Directed Study Report

Name: Raghavender Muppavaram

Instructor: Dr. Xiaohui Yuan

Title: Shadow Detection Using Double Threshold Neural Networks.

Abstract:

A novel double-threshold pulse coupled neural networks (DTPCNN) is proposed and applied to shadow detection. This method tries to reduce the false detection of shadows in a single image. The problem using single dynamic threshold PCNN method is that, shadows whose intensity and hue fall in between those of the scene and objectives are often viewed as non-shadows. Moreover, entities with similar or darker hue and intensity may be wrongly classified as shadows. To solve this problem, we have introduced a new approach where we use two different dynamic thresholds that iteratively alter. The upper and lower limits of detecting shadows are determined respectively by a higher threshold that decreases iteratively and a lower one that increases iteratively. The detection result is obtained by a fusion of two detection components.

Keywords:

Shadow detection, double-threshold pulse coupled neural networks (DTPCNN).

Problem:

The problem with the previous shadow detection methods is, the entities with similar or darker hue and intensity may be wrongly classified as shadows.

Solution:

The above problem is solved by using the proposed method Double threshold pulse coupled neural networks. In this approach, we take two threshold that iteratively alter. The upper threshold decreases and the lower threshold increases with every iteration until the stop constraints are met.

Description of proposed method:

The proposed method is double threshold pulse coupled neural networks, used to detect the shadows from single images. In this approach, we consider two dynamic thresholds that iteratively alter. The upper threshold decreases and the lower threshold increases until the stop constraints are met. The detection result is obtained by a fusion of two detection components.

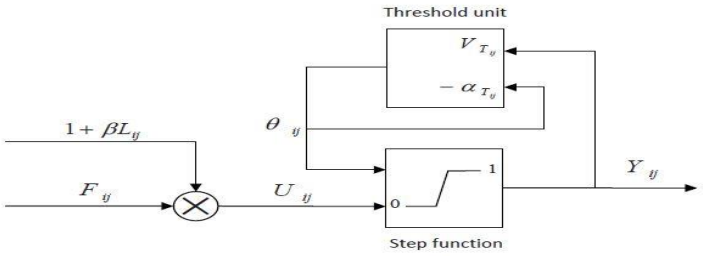


Fig1: PCNN model

Compared with other PCNN models, the greatest difference and improvement of the DTPCNN model is the structure of the threshold regulator.

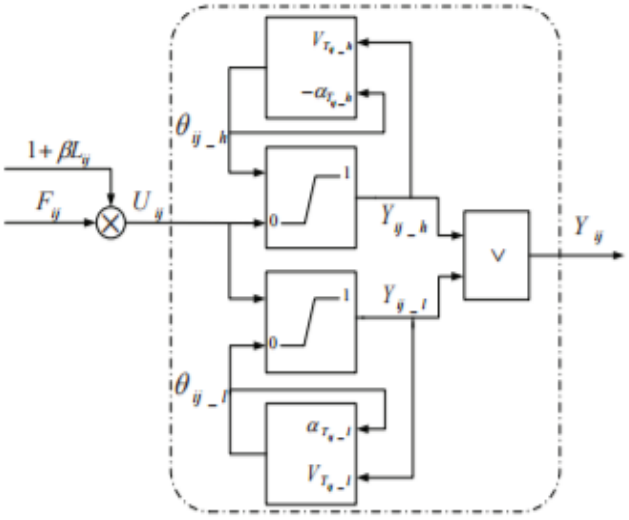


Fig2: Proposed DTPCNN model

In this proposed approach, we consider every pixel of the image as a neuron of DTPCNN. The intensity of the pixel, $I_{i,j}$ is viewed as the external stimuli of a neuron. The feeding input $F_{i,j}$ receives the external stimulus $I_{i,j}$ and output pulse Y from neighboring neurons.

Equation for feeding input:

$$F_{ij}[n] = \sum_{k,s} M_{ijks} Y_{ks}[n-1] + I_{ij}$$

The linking input $L_{i,j}$ receives the input from pulses from the neighboring neurons.

Equation for Linking input:

$$L_{ij}[n] = \sum_{k,s} W_{ijks} Y_{ks}[n-1]$$

In the modulation field, *Lij* is combined with and further modulated with *Fij* to form the internal activity *Uij* which will be delivered to the pulse generator. The pulse generator compares *Uij* with the dynamic thresholds, namely the higher threshold *θij h* and the lower one *θij l*, to determine whether the neuron fires or not. If the *Uij* is greater than *θij h*, the neuron is fired and the output pulse *Yij h* = 1. On the other hand, if *θij l* exceeds *Uij* , the neuron is also fired and the output pulse *Yij l* = 1. To prevent the neuron from being fired again, the higher dynamic threshold *θij h* will be enlarged and the lower one *θij l* will be decreased. The above steps are constantly iterated until the stop constraints met.

Equation for internal activity:

$$U_{ij}[n] = F_{ij}[n](1 + \beta L_{ij}[n])$$

Here the n denotes the iteration times. i,j refers to the pixel position in the image. The M and W are the synaptic gain strengths for the feeding and linking inputs. Here the M and W are replaced by K, where K

is the 3 * 3 kernel which has 0 at the center and 1 for the others. Beta is the linking coefficient of internal activity. Since the parameters values are not given in the algorithm, I have chosen different parameters for different inputs which yield the best results for the inputs. The beta values always change for different inputs which was referred from the paper [1]. G_h and G_l are the higher threshold and lower threshold segmentation matrix. Q_h and Q_l are the high threshold and lower threshold quotient matrix. SD is the shadow detection result obtained by summing Q_h and Q_l values. The following equations are simplified as delta higher and delta lower.

$$\theta_{ij\_h}[n] = e^{-\alpha_{T_{ij\_h}}}\theta_{ij\_h}[n-1] + V_{T_{ij\_h}}Y_{ij\_h}[n]$$

$$\theta_{ij\_l}[n] = e^{-\alpha_{T_{ij\_l}}}\theta_{ij\_l}[n-1] + V_{T_{ij\_l}}Y_{ij\_l}[n]$$

The proposed DTPCNN algorithm:

(1) Initialize the parameters and matrices as $min = 0.004$, $L = 0, U = 0, Y = 0$. Normalize F as $min \leq F \leq 1$.
(2) $L = Y * K, U_{ij} = F_{ij}(1 + \beta L_{ij})$.
(3) If $U_{ij} > \theta_{ij\_h}$,
$Y_{ij\_h} = 1$,
$G_{ij\_h} = \theta_{ij\_h}$,
$Q_{ij\_h} = F(i,j)/G_{ij\_h}$,
$\theta_{ij\_h} = 100$,
else
$Y_{ij\_h} = 0$,
$\theta_{ij\_h} = \theta_{ij\_h} - \Delta_h$.
(4) If $U_{ij} < \theta_{ij\_l}$,
$Y_{ij\_l} = 1$,
$G_{ij\_l} = \theta_{ij\_l}$,
$Q_{ij\_l} = F(i,j)/G_{ij\_l}$,
$\theta_{ij\_l} = 0$,
else
$Y_{ij\_l} = 0$,
$\theta_{ij\_l} = \theta_{ij\_l} + \Delta_l$.
(5) Iterate (2) to (4) N times. $SD = Q_h + Q_l$.

- Initially read input image into I variable.
- After reading the image normalize it as min <= I <= 1, where min value is 0.004.
- Initialize the parameters as zero i.e. L=0, U=0 and F=0.
- Number of iteration are initialized to 100.
- After initializing the parameters, update the feeding input, linking input and internal activity.
- Step function is used to get Q_h values. i.e. If internal activity $U_{i,j}$ is greater than the higher threshold the neuron is fired and the output pulse $Y_{i,j}$ =1.
- Similarly, step function is used to get the Q_l values.
- The above steps are constantly iterated until the stop constraints are met.
- The result Q_h and Q_l values are saved to SD where we detect the shadows and the output image is projected as a binary image a
- Here the image is rgb image I am running the algorithm on three channels i.e. R, G, B. Combine the three channels to get the resultant shadow image.

Code Snippet:

Dtpcnn.m

```
function Y=dtpcnn(I)
   %3 * 3 kernel as mentioned in the paper
   K = [1 1 1; 1 0 1; 1 1 1];
   %K = convn(I, K, 'same');
   [r, w] = size(I);
   %parameters
   L = zeros(size(I));
   U = zeros(size(I)); Y = U;
   theta_L = ones(size(I)); theta_H= ones(size(I));
   G_H = theta_H; G_L = theta_L;
   Y_L = U; Y_H= U;  Q_H = U; Q_L = U;
   N = 100; beta = 7; V_theta = 6; alpha_theta = 0.2;
   F = I;
   for n = 1:N
      %Linking input
      L = convn(Y, K, 'same');
      for i= 1:r
         for j=1:w
            %Feeding input
            F(i,j) = L(i,j) + I(i,j);
            %Internal activity
            U(i, j) = F(i, j)*(1 + beta*L(i, j));
        %Algorithm implemetation of the paper
            if(U(i,j) > theta_H(i,j))
               Y_H(i, j) = 1;
               G_H(i, j) = theta_H(i, j);
               Q_H(i,j) = F(i,j)/G_H(i,j);
               theta_H(i,j) = 100;
            else
               Y_H(i, j) = 0;
               theta_H(i, j) = theta_H(i, j) - 0.0038;
            end

            if(U(i,j) < theta_L(i,j))
               Y_L(i, j) = 1;
               G_L(i, j) = theta_L(i, j);
               Q_L(i,j) = F(i,j)/G_L(i,j);
               theta_L(i,j) = 0;
            else
               Y_L(i, j) = 0;
               theta_L(i, j) = theta_L(i, j) +0.0038 ;

            end

             SD = Q_H(i,j) + Q_L(i,j);
             Y(i,j) = SD;
         end
```

```
      end
   end
end
```

main.m:

```
clc;
clear all;
%close all;

%Reading the image
I= im2double(imread('C:\Users\Raga\Downloads\Directed study code\dcnn2.jpg'));
% Normalizing the values
range = max(I(:)) - min(I(:));
I = (I- min(I(:))) / range;
range2 = 1 - 0.004;
I = (I * range2) + 0.004;
%Applying dtpcnn to R channel
R=dtpcnn(I(:,:,1));
figure;
imshow(R);
%Applying dtpcnn to G channel;
G=dtpcnn(I(:,:,2));
figure;
imshow(G);
%Applying dtpcnn to B channel;
B=dtpcnn(I(:,:,3));
figure;
imshow(B);

Concatenating the R,G,B cahnnels ;
Y=cat(3, R, G, B);
figure;
imshow(Y);
Shadow image output
figure;
imshow(im2bw(Y));
```

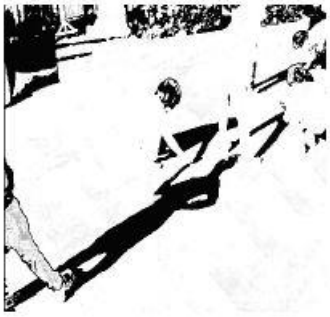
Challenges faced:

The major challenge is to select different Beta values for different input images.

Experiments and analysis:

This algorithm is tested on different input images. One of the images are taken from the paper to test the algorithm. I have done the experiments on several images. The shadow detection depends on the beta value for different beta values the results are different, I have chosen the value which gives the best output. I am showing some of the example image on how the results change for different beta values. I have also calculated accuracy for all the images below.

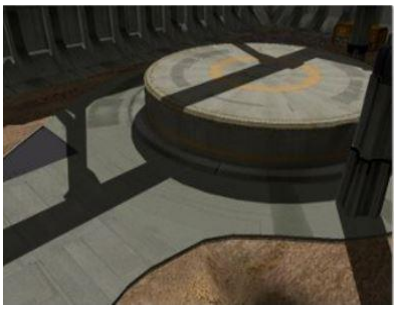
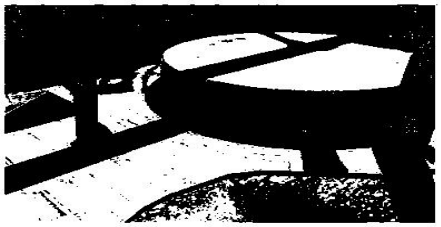
1.


a) Input image b) Result from paper c) Output Image

The beta value for this image is 1.95.

2.

a) input image b) output image

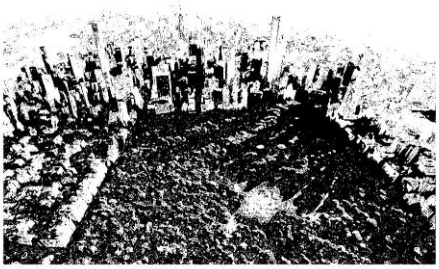
The beta value for the image is 3.5

3.


a) Input image b) output image

The beta value for the image is 8.

4.

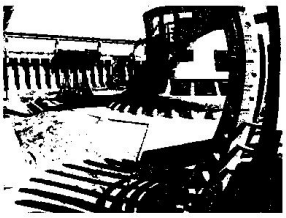


a) Input image

b) output image.

The beta value is 3.5.

5.

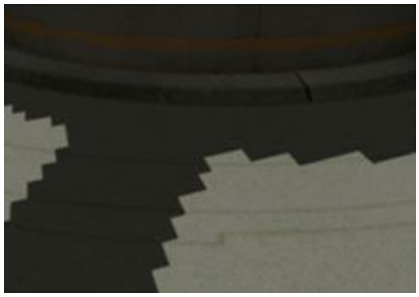


a) Input image

b) output image

The beta value is 10.5

6.

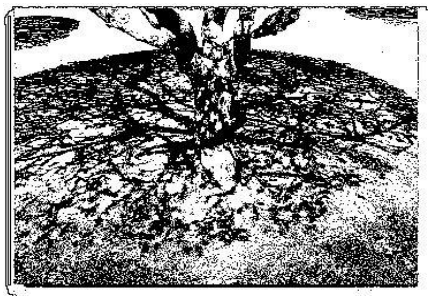


a) Input image

b) output image

The beta is 9.5.

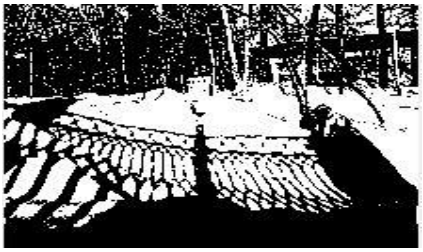
7.



a) Input image b) output image

The beta value for this image is 4.

8.



a) Input image b) output image

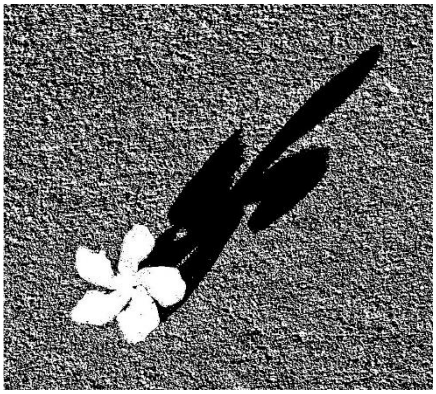
The beta value for this image is 5.5.

9.


a: Input image b: output image

The beta value for the image is 6.

10.

a: Input image b: output image

The beta value for the image is 8**.**

Note: The black part represents the shadows in the output image**.**

Accuracy results:

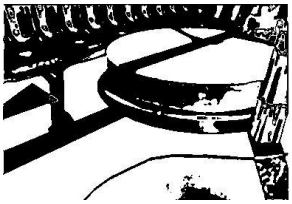
Images	Accuracy
Image 1	83.4
Image 2	87.34
Image 3	79.80
Image 4	74
Image 5	74
Image 6	70
Image 7	82.6
Image 8	78.4
Image 9	76
Image 10	74.4

Results generated after changing beta value:

1.

The beta value is changed from 1.95 to 3 for this image.

2.



The beta value is changed from 9 to 1.5.

3.

The beta value is changed from 8 to 2.
These are some of the examples to show how beta value affects the shadow detection.

Conclusion:

The proposed method DTPCNN significantly reduces the false detection of shadows in a single image where the color and intensity of some non-shadow regions are close to or even lower than those in shadow regions. The beta value is not constant, it changes every time for different experiments. The values of the various parameters depend on the scenes. The experimental results verify that the proposed approach in this paper yields the best performance among all approaches used in comparison. This algorithm works efficiently and gives accurate results for shadow detection.

References:

[1]. X. D. Gu, D. H. Yu, L. M. Zhang, "Image Shadow Removal Using Pulse Coupled Neural Network," *IEEE Trans. Neural Networks*, vol. 16, no. 3, pp. 692-698, May 2005.

[2]. J. L. Johnson, M. L. Padgett, "PCNN Models and Applications,"*IEEE Trans. Neural Networks*, vol. 10, issue.3, pp. 480-498, 1999.