Project Report

Stop sign and Pedestrian Crossing sign recognition using Template Matching and Image Pyramid

Computer Vision (CSE 5524)

Jim Davis

Fall 2021

Shree Sai Charan Nannapaneni

The Ohio State University

Columbus, OH

1 December 2021

Abstract

Find if the given input image has a STOP sign or Pedestrian Crossing sign in it and detect where it is and what sign it is. It is done using Template Matching and Image Pyramid. Using all the images in RGB format gave acceptable results i.e., not accurate all the time, whereas converting all the images to HSV format gave right results almost all the time.

Introduction

The goal of this project is to detect if there is a STOP sign or a Pedestrian Crossing sign in an image. Two traditional Computer Vision techniques called Template Matching and Image Pyramid are used to do this. An example of STOP sign and Pedestrian Crossing sign templates is shown in Fig 1.

Fig 1. Pedestrian Crossing sign (left) and STOP sign (right)

Normalized Cross Correlation (NCC) Template Matching method is used to handle any illumination changes and a 4-level Gaussian Image Pyramid is used to take care of any scaling issues. A 4-level Gaussian Image Pyramid is formed by smoothing an image then down sampling it four times.

Got the data mostly from Google Images and found a small dataset which can be used as input images from GitHub. Link - https://github.com/mbasilyan/Stop-Sign-Detection/tree/master/Stop%20Sign%20Dataset

Methodology

A total of 8 templates are chosen, 4 STOP sign templates and 4 Pedestrian Crossing sign templates each of different sizes. This is done to reduce the running time if the input image is very large. All 8 Templates can be seen in Fig 2.

Fig 2. 8 Templates used in the project

The sizes of each of these templates are around 40x40, 67x67, 100x100, 133x133. Had written the code in such a way that if one of the dimensions of the input image is greater than 1200, choose a STOP sign template of size 140x140 and a Pedestrian Crossing sign template of size 133x133. Similarly, if one dimension of input image is between 900 and 1200, choose template of size 100x100 and if between 600 and 900, chose template of size 67x67 and if one of the dimensions is less than 600, then always choose template of size 40x40.

After choosing the proper templates, convert them to HSV format along with the input image and do Template Matching (Normalized Cross Correlation) of input image (level 0 in Gaussian Pyramid) w.r.t both Pedestrian Crossing template and STOP sign template and get NCC values. Repeat this same process for level-1, level-2, and level-3 images in the Image Pyramid. An example of Image Pyramid can be seen in Fig 3.

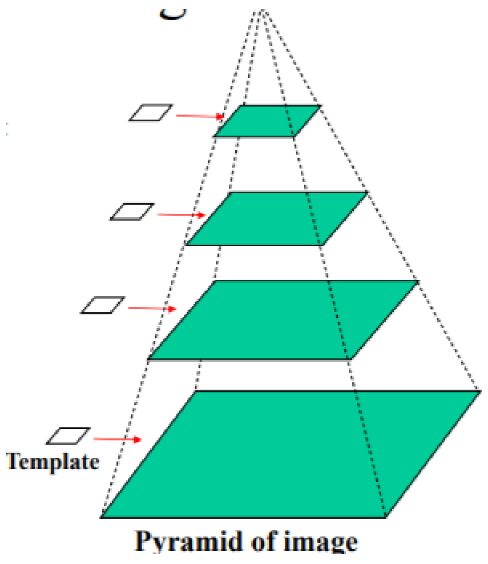

Fig 3. 4-level Gaussian Image Pyramid

After getting all the NCC values, select a patch from one of the levels of Pyramid which has the highest NCC value and draw a red bounding box around it.

Results

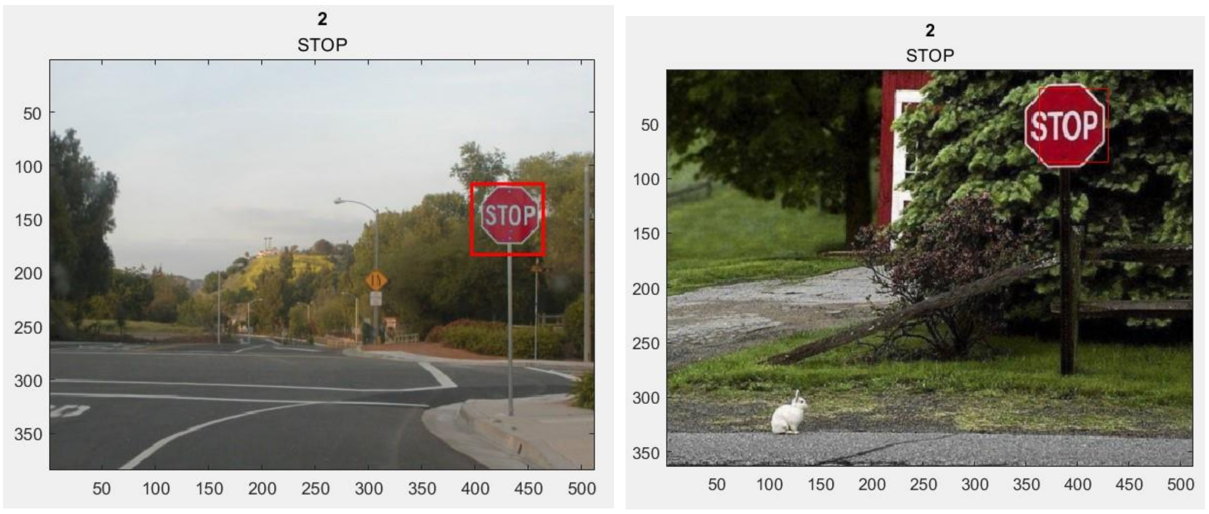

Fig 4. STOP sign detection results

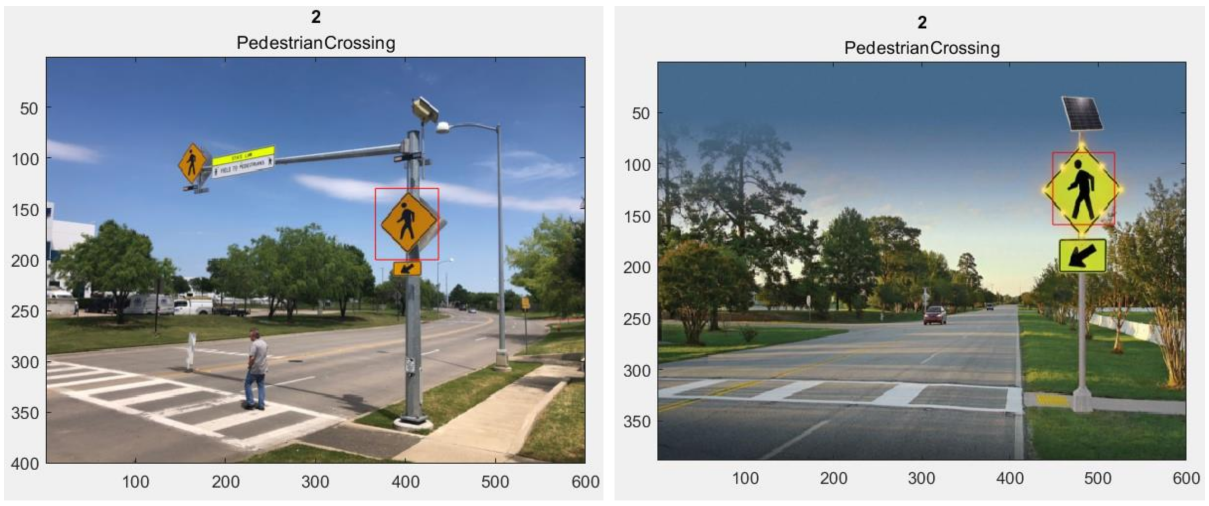

Fig 5. Pedestrian Crossing sign detection results

If the input image is large it takes a lot of time to detect and this process of combining template matching and image pyramid is somewhat like exhaustive search. Tried changing background of the templates to see if it makes any difference and found out that white background templates give higher NCC values (therefore slightly better results) than black background templates.

Conclusion

In the results above, we can observe the STOP signs and Pedestrian Crossing signs being detected properly. These are the results I got after changing the image format from RGB to HSV to use the red colour in STOP sign and yellow colour in Pedestrian Crossing sign to detect them. Other techniques like edge detection (to use unique shape of these signs) and can be used to get better results and Machine Learning can also be used obviously.

Appendix : MATLAB code

```
serch = double(imread('66 .jpg')); %%% Input Image
serchsv= rgb2hsv(serch);

if size(serch,1) > 1200
    temp = double(imread('stoprecent140.png'));  %%% Stop
template of size 140x140
    ptemp = double(imread('pedestriantempu133.png'));
%%% Pedestrian crossing template of size 133x133
    tempsv = rgb2hsv(temp);
    ptempsv = rgb2hsv(ptemp);
elseif size(serch,1) > 950
    temp = double(imread('stoprecent100.png'));  %%% Stop
template of size 100x100
    ptemp = double(imread('pedestriantempu100.png'));
%%% Pedestrian crossing template of size 100x100
    tempsv = rgb2hsv(temp);
    ptempsv = rgb2hsv(ptemp);
elseif size(serch,1) > 600
    ptemp = double(imread('pedestriantempu70.png'));  %%%
Pedestrian crossing template of size 70x70
    temp = double(imread('stoprecent67.png'));  %%% Stop
template of size 67x67
    tempsv = rgb2hsv(temp);
    ptempsv = rgb2hsv(ptemp);
else
    temp = double(imread('stoprecent40.png'));  %%% Stop
template of size 40x40
    ptemp =
double(imread('pedestriancrossingtemp_40x40.png'));  %%%
Pedestrian crossing template of size 40x40
    tempsv = rgb2hsv(temp);
    ptempsv = rgb2hsv(ptemp);
end

onelastfunc(tempsv,ptempsv,serchsv);
%% functions

function onelastfunc(temp,ptemp,serch)
    serch2 = sample(blurr(serch));
    serch3 = sample(blurr(serch2));
    serch4 = sample(blurr(serch3));
    forStop = Pyramid(temp,serch,serch2,serch3,serch4);
    forPedestrian =
Pyramid(ptemp,serch,serch2,serch3,serch4);
    if max(forStop(:,1)) > max(forPedestrian(:,1))
```

```
         if forStop(1,1) == max(forStop(:,1))
             lastfunc( forStop(1,3), forStop(1,2) ,serch,
'1', 'STOP', size(temp,1), size(temp,2));
         elseif forStop(2,1) == max(forStop(:,1))
             lastfunc( forStop(2,3), forStop(2,2)
,serch2, '2', 'STOP', size(temp,1), size(temp,2));
         elseif forStop(3,1) == max(forStop(:,1))
             lastfunc( forStop(3,3), forStop(3,2)
,serch3, '3', 'STOP', size(temp,1), size(temp,2));
         elseif forStop(4,1) == max(forStop(:,1))
             lastfunc( forStop(4,3), forStop(4,2)
,serch4, '4', 'STOP', size(temp,1), size(temp,2));
         end
    else
        if forPedestrian(1,1) == max(forPedestrian(:,1))
            lastfunc( forPedestrian(1,3),
forPedestrian(1,2) ,serch, '1', 'PedestrianCrossing',
size(ptemp,1), size(ptemp,2));
        elseif forPedestrian(2,1) ==
max(forPedestrian(:,1))
            lastfunc( forPedestrian(2,3),
forPedestrian(2,2) ,serch2, '2', 'PedestrianCrossing',
size(ptemp,1), size(ptemp,2));
        elseif forPedestrian(3,1) ==
max(forPedestrian(:,1))
            lastfunc( forPedestrian(3,3),
forPedestrian(3,2) ,serch3, '3', 'PedestrianCrossing',
size(ptemp,1), size(ptemp,2));
        elseif forPedestrian(4,1) ==
max(forPedestrian(:,1))
            lastfunc( forPedestrian(4,3),
forPedestrian(4,2) ,serch4, '4', 'PedestrianCrossing',
size(ptemp,1), size(ptemp,2));
        end
    end
end

function lastfunc(x,y,serchh,title1,title2,temsz1,temsz2)
    imagesc(uint8(hsv2rgb(serchh)));
    title(title1,title2);
    hold on
    rectangle('Position', [ x  y temsz2 temsz1 ] ,
'EdgeColor','r',  'LineWidth',2);
    hold off
end

function matri = Pyramid(temp,serch,serch2,serch3,serch4)
```

```
    NCC1 = templateMatching(temp,serch);
    NCC2 = templateMatching(temp,serch2);
    NCC3 = templateMatching(temp,serch3);
    NCC4 = templateMatching(temp,serch4);
    matri(1,1) = max(NCC1,[],'all');
    matri(2,1) = max(NCC2,[],'all');
    matri(3,1) = max(NCC3,[],'all');
    matri(4,1) = max(NCC4,[],'all');
    [matri(1,2),matri(1,3)] = find(NCC1 == matri(1,1));
    [matri(2,2),matri(2,3)] = find(NCC2 == matri(2,1));
    [matri(3,2),matri(3,3)] = find(NCC3 == matri(3,1));
    [matri(4,2),matri(4,3)] = find(NCC4 == matri(4,1));
end

function NCC = templateMatching(temp,serch)
    nc1 = zeros((size(serch,1)-
size(temp,1)+1),(size(serch,2)-size(temp,2)+1));
    nc2 = zeros((size(serch,1)-
size(temp,1)+1),(size(serch,2)-size(temp,2)+1));
    nc3 = zeros((size(serch,1)-
size(temp,1)+1),(size(serch,2)-size(temp,2)+1));
    rtempmean = mean(temp(:,:,1),'all');
    gtempmean = mean(temp(:,:,2),'all');
    btempmean = mean(temp(:,:,3),'all');
    rtempstd = std(temp(:,:,1),0,'all');
    gtempstd = std(temp(:,:,2),0,'all');
    btempstd = std(temp(:,:,3),0,'all');
    tempr = floor((size(temp,1))/2);
    tempc = floor((size(temp,2))/2);
    for i = (tempr+1) : (size(serch,1)-tempr)
        for j = (tempc+1) : (size(serch,2)-tempc)
            temp2 = serch((i-tempr):(i+tempr),(j-
tempc):(j+tempc),:);
            [nc1((i-tempr),(j-tempc)),nc2((i-tempr),(j-
tempc)),nc3((i-tempr),(j-tempc))] = func(temp2,
temp,rtempmean,gtempmean,btempmean,rtempstd,gtempstd,btem
pstd);
        end
    end

    NCC = zeros((size(serch,1)-(2*tempr)),(size(serch,2)-
(2*tempc)));

    for i = 1:size(NCC,1)
        for j = 1:size(NCC,2)
            NCC(i,j) = (1/((size(temp,1)*size(temp,2))-
1))*(nc1(i,j)+nc2(i,j)+nc3(i,j));
```

```
        end
    end
    NCC = NCC/3;
end

function [n1,n2,n3] =
func(temp2,temp,rtempmean,gtempmean,btempmean,rtempstd,gt
empstd,btempstd)
    rpatmean = mean(temp2(:,:,1),'all');
    gpatmean = mean(temp2(:,:,2),'all');
    bpatmean = mean(temp2(:,:,3),'all');
    rpatstd = std(temp2(:,:,1),0,'all');
    gpatstd = std(temp2(:,:,2),0,'all');
    bpatstd = std(temp2(:,:,3),0,'all');
    n1 = 0;
    n2 = 0;
    n3 = 0;
    for k = 1:size(temp,1)
        for l = 1:size(temp,2)
            n1 = n1 + ((temp2(k,l,1)-
rpatmean)*(temp(k,l,1)-rtempmean)/(rpatstd*rtempstd));
            n2 = n2 + ((temp2(k,l,2)-
gpatmean)*(temp(k,l,2)-gtempmean)/(gpatstd*gtempstd));
            n3 = n3 + ((temp2(k,l,3)-
bpatmean)*(temp(k,l,3)-btempmean)/(bpatstd*btempstd));
        end
    end
end

function levelx = sample(image)
    x = size(image);
    levelx = zeros(ceil(x(1)/2) , ceil(x(2)/2), x(3));
    for d = 1:ceil(x(1)/2)
        for e = 1:ceil(x(2)/2)
            levelx(d,e,:) = image((2*d-1),(2*e-1),:);
        end
    end
end

function temp2 = blurr(exp)
    w1 = [ 0.05 0.25 0.4 0.25 0.05];
    w2 = [ 0.05 ; 0.25 ; 0.4 ; 0.25 ; 0.05];
    temp1 = imfilter(exp,w1);
    temp2 = imfilter(temp1,w2);
end
```