ELEC408-Computer Vision
HW1-Hybrid Images
Report

14.10.2016

EKİN AYÜREK
ID:31459

Table of Contents

1. Introduction

Hybrid images are combination of two different images such that one of them is low-pass filtered the other one is high-pass filtered in spatial domain. Thus, hybrid images have two different interpretations which may toggle with distance. If viewer move away from the image, the viewer begins to see more of the low-pass filtered image, else he begins to see more of the high-pass filtered image. [1] The method is basic: we filter one image with Gaussian low-pass filter and we filter the other image with filter which is (1-lowpass). After that, we sum these two filtered image together. The result image will be a hybrid image. In this assignment, we are required to generate hybrid images in MATLAB without using imfilter, conv, conv2, etc. functions. [2]

2. Filtering Algorithm

We need to implement our imfilter() function which basically convolves filter with the image. Our images are 3 channel 2D images.

1) I define my_imfilter() function which takes two input: filter and image and returns filtered image that has same resolution with input image.

```
function output = my_imfilter(image, filter)
```

2) I flip rows and columns of filter array for the convolution operation.

```
filter = fliplr(filter);
filter = transpose(fliplr(transpose(filter)));
```

3) I get sizes of the filter and image.

```
size_of_image = size(image);
size_of_filter = size(filter);.
```

4)I assume that input image can be double, single, uint8 format in MATLAB because of that I define different type of output array according given input image. In addition, uint8 format in MATLAB can not be multiplied with double filter array. Therefore, I first converted it to double and converted back in the end of the function.

```
type = 0;
if isa(image,'uint8')
    image = double(image);
    output = zeros(size_of_image, 'uint8');
    type = 0;
elseif isa(image,'double')
    output = zeros(size_of_image,'double');
    type = 1;
elseif isa(image,'single')
    output = zeros(size_of_image, 'single');
    type = 2;
else
```

```
    display('Image format is not valid, try again')
end
```

5)In order to get output image that has same resolution with the input image I pad the image with zeros according to filter size

```
pad_size = [(size_of_filter(1)-1)/2 , (size_of_filter(2)-1)/2];
padded_image = padarray(image,pad_size);
```

6)I convolved 3 channel(RGB) separately. For each (i,j) pairs, I multiplied a filter sized partition of image with filter and write summation to the element in the (i,j) index of output. For uint8 integer I converted sum to uint8

```
for z = 1:3
  padded_channel = padded_image(:,:,z);
     for i = 1:size_of_image(1)
        for j = 1:size_of_image(2)
 intersection = padded_channel(i:i+(size_of_filter(1)-
1),j:j+size_of_filter(2)-1) .*filter ;
           if type==0
               output(i,j,z) =uint8(round(sum(sum(intersection))));
            else
output(i,j,z) = sum(sum(intersection));
            end
        end
    end
end
```

3. Creating Hybrid Images

Our images have already aligned therefore the hybrid images which are generated from those images looks very well. However, we should adjust the standard deviation(cut-off frequency) parameter of our Gaussian low-pass filter for each image. Our Gaussian filter’s size is defined as 4*cut-off frequency+1. I tried dog.bmp and cat.bmp image with given sigma value. It looks very well. For other images I try number near given sigma and record the best sigma for each picture.

4. Resulting Hybrid Images

4.1. Cat and Dog

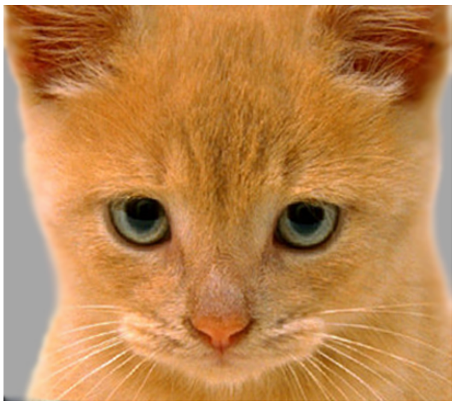

Figure 1 Cat Original

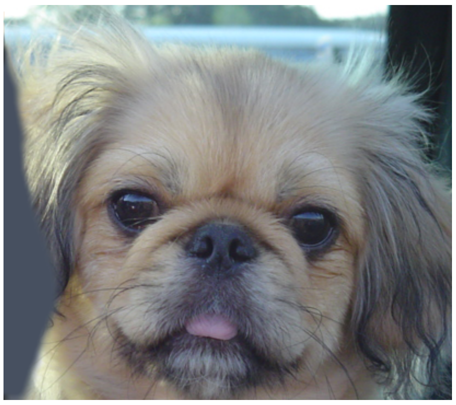

Figure 2 Dog Original

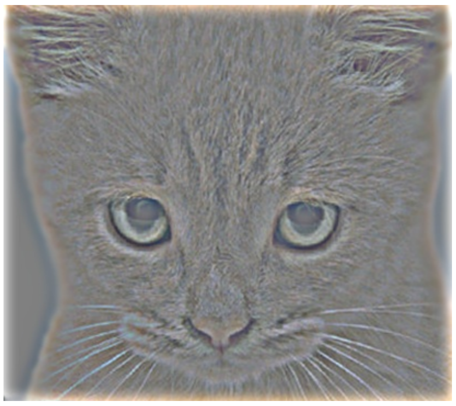

Figure 3 Cat High-Passed

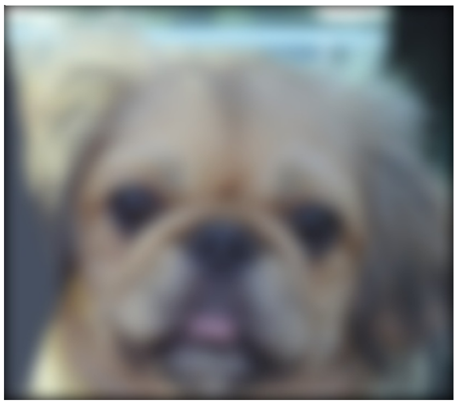

Figure 4. Dog Low-Passed

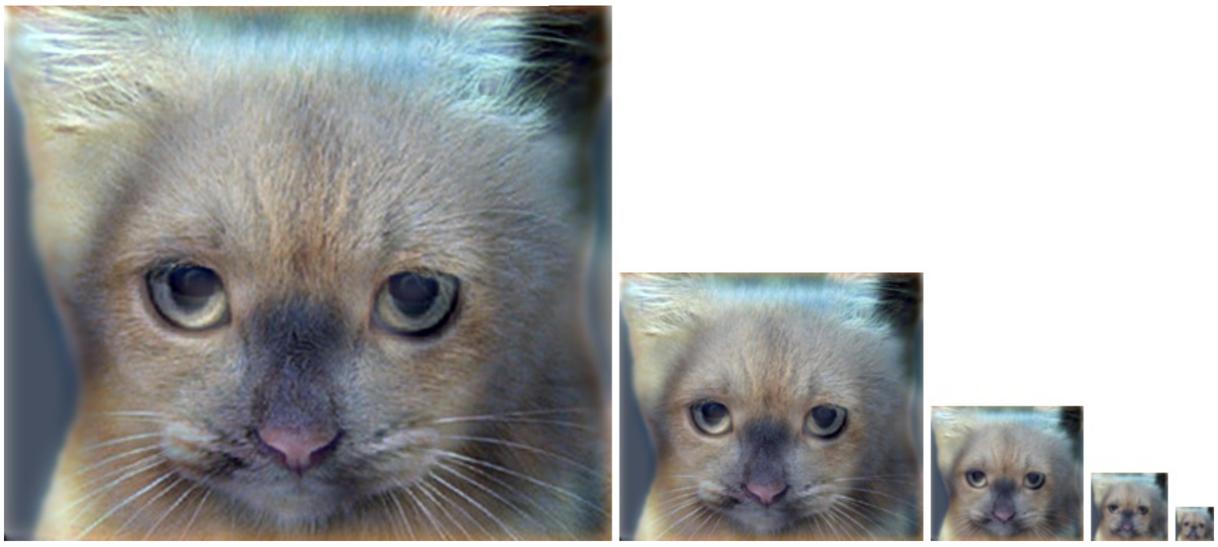

Figure 5 Hybrid images scaled Cat and Dog

Cut-off frequency was: 7

4.2. Marilyn and Einstein

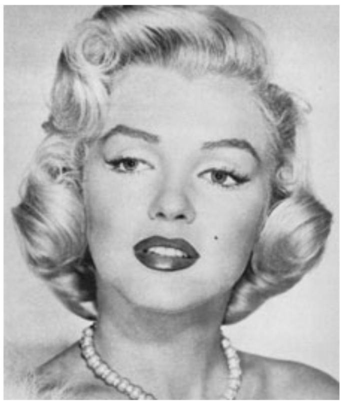

Figure 2 Marilyn Original

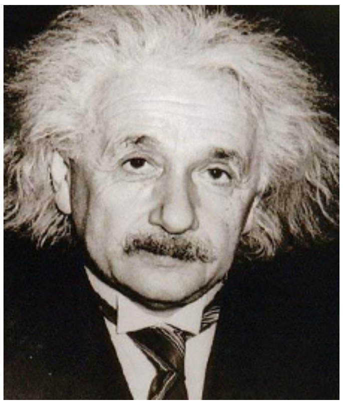

Figure 2 Einstein Original

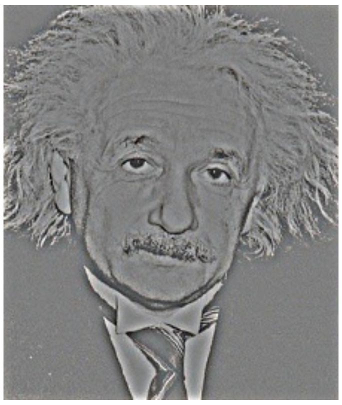

Figure 3 Einstein High-Passed

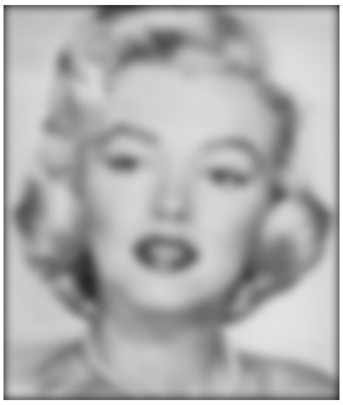

Figure 4. Marilyn Low-Passed

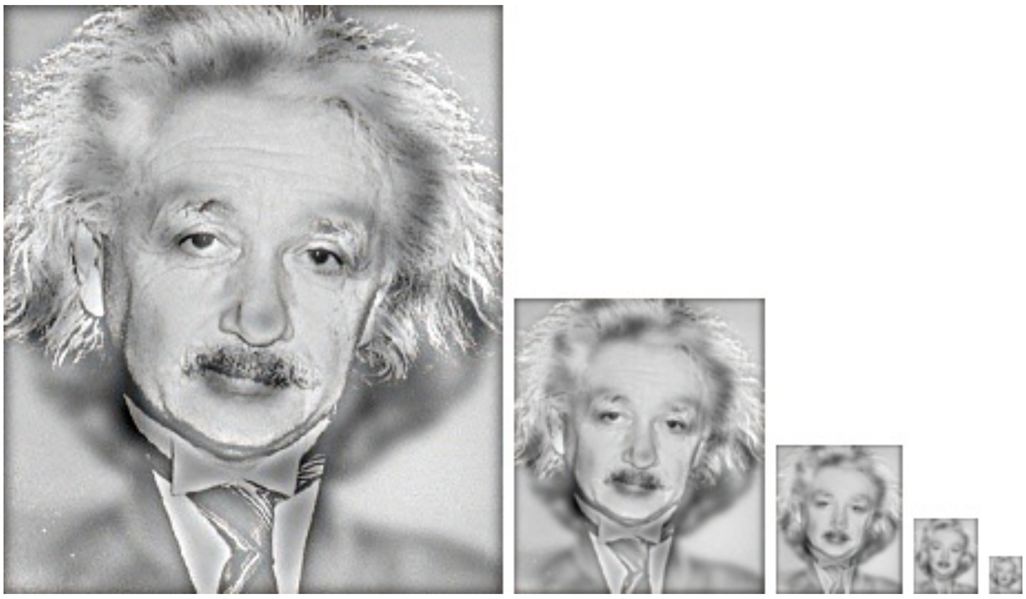

Figure 5 Hybrid images scaled Einstein and Marilyn

Cut-off frequency was: 3.5

4.3. Plane and Bird

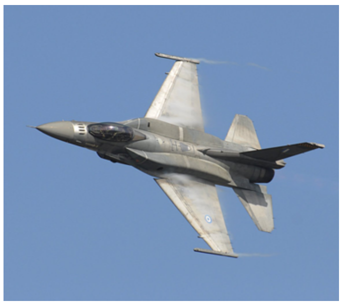
Figure 3 Plane Original

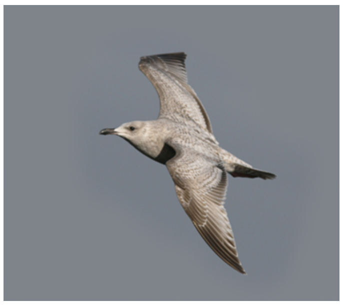
Figure 2 Bird Original

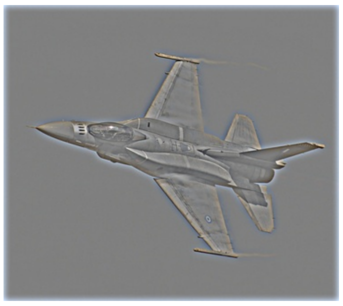
Figure 3 Plane High-Passed

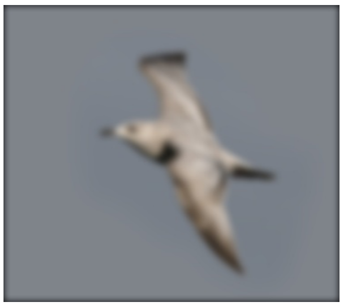
Figure 4. Bird Low-Passed

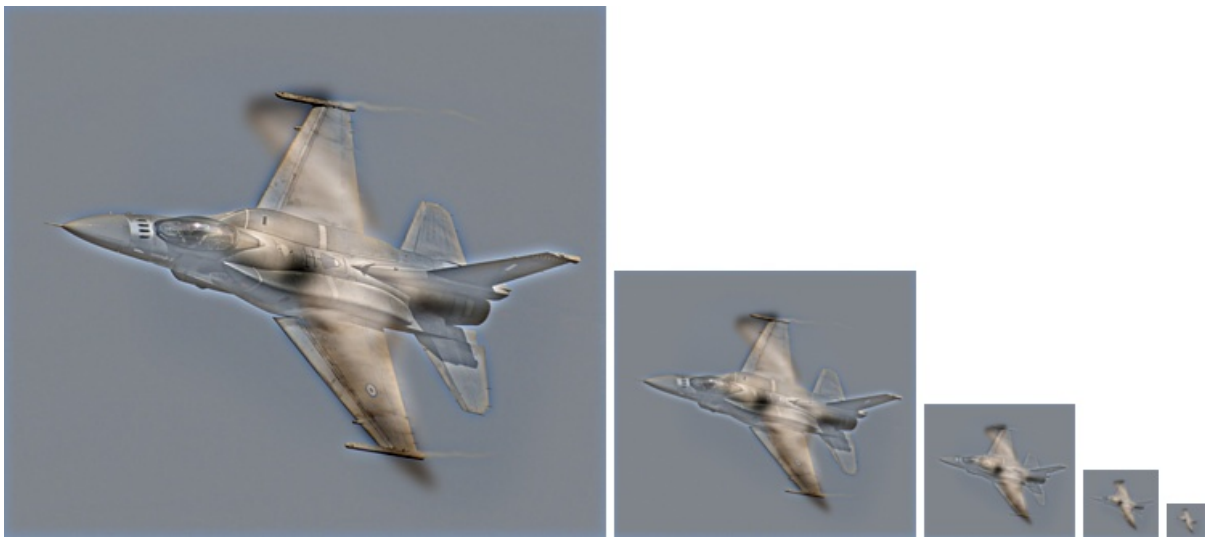
Figure 5 Hybrid images scaled plane and bird

Cut-off frequency was: 4

4.4. Bicycle and Motorcycle

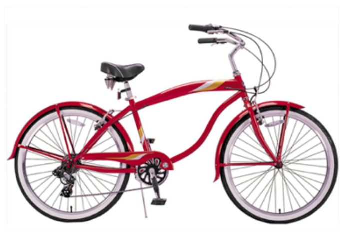
Figure 4 Bicycle Original

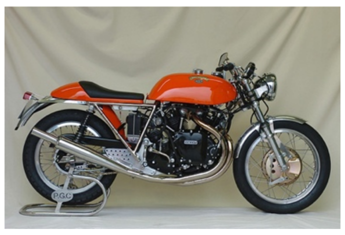
Figure 2 Motorcycle Original

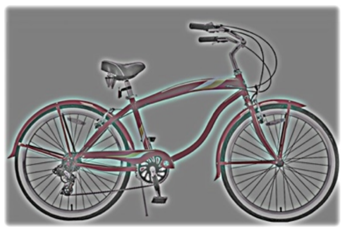
Figure 3 Bicycle High-Passed

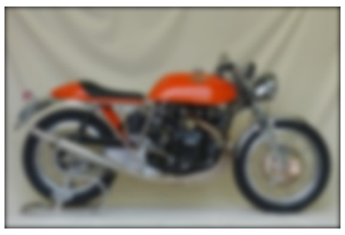
Figure 4. Motorcycle Low-Passed

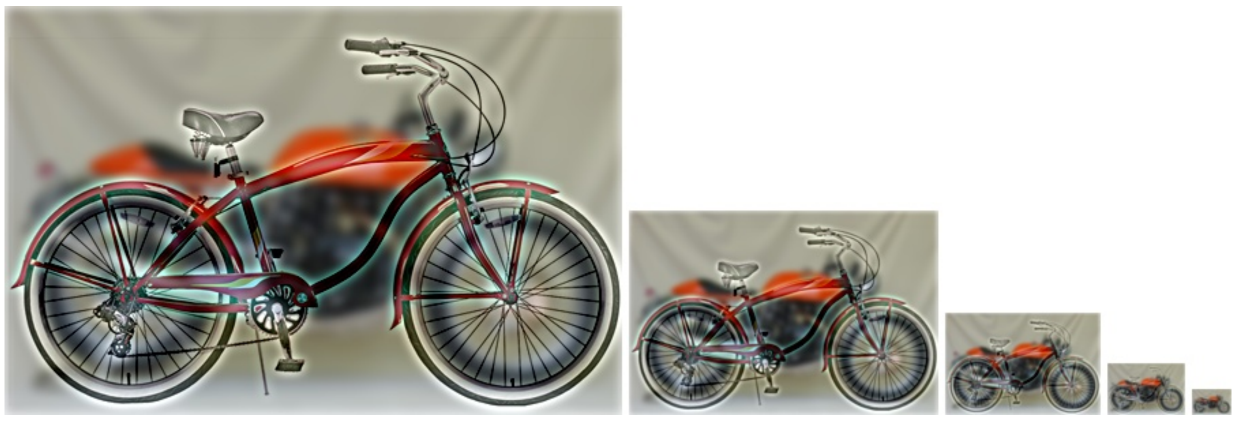
Figure 5 Hybrid images scaled bicycle and motorcycle

Cut-off frequency was: 4

4.5. Fish and Submarine

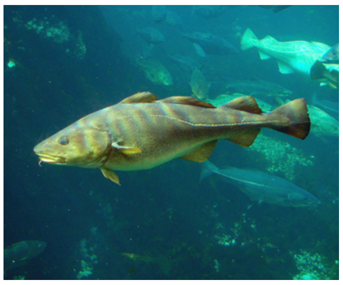
Figure 6 Fish Original

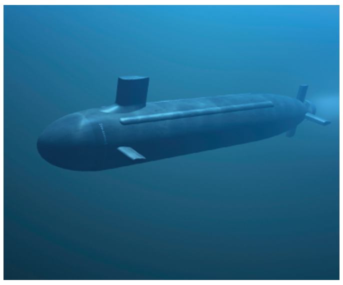
Figure 2 Submarine Original

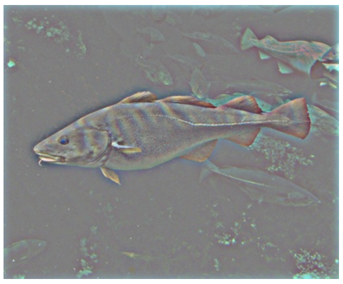
Figure 3 Fish High-Passed

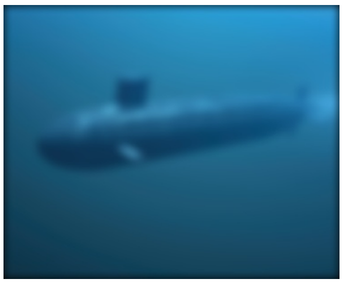
Figure 4. Submarine Low-Passed

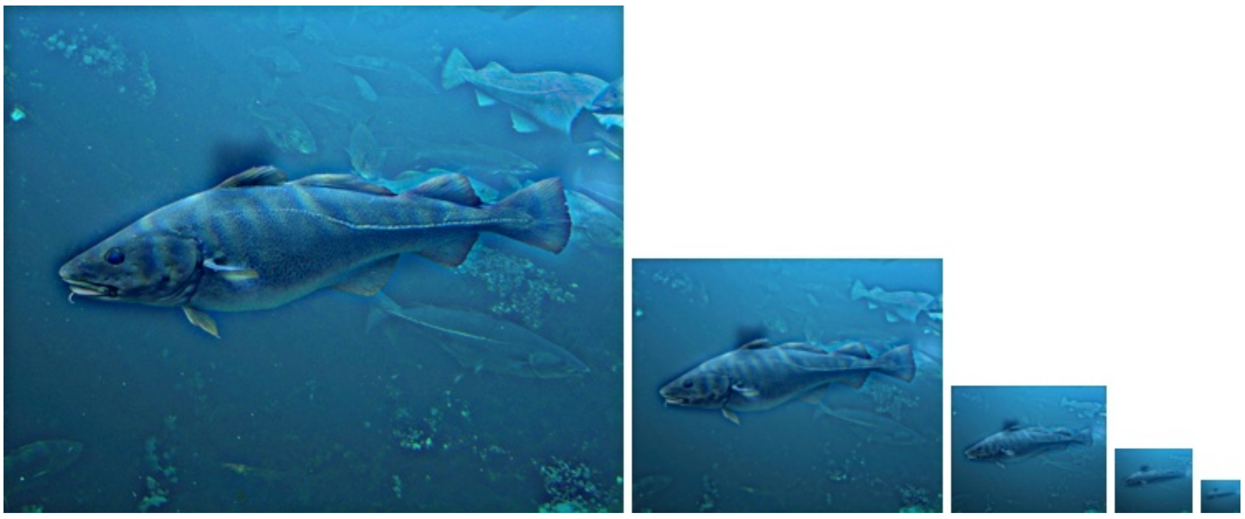
Figure 7 Hybrid images scaled Fish and Submarine

Cut-off frequency was: 5

5. Conclusion

I implement my own imfilter() function and I learn how to convolve 2D arrays. Now, I am able to create hybrid images in any size and in any type with my MATLAB code. I generated 5 hybrid image examples from our assignment files. My codes are in the appendices.

6. Bibliography

[1] A. Oliva, T. Antonio and P. G. Schyns, “Hybrid images,” 2006.

[2] Y. Yemez, “COMP 508 Assignment 1,” 2016. [Online]. Available: http://home.ku.edu.tr/%7Eyyemez/comp508/Assignment1.html. [Accessed 13 10 2016].

7. Appendices

7.1. my_imfilter.m

```
function output = my_imfilter(image, filter)
% This function is intended to behave like the built in function imfilter()
% See 'help imfilter' or 'help conv2'. While terms like "filtering" and
% "convolution" might be used interchangeably, and they are indeed nearly
% the same thing, there is a difference:
% from 'help filter2'
%    2-D correlation is related to 2-D convolution by a 180 degree rotation
%    of the filter matrix.

% Your function should work for color images. Simply filter each color
% channel independently.

% Your function should work for filters of any width and height
% combination, as long as the width and height are odd (e.g. 1, 7, 9). This
% restriction makes it unambigious which pixel in the filter is the center
% pixel.

% Boundary handling can be tricky. The filter can't be centered on pixels
% at the image boundary without parts of the filter being out of bounds. If
% you look at 'help conv2' and 'help imfilter' you see that they have
% several options to deal with boundaries. You should simply recreate the
% default behavior of imfilter -- pad the input image with zeros, and
% return a filtered image which matches the input resolution. A better
% approach is to mirror the image content over the boundaries for padding.

% % Uncomment if you want to simply call imfilter so you can see the desired
% % behavior. When you write your actual solution, you can't use imfilter,
% % filter2, conv2, etc. Simply loop over all the pixels and do the actual
% % computation. It might be slow.
% output = imfilter(image, filter);



%%%%%%%%%%%%%%%%
% Your code here
%First flip of the columns the filter for the convolution operation
 filter = fliplr(filter);
 %Flip the rows for the convolution operation
 filter = transpose(fliplr(transpose(filter)));



%size of image and filter are used in convolution summation limits
size_of_image = size(image);
size_of_filter = size(filter);

%image input can be three different type such that: standart uint8 RGB,
%doubled(im2double) and single(im2single). If its uint8 it should be
%converted to double for filter operation. Because matlab cannot perform
%uint8*double operation, I evaluate those situations
type = 0;
if isa(image,'uint8')
    %The input is in the uint8 format, I convert it to double
    image = double(image);
    %I initialize the output image with given input type
    output = zeros(size_of_image, 'uint8');
    %uint8 is named as type 0
    type = 0;
elseif isa(image,'double')
    %if the input is double no need to convert it
    %I initialize the output image with given input type
    output = zeros(size_of_image,'double');
    %double is enumarated as type 1
    type = 1;
elseif isa(image,'single')
    %if the image is single no need to convert it
     %I initialize the output image with given input type
    output = zeros(size_of_image, 'single');
    %single is enumarated as type 2
    type = 2;
else
    display('Image format is not valid, try again')
end

%pading is applied for achieve the same resolution after convolution
pad_size = [(size_of_filter(1)-1)/2 , (size_of_filter(2)-1)/2];
padded_image = padarray(image,pad_size);

%filter applied channel by channel
for z = 1:3
    % get one channel of the image
    padded_channel = padded_image(:,:,z);
    %convolution operation
    for i = 1:size_of_image(1)
        for j = 1:size_of_image(2)
                intersection = padded_channel(i:i+(size_of_filter(1)-1),j:j+size_of_filter(2)-1) .*filter ;
                if type==0
                    %if the image uint8. It should be converted back
                    output(i,j,z) =uint8(round(sum(sum(intersection))));
                else
                    %if the image double or single no need to convert back
                    output(i,j,z) = sum(sum(intersection));
                end
        end
    end
end
%%%%%END OF THE CODE%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
```

7.2. proj1.m

```matlab
% Before trying to construct hybrid images, it is suggested that you
% implement my_imfilter.m and then debug it using proj1_test_filtering.m

% Debugging tip: You can split your MATLAB code into cells using "%%"
% comments. The cell containing the cursor has a light yellow background,
% and you can press Ctrl+Enter to run just the code in that cell. This is
% useful when projects get more complex and slow to rerun from scratch

close all; % closes all figures

%% Setup
% read images and convert to floating point format
image1 = im2single(imread('../data/motorcycle.bmp'));
image2 = im2single(imread('../data/bicycle.bmp'));

% Several additional test cases are provided for you, but feel free to make
% your own (you'll need to align the images in a photo editor such as
% Photoshop). The hybrid images will differ depending on which image you
% assign as image1 (which will provide the low frequencies) and which image
% you asign as image2 (which will provide the high frequencies)

%% Filtering and Hybrid Image construction
cutoff_frequency = 4; %This is the standard deviation, in pixels, of the
% Gaussian blur that will remove the high frequencies from one image and
% remove the low frequencies from another image (by subtracting a blurred
% version from the original version). You will want to tune this for every
% image pair to get the best results.
filter = fspecial('Gaussian', cutoff_frequency*4+1, cutoff_frequency);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
% YOUR CODE BELOW. Use my_imfilter create 'low_frequencies' and
% 'high_frequencies' and then combine them to create 'hybrid_image'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
% Remove the high frequencies from image1 by blurring it. The amount of
% blur that works best will vary with different image pairs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%

low_frequencies = my_imfilter(image1,filter);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
% Remove the low frequencies from image2. The easiest way to do this is to
% subtract a blurred version of image2 from the original version of image2.
% This will give you an image centered at zero with negative values.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%

high_frequencies = image2-my_imfilter(image2,filter);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Combine the high frequencies and low frequencies
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%

hybrid_image = high_frequencies + low_frequencies;

%% Visualize and save outputs
figure(1); imshow(low_frequencies)
figure(2); imshow(high_frequencies + 0.5);
vis = vis_hybrid_image(hybrid_image);
figure(3); imshow(vis);
imwrite(low_frequencies, 'low_frequencies.jpg', 'quality', 95);
imwrite(high_frequencies + 0.5, 'high_frequencies.jpg', 'quality', 95);
imwrite(hybrid_image, 'hybrid_image.jpg', 'quality', 95);
imwrite(vis, 'hybrid_image_scales.jpg', 'quality', 95);
```