Theory:

The MOSSE algorithm [16] introduced the correlation filter technology into the visual tracking field. This kind of algorithm can adapt to the problems of occlusion and rotation and achieve an amazing tracking speed of 669 fps. Running 26 times faster than the advanced MIL algorithm, the MOSSE filter is trained by the first frame and can have strong robust performance for illumination, scale and posture variation. When the target is occluded, the algorithm can determine the status of object tracking and update the filter parameters according to the PSR value. When the object reappears, it can be tracked again.

In the MOSSE algorithm, to create a fast tracker, the fast Fourier transform (FFT) is used to calculate the correlation in the Fourier domain. First, calculating the 2D Fourier transform of the input image (*F = F(f)*) and filter (*H = F(h)*). The convolution theorem states that correlation is the element multiplication in the Fourier domain. The symbol $\odot$ represents element-by-element multiplication, * indicates complex conjugate and the representation of correlation is as follows:

$$g = f \otimes h$$

where *g*, *f* and *h* represent response output, input image and filter template, respectively. It can be seen that we only need to determine the filter template *h* to get the response output. The fast Fourier transform (FFT) is used in above equation. Therefore, the convolution operation becomes a point multiplication operation, which greatly reduces the amount of calculation. That is, the above formula becomes:

$$F(g) = F(f \otimes h) = F(f) \cdot F(h)^*$$

Then, the above formula is abbreviated as follows:

$$G = F \cdot H^*$$

and the next task to track is to find the filter template *H*:*

$$H^* = \frac{G}{F}.$$

In the process of actual tracking, we need to consider the influence of factors such as the appearance of the object. At the same time, considering the m images of the object as a reference can significantly improve the robustness of the filter template. The MOSSE model formula is as follows:

$$\min_{H^*} = \sum_{i=1}^{m} | H^* F_i - G_i |^2$$

after a series of transformations, a closed solution is obtained:

$$\boldsymbol{H^* = \frac{\sum_i G_i \cdot F_i^*}{\sum_i F_i \cdot F_i^*}}$$

The algorithm tracks the object by correlating filters on the search window in the next frame. The new position of the object is represented by the maximum value of the associated output. Then performs an online update in the new location. The tracker update method uses the following formula:

$$H^* = \frac{A_i}{B_i}$$

Where A_i and B_i are:

$$A_i = \eta G_i \odot F_i + (1 - \eta) A_{i-1}$$

$$B_i = \eta F_i \odot F_i^* + B_{i-1}$$

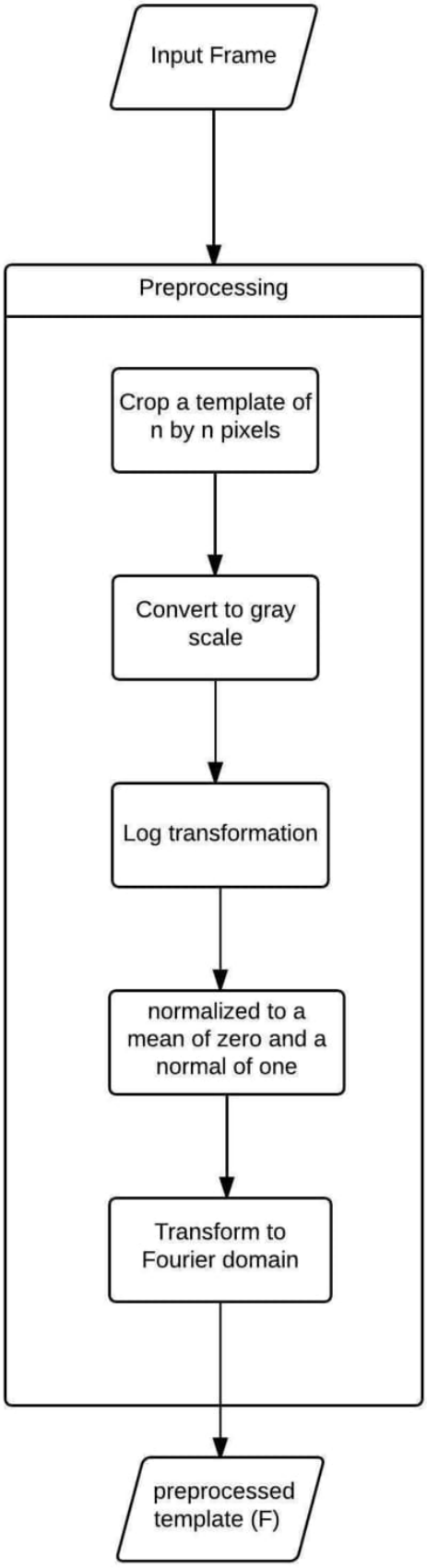
Finally, the PSR value is used for failure detection:

$$PSR = \frac{Peak - mean}{std}$$

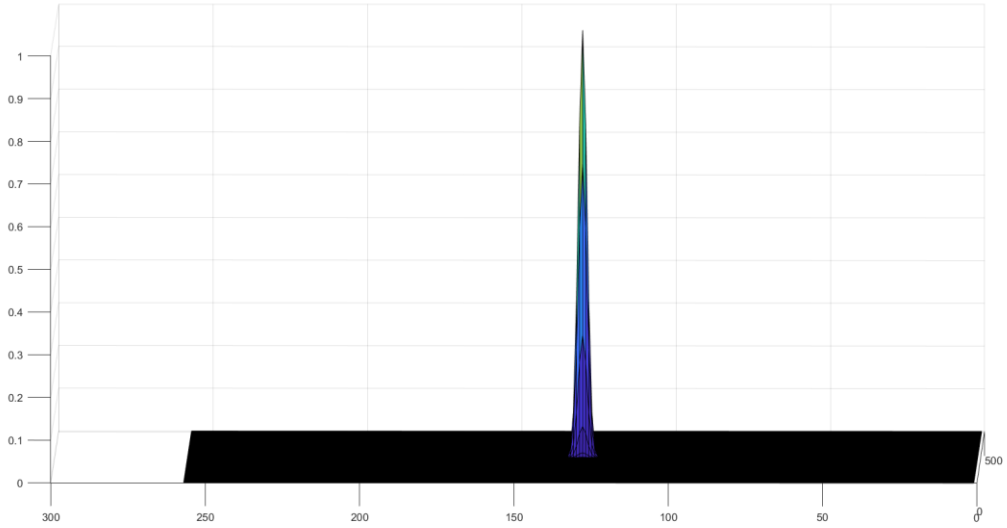
In the experiment, the PSR value between 20 and 60 is considered to be a good tracking effect. When the PSR value is lower than 7, it is judged as tracking failure and the template is not updated.

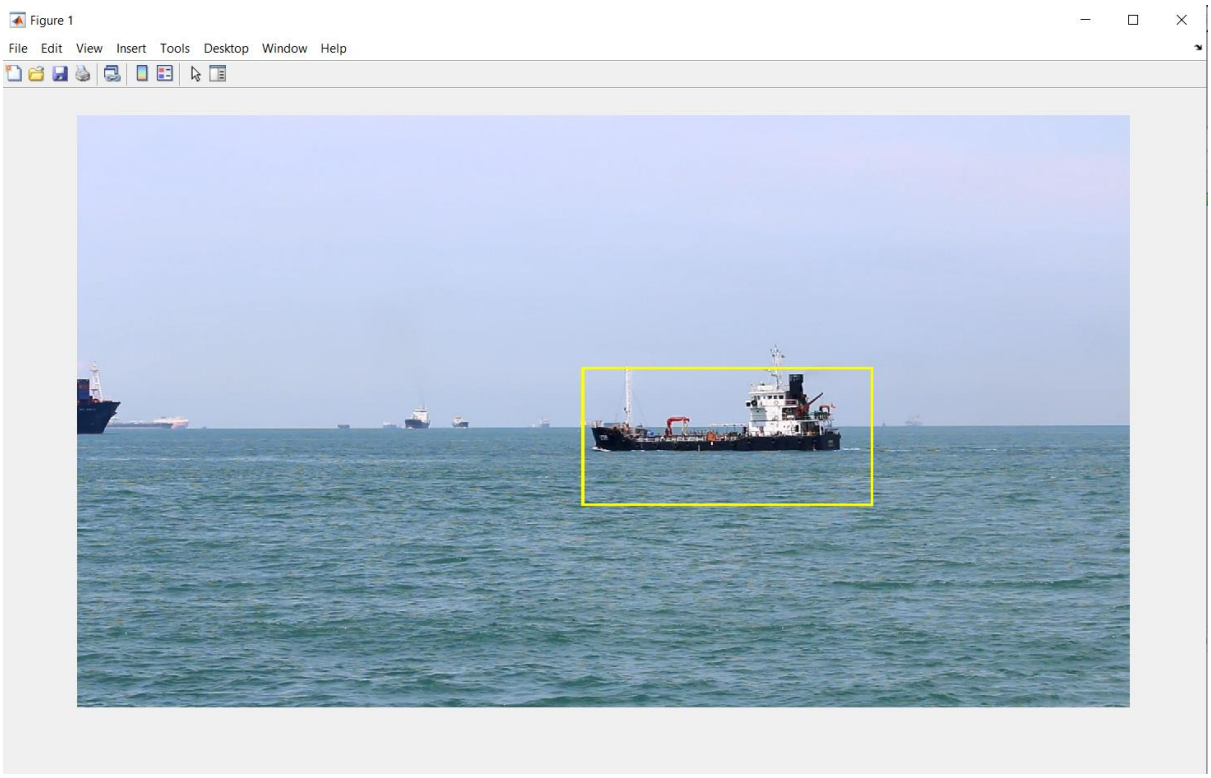
The MOSSE algorithm overall can adapt to small-scale variation, but it cannot adapt to large-scale variation. In addition, the MOSSE algorithm uses grayscale features that are not powerful enough and expressive in general. The sample sampling of the MOSSE algorithm is still a sparse sampling, and the training effect is general.

FLOWCHART


Input Frame
Preprocessing
Crop a template of n by n pixels
Convert to gray scale
Log transformation
normalized to a mean of zero and a normal of one
Transform to Fourier domain
preprocessed template (F)

Result:





Conclusion:

- We implemented successfully the MOSSE correlation filter using MATLAB.
- We tested it on ship video.
- We first convert the videos to frames, then apply MOSSE correlation filtering.

APPENDIX

MOSSE CORRELATION FILTER:


```
clc;
close all;
clear;
path='imgs';
```

Video frames to pictures:(Only one time run this portion)

```
% vid=VideoReader('ship.avi');
%   numFrames = vid.NumberOfFrames;
%   n=numFrames;
% for i = 1:1:n
%   frames = read(vid,i);
%   imwrite(frames,['C:\Users\eranu\OneDrive - Indian Institute of Technology
Bhubaneswar\Documents\MATLAB\ADSP LAB\Exp4\MOSSE Final\imgs\imgs' int2str(i), '.tif']);
% end
```

Reading Images

```
filenames=dir(fullfile(path,'*.tif'));
noi=numel(filenames);   %number of images
N=noi;     %No. of images
f=fullfile(path, filenames(1).name);
im = imread(f);
f = figure('Name', 'Select object to track'); imshow(im);
template = getrect;
close(f); clear f;
c1=template(2)+template(4)/2;
c2=template(1)+template(3)/2;
```

Preprocessing Step1: Convert template RGB to Gray

```
template_img = rgb2gray(im);
template_img = imcrop(template_img, template);
```

```
template_img=  mat2gray(template_img);
imshow(template_img);
```

Preprocessing Step2: Apply Log transformation

```
a = double(template_img)/255; %Normalized Image for log transformation
c = 300; % Constant
temp_log = c*log(1 + (a)); % Log Transform
imshow((temp_log)),title('Log Transformation Image');
```

Preprocessing Step3: Normalize to zero mean and Unit variance

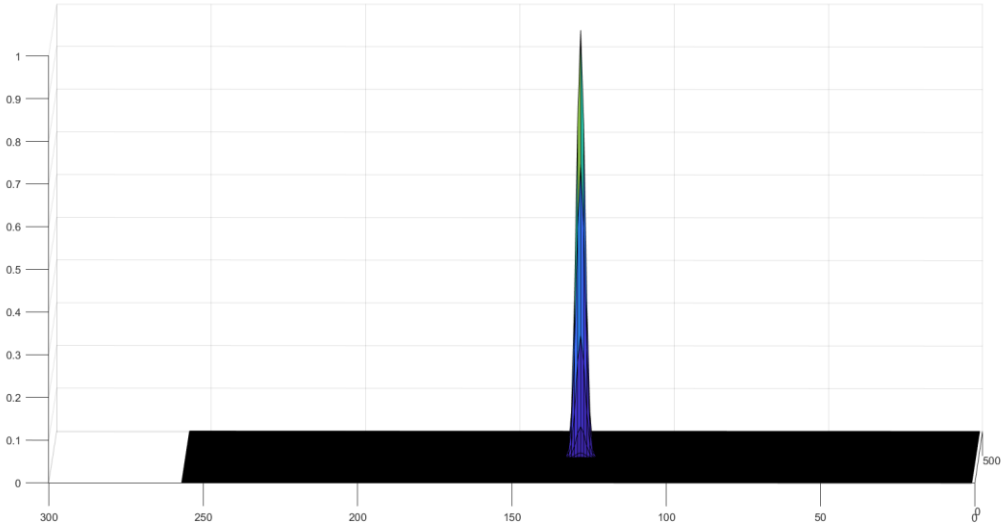
```
temp_norm = (temp_log - mean(temp_log, 'all')) ./ std2(temp_log);
% mean(temp_norm,'all')
% std2(temp_norm)
% imshow((temp_norm))
```

Preprocessing Step4: Transform to Fourier Domain

```
Fi=fft2(temp_norm);
%imshow((abs(F)))
```

synthetically generate the synthetic target image

```
sigma = 2;
g_size = size(im);      %Size of first frame
[xx,yy] = ndgrid(1:g_size(1), 1:g_size(2));
g=(exp(-((xx-c1).^2 + (yy-c2).^2)./(2*sigma)));
g = mat2gray(g);
surf(g);
g = imcrop(g, template); %
G = fft2(g);
height = size(g,1);
width = size(g,2);
```



MOSSE filter Initialization

```
epsilon=0.00001;   % To avoid zero by zero form
Ai = (G.*conj(Fi));
Bi = (Fi.*conj(Fi))+epsilon;

eta = 0.125;
for i=1:N


    f=fullfile(path, filenames(i).name);
    im = imread(f);
    if (size(im,3) == 3) %if RGB
        img = rgb2gray(im); %then convert to grayscale
    end
    if (i == 1) %For the first image
        Ai = eta.*Ai;
        Bi = eta.*Bi;
    else
        Hi = Ai./Bi; % for second picture onwards, solve the filter H*
        %Preprocessing%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        fi = imcrop(img, template);
        fi=imresize(fi, [height width]);
        %Log transformation
```

```
        a = double(fi)/255; %Normalized Image for log transformation
        c = 300; % Constant
        temp_log = c*log(1 + (a)); % Log Transform
        %end of log transform
        %Normalize to zero mean and unit variance
        temp_norm = (temp_log - mean(temp_log,'all')) ./ std2(temp_log);
        Fi=fft2(temp_norm);
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        gi = uint8(255*mat2gray(ifft2(Hi.*Fi)));
        max_g = max(gi(:));
        [xxx, yyy] = find(gi == max_g); % Max g location
        dx = mean(xxx)-height/2;
        dy = mean(yyy)-width/2;
        template = [template(1)+dy template(2)+dx width height]; %update coordinates
        %Preprocessing%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        fi = imcrop(img, template);
        fi=imresize(fi, [height width]);
        %Log transformation
        a = double(fi)/255; % Normalized Image for Log transformation
        c = 300; % Constant
        temp_log = c*log(1 + (a)); % Log Transform
        % end of Log transformation
        %Normalize to zero mean and unit variance
        temp_norm = (temp_log - mean(temp_log,'all')) ./ std2(temp_log);
        Fi=fft2(temp_norm);
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %Last updated Ai and Bi
        Ai = eta.*(G.*conj(Fi)) + (1-eta).*Ai;
        Bi = eta.*(Fi.*conj(Fi)) + (1-eta).*Bi;
    end
    % Preview images after each iteration
    result = insertShape(im, 'Rectangle', template,'LineWidth',5);
    imshow(result);
end
```

Published with MATLAB® R2020b

References

1. Bolme, David S., et al. "Visual object tracking using adaptive correlation filters." 2010 IEEE computer society conference on computer vision and pattern recognition. IEEE, 2010.
2. Liu, S., Liu, D., Srivastava, G. et al. Overview and methods of correlation filter algorithms in object tracking. Complex Intell. Syst. (2020). https://doi.org/10.1007/s40747-020-00161-4.
3. Sidhu, Rumpal Kaur. Tutorial on minimum output sum of squared error filter. Diss. Colorado State University, 2016.