Zewail City for science and technology
Computer Vision CIE 552
SPRING 2021

IMAGE HYBRIDIZATION:

Project documentation

Asmaa Shaban Ismail
ID: 201600780

DATE
Apr 04th, 2021

I. INTRODUCTION:

This project is an implementation of a simple image hybridization algorithm along with all of the required subroutines. Among the subroutines implemented is a function- “my_imfilter”- that applies a given filter kernel to an image in one of two ways; spatial domain convolution, or frequency domain filtering using FFT. The benefit gained from filtering in the frequency domain is a significant reduction in the computational complexity and runtime- as can be observed when comparing the performance of the function in the two modes. The reason for this improvement is that convolution in the frequency domain becomes a simple multiplication. The function can also apply the filter using convolution, or correlation; as determined by the user’s input. You also get to choose the padding method used in spatial convolution mode; the provided options are zero padding, reflect, edge and mean.

II. Subroutines usage and parameters:

def fft_filter(img, kernel, Visualize_process=False, img_name=''):

This function applies the filter kernel to the image in the frequency domain. It handles both grayscale and 3-channel RGB/BGR images.

:param Visualize_process: a boolean flag for visualization. Set True if you want to see the spectrum of the image, the mask, the filtered spectrum and the output filtered image. It outputs the aforementioned images both in pop-up windows using imshow, and as saved .jpg files in the results directory. Defaults to False.

:param img_name: provides an option to choose a unique name for the images saved in the results directory (if visualize is True). If not provided a generic name will be given and you risk the images being overwritten with every run.

:param img: the image to be filtered

:param kernel: the filter kernel to be used in the filtering

:return: the filtered image

def correlate_one_channel(original_image, padded_image, filter_kernel):

Mainly used as a helper subroutine in "my_imfilter". It performs the spatial domain correlation of a 2d image (grayscale/single channel) with a given filter kernel.

:param original_image: original (un-padded) 2d image to be filtered

:param padded_image: the same image padded to the appropriate size.

:param filter_kernel: the filter kernel to be used in the correlation/convolution process

:return: the filtered output (grayscale- single channel) image resulting from the convolution/correlation.

def my_imfilter(image: np.ndarray, filter_kernel: np.ndarray, padding_method='zero', method='convolution',
domain='spatial', im_name_fft="", visualize_fft=False):

filters the given image with the filter kernel. Supports grayscale and 3-channel RGB/BGR. You can choose the filtering method (convolution/ correlation) by setting the method flag. Can do either spatial domain convolution or FFT-based convolution according to the domain flag input. It also provides several options for the padding method which you can set using the padding_method flag.

:param visualize_fft: a visualization flag to show the spectra of images during fft calculations. Gets passed to fft_filter which is called within my_imfilter as a subroutine in case the domain was set to 'fft'.

:param im_name_fft: file name to give the spectrum images if visualize is True. Gets passed to fft_filter.

:param domain: set to 'spatial': (default) to filter using regular convolution/correlation in the time domain.
Set to 'fft': to filter in the frequency domain.

:param image: input image, gray scale or 3 channel RGB/BGR

:param filter_kernel: filter kernel

:param padding_method: supports 'zero', 'reflect', 'edge' or 'mean' padding. Defaults to ‘zero’.

:param method: filter application method, 'convolution' or 'correlation'

:return: the filtered image

def gen_hybrid_image(image1: np.ndarray, image2: np.ndarray, cutoff_frequency: float, ksize, Domain='spatial'):

Generates a hybrid image from two images. Uses my_imfiler as a subroutine

:param: image1 : The image from which to take the low frequencies.
:param: image2 : The image from which to take the high frequencies.
:param: cutoff_frequency : The standard deviation, in pixels, of the Gaussian blur which will remove high frequencies.
:param: method: convolution method, 'spatial' for regular spatial domain convolution or 'fft' for frequency domain.

III. Results:

Part 1:

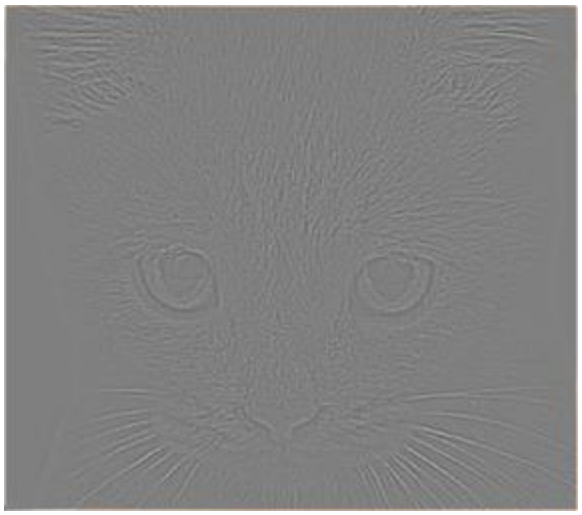
A. Results from the spatial convolution:

1- image passed through an identity filter (unchanged):

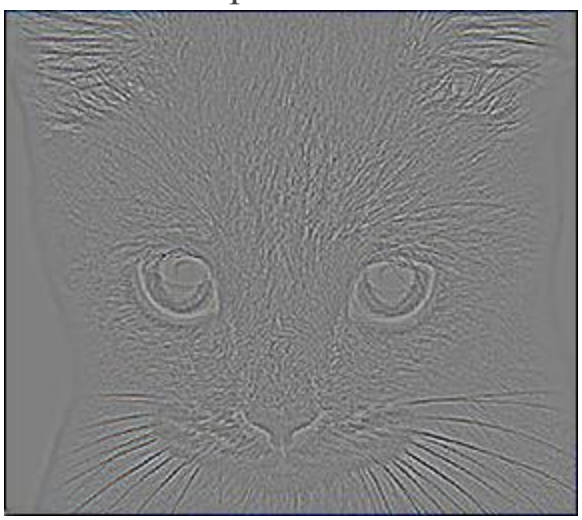
2- Significantly blurred by CV2 generated Gaussian kernel :

3- a little bit blurred by a box filter:

4- Simple High pass filter (original- blurred):



5- Discrete laplacian HPF:

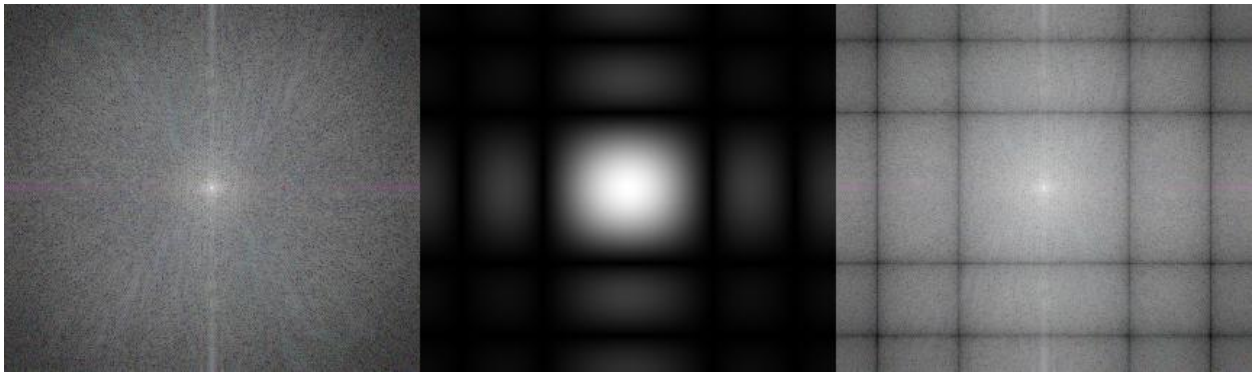


6- Sobel operator:

B. Testing the fft filtering method:

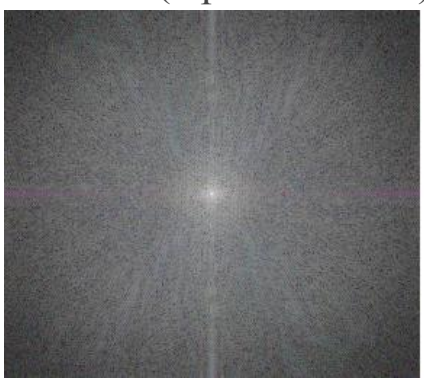
Gaussian blur FFT output:

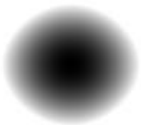
The image spectrum, the filter spectrum and the filtered spectrum in order:



Laplacian HPF output fft output (note I didn’t add 0.5):

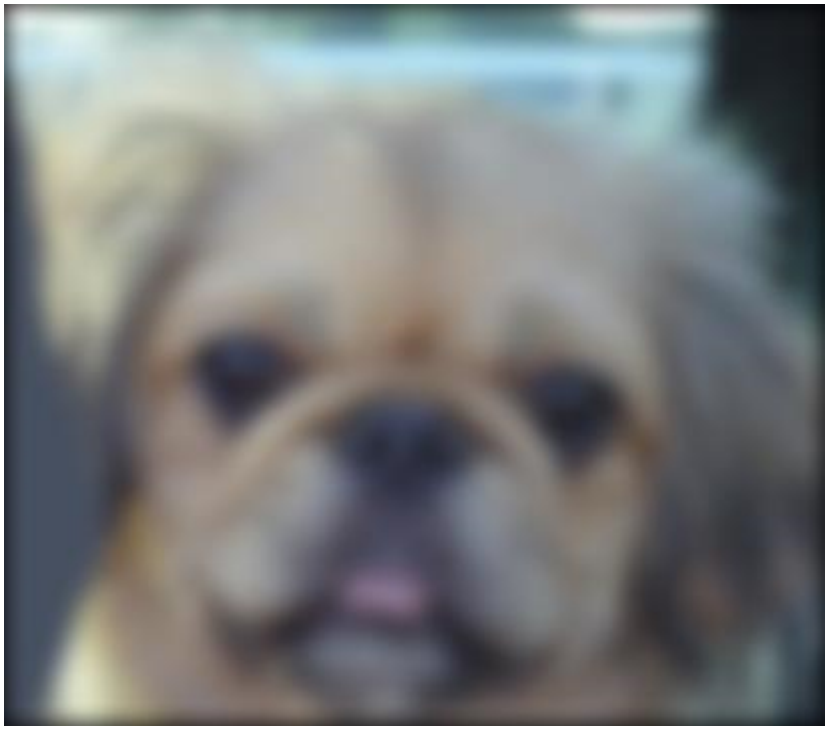
The image spectrum, the filter spectrum and the filtered spectrum in order: (laplacian filter)





Part 2 : Hybridization

A- Puppy and Kitten:
cutoff_frequency = 6 kernel_size = 23
1- Low pass image:



2- High pass Image:

3- Hybrid Image:

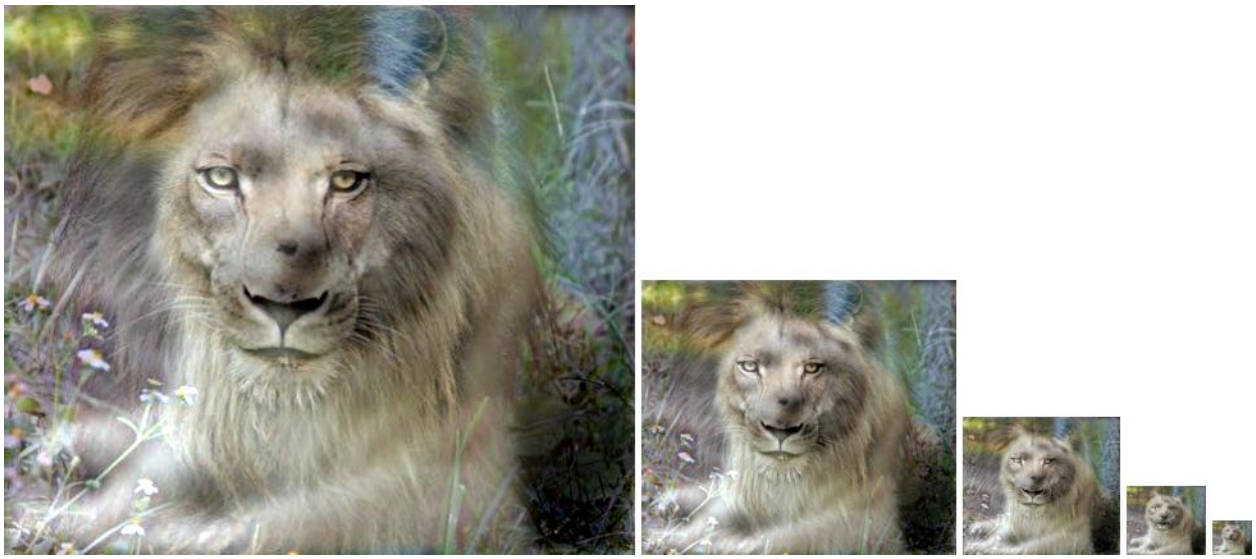
4-Visualization:

B. Using my own images:
Cameron and Zabu are an unlikely- but a very happy- couple of a male African lion and a white bengal tigress. They live at the Big Cat Rescue sanctuary in Florida where they're known as “Cambu”.

Original images vs the hybrid:

Hybrid :

Visualized:



Cam and Zabu being adorable:


BIGCAT RESCUE
BigCatRescue.org