Cmpe 362

HOMEWORK 3

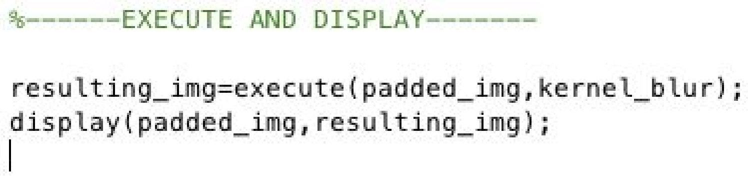
REPORT

Şadi Uysal

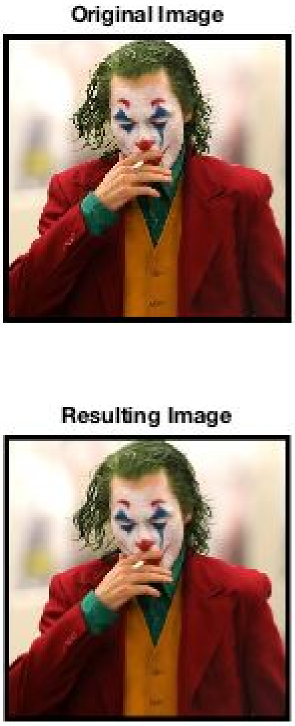
2015400162

Please change “EXECUTE AND DISPLAY” parts to get corresponding results

1.1



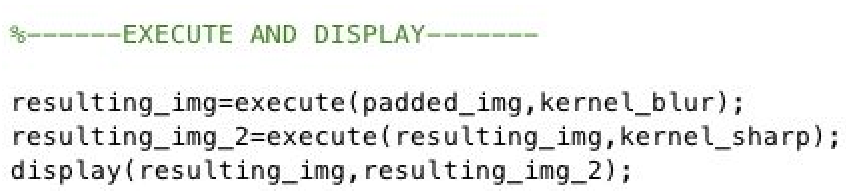
```
%------EXECUTE AND DISPLAY-------

resulting_img=execute(padded_img,kernel_blur);
display(padded_img,resulting_img);
```



Original Image

Resulting Image

1.2

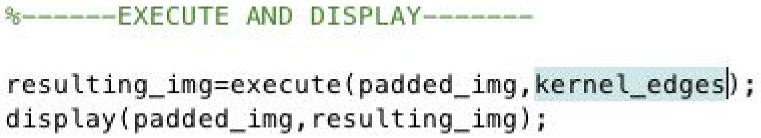


```
%------EXECUTE AND DISPLAY-------

resulting_img=execute(padded_img,kernel_blur);
resulting_img_2=execute(resulting_img,kernel_sharp);
display(resulting_img,resulting_img_2);
```

Original Image

Resulting Image

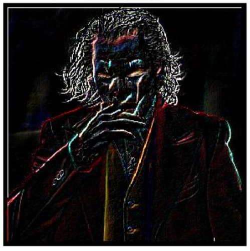
1.3



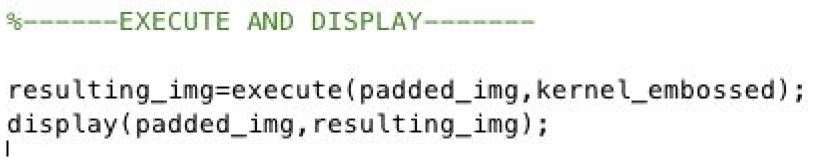
```
%-------EXECUTE AND DISPLAY-------

resulting_img=execute(padded_img,kernel_edges);
display(padded_img,resulting_img);
```

Original Image

Resulting Image

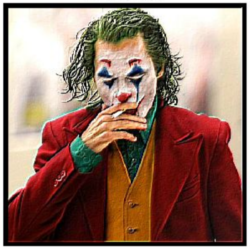


1.4



```
%-------EXECUTE AND DISPLAY--------

resulting_img=execute(padded_img,kernel_embossed);
display(padded_img,resulting_img);
```

Original Image



Resulting Image

CODE:

```
original_img = imread('jokerimage.png');
padded_img = padarray(original_img,[10 10],0,'both');

%------------KERNELS--------------
% Kernel that adds blur to your image
kernel_blur = ones(3, 3)/(3*3);
% Kernel that sharpens your image
kernel_sharp = [0 -1 0; -1 5 -1; 0 -1 0];
%Highlight Edges Kernel
kernel_edges = -ones(3, 3);
kernel_edges(1,1)=8;
%Kernel that makes your image embossed
kernel_embossed = [-2 -1 0 ; -1 1 1 ; 0 1 2];

%------EXECUTE AND DISPLAY-------

resulting_img=execute(padded_img,kernel_embossed);
display(padded_img,resulting_img);

%-------------HELPER FUNCTIONS--------------
%function for displaying images
function display(img_1, img_2)
% Display the original color image.
subplot(2, 1, 1);
imshow(img_1);
title('Original Image');
% Display the resulting image.
subplot(2, 1, 2);
imshow(img_2);
title('Resulting Image');
end

%function execute
function result = execute(img, kernel)
   % blue,green,red matricies.
   blue = img(:, :, 3);
   green = img(:, :, 2);
```

```
    red = img(:, :, 1);
    % Convolution of 3 color matricies.
    redBlurred = conv_img(red, kernel);
    greenBlurred = conv_img(green, kernel);
    blueBlurred = conv_img(blue, kernel);
    % Combine color matricies.
    result = cat(3, uint8(redBlurred), uint8(greenBlurred), uint8(blueBlurred));
end

%function for image convolution
function result = conv_img(img, kernel)
    % Get the dimensions of the image.
    [rows, columns] = size(img);
    [k_rows, k_columns] = size(kernel);
    temp=fliplr(kernel);
    flipped_kernel=flipud(temp);
    center = floor((size(flipped_kernel)+1)/2);
    %bottom size according to center rows
    b_size = k_rows - center(1);
    %top size according to center rows
    u_size = center(1) - 1;
    %right size according to center cols
    r_size = k_columns - center(2);
    %left size according to center cols
    l_size = center(2) - 1;

    temp = zeros(rows + u_size + b_size, columns + l_size + r_size);
    for i = 1 + u_size : rows + u_size
        for j = 1 + l_size : columns + l_size
            temp(i,j) = img(i - u_size, j - l_size);
        end
    end
    %matrix for conv result
    result = zeros(rows , columns);
    for img_row = 1 : rows %iterate over img
        for img_col = 1 : columns
            for k_row = 1 : k_rows %iterate over kernel
                for k_col = 1 : k_columns
                    r_off = img_row - 1; %row offset for temp matrix
                    c_off = img_col -1; %col offset for temp matrix
                    result(img_row, img_col) = result(img_row, img_col) + (temp(k_row +
r_off, k_col + c_off) * flipped_kernel(k_row, k_col));
```

```
                end
            end
        end
    end
end
```