6. Segmentarea obiectelor in miscare prin modelarea si eliminarea fundalului ("Background Subtraction")

6.1. Introducere

Scopul acestui laborator este de a prezenta o metoda de segmentare a obiectelor aflate in miscare filmate cu o camera fixa. Scena fiind statica, pixelii scenei vor fi clasificati ca si fundal (background) in timp ce pixelii obiectelor in miscare (cu prezenta tranzitorie in scena) vor fi clasificati ca si obiect (foreground). Ca realizare practica se propune implementarea primelor doua metode de segmentare prezentate in cursul 5: prin diferenta simpla intre imagini successive si prin modelarea background-ului (simpla sau selectiva).

6.2. Segmentarea pixelilor obiect (foreground)

Cerinta fundamentala a acestei metode este de a segmenta/eticheta pixelii care apartin obiectelor (foreground). Pentru aceasta se va construi un model (imagine) a fundalului (background) si pentru fiecare cadru (frame) de imagine *i* se va testa conditia la nivelul fiecarui pixel *(x,y)*:

$$diff = |frame_i(x,y) - backgnd_i(x,y)| > Th \quad (1)$$

Unde: *frame* este imaginea curenta *i*
backgnd este modelul fundalului la momentul current
Th – valoare de prag aleasa arbitrar

Operatia respectiva se poate aplica atat pe imagini color (multichannel) sau grayscale (un singur canal). Pentru simplitate in lucrarea de fata se va lucre cu imagini grayscale (un canal) si ecuatia de mai sus se poate scrie:

$$diff = |gray_i(x,y) - backgnd_i(x,y)| > Th \quad (2)$$

Unde: $gray_i$ este imaginea curenta grayscale obyinuta prin urmatoarea conversie:

```
cvtColor(frame, gray, CV_BGR2GRAY);
```

Aceste imagini se pot declara in OpenCv de tipul *Mat*:

```
Mat frame, gray, backgnd, diff;
```

iar diferenta in modul dintre imaginea curenta *gray* si modelul *backgnd* se poate calcula folosind functia OpenCV *absdiff* cu rezultatul plasat in *diff*:

```
absdiff(gray, backgnd, diff);
```

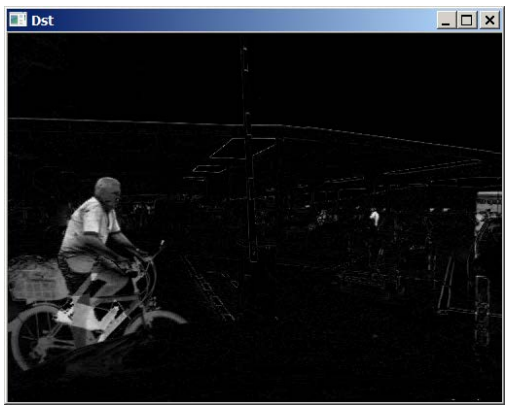


Fig. 6.1. Exemplu cu rezultatul imaginii diferenta *diff*

Pixelii care satisfac conditia din ec. (2) se pot eticheta cu o culoare specifica (de ex. 255 – white). Pentru aceasta trebuie parcursa imaginea diferenta *diff* si testata conditia din ec. (2) la nivel de pixel. Pentru o imagine cu 1 canal de culoare (indexata/grayscale) aceasta conditie se poate testa prin urmatoarea secventa:

```
if (diff.at<uchar>(i,j) > Th)
       dst.at<uchar>(i,j) = 255;
```

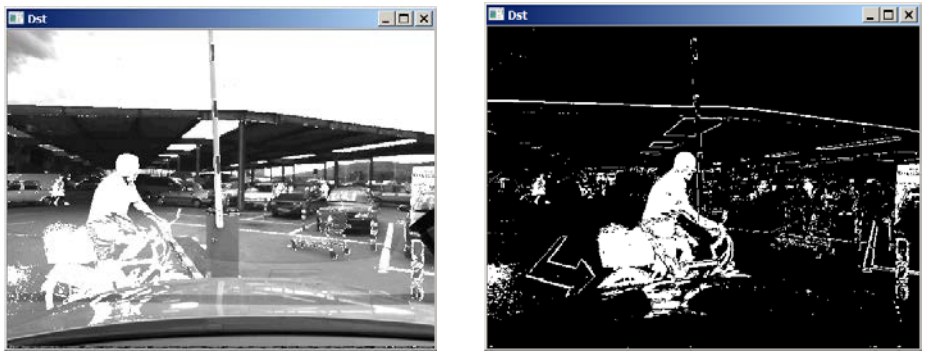
Unde: *dst* este imaginea de iesire pt. vizualizarea rezultatului.

Imaginea de iesire trebuie initializata la nivelul fiecarui cadru achizitionat!. Daca se doreste suprapunerea pixelilor obiect etichetati/colorati cu 255 (white) peste imaginea originala (grayscale) atunci initializarea ei se poate face astfel:

```
dst=gray.clone();
```

Daca se doreste ca imaginea fundal peste care se afiseaza pixelilor obiect sa fie neagra atunci initializarea se poate face in felul urmator:

```
dst = Mat::zeros( gray.size(), gray.type() );
```



a. b.

Fig. 6.2. Vizualizarea pixelilor etichetati ca si obiect (alb): a – peste imaginea sursa grayscale; b- peste un fundal negru;

6.3. Modelarea pixelilor de fundal (background)

Aceasta modelare se va face in functie de algoritmul de Background Subtraction (BS) ales pentru implementare.

6.3.1. Diferenta dintre cadre

In acest caz $backgnd_{i+1} = gray_i$ (cadrul grayscale current (in momentul *i*) va fi model al fundalului in cadrul (momentul *i+1*). Aceasta se poate face in bucla de achizitie/prelucrare a imaginilor/cadrelor prin apelul functiei *gray.clone()* plasat dupa calculul diferentei absolute dintre cadrul current si modelul current, ca si in secventa de mai jos:

```
absdiff(gray, backgnd, diff);
backgnd = gray.clone();
```

6.3.2. Modelarea fundalului prin mediere ponderata (running average)

In acest caz $backgnd_{i+1}$ (modelul fundalului in momentul *i+1*) se va calcula prin medierea ponderata a imaginii grayscale curente si modelul fundalului de la momentul prezent:

$$backgnd_{i+1}(x,y) = \alpha * gray_i(x,y) + (1-\alpha)backgnd_i(x,y) \quad (3)$$

Unde α este un coefficient de ponderare sub-unitar ($\alpha << 1-\alpha$, cu valori tipice de 0.01 .. 0.05).

Ecuatia (3) se poate implementa prin apelul functiei OpenCV *addWeighted* plasat in bucla de achizitie/prelucrare a imaginilor/cadrelor dupa calculul diferentei absolute dintre cadrul current si modelul current ca in secventa de mai jos:

```
absdiff(gray, backgnd, diff);
addWeighted(gray, alpha, backgnd, 1.0-alpha, 0, backgnd);
```

6.3.2. Modelarea fundalului prin mediere ponderata selectiva (selective running average)

Diferenta fata de 5.4.2 este ca actualizarea modelului pentru pixeli de fundal se va face selectiv:

- daca pixelul current este clasificat ca si obiect, nu se schimba modelul fundalului pt. pixelul respectiv
- daca pixelul current este clasificat ca si fundal atunci el contribuie la actualizarea modelului fundalului

```
// current pixel is forground (object) -> color in white
if (diff.at<uchar>(i,j) > Th)
	dst.at<uchar>(i,j) = 255; // current pixel is forground (object) -> color in white
	// selective running average -> no change to the background pixel model value
	// backgnd.at<uchar>(i,j) = backgnd.at<uchar>(i,j);
else // current pixel is background -> update background model for the current pixel
	backgnd.at<uchar>(i,j) =
			alpha*gray.at<uchar>(i,j) + (1.0-alpha)*backgnd.at<uchar>(i,j);
```

6.4. Mersul lucrarii

1. Pentru implementarea metodelor se va crea o functie de procesare noua in OpenCV Application. Fisierul video se va citi de pe disk (vezi exemplul existent `testVideoSequence()`). La rulare avansarea in fisierul video se face frame cu frame prin apasarea oricarei taste. Tasta ESC permite oprirea vizualizarii.

2. Se vor face urmatoarele initializari:

```
Mat frame, gray; //current frame: original and gray
Mat backgnd;   // background model
Mat diff;             //difference image: |frame_gray - bacgnd|
Mat dst;              //output image/frame
char c;
int frameNum = -1;    //current frame counter

const int method = 0;
// method =
//                    1 - frame difference
//                    2 - running average
//                    3 - running average with selectivity
const unsigned char Th = 15;
const double alpha = 0.05;
```

3. Se va implementa metoda de segmentarea pixelilor de fundal (6.2) folosind cele trei variante expuse pentru modelarea fundalului (6.3.1 … 6.3.3).

Observatie: Operatiile aferente segmentarii nu se vor aplica asupra primului cadru din secventa, ci doar pentru urmatoarele. In primul cadru se recomanda initializarea fundalului cu cadrul grayscale current, dupa sablonul de mai jos:

```
for(;;){
        cap >> frame; // achizitie frame nou
        if( frame.empty()  )
         {
                printf("End of video file\n"));
                break;
         }

        ++frameNum;
        if (frameNum == 0)
                imshow( “sursa”, frame);     // daca este primul cadru se afiseaza doar sursa

        cvtColor(frame, gray, CV_BGR2GRAY);
        //Optional puteti aplica si un FTJ Gaussian

        //Se initializeaza matricea / imaginea destinatie pentru fiecare frame
        //      dst=gray.clone();
        // sau
        //
        //      dst = Mat::zeros( gray.size(), gray.type() );

        const int channels_gray = gray.channels();
        //restrictionam utilizarea metodei doar pt. imagini grayscale cu un canal (8 bit / pixel)
        if (channels_gray > 1)
                return;

        if (frameNum > 0 ) // daca nu este primul cadru
        {

                //------ SABLON DE PRELUCRARI PT. METODELE BACKGROUND SUBTRACTION -------
                // Calcul imagine diferenta dintre cadrul current (gray) si fundal (backgnd)
                // Rezultatul se pune in matricea/imaginea diff
                // Se actualizeaza matricea/imaginea model a fundalului (backgnd)
                // conform celor 3 metode:
                // met 1: backgnd = gray.clone();
                // met 2: addWeighted(gray, alpha, backgnd, 1.0-alpha, 0, backgnd);

                // Binarizarea matricii diferenta (pt. toate metodele):
                //  Se parcurge sistematic matricea diff
```

```
                        //daca valoarea pt. pixelul current diff.at<uchar>(i,j) > Th
                                // marcheaza  pixelul din imaginea destinatie ca obiect:
                                dst.at<uchar>(i,j) = 255 // pentru toate metodele
                        // altfel
                                // actualizeaza model background (doar pt. met 3), pt. fiecare pixel
                //-------------------------------------------------------------------

                // Afiseaza imaginea sursa si destinatie
                imshow( "sursa", frame);     // show source
                imshow("dest", dst);         // show destination

                // Plasati aici codul pt. vizualizarea oricaror rezultate intermediare
                // Ex: afisarea intr-o fereastra noua a imaginii diff
        }
        else    // daca este primul cadru, modelul de fundal este chiar el
                backgnd = gray.clone();

        // Conditia de avansare/terminare in cilului for(;;) de procesare
        c = cvWaitKey(0);  // press any key to advance between frames
        //for continous play use cvWaitKey( delay > 0)
        if (c == 27) {
                // press ESC to exit
                printf("ESC pressed - playback finished\n"));
                break;  //ESC pressed
                }
}
```

4. Se vor vizualiza in imaginea de iesire (*dst*) atat rezultatele intermediare (imaginea *diff* – fig. 1) cat si rezultatul final al segmentarii (fig. 2. a sau b) (img. *dst*). Pentru teste se vor putea folosi filmele din fisierele *BS.avi*, *taxi.avi*, *walkcircle.avi*, *walkstraight.avi*, *campus.avi*, *laboratory.avi*). Puteti sa le descarcati de aici: http://users.utcluj.ro/~tmarita/HCI/Media/Video/ si le copiati in folderul *Videos* din *OpenCVApplication*.

5. Pe rezultatul final al segmentarii (fig 2.b – pixelii de fundal colorati in negru) se pot aplica operatii morfologice (pixelii de obiect in operatiile morfologice din OpenCV sunt considerati cei albi), pentru eliminarea de zgomote (pixeli singulari).
Exemplu de cod care realizeaza 2 eroziuni succesive urmate de 2 dilatari succesive cu un element structural de tip cruce de dimensiune 3x3 cu elemntul de referinta central:

```
Mat element = getStructuringElement( MORPH_CROSS, Size( 3, 3 ) );
erode ( dst, dst, element, Point(-1,-1), 2 );
dilate( dst, dst, element, Point(-1,-1), 2 );
```

6. Pt. fiecare cadru/frame se va masura timpul de procesare [ms] si se va afisa la linia de comanda (vezi Anexa 2)

Bibliografie

[1] http://docs.opencv.org/index.html
[2] The OpenCV Tutorials, Release (`doc\opencv_tutorials.pdf`), 2.2. How to scan images, lookup tables and time measurement with OpenCV.
http://docs.opencv.org/doc/tutorials/core/how_to_scan_images/how_to_scan_images.html#howtoscanimagesopencv, `doc\opencv_tutorials.pdf`
[3] OpenCV_Tutorials (`doc\opencv_tutorials.pdf`), 3.2 Eroding and Dilating, 3.3 More Morphology Transformations:
http://docs.opencv.org/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html#morphology-1

ANEXA 1

Metoda de acces la pixellii individuali dintr-o imagine in OpenCV (mai accesibila de inteles dar nu neaparat cea mai rapida) folosita intr-un exemplu care implementeaza negativul unei imagini. Pentru accesul cel mai rapid din punctul de vedere al timpului de parcurgere al imaginii consultati si celelalte exemple prezentate in [2].

```
Mat src, dst; // source and destination images
src = imread(opened_file_path , CV_LOAD_IMAGE_UNCHANGED);     // Read the file
/*-------------------------------------------------------------------------------
The second argument of imread specifies the format in what we want the image. This
may be:
CV_LOAD_IMAGE_UNCHANGED (<0) loads the image as is
CV_LOAD_IMAGE_GRAYSCALE ( 0) loads the image as an intensity one
CV_LOAD_IMAGE_COLOR (>0) loads the image in the RGB format
-------------------------------------------------------------------------------*/
dst=src.clone(); // copy source image in destination
// accept only char type matrices
CV_Assert(src.depth() != sizeof(uchar));
const int channels = src.channels();
switch(channels)
	{
	case 1:  //  8bit image (1 channel image)
		{
		for( int i = 0; i < src.rows; ++i)
			for( int j = 0; j < src.cols; ++j )
				//performs image negative
				dst.at<uchar>(i,j) = 255 - src.at<uchar>(i,j);
	break;
		}
	case 3:  // 24 bit (3 channels image)
		{
		Mat_<Vec3b> _src=src;
		Mat_<Vec3b> _dst=dst;
		for( int i = 0; i < src.rows; ++i)
			for( int j = 0; j < src.cols; ++j )
			{
				_dst(i,j)[0] = 255 - _src(i,j)[0];
				_dst(i,j)[1] = 255 - _src(i,j)[1];
				_dst(i,j)[2] = 255 - _src(i,j)[2];
			}
		dst = _dst;
		break;
		}
	}
```

ANEXA 2

Masurarea timpului de executie - „2.9 Utility and System Functions and Macros” (doc\opencv_tutorials.pdf):

```
double t = (double)getTickCount(); // Get the current time [s]

// . . . insert here the processing functions / code

// Get the current time again and compute the time difference [s]
t = ((double)getTickCount() - t) / getTickFrequency();
// Print (in the console window) the processing time in [ms]
printf("%d - %.3f [ms]\n", frameNum, t*1000);
```