UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH

PRÍRODOVEDECKÁ FAKULTA

ZOBRAZOVANIE GEOGRAFICKÝCH OBJEKTOV V ROZŠÍRENEJ REALITE NA SMARTFÓNOCH

2015 **Jakub Gregorek**

Poďakovanie

Veľké poďakovanie patrí vedúcemu práce RNDr. Františkovi Galčíkovi, PhD. a Mgr. Michalovi Gallayovi, PhD. z Ústavu geografie PF UPJŠ.

Obsah

Úvod a ciele práce

Slovensko je hornatá krajina s atraktívnym prostredím a obrovským potenciálom v oblasti turistického ruchu. Pri výletoch po našich horách sa turisti často stretávajú s veľkolepým výhľadom na okolitú krajinu. Mnohokrát pritom nevedia, na aké hory sa pozerajú, či sa tam nachádzajú turistické chaty, chodníky, lanovky alebo iné zaujímavé miesta a potenciálne ciele ich budúcich výletov.

Na riešenie tohto problému môžu byť použité dobre rozšírené smartfóny. Už dnes existuje viacero aplikácií, ktoré sa dotýkajú tejto oblasti. Najviac nás zaujali open source riešenia *Show Me Hills* a *Mixare*. *Show Me Hills* funguje na platforme *Android* a zobrazuje mená hôr v obraze snímanom kamerou. Aplikácia *Mixare* podporuje aj *iOS* a je o niečo všeobecnejša, v obraze vyznačuje *POI - Point Of Interest*, s ktorými sa stretávame napríklad v auto navigáciách.

Spomínané programy vôbec neanalyzujú videný obraz a sú úplne „slepé“. Odhadujú polohu objektov len na základe znalosti zorného uhla kamery a dát získaných z GPS a kompasu. S tým súvisí ich nepresnosť a potreba kalibrácie kamery pri prvom použití.

V práci prezentujeme spôsob, ako tieto aplikácie vylepšiť. Nami navrhované vylepšenie je založené na detekcii horizontu v snímanom obraze. Tento pre človeka triviálny problém nie je až taký triviálny pre počítače alebo smartfóny. Ako vieme, pixely obrazu môžeme reprezentovať napríklad ako trojice čísel prislúchajúcich červenej, modrej a zelenej farbe. Ak si predstavíme namiesto obrazu dvojrozmerné pole takýchto trojíc, asi už nebudeme schopní identifikovať horizont tak ľahko ako predtým. Úspešná detekcia horizontu v obraze stále nestačí na to, aby sme vedeli povedať aké objekty sa nachádzajú v zornom poli kamery.

Ďalším krokom je preto výpočet očakávaného horizontu, ktorý používateľ uvidí z daného miesta. Tu je potrebná znalosť terénu okolitého prostredia.

Následným porovnaním videného a vypočítaného horizontu namapujeme videnú krajinu na digitálny model prostredia. Na základe toho budeme schopní presne identifikovať jednotlivé hory a zároveň určiť polohu ďalších turisticky atraktívnych miest.

Výzvou v tejto oblasti, je navrhnúť spomínané postupy s ohľadom na obmedzený výkon smartfónov a minimalizovať veľkosť uložených geodát.

Ciele práce

1. Preskúmať a analyzovať existujúce riešenia na zobrazovanie geografických objektov v rozšírenej realite na smartfónoch.
2. Preskúmať a prakticky overiť možnosti vylepšenia identifikácie geografických objektov s využitím metód počítačového videnia (napr. aj s využitím výškového modelu lokálneho prostredia).

1 Návrh riešenia

Naše riešenie sa dá rozdeliť na tri samostatné podproblémy:

- detekcia horizontu v obraze snímanom kamerou,
- výpočet predpokladaného horizontu z geografických dát,
- namapovanie detegovaného a vypočítaného horizontu.

2 Detekcia horizontu

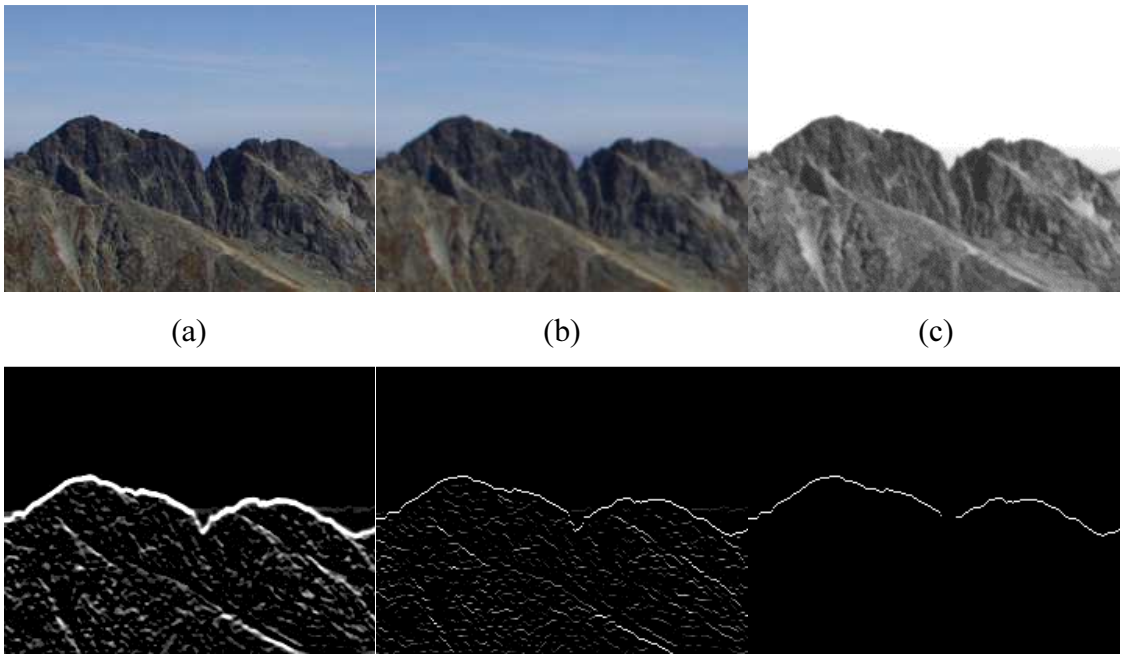
Štandardné techniky pri identifikovaní objektov ako SURF a SIFT sú v našom prípade nevhodné z dôvodu citlivosti na svetlo, vplyvu osvetlenia, prekrytia objektov atď. Preto bolo potrebné identifikovať iný typ význačných bodov, kde sa ako stabilná štruktúra núka krivka horizontu.

2.1 Detekcia na báze detektora hrán

Návrh algoritmu sme založili na farbách význačných pre oblohu. Viacero prác [4,5] v rámci preprocessingu predpokladá, že na oblohe prevládajú odtiene modrej.

Pre elimináciu šumu aplikujeme na vstupný obraz gaussovské rozostrenie. Následne prevedieme farebný obraz na šedotónový, kde sa na výslednej svetlosti pixelov podieľa najmä modrá zložka svetla. Potom aplikujeme sobelov operátor, ktorý funguje ako detektor hrán, na zistenie strmosti prechodov medzi odtieňmi sivej v smere y osi. Horizont by sa mal vyznačovať prechodom od svetlých odtieňov k tmavým, preto

opačné prechody odfiltrujeme, a zvyšné preškálujeme a tresholdujeme. Potom aplikujeme po stĺpcoch non-maximum suppression. V závere vytvoríme z takto upraveného obrázka graf ohodnotený podľa svetlosti pixelov a spustíme algoritmus na zistenie minimálneho rezu.



(a) (b) (c)

(d) (e) (f)

Obr. 1: Postup výpočtu horizontu: a) originál, b) gaussian blur, c) šedotónový obraz, d) odfiltrovaný, preškálovaný a tresholdovaný sobelov filter, e) non-maximum suppression, f) výsledok

```
pre každé [x, y] obrazu {
  Svetlosť <- 0.299 * Červená(Farebný [x,y]) + 0.587 *
           Zelená(Farebný [x,y]) + 0.114 * Modrá(Farebný
           [x,y]);
  Modrosť <- Max(Min(Modrá(Farebný[x,y]) -
           Zelená(Farebný[x,y]), Modrá(Farebný[x,y]) -
           Červená(Farebný[x,y])), 0);
  Šedotónový[x,y] <- 0.5 * Svetlosť + 1 * Modrosť;
}
```

Kód 1: Prevod farebného vstupu na šedotónový obraz

Algoritmus využíva základné funkcie *OpenCV* a stačí mu niekoľko násobný prechod vstupného obrazu, z čoho vyplýva jeho rýchlosť.

2.2 Detekcia metódou region growing

Algoritmus založený na region growingu sme na rozdiel od [6] implementovali prehľadávaním do šírky.

Na vstupný obraz najprv aplikujeme bilaterálny filter v ktorom váhy závisia nie len od Euklidovskej vzdialenosti pixelov ale aj rádiometrických rozdielov. V porovnaní s gaussovským rozostrením zachováva ostré hrany.

Následne obraz prevedieme do *CieLabu* – uniformného trojdimenzionálneho farebného priestoru. Na rozdiel od *RGB* sa vyznačuje vlastnosťou, že podobnosť farieb vnímaných ľudským okom je vyjadrená Euklidovskou vzdialenosť bodov reprezentujúcich dané farby. Spustíme vyhľadávanie do šírky, ktorým rozdelíme obraz na oblasti na základe podobnosti farieb.

```
PočetOblastí = 0;

pre každé x, y {
  ak Area[x,y] je Nedefinovaná {
    početOblastí++;
    Oblasť[x,y] <- početOblastí;
    pridaj [x,y] do radu Q;

  kým Q nie je prázdny {
      vyber [xx,yy] z radu Q;
      pre všetky nespracované susedné pozície [xs,ys]
      pozície [xx,yy] {
        ak podobnosť farieb C[xx,yy] a C[xs,ys] < prah {
          Oblasť[xs,ys] <- početOblastí;
          pridaj [xs,ys] do Q;
        }
      }
    }
  }
}
```

Kód 2: Delenie obrazu na oblasti na základe farby

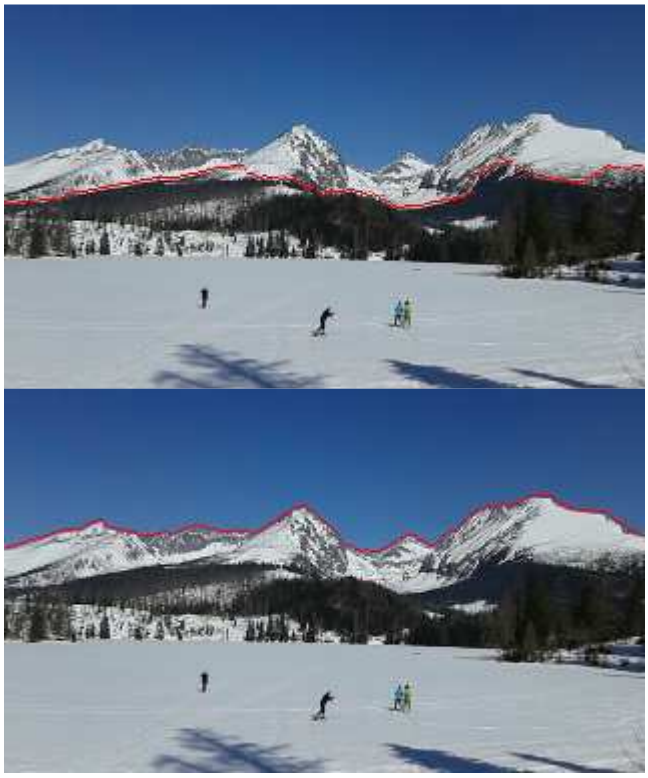
Na záver prehlásime všetky príliš svetlé oblasti nachádzajúce sa v hornej osmine obrazu za časti oblohy a určíme horizont.

2.3 Porovnanie algoritmov

Pre zber dát na testovanie algoritmov sme navrhli jednoduchú aplikáciu, ktorá zachytáva obraz z kamery smartfónu v náhľadovom rozlíšení a ukladá zemepisné súradnice používateľa, azimut, zorný uhol kamery, deklináciu magnetického poľa zeme, údaje z akcelerometra a kompasu a ďalšie parametre.

Algoritmus na báze detektora hrán bol testovaný na rôznych vstupoch a dáva dobré výsledky na fotkách letnej krajiny s jasnou oblohou. Na dátach, ktoré sme zozbierali v Tatrách so zasneženými horami zlyhal a ukázal sa ako nevhodný na identifikáciu horizontu pri pohľade na zimnú krajinu. To bolo hlavným dôvodom pre návrh detekcie založenej na metóde region growing.

Tento algoritmus sa vie zároveň lepšie vysporiadať s prudkými prechodmi medzi svetlými a tmavými farbami v oblastiach mimo horizontu, zasneženou krajinou, oblakmi atď.



Obr. 2: Nesprávne (hore – prvý návrh) a správne (dole - aktuálny návrh) zdetegovaný horizont pri pohľade na zimnú krajinu pri Štrbskom plese.

Obidva algoritmy sme prototypovali v C++ s použitím OpenCV knižnice [1].

3 Výpočet horizontu

Pri výpočte horizontu sme používali výškovú mapu s rozlíšením $8{,}33.10^{-4}$ stupňa, ktorá bola vytvorená počas špeciálnej misie raketoplánu *Endeavour* v roku 2000 (SRTM).

Krajinný priestorový model sme získali v textovom ERSI Grid formáte, ktorý sa ukázal ako nevhodný. Jeho hlavnými nevýhodami sú neefektívny prístup k dátam a veľkosť, v našom prípade až 280 MB.

3.1 Kompresia údajov

V snahe riešiť spomínané problémy sme navrhli vlastný binárny formát, s čím súvisí aj implementácia konvertora do tohto formátu.

Najprv sme previedli dáta do binárnej podoby a pre uloženie hodnôt nadmorskej výšky zvolili dátový typ *short*. Výsledkom bol 140 MB veľký súbor. Použitie *GZIP* kompresie zmenšilo dáta na 65 MB. V závere sme zvolili diferenčné uloženie dát. Každá hodnota je rozdielom predchádzajúcej hodnoty a samej seba. Vďaka lepšej komprimovateľnosti sme sa tak dostali až na 50 MB.

3.2 Mapová mriežka

Kompresia ale nerieši problém s problematickým prístupom k dátam a nutným načítavaním celého súboru pred výpočtom horizontu. To je zdĺhavé a z pohľadu pamäťovej náročnosti nerealizovateľné na mobilných zariadeniach.

Mapu sme preto rozdelili do štvorcových územných buniek, ktoré môžu obsahovať viacero typov dátových kolekcií uchovávajúcich výškové mapy rôznej presnosti, informácie o vrcholoch hôr atď.

Vnútorná štruktúra súboru je tvorená hlavičkou mapy na začiatku, nasledovanou hlavičkami buniek, ktoré obsahujú metadáta a hlavičky dátových kolekcií. Tie odkazujú na samotné dáta na konci súboru.

Obr. 3: Vnútorná štruktúra súboru s geodátami

Metadáta v bunkách zväčšili výsledný súbor o niekoľko MB v závislosti od parametrov buniek, ale umožňujú rýchly prístup k dátam bez potreby čítania celého súboru.

3.3 Algoritmus na výber relevantných buniek

Pred výpočtom pohľadu na krajinu náš algoritmus vyberie bunky mapy, ktoré obsahujú informácie o území videnom používateľom a spracúvanie dát následne prebieha po bunkách. Vďaka tomu nepotrebujeme v jednom okamihu držať v operačnej pamäti veľké množstvá dát a nepresiahneme tak limit operačnej pamäte vyhradenej pre jednu aplikáciu.

```
Bunky <- BunkyVOkolí(lat, long);

pre každé b z Bunky {
  pre každý rohovýBod b {
    ak UholOsZ(rohovýBod) je medzi(azimut-zornýUhol/2,
    azimut+zornýUhol/2) {
      pridaj b do ViditeľnéBunky;
    }
  }
}
```

Kód 3: Viditeľnosť buniek

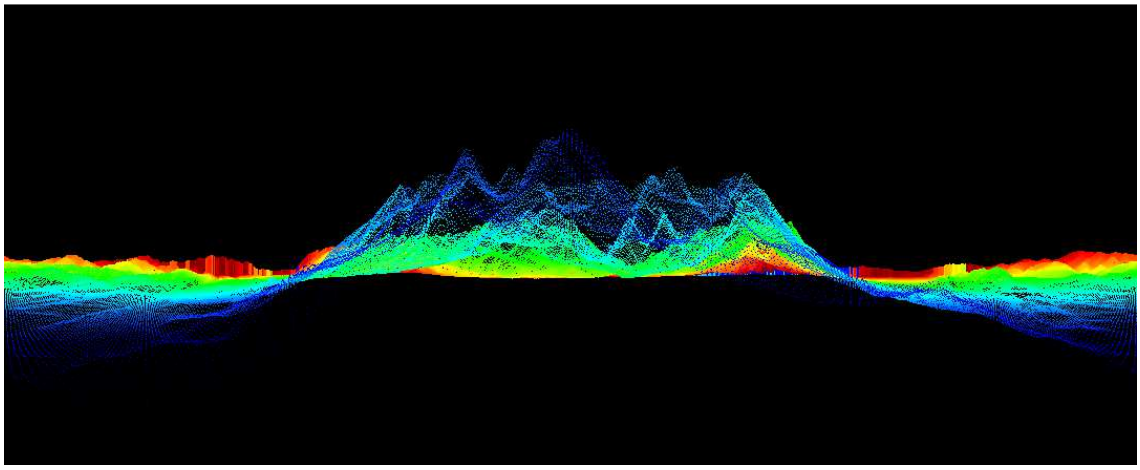
3.4 Výpočet panoramatického pohľadu

Vstupom algoritmu na výpočet sú zemepisné súradnice, nadmorská výška používateľa, azimut a parametre kamery. Výpočet začíname presunutím stredu súradnicovej sústavy výškovej mapy do bodu pozorovateľa, pričom súradnicami sú zemepisná výška, šírka a nadmorská výška. Následne otočíme všetky body okolo osi nadmorskej výšky o uhol azimutu. Transformované body na záver premietneme v perspektíve. Výsledok vidíme nasledujúcom obrázku.

```
BunkyNaProjekciu <- ViditeľnéBunky(lat, long, alt, azimut,
zornýUhol);

pre každé b z BunkyNaProjekciu {
  VýškovéBodyBunky <- NačítajVýškovéBodyBunky(b);
  pre každé [lat, long, alt] z VýškovéBodyBunky {
    [x, y] <- Projekcia(lat, long, alt, azimut, zornýUhol);
    Panoráma[x, y] <- min(Panoráma[x, y],
                  Vzdialenosť(GPSPoloha, [lat, long]));
  }
}
```

Kód 4: Projekcia po bunkách



Obr. 4: výškové body premietnuté do 360° panorámy, farba bodov závisí od ich vzdialenosti od používateľa (najbližšie tmavomodrá, nasledovaná svetlomodrou, zelenou, žltou, oranžovou až po červenú v najväčšej vzdialenosti).

Vieme generovať pohľady dané azimutom a horizontálnym zorným uhlom kamery ale aj 360° panorámy.

Ukázalo sa, že pri výpočtoch nie je potrebné zohľadňovať zakrivenie zemského povrchu. Predstavuje približne 12 metrov vo vzdialenosti 100km, čo v porovnaní s presnosťou výškovej mapy (16 m) nie je významné hodnota.

3.5 Výpočet horizontu

Po premietnutí sa medzi bodmi obrazu nachádzajú prázdne miesta. Tie vyplníme tak, že každý bod spojíme úsečkou s jeho štyrmi susedmi. Následne vypĺňame priestor vo vzniknutých štvorcoch. Prechádzame hĺbkovou mapou po stĺpcoch a do vzdialenejších bodov vpisujeme vždy najbližšiu hodnotu vzdialenosti, ktorú sme prečítali. Dostávame tak súvislú hĺbkovú mapu z ktorej vieme určiť horizont. Pre body horizontu platí, že vzdialenosť bodu nad nimi je v nekonečne. V našom prípade ponechávame aj body so vzdialenosťou horného suseda väčšou ako 10 km. Vyznačujeme tak terén, ktorý môže byť súčasťou horizontu v počasí s menšou dohľadnosťou.

```
pre každé [x,y,vzdialenosť] z transformovanéBody {
  ak [x,y] je v projekčnej ploche {
    depthMap[x,y] <- min(depthMap[x,y], vzdialenosť);
    ak hornýSused([x,y]) [xh,yh] je v projekčnej ploche {
      spoj([x,y],[xh,yh]);
    }
    ak pravýSused([x,y]) [xr,yr] je v projekčnej ploche {
      spoj([x,y],[xr,yr]);
    }
  }
}
```

Kód 5: Spájanie susedných bodov

```
pre každý stĺpec z projekčnej plochy {
  currentDepth <- MAX_VALUE;
  pre x medzi(0,dĺžka(stĺpecc)) {
    ak (stĺpec[x] > currentDepth) {
      stĺpec[x] = currentDepth;
    } inak {
      currentDepth = stĺpec[x];
    }
  }
}
```

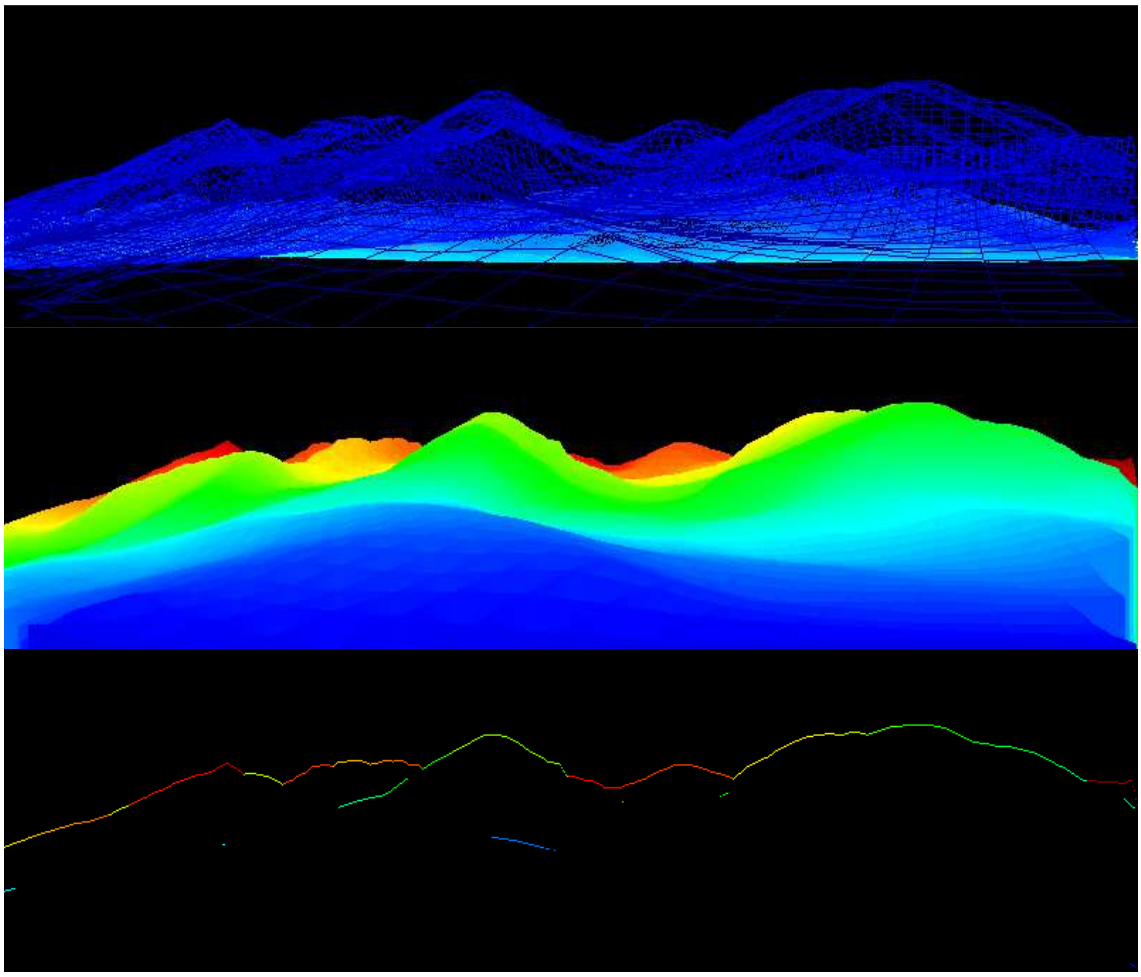
Kód 6: Vypĺňanie priestoru medzi bodmi

```
pre každý stĺpec z projekčnej plochy {
  currentDepth <- MAX_VALUE;
  lastHorizonPixel <- -(ignoredPixelsBelow+1);
  pre x medzi(0,dĺžka(stĺpecc)) {
    ak (stĺpec[x] > currentDepth) {
      stĺpec[x] = MAX_VALUE;
    } inak {
      ak currentDepth - stlpec[x] < distanceTreshold {
        currentDepth = stĺpec[x];
        stĺpec[x] = MAX_VALUE;
      } inak {
        currentDepth = stĺpec[x];
        ak x – lastHorizonPixel <= ignoredPixelsBelow {
          stĺpec[x] = MAX_VALUE;
        } inak {
          lastHorizonPixel = x;
        }
      }
    }
  }
}
```

Kód 7: Filtrovanie horizontu

Vzhľadom k obmedzenému výkonu mobilných zariadení bude prebiehať výpočet horizontu pre polohu získanú z GPS v priebehu štartu aplikácie. Prepočítaný bude v prípade, ak sa používateľ presunie o vzdialenosť, pri ktorej je badateľná jeho zmena.

Efektívny prístup k dátam a algoritmus na výber relevantných buniek výrazne skrátil čas potrebný na generovanie horizontu zo 45,5 na 1,4 sekundy. Použitie kompresie má pritom minimálny dopad na výkon. Časy boli merané na priemernom notebooku.



Obr. 5: hore – sieť susedných bodov, stred – vyplnený medzibodový priestor, dole – výsledný horizont

Záver

V práci sme preskúmali a zanalyzovali súčasné riešenia na zobrazovanie geografických objektov v rozšírenej realite a navrhli sme spôsob ich vylepšenia s využitím počítačového videnia a znalosti geografického prostredia.

Okrem využitia dát z GPS a kompasu, navrhujeme analyzovať obraz snímaný kamerou a identifikovať v ňom horizont, ktorý dokážeme úspešne detegovať. Zároveň, sme ukázali spôsob ako efektívne počítať horizont z výškovej mapy okolitého prostredia.

Zarovnanie videnej krajiny na známy digitálny model namapovaním horizontov patrí k aktuálnym výzvam práce a je v štádiu riešenia.

Navrhovaný prístup ponúka široké možnosti ďalšieho rozšírenia. Na základe presného namapovania pozorovanej krajiny na známy model prostredia je možné určiť súradnice a nadmorskú výšku každého bodu zobrazeného na displeji smartfónu a zostrojiť hĺbkovú mapu pozorovanej krajinnej oblasti. Dostávame exteriérový *Kinect*, čo je skvelý základ pre pokročilé zobrazovanie prvkov rozšírenej reality ako napríklad turistické chodníky, vrstevnice, chaty, lanovky atď. Okrem toho sa bude možné do budúcna zbaviť potreby kompasu a kalibrácie kamery.

Podobné aplikácie majú navyše do budúcna perspektívu prispieť k rozvoju turistického ruchu na Slovensku a stať sa sprievodcom turistov po našich horách.

Zoznam použitej literatúry

[1] Baggio, Daniel Lélis. Mastering OpenCV with Practical Computer Vision Projects. Packt Publishing Ltd, 2012., ISBN: 978-1-84951-783-6

[2] Carmigniani, Julie, et al. "Augmented reality technologies, systems and applications." Multimedia Tools and Applications 51.1 (2011): 341-377.

[3] Boroujeni, Nasim Sepehri, S. Ali Etemad, and Anthony Whitehead. "Robust horizon detection using segmentation for uav applications." Computer and Robot Vision (CRV), 2012 Ninth Conference on. IEEE, 2012.

[4] Irfanullah, Kamal Haider, Qasim Sattar, Sadaqat-ur-Rehman, Amjad Ali, An Efficient Approach for Sky Detection, In IJCSI International Journal of Computer Science Issues, ISSN (Print): 1694-0814, 2013, Vol. 10, Issue 4, No 1

[5] T.G. McGee, R. Sengupta, and K. Hedrick. Obstacle detection for small autonomous aircraft using sky segmentation. In ICRA 2005, 2005.

[6] Schmitt, F., Priese, L. Sky Detection in CSC-segmented Color Images.. In VISAPP (2) : INSTICC Press, 2009. - ISBN 978-989-8111-69-2, S. 101-106