Matlab Image Processing Results

Multiplying 2 matrices

```
>> A = [1 2 3 4 5;2 3 4 5 2;1 2 4 5 3;1 2 3 4 2;1 3 4 5 2]

A =

    1    2    3    4    5
    2    3    4    5    2
    1    2    4    5    3
    1    2    3    4    2
    1    3    4    5    2

>> B = [0 0 0 0 1;0 0 0 1 0;0 0 1 0 0;0 1 0 0 0;1 0 0 0 0]

B =

    0    0    0    0    1
    0    0    0    1    0
    0    0    1    0    0
    0    1    0    0    0
    1    0    0    0    0

>> C = A*B

C =

    5    4    3    2    1
    2    5    4    3    2
    3    5    4    2    1
    2    4    3    2    1
    2    5    4    3    1
```

Multiplying two matrices element wise

```
>> D = A.*B

D =

    0    0    0    0    5

    0    0    0    5    0

    0    0    4    0    0

    0    2    0    0    0

    1    0    0    0    0
```

Loading an image and displaying it

```
>> Image = imread('image.jpg');

>> imshow(Image);
```

Using size function to get the dimensions

```
>> size(Image)

ans =

  527   518     3
```

Displaying RGB layers separately

```
>> R = Image(:,:,1);

>> G = Image(:,:,2);

>> B = Image(:,:,3);

>> imread(R);
```

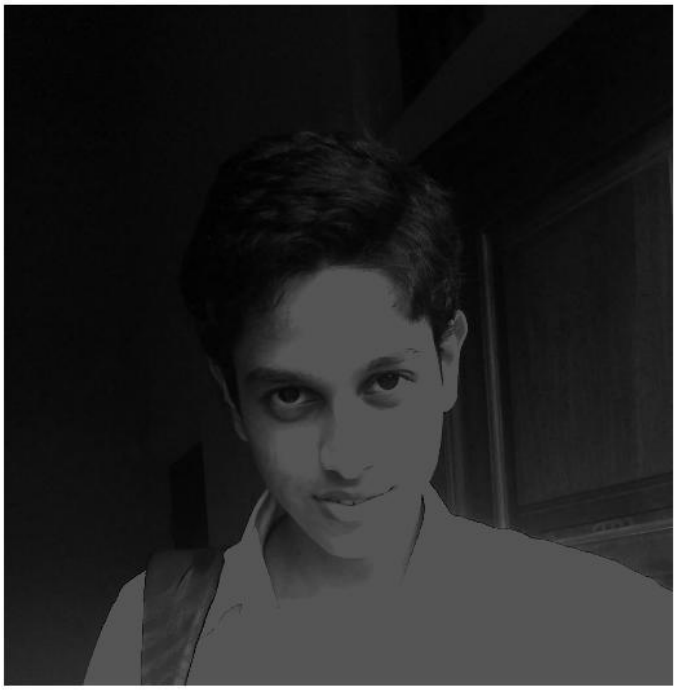
>> imshow(G);

>> imshow(B);

Converting the colored image to gray scale

>> GrayScale = (R + G + B) / 3;

>> imshow(GrayScale);



>> GrayScale2 = rgb2gray(Image);

>> imshow(GrayScale2);

Image convolution using my own function

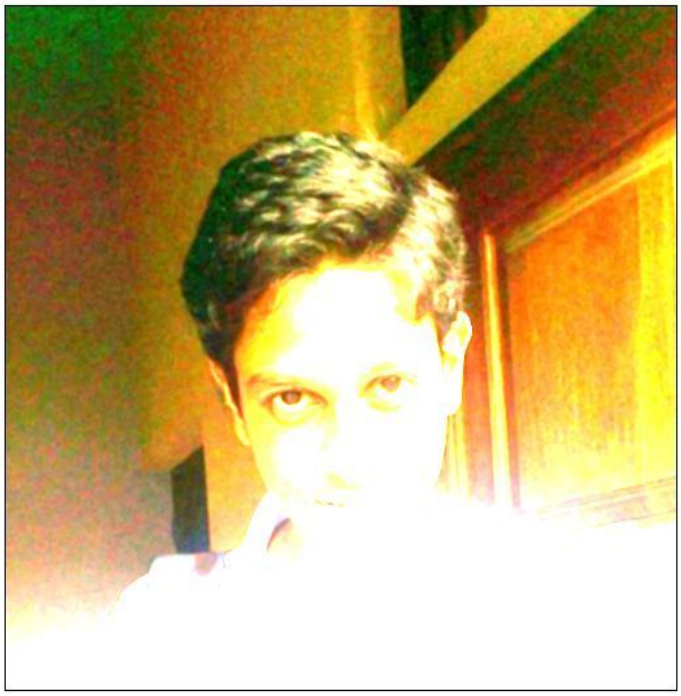
```
>>edit convert

function [Output] = convert(Image,Kernel)

Output = zeros(size(Image,1),size(Image,2));
Rows = int32(size(Image,1));
Cols = int32(size(Image,2));
K_Size = int32(size(Kernel,1));

for i=(K_Size/2):(Rows -(K_Size/2) + 1),
        for j=(K_Size/2):(Cols -(K_Size/2) + 1),

                Tot=0;
                a = i+1-(K_Size/2);
                b = i-1+(K_Size/2);
                c = j+1-(K_Size/2);
                d = j-1+(K_Size/2);

                Reduced_Matrix = Image(a:b,c:d);
                Reduced_Matrix = Reduced_Matrix.*Kernel;
                Tot = sum(Reduced_Matrix(:));
                Output(i,j) = Tot;

        end
end

>> Image = im2double(Image);
>> R = Image(:,:,1);
>> G = Image(:,:,2);
>> B = Image(:,:,3);
>> Kernel = [1 1 1;1 1 1;1 1 1]

Kernel =

   1    1    1
   1    1    1
   1    1    1
```

```
>> Converted_R = convert(R,Kernel);
>> imshow(Converted_R);
```

```
>> Converted_G = convert(G,Kernel);

>> imshow(Converted_G);
```

```
>> Converted_B = convert(B,Kernel);

>> imshow(Converted_B);
```

```
>> Converted_Image = cat(3,Converted_R,Converted_G,Converted_B);

>> imshow(Converted_Image);
```



Using ‘conv2’ function

```
>> Converted_R2 = conv2(R,Kernel);

>> Converted_G2 = conv2(G,Kernel);

>> Converted_B2 = conv2(B,Kernel);

>> Converted_Image2 = cat(3,Converted_R2,Converted_G2,Converted_B2);

>> imshow(Converted_Image2);
```

Image Convolutions

1. Image blurring

a. Simple Box Blur

```
>> Kernel = [1/9 1/9 1/9;1/9 1/9 1/9;1/9 1/9 1/9]

Kernel =

   0.1111    0.1111    0.1111

   0.1111    0.1111    0.1111

   0.1111    0.1111    0.1111

>> Converted_R = convert(R,Kernel);

>> Converted_G = convert(G,Kernel);

>> Converted_B = convert(B,Kernel);

>> Converted_Image = cat(3,Converted_R,Converted_G,Converted_B);

>> imshow(Converted_Image);
```

b. Gaussian Blur

```
Kernel =

    0     0     0     5     0     0     0
    0     5    18    32    18     5     0
    0    18    64   100    64    18     0
    5    32   100   100   100    32     5
    0    18    64   100    64    18     0
    0     5    18    32    18     5     0
    0     0     0     5     0     0     0

>> tot = sum(Kernel (:))

tot =

      1068

>> Kernel = Kernel / tot

Kernel =

         0         0         0    0.0047         0         0         0
         0    0.0047    0.0169    0.0300    0.0169    0.0047         0
         0    0.0169    0.0599    0.0936    0.0599    0.0169         0
    0.0047    0.0300    0.0936    0.0936    0.0936    0.0300    0.0047
         0    0.0169    0.0599    0.0936    0.0599    0.0169         0
         0    0.0047    0.0169    0.0300    0.0169    0.0047         0
         0         0         0    0.0047         0         0         0
```

Implementation using my function

Using conv2 function

2. Edge detection

Kernel =

-1 -1 -1

-1 8 -1

-1 -1 -1

Edge detection with Sobel Edge Operator

a. Horizontal

Kernel =

-1	-2	-1
0	0	0
1	2	1

b. Vertical

Kernel =

-1	0	1
-2	0	2
-1	0	1

c. With both convolutions applied

3. Sharpening

Kernel =

0	-1	0
-1	5	-1
0	-1	0

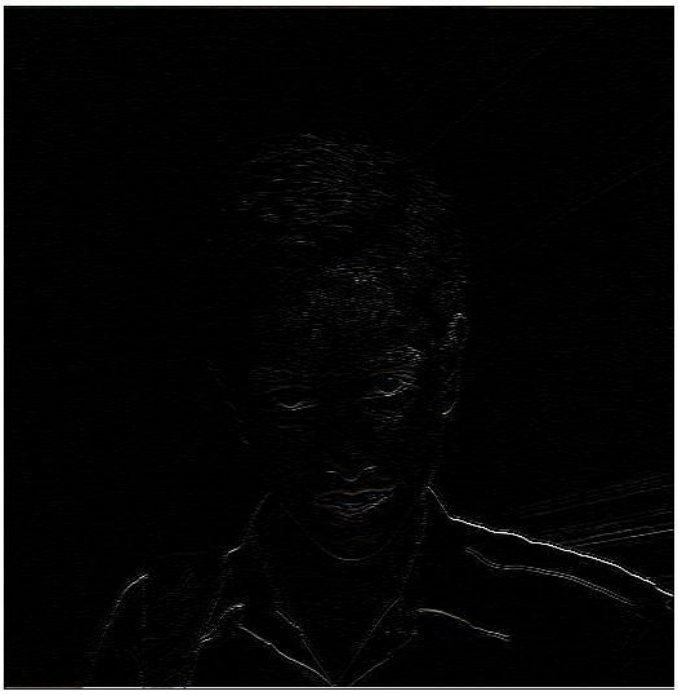
Other Convolutions

1. Line detection

a. Horizontal Lines

Kernel =

-1	-1	-1
2	2	2
-1	-1	-1



b. Vertical lines

Kernel =

-1 2 -1

-1 2 -1

-1 2 -1

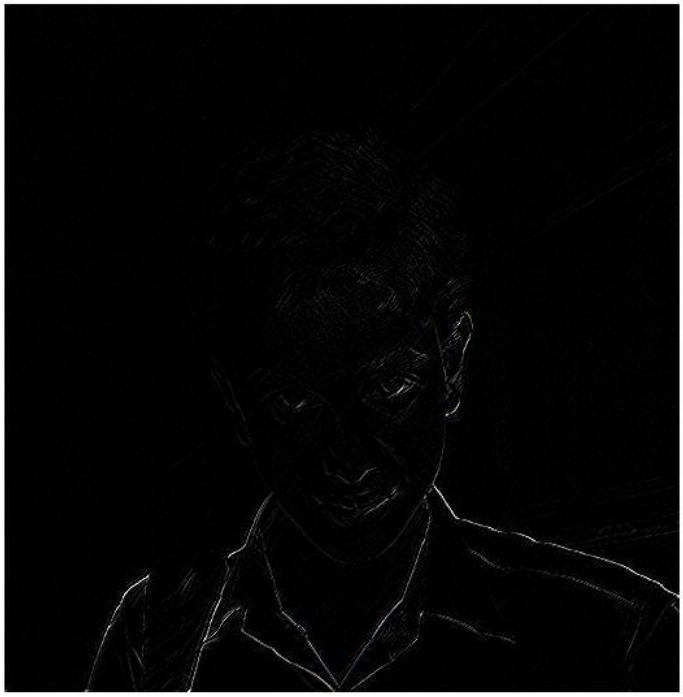
c. 45^0 lines

Kernel =

-1 -1 2

-1 2 -1

2 -1 -1



d. 135^0 lines

Kernel =

2 -1 -1

-1 2 -1

-1 -1 2