O METODĂ ROBUSTĂ DE SUPRAVEGHERE VIDEO BAZATĂ PE DETECŢIA MIŞCĂRII

Bogdan Vasilescu[1], Tiberiu Stănescu[1], Gabriel Rădulescu[2]
1 – Student; 2 – Conf.dr.ing.
Universitatea Petrol-Gaze din Ploieşti

Introducere

Securitatea completă a unor obiective de interes nu poate fi asigurată fără o supraveghere video eficientă, care să permită atât monitorizarea în timp real a evenimentelor cât şi înregistrarea video, în scopul verificărilor ulterioare. În plus, supravegherea „inteligentă" implică şi detecţia automată a încercărilor de intruziune în zonele de securitate a obiectivului monitorizat şi alarmarea în timp util a personalului de intervenţie.

Prezenta lucrare şi-a propus prezentarea unui sistem de supraveghere video realizat de autori, sistem ce implementează şi funcţii de alarmare la sesizarea prezenţei persoanelor în zona de interes, respectiv de înregistrare a secvenţelor video asociate derulării acestor evenimente. Specificul acestuia este faptul că foloseşte tehnici moderne de analiză în timp real a imaginilor digitale în scopul detecţiei intruziunii, înregistrarea video fiind realizată de asemenea în tehnică digitală.

Dimensiunea software asociată sistemelor de supraveghere video a cunoscut o dezvoltare spectaculoasă o dată cu progresele înregistrate de tehnica de calcul. Astfel, algoritmii folosiţi pentru procesarea de imagini îşi găsesc acum solide fundamente matematice [6], dând naştere la noi abordări mai rapide şi mai precise, ce pot rezolva probleme anterior inabordabile. De cele mai multe ori, creşterea de viteză câştigată prin folosirea unui algoritm mai rapid este considerabilă, în anumite cazuri atingând câteva ordine de mărime [5]. De aceea, algoritmii mai rapizi fac aplicabile multe tehnici de procesare a imaginilor şi reduc considerabil costurile fizice ale sistemelor.

Principii şi metode de detecţie a mişcării

Analiza mişcării reprezintă un domeniu provocator şi în continuă expansiune. Există tehnici variate pentru detectarea mişcării, principial percepute ca operaţii filtru pentru imaginile spaţiu-timp [1].

Intuitiv, se asociază mişcarea cu schimbările. Astfel, discuţia despre analiza mişcărilor începe prin observarea diferenţelor dintre două imagini ale unei secvenţe. Figurile 1 a şi b descriu o pereche de imagini dintr-o zonă de construcţii. Există anumite diferenţe între acestea care nu sunt evidente la o comparare directă.

Cu toate acestea, dacă se extrage o imagine din cealaltă diferenţele devin imediat vizibile (figura 1.c).

Fig.1: a,b - Pereche de imagini din zona de construcţie a unei clădiri; c – Diferenţa dintre imaginile a şi b.

În colţul din stânga jos al imaginii, un camion s-a mutat, în timp ce maşina aflată chiar în spatele acestuia este evident parcată. În centrul imaginii se observă conturul unui pieton ce este foarte puţin vizibil în imaginile originale. Punctul strălucitor dintr-o linie din partea de sus a imaginii reprezintă biciclişti ce se deplasează de-a lungul unei alei pentru biciclete. Datorită mărimii duble a contururilor se poate presupune că bicicliştii se deplasează mai repede decât pietonul. Deşi aceasta este doar o descriere calitativă, este evident că analiza mişcării ajută mult la înţelegerea unei astfel de scene.

Prin urmare, se pot sintetiza cele afirmate până acum şi se poate spune că mişcarea rezultă din schimbarea pe un orizont de timp determinat a valorilor nivelurilor de gri [2]. Din nefericire, reciproca, anume faptul că toate schimbările nivelurilor de gri sunt datorate mişcării, nu este adevărată şi se pot da o serie de exemple în acest sens.

Descrierea aplicaţiei de supraveghere video

Aplicaţia este structurată ca în figura 2. Fluxurile video sunt preluate de un selector [8], fiind succesiv supuse unor prelucrări primare (redimensionare, îmbunătăţirea contrastului sau a luminozităţii etc.) pentru a fi aduse la o formă optimă pentru modulul detector. Acesta implementează o serie de algoritmi de detecţie a mişcării, reprezentaţi prin componentele funcţionale Det1...Det4, şi poate fi uşor extins prin adăugarea de submodule suplimentare. La detectarea unor zone cu mişcare se generează un semnal de alarmă şi se evidenţiază pe imagine regiunile de interes, oferindu-se totodată posibilitatea salvării pe disc a secvenţei în cauză pentru verificări ulterioare.

Există mai multe abordări pentru detecţia mişcării într-o secvenţă video continuă. Toate sunt bazate pe compararea cadrului curent cu un cadru preluat din secvenţele anterioare sau cu fundalul.

O primă abordare (detector tip 1) se bazează pe comparaţia dintre cadrul curent şi cel anterior folosind filtrele de diferenţă şi prag [7]. După acest pas se

obţine o imagine ce va conţine pixeli albi în locurile în care cadrul curent diferă faţă de cadrul anterior pentru o valoare de prag specificată. Dacă numărul acestora va fi mai mare decât un nivel predefinit de alarmare, se poate semnala detectarea unui eveniment de mişcare şi se pot evidenţia regiunile în care apare mişcarea prin culoarea roşie.

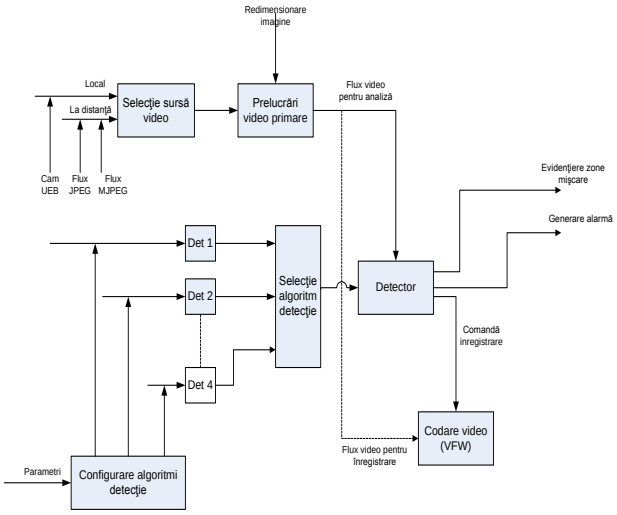

Fig. 2 - Structura aplicaţiei

Exemplul următor este un extras din codul C# asociat aplicaţiei dezvoltate, ce ilustrează succesiunea acestor prelucrări.

```
// crearea unui filtru
Difference differenceFilter = new Difference();
Ifilter thresholdFilter = new Threshold( 15 );
// setarea fundalului pentru suprapunere la filtrul diferenţă
differenceFilter.OverlayImage = BackgroundFrame;
// aplicarea filtrelor
Bitmap tmp1 = differenceFilter.Apply ( currentFrame );
Bitmap tmp2 = thresholdFilter.Apply ( tmp1 );
```

Rezultatul este prezentat în figura 3.a. Principalul dezavantaj al acestei abordări este reprezentat de imposibilitatea observării întregului obiect în mişcare, întrucât dacă obiectul se mişcă lin apar doar schimbări minore de la un cadru la altul.

Următoarea abordare propune o posibilă îmbunătăţire a acestui procedeu, prin compararea cadrului curent nu cu anteriorul, ci cu primul cadru din secvenţa video. Astfel, dacă nu a existat niciun obiect în cadrul iniţial, compararea cadrului curent cu primul va reda întregul obiect aflat în mişcare, independent de viteza lui de deplasare. Şi această abordare are însă un dezavantaj major. Dacă, de exemplu,

a existat în primul cadru un obiect care a dispărut ulterior, se va detecta întotdeauna mişcare în locul în care el s-a aflat. Deşi se poate reînnoi cadrul iniţial periodic, nici această soluţie nu dă rezultate bune în cazurile în care nu se poate garanta că primul cadru conţine numai fundal static. Poate exista însă şi o situaţie inversă. Prin plasarea unui tablou pe perete se detectează mişcare până când cadrul iniţial va fi reînnoit.

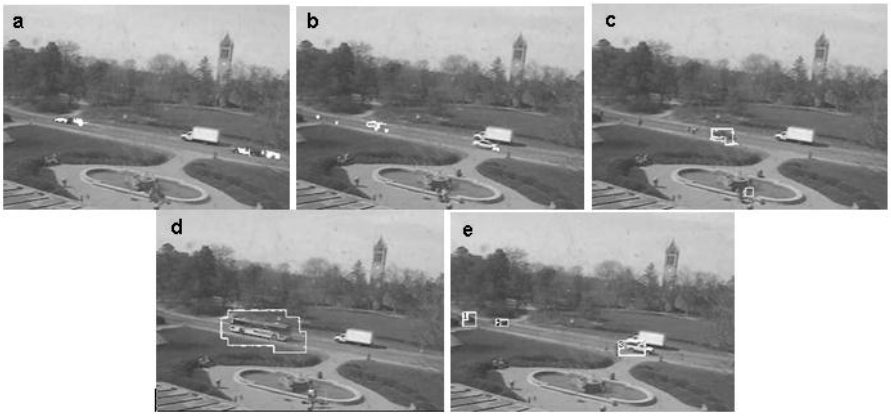

Fig.3: a-Detector 1; b-Detector 2; c-Detector 3; d-Detector 3 optimizat; e-Detector 4.

Cei mai eficienţi algoritmi sunt bazaţi pe crearea aşa-numitului fundal al scenei şi compararea fiecărui cadru curent cu acesta. Există multe abordări pentru crearea scenei, dar majoritatea sunt foarte complexe. O posibilă implementare consideră iniţial primul cadru din secvenţa video ca fiind fundalul, iar apoi „mută" fundalul către cadrul curent prin aplicarea de filtre specifice [7], conform secvenţei următoare:

```
// creare filtru
MoveTowards moveTowardsFilter = new MoveTowards( );
//mută fundalul către cadrul curent
moveTowardsFilter.OverlazImage = currentFrame;
Bitmap tmp = moveTowardsFilter.Apply( backgroundFrame );
// şterge vechiul fundal
backgroundFrame.Dispose( );
backgroundFrame = tmp;
```

Rezultatele obţinute cu această a doua clasă de algoritmi sunt prezentate în figura 3.b (detector tip 2).

O altă posibilă abordare foloseşte la bază un filtru special de netezire (Pixellate [7]), aplicat înaintea celorlalte procesări ulterioare:

```
// crearea filtrului
Ifilter pixellateFilter = new Pixellate( );
// aplicarea filtrului
Bitmap newImage = pixellateFilter( image );
```

Urmează deplasarea fundalului către cadrul curent, aşa cum s-a arătat mai sus.

După evidenţierea pe canalul roşu a regiunilor cu mişcare se obţin rezultatele prezentate în figura 3.c. Această a treia clasă de algoritmi (identificată ca detector tip 3) oferă şi posibilităţi de optimizare, dintre care s-au abordat câteva prin rescrierea unor filtre şi împărţirea cadrului în 64 de regiuni (o matrice 8 x 8) pentru îmbunătăţirea timpilor de calcul prin procesări locale. Rezultatele obţinute sunt prezentate în figura 3.d (detector tip 3 optimizat).

Abordările precedente evidenţiază prin curbe obiectele în mişcare. Dacă se doreşte încadrarea printr-un dreptunghi a obiectelor (sau aflarea numărului, poziţiei, lăţimii sau înălţimii) este nevoie de biblioteci de funcţii specializate. Prin urmare, folosind astfel de filtre speciale se pot determina numărul, poziţia şi dimensiunile obiectelor pe o imagine binară.

Spre exemplu, următoarea secvenţă de cod încadrează obiectele în mişcare printr-un dreptunghi şi afişează numărul acestora:

```
BlobCounter blobCounter = new BlobCounter( );
// obţinerea dreptunghiurilor obiectelor
blobCounter.ProcessingImage( thresholdedImage );
Rectagle[] rects = blobCounter.GetObjectRectangles( );
// crearea obiectului graphic din imaginea iniţială
Graphics g = Graphics.FromImage( image );
using ( Pen pen = new Pen( Color.Red, 1 ) ) {
      int n = 0;
      // desenarea fiecărui dreptunghi
      foreach ( Rectangle rc in rects ) {
            g.DrawRectangle( pen, rc );
            if ((n < 10)&&(rc.Width>sensibility)&&(rc.Height>sensibility))
            {
                g.DrawImage( numbersBitmaps[n], rc.Left, rc.Top, 7, 9 );
                n++;
            }
      }
}
```

Rezultatul obţinut prin aplicarea acestui procedeu (detector tip 4) este prezentat în figura 3.e.

Concluzii

Sistemul prezentat în lucrarea de faţă reprezintă o soluţie robustă la nevoia de detectare şi prevenire prin supraveghere video a manifestărilor infracţionale, cu aplicare directă în foarte multe sectoare din industrie şi nu numai.

Probabil cea mai cunoscută utilizare a sistemelor de supraveghere este aceea din bănci, magazine, instituţii guvernamentale, cazinouri, etc. În niciun caz supravegherea nu se limitează însă la acest tip de aplicaţii. Sistemele mai pot fi utilizate şi pentru supravegherea traficului în intersecţii, supravegherea anti-vandalism, controlul producţiei într-o fabrică sau supravegherea locuinţelor.

Dezvoltarea aplicaţiei a generat însă o serie de alte probleme apărute indiferent de algoritmul folosit. Dintre acestea se pot aminti interferenţa cauzată de sursele de lumină stradale care creează senzaţia falsă de mişcare sau situaţiile în care este prezent şi cerul în scenă şi mişcarea norilor este semnalizată în mod greşit.

O posibilă soluţie este stabilirea valorii de prag (pentru filtrul Threshold) la o diferenţă de 30-40 de niveluri de gri. Această abordare, deşi rezolvă în mare măsura problema anterioară, produce rezultate nesatisfăcătoare în scene slab iluminate.

O altă soluţie parţială este stabilirea unei măşti pentru a aplica algoritmii doar asupra zonelor de interes [4]. O astfel de mască trebuie să aibă aceleaşi dimensiuni ca imaginea captată şi trebuie să fie o imagine binară. Detecţia mişcării va avea astfel loc doar în zonele albe. Metoda are însă şi dezavantaje deoarece, deşi permite excluderea zonelor cu mişcare constantă, nu rezolvă întreaga problemă cu lumină schimbătoare, umbre sau calitatea scăzută a semnalului video.

O altă direcţie de cercetare viitoare este reprezentată de folosirea histogramei pentru evaluarea fundalului. Un astfel de parametru disponibil în mod dinamic furnizează informaţii despre fundal şi condiţiile de iluminare.

În concluzie, sistemele de supraveghere video prin înregistrare la detecţia mişcării, categorie din care face parte şi sistemul propus de lucrare, cunosc o dezvoltare puternică şi oferă o serie de avantaje faţă de supravegherea video continuă. Dintre acestea se pot menţiona avantajul unei economii de spaţiu de stocare a secvenţelor preluate, precum şi posibilitatea verificării periodice a acestora.

Bibliografie

[1] Jähne, B., *Digital Image Processing*, Ed. Springer-Verlag, Engineering Online Library, 2002
[2] Tinku, A., Ajoy, R., *Image Processing - Principles and Applications*, Ed. John Wiley&Sons, New Jersey, 2005.
[4] Girosi, F., Verri, A., Torre, V., *Constraints for the computation of optical flow in Proceedings Workshop on Visual Motion*, Irvine, Washington, 1989.
[5] Burt, P.J., Hong, T.H., Rosenfeld, A., *Segmentation and estimation of image region properties through cooperative hierarchical computation*, IEEE Trans. SMC, 11:802–809, 1981.
[6] Canny, J. F., *A computational approach to edge detection*, PAMI, 1986.
[7] *http://www.codeproject.com/cs/algorithms/aforge.asp*
[8] *http://www.codeproject.com/cs/media/directxcapture.asp*