Median-based background subtraction

1. Introduction

A median-based approach to background subtraction was described in Davies (2017), Chapter 22, Section 22.3. The basis of the approach was to model the background by averaging videos over many frames, following which subtraction of the background from the current image would ideally lead to segmentation of any moving objects. However, using a temporal mean filter for averaging would result in the background containing blurred images of moving objects, so subtraction could not result in clear segmentation. On the other hand, using a temporal median filter would give the chance of eliminating outliers including all moving objects, following which background subtraction should result in greatly improved segmentation. Davies (2017) also indicated that some sort of 'restrained' temporal median filter might provide even better segmentation performance.

In what follows we start by making a careful examination of the temporal median filter approach. In fact, it is difficult to design an ideal temporal median filter because averaging would have to take place over a past version of the whole video. Hence the algorithm presented below uses a running temporal median ('rmedian'), acting in a similar way to a running average (for which the latest output is typically one tenth of the latest input sample plus nine tenths of the previous output). The core of the running median is given by the following piece of code:

```
% running median
delta=3;
for y=1:iy
  for x=1:ix
    inten=t0(y,x); % intensity of current pixel
    if nn==1, rmedian=inten; else, rmedian=t1(y,x); end
    if rmedian>inten, rmedian=rmedian-delta;
    elseif rmedian<inten, rmedian=rmedian+delta;
    end
    t1(y,x)=rmedian;
  end
end
```

Thus, the basic method is to increase the running median by a small fixed amount \( \delta \) if it is less than the current intensity, and to decrease it by \( \delta \) if it is more than the current intensity. Clearly, a running median of this type will always move towards the ideal median, though it will eventually tend to oscillate about the true position. There is therefore tension between making \( \delta \) large enough to allow strong learning and making it small enough to keep the final oscillations below the image noise level. In fact, such oscillations are seldom observed in practice.

The first part of the algorithm is presented below. It includes various parameter definitions, details of how the video frames are input, and then a section showing the temporal median computation. In fact, the latter is more complete than the running median code given above, as it includes three subsections: the first is a running mean average, which acts as a stable first approximation for the first few images; the second is the running median average, which runs for rather longer; and the third is the 'restricted' running median, which does almost the same as the running median, but
ignores signals that are more than a set distance $devn$ away from the current median approximation, because they are deemed to be outliers that are not only irrelevant but also misleading for updating the current median. Notice that the last few lines of code show the absolute deviation between $rmedian$ and the current intensity being computed and thresholded to indicate whether the current pixel is likely to be an inlier (background) or an outlier (a moving object)—see Figure 1(a)–(c).

```matlab
% program to find restricted median background
% t1=zeros(240,320); dt1=double(t1); % initialise rmedian image
% t2=uint8(zeros(240,320)); t3=uint8(zeros(240,320));
for nn=1:10:1128
    strn=num2str(nn,'%d');
    str='bgsubtraction';
    str0=strcat('roadvideo\Frame',strn,'.bmp');
    str1=strcat('roadvideo\Frame',strn,str,'.bmp');
    frame=imread(str0);
    r1=frame(:,:,1); g1=frame(:,:,2); b1=frame(:,:,3);
    t0=rgb2gray(frame);
    [iy,ix]=size(t0);
    dt0=double(t0);
    % average over n frames
    M=3;
    delta=3;
    devn=90;
    thr=10;
    for y=1:iy
        for x=1:ix
            inten=dt0(y,x); % intensity of current pixel
            if nn==1, rmedian=inten; else, rmedian=dt1(y,x); end
            if nn<50 % mean
                rmedian=(inten+M*rmedian)/(M+1);
            elseif nn<100 % median
                if rmedian>inten, rmedian=rmedian-delta; elseif rmedian<inten, rmedian=rmedian+delta; end
            else % (nn>=100) restricted median
                if rmedian>inten && rmedian-inten<devn
                    rmedian=rmedian-delta;
                elseif rmedian<inten && inten-rmedian<devn
                    rmedian=rmedian+delta;
                end
            end
            dt1(y,x)=rmedian;
            inten2=abs(inten-rmedian);
            if inten2>thr, inten2=255; else, inten2=0; end
            t2(y,x)=uint8(inten2);
        end
    end
    % finally, convert rmedian output in dt1 to uint8 format
    dt1n=(dt1-min(dt1(:)))/(max(dt1(:))-min(dt1(:))); % normalise dt1
    t1=im2uint8(dt1n); % the uint8 function won't do this properly
```

We now move on to analysis of the moving object indicator information. Figure 1(c) shows that the indicator is very noisy. Clearly, it is necessary to eliminate this problem as far as possible. Here, this is attempted by using morphological analysis—specifically by applying a 2-pixel erosion operation followed by a 4-pixel...
dilation operation. In order to make these operations as isotropic as possible, they are applied in approximately circular regions of radius $\sqrt{5} \approx 2.24$ and $\sqrt{18} \approx 4.24$ respectively (in fact, the actual shapes are closer to octagons than to circles). Basically, a 2-pixel erosion eliminates the majority of the noise points; then a 2-pixel dilation restores the remaining moving objects to their previous sizes. However, a further 2-pixel dilation is applied so that a small amount of reconnection of the moving objects can occur. These aspects are well illustrated in Figure 1(c)–(e).

![Figure 1](https://example.com/figure1.png)

Figure 1 Stages in segmenting the moving objects. (a) Current frame. (b) Running median. (c) Result of background subtraction. (d) Result of 2-pixel erosion. (e) Result of 4-pixel dilation. (f) Output mage, with green and mauve graphics boundaries. In (b), notice that the large car appears only as a highly attenuated 'ghost' after just four applications of the running median.
Finally, edge detection is applied in image space t2, so that moving objects can be highlighted with a green border. In addition, the roadway region is shown approximately bounded by two mauve lines. At the same time, moving objects outside the bounded region are eliminated, as they are most likely to be due to the motions of leaves or other vegetation. However, it is abundantly clear from Figure 1(c) and (d) that most of the vegetation has already been eliminated by the morphological erosion operation. At this stage, the majority of the extraneous objects have been removed and the algorithm is seen to be reasonably effective at locating moving vehicles: the two remaining problems are that many of the vehicles contain gaps within their outlines: in addition, in many cases their shadows accompany them—see Figure 1(f) and Figure 2(a) and (b). Evidently, a higher degree of intelligence is required to eliminate these problems. In principle, this can be tackled by (a) model-based shape analysis, and (b) using inter-frame processing to ensure consistency.
Figure 2 Segmented vehicles. (a) A bus and a car being tracked: the car outline also contains the shadow. (b) A bus being tracked, leaving very little 'ghosting' behind it: again, the shadow appears within the object boundary.

```matlab
% display results
winsize=3;
border=(winsize-1)/2;
for y=1+border:iy-border
    for x=1+border:ix-border
        P4=t2(y-1,x-1);P3=t2(y-1,x);P2=t2(y-1,x+1);
P5=t2(y ,x-1);P0=t2(y ,x );P1=t2(y ,x+1);
P6=t2(y+1,x-1);P7=t2(y+1,x);P8=t2(y+1,x+1);
        intenr=t0(y,x); inteng=t0(y,x); intenb=t0(y,x);

        % put border around objects
        if (P0>0) && ((P1==0)||(P2==0)||(P3==0)||(P4==0)||(P5==0)||(P6==0)||(P7==0)||(P8==0))
            if y-202+x>=0 && y+32+ix-2*x>=0
                intenr=0; inteng=255; intenb=0; % green
            end
        end

        % insert two mauve lines to demarcate roadway region
        if abs(y-202+x)<1 && y>=18 && rem(y,2)==0
            intenr=255; inteng=0; intenb=255; % mauve
        end

        if abs(y+32+ix-2*x)<1 && y>=18 && rem(y,2)==0
            intenr=255; inteng=0; intenb=255; % mauve
        end

        r1(y,x)=intenr; g1(y,x)=inteng; b1(y,x)=intenb;
    end
end
frameout(:,:,1)=r1; frameout(:,:,2)=g1; frameout(:,:,3)=b1;
imwrite(frameout,str1);
end % nn
```

Access to the images used for testing the above algorithm

The 'roadvideo' frames used for generating Figures 1 and 2 appear as bmp files in the Image data section of the Davies (2017) website, in the form Frame1–Frame1121.