

Automatikus, tábla alapú LiDAR - kamera kalibráció

Tokarjev Andrej¹, Keszler Anita¹

HUN-REN Számítástechnikai és Automatizálási Kutatóintézet¹
{tokarjev,keszler.anita}@sztaki.hun-ren.hu

Kivonat A kamera és LiDAR fúzió egyre nagyobb szerepet kap a percepciós rendszerekben különböző alkalmazási területeken, mint például az autonóm járművezetés vagy az intelligens megfigyelés. Az ilyen rendszerek hatékony működése érdekében elengedhetetlen a szenzorok pontos kalibrációja. A cikk kamera és LiDAR kalibrációjára mutat be egy sakktábla detektálására épülő módszert. Az algoritmus újdonsága egyrészt a sakktábla pontfelhőn történő detektálásában, másrészt a kalibráció során használt hibafüggvény meghatározásában rejlik. Ez egy gyakorlati felhasználásra kiélezett teljesen automatizált kalibrációs eljárás, amely nem igényel kézi beavatkozást a szoftver futása közben. Az algoritmus beltéri és kültéri körülmények között is tesztelésre került, valamint több különböző LiDAR és kamera párosításánál is működőképesnek bizonyult.

1. Bevezetés

A manapság használt megfigyelőrendszerek és autonóm járművek tervezésekor már nem csupán egyetlen típusú érzékelőt használnak, hanem többféle szenzort integrálnak, hogy átfogóbb képet alkossanak a környezetükről. Ez a cikk két, gyakran alkalmazott szenzor, a kamera és a LiDAR összehangolásával, azaz kalibrációjával foglalkozik egy sakktábla detektálásán alapuló módszerrel.

A cikk az alábbi módon épül fel. A 2. fejezet a kapcsolódó irodalmat dolgozza fel. A kalibrációs eljárást a 3. fejezetben mutatjuk be: 3.1. egy rövid összefoglalót tartalmaz, a további alfejezetek pedig részletesen ismertetik az egyes lépéseket. A beltéri és kültéri tesztekhez kapcsolódó eszközöket és eredményeket a 4. fejezet tartalmazza. A cikket az összefoglalás zárja az 5. fejezetben.

2. Irodalmi áttekintés

A kalibráció biztosítja, hogy a LiDAR által generált távolsági adatok és a kamera információi térben pontosan illeszkedjenek egymáshoz. Ez elengedhetetlen például a tárgyak felismeréséhez és azok 3D helyzetének meghatározásához. Bár a kamera kalibrációját sok esetben megoldott problémának tekintik, a LiDAR-kamera kalibráció aktívabban kutatott terület, többek között az újonnan megjelenő LiDAR technológiák miatt. A kalibrációs algoritmusok csoportosításának

egyik fő szempontja, hogy a folyamat során szükség van-e egy vagy több kalibrációs objektumra (*target-based*) vagy speciális objektumok nélkül is működik a kalibráció (*targetless*).

Az utóbbi kategóriába tartozó módszerek nemrégiben jelentek meg a képi jellemzők kinyerésének fejlődésével. Ezek a megoldások a képen detektálható környezeti elemeket (például éleket és felületeket) használják fel a külső kalibrációs paraméterek meghatározásához szükséges geometriai korlátok létrehozására. Az objektum nélküli kalibrációk további két alcsoportra oszthatók aszerint, hogy statikus vagy dinamikus megoldást alkalmaznak-e.

A statikus módszerek hasonlítanak a kalibrációs objektumokat használó megoldásokhoz, csak az előre elkészített célobjektum(ok) helyett a környezet struktúráit és textúra elemeit használják fel. A [1] irodalomban például éleket detektálnak a szerzők a képen és a pontfelhőn is, és a visszavetítési hibát próbálják minimalizálni. Egy másik ilyen megoldást ismertet a [2], ahol egyenesek megfeleltetése a cél. Ezeknek a módszereknek többek között az a hátránya, hogy követelményeket támasztanak a környezettel szemben. [2] esetén a szerzők kitérnek rá, hogy szükség van megfelelő számú párhuzamos egyenesre a kalibráció sikerességéhez. Hasonló a helyzet a [3] cikkben leírt módszerrel, ami egyenesek detektálásán alapul (utak széle és lámpaoszlopok). Az ilyen megkötések akadályozzák ezen algoritmusok általános felhasználását.

A mozgás alapú eljárások előnye, hogy ezekre a feltételekre nincs szükségük. [4] egy olyan módszert javasol, ahol a kalibrációhoz szükséges paramétereket egy SLAM-ből származó trajektória hiba minimalizálásával határozzák meg a kamera és LiDAR által szolgáltatott adatokból. [5] a kezdeti 2D-3D megfeleltetést SuperGlue [6] használatával végzi. A [7] egy odometria alapú algoritmust mutat be. Bár számos előnyük van, több tanulmány is megkérdőjelezi a robusztusságukat, például [8], [9] [10]. A képi jellemzők kinyerésének teljesítményét ugyanis nagyban befolyásolhatják a kamera torzítási együtthatói és a képi jellemzők detekciója is, amely torzítatlan képek esetén is hibákat okozhat. A pontos időszinkronizáció is elengedhetetlen a jó eredmény érdekében, ami több kamera esetén problémát jelenthet [5].

Ezzel szemben a célobjektumot használó, azaz a *target-based* kalibrációs eljárások általában stabilabban működnek. A "közönséges" sakktábla az egyik leggyakrabban használt kalibrációs cél objektum [10,11,12,13,14,15], de sok egyéb megoldás is létezik, például: 3D sakktáblát [8,16], piramis alakzatban elrendezett sakktáblákat [17], kör alakú lyukakkal kiegészített sakktáblákat [18], gömböt [19] vagy éppen közönséges dobozokat [20] alkalmazó módszerek.

Az elterjedtsége és a könnyű előállíthatósága miatt az általunk javasolt algoritmus hagyományos sakktáblán alapszik. A sakktáblát használó módszerek kulcsfontosságú lépése rendszerint a tábla sarkainak (vagy más speciális pontjának) beazonosítása a pontfelhőn. Erre mutat példát a [10] és a [13] cikk, valamint [11] tartalmaz egy összefoglalót is. [11] emellett egy speciális differenciálható sakktábla mintát is bemutat, míg [12] intenzitás értékeket használ a tábla detektálásához és a 3D sarkok helyének beazonosításához. A sarkok meghatározása egy hagyományos sakktáblán a nem repetitív szkennelési mintázatot használó

ló LiDAR-ok esetén (*Nonrepetitive scanning* vagy *NRS-LiDAR*) az egyenetlen pontfelhő miatt nehézséget jelent, ezért az általunk javasolt megoldás nem épít erre.

[9] arra hívja fel a figyelmet, hogy a különböző LiDAR technológiák továbbra is kihívást jelentenek, mivel számos eljárás kihasználja az elterjedtebb forgó LiDAR-ok pontfelhőjének egyenletes mintázatát. A [10]-ben ismertetett módszer például a tábla pontfelhőből történő kiválasztásánál felhasználja, hogy a tábláról visszaverődő, egy gyűrűből származó pontok egy egyenes szakaszt alkotnak.

Mivel a tábla detektálása a pontfelhőn számításigényes lehet, sok algoritmus valamilyen kezdeti becslésre hagyatkozik, hogy csökkentse a futási időt. A [15]-ben a szerzők például a kamera kalibrálása során állítják elő azt a kezdeti becslést a tábla helyére, amiből azután a keresés a pontfelhőn kiindul. Az általunk javasolt algoritmus ezzel szemben egy háttérleválasztási lépéssel szűkíti a keresési teret.

Az általunk javasolt módszer automatizált, nincs szükség kézi beavatkozásra, emellett tesztelésre került több különböző technológiát alkalmazó LiDAR-on is.

3. A kalibrációs eljárás ismertetése

Ebben a fejezetben bemutatásra kerül a javasolt kalibrációs eljárás. A rövid elméleti rész és összefoglalás után az egyes lépések részleteit is ismertetjük.

A pontok leképezése. Legyen $[x_L \ y_L \ z_L]^\top$ egy pont a LiDAR koordináta-rendszerében. Ezt a T kalibrációs mátrixal vihetjük át a kamera koordináta-rendszerébe:

$$\begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x_L \\ y_L \\ z_L \end{bmatrix} = \mathbf{T} \begin{bmatrix} x_L \\ y_L \\ z_L \end{bmatrix}, \quad (1)$$

A kamera koordináta-rendszerből $[x_C \ y_C \ z_C]^\top$ -t a K kamera mátrix segítségével képezhetjük le a képre:

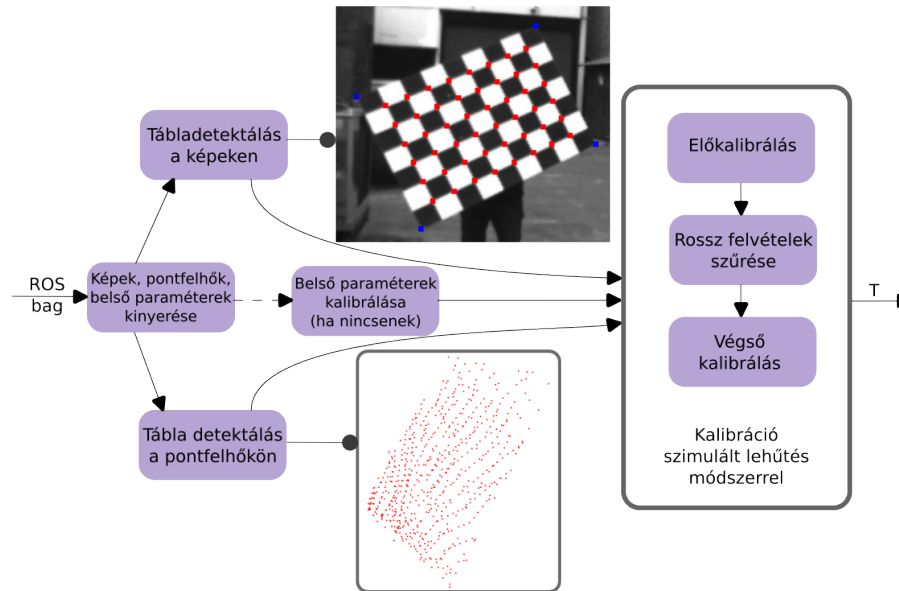
$$z_C \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = \mathbf{K} \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix}, \quad (2)$$

A teljes transzformáció:

$$z_C \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}\mathbf{T} \begin{bmatrix} x_L \\ y_L \\ z_L \end{bmatrix}, \quad (3)$$

A fenti egyenleten alapuló $\bar{p}_L = [x_L \ y_L \ z_L]^\top$ -t $[u \ v]^\top$ -be átvivő transzformációt röviden $\text{proj}(\mathbf{K}, \mathbf{T}, \bar{p}_L)$ -vel jelöljük:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \text{proj}(\mathbf{K}, \mathbf{T}, \bar{p}_L) \quad (4)$$



1. ábra. A teljes eljárás folyamatábrája. A belső kamera paraméterek (\mathbf{K}) kalibrálására csak akkor van szükség, ha azok nem állnak rendelkezésre. A kalibráció során az elő- és végső kalibrációs fázisok egy szimulált lefutás alapú módszert használnak. Az algoritmus kimenete a \mathbf{T} kalibrációs mátrix.

3.1. Rövid áttekintés

A mérések során a kamerát és a LiDAR-t egy közös állványra szereltük, emiatt az egymáshoz viszonyított helyzetüket állandónak tekinthetjük. Az algoritmust az 1. ábra szemlélteti, főbb lépései a következők:

1. Az adatok rögzítése (3.2. fejezet)
2. Képek, pontfelhők, belső paraméterek kinyerése (3.3. fejezet)
3. Belső paraméterek kalibrálása, amennyiben nem álltak rendelkezésre (3.3. fejezet)
4. A kalibrációs tábla detektálása a kameraképen (3.4. fejezet).
5. A kalibrációs tábla detektálása a LiDAR pontfelhőn (3.5. fejezet)
6. Kalibráció (előkalibrálás, rossz felvételek szűrése, végső kalibrálás) (3.6. fejezet)

3.2. Képek, pontfelhők és belső paraméterek kinyerése

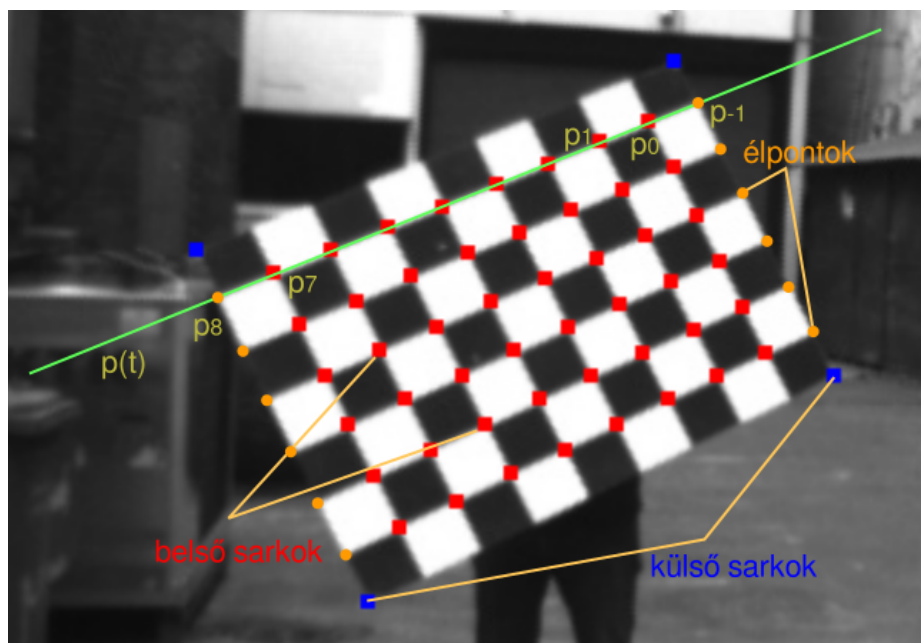
A szenzorok adatait egy ROS bag fájlban rögzítettük. Egy előfeldolgozási lépés során végigiterálunk a bag-ben lévő kép és pontfelhő adatokon, majd időbéllyeg alapján az egymáshoz legközelebbieket összepárosítjuk. A továbbiakban egy összetartozó pontfelhőt és kameraképet *frame*-nek nevezünk. Amennyiben rendelkezésre állnak a kamera belső paraméterei, azokat is elmentjük.

3.3. Belső kamera paraméterek (K) kalibrálása

Ha a kamera belső paraméterek (belső kamera mátrix, torzítási paraméterek) nem állnak rendelkezésre, előbb azok kerülnek kalibrálásra az OpenCV megfelelő függvényeinek használatával. A `cv::findChessboardCornersSB()` függvénnyel megkeressük a tábla belső sarokpontjait a képeken. A sarkokból, illetve azok ismert koordinátáiból a tábla koordináta-rendszerében a `cv::calibrateCamera()` függvény előállítja a kamera belső paramétereit.

3.4. A kalibrációs tábla detektálása a kameraképen

Először a belső kamera paraméterek segítségével elvégezzük a torzításmentesítést a képeken. Ezután az előző szakaszban leírt módon a sakktábla detektálásához a kameraképen az OpenCV `findChessboardCornersSB()` függvényét alkalmazzuk, aminek segítségével meghatározhatóak a tábla belső sarokpontjai. Ezt felhasználva kiszámolhatjuk a külső sarkok pozícióját is. A módszert a 2. ábra szemlélteti és a következő lépésekből áll. Az egy egyenesre eső belső sarokpontokból (piros) álló sorozatot meghosszabbítva kiszámoljuk a tábla élére eső pontokat (narancssárga). Az élre eső pontok sorozatának meghosszabbításával pedig megkapjuk a külső sarokpontokat (kék).



2. ábra. Sakktábla sarokpontok meghatározása a kameraképen. A belső sarokpontokból (piros) előbb kiszámoljuk az élre eső pontokat (narancssárga), majd ezekből a külső sarokpontokat (kék).

A pontsorozat meghosszabbítása. Vegyünk egy sort a sakktábla belső sarkából. Az ezekre illesztett 3 dimenziós egyenest a kamera koordináta-rendszerében írjuk le az alábbi egyenlettel:

$$\bar{r}(t) = \bar{a} \cdot t + \bar{b} \quad (5)$$

olyan \bar{a} és \bar{b} választással, hogy az egyenes t egész értékein vegye fel a sarokpontokat. Alkalmazva az (2) vetítést $\bar{r}(t)$ -re:

$$\bar{r}(t)_z \begin{bmatrix} \bar{p}(t)_x \\ \bar{p}(t)_y \\ 1 \end{bmatrix} = \mathbf{K} \cdot \bar{r}(t) \quad (6)$$

ahol $\bar{p}(t)$ az $\bar{r}(t)$ vetülete. Ezt kifejtve az alábbi egyenletet kapjuk:

$$\bar{p}(t) = \begin{bmatrix} (c_x \cdot t + d_x)/(a_z \cdot t + b_z) \\ (c_y \cdot t + d_y)/(a_z \cdot t + b_z) \end{bmatrix} \quad (7)$$

A fenti egyenlet paraméterei ismeretlenek, azonban ismerjük $\bar{p}(0)$, $\bar{p}(1)$, $\bar{p}(2)$ -t. Ezek a belső sarkok a kép koordinátarendszerében, amik meghatároznak egy egyenletrendszert. A b_z -t önkényesen 1-nek választjuk, a többi paraméter pedig kifejezhető az egyenletrendszerből az alábbi módon.

$$d_x = \bar{p}(0)_x \cdot b_z, \quad (8)$$

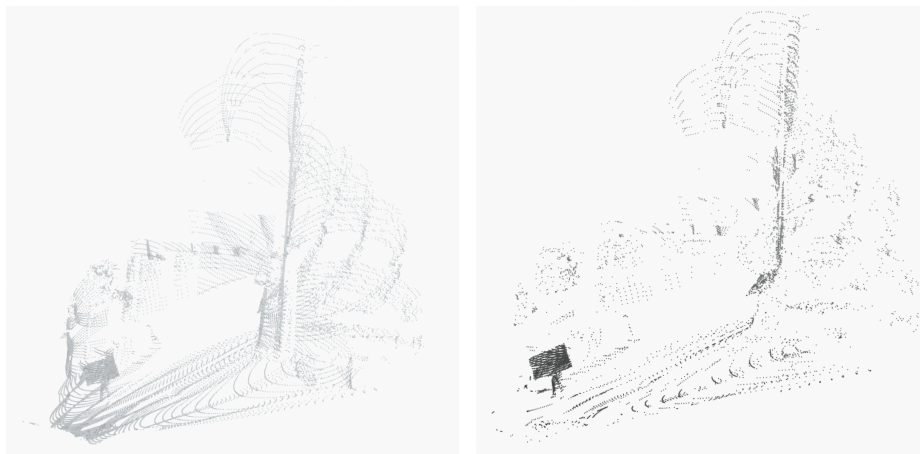
$$a_z = (d_x - (2 \cdot \bar{p}(1)_x - \bar{p}(2)_x) \cdot b_z)/(2 \cdot \bar{p}(1)_x - 2 \cdot \bar{p}(2)_x), \quad (9)$$

$$c_x = (\bar{p}(1)_x \cdot (a_z + b_z) - d_x) \quad (10)$$

Hasonló összefüggések írhatóak fel az y koordinátákra is. Ezzel tetszőleges t -re meghatároztuk $p(t)$ -t. A sor meghosszabbításánál levő pontok $p(-1 - \text{delta})$ -nak és $p(n + \text{delta})$ -nak felelnek meg, ahol n a pontok száma a sorban, delta pedig a szegély relatív mérete a sakktábla cellájához képest. (Ha nincs szegély, ahogy a 2. ábrán látható táblán sem, akkor $\text{delta} = 0$.)

3.5. A kalibrációs tábla detektálása a LiDAR pontfelhőn

A kalibrációs tábla keresése a teljes pontfelhőn előzetes információk nélkül egy számításigényes feladat lenne. Ezért az általunk javasolt algoritmus a háttér leválasztásával kezdődik. Az első frame-et háttérnek tekintjük és ezt leválasztjuk minden egyes további frame-ből. Erre mutat példát a 3. ábra. A leválasztás úgy történik, hogy minden ponthoz megkeressük a hozzá legközelebbi irányvektorú pontot a háttéren, és ha a két pont mélysége egy hibahatáron belül egyezik, akkor háttérnek tekintjük. A háttér leválasztása nem elengedhetetlen része az algoritmusnak, az esetek többségében ennek elhagyásával is működik a detektálás. A lépés célja a stabilitás és a hatékonyság növelése; egyrészt kiszűrünk lehetséges false positive eseteket (a táblához hasonló méretű/alakú tárgyakat,



(a) Teljes pontfelhő, 23354 pont.

(b) Előtér pontfelhő, 6243 pont.

3. ábra. Livox AVIA LiDAR-ral készült kültéri felvétel, 0.1 sec integrációs idővel a háttér leválasztása előtt és után.

1. Algorithm Tábla kereső algoritmus

Output: points

```

function FINDBOARD
  bestPoints  $\leftarrow$  empty set
  bestScore  $\leftarrow$  infinity
  for iter  $\leftarrow$  0 to numIter do
    points  $\leftarrow$  FINDCLUSTER()
    score  $\leftarrow$  score based on Equation (15)
    if score < bestScore then
      bestScore  $\leftarrow$  score
      bestPoints  $\leftarrow$  points
    end if
  end for
  return bestPoints
end function

```

pl. ajtó, TV), másrészt jelentősen csökkentjük keresési teret, növelve a keresés hatékonyságát.

Ezután a megmaradt pontokon futtatjuk a *Tábla kereső algoritmus-t* (Algoritmus 1). Az eljárás során többször lefutattunk egy véletlen faktort is tartalmazó klaszter kereső algoritmust (Algoritmus 2), ami minden iterációban visszaad egy ponthalmazt, amelyet aszerint értékelünk, hogy mennyire felel meg a táblának. Az értékelő függvényt a (15) egyenlet foglalja össze. A végén azt a ponthalmazt fogadjuk el a tábla reprezentánsának, ami a legjobb értékelést kapta.

2. Procedure A klaszter kereső algoritmus

Output: points

```

function FINDCLUSTER
  startCell  $\leftarrow$  Get a random cell from cells
  if size(startCell.points) < 3 then
    return None
  end if
  points  $\leftarrow$  cells.points
  for iter  $\leftarrow$  0 to ransacIterNum do
    plane  $\leftarrow$  FITPLANE(points)
    queue  $\leftarrow$  a queue data structure, initialized with startCell
    while queue is not empty do
      cell  $\leftarrow$  front(queue)
      if cell marked as visited then
        continue
      end if
      mark cell visited
      goodCell  $\leftarrow$  false
      for each point in cell.points do
        if point on plane then
          goodCell  $\leftarrow$  true
          add point to points
        end if
      end for
      if not goodCell then
        continue
      end if
      for each neighbour adjacent to cell do
        add neighbour to queue
      end for
    end while
  end for
  return points
end function

```

A klaszter kereső algoritmus (Algoritmus 2) ismertetése. Ennek az eljárásnak a célja, hogy találjon egy összefüggő, hibahatáron belül egy síkra eső ponthalmazt. Az egymáshoz közeli LiDAR pontok hatékony keresése érdekében felbontjuk a teret diszjunkt cellákra térszögek alapján úgy, hogy minden pont egy cellához tartozzon. Összefüggő ponthalmaz alatt azt értjük, hogy a pontokat tartalmazó cellák élszomszédos összefüggő komponenst alkotnak. A keresés egy legalább 3 pontot tartalmazó cella véletlenszerű kiválasztásával, és a pontokra illesztett kezdeti síkkal indul. Ebből a cellából kiindulva egy szélességi keresést hajtunk végre azon élszomszédos cellák mentén, amikben legalább egy pont illeszkedik a síkra. A szélességi keresés eredményeként kapott ponthalmazra illesztett sík különbözhet a kezdeti síktól.

A fenti eljárást, a RANSAC módszer [21] alapján, többször elvégezzük ugyanabból a cellából indulva, de az előző iteráció végén kapott síkkal. Ezzel folyamatosan finomítjuk a kezdeti pontatlan, egy cella pontjaira illesztett síkot.

Az Algoritmus 1-ben használt ponthalmaz értékelő függvény bemutatása. A ponthalmaz kiértékelése során (Algoritmus 1, *score* meghatározása) a következő szempontokat vesszük figyelembe: a ponthalmaz számosságát, a pontok szórását a rájuk illesztett síktól, az alakzat átmérőjének és kerületének eltérését a kalibrációs tábla ismert méreteitől. A fentieket külön-külön számszerűsítjük és összeadjuk. A kiértékelő függvényt úgy alakítjuk, hogy a kisebb érték jelentse a jobbat. Nézzük az egyes szempontokhoz tartozó hiba függvényeket.

Számosság. Legyen n a ponthalmaz mérete. Ekkor a számossághoz tartozó hiba függvény:

$$S_{szám} = 1/n \quad (11)$$

Eltérés a síktól. A síktól való eltérést a pontok síktól való átlagos négyzetes távolságának összegeként fejezzük ki. Legyen d_i az i . pont távolsága a rájuk illesztett síktól. A síktól való eltérés hibafüggvénye:

$$S_{sík} = \sum_i d_i^2/n \quad (12)$$

Az alakzat átmérője. Legyen D' a ponthalmaz két legtávolabbi pontjának távolsága, D pedig a fizikai tábla átmérője. Ekkor az ehhez kapcsolódó hibafüggvény:

$$S_{diam} = (D' - D)^2 \quad (13)$$

Az alakzat kerülete. Legyen P' a síkra vetített ponthalmaz konvex burkának a kerülete, P a fizikai tábla átmérője.

$$S_{perim} = (P' - P)^2 \quad (14)$$

A ponthalmaz *score* értéke pedig a fentiek súlyozott összege:

$$score = \alpha_1 \cdot S_{szám} + \alpha_2 \cdot S_{sík} + \alpha_3 \cdot S_{diam} + \alpha_4 \cdot S_{perim} \quad (15)$$

A paraméterek megfelelő értékeit tesztek alapján határoztuk meg:

$$\alpha_1 = 100, \alpha_2 = 1000, \alpha_3 = 1, \alpha_4 = 1 \quad (16)$$

3.6. Kalibrálás

A kalibrálás célja a LiDAR és a kamera koordináta-rendszerek egymáshoz képesti helyzetének meghatározása, amit a \mathbf{T} kalibrációs mátrix fejez ki. A kamera belső paramétereinek és \mathbf{T} ismeretében a pontfelhő pontjait rávetíthetjük a képre. A cél olyan \mathbf{T} meghatározása, amivel a vetített felhő pontok a lehető legjobban illeszkednek a képen detektált táblára. Ezt számszerűsíti a hibafüggvény.

Ahogy a teljes eljárás lépéseinek összefoglalójában (3.1. fejezet) már szerepelt, a kalibráció három fő fázisból áll: előkalibrálás, rossz felvételek (azaz a hibás frame-k) szűrése, valamint a végső kalibrálás. A jelen fejezetben ez a három fázis és a kalibráció minőségét leíró hibafüggvény kerül bemutatásra.

A kalibrációs hibafüggvény meghatározása. Tekintsünk egy kétdimenziós l egyenest a következő alakban:

$$l(\bar{p}) = a \cdot p_x + b \cdot p_y + c = 0, \text{ ahol } a^2 + b^2 = 1. \quad (17)$$

Legyen \bar{q} egy kétdimenziós pont. Ekkor az egyenes és a pont előjeles távolságát a következőképpen határozzuk meg:

$$\text{dist}(l, \bar{q}) = l(\bar{q}) = a \cdot q_x + b \cdot q_y + c \quad (18)$$

Az $[a, b]$ vektort nevezzük az egyenes normálvektorának.

Definiáljuk a következő függvényt egy l egyenes és egy $Q \subset \mathbb{R}^2$ ponthalmaz viszonyára:

$$\text{diff}(l, Q) = \left(\max_{\bar{q}_j \in Q} \text{dist}(l, \bar{q}_j) \right)^2 \quad (19)$$

Ezt egy 3-dimenziós ponthalmazra is kiterjeszthetjük, ha előbb alkalmazzuk rá a (4) egyenletben definiált $\text{proj}(\mathbf{K}, \mathbf{T})$ leképezést:

$$\text{diff}_{3D}(P, l) = \text{diff}(\text{proj}(\mathbf{K}, \mathbf{T}, P), l) \quad (20)$$

Az egy frame-re vonatkozó kalibrációs hibafüggvény:

$$f(\mathbf{T}, \text{frame}) = \frac{\sum_{j=1}^4 \text{diff}_{3D}(P, \text{border}_j)}{4}, \quad (21)$$

ahol P a frame-en detektált táblapontok, border_j pedig a j . élének egyenese a frame-hez tartozó kameraképen. Egy N frame-ből álló adathalmaz esetén a kalibrációs hibafüggvény a következő módon írható fel:

$$f(\mathbf{T}, \text{frames}) = \frac{\sum_{i=1}^N \sum_{j=1}^4 \text{diff}_{3D}(P_i, \text{border}_{ij})}{4N}, \quad (22)$$

ahol P_i az i . frame-en detektált tábla pontjai, border_{ij} pedig a tábla j . élének az egyenese az i . frame-hez tartozó kameraképen. A visszaadott érték mértékegysége pixel (px), és azt mutatja, hogy mennyire tér el a vetített pontfelhő konvex burka a tábla éleitől.

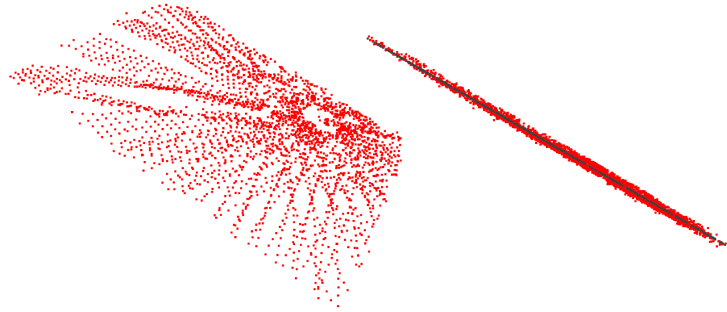
A kalibrálás során a feladat az $f(\mathbf{T}, \text{frames})$ függvény minimumának meghatározása, amit két fázisban, egy szimulált lehtés alapú algoritmussal végzünk. A két keresési fázis között pedig végrehajtottunk egy szűrést a frame-eken.

Előkalibrálás. A keresés első fázisában egy elő-optimalizálás történik a frame-ek egy kisebb részhalmazán. Jelenleg ez 40 frame-t jelent, ami tapasztalati úton lett meghatározva. Az így kapott \mathbf{T}_0 használható előzetes becslésként a továbbiakban.

Hibás frame-k szűrése. Ebben a lépésben a \mathbf{T}_0 -t felhasználva eldobásra kerülnek a hibás frame-k. Akkor minősül egy frame hibásnak, ha a hibafüggvény arra vonatkozó értéke (lásd (21) egyenlet) meghaladja az összes frame-ből számolt medián κ -szorosát. Erre azért van szükség, mert a pontfelhőn nem mindig sikerül jól lokalizálni a táblát. A tábla pozíciójától függően összeérhet a környezettel (5b. ábra), de ki is lóghat a LiDAR látószögéből. A szűrésnek köszönhetően ezek a frame-k automatikusan eldobásra kerülnek (nincs szükség kézzel végzett szelektálásra), ezzel növelve az eljárás használhatóságát.

Végző kalibrálás. A hiba optimalizálásának második fázisában az összes megtartott frame felhasználásával finomítjuk az első fázisban kapott \mathbf{T}_0 -t.

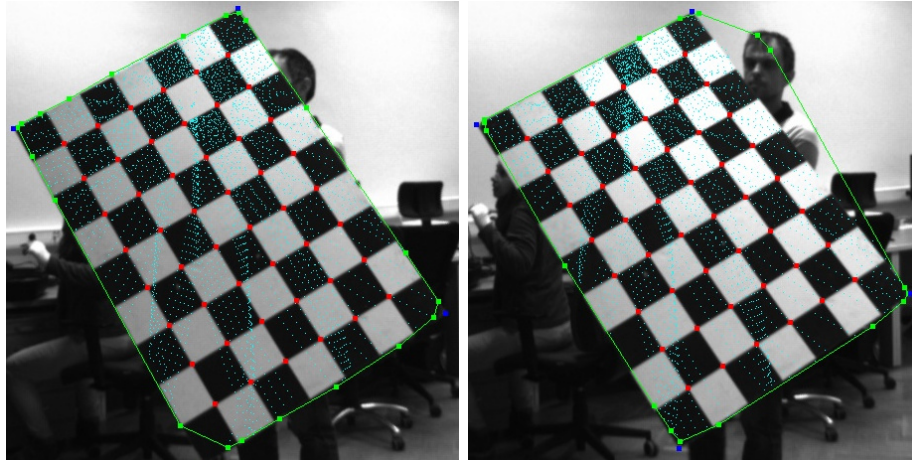
Mindkét fázisban több szálon futtatjuk a keresést egymástól függetlenül, és annak az eredményét fogadjuk el, ami a legjobb $f(\mathbf{T}, frames)$ hiba értéket adja.



4. ábra. LiVOX Avia által detektált tábla a felhőn, különböző nézetekből. Piros: felhőpontok, fekete: az előbbi pontok síkra vetítve. Megfigyelhető a LiVOX eltérő sűrűségű mintázata, ami miatt a klasszikus sarokpont kereső algoritmusok nem mindig működnek.

4. Teszteredmények

Ez a fejezet példákat mutat az eljárás egyes lépéseire beltéri és kültéri környezetben (háttérleválasztás, tábladetekció, pontfelhő képre vetítése) a különböző LiDAR technológiák esetén. Továbbá egy táblázat összefoglalja a tesztelésre



(a) Livox Tele-15 LiDAR és Point Grey kamera. Hiba: 2.7px

(b) Livox Tele-15 LiDAR és Point Grey kamera. Hiba: 19.7px. Hibás detekció; kiszűrésre kerül az előkalibráció után.

5. ábra. A tábla képére vetített LiDAR pontok (apró világoskék), a pontok konvex burka (zöld), a képen detektált belső (piros) és külső (kék) sarokpontok.

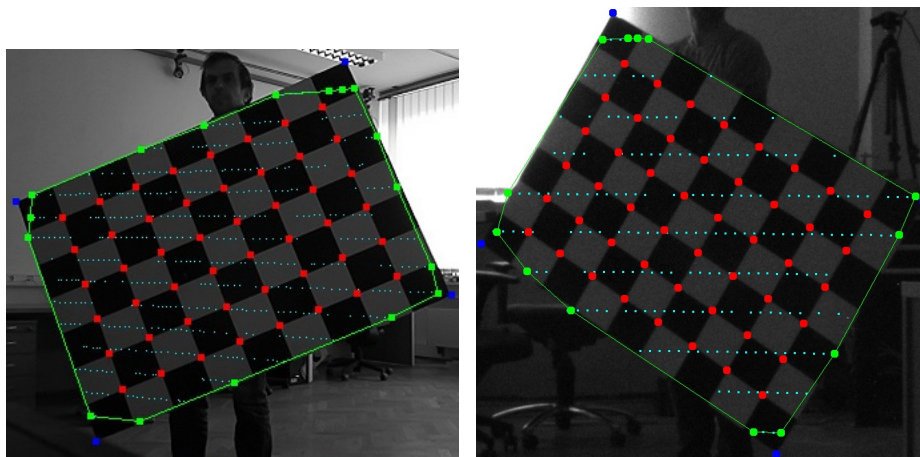
használt eszközöket, és az azokkal elért teszteredményeket. Ezeken túl implementációs részleteket (hardver, paraméter beállítások) is ismertetünk.

Az algoritmus paramétereinek optimális beállítása és az eredmények részletes kiértékelése még folyamatban van. Az eddig elvégzett tesztek alátámasztják a módszer használhatóságát mind forgó LiDAR-ok, mind pedig a nem-repetitív szkennelési mintát használó LiDAR-ok esetén.

Az 4. ábra a Livox Avia LiDAR adatokon detektált sakktábla pontfelhőjét mutatja. Az oldalnézeten látszik a pontok szórása is a vetítési síkhoz képest.

A 5. ábrán a Livox Tele-15 LiDAR és a Point Grey kamera kalibrációjához tartozó tesztképek szerepelnek. A 5a. ábrán egy 2.7px hibaértékű példa látható. A 5b. ábra hibás detekcióra mutat példát, ahol az algoritmus a tábla környezetét is a tábla részének tekintette. Ez olyankor fordulhat elő, ha valamilyen környezeti tárgy túl közel kerül a táblához. A cella mérete (lásd: Algoritmus 2.) határozza meg, hogy ez milyen távolságnál következik be.

Az Ouster LiDAR kalibrálásának tesztképei láthatóak a 6. ábrán két különböző kamerával (Zed2 és Point Grey) a hozzájuk tartozó hiba értékkel együtt. Mivel az algoritmus nem használ sarokdetekciót a pontfelhőn, a sarkok környékén előfordulhat pontatlanság úgy, hogy a tábla élei és a becsült tábla körvonal között pontos az illeszkedés.



(a) Ouster LiDAR és Zed2 kamera. Hiba: 0.98px (b) Ouster LiDAR és Point Grey kamera. Hiba: 2.0px

6. ábra. A tábla képére vetített LiDAR pontok (apró világoskék), a pontok konvex burka (zöld), a képen detektált belső (piros) és külső (kék) sarokpontok.

4.1. Implementáció

A teszteket egy Intel Core i7-7700K CPU-val felszerelt számítógépen futtatuk. Az átlagos futási idő egy frame-re 0.28 sec. A futtatáshoz használt paraméterek az alábbi listában vannak összefoglalva.

4.2. A LiDAR-kamera párok kiértékelése

A 1. táblázat ismerteti a tesztek során használt LiDAR-kamera párokat, a kamera felbontását, a tesztelés körülményeit és az eredményeket. A 3.6. fejezetben leírt módon a rendelkezésre álló frame-k közül elhagyjuk a hibásakat az előkalibráció után. A frame-k teljes száma, illetve a szűrés után megmaradt frame-k száma szerepel a táblázat **össz. frame** és **jó frame** oszlopában. Megfigyelhető, hogy az egyes teszteknel jelentősen eltér a megtartott, azaz jó frame-k aránya. Ennek fő oka, hogy az OpenCV megfelelő függvényei nem mindig ismerik fel a képen táblát, ez főleg az alacsony felbontású kamerák esetében jelentkezik.

A **hiba [px]** oszlopban a (22) egyenlet szerinti hibaérték szerepel. A különböző szenzorpárok összevethetőségének érdekében egy normalizált hiba érték is szerepel a táblázatban (**hiba [px] (norm)**). Mivel a bemutatott eljárás nem használ sarokpont detekciót a pontfelhőn, a gyakori hibamérési módszer - azaz a 3D sarokpontok 2D képre történő vetítési hibája - nem alkalmazható.

LiDAR	kamera	felbontás	beltér/ kültér	össz. frame jó frame	hiba [px]	hiba [px] (norm)*
Livox Avia	Point Grey**	1288 x 964	kültér	2064 1008	1.87	1.45
Livox Avia	Flir GS3-U3	2048 x 2048	kültér	2058 1209	3.92	1.84
Livox Avia	Zed2i	640 x 360	kültér	1538 171	0.84	1.31
Livox Avia	Point Grey	1288 x 964	beltér	1655 814	2.80	2.17
Livox Tele-15	Point Grey	1288 x 964	beltér	492 249	3.34	2.59
Ouster	Point Grey	1288 x 964	beltér	687 135	2.82	2.19
Ouster	Zed2	640 x 360	beltér	412 37	1.03	1.62
Hesai XT32M2X	Zed2i	640 x 360	kültér	769 81	0.87	1.36

1. táblázat. A tesztelés során használt LiDAR és kamera párok. A hiba értékét a 3.6. fejezetben leírtak szerint határoztuk meg. Livox integrációs idő 0.1sec minden adathalmaznál.

* 1000 pixel szélességű képre normalizált hiba, a könnyebb összehasonlíthatóság érdekében.

** Halszem optikával.

5. Összefoglalás

A cikk egy sakktábla alapú LiDAR-kamera kalibrációs eljárást mutatott be. A megoldás teljes mértékben automatizált, a szoftver futtatása során nincs szükség kézi beavatkozásra. Képes kiszűrni az esetleges hibás frame-eket, ezzel növelve az algoritmus robusztusságát. Beltéren és kültéren egyaránt tesztelésre került, és többféle LiDAR-kamera kombinációval is eredményesen működött. Az algoritmus jelenleg is fejlesztés alatt áll, a jövőben többek között további tesztelést és más algoritmusokkal történő összehasonlítást is tervezünk. A futtatás során használt paraméterek optimális értékének és a minimálisan szükséges frame-k számának meghatározása szintén megoldandó feladat. Érdeemes megvizsgálni az integrációs idő hatását a kalibráció pontosságára is a Livox LiDAR-ok esetén. Összességében elmondható, hogy az eddig elvégzett tesztek alapján az algoritmus ígéretes megoldás általános gyakorlati alkalmazásra, a LiDAR technológiától és a kamera típusától függetlenül.

Köszönetnyilvánítás

A publikációban szereplő kutatást a HUN-REN SZTAKI részben az Autonóm Rendszerek Nemzeti Laboratórium keretében (RRF-2.3.1-21-2022-00002), valamint a TKP2021-NVA-01 projekt támogatásával valósította meg. A TKP2021-NVA-01 számú projekt az Innovációs és Technológiai Minisztérium Nemzeti Kutatási Fejlesztési és Innovációs Alapból nyújtott támogatásával, a TKP2021-NVA pályázati program finanszírozásában valósult meg.

Hivatkozások

1. C. Yuan, X. Liu, X. Hong, and F. Zhang, „Pixel-level extrinsic self calibration of high resolution lidar and camera in targetless environments,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7517–7524, 2021.
2. Z. Bai, G. Jiang, and A. Xu, „Lidar-camera calibration using line correspondences,” *Sensors*, vol. 20, no. 21, 2020.
3. T. Ma, Z. Liu, G. Yan, and Y. Li, „Crlf: Automatic calibration and refinement based on line feature for lidar and camera in road scenes,” 2021. [Online]. Available: <https://arxiv.org/abs/2103.04558>
4. G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, „Automatic extrinsic calibration of vision and lidar by maximizing mutual information,” *Journal of Field Robotics*, vol. 32, no. 5, pp. 696–722, 2015.
5. K. Koide, S. Oishi, M. Yokozuka, and A. Banno, „General, single-shot, targetless, and automatic lidar-camera extrinsic calibration toolbox,” 2023. [Online]. Available: <https://arxiv.org/abs/2302.05094>
6. P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, „Superglue: Learning feature matching with graph neural networks,” 2020. [Online]. Available: <https://arxiv.org/abs/1911.11763>
7. R. Ishikawa, T. Oishi, and K. Ikeuchi, „Lidar and camera calibration using motion estimated by sensor fusion odometry,” 2018. [Online]. Available: <https://arxiv.org/abs/1804.05178>

8. Z. Huang, X. Zhang, A. Garcia, and X. Huang, „A novel, efficient and accurate method for lidar camera calibration,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 14 513–14 519.
9. H. Huang, M. Zhang, L. Li, J. Hu, and H. Wang, „Gtsclib: Generalized target segmentation for target-based extrinsic calibration of non-repetitive scanning lidar and camera,” *IEEE Transactions on Automation Science and Engineering*, (early access), pp. 1–13, 2024.
10. J. Jiao, F. Chen, H. Wei, J. Wu, and M. Liu, „Lce-calib: Automatic lidar-frame/event camera extrinsic calibration with a globally optimal solution,” *IEEE/ASME Transactions on Mechatronics*, vol. 28, no. 5, pp. 2988–2999, 2023.
11. L. F. T. Fu, N. Chebrolu, and M. Fallon, „Extrinsic calibration of camera to lidar using a differentiable checkerboard model,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 1825–1831.
12. J. Cui, J. Niu, Y. He, D. Liu, and Z. Ouyang, „Aclc: Automatic calibration for non-repetitive scanning lidar-camera system based on point cloud noise optimization,” *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1–14, 2024.
13. J. Han, Z. Zhang, Z. Wang, J. Li, X. Guo, and X. Kang, „Extrinsic calibration of a binocular camera and lidar based on neural networks,” *IEEE Sensors Journal*, vol. 23, no. 23, pp. 29 271–29 282, 2023.
14. F. Itami and T. Yamazaki, „A simulation study on calibration of a lidar with respect to a camera by using point and plane constraints,” 11 2023.
15. H. Abbasi, A. Dey, I. Lam, Z. Sharifisoraki, E. Ali, M. Amini, S. Rajan, J. Green, and F. Kwamena, „A step-by-step approach for camera and low-resolution-3d-lidar calibration,” in *2023 IEEE International Conference on Consumer Electronics (IC-CE)*, 2023, pp. 1–5.
16. M. Kaiser, T. Brusa, M. Bertsch, M. Wyss, S. Ćuković, G. Meixner, and V. M. Koch, „Extrinsic calibration for a modular 3d scanning quality validation platform with a 3d checkerboard,” *Sensors*, vol. 24, no. 5, p. 1575, 2024.
17. B. Zhang, Y. Zheng, Z. Zhang, and Q. He, „Lidar and camera calibration using pyramid and checkerboard calibrators,” in *2023 IEEE 8th International Conference on Big Data Analytics (ICBDA)*. IEEE, 2023, pp. 187–192.
18. G. Yan, F. He, C. Shi, P. Wei, X. Cai, and Y. Li, „Joint camera intrinsic and lidar-camera extrinsic calibration,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 446–11 452.
19. T. Tóth, Z. Pusztai, and L. Hajder, „Automatic lidar-camera calibration of extrinsic parameters using a spherical target,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 8580–8586.
20. Z. Pusztai, I. Eichhardt, and L. Hajder, „Accurate calibration of multi-lidar-multi-camera systems,” *Sensors*, vol. 18, no. 7, 2018. [Online]. Available: <https://www.mdpi.com/1424-8220/18/7/2139>
21. M. A. Fischler and R. C. Bolles, „Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, no. 6, p. 381–395, 1981.