Gradus Vol 1, No 1 (2014) 74-80
ISSN 2064-8014

VALÓS IDEJŰ KÉZ ÉS UJJHEGY ÉRZÉKELŐ ELJÁRÁS

A REAL-TIME HAND AND FINGERTIP DETECTION METHOD

Bolla Kálmán [1], Szatmári Ferenc [1]*

[1] Informatika Tanszék, Gépipari és Automatizálási Műszaki Főiskolai Kar, Kecskeméti Főiskola, Magyarország

Kulcsszavak:
képfeldolgozás, kézfelismerés

Keywords:
image processing, hand detection

Cikktörténet:
Beérkezett 2015. október 10.
Átdolgozva 2015. október 31.
Elfogadva 2015. november 5.

Összefoglalás
Munkánkban egy vizuális érzékelésen alapuló kézfej és ujjhegy érzékelő megoldást mutatunk be, amely egyúttal valós idejű feldolgozást is lehetővé tesz. Algoritmusunk helyességét kísérletek segítségével bizonyítottuk, mértük annak futási idejét és működésének pontosságát is.

Abstract
In our paper a visual perception based hand segmentation and finger detection method is presented which provides real-time execution, as well. The present algorithm was proven by experiments; the algorithm execution time and the detection accuracy were also measured.

1 Bevezetés

Az ember és gép közötti interakció napjainkban egyre fontosabbá válik. Számos alkalmazásban használják már az effajta kommunikáció előnyeit, kiváltképp a szórakoztató elektronikában. A gépek ember által való vezérlésében nagy szerep jutott az emberi kéz mozdulatainak. Ez nem is meglevő, mivel az emberi kéz több mint 25 szabadsági fokkal rendelkezik [1], így rengeteg különböző lehetőséget biztosít a gépek irányításához. Azonban egy robusztusan működő, pontos kézmozdulat és pozíciót meghatározó eljárás létrehozása komoly kihívást jelent. Az adott témában a kutatók már számos megoldást javasoltak, az utóbbi időben pedig előtérbe került az algoritmusok végrehajtási idejének csökkentése és a valós idejű végrehajtás megvalósítása [1][2]. Célunk egy saját kézfelismerő eljárás kidolgozása volt, mellyel a kéz és az ujjvégek pozícióját pontosan meg tudjuk határozni. Ezen túl a mi eljárásunkban is nagy figyelmet kap a valós idejű szegmentáció megvalósítása.

Cikkünk a következőképpen épül fel: 2. fejezetben részletezzük saját kézfej érzékelő eljárásunkat, a 3. fejezetben pedig az ujjvégek azonosításának módszerét mutatjuk be. 4. fejezetünkben a kísérletek és tesztelések során tapasztalt eredményeinket írjuk le.

2 A kézfej érzékelése

Célunk egy olyan kézfej és ujjhegy érzékelő eljárás létrehozása volt, amely megbízhatóan és pontosan működik – elsősorban laboratóriumi körülmények között – és általa megvalósítható az ember-gép közötti interakció. Mivel a kéz és az ujjak számos különböző pozíciót fel tudnak venni, ezért néhány korlátozást kell bevezetnünk annak érdekében, hogy az algoritmusunk segítségével valós idejű végrehajtást érjünk el. A kéz érzékelése akkor fog megvalósulni, ha a felhasználó tenyere a kamera érzékelőjével szemben helyezkedik el, azzal párhuzamosan, így számos kézpozícióval nem kell már a későbbiekben foglalkozni. Továbbá a kéz, a rögzített képen több méretben is megjelenhet, attól függően, hogy az emberi kéz közel vagy távol helyezkedik el a

* E-mail cím: bolla.kalman@gamf.kefo.hu

kamera érzékelőtől. Ebben az esetben is csökkenteni kell a felmerülő esetek számát, ezért a kamerától számolt 60 cm és 80 cm közötti távolságban fog az eljárásunk megfelelően működni.

Az általunk kidolgozott megoldásban Microsoft Kinect eszközt használunk a rendszer érzékelőjeként, segítségével a hagyományos kamerakép mellett mélységinformációkhoz is juthatunk. A mélységi adatokat elsősorban arra használtuk, hogy a 60 cm-nél közelebbi és 80 cm-nél távolabbi objektumokat ki tudjuk szűrni a további feldolgozási lépésekből. Mivel a mélységkép után a színes kép alapján is szeretnénk dolgozni, ezért a két szenzor által rögzített információkat egymásra kellett illeszteni. Az illesztés eredményeképpen egy olyan képet kaptunk, amely csak a meghatározott intervallumban található objektumok szín információit tartalmazta. Minden egyéb pixelt háttérnek tekintettük és fekete színnel jelöltük. Ezek után csak az illesztett színes képek alapján dolgoztunk, amely feldolgozását három részre bontottuk. Első részében egy előfeldolgozást hajtunk végre annak érdekében, hogy a kézfelismerés szempontjából fontos régiókat kiemeljük. Második lépésként megvalósítjuk magának a kézfej képen való azonosítását, végül a tenyér és az ujjhegyek érzékelése történik.

Az előfeldolgozási mechanizmus alapvetően a bőrszín szegmentációját alkalmazza, amivel részben megvalósítható a kézfej régió elkülönítése a további objektumoktól. Szakirodalomban található megoldások közül többet is megvizsgáltunk, ezek közül az YCbCr színtartomány alapú szegmentációt [3] találtuk a legmegfelelőbbnek. Ebben a lépésben a bemeneti (már illesztett) színes képet egy bináris képpé alakítjuk, ahol a 1-essel jelölt pixeleket a bőrfelszín részének, a 0-val jelölt pixeleket pedig a háttér részének tekintjük. Az előfeldolgozás eredményét a 1. ábra (a) része mutatja.

Színalapú szegmentálás után a kézre hasonlító régiót ki kell választani a kamera előterébe kerülő többi bőrszínű objektum közül (például emberi arc). Erre a feladatra a szegmentációból kapott bináris képet fogjuk használni és egy előre definiált Haar-leíróval, valamint egy osztályozó algoritmussal valósítjuk meg a kezet tartalmazó képrész kiválasztását. Az emberi kezet egyetlen Haar-leíróval modelleztük (1. ábra (c) része), feladata a potenciális régiók kiválasztása a bináris képen. Erre a megoldásra azért esett a választásunk, mert azt feltételezzük, hogy egy terület alapú megoldással meghatározhatjuk hol található a kézfej és alkar egy része a képen. Az 1. ábra (c) részén látható a előre definiált területfelosztás, egy négyszög alakú képrészleten (1. ábra (c) részén fekete színnel) a bőrszínhez tartozó pixeleket feltételezünk, körülötte (1. ábra (c) részén fehér színnel) pedig egyéb színű objektumokat. A leíró alapján három értéket számolunk: a fehér és fekete területek alatt található pixelértékek összege és a két terület különbsége.

A korábban meghatározott korlátozások mellett is rengeteg különböző előfordulás lehet (elsősorban az ujjak lehetséges pozíciói miatt), ezért a bináris képen további transzformációkat kell végrehajtani a sikeres osztályba sorolás érdekében. Először is a kapott képet alulmintavételezzük 160x120 pixel felbontásra. Erre azért van szükség, mert a kézfej érzékelésénél elegendő egy durva alakzat használta is, az ujjak pozíciójának pontos meghatározására csak a későbbiekben lesz szükség. Az alulmintavételezett képen végrehajtott morfológiai nyitás és zárás után egy olyan eredményt kaptunk, amellyel csökkenteni tudtuk a különbséget a széttárt és az összecsukott ujjak pozíciói, valamint a hüvelykujj helyzetének eltérései között. Ezzel a módszerrel a későbbi osztályzás során sokkal pontosabb eredményt fogunk kapni. A mintavételezett kép és a morfológiai előfeldolgozás eredménye megtekinthető a 1. ábra (b) részén.

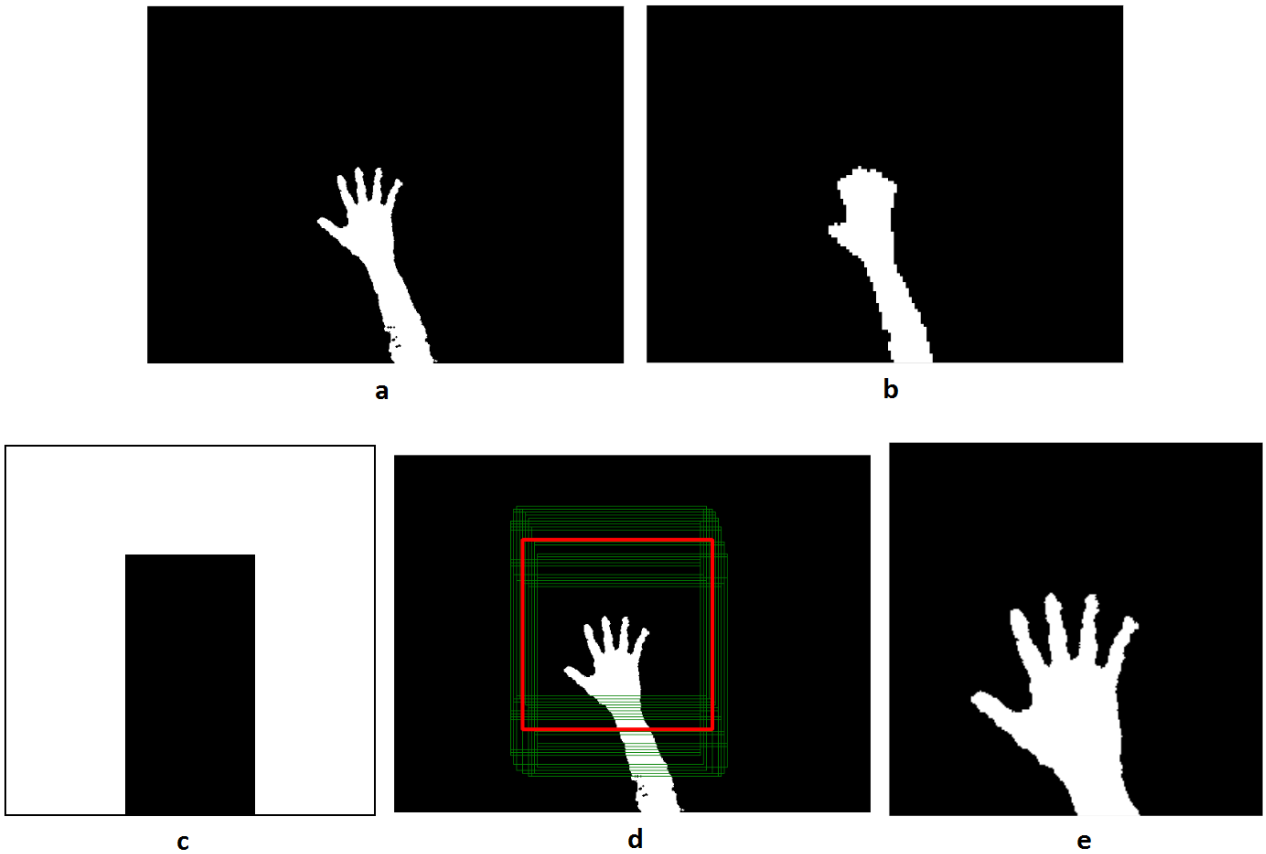
A módosított bináris képből integrál képet számolunk, amely egy alternatív képreprezentáció lesz. Ennek segítségével egy intenzitás kép (esetünkben ez egy bináris kép lesz) négyszög alakú képrészletében meg tudjuk mondani a képrész alatti pixelek összegét, mellyel lehetővé válik ezen objektumok észlelése. Az integrál kép x,y pozíciójában található érték tartalmazza a felette és tőle balra található pixelek intenzitás értékeinek összegét:

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y') \tag{1}$$

Erről a módszerről további részleteket Viola és Jones cikke [4] tartalmaz, ők az emberi arc szürkeárnyalatos képen való érzékelését valósították meg integrál kép használatával.

Mivel a kép egy részletében fog megjelenni a kézfej, ezért egy 64x64 méretű kereső ablakot definiáltunk, amelyet a teljes integrál képen végigmozgatunk. A keresőablakban az előre definiált Haar-elemmel számolunk (1. ábra (c) része), a fehér régió összegét kivonjuk a fekete régió összegéből, így egyetlen értéket kapunk keresőablakonként.

A képrészlet terület értékeinek megállapítása után osztályba sorolás következik, amely során feltételezzük, hogy a kapott tulajdonsággal biztonságosan meg tudjuk mondani egy képrészletről, hogy az tartalmazza-e az általunk keresett objektumot vagy sem. Erre a feladatra egy a gépi tanulás témakörében jól ismert osztályozó algoritmust használtunk, az AdaBoost-ot. Az AdaBoost lényege, hogy gyengén osztályozó tanuló algoritmusok (esetünkben döntési fák) kombinációjával tetszőleges pontosságú tanuló algoritmust kaphatunk. A tanítás során a gyengén osztályozókat több iteráción keresztül súlyozzuk, végül a betanított algoritmus a gyengén osztályozók súlyozott szavazata alapján fog dönteni. Tanuló algoritmusról lévén szó a tanításhoz és teszteléshez képekre volt szükség, megfelelő arányban kellett negatív és pozitív képeket előállítanunk. Az AdaBoost bemenetének a keresési ablakból számolt terület értékeket adjuk meg, kimenete pedig két osztályra fog korlátozódni: az ablakon belül megtalálható-e az emberi kéz vagy sem. Az osztályozónk tesztelése során arra figyeltünk fel, hogy egy kéz esetén a képen több kéz jelölt is megjelenhet (1. ábra (d) része). Erre a problémára a legjobb megoldás, ha vesszük a kéz jelölteket tartalmazó ablakok átlagát (1. ábra (d) része). A kísérleteink során azt tapasztaltuk, hogy a kéz mozgatása során is stabilan ugyanott jelölte az emberi kezet.



1. ábra. Kézfej érzékelésének egyes lépései; (a) Bőrszín szegmentációjának eredménye; (b) alulmintavételezés és morfológiai műveletek utáni kéz; (c) A definiált Haar-elem; (d) AdaBoost osztályozó algoritmus alapján meghatározott kéz jelöltek (zöld színnel) és a jelölteket tartalmazó ablakok átlaga (piros színnel); (e) A kézfejet tartalmazó képrészlet

3 Ujjhegyek detektálása

A kézfej sikeres szegmentációja után az ujjhegyek és a tenyérközéppont meghatározása algoritmusunk következő eleme. A feladat komplexitása miatt itt is több részre kellett bontani a teendőket, és egyúttal ügyelni kellett arra is, hogy a feldolgozási idő számottevően ne nőjön meg.

Első lépésként a kiválasztott képrészleten belül számoljuk az objektumhoz tartozó pixelek centroid pontját (x_c és y_c) az alábbiak alapján:

$$M_{00} = \sum_x \sum_y I(x,y),\ M_{10} = \sum_x \sum_y x \cdot I(x,y),\ M_{01} = \sum_x \sum_y y \cdot I(x,y) \tag{2}$$

$$x_c = \frac{M_{10}}{M_{00}},\ y_c = \frac{M_{01}}{M_{00}}, \tag{3}$$

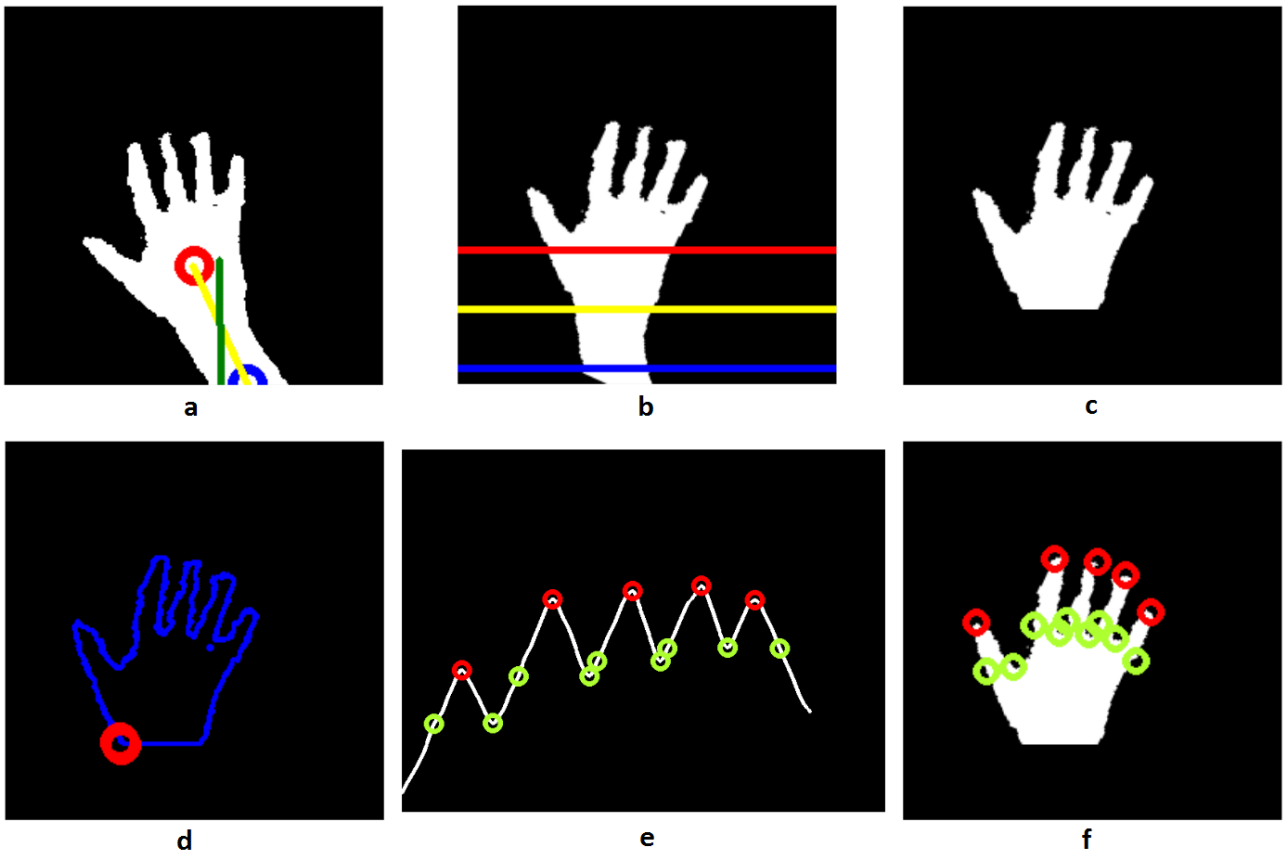
ahol *I(x,y)* az *I* bináris kép *x*,*y* koordinátáján megtalálható pixel értéke. Mivel a hasznos pixelek értéke 1, a háttér pixelei pedig 0 értéket vesznek fel, ezért a kéz centroid pontját fogjuk megkapni.

Feltételezzük, hogy az alkar egy része is látszódik a képrészlet alján, ezért vesszük annak legalsó sorban található hasznos pixelek középpontját. A két pont közé húzott egyenes alapján tudunk egy előzetes szöget számolni, mellyel a képet be tudjuk forgatni. Ezzel azt szeretnénk elérni, hogy a kézfelismerő eljárás során érzékelt elforgatott kézfejnek egy általunk meghatározott beforgatott állapotával tudjunk a későbbiekben dolgozni. Ennek az eredményét az 2. ábra (a) és (c) részén láthatjuk. Hozzá kell tenni, hogy ez a módszer nem minden esetben működik jól, a pontos beforgatáshoz szükség lenne a csukló középpontjára és mind az öt ujjhegyre, mindezek alapján már a kézfej pozícióját egészen pontosan meg lehetne becsülni, viszont ezek az adatok még nem állnak a rendelkezésünkre a felismerés jelen állapotában.

A következő lépésként számoljuk a beforgatott kép integrál képét. Erre azért van szükség, mert a kézfejet el szeretnék választani az alkartól, az integrál kép alapján pedig biztonságosan meg tudjuk majd mondani az alkart és a kézfejet elválasztó csukló vonalát. A csukló középpontját és dőlésszögét többféle módszerrel is meghatározhatjuk, a mi választásunk Chen és Fujiki megoldására [5] esett. Módszerük lényege az 3. ábra (a) részén látható. Feltételezik, hogy a csukló ott található, ahol a horizontálisan összegzett hasznos pixelek száma a minimális értéket veszi fel, a tenyérközéppont pedig a maximális helyen fog megjelenni. Ezt az adatsort könnyen előállíthatjuk az előbb számolt integrál értékek alapján. Természetesen a kapott adatsor a valóságban zajjal terhelt lesz az érzékelés hibái miatt, ezért az adatsor javításához Gauss-szűrőt használtunk:

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4}$$

A kísérletek során viszont azt vettük észre, hogy inkább a minimum és a maximum pont között található a valódi csuklóvonal, így a minimum pont helyett a két szélsőérték közötti értékkel dolgoztunk a továbbiakban (2. ábra (b) és (c) része). Érdemes még megemlíteni, hogy ez az eljárás akkor működött jól, ha legalább egy ujja látszódik a képen a felhasználónak, ökölbe szorított kéznél a tenyér egy részét is levágta.

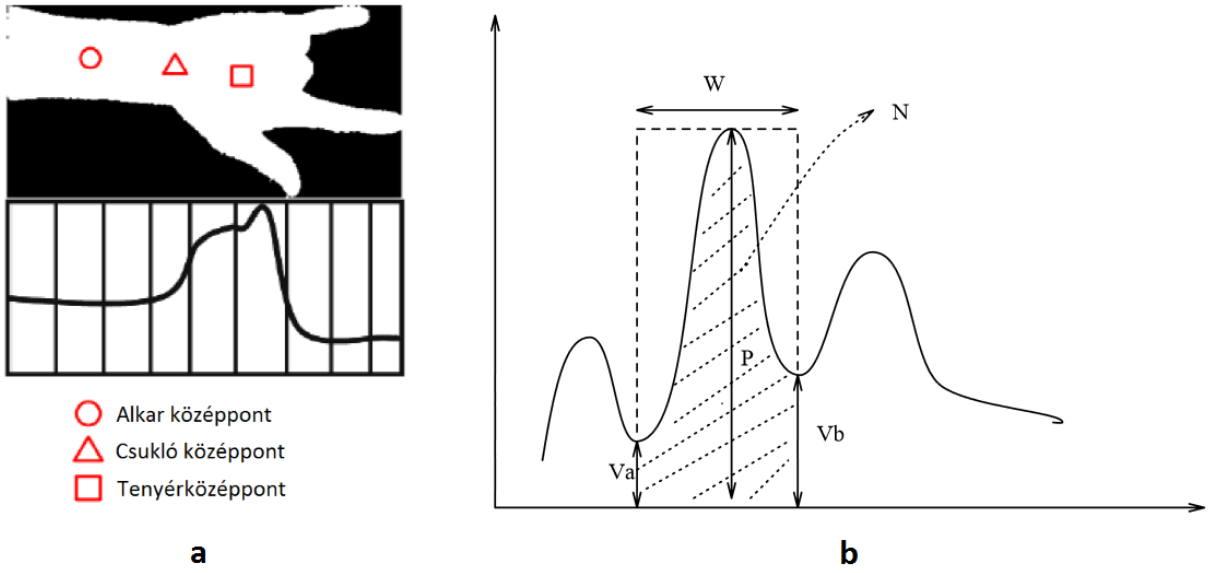


2. ábra. Ujjhegyek érzékelésének egyes lépései; (a) Kéz centroid pont pirossal, alkar pont kékkel, a kettő pont közötti egyenes sárgával, a forgatás utáni egyenes pedig zölddel jelölve; (b) Pontok alapján számolt szög alapján beforgatjuk a kézfejet, ezek alapján már meghatározható a csukló vonala (sárga színnel); (c) A csuklóvonal alapján levágott kézfej; (d) Kontúrdetektálás eredménye és a kontúrkezdőpont pirossal jelölve; (e) Radiális távolságok az új kezdőponttól, valamint az érzékelt ujjhegyek piros színnel; (f) A teljes ujjhegy felismerő eljárás eredménye

Miután a csuklóvonal alapján levágtuk az alkart az elforgatott képről (2. ábra (c) része), kontúrdetektálással segítjük elő az ujjvégek érzékelését. Mivel a kontúrpontok nem olyan sorrendben lesznek, amely nekünk a későbbi számításokhoz szükséges lesz, ezért először megkeressük melyik pont van a kontúrból legközelebb a csukló középpontjához. Továbbá azt feltételezzük, hogy a keresett pont *x* koordinátája kisebb legyen, mint a csuklóközéppont *x* koordinátája, ami azt fogja jelenteni, hogy a képen mindig a csuklóközéppont bal oldalán helyezkedik el ez a kontúrpont. A keresett kontúrpont alapján újrarendezzük a pontok listáját és számoljuk a minden egyes pont és az új kezdőpont közötti Euklideszi távolságot, így megkapjuk az alakzat radiális távolságait [6]. A radiális távolságot arra fogjuk felhasználni, hogy meghatározzuk az ujjhegyek pozícióját, melyek jól láthatóan a lokális maximumok lesznek a radiális távolságokat reprezentáló függvényen (2. ábra (e) része). Az ujjhegyek megkeresése előtt ebben az esetben is Gauss-szűrést alkalmaztunk, a csúcsok érzékeléséhez pedig a Peakiness tesztet [7] választottuk (3. ábra (b) része). A teszt használatával a függvényben található csúcsokat tudjuk egyetlen értékkel jellemezni a csúcsok szélessége, magassága és a függvény alatti terület alapján. Csúcsok jellemzése:

$$Peakiness = \left(1 - \frac{(V_a + V_b)}{2P}\right) \cdot \left(1 - \frac{N}{(W \cdot P)}\right) \tag{5}$$

A teszt megfelelő paraméterezése lehetővé tette az ujjak biztonságos érzékelését, segítségével még a 3 és 4 összezárt ujj állapotát is el tudunk egymástól különíteni. Sajnálatos módon, ha két ujj összezárva volt, a radiális távolságadatok alapján nem tudtuk egyértelműen megmondani, hogy egyetlen ujjat vagy két összezárt ujjat találtunk meg éppen, mivel a két állapot csak minimálisan különbözött egymástól. Ezen kívül a radiális adatok felhasználhatóak voltak a hüvelykujj jelenlétének meghatározásához is, a kontúr referenciaponthoz a hüvelykujj csúcsa mindig közelebb helyezkedett el (jobb kéz esetén), mint a többi ujj esetében.



3. ábra. (a) Chen és Fujiki megoldása [5] a csukló vonalának meghatározására; (b) Peakiness teszt [7] a jó csúcsok kiválasztásához; V_a és V_b jelöli a csúcshoz tartozó völgyek magasságát, W a jó csúcs szélessége, P a csúcs magassága, N pedig a függvény alatti terület

4 Kísérletek és eredmények

Algoritmusunk tesztelése során egyaránt használtunk előre felvett videókat – amelyek az eljárás végső változatának létrehozásában segítettek – illetve a Kinect által biztosított előképeket. Programunkat C# nyelven készítettük el és felhasználtuk az OpenCV [8] által biztosított képfeldolgozási függvénykönyvtárat.

A kézfej érzékeléséhez használt osztályozó algoritmusnál a tanító adatokat szintén mi állítottuk elő, összesen 1566 képrészletet és abból számolt terület értékeket készítettünk el. A sikeres betanítás érdekében fontos volt, hogy a negatív adatok túlsúlyban legyenek, ezért az összes kép 1524 negatív és 42 darab pozitív képre lett szétbontva. Természetesen a tanítás után tesztelésre is szükség volt, a teszteléshez használt adathalmazunk 229 elemből állt, melyből 199 volt a negatív és 30 a pozitív. A megfelelő tanító adatsor megtalálásával sikerült elérni, hogy a tesztelő adatsoron az osztályozás 97,82%-ban működött jól, 2,18%-át sorolta a negatív képeket a jók közé, valamint egyszer sem fordult elő, hogy rossznak ítélt volna olyan képeket, amin a kézfej látható.

Másik fontos szempont volt az algoritmus futási ideje, célunk az volt, hogy közel valós idejű feldolgozást érjük el. Ennek eredményeképpen mértük az egyes képfeldolgozási feladatok futási idejét, és sikerült az OpenCV-n alapuló megoldásunkat úgy optimalizálni, hogy a kéz lokalizációja átlagosan 48,78 ms-ot, a tenyér és ujjvégek átlagos felismerési ideje pedig 12,39 ms-ot vett igénybe.

5 Összefoglalás

Cikkünkben egy valós idejű kézfej és ujjhegy érzékelő algoritmust mutattunk be, megoldásunkat pedig két részfeladatra bontottuk szét. Elsőként a teljes képen meg kellett határozni magának a kéznek a pozícióját, későbbiekben a csukló alapján a kézfej és az ujjhegyek szegmentációját valósítottuk meg. Algoritmusunk működésének helyességét előre rögzített videók segítségével és az érzékelő által biztosított élőképen végzett kísérletekkel bizonyítottuk. Munkánkban fontos szerepet kapott az eljárás futási ideje is, átlagosan 60 ms szükséges egy rögzített kép feldolgozása.

Irodalom

[1] K. Mei, L. Xu, B. Li, B. Lin, F. Wang, "A real-time hand detection system based on multi-feature", Neurocomputing, Elsevier, Vol. 158, pp. 184–193, 2015.
[2] M. K. Bhuyan, K. F. MacDorman, M. K. Kar, D. R. Neog, B. C. Lovell, P. Gradde, " Hand pose recognition from monocular images by geometrical and texture analysis", Journal of Visual Languages and Computing, Elsevier, Vol. 28, pp. 39–55, 2015.
[3] J. A. M. Basilio, G. A. Torres, G. S. Pérez , L. K. T. Medina, H. M. P. Meane, "Explicit Image Detection using YCbCr Space Color Model as Skin Detection", Proceedings of the 2011 American conference on applied mathematics and the 5th WSEAS international conference on Computer engineering and applications, IEEE ICCE pp. 123-128, 2011.
[4] P. Viola, M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", Proceedings of the 2001 IEEE Computer Society Conference, IEEE, Vol. 1, pp. 511–512, 2001.
[5] W. Chen, R. Fujiki, D. Arita, R. Taniguchi, "Real-time 3D Hand Shape Estimation based on Image Feature Analysis and Inverse Kinematics", Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference, pp. 247-252, 2007.
[6] E. Yörük, E. Konukoglu, B. Sankur, "Shape-Based Hand Recognition", IEEE Transactions on Image Processing, Vol. 15, No. 7, 2006.
[7] M. Shah, "Fundamental of Computer Vision", University of Central Florida, 1997.
[8] OpenCV, http://opencv.org/, [online]