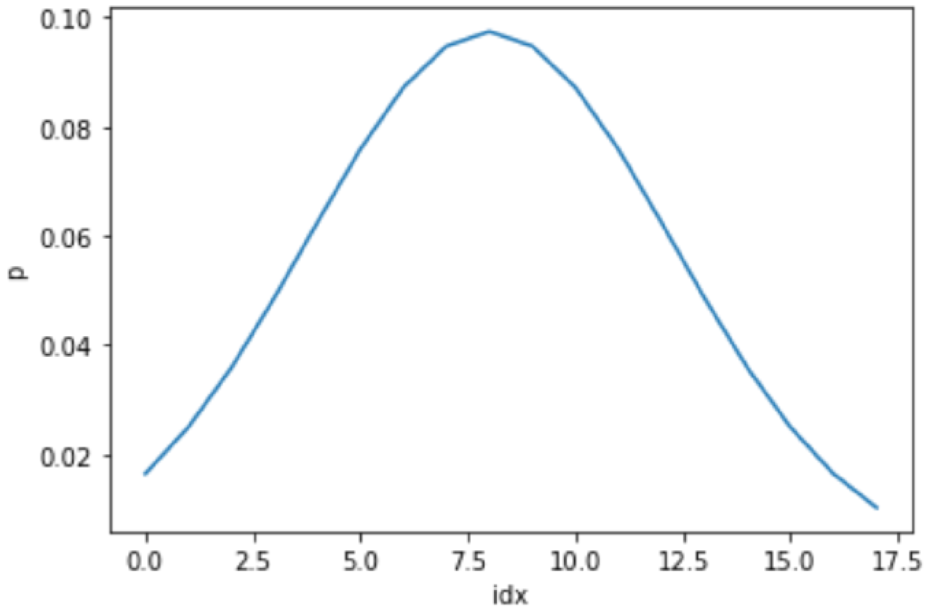
CS x476 Project 1

Zhaodong Yang
halyang@gatech.edu
zyang645
903748903

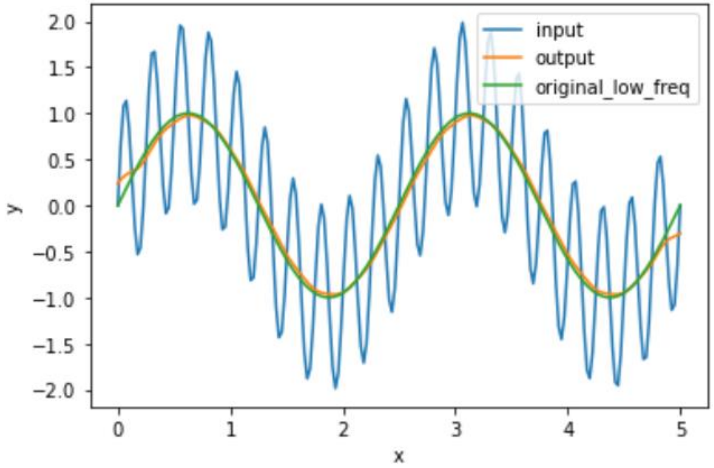
Part 1: 1D Filter

<insert visualization of the low-pass filter from proj1.ipynb here>



Part 1: 1D Filter

<insert visualization of filtered combined signal from proj1.ipynb here>

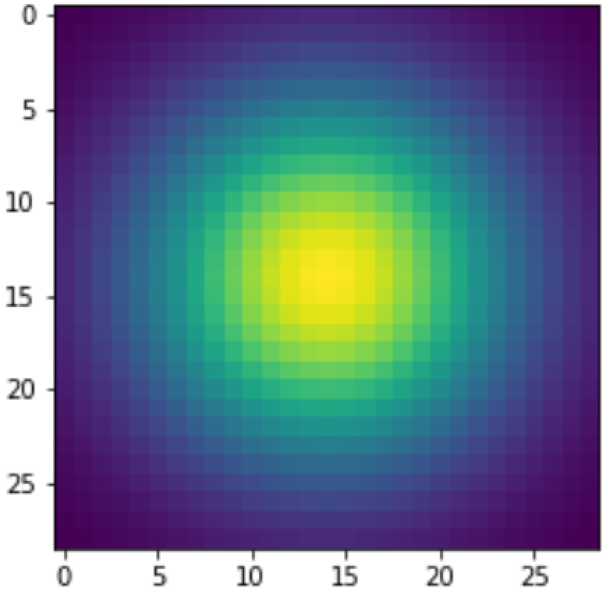


Describe your implementation in words and reflect on the checkpoint questions.

I slided the kernel from the beginning of the padded signal to the end, making the output signal the cross-correlation of the combined signal and the kernel. The filter attenuated the high-frequency signal by a large magnitute, and the low-frequency signal was relatively unaffected. The unit test passed.

Part 2: Image Filtering

<insert visualization of the 2D Gaussian kernel from proj1.ipynb here>



<Describe your implementation of my_imfilter() in words.>

I created a 1D Gaussian kernel using the former function. Then I calculated the outer product of the 1D kernel and itself. After normalizing the result, I got the 2D Gaussian kernel.

Part 2: Image filtering

Identity filter

<insert the results from proj1_test_filtering.ipynb using 1b_cat.bmp with the identity filter here>

Small blur with a box filter

<insert the results from proj1_test_filtering.ipynb using 1b_cat.bmp with the box filter here>

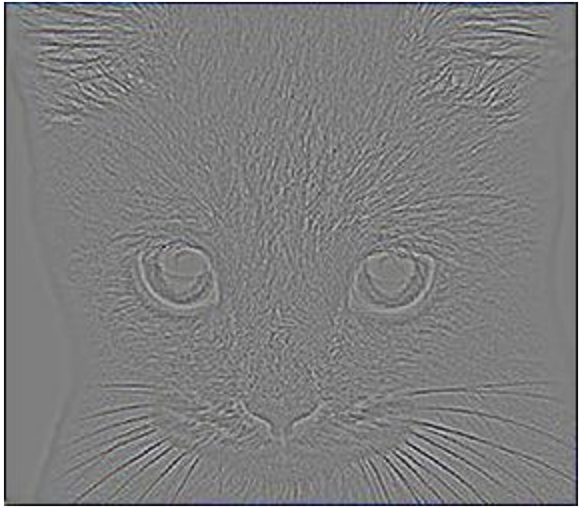
Part 2: Image filtering

Sobel filter

<insert the results from proj1_test_filtering.ipynb using 1b_cat.bmp with the Sobel filter here>

Discrete Laplacian filter

<insert the results from proj1_test_filtering.ipynb using 1b_cat.bmp with the discrete Laplacian filter here>



Part 2: Hybrid images manually using Pytorch

<Describe your implementation of create_hybrid_image() here.>

I used my_imfilter to create low frequency parts of the two images. Then I minused image2 by low frequency image of image2 to get its high frequency image. Then I added the high frequency image of image2 with the low frequency image of image1 and clamped the output image. Then I got the hybrid image.

Cat + Dog

Cutoff frequency: 7

Part 2: Hybrid images manually using Pytorch

Motorcycle + Bicycle

Cutoff frequency: 7

Plane + Bird

Cutoff frequency: 7

Part 2: Hybrid images manually using Pytorch

Einstein + Marilyn

Cutoff frequency: 7

Submarine + Fish

Cutoff frequency: 7

Part 3: Hybrid images with PyTorch operators

Cat + Dog

Motorcycle + Bicycle

Part 3: Hybrid images with PyTorch operators

Plane + Bird

Einstein + Marilyn

Part 3: Hybrid images with PyTorch operators

Submarine + Fish

Part 1 vs. Part 2

<Compare the run-times of Parts 1 and 2 here, as calculated in proj1.ipynb. What can you say about the two methods?>

The run-time of part 1 is 27.792, and the run-time of part 2 is 1.054. Part 2 is way much faster than part 1. It should be because torch.nn.functional.conv2d() is faster than my_imfilter().

Tests

Anaconda Prompt - jupyter-notebook proj1_code/proj1_test_filtering.ipynb

```
(cv_proj1) D:\Gatech\2021Fall\ComputerVision\proj1_release\proj1_release>pytest proj1_code/proj1_unit_tests
================================================= test session starts =================================================
platform win32 -- Python 3.6.13, pytest-5.0.1, py-1.10.0, pluggy-0.13.1
rootdir: D:\Gatech\2021Fall\ComputerVision\proj1_release\proj1_release
collected 16 items

proj1_code\proj1_unit_tests\test_2d.py ......                                                                   [ 37%]
proj1_code\proj1_unit_tests\test_create_1d_gaussian_kernel.py ...                                               [ 56%]
proj1_code\proj1_unit_tests\test_dft.py ..                                                                      [ 68%]
proj1_code\proj1_unit_tests\test_median_filter.py ...                                                           [ 87%]
proj1_code\proj1_unit_tests\test_my_1d_filter.py ..                                                             [100%]

================================================== warnings summary ===================================================
C:\Users\Hal Yang\AppData\Local\conda\conda\envs\cv_proj1\lib\site-packages\torchvision\io\video.py:2
  C:\Users\Hal Yang\AppData\Local\conda\conda\envs\cv_proj1\lib\site-packages\torchvision\io\video.py:2: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
    import imp

-- Docs: https://docs.pytest.org/en/latest/warnings.html
======================================= 16 passed, 1 warnings in 16.57 seconds ========================================
```

Conclusions

The change of cutoff standard deviation value is not so obvious. But I can feel the low frequency image becomes more dominant when I set the cutoff standard deviation value higher. But most of the time the high frequency image is more dominant to the whole content of the image than the low frequency image, which means you always see the high frequency image in the hybrid image at the first glance. So it does matter a lot when we swapping images within a pair. For these project. I think the most challenges comes from the huge workload and learning Pytorch as a novice. It took me four days(about 35 hours) doing nothing but proj1 to complete it.

Note

The following slide is:

- REQUIRED for **6476** students
- Extra credits for **4476** students.

Image Filtering using DFT

I transferred the DFT formula to matrix multiplication : F = UfU. Then I could use the DFT matrix function dft_matrix() to generate a DFT matrix U and multiply it with the image to get the DFT of the image. Then I set the value of the pixel which is near the edge of the DFT tensor to zero to cut off high frequency part. After that I use f = UFU again to get the reverse DFT of the DFT tensor, which is the image without high frequency part.

Note

The following slide is:

- Extra credit for **ALL** (4476+6476)

Add some cool hybrid images!