Intelligens ébresztőóra

Czompó Gábor, Sánta Zsolt, Tasi Tamás

2011. május 9.

1. Bevezető

Ez a projekt a Szegedi Tudományegyetem 2010/2011-2. félévében meghirdetett *Számítógépes látás* kurzus keretein belül elkészítendő kötelező feladat. A csapat tagjai:

Név	E-mail cím
Czompó Gábor	Czompo.Gabor@stud.u-szeged.hu
Sánta Zsolt	Santa.Zsolt.1@stud.u-szeged.hu
Tasi Tamás	Tasi.Tamas.Samuel@stud.u-szeged.hu

1.1. Feladatkiírás

A feladat egy intelligens ébresztőóra tervezése, ami figyeli felhasználót és addig folytatja az ébresztést, amíg ténylegesen fel nem kel. A megoldásnak robusztusnak kell lennie az esetleges felhasználótól független mozgó elemekre (pl: házi állatok vagy emberek) és csak a felhasználót kell követnie.

1.2. Alternatívák

A projekthez nagyon hasonló munka található a következő munkában: [4]. A szerző egy adaptív ébresztőóra tervét vázolja, ami a mozgásmennyiségtől függően meghatározza a felhasználó alvási fázisát és az ébresztést egy éberebb alvási fázisban végzi.

2. Rendszerterv

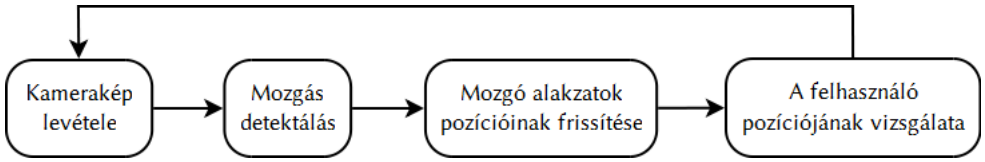
A rendszer egy számítógépből (amin a programunk fut) és egy egyszerű webkamerából áll. A webkamerát úgy kell felszerelni, hogy az alvóhelynél nagyobb méretű területet tudjon megfigyelni. A felszerelés után meg kell adni az alvóhely területét (pl: az ágy befoglaló téglalapját), amire a vezérlőprogram lehetőséget biztosít. Ezután be lehet állítani az ébresztés időpontját.

Ha elérjük a beállított időpontot, elindul az ébresztés és a rendszer folyamatosan figyeli a felhasználót. Akkor mondhatjuk, hogy a felhasználó felébredt, ha a detektált alakja teljesen elhagyta az alvóhelyet és nem tér vissza oda. Fontos megjegyezni, hogy a rendszer csak egyetlen ember ébredését képes követni, aki az alvóhelyen először megmozdul az ébresztésre.

A projektet teljesen a MATLAB rendszer segítségével szeretnénk megvalósítani. A kamerakép elkészítéséhez a MATLAB *Image Acquistion Toolbox*-át használjuk [1].

3. Az algoritmusok

A videót valós időben, képekként kapjuk a kameráról, majd ezeket dolgozzák fel az algoritmusok. A feldolgozás az ébresztéssel egy időben kezdődik. A videón először detektáljuk a mozgó objektumokat, majd azok pozícióit összevetjük a korábban detektált mozgó objektumokéval. Így megtudjuk, hogy az új pozícióba melyik korábbi objektum került. Ezután megvizsgáljuk a felhasználó alakjának befoglaló téglalapját és amennyiben teljes terjedelmével kívül esik az alvóhely befoglaló téglalapjából, a felhasználót ébernek tekintjük és befejezzük az ébresztést. A 1. ábra összefoglalja a rendszer működését.



1. ábra. A fő ciklus

3.1. Mozgás detektálás

A mozgás detektálás első lépéseként szeretnénk elkülöníteni az előtér és a háttér pixeleket egymástól egy alkalmas *background subtraction* technikával. Fontos szempont, hogy a detektálás valós időben történjen és nem áll rendelkezésre a teljes videó. A *background subtraction* technikákról összefoglalók találhatók itt: [8, 7, 2]. Ehhez a projekthez az egyszerűbb *frame difference* technikát választottuk, azaz minden iterációban vesszük két kép különbségének abszolút értékét és amennyiben ez egy küszöbérték felett van az előtérhez vesszük, különben a háttérhez:

$$F(p) = \begin{cases} 1, & |frame_i(p) - frame_{i-1}(p)| > T_1 \\ 0, & else \end{cases} \tag{1}$$

Ezután a képet felosztjuk 16x16-os blokkokra, majd összesítjük az előtér pixeleket a blokkokban és ahol ezek száma meghalad egy küszöböt, megjelöljük a blokkot.

$$M = \left\{ B \text{ blokk } \mid \sum_{p \in B} F(p) > T_2 \right\} \tag{2}$$

A második mérföldkőre készülve, tesztjeink kimutatták, hogy szükséges a lehet blokkok közötti összefüggőség javítása, ezért a kapott blokk képre (egy bináris kép) végrehajtunk egy morfológiai zárást.

Végül a blokkok között 8-szomszédságot vizsgálva felcímkézzük az összetartozó blokkokat. Az így kapott címkézéssel meghatározhatjuk az összetartozó blokkok befoglaló téglalapjait. **Ezeken elvégzünk egy újabb küszöbölést, amennyiben a mozgó objektumhoz tartozó blokkok száma nagyobb egy T_3 küszöbnél továbbadjuk a következő fázisnak. A további felhasználás céljából kiszámítjuk a szegmentált objektumok RGB hisztogramait. A hisztogram számításánál 8-8-8-as kvantálást alkalmazunk, így végül 512 színárnyalattal dolgozunk. A hisztogramot a számításnál normaliazáljuk (később szükség lesz rá).**

3.1.1. Alternatívák

Alternatívaként megemlíthetünk egy főkomponens analízisen alapuló szegmentálást, amit egyelőre elvetettünk, mivel nem valós időben vizsgálja a képkockákat [9, 5]. További kísérletek során megvizsgálhatjuk, mennyire befolyásolja a referencia képkockák száma az eljárás hatékonyságát.

Végül megvizsgáltunk további két mozgás detektáló eljárást is, de mindkét modellt olyan bonyolultabb esetek lekezelésére dolgozták ki (pl: instabil kamera, dinamikusan mozgó háttér), amelyek nálunk egyelőre nem fordulnak elő [3], [6].

3.2. Mozgás követés

A mozgás követése a [10] cikkben javasolt megoldáson alapul. A detektált objektumokat összevetjük korábban letárolt objektumokkal és kiszámítunk egy távolságot. Minden letárolt objektumra meghatározzuk a legközelebbi detektált objektumot és amennyiben a távolság egy előre lekötött küszöbnél kevesebb frissítjük a tárolt objektumot. A követés során feltételezzük, hogy az objektumok nem tűnhetnek el (kivéve, ha elhagyják a képet). A távolság függvényünk egyelőre tesztelés alatt van, de jelenleg a következő formájú:

$$d(O_1, O_2) = \lambda_1 \|c_1 - c_2\|^2 + \lambda_2 D_{KL}(O_1.colorHist, O_2.colorHist),$$

ahol $c_i, i = 1, 2$ az adott objektum befoglaló téglalapjának középpontjai, $\lambda_i, i = 1, 2$ a komponensek súlyai ($\lambda_1 = 1.5$ és $\lambda_2 = 150$) és D_{KL} a cikkben javasolt Kullback-Leibler távolság két színeloszlásra:

$$D_{KL}(P, Q) = \sum_{i \in RGB} P(i) \log \left(\frac{P(i)}{Q(i)} \right).$$

Kísérleteinkből azt a következtetést vonhattuk le, hogy szükséges a mozgás követésre a fentebb leírt algoritmus, mivel nem feltételezhetjük, hogy az alvó személy minden képkockán mozogni fog. Ebből következően előfordulhat, hogy eltűnik a detektált objektumok közül és így nehezebb több képkockán keresztül észlelni. A színeloszlás vizsgálata igazából akkor jelentős, ha két közel azonos pozíciójú objektumunk van, hiszen ilyenkor a pozíció nem feltétlenül ad pontos szegmentálást.

Az utolsó mérföldkőre készülve tesztjeink kimutatták, hogy szükséges az objektumok listájának a karbantartása és a detektált, de nem mozgó objektumok törlése. Nem mozgó az az objektum, ami 5 másodpercen keresztül nem módosul és a távolsága az ébresztési területtől nagyobb, mint az ébresztésre definiált távolság küszöbérték. Ez alól csak az az eset kivétel, amikor egyetlen objektumunk van, az ébresztendő személy.

3.3. Ébresztés

A felhasználót akkor tekintjük ébernek, ha elhagyja a korábban kijelölt alvóhely területét. Az alvó embernek az ágy területén detektált mozgó objektumok közül azt tekintjük, amire igazak a következő szempontok:

- A legkisebb a távolsága az ágytól (középpontok távolsága)
- Legalább 20 framen át mozog. Ez kb. 4 másodperc folyamatos mozgást (ébredezést) jelent.
- Az ágytól vett távolsága nem haladhatja meg az ébresztésre definiált küszöbérték másfélszeresét.

Amennyiben nincs ilyen objektum vagy a követett objektum nagy mértékben eltávolodik a korábban detektált változatától, az alvó ember objektumát kinullázzuk és az ébresztést folytatva a következő iterációban újrapróbálkozunk. Végül, ha a ébresztendő személy távolsága meghaladja az ébresztési küszöböt, úgy vesszük, hogy felébredt és befejezzük az ébresztést.

4. Második mérföldkő

A második mérföldkőre kétféle tesztelést végeztünk, mindkét esetben az első mérföldkőre javasolt algoritmussal. Először egy vizuális tesztelést készítettünk, majd megvizsgáltuk az implementált algoritmusok futási idejét. A vizuális tesztelés eredménye megtekinthető a mellékelt videó fájlokban. Összességében robusztusan tudjuk detektálni a mozgó objektumokat és a körvonalakkal elértük a kívánt pontosságot. Problémát jelent a mozgás követésnél említett hiányosság, de erre közben adtunk egy megoldást, továbbá növeltük a detektált objektumok belső összefüggőségét egy morfológiai zárással. Végül észrevettük, hogy nem elég vizsgálni a követendő objektum középpontját az ébresztésnél, mivel ez elhagyhatja az ágy területét anélkül, hogy az ébresztendő személy elhagyná az ágyat.

A futási idő vizsgálatát a MATLAB beépített függvényei segítségével végeztük és valós időt számítottunk. A tesztet kettéosztottuk az algoritmus futási és a kirajzolási műveletekre. A tesztelést befolyásolja a levett kép felbontása, ezért ezt is feltüntetjük:

1. táblázat. A futási idők tesztelése

Felbontás	Algoritmus futási idő	Kirajzolási műveletek
320x240	0.0194	0.0551
640x480	0.0544	0.0679

Mivel a tesztgépeink teljesítménye közel azonos, átlagoltuk a mérési eredményeket. A blokkok méretét felbontásonként változtattuk 8, illetve 16 darabra.

A fő algoritmussal párhuzamosan folyamatosan készült a program felhasználói felülete is.

Hivatkozások

[1] Matlab image acquistion toolbox. `http://www.mathworks.com/help/toolbox/imaq/`.

[2] Seth Benton. Background subtraction, part 1: Matlab models. `http://www.eetimes.com/design/signal-processing-dsp/4017685/Background-subtraction-part-1-MATLAB-models`, 2008.

[3] Tadaaki Hosaka, Takumi Kobayashi, and Nobuyuki Otsu. Object detection using background subtraction and foreground motion estimation. *IPSJ Transactions on Computer Vision and Applications*, 3:9--20, 2011.

[4] Jasper Kuijsten. Adaptive alarm clock using movement detection to differentiate sleep phases. *Bachelor Project*, 2010.

[5] Kui Liu, Qian Du, He Yang, and Ben Ma. Optical flow and principal component analysis-based motion detection in outdoor videos. *EURASIP J. Adv. Signal Process*, 2010:1:1--1:6, February 2010.

[6] K. A. Patwardhan, G. Sapiro, and V. Morellas. Robust foreground detection in video using pixel layers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(4):746--751, 2008.

[7] Massimo Piccardi. Background subtraction techniques: a review. `http://www-staff.it.uts.edu.au/~massimo/BackgroundSubtractionReview-Piccardi.pdf`, 2004.

[8] Mandeep Singh. Improved morphological method in motion detection. *International Journal of Computer Applications*, 5(8):5--8, August 2010. Published By Foundation of Computer Science.

[9] Nicolas Verbeke and Nicole Vincent. A pca-based technique to detect moving objects. In Bjarne Ersbøll and Kim Pedersen, editors, *Image Analysis*, volume 4522 of *Lecture Notes in Computer Science*, pages 641--650. Springer Berlin / Heidelberg, 2007.

[10] Tao Yang, Stan Z. Li, Quan Pan, and Jing Li. Real-time multiple objects tracking with occlusion handling in dynamic scenes. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 970--975, Washington, DC, USA, 2005. IEEE Computer Society.