Tartalomjegyzék

Optical Flow meghatározása energiaminimalizációs módszerrel

Horváth Péter, Kató Zoltán

Szegedi Tudományegyetem, Informatikai Tanszékcsoport,
Pf. 652, 6701 Szeged, Fax: 62/546 397
Horvath.Peter.3@stud.u-szeged.hu, kato@inf.u-szeged.hu

Absztrakt. A cikkünkben *optical flow* becslésére mutatunk be algoritmust melynek legfontosabb rész a Mumford-Shah által bevezetett energia függvény megoldása, melyet kiterjesztünk mozgások detektálására. Az inicializálásra használt vektor mezőre bemutatunk egy *multi-scale* technikát, melyet egy *k-means* klaszterező algoritmus követ. Ezen kezdeti optical flow-t, mint inicializálást használjuk az energiaminimalizálás során, melyet *simulated annealing* algorimussal optimalizálunk.

1. Bevezetés

A mozgás nagyságának és irányának meghatározása nehéz de igen sok gyakorlati alkalmazásban fontos probléma, melyhez gyors algoritmusokra van szükségünk. P. Anandan 1989-ben megjelent cikkében [1] találkozhatunk *multi-scale* technikával, ezen alapötlet továbbfejlesztésével készítettük a cikk második részében bemutatásra kerülő eljárást, mely a mozgás nagyságára és irányára próbál becslést adni. Először bemutatjuk a triviális algoritmust melynek futásideje $\Theta(N^4)$. Könnyebb dolgunk van ha feltesszük, hogy nem lehetnek a képen egy pixelnél nagyobb mozagások, erre is bemutatunk egy technikát, majd megnézzük hogyan tudjuk elérni azt, hogy ne legyenek egy pixelnél nagyobb mozgások. P. J. Burt és E. H. Adelson 1983-ban megjelent cikkében [2] bemutat egy *Gauss-piramis* nevű képkicsinyítési eljárást, mely segítségével felére kicsinyítjük az adott képet, de nem egyszerű átlagolási technikával hanem egy Gauss konvolúciós maszkot használva, ennek többszöri alkalmazásával egy piramist állítunk elő, melynek megfelelően magas szintjén már nincsenek egy pixelnél nagyobb mozgások. Itt meghatározzunk a kis mozgásokat, majd a piramis egyel alacsonyabb szintjére ugrunk és itt finomítjuk a becslésünket és így folytatjuk az eljárást a legalsó szintig. Mivel minden szinten minden egyes pixelnek csak egy nagyságú környezetét kell vizsgálnunk igy csak konstans időt töltünk minden pixel vizsgálatával amely azt jelenti, hogy a futásidő $\Theta(N^2 log^2 N^2)$-re csökken. Az így kapott becslésünk bár jól közelíti a valóságot de sok hibaforrás lehetséges, zajos képen előfordul, hogy egy-egy vektor nem mutat korrekt eredményt azaz egészen más irányba mutat mint a környezetében lévő szomszédok ezt a hibát simítással lehet javítani, a másik hibája az algoritmusnak, melyen még a simítás sem segít: a homogén mozgó területek illetve a háttérbe beleolvadó mozgó objektumok detktálása,

mivel szín alapú a folyamat, nem érzékelheti ezeket. Mindenzen hibák miatt döntöttünk energiaminimalizációs módszerek használata mellett. Bemutatjuk Mumford-Shah által kidolgozott szegmentálási eljárást [6], melyet átdolgoztunk optical flow meghatározására. F. Gibou és R. Fedkiw [5] publikáltak egy módszert melyben *k-means* algoritmus alkalmazása után végezték a szegmentálást, módszerünkben mi is k-means algoritmussal inicializáljuk az energiaminimalizáló függvény klasztereit. Miután megtörtént a kezdeti osztályok beállítása a függvény minimalizálására több lehetőségünk is van, cikkünkben *Simulated Anealing* felhasználásával minimalizálunk, de T. F. Chan és L. A. Vese [3] cikkében található *Level-Set* eljárás segítségével is megoldható a probléma. N. Paragios [7] 2000-ben publikált PhD dolgozatában foglalkozik aktiv kontúrok energiájának minimalizációjával, *Level-Set* függvények optimalizálásával, mozgás detektálásával. D. Cremers [4] PhD dolgozatában az aktív kontúrokat további jellemzők hozzáadásával bővíti, így a priori tudást visz a rendszerbe, mely működését mozgás detektálására is bemutatja.

2. Optical flow becslése

Az optical flow egy képsorozat két időben egymást követő képkockáján végbemenő mozgásokat mutató vektormező. Jelölje I_1 és I_2 a képsorozat első és második szekvenciáját, melyek között eltelt Δt idő alatti mozgások vektoraira fogunk becslést adni. $\overline{OF[i,j]}$ jelölje azt, hogy az I_1 kép $[i,j]$. pixelén lévő objektum Δt idő alatt milyen irányban és mennyit mozgott, azaz hol található az I_2 képszekvencián ($[i,j] + \overline{OF[i,j]}$).

2..1. Triviális algoritmus a mozgás meghatározására

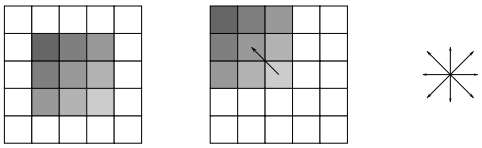
A triviális algoritmus az első képkocka minden pixelére megvizsgálja a második képen levő összes pixelt és közülük a leghasonlóbbat választja ki:

$$\overline{OF[i,j]} = \arg \min_{\substack{k=1\dots N-1 \\ l=1\dots N-1}} |I_1[i,j] - I_2[k,l]| - [i,j]. \tag{1}$$

$I_1[i,j]$ és $I_2[i,j]$ az első és második képszekvencia pixelei. A triviális algoritmus futásideje a kép pixelszámának másodfokú polinomiális függvénye, $\Theta((M \times N)^2)$, ahol $M \times N$ a kép mérete. Ez gyakorlatban nem használható, ezért a probléma megoldására most bemutatunk egy $\Theta(N^2 log_2^2 N^2)$ futásidejű algoritmust.

2..2. Maximálisan egy pixel nagyságú mozgások meghatározása

Ha feltételezzük, hogy a képen egy pixel nagyságúnál nagyobb mozgások nem lehetnek, akkor könnyebb dolgunk van. Ebben az esetben az előző részben bemutatott mószert használva számíthatjuk a mozgásvektorokat, de csak egy pixel nagyságú környezetet vizsgálva, így két dimenziós mozgásvektorokat kapunk, melyek értékei $(-1 \dots 1)$ közöttiek lesznek. Ez a módszer nagyon gyors eredményt szolgáltat, de nem minden esetben korrekt. Ennek javított változata amikor a



1. ábra: Eljárás maximálisan egy pixel nagyságú mozgás meghatározására.

keresett pixel valamilyen környezetét vizsgáljuk (1. ábra), célszerű ezt súlyozni egy Gauss normális eloszlású mátrixszal, mely mátrix elemeinek összege 1. A normál eloszlás sűrűségfüggvénye segítségével számoljuk ki a Gauss mátrixot a következőképpen:

$$G[x,y] = f_{\underline{m},\Sigma}(x,y) \tag{2}$$

$$G'[x,y] = G[x,y] / \sum_{k,l=-\lfloor R/2 \rfloor}^{\lfloor R/2 \rfloor} G[x,y], \tag{3}$$

ahol R a környezet nagysága, így a mozgásvektorok keresése a következőképp zajlik. $OF[i,j] = [x,y]$, ahol $[x,y]$:

$$\arg \min_{\substack{-1 \le x \le 1 \\ -1 \le y \le 1}} \sqrt{\sum_u \sum_v G'[u,v](I_1[i+u, j+v] - I_2[i+u+x, j+v+y])^2}. \tag{4}$$

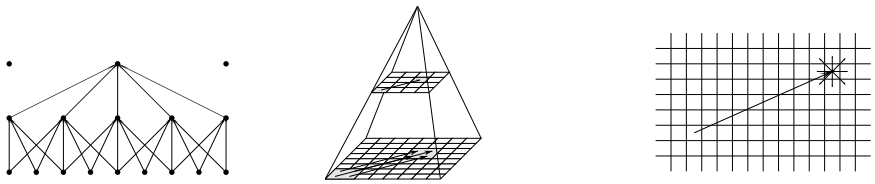
Az eddigiek során megismertük hogyan lehet maximum egy pixel nagyságú mozgásokat meghatározni, most megnézünk egy eljárást, mely segítségével a nagyobb mozgásokat egy pixel nagyságúra redukálhatjuk.

2..3. Piramis nagyságának meghatározása és felépítése

Tekintsünk egy képet, amelyen egy x nagyságú mozgást találhatunk, ha ezt a képet n-szeresére nagyítjuk, akkor a mozgás is n-szeres lesz. Ezt az állítást felhasználva ha felére csökkentjük a képet akkor a mozgások is felére csökkenek. Így ha a képet x-edére csökkentjük, akkor a mozgás pont 1 egység nagyságú lesz. Itt bemutatunk egy eljárást, amely segítségével igen gyorsan felére csökkenthetjük a kép méretét, az eljárás a Gauss piramis [2] felépítésének része.

Az elkészítéshez egy normál eloszlású $N(0,1)$ paraméterű fentebb bemutatott Gauss mátrixot használunk. Egy $2n+1 \times 2n+1$ nagyságú képből indulunk és egy $n+1 \times n+1$ mértűt készítünk. Konvolúciót alkalmazunk, a (2 a). ábrán látható az 1 dimenziós megvalósítás.

Az eljárást alaklmazva elértük, hogy a képet lineáris futásidejű algoritmussal felére csökkentettük. Itt jegyezzük meg, hogy több más lehetőségünk is van a képet átméretezni, például a lefedett pixelek átlaga esetleg súlyozott átlaga. Így elértük azt, hogy a mozgások a képen a felére csökkentek. A gondolatmenetet tovább folytatva a kapott képet ismét felér csökkentjük és így a mozgások negyedére, nyolcadára,... csökkennek. Így az x nagyságú mozgás a k. lekicsinyítés után $x/2^k$ nagyságú lesz. Már megismertük a maximum 1 pixel nagyságú



(a) (b) (c)

2. ábra: A Gauss piramis kiszámításának menete (a) ábra, a visszafelé dolgozó stratégia (b) ábra, a becslés finomítása (c) ábra.

mozgások detektálását. Ha tudjuk, hogy maximum k nagyságú mozgások vannak a képen akkor $\log_2 k$ darab képből kell állnia a piramisnak, így a $\log_2 k$. képen már csak maximum 1 pixel nagyságú mozgások lesznek.

2..4. Visszafelé dolgozó stratégia

A piramis felépítése során az utolsó képen meghatározzuk a maximum 1 pixel nagyságú mozgásokat, ezután átugrunk az utolsó előtti szintre, itt az imént megállapított mozgások kétszer akkorák lesznek ám ez nyilván csak egy durva becslés. Tehát az $i+1$. szinten lévő Optical Flow-ból úgy készítjük a nagyobb i. szinten lévőt, hogy az $OF_i(2k, 2l)$, $OF_i(2k+1, 2l)$, $OF_i(2k, 2l+1)$ és $OF_i(2k+1, 2l+1)$ elemei megkapják az $OF_{i+1}(k, l)$ vektor hosszának kétszeresét (2 b. ábra). Ezek után jön az algoritmus egyik kulcslépése, ez a finomítás, amley során a vektorokról lévő durva becslésünket finomítjuk. A finomítás a durva becslés 3×3-as környezetében megkeresi pontosan melyik pixel is lehet a második képen a mozgás korrekt helye. Ezt pedig a 2..2. fejezetben ismertetett maximálisan egy pixel nagyságú mozgások detektálására szolgáló módszer segítségével végezzük (2 c. ábra).

3. Az Optical Flow simítása a Mumford-Shah energiaminimalizációs eljárás segítségével

Mumford és Shah [6] bemutatott egy általános szegmentálási módszert, a szegmentálást egy függvény segítségével írták fel:

$$E(f, \Gamma) = \mu^2 \int\int_R (f-g)^2 dxdy + \int\int_{R-\Gamma} \|\nabla f\|^2 dxdy + \nu|\Gamma|, \qquad (5)$$

ahol f a szegmentálás, g az eredeti kép, Γ a határ a régiók között és R a régiók összessége. Az első rész kicsi, ha a g hasonlít az f-re, a második a simaságot biztosítja és a harmadik a kontúr hosszának minimalizálásáról gondoskodik. Itt bemutatjuk az M-S speciális esetét optical flowra:

$$e = \alpha \int\int_\Omega (I_2(x,y) - I_{shynt}(x,y))^2 dxdy + \beta \int\int_{\Omega\setminus\Gamma} E_{OF}(x,y) dxdy + |\Gamma|, \quad (6)$$

ahol α és β súlyparaméterek, I_{shynt} egy olyan kép, melyet az első képszekvencia és az optical flow segítségével készítünk el a következőképpen $\forall i, j$:

$$I_{synt}(i + OF(i,j).x, j + OF(i,j).y) = I_1(i,j). \tag{7}$$

Legyen E_{OF} az optical flow parciális deriváltja, ez a homogén területeken kis értékeket vesz fel viszont a határokon az értéke magas. Közelítsük a következőképpen:

$$E_{OF} = \sqrt{S_1(OF.x)^2 + S_2(OF.x)^2 + S_1(OF.y)^2 + S_2(OF.y)^2}, \tag{8}$$

ahol S_1 és S_2 a Sobel gradiens operátorok. Az egyenlet harmadik része $|\Gamma|$, ahol Γ a határpontok halmaza és $|\Gamma|$ ezen pontok száma, azaz a kontúr hossza.

4. Az Optical Flow, k-Means és a Mumford-Shah algoritmusok kapcsolata

A 2. fejezetben bemutattuk hogyan tudjuk az optical flow-t becsülni. A 3. fejezetben láthattuk a Mumford-Shah energiafüggvényt és optimalizálását. Láthattuk, hogy az M-S eljáráshoz a kezdeti kép klaszterezésére van szükségünk. Erre a célra használunjuk az u.n. *k-means* klaszterező eljárást [5], mely segítségével az optical flow vektorait osztályokba soroljuk, majd ezen osztályokkal mint az M-S inicializáció lépése oldjuk a meg a problémát. Az algoritmus:

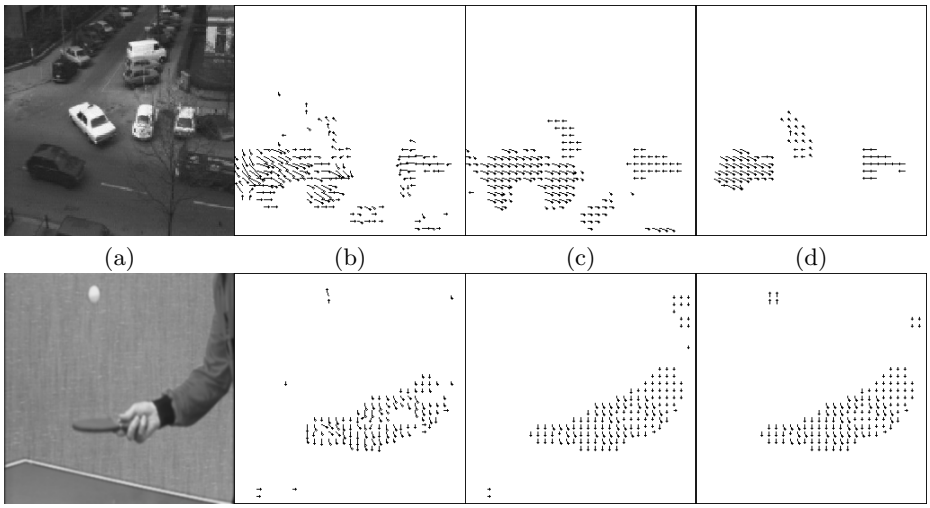
```
Optical Flow(Image 1, Image 2): OF
    Estimation(Image 1, Image 2) =: OF;
    K-means(OF, int Clusters);
    Mumford-Shah(Image 1, Image 2, OF) =: OF;
        a. Fast Marching;
        b. Simulated Annealing;
        c. ...
  return OF;
```

A k-means algoritmusban a *Clusters* a keresett osztályokra ad felső korlátot. Az M-S megoldásánál több módszer közűl választhatunk [7].

5. Kísérleti eredmények

Az energiafüggvény inicializálását a 2. fejezetben bemutatott algoritmussal végezzük, majd k-means algoritmussal klasztereztük a vektorokat. Az energiafüggvény minimalizálását szimulált hűtéssel végeztük. Két tesztesetet mutatunk be az egyik a jól ismert hamburgi taxi sorozatból való a másikon egy asztalitenniszező játékos.

Hamburg taxi sorozat: Az eredeti kép a (3 a). ábrán látható. A sorozat 15. és 18. szekvenciája között számoltuk az optical flow-t. A becslést a (b) ábrán láthatjuk, esetek túlnyomó többségében helyesek a vektorok, de mivel a kép eléggé zajos találunk hibákat. A (c) ábrán a k-means algoritmus eredménye, kezdetben 10 osztállyal, és a (d) ábrán a szimulált hűtés eredménye.



(a) (b) (c) (d)

(e) (f) (g) (h)

3. ábra: A hamburgi taxi sorozat 15. szekvenciája (a), a becsült optical flow (b), a k-means klaszterezés eredménye (c) és az M-S megoldása szimulált hűtéssel (d). Az asztaliteniszező kép (e), az optical flow becslése (f), a k-means eredménye (g) és az M-S enegiafüggvény szimulált hűtéssel optimalizálva (h).

Asztalitenniszező játékos: A (3 e, f, g és h) ábrákon láthatjuk a képeket, a minősége jobb mint a taxi sorozaté, így a legtöbb alkalmazásban már a (3 f) ábrán látható becslés is jól használható, a (g) ábrán a k-means utáni klaszterezés és a (h)-n a szimulált hűtés eredménye.

Irodalom

1. P. Anandan: A Comutational Framework and an Algorithm for the Measurement of Visual Motion, Int. J. Computer Vision, Vol. 2, No. 3, pp. 283-310
2. P. J. Burt and E. H. Adelson: The Laplacian Pyramid as a Compact Image Code, IEEE Transactions on Communications, Vol. COM-31, no. 4, april 1983 pp. 532-540
3. T. F. Chan, B. Y. Sandberg, L.A. Vese: Active Contours without Edges for Vector-Valued Image, N00014-96-1-0277, DMS-9626755.
4. D. Cremers: Statistical Shape Knowledge in Variational Image Segmentation, PhD Thesis 2002, Mannheim.
5. F. Gibou, R. Fedkiw: A Fast Hybrid k-Means Level Set Algorithm for Segmentation
6. D. Mumford and J. Shah: Optimal Approximations by Piecewise Smooth Functions and Associated Variational Problems, Communications on Pure and Applied Mathematics, Vol. XLII pp. 577-685, 1989.
7. N. Paraigos: Geodesic Active Regionas and Level Set methods, PhD Thesis 2000