CV HW2 - Sarah Brown

March 13, 2021

1 ECE 5973 - Computer Vision

1.1 Homework 2

1.1.1 Sarah Brown

2 Question 1

Picture one is a picture of a sign and picture two is a picture of my friend holding up a notebook.

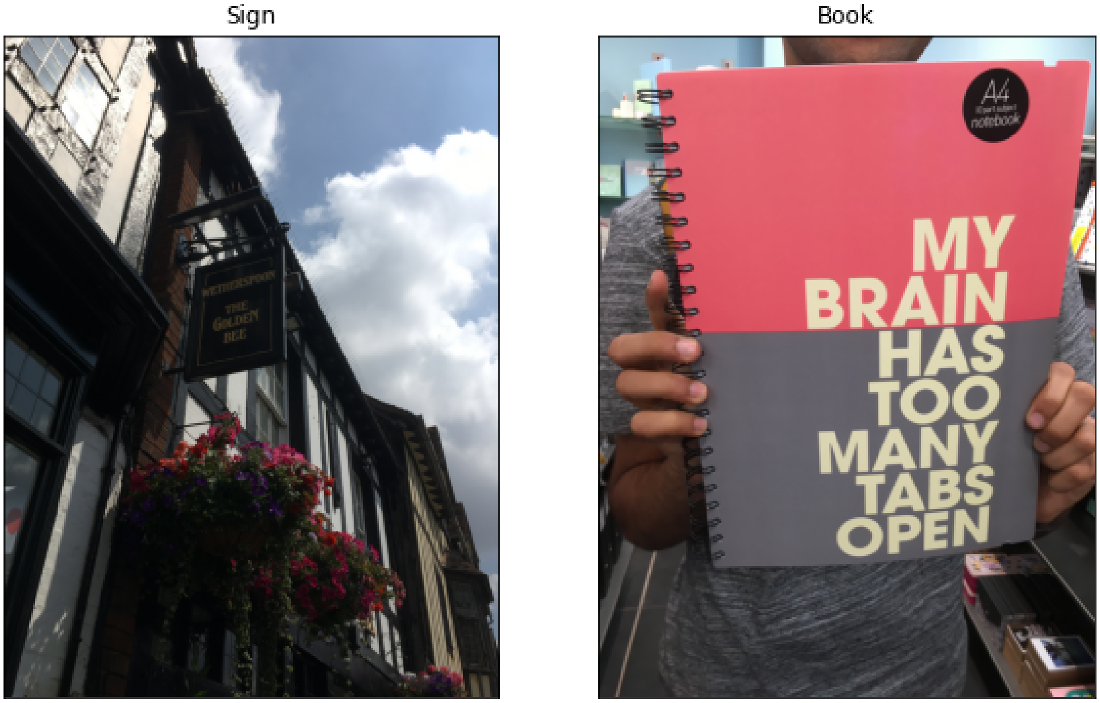
2.1 Question Details

(10 points) Please use two photos of your own (please don't use stock photos) to create a hybrid image (see this). Basically you just need to add a low-pass filtered image of one photo with a high-pass filtered image of another one. The simplest approach is probably approximating a low-pass filter with a Gaussian filter and a complementary high-pass filter with (1—"Gaussian filter"). That is, we can obtain a high-pass filtered image by subtracting the original image by a low-pass filtered image. Of course, you may also achieve something similar by playing with the fourier transformed images or discrete cosine transformed (DCT) images also.

[2]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
import cvui
%matplotlib inline

sign = cv2.imread('sign.JPG')
book = cv2.imread('book.JPG')

rgb_sign = cv2.cvtColor(sign, cv2.COLOR_BGR2RGB)
rgb_book = cv2.cvtColor(book, cv2.COLOR_BGR2RGB)

plt.figure(figsize=(10,10))
plt.subplot(121), plt.imshow(rgb_sign), plt.title("Sign"), plt.xticks([]), plt.
 ↪yticks([])
plt.subplot(122), plt.imshow(rgb_book), plt.title("Book"), plt.xticks([]), plt.
 ↪yticks([])
```

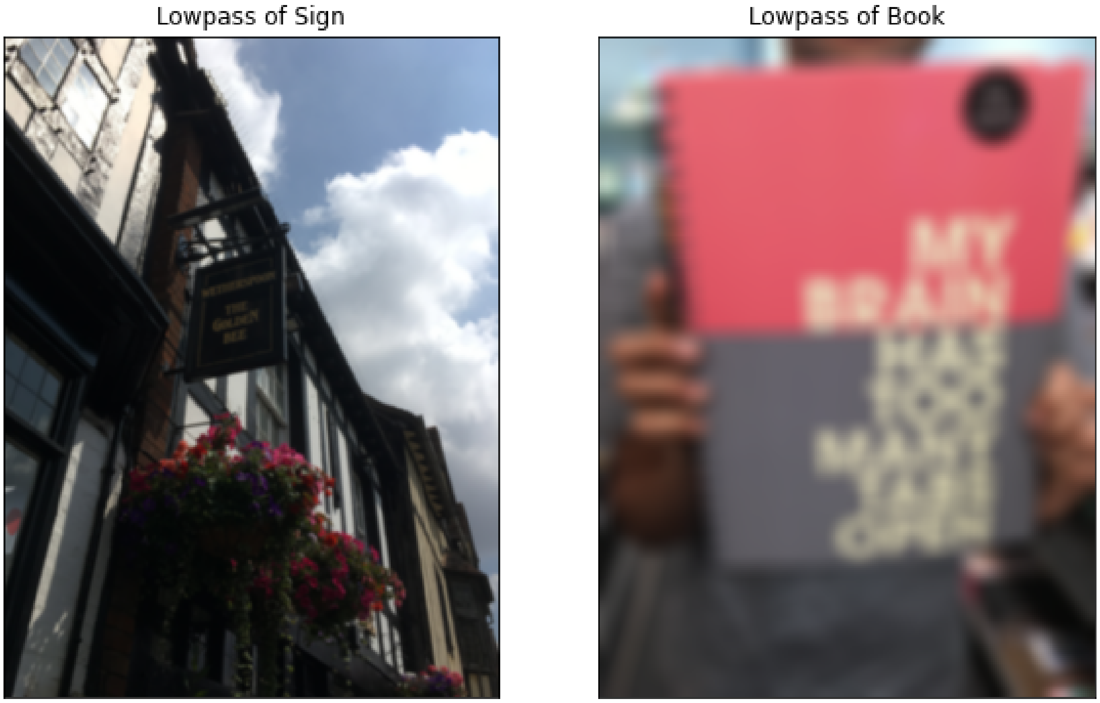
```
[2]: (<AxesSubplot:title={'center':'Book'}>,
      <matplotlib.image.AxesImage at 0x7fca30c60cd0>,
      Text(0.5, 1.0, 'Book'),
      ([], []),
      ([], []))
```



```
[3]: sign_low = cv2.GaussianBlur(rgb_sign, (27,27), 9)
     book_low = cv2.GaussianBlur(rgb_book, (165,165), 55)
     sign_high = rgb_sign - sign_low
     book_high = rgb_book - book_low

     plt.figure(figsize=(10,10))
     plt.subplot(121), plt.imshow(sign_low), plt.title("Lowpass of Sign"), plt.
      ↪xticks([]), plt.yticks([])
     plt.subplot(122), plt.imshow(book_low), plt.title("Lowpass of Book"), plt.
      ↪xticks([]), plt.yticks([])

     plt.figure(figsize=(10,10))
     plt.subplot(121), plt.imshow(sign_high+128), plt.title("Highpass of Sign"), plt.
      ↪xticks([]), plt.yticks([])
     plt.subplot(122), plt.imshow(book_high+128), plt.title("Highpass of Book"), plt.
      ↪xticks([]), plt.yticks([])

     combined = book_low + sign_high
```

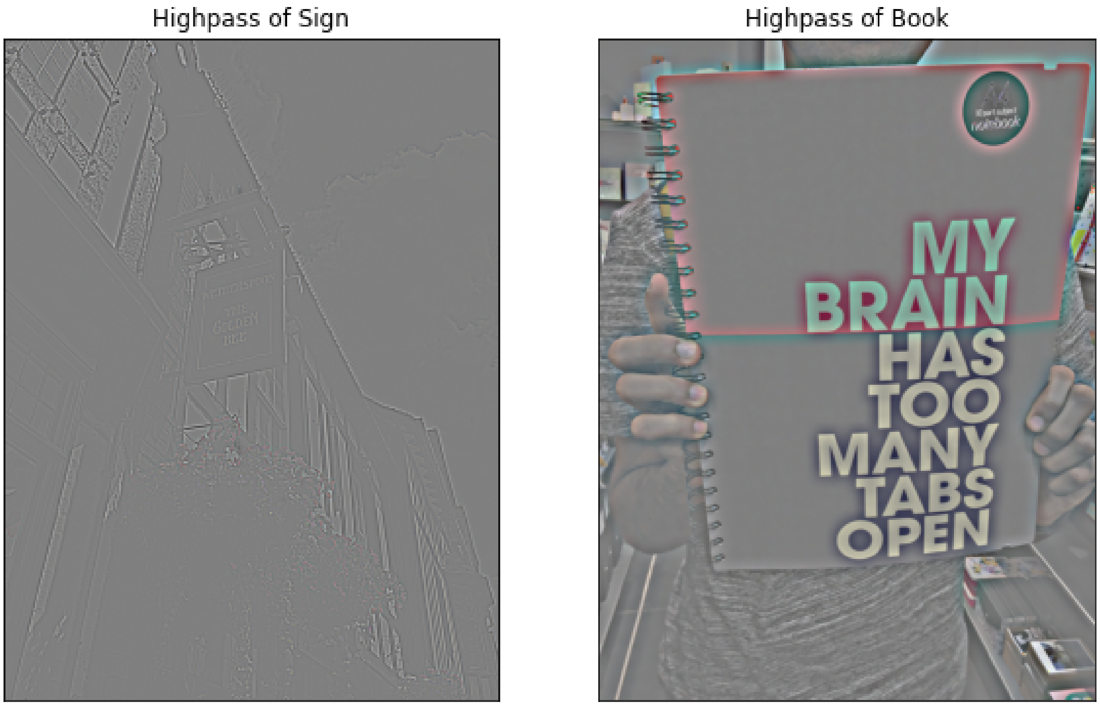
```
plt.figure(figsize=(10,10))
plt.imshow(combined), plt.xticks([]), plt.yticks([])
```

```
[3]: (<matplotlib.image.AxesImage at 0x7fca30ac6cd0>, ([], []), ([], []))
```



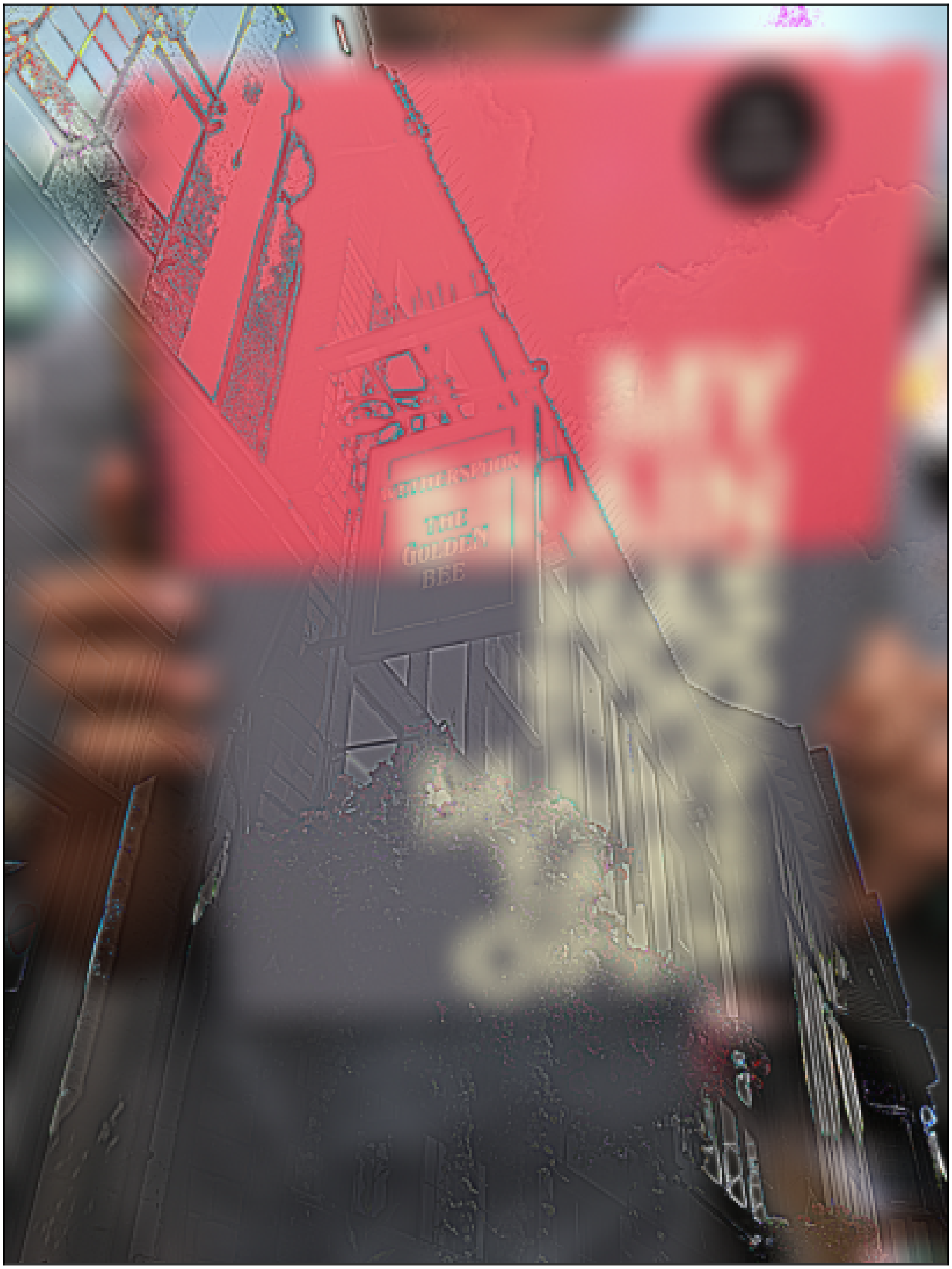
Lowpass of Sign

Lowpass of Book



Highpass of Sign

Highpass of Book



[ ]:

```
sign_low = cv2.GaussianBlur(rgb_sign, (27,27), 9)
book_low = cv2.GaussianBlur(rgb_book, (165,165), 55)
sign_high = rgb_sign - sign_low
```

```
book_high = rgb_book - book_low

plt.figure(figsize=(10,10))
plt.subplot(121), plt.imshow(sign_low), plt.title("Lowpass of Sign"), plt.
 ↪xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(book_low), plt.title("Lowpass of Book"), plt.
 ↪xticks([]), plt.yticks([])

plt.figure(figsize=(10,10))
plt.subplot(121), plt.imshow(sign_high+128), plt.title("Highpass of Sign"), plt.
 ↪xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(book_high+128), plt.title("Highpass of Book"), plt.
 ↪xticks([]), plt.yticks([])

greyFinal = cv2.

plt.figure(figsize=(10,10))
plt.imshow(book_low+sign_high)
```

3 Question 2

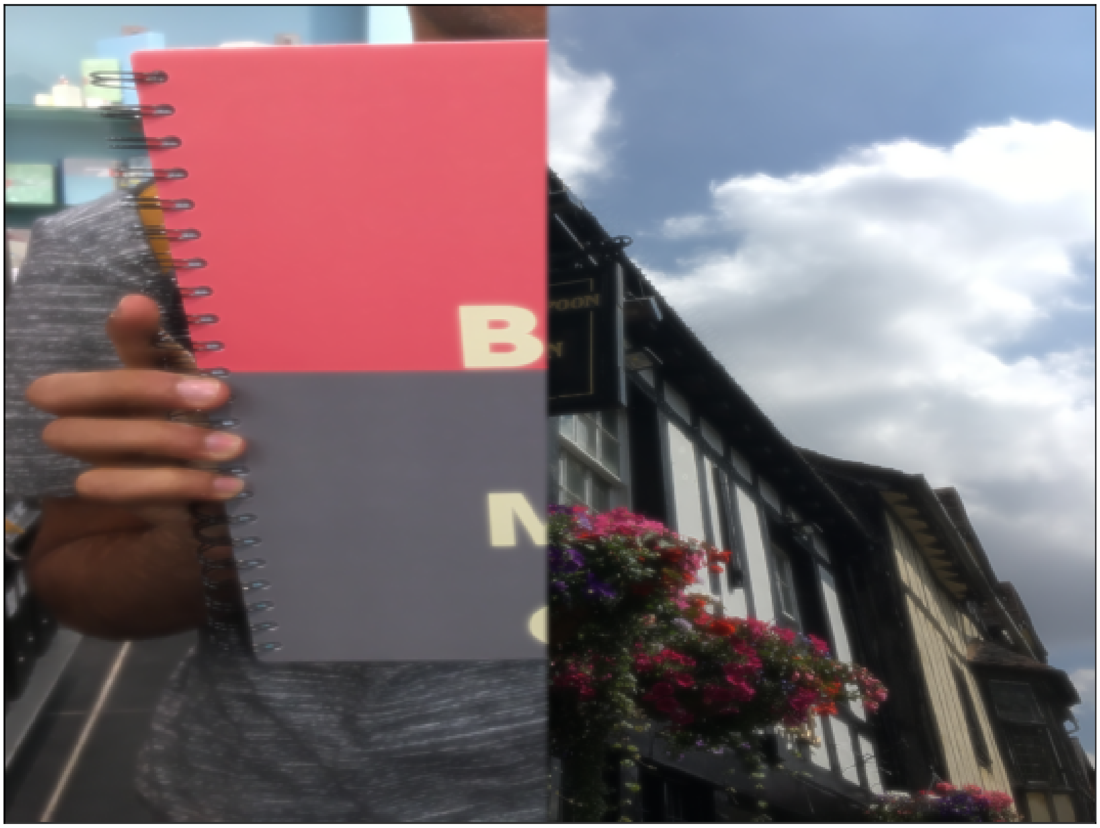
(10 points) Please use the same photos to create a composting image with one side from one image and another side from another image. Please decompose your images into a Laplacian pyramids with 5 levels

[4]:

```
sign = cv2.imread('sign.JPG')
book = cv2.imread('book.JPG')

dimension = (3840,2880)
book32 = cv2.resize(book, dimension, interpolation = cv2.INTER_AREA)
sign32 = cv2.resize(sign, dimension, interpolation = cv2.INTER_AREA)

gaussBook = book32.copy()
gaussPyBook = [gaussBook]

for i in range(5):
    gaussBook = cv2.pyrDown(gaussBook)
    gaussPyBook.append(gaussBook)

gaussSign = sign32.copy()
gaussPySign = [gaussSign]

for i in range(5):
    gaussSign = cv2.pyrDown(gaussSign)
    gaussPySign.append(gaussSign)
```

```
laplaceBook = [gaussPyBook[4]]
for i in range (4,0,-1):
    gaussExtendB = cv2.pyrUp(gaussPyBook[i])
    lB = cv2.subtract(gaussPyBook[i-1], gaussExtendB)
    laplaceBook.append(lB)

laplaceSign = [gaussPySign[4]]
for i in range (4,0,-1):
    gaussExtendS = cv2.pyrUp(gaussPySign[i])
    lS = cv2.subtract(gaussPySign[i-1], gaussExtendS)
    laplaceSign.append(lS)

laplacePyramid = []

for bo, si in zip(laplaceBook, laplaceSign):
    rows, cols, dpt = bo.shape
    ls = np.hstack((bo[:,0:int(cols/2)], si[:,int(cols/2):]))
    laplacePyramid.append(ls)

reconstruct = laplacePyramid[0]

for i in range(1,5):
    reconstruct = cv2.pyrUp(reconstruct)
    reconstruct = cv2.add(reconstruct, laplacePyramid[i])


reconstructRGB = cv2.cvtColor(reconstruct, cv2.COLOR_BGR2RGB)
plt.figure(figsize=(10,10))
plt.imshow(reconstructRGB), plt.xticks([]), plt.yticks([])
```

```
[4]: (<matplotlib.image.AxesImage at 0x7fca30aaa700>, ([], []), ([], []))
```



4 Question 2 - Bonus Credit

Extra credit (5 points). Create a trackbar to vary the number of levels of decomposition as shown in class. You can use the cvui package.

[5]:
```
window_name = "joined images"
level = 5
n = [level]
bar_xloc = 10
bar_yloc = 10
bar_length = 100
min_n = 1
max_n = 5

sign = cv2.imread('sign.JPG')
book = cv2.imread('book.JPG')

dimension = (3840,2880)
book32 = cv2.resize(book, dimension, interpolation = cv2.INTER_AREA)
sign32 = cv2.resize(sign, dimension, interpolation = cv2.INTER_AREA)
```

```
gaussBook = book32.copy()
gaussPyBook = [gaussBook]

for i in range(5):
    gaussBook = cv2.pyrDown(gaussBook)
    gaussPyBook.append(gaussBook)

gaussSign = sign32.copy()
gaussPySign = [gaussSign]

for i in range(5):
    gaussSign = cv2.pyrDown(gaussSign)
    gaussPySign.append(gaussSign)

laplaceBook = [gaussPyBook[4]]
for i in range (4,0,-1):
    gaussExtendB = cv2.pyrUp(gaussPyBook[i])
    lB = cv2.subtract(gaussPyBook[i-1], gaussExtendB)
    laplaceBook.append(lB)

laplaceSign = [gaussPySign[4]]
for i in range (4,0,-1):
    gaussExtendS = cv2.pyrUp(gaussPySign[i])
    lS = cv2.subtract(gaussPySign[i-1], gaussExtendS)
    laplaceSign.append(lS)

laplacePyramid = []

for bo, si in zip(laplaceBook, laplaceSign):
    rows, cols, dpt = bo.shape
    ls = np.hstack((bo[:,0:int(cols/2)], si[:,int(cols/2):]))
    laplacePyramid.append(ls)

reconstruct = laplacePyramid[0]
for i in range(1,5):
    reconstruct = cv2.pyrUp(reconstruct)
    reconstruct = cv2.add(reconstruct, laplacePyramid[i])

reconstructSmall = cv2.resize(reconstruct, (960,720))
cvui.init(window_name)
while True:
    trackbar = cvui.trackbar(reconstructSmall, bar_xloc, bar_yloc, bar_length,␣
 ↪n, min_n, max_n, 1, '%.0Lf')
    cv2.imshow(window_name, reconstructSmall)
    if trackbar:
        reconstruct = laplacePyramid[0]
        for i in range(1,int(n[0])):
            reconstruct = cv2.pyrUp(reconstruct)
            reconstruct = cv2.add(reconstruct, laplacePyramid[i])
        reconstructSmall = cv2.resize(reconstruct, (960,720))

    if cv2.waitKey(1) & 0xFF == ord('q'): # btw, you need to click the screen␣
 ↪first. And then
                                     # press q to quit
        break
cv2.destroyAllWindows()
```

[6]:

```
reconstructSmallRGB = cv2.cvtColor(reconstructSmall, cv2.COLOR_BGR2RGB)
plt.figure(figsize=(10,10))
plt.imshow(reconstructSmallRGB), plt.xticks([]), plt.yticks([])
```

[6]:

```
(<matplotlib.image.AxesImage at 0x7fca35d99e80>, ([], []), ([], []))
```