Az emberi fej térbeli helyzetének és a tekintet irányának meghatározása

Bertók Kornél, Fazekas Attila, Sajó Levente

Debreceni Egyetem
Informatikai Kar
Debreceni Képfeldolgozó Csoport
H-4010 Debrecen, Pf.:12.
bkornel@gmail.com
attila.fazekas@inf.unideb.hu
sajolevente@inf.unideb.hu

Absztrakt. A cikk témája egy multimodális ember-gép kommunikációt leíró elméleti modell egy moduljának gyakorlati megvalósítása. A modell az eddigi gyakorlatnál emberközelibb módon ragadja meg a számítógép és a felhasználó közötti interakciót. Amit úgy kíván elérni, hogy tanulmányozza az ember-ember kommunikáció olyan alapvető jellemzőit, amelyek technológiailag egyrészt relevánsak, másrészt pedig megvalósíthatóak. Az így szerzett elméleti tudásra támaszkodva egy új modellt állít fel az ember-gép interakció tetszőleges és változatos formáira. Ebbe a sorba illeszkedik be az emberi fej térbeli helyzetének, illetve azon belül a tekintet számítógépes meghatározása.

Kulcsszavak: fejpozíció számítás, POSIT, szivárványhártya detektálás, tekintet követés.

1. Bevezetés

1963-ban a Stanford Kutatóintézetben Douglas Engelbart felvetette egy új beviteli eszköz gondolatát. Mára ez az egérként elhíresült eszköz mindenki számára ismert és magától értetődően használható lett, de bevezetése óta nem igazán történt előrelépés a hétköznapi használatra szánt számítógépes perifériák piacán. Mára a technika fejlődésével megjelent az igény a számítógépek könnyű használhatóságára. Szükségessé vált, hogy az ember ne csak közvetve a kezével tudjon kommunikálni a számítógéppel. Újfajta ember-számítógép interfészek megjelenésére van szükség. Az ember-számítógép interakció kutatási feladatai közé tartozik, hogy olyan új, alternatív kommunikációs (adat ki- és beviteli) eszközöket és módszereket fejlesszen, amelyek segítik az ember-gép kapcsolatot az ember számára minél természetesebbé, magától értetődővé tenni. A mai luxusautókban a fedélzeti számítógépek már megértik a szóbeli utasításokat, sőt akár intelligens kivetítőkre is lehet ujjal rajzolni. Azonban az efféle interfészek nem csak szórakozásra használhatók, megkönnyíthetik a fogyatékkal élők életét is, mint például a szemkövető, vagy jelbeszéd-felismerő rendszerek.

Az előbbiek miatt a mára már kihagyhatatlan perifériának számító webkamera is egy fontos beviteli eszközként használható. Képnézegetők, játékok, web-böngészők irányítását lehet megkönnyíteni a kamera képéből kinyerhető mozgásinformáció felhasználásával. Nyugodtan kijelenthetjük, hogy ez a technika – vagyis a felhasználó webkamerával történő megfigyelése – egészen új lehetőségeket teremtett az ember-számítógép kommunikációban.

Ebben a cikkben az ember-számítógép kommunikáció egy újszerű felhasználói eszközével, a felhasználó, vagyis az emberi fél figyelmének számítógépes elemzésével foglalkozom. A rendszer webkamerák által szolgáltatott 800x600 vagy magasabb felbontású videó folyamokon határozza meg a fej térbeli pozíciójához tartozó forgatási mátrixot és eltolási vektort (3. Fejezet) Valamint a tekintet esetében a pupilla koordinátáit (4. Fejezet). Egy speciális eljárást dolgoztunk ki arra vonatkozóan, hogy a címben megjelölt feladatokat a lehető legtöbb képkockán, valós időben tudjuk megoldani. Mivel mind a fej térbeli helyzetére, mind a tekintet irányának meghatározására vonatkozó eljárásunk egy térbeli fejmodellen alapszik, így a térbeli alappontokhoz tartozó webkamera képeken megjelenő arci karakterisztikus pontokat nagy biztonsággal és gyorsasággal kell detektálnunk (2. Fejezet).

2. Arci jellemzők detektálása és követése

A rendszerünk sikeres működéséhez elengedhetetlen egy nagy megbízhatóságú, robosztus objektumdetektáló rendszer. A robosztusságon azt értjük, hogy a rendszernek érzéketlennek kell lennie a beérkező kép minőségére.

Az IPGD csoportban 2008. első felében az Intel OpenCV [1] szabadon felhasználható képfeldolgozó könyvtár segítségével elkészítettünk egy olyan szoftver eszközt, amely a gyakorlati kísérletek során bebizonyította, hogy megfelel az előbbi bekezdésben megfogalmazott elvárásainknak. Az alábbiakban ennek a rendszernek az elméleti hátterét tekintjük át, a hozzá kapcsolódó algoritmusokkal együtt. Ezzel a komponenssel határozzuk meg a fej térbeli helyzetének és a tekintet irányának megállapításához szükséges arci karakterisztikus pontok koordinátáit a bemeneti képen.

2.1. Viola-Jones detektorok

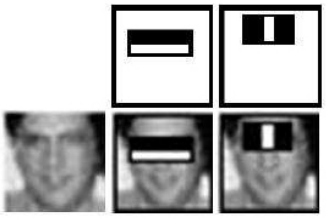
Paul Viola és Michael J. Jones egy olyan frontális arcdetektor rendszert alakítottak ki [2], mely eléri a megjelenése előtt és után publikált legjobb eredmények [3,4,5] találati és hamis pozitív arányát. Az arcdetektor mintájára szem-, száj-, orr-, szivárványhártya detektorokat készítettünk az OpenCV könyvtár segítségével, melyben Viola és Jones közzétették arcdetektorukat és az implementált tanítási algoritmust.

Arcdetektáló rendszerük gyorsasága tisztán kiemelkedik az eddigi megközelítések közül. Valós időben 30 kép/másodperc teljesítményt érhetünk el, 640x480 pixel felbontású kameraképeken egy 1.6 GHz-es Intel Pentium M processzorral rendelkező notebook segítségével. A Viola-Jones detektorok csak szürkeskálás képekből kapott

információk alapján képesek magas feldolgozási arányt elérni. A detektorok felépítését az alább kifejtett három fő témakör köré csoportosíthatjuk.

2.1.1. Az integrált képreprezentáció bevezetése.

A rendszer a kép intenzitásértékei helyett képrégiók jellemzőivel dolgozik [2]. Ezek a jellemzők a Haar-féle bázisfüggvényekre emlékeztetnek, bár néhány esetben valamivel bonyolultabbak azoknál. A jellemzők többféle méret melletti gyors kiszámítása érdekében vezetjük be az integrál képet, amely az eredeti képből képpontonként néhány elemi művelettel – tehát rendkívül gyorsan és hatékonyan – előállítható. Az integrált képreprezentáció ismeretében a jellemzők értéke (lásd 1. ábra) bármely pozícióban vagy skálázás mellett konstans időben határozható meg.



1. ábra: Az AdaBoost által választott első és második jellemző (az arcdetektor tanítása során). Értékük úgy számolódik, hogy a fehér téglalapon belüli pixelek összegét kivonjuk a feketén belüliek összegéből.

2.1.2. AdaBoost algoritmus

A második fontos tulajdonság egy osztályozó létrehozásának folyamata, a fontos jellemzők egy kis halmazának kiválasztásával. Erre egy AdaBooston alapuló tanuló algoritmust használtak. Egy kép részablakán belül az összes Haar jellemzők száma nagyon nagy, sokkal nagyobb, mint a pixelek száma. Azért, hogy a gyors osztályozást biztosítsuk, a tanuló folyamatnak ki kell zárnia az elérhető jellemzők nagy többségét, és a kritikus jellemzők kis halmazára kell koncentrálnia. Tieu és Viola munkája által motiváltan, a jellemző kiválasztás az AdaBoost egy egyszerű módosítása lett: minden gyenge osztályozót kényszerítenek, hogy annak eredménye csak egy jellemzőtől függjön. Ennek eredményeként a Boost folyamat minden köre, amely új gyenge osztályozót választ, megfelel egy jellemző kiválasztó folyamatnak [6].

Bővebben kifejtve, ha adott a jellemzőknek, továbbá pozitív és negatív tanítópéldáknak egy-egy halmaza, akkor a gépi tanulás bármely megközelítése felhasználható egy osztályozó függvény készítéséhez. Feltételezve, hogy a detektor alap felbontása 24×24-es ablak, azt kapjuk, hogy a téglalap jellemzők halmazának számossága az ablakban igen nagy: 45.396 darab. Ez a szám sokkal nagyobb, mint a képpontok száma. Habár minden jellemző hatékonyan számítható, a jellemzők teljes halmazának számítása megengedhetetlen. Ezeknek a jellemzőknek már igen kicsi halmazával is létrehozható hatékony osztályozó. Az igazi kihívás ezeknek az jellemzőknek a megtalálása.

Eredeti formájában az AdaBoost gyenge osztályozó függvények kombinálásával hoz létre egy erős osztályozót, ezáltal tetszőleges tanuló algoritmus javítására is használható. A tanuló algoritmust úgy tervezték meg, hogy azt az egyszerű téglalap jellemzőt válassza ki, amely a legjobban választja szét a pozitív és negatív példákat. Minden jellemzőhöz a gyenge tanuló meghatároz egy optimális küszöbosztályozó függvényt, ami a példák minimális számú halmazát osztályozza rosszul. Így egy $h_j(x)$ gyenge osztályozó egy f_j jellemzőből, egy θ_j küszöbből és egy p_j paritásból áll, ahol a paritás az egyenlőtlenség jel irányát jelöli:

$$h_j(x) = \begin{cases} 1, \text{ha } p_j f_j(x) < p_j \theta_j, \\ 0, \text{különben}, \end{cases} \quad (1)$$

ahol x egy 24×24 pixel felbontású képablak.

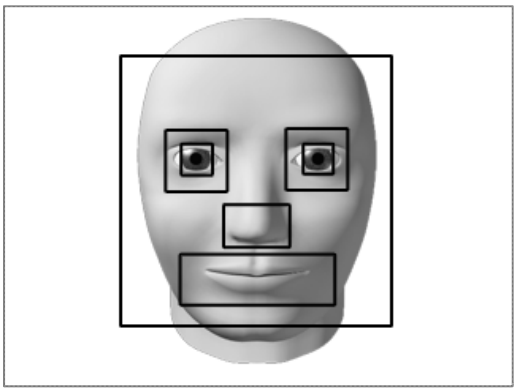
2.1.3. Kaszkád struktúra

Az utolsó lépés, hogy az egyre bonyolultabb osztályozókat láncba, vagyis egy vízesésszerű sémába rendezzük. Az ötlet azon az észrevételen alapul, hogy gyakran egyszerű azt eldönteni, hogy egy objektum hol fordulhat elő, így a bonyolultabb feldolgozást csak ezeken a „bíztató" területeken kell elvégezni. Tehát könnyű olyan gyenge osztályozókat készíteni, amelyek a negatív példák nagy részét visszautasítják, viszont a pozitívokat elfogadják. Így egyszerűbb osztályozókat használunk a negatív példák elutasítására, és az összetettebb osztályozókat már csak a biztató területeken futtatjuk le, hogy minél alacsonyabb hamis pozitív arányt érhessünk el.

A végső osztályozó egy elfajuló döntési fa, amelyet „vízesésnek" nevezünk [7]. Az első osztályozó pozitív válasza elindítja a második osztályozót, amely szintén nagyon magas detektálási aránnyal rendelkezik. Ha ez pozitív választ ad, elindul a harmadik osztályozó, és így tovább. Ha bármely pontnál a válasz nemleges, az ablakot azonnal visszautasítjuk.

2.2. Detektorok hierarchiája a programon belül

Az arc, orr, száj, balszem, jobbszem és szivárványhártya detektorok csak a kép egyes területein futnak csak le (lásd 2. ábra). A detektorok téglalapok sorozatával térnek vissza, melyek a keresett objektumot tartalmazhatják. Míg az arcdetektor a teljes képkockán lefut, addig a szemdetektor csak az arcdetektor felső felének eredményét dolgozza fel (amennyiben az arcdetektor talált eredményt). Értelemszerűen a balszem detektor csak az arc felső felének bal oldalán fut le, a jobbszem detektor pedig csak a jobb oldalán. Ennek megfelelően az orrdetektor csak az arcdetektor által szolgáltatott ablak közepén fog lefutni, a szájdetektor pedig csak az ablak alsó felében fog keresni. Ha az arcdetektor nem talált eredményt, akkor egyetlen másik detektor sem fog lefutni az aktuális képkockán.



2. ábra: Objektumdetektálás egy 1024×768 pixel felbontású képen. Az egyes detektorok értelmezési tartományát befolyásolhatja a többi detektor által visszaadott ablak.

A szemdetektor eredménylistájából már nem a legutolsó és egyben legnagyobb eredményét választom ki a további szivárványhártya kereséshez, hanem azok közül az eredmények közül választom ki a legnagyobbat, melyekre teljesülnek az alábbi feltételek:

- A szemdetektor által talált téglalap szélessége legyen nagyobb az arcdetektor eredmény szélességének 30%-ánál.
- A szemdetektor által talált téglalap az arcdetektor eredményének felső felében legyen, mivel egy emberi arcon a szemek az arc felső felében helyezkednek el.

Ha van eredménye a szemdetektoroknak, akkor lefuttatom a szivárványhártya detektorokat azon belül. Ha az egyik oldali szemdetektor nem talál eredményt, akkor a képrészletet vertikálisan tükrözzük, és a másik oldali szemdetektort futtatjuk rajta, majd ezek után próbálunk szivárványhártyát detektálni.

A szemkövető program teljes detektorhierarchiáját minden beérkező képen futtatva 15 kép/másodperc teljesítményt érhetünk el, egy 2.8 GHz-es Intel Pentium Core 2 Quad processzorral rendelkező PC-n, egy 1024×768 pixel felbontású webkamera képen. A program megfelelő futásához azonban elég, ha csak minden ötödik képen futtatjuk a program teljes detektor hierarchiáját, így látható, hogy a valós idejű teljesítményt gond nélkül elérhetjük akár jóval kisebb teljesítményű számítógépeken is.

2.3. Az arci jellemzők követése

Az előző fejezetben megadtunk egy módszert, mely segítségével magas detektálási arány és alacsony számítási idő mellett, lehetőségünk nyílik az arc és az egyes arci jellemzők detektálására. A fej térbeli helyzetének, illetve a tekintet irányának meghatározásához viszont minden egyes képkockákon tudnunk kell a szemek, a száj, az orr és a szivárványhártyák helyzetét. Könnyen belátható, hogy az előbbi pontok együttes detektálásának a valószínűsége alacsony. Így ha egyszer már detektáltunk egy pontot, akkor annak meg kell jósolnunk a helyzetét azokon a képkockákon, amelyeken a detektálás sikertelen volt.

Tehát vizsgálnunk kell a vizuálisan reprezentált tárgyak elmozdulását, amit általában képpontokból eredő vektorokkal írunk le. Ebben a fejezetben a mozgást és azon belül a mozgás irányát próbáljuk meghatározni. Az alábbiakban látni fogjuk, hogyan lehet a feladatot hatékonyan és gyorsan megoldani. A mozgásérzékelés megvalósítására több módszer is létezik, például a fázis korreláció [8], mely azon alapul, hogy az eredeti képen az eltolás a frekvenciatérben fáziseltolódásként jelenik meg. Egy másik algoritmus az illesztő algoritmus [9], amely egy szabályosan kiválasztott képrészletekre alkalmazott egyszerű mintaillesztő algoritmus. Azonban mi a Lucas-Kanade módszernek egy módosított változatát választottuk, a Kanade-Lucas-Tomasi (továbbiakban csak KLT) követő algoritmust. A Lucas-Kanade módszer viszonylag régóta ismert. Azon alapszik, hogy érdemes csak néhány előre meghatározott jellemző pontoknak kiszámítani a mozgását [10], így megvalósítása egyrészt gyengébb hardveren is lehetséges, másrészt a mi modellalapú rendszerünkbe is jól illeszkedik.

2.4. KLT algoritmus

Az algoritmus leginkább abban különbözik a Lucas-Kanade módszertől, hogy egy új szimmetrikus számítási módszert vezettek be a mozgás számítására az egymást követő képekre. Ennek köszönhetően az algoritmus használhatóvá vált valós időben is. A KLT algoritmus a Viola-Jones detektorokhoz hasonlóan csak szürkeárnyalatos képekre alkalmazható. Eredeti célunk az volt, hogy a mozgásokat minden egyes képkockán észleljük. Ehhez meg kell találnunk az egyes tárgyak elmozdulását az előző képkockához képest. A megoldáshoz a mintaillesztés algoritmusát alkalmazzuk. Ehhez minden egyes képkockán ki kell választanunk egy négyzet alakú területet, majd a rákövetkező képkockán meg kell találnunk azt a területet, amely a leginkább hasonlít az eredetihez. Adott tehát az egyik képen egy minta, amit meg kell találni a rákövetkező képkockán. Amennyiben elég sűrűn mintavételezünk, akkor feltételezhetjük, hogy:

- Az adott minta a következő képkockán nem mozdul el jelentősen, tehát a pozíciója közel marad az eredeti pozícióhoz.
- A minta csak kicsit változik meg a két kép között, azaz nem távolodik–közeledik a kamerához, továbbá az elfordulás és torzulás hatása is elhanyagolható.

Van tehát egy mintánk, és a rákövetkező képen keressük azt a pozíciót, ahol a hasonló méretű és alakú minta a leginkább hasonlít az eredeti mintához. A mintákat több módon is össze lehet hasonlítani. Az egyik lehetséges módszer, hogy a képből és annak gradienséből következtetünk a minta optimális helyére. Ebben az esetben tulajdonképpen egy Taylor-soros közelítés alapján számítjuk ki a minta elmozdulását [11].

A KLT algoritmus tehát úgy számolja ki az optimális helyet, hogy a megfelelő eltolás vektort alkalmazva az eredeti mintát odébb teszi (vagyis iterálja az eljárást), és arról a pozícióról próbálkozik újra, egészen addig, ameddig az elmozdulás kisebb lesz egy előre rögzített nagyon kicsi határértéknél. A gradiens alapú összehasonlítás előnye, hogy viszonylag gyors, hátránya, hogy tévedhet, mivel nincsen garancia arra, hogy az elsőfokú Taylor-soros egyszerűsítés során nem veszik el olyan információ,

amely félreviszi az eredményt. A gradiens alapú módszernek további előnye, hogy torzításokat is (pl. a perspektíva miatti deformációkat) is be lehet vinni a számítási idő növekedése nélkül [12].

3. Fej térbeli helyzetének meghatározása

Az ember-gép kommunikációban az emberi fej térbeli helyzetének ismerete fontos képesség a számítógép számára, mivel a gép ezáltal képet kaphat a kommunikációs partnere figyelmének középpontjáról, kommunikációban való részvételének hajlandóságáról, illetve annak egyéb belső állapotáról, viselkedésmódjáról. Az elmúlt években több eljárást is kifejlesztettek a probléma megoldására. A módszerek alapulhatnak az arc külső megjelenésén, vagy a karakterisztikus pontjainak egy modelljén.

A megjelenés alapú modellek az arc egészét használják fel a fejpozíció számításához, ez az eset tehát felfogható egyfajta osztályozási problémaként is. A hasonló helyzetű fejpozíciókat halmazokba rendezik, majd minden egyes halmazra készítenek egy osztályozót. Ezen eljárások között találjuk Schneiderman és Kanade [13] statisztikai módszerét, vagy a Meynet [14] által bevezetett döntési fa szerkezetű osztályozókat, melyekkel hierarchikusan mintavételezhető a fejpozíciók tere. A megjelenés alapú modellek nagy problémája, hogy a fejpozíciók rögzített számú halmazba történő besorolása nem teszi lehetővé az olyan kifinomult gesztusok modellezését, mint pl. a fejrázás, vagy a bólintás.

A modellalapú fejpozíció számítások a fej geometriai modelljén alapulnak. A geometriai modell elkészíthető pl. az egyes arci karakterisztikus pontok koordinátái alapján, de ettől eltérő megvalósítások is léteznek. Pl. a Dornaika és Ahlberg [15] által közzétett módszerben a fej geometriáját az arci jellemzők kontúrjának leírásával adják meg. A modell alapú eljárások hátránya, hogy számításigényesebbek és hogy több közülük kézi inicializációt igényel.

Koordináta	X	Y	Z
Orr	0	0	0
Bal szem	-15	15	-10
Jobb szem	15	15	-10
Száj	0	-15	-10

1. táblázat: A programban használt fejmodell koordinátái. A koordináták csak az arányosság kifejezésére szolgálnak.

A fej térbeli helyzetének meghatározására egy olyan modell alapú eljárást dolgoztunk ki, mely számításigényét tekintve használható valós idejű alkalmazásként is, valamint nélkülöz mindenféle kézi beavatkozást. A kapcsolatot a kétdimenziós kamera sík és a háromdimenziós fejpozíció tér között a következő fejezetben ismertetett POSIT eljárás adja meg. Az eljáráshoz szükségünk van egyrészt a fej térbeli modelljét alkotó alappontokra (lásd 1. táblázat), ahol a koordináták rögzítése a fej geometriájának megadására szolgál, így nem korlátozza az eredményt. Továbbá szükségünk van az

ezekhez tartozó képsíkbeli – azaz webkamera képbeli – pontok koordinátáira, valamint a kamera néhány paraméterére (felbontás, fókusztávolság).

3.1. Modell alapú pozíciódetektálás

Rögzített számú jellemző ponttal megadott tárgyak térbeli helyzetének vagy irányának ismerete fontos lehet egyes kalibrációs, térképészeti, vagy tárgykövető alkalmazás esetében. A pozíciódetektálás során a háromdimenziós euklideszi tér azon pontjait szeretnénk megadni, melyek csupán a kétdimenziós képsíkon ismertek számunkra. A problémát az alábbi módon formalizálhatjuk:

$$\text{Legyen adva } \mathrm{P} = \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \end{bmatrix}, \tag{2}$$

a vizsgált objektumnak egy tetszőleges pontja a saját koordinátarendszerében.

$$\text{Legyen továbbá } \mathrm{p} = \begin{bmatrix} \mathrm{x}' \\ \mathrm{y}' \\ \mathrm{z}' \end{bmatrix}, \tag{3}$$

melyet úgy kaptunk meg, hogy P-t vetítettük a képsíkra. Természetesen a mélységinformáció – a z koordináta – a vetítés során eltűnik, így a normalizálás eredményeként kapott pontot célszerű átírni az alábbi formába:

$$\mathrm{p} = \begin{bmatrix} \mathrm{x}' \\ \mathrm{y}' \end{bmatrix}. \tag{4}$$

Értelemszerűen a térbeli és a képsíkbeli pontok függnek egymástól, tehát ha a modell egy pontja elmozdul, akkor ez változást fog előidézni a vetítésben is. A térbeli és a vetített pontok között az alábbi módon adhatjuk meg a kapcsolatot:

$$\mathrm{p} = \mathrm{RP} + \mathrm{t}, \tag{5}$$

ahol R a forgatási-, és t az eltolási vektor. Tehát egy olyan eljárást szeretnénk definiálni, mely képes kiszámítani bizonyos alappontokkal megadott, webkamera képeken elhelyezkedő síkbeli objektumok térbeli helyzetét, vagyis az R és a t vektort.

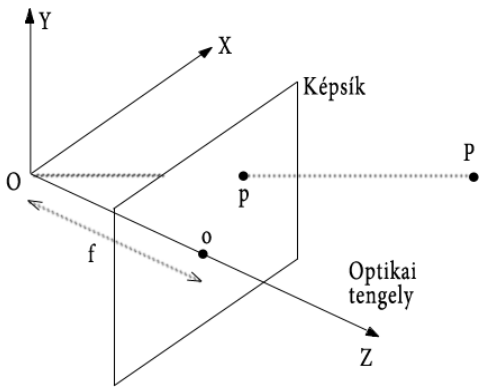
Több eljárás is létezik erre vonatkozólag, mi a lentebb ismertetett POSIT eljárást [16,17] használtuk fel, mely 25 sorban implementálható, így az előzetes feltételezéseink szerint hatékonyan alkalmazható valós időben is. A továbbiakban feltételezzük, hogy ismerjük a vizsgált térbeli objektumnak a geometriáját, amit véges számú térbeli alapponttal adunk meg. Továbbá feltesszük azt is, hogy a webkamera képeken ismerjük négy vagy több olyan pontnak a koordinátáját, melyek a térbeli objektumon nem esnek egy síkba (esetünkben az 1. táblázat pontjai). Fontos azt is leszögezni, hogy az eljárás iteratív módon számolja ki térbeli pozíciót és csak a lineáris algebra eszközeiből építkezik. A POSIT működéséhez –a Newton-Raphson és

az arra építkező eljárásokkal ellentétben [18] – nem szükséges a kezdőpozíció rögzítése és a mátrixok invertálása az iteráció törzsében. A számításokhoz egy webkamera áll a rendelkezésünkre, tehát nem használhatjuk ki a sztereó látás előnyeit sem a mélységérzékelés során.

3.2. Vetítési modell

A fej térbeli helyzetének kétdimenziós pontok alapján történő meghatározásához szükség van egy vetítési modellre. A gyakorlatban általában a perspektivikus, vagy gyenge perspektivikus vetítési modelleket szokták alkalmazni. Mint minden vetítés az előbb felsorolt kettő is egy dimenziócsökkentő művelet, tehát olyan transzformáció, amely n-dimenziós objektumokat kisebb dimenziós terekbe visz át. A vetítés eredménye a vetület, ami egy térbeli síkon, a vetítési síkon képződik. A tárgy-, és képpontokon átmenő egyenest vetítősugárnak nevezzük. Az egyes tárgypontok képe pedig a vetítősugár döféspontja a vetítési síkkal.

Amikor perspektív vetítésről beszélünk, akkor a vetítősugarak mindegyike áthalad egy vetítési középponton, a centrumponton. A perspektív vetítés az objektumok realisztikus ábrázolását teszi lehetővé, ilyenkor a távolabbi objektumok kisebbnek tűnnek, a vetítési síkkal nem párhuzamos egyenesek pedig egy pont felé tartanak. Perspektív vetítés esetén a látótér tulajdonképpen egy végtelen piramisként képzelhető el, amelynek csúcsa a nézőpont, vagy szem. A perspektivikus vetítés modellje egy képsíkból, valamint a vetítés középpontjából – az O pontból – áll (lásd 3. ábra). A vetítés középpontja és a képsík közötti távolság f, amit gyújtótávolságnak is szokás hívni. Az O-ból kiinduló és a képsíkot metsző egyenes az optikai tengely.



3. ábra: A perspektivikus vetítés térbeli modellje.

A perspektív vetítés legnagyobb problémája, hogy nem-lineáris transzformáció, így valós idejű használata nem kifejezetten javallott. Célszerű tehát közelíteni egy lineáris transzformációval, például a merőleges vetítéssel, ami a perspektíva határértékének is tekinthető az $f \to \infty$ miatt. Az approximációra a gyenge perspektivikus vetítés modelljét használjuk fel, amely tulajdonképpen egy merőleges vetítésből és egy

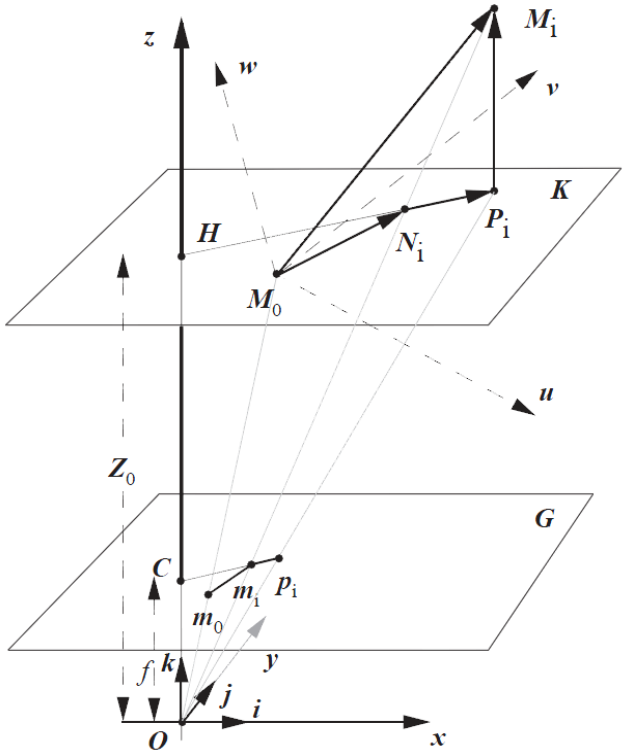
skálázásból áll. Továbbá csak akkor hajtható végre, ha fennáll az alábbi két előfeltétel:

- A vetíteni kívánt tárgy közel van az optikai tengelyhez.
- A tárgy mérete vagy kiterjedése kisebb, mint az átlagos távolsága a kamerától.

A gyenge perspektivikus vetítést értelmezhetjük úgy is, hogy a transzformáció során minden pontot ugyanazzal a faktorral skálázunk.

3.3. POS algoritmus

A pozíció kiszámítására vonatkozó eljárásunk két metódus kombinációjaként jön létre. Az első a POS (Pose from Orthography and Scaling) mely az előbb ismertetett gyenge perspektivikus vetítésen alapul.



4. ábra: A kameramodell. M_i a fejmodell egy pontja, m_i a neki megfelelő perspektivikusan vetített pont a G képsíkon. p_i pedig a gyenge perspektivikus vetítése M_i-nek.

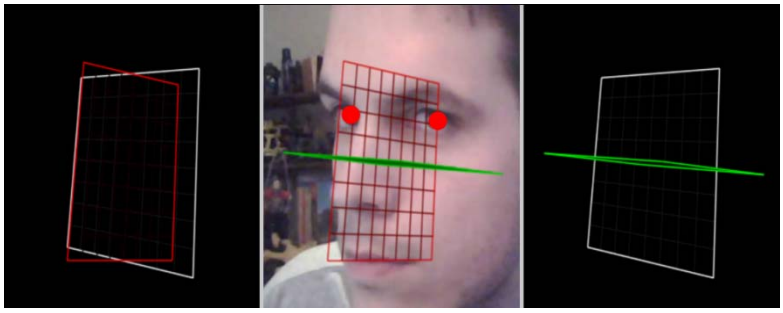
Konkretizálva a problémát, a 4. ábrán egy klasszikus kameramodell látható, melyen a vetítés középpontja O, a képsíkot pedig G-vel jelöltük. A képsík két tengelyét O_x-et és O_y-t tekinthetjük a kameraszenzornak, O_z pedig az optikai tengelynek feleltethető meg. Továbbá azt is tudjuk, hogy a képsík f távolságra van O-tól. A tárgy-

koordinátarendszer középpontja M_0, a tengelyei pedig rendre M_{0u}, M_{0v} és M_{0w}. Mivel ismerjük a fej térbeli geometriáját, így az összes M_i pont tárgy-koordinátarendszerbeli koordinátája ismert. A problémát az jelenti, hogy a fej térbeli pozíciója egyelőre még nem ismert, így az M_i pontok kamera-koordinátarendszerbeli koordinátái sem ismertek számunkra. Jelöljük ezeket a koordinátákat (X_i, Y_i, Z_i)-vel. E koordinátarendszerbeli koordináták kiszámításában lesz segítségünkre a POSIT algoritmus. Az ábrán csak az M_0 és M_i pontokat szerepeltettük.

A forgatási mátrix kiszámításához a képsíkhoz tartozó i és j egységvektorokat kell felírni a tárgy-koordinátarendszerben. Az eltolási vektort pedig a tárgy-koordinátarendszerben felírt i és j vektorok keresztszorzataként kaphatjuk meg [17]. Tehát a POS algoritmus egy skálázott merőleges vetítéssel közelíti a gyenge perspektivikus vetítést. Fogalmazhatunk úgy is, hogy egy lineáris egyenletrendszer megoldásával számítja ki a forgatási mátrixot és az eltolási vektort.

3.4. POSIT algoritmus

A POS algoritmus során tulajdonképpen az m_i képsíkbeli pontokból kaptunk egy közelítést a tárgy pozíciójára vonatkozóan. Jelöljük a közelítő fejpozícióhoz tartozó térbeli pontokat M_i^*-gal. Toljuk el ezeket az M_i^* pontokat abba a pozícióba, amit egy helyes számítás eredményeként kaptunk volna. Majd a M_i^* pontokhoz tartozó m_i^* képsíkbeli pontokra futtassuk le újból a POS eljárást [17]. A POSIT eljárást négy-öt iteráción át futtatva megkapjuk a helyes pozíciót. Az algoritmus eredménye az 5. ábrán látható.



5. ábra: A fejpozíció, illetve a szivárványhártya detektálás kimenete. Az ábra két szélén láthatók a fej térbeli helyzetét szemléltető síkok. Ezeknek a kombinációja látható a középső ábrán, amelyen a piros körök a szivárványhártya detektorok eredményeit jelentik.

4. Tekintetkövetés

Az ember-gép kommunikációban a figyelem középpontjának vizsgálata során fontosabb az emberi fél nézési irányának meghatározása, mint a fej térbeli helyzetének ismerete. Ennek az a magyarázata, hogy a fejünk és a szemünk egymástól

függetlenül mozoghat, így a fej helyzetéből csak akkor következtethetünk a figyelem középpontjára, ha a felhasználó szeme nem látható a webkamera képeken. Tehát a figyelem középpontjának meghatározásához szükségünk van egy algoritmusra, mellyel számítógépesen követni tudjuk az emberi tekintetet.

Az elmúlt évek alatt folyamatosan fejlődött a szem és tekintet követésének technológiája. Először csupán arra használták, hogy megtudhassuk, milyen gyorsan és hová néz az ember a képernyőn. A legújabb fejlesztések és törekvések –ahogy a mi fejlesztéseink is – inkább a szemmel történő irányításra koncentrálnak. Mindazonáltal megjegyezzük, hogy a tekintetfigyelés technológiája nem új, hiszen évek óta foglalkoznak ezzel a fejlesztők. Kezdetben hatalmas, kamerákkal felszerelt sisakokat kellett hordaniuk a vállalkozó szellemű tesztelőknek, azonban a technológia az idők folyamán letisztult. A 90-es évektől kezdve jelentek meg azok az eszközök, melyek teljesen lebénult, csupán a szemüket mozgatni képes embereknek segítettek a külvilággal kommunikálni, használva a szemmozgást figyelő szenzorokat és a képernyőn megjelenített virtuális billentyűzetet. A legújabb törekvések azonban ezen is túlmutatnak, a hétköznapi használatba szeretnék bevinni a technológiát, magyarán a szemkövetést a lehető legegyszerűbb és a legszélesebb körben elérhető eszközökkel szeretnék megvalósítani (pl. személyi számítógép és webkamera).

A szemkövetés megvalósításához, első lépésben a szivárványhártya középpontját próbáljuk megtalálni. Erre vonatkozólag több eljárást is kipróbáltunk. Próbálkoztunk pl. Hough transzformáció segítségével köröket, valamint körszeleteket detektálni a fejpozíció kiszámításánál használt szemdetektorok eredményein, de csak akkor jártunk sikerrel, amikor a webkamera előtt ülő felhasználó meredten nyitott szemekkel bámult a kamerába. Ezután próbálkoztunk a pupillának, mint relatíve sötét foltnak és a szemfehérjének, mint relatíve világos foltnak a detektálásával [19], de ezzel a módszerrel sem jártunk sikerrel, mivel a szemöldökök, a szempillák, az erős smink és a világos bőr bezavart a számításokba. Ezt követően megpróbáltunk egy infra kamerával és infra LED-eknek egy csoportjával vörös szem effektust létrehozni a kameraképeken [20,21], de egy speciális szinkronizációs hardver hiányában ezt a módszert sem tudtuk a gyakorlatban megfelelő hatékonysággal alkalmazni.

Vizsgálataink során rájöttünk, hogy az előbbi módszerek főként a szivárványhártya alacsony felbontásának köszönhetően vallottak kudarcot a gyakorlatban. Így egy olyan eljárást kifejlesztésére fektettük a hangsúlyt, mely képes lehet alacsony felbontású webkamera képeken is szivárványhártyát detektálni. A detektálás során a pupilla, szivárványhártya, szemfehérje és szemhéj részek geometriáját és mintázatát szerettük volna felhasználni, így készítettünk egy Viola-Jones detektort az előbb felsorolt objektumokra vonatkozólag. Az így kapott detektorokat a bal-, és jobbszem detektorok eredményeire futtatva, már kellően nagy pontossággal (lásd 5. ábra) tudjuk lokalizálni a szivárványhártyát, valamint annak középpontját, a pupillát.

5. Összefoglalás

A cikkben röviden összefoglaltuk egy, az ember-gép kommunikációban az emberi fél figyelmét elemző rendszernek egy lehetséges megvalósítását. A rendszer már működőképes, de a részletes tesztelési szakasz még nem érkezett el, így a komolyabb összehasonlítások csak később fognak elérkezni. Azonban előjáróban néhány szót

elárulhatunk a várható eredményekről és teljesítményről. A fej térbeli helyzetét meghatározó modul teljesítménye csak az egyes arci jellemzőkre betanított Viola-Jones detektoroktól függ, melyek hatásfokát már leírtuk a 2.2-es fejezetben. Az arci jellemzők helyes pozíciójának ismeretében a POSIT eljárás minden esetben jó eredményt fog szolgáltatni, ha a felhasználó 20−80cm-re ül a webkamera előtt, ennél távolabb azonban már a detektorok sem adnak vissza pozitív találatot. Az előbbi távolságadatok természetesen függnek a kamera optikájának látószögétől és fókusztávolságától. A tesztjeink során egy Logitech® Webcam Pro 9000 típusú webkamerát használtunk fel, 1024×768-as felbontáson. A szivárványhártya detektorokra vonatkozóan még nem tudunk tesztadatokat szolgáltatni, de már most tisztán látszik, hogy nagyságrendekkel jobb eredményt szolgáltat, mint az általunk kipróbált összes eddigi eljárás.

Irodalom

1. Intel® IPP. (2010) Open Source Computer Vision Library. [Online]. http://sourceforge.net/projects/opencvlibrary/
2. R. Lienhart and J. Maydt, "An Extended Set of Haar-like Features for Rapid Object Detection," *IEEE International Conference on Image Processing*, pp. 900-903, 2002.
3. S. Phimoltares, C. Lursinsap, and K. Chamnongthai, "Face detection and facial feature localization without considering the appearance of image context," *Image and Vision Computing*, vol. 25, pp. 741-753, 2007.
4. L. Kyoung-Mi, "Component-based face detection and verification," *Pattern Recognition Letters*, no. 29, pp. 200-214, 2008.
5. H. Lin-Lin and S. Akinobu, "A multi-expert approach for robust face detection," *Pattern Recognition*, vol. 39, pp. 1695-1703, 2006.
6. K. Tieu and P. Viola, "Boosting Image Retrieval," *International Journal of Computer Vision*, vol. 56, pp. 17-36, 2004.
7. Y. Amit, D. Geman, and K. Wilder, "Joint induction of shape features and tree classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 11, pp. 1300-1305, 1997.
8. E. D. Castro and C. Morandi, "Registration of translated and rotated images using finite Fourier transforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, 1987.
9. G. Aglika, K. Chandrika, and S. C. Cheung, "BlockMatching for Object Tracking," Lawrence Livermore Nation Laboratory, Technical Report, 2003.
10. T. Carlo and K. Takeo, "Detection and tracking of point features," Carnegie Mellon University, Pittsburgh, Technical Report, 1991.

11. L. D. Bruce and K. Takeo, "An iterative image registration technique with an application to stereo vision," *International Joint Conference on Artificial Intelligence*, p. 674–679, 1981.
12. S. Baker and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221-255, 2004.
13. H. Schneiderman and T. Kanade, "Object Detection Using the Statistics of Parts," *International Journal of Computer Vision*, vol. 56, p. 151–177, 2004.
14. J. Meynet, T. n. Arsan, J. C. Mota, and J. P. Thiran, "Fast multiview tracking with pose estimation," TR-ITS.2007.01, 2007.
15. F. Dornaika and J. Ahlberg, "Fast and reliable active appearance model search for 3D face tracking," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 34, p. 1838–1853, 2004.
16. T. Gramegna, L. Venturino, G. Cicirelli, G. Attolico, and A. Distante, "Optimization of the POSIT algorithm for indoor autonomous navigation," *Robotics and Autonomous Systems*, vol. 48, pp. 145-162, 2004.
17. D. DeMenthon and L. S. Davis, "Model-based object pose in 25 lines of code," *International Journal of Computer Vision*, vol. 15, pp. 123-141, 1995.
18. Y. John, "A General Photogrammetric Method for Determining Object Position and Orientation," *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 129-142, 1989.
19. R. M. Bodade and S. N. Talbar, "Dynamic iris localisation: A novel approach suitable for fake iris detection," 2009.
20. Y. J. Koa, E. C. Lee, and K. R. Park, "A robust gaze detection method by compensating for facial movements based on corneal specularities," *Pattern Recognition Letters*, vol. 29, pp. 1474-1485, 2008.
21. Z. Zhu and Q. Ji, "Novel Eye Gaze Tracking Techniques Under Natural Head Movement," *IEEE Transactions on Biomedical Engineering*, vol. 54, pp. 2246-2260, 2007.