Image filtering and Hybrid images

Taneesh
2018csb1186

Indian Institute of Technology
Ropar

Abstract

Here I present the detailed report of assignment 1 of course CS518, Computer Vision where instead of using built in image filters we are implementing correlation filters ourselves which can deal with grayscale as well as color images. Using this correlation filtering algorithm we are creating hybrid images which is a simplified version of SIGGRAPH 2006 paper by Oliva, Torralba, and Schyns [1].

1 Introduction

The first part of the assignment is implementing how to filter an image if we are given an image and a filter in matrix form, we did not use filters provided by different libraries but implemented our own filter which just correlated each part of the image with the filter progressively and then stores the new values in some other array which makes our new image.

Given the fitler applier is ready now we have to use this to generate hybrid images.

The key idea:::

If we are near an image then low frequencies are visible and when we are away from image high frequencies are prominently visible.

So if we can add two different images (which are kind of similar in some features like cat and dog or fist or submarine etc.) one with high frequencies and other with low frequencies then we can get an hybrid image which will show something else when nearby and some other thing when far away.

So here we implement *my_imfilter(image, filter)* function to filter out an image given the filter.

We first test it with some filters provided in proj1_test_filtering.py

After that we use this filtering function to correlate a gaussian and our images.

From there we find low frequencies and high frequencies.

Add these frequencies and save our images.

To visualize downscaling has been done which lets us to observe if the image is really hybrid.

Other functions used are normalize(image) as we were applying operations on images the values of the images were going out of bound, and this was causing an error to reslove the error we normalize the image, in other words map the image again to (0 to 1) in float.

Unlike matlab we did not have fspecial(gaussian) so we implemented our own gaussian_filter(shape, sigma) function which returns a float gaussian array.

© 2019. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

2 Methodology

We first show the implementation of the my_imfilter(image, filter) function.

In order to implement *my_imfilter(image, filter)*, we just correlate our given image matrix with the given filter which is just point wise multipication of the filter and each submatrix in the image for each pixel and then we store the sum of all the values of the matrix thus obtained and set the value of the output image's pixel as this one.

There is a bit of glitch here at boundaries as filter will definitely go out of bounds as when it goes to boundary or near boundary pixels so we have to pad our image matrix with appropriate number of zeroes so that when filter gets out of boundaries the values of zeroes gets multiplied.

Below is step wise approach in words form.

(1) First check if the dimensions of filter are both odd, else show error and return.

(2) Now initialize the output array in code it's filtered_image by copying the given image.

(3) The number rows to be padded on front and back are number of rows of filter divided by 2 as integer and similarly for columns. Hence we get the new dimension of padded array.

(4) Now check if the image is grayscale or rgb.

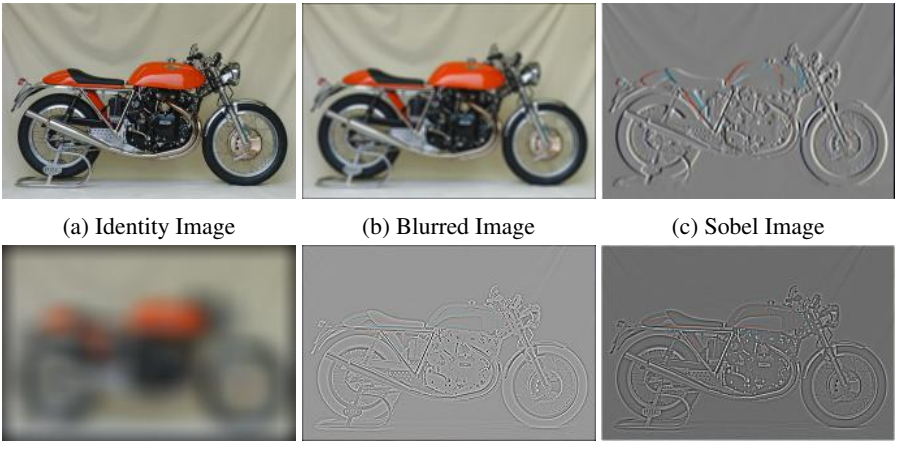
If image is rgb then we will intialize a 3d padded_array else a 2d padded_array.

Initialize the padded matrix and put the image in centre of it.

(3) Initialize an empty array of size same as image, Run the correlation/convolution (here correlation is done) algorithm on the image with the help of filter given. For each pixel point-wise multiply the subarray of size equal to filter's size with the filter get the sum and put the value of pixel in output as this sum.

(4) Finally return the output array which is our

Let's see some of the outputs our my_imfilter(image,filter)



(a) Identity Image (b) Blurred Image (c) Sobel Image

(d) Large Blur Image (e) Laplacian Image (f) High Pass Image

Figure 1: Different images obtained via proj1_test_filtering.py

Algorithm 1 my_imfilter(img, imfilter)

if $filter.width\%2 == 1 and filter.length\%2 == 1$ **then**
  $output \leftarrow img.copy()$
  $padding\_horizontal \leftarrow (filter\_length - 1)/2$
  $padding\_vertical \leftarrow (filter\_width - 1)/2$
  if $imgisgrayscale$ **then**
    $final\_shape \leftarrow (img\_length + 2 * pad\_length, img\_width + 2 * pad\_width)$
    $padded\_array \leftarrow matrix\_of\_zeroes(shape = final\_shape)$
    $padded\_matrix[pad\_length : img\_length + pad\_length, pad\_width : img\_width + pad\_width] \leftarrow img$
    for $i \leftarrow 0 : img\_length$ **do**
      for $j \leftarrow 0 : img\_width$ **do**
        $mult\_mat \leftarrow mat\_mult(padded\_matrix[i : i + fil\_length, j : j + fil\_width], imfilter)$
        $output[i, j] \leftarrow sum\_matrix(mult\_mat)$
      end for
    end for
  else
    $final\_shape \leftarrow (img\_length + 2 * pad\_length, img\_width + 2 * pad\_width, num\_ch)$
    $padded\_array \leftarrow matrix\_of\_zeroes(shape = final\_shape)$
    $padded\_matrix[pad\_length : img\_length + pad\_length, pad\_width : img\_width + pad\_width] \leftarrow img$
    for $k \leftarrow 0 : img\_channels$ **do**
      for $i \leftarrow 0 : img\_length$ **do**
        for $j \leftarrow 0 : img\_width$ **do**
          $mult\_mat \leftarrow mat\_mult(padded\_matrix[i : i + fil\_length, j : j + fil\_width, k], imfilter)$
          $output[i, j, k] \leftarrow sum\_matrix(mult\_mat)$
        end for
      end for
    end for
  end if
  return output
else
  $returnerror$
end if

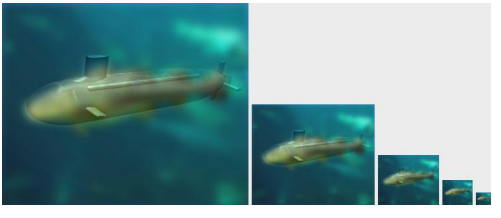
Algorithm 2 creating hybrid images

$img1, img2$
$cutoff\_frequency = acceptablevalue$
$gaussian\_dim \leftarrow cutoff\_frequency * 4 + 1$
$gaussian \leftarrow gaussianof(gaussian_dim, cutoff\_frequency)$
$lowfrequenies \leftarrow my\_imfilter(img1, imfilter)$
$higfrequencies \leftarrow img2 - my\_imfilter(img2, imfilter)$
$hybrid \leftarrow higfrequencies + lowfrequenies$
$normalize$
$downscalseandsave$

3 Results and Observations

Following are differen hybrid images obtained from the algorithm.

(a) Low Freq (b) High Freq (c) Hybrid Image



(d) Scales

Figure 2: Different images obtained from proj1.py using a pair of images(Fish-Submarine)

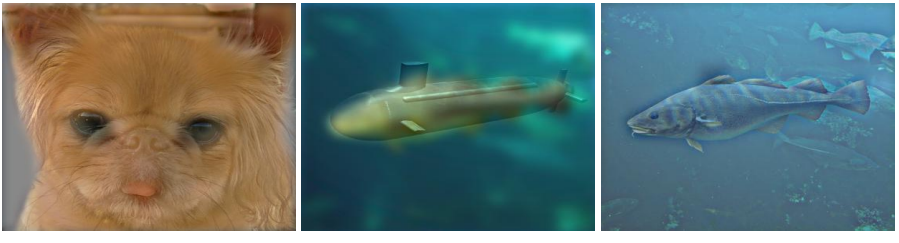
Hence we can see that our utility function for filtering is producing correct output and hence filters are used for a wide variety of applications including edge detection, blurring, sharpening etc.

Also in case of hybrid images, the combination of low frequencies and high frequencies of two different images give an effect as if there are two different images High frequencies mean an abrupt change in the value that is a sharp change Low frequencies mean a gradual smooth change. It's clear that eyes won't be ablt to detech smooth change from far away so only high frequencies are visible we have simulated this effect using scales as we scale down the image it gets far from us, all this is the essence of this assignment and we have verified all this with programming,maths and input data.

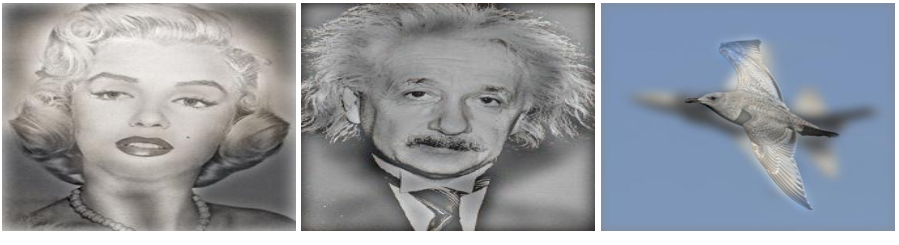
References

[1] Philippe. G. Schyns Aude Oliva, Antonio Torralba. Hybrid images. 2006.

(a) Bicycle-Motorcycle (b) Motorcycle-Bicycle (c) Dog-Cat



(d) Cat-Dog (e) Fish-Submarine (f) Submarine-Fish



(g) Einstein-Marilyn (h) Marilyn-Einstein (i) Plane-Bird

Figure 3: Different images obtained via proj1.py using a pair of images