfigyeléseink szerint a törlendő pályák általában rövidek, vagy az általuk jelzett objektumok közel konstans pozícióban vannak (korábban feltettük, hogy az objektumok mozognak). Az utófeldolgozó lépésben két feltételt kötöttünk ki a trajektóriákra:

- A trajektória hosszának nagyobbnak kell lennie mint $t_{\text{length}}$
- A pontok koordinátáinak szórása legyen nagyobb mint $t_{\text{variance}}$

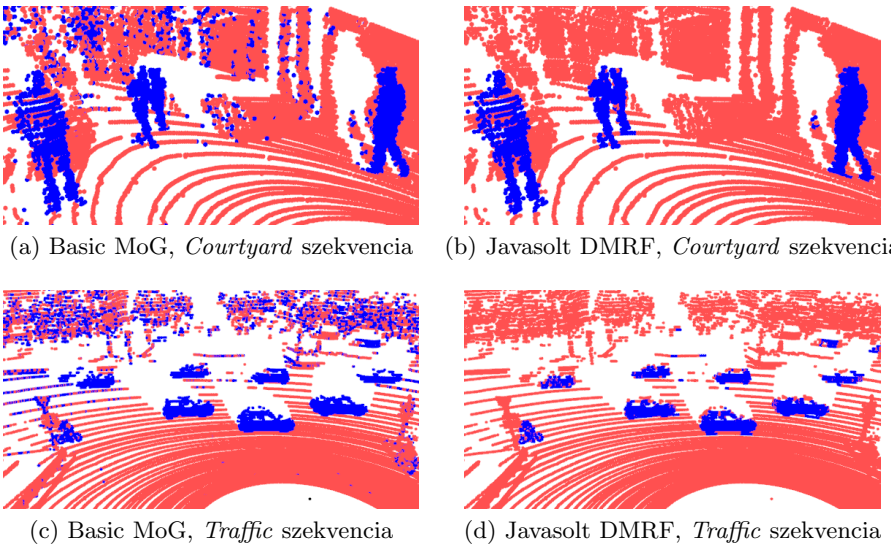
Azokat a trajektóriákat, amelyek egyik fenti feltételt sem teljesítik, töröljük, és a maradék lesz a követés kimenete.

## 6. Kiértékelés

Szakirodalmi relevanciája miatt részletes kvantitatív kiértékelést az előtér-háttér szegmentációs modulra végeztünk. Az objektumkövetésről kvalitatív teszteket mutatunk be a 7. fejezetben.

Az előtérdetekciós módszert valós Lidar szekvenciákon teszteltük, mind videó felügyeleti (*Courtyard*), mind forgalom monitorozási (*Traffic*) esetekre (4. ábra). Az adatokat a Velodyne HDL 64E S2 kamerával vettük fel, amely $R = 64$ vertikálisan pozicionált szenzort tartalmaz. A *Courtyard* szekvencia 2500 pontfelhőt tartalmaz, melyen négy ember sétál egy $25m^2$-es területen, 1-5$m$ távolságra a Lidartól, kereszteződő pályákkal. Az eszköz forgási sebessége 20Hz volt. A háttérben mozgó növényzet nagy kihívást jelent a pontos klasszifikációban. A *Traffic* szekvencia 5Hz-es forgási sebességgel lett felvéve egy autó tetejéről, amely egy nagy-forgalmú úton a piros lámpánál vár. Az adaptív háttér modell inicializálása után 160 pontfelhő volt alkalmas a forgalom analízishez. A DMRF modellt három referencia megoldással hasonlítottuk össze:

1. *Basic MoG*, 3. fejezetben mutattuk be, mely a [3] alapján működik, on-line paraméter frissítéssel [10].
2. *uniMRF*, 4. fejezetben mutattuk be, mely részben alkalmazza [12] uniform előtér modelljét.
3. *3D-MRF*, mely a három-dimenziós MRF modellt mutatja be, hasonlóan [7]-hez. Itt a pont szomszédságokat az eredeti $\mathcal{L}^t$ pontfelhőben definiáljuk Euklideszi távolság alapján, és a (2) háttér fitnesz értéket használjuk az adat modellhez. A gráf-vágás alapú algoritmust [11] MRF energia optimalizációként alkalmaztuk.



(a) Basic MoG, *Courtyard* szekvencia (b) Javasolt DMRF, *Courtyard* szekvencia

(c) Basic MoG, *Traffic* szekvencia (d) Javasolt DMRF, *Traffic* szekvencia

**4. ábra:** Pontfelhő klasszifikációs eredmények a *Basic MoG* és a bemutatott *DMRF* modellel: előtér pontok kékkel jelölve.

Kvalitatív eredmények két minta pontfelhőn a 4. ábrán láthatók. Ground Truth (GT) pontfelhő generáláshoz létrehoztunk egy 3D pontfelhő annotációs eszközt, mellyel manuálisan tudunk felcímkézni régiókat a pontfelhőn. Ezután annotáltunk 700 pontfelhőt a *Courtyard* szekvenciából, és 50 pontfelhőt a *Traffic* szekvenciából. A kvantitatív kiértékeléshez a pont szintű F-mértéket használtuk [14], melyet a precizitás és recall értékek harmonikus átlagából számíthatunk. Ezen kívül mértük a feldolgozás sebességét is pontfelhő per másodperc egységekben (fps). A numerikus teljesítmény analízis eredményei a 1. táblázatban láthatók. Az táblázatból látható, hogy a bemutatott módszer felülmúlja a *Basic MoG* és *uniMRF* módszereket az F-mérték szempontjából mindkét szekvencián, a különbségek leginkább a *Courtyard*-on érzékelhetők. A *3D-MRF* módszerrel összehasonlítva, a mi modellünk hasonló eredményeket adott precizitásban, de a *DMRF* módszer sokkal gyorsabb. Megfigyelhető, hogy eltérően a *3D-MRF* módszertől, a mi mélységkép alapú technikánk kevésbé függ a pontfelhő méretétől. A *Traffic* szekvencián, mely közel 260000 pontot tartalmaz pontfelhőnként, mi 16fps sebességet értünk el, míg a *3D-MRF* módszer csak 2fps gyorsaságra volt képes.

## 7. Alkalmazás

Az ismertetett módszernek több felhasználási módja lehet, ahogy az 1. fejezetben említettük többek között a biometrikai azonosítás, vagy az esemény-elemzés. Másik fontos alkalmazáscsoport a helyszín virtuális rekonstrukciója interaktív virtuális valóság rendszerek, animációs és fimipari alkalmazások számára. Rend-

| | Szekvencia | Elemek száma | Basic MoG | uniMRF | 3D-MRF | **DMRF** |
|---|---|---|---|---|---|---|
| F-mérték (%-ban) | *Courtyard* | 4 obj/fr. | 55.7 | 81.0 | 88.1 | 95.1 |
| | *Traffic* | 20 obj/fr. | 70.4 | 68.3 | 76.2 | 74.0 |
| Sebesség (fps) | *Courtyard* | 65K pts/fr. | 120 fps | 18 fps | 7 fps | 16 fps |
| | *Traffic* | 260K pts/fr. | 120 fps | 18 fps | 2 fps | 16 fps |

**1. táblázat:** Numerikus kiértékelés a *Courtyard* és a *Traffic* szekvenciákon: detekciós pontosság (F-mérték %-ban) és feldolgozási sebesség (fps, asztali számítógépen mérve)

szerünkben ezt a terület rekonstrukciójával értük el, vagyis a pontfelhőből poligonhálót alkottunk, úgy hogy az egyes detektált objektumok a poligonhálóban is különálló egységek maradjanak. A rekonstrukciót a *Courtyard* szekvencián végeztük el.

### 7.1. Környezet rekonstrukció

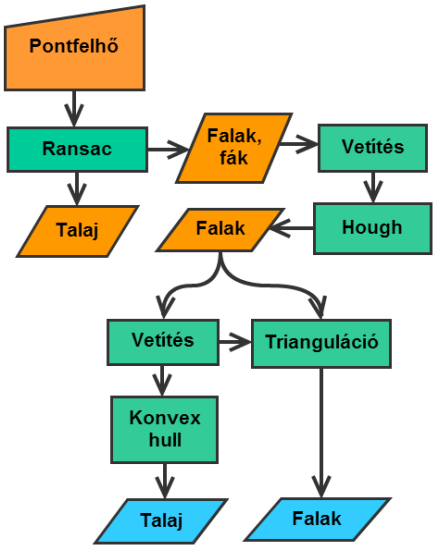
Elsőként a statikus háttérkörnyezet rekonstrukciójával foglalkozunk. A *Courtyard* szekvencián az előtér pontokat (azaz többnyire a járőkelőket) kivéve a pontfelhőből, többféle helyszínelem marad: talaj, fal, fák, egyéb háttérobjektumok. A talaj pontjait a RANSAC [15] algoritmussal síkot illesztve automatikusan detektáltuk. A talaj pontjai alapján kiszámítható a talaj átlagos magassága is, melyet a következő lépésben használunk fel. A sík illesztés után megmaradt pontokat függőlegesen levetítjük, majd a kapott két-dimenziós képen Hough transzformációval [16] megkeressük az egyeneseket, melyek nagy valószínűséggel a pontfelhőben falakat reprezentálnak. A pontokból poligonokat triangulációval készítünk a Ball-pivoting algoritmust felhasználva. A növényzet és a kisebb háttérobjektumok automatikus rekonstrukciója, valamint a textúrázás automatizált véghezvitele későbbi munkáink célja, egyelőre ezeket a lépéseket manuálisan végeztük el. A környezetrekonstrukciós algoritmus folyamatábrája a 5. ábrán látható, a folyamat lépéseire példa képkockák a 6. ábrán találhatók.

### 7.2. Járókelők "rekonstrukciója"

A mozgó járókelők élethű rekonstrukciója nem megoldható a pontfelhő alapján, mivel az túl ritka és csupán 2.5D információt nyújt. Modellünkben ezért előre felvett sétáló modelleket helyeztünk a virtuális környezetbe, melyek a valós mérési eredmények alapján kinyert járókelő trajektóriákat követik. Az élethű textúrázott mozgó modellek az MTA SZTAKI GMSZL Laboratóriumának 4D stúdiójában készültek[18] [19]. A rekonstrukciós eredmény egyik minta képkockája az 7. ábrán látható.

## 8. Konklúzió

Dinamikus MRF modellt mutattunk be előtér szegmentációra pontfelhőkön, melyeket egy forgó Lidar eszközből nyertünk ki. Hatékony térbeli előtér szűrőt




**5. ábra:** A környezet rekonstrukciójához szükséges pontfelhő szegmentáció algoritmusa.

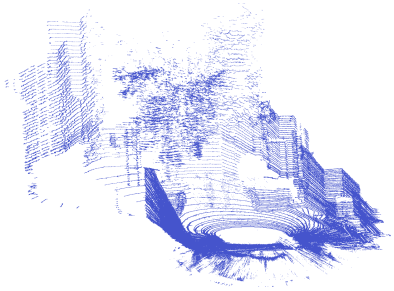
ismertettünk, mely csökkenti a forgási szög kvantálásából és a háttér mozgásából adódó hibákat. Követő algoritmust használtunk az objektumok elkülönítéséhez, és a trajektóriák meghatározásához. A modellt kvantitatíve kiértékeltük a Ground Truth alapján, és kiemeltük az előnyeit három másik módszerrel összehasonlítva. Végül az elkészült algoritmus alkalmazását elősegítő rekonstrukciót mutattuk be.

## 9. Köszönetnyilvánítás

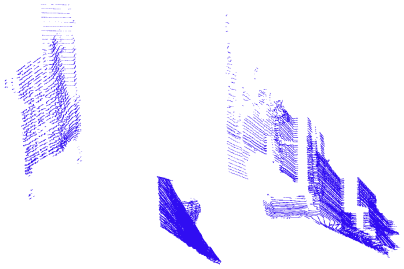
A cikkben közölt eredmények az MTA SZTAKI integrált 4D (i4D) belső támogatású projektje keretein belül készültek el. A szerzők köszöntet mondanak az Országos Tudományos Kutatási Alapprogramok (OTKA #101598) támogatásáért, a harmadik szerző munkáját a Magyar Tudományos Akadémia Bolyai János ösztöndíja is támogatta. Köszönet illeti az MTA SZTAKI GMSZL laborja dolgozóit együttműködésükért, különösen Jankó Zsoltot a 4D modellek elkészítéséért és a virtuális helyszínek összeállításáért.

## Irodalom

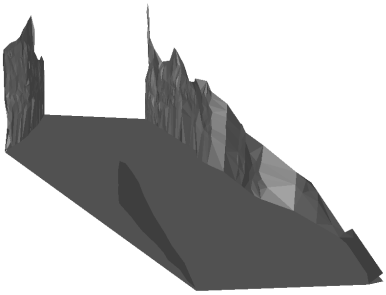
1. Kuba, A., Volcic, A.: Characterisation of measurable plane sets which are reconstructable from their two projections. Inverse Problems **4** (1988) 513–527
2. Benedek, Cs., Molnár, D., Szirányi T.: A Dynamic MRF Model for Foreground Detection on Range Data Sequences of Rotating Multi-Beam Lidar. International Workshop on Depth Image Analysis, Tsukuba City, Japan, November 2012, to appear in Lecture Notes in Computers Science, Springer, 2012



(a) Szegmentált háttér pontok.



(b) Falnak klasszifikált pontok.



(c) A fal és a talaj poligonhálója.

(d) Rekonstruált környezet.

**6. ábra:** A pontfelhő szegmentációs lépésekről minta képkockák.


3. Kaestner, R., Engelhard, N., Triebel, R., Siegwart, R.: A Bayesian approach to learning 3D representations of dynamic environments. Proc. International Symposium on Experimental Robotics (ISER), Berlin, 2010. Springer Press.
4. Kalyan, B., Lee, K.W., Wijesoma, W.S., Moratuwage, D., Patrikalakis, N.M.: A random finite set based detection and tracking using 3D LIDAR in dynamic environments. IEEE International Conference on Systems, Man, and Cybernetics (SMC), pages 2288–2292, Istanbul, Turkey, 2010. IEEE.
5. Muhammad, N., Lacroix, S.: Calibration of a rotating multi-beam Lidar. International Conference on Intelligent Robots and Systems (IROS), pages 5648–5653, Taipei, Taiwan, 2010. IEEE.
6. Spinello, L., Luber, M., Arras, K.O.: Tracking people in 3D using a bottom-up top-down detector. IEEE International Conference on Robotics and Automation (ICRA), pages 1304–1310, Shanghai, China, 2011.
7. Lafarge F., Mallet, C.: Creating large-scale city models from 3D-point clouds: A robust approach with hybrid representation. Int. J. of Computer Vision, 2012.
8. Schiller, I., Koch, R.: Improved video segmentation by adaptive combination of depth keying and Mixture-of-Gaussians. Proc. Scandinavian Conference on Image Analysis, Ystad, Sweden, volume 6688 of *LNCS*, pages 59–68, 2011.
9. Langmann, B., Ghobadi, S.E., Hartmann, K., Loffeld, O.: Multi-modal background subtraction using gaussian mixture models. ISPRS Symposium on Photogrammetric Computer Vision and Image Analysis, pages 61–66, 2010.

**7. ábra:** Objektum követés és rekonstrukciós eredmények. Bal felül: nyers pontfelhő; bal alul: szegmentált és elkülönített objektumok; jobb felül: trajektóriák felülnézetből; jobb alul: rekonstruált környezet, a stúdióobjektumok elhelyezése a valós alakzat pozíciókba.


10. Stauffer, C., Grimson, W.E.L.: Learning patterns of activity using real-time tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22:747–757, 2000.
11. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(9):1124–1137, 2004.
12. Wang, Y., Loe, K.F., Wu, J.K.: A dynamic conditional random field model for foreground and shadow segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(2):279 –289, 2006.
13. Kuhn, H.W.: The Hungarian Method for the assignment problem. Naval Research Logistics Quarterly, 2:83–97, 1955.
14. Benedek, C., Szirányi, T.: Bayesian foreground and shadow detection in uncertain frame rate surveillance videos. IEEE Transactions on Image Processing, 17(4):608 – 621, 2008.
15. Fischler, M.A., Bolles, R.C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM 24 (6): 381–395. (June 1981)
16. Duda, R.O., Hart, P.E.: Use of the Hough Transformation to Detect Lines and Curves in Pictures. Comm. ACM, Vol. 15, pp. 11–15 (January, 1972)
17. Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G.: The ball-pivoting algorithm for surface reconstruction. IEEE Transactions on Visualization and Computer Graphics, 5(4), Oct-Dec, 1999, pp. 349-359

18. Hapák, J., Jankó, Z., Chetverikov, D.: Real-Time 4D Reconstruction of Human Motion. Proc. 7th International Conference on Articulated Motion and Deformable Objects (AMDO 2012), Mallorca, Spain, Lecture Notes in Computer Science, Springer, vol. 7378, pp. 250-259, 2012.
19. Jankó, Zs., Chetverikov, D., Hapák, J.: 4D Reconstruction Studio: Creating dynamic 3D models of moving actors. Hungarian Computer Graphics and Geometry Conference, 2012