note 98 views

HW1: Histograms, Filters, Deconvolution, Blending

The goal in this assignment is to get you acquainted with filtering in the spatial domain as well as in the frequency domain.
Laplacian Blending using Image Pyramids is a very good intro to working and thinking in frequencies, and Deconvolution is a neat trick.

You tasks for this assignment are:

- Perform Histogram Equalization on the given input image.
- Perform Low-Pass, High-Pass and Deconvolution on the given input image.
- Perform Laplacian Blending on the two input images (blend them together).

Histogram Equalization

Refer to the readings on @43, particularly to Szeliski's section 3.4.1, and within it to eqn 3.9.
Getting the histogram of a grayscale image is incredibly easy (Python):

```
h = np.histogram(im, 256)
# or
h = cv2.calcHist(...)
```

Your image is color, so split it to it's channels with `cv2.split()`, and work on each channel.
A histogram is a vector of numbers. If you wish to visualize it use either

```
pyplot.hist(im, bins=256) #this will calculate the histogram and visualize it
# or if you pre calculated your histogram:
pyplot.bars(...) # https://stackoverflow.com/questions/5328556/histogram-matplotlib
```

Once you have the histogram, you can get the cumulative distribution function (CDF) from it

```
cdf = np.cumsum(h)
```

Then all you have left is to find the mapping from each value [0,255] to its adjusted value (just using the CDF basically).
Validate your results vs. OpenCV's `equalizeHist()` function.
Do not use `cv2.equalizeHist()` directly to solve the exercise!
We will expect to see in your code that you get the PDF and CDF, and that you manipulate the pixels directly (avoid a for loop, though).

Low-Pass, High-Pass and Deconvolution in the Frequency Domain

This is an easy one, just follow the tutorials and you should be fine:
http://docs.opencv.org/master/de/dbc/tutorial_py_fourier_transform.html
http://docs.opencv.org/2.4/doc/tutorials/core/discrete_fourier_transform/discrete_fourier_transform.html

For your LPF, mask a **20x20** window of the center of the FT image (the low frequencies).
For the HPF - just reverse the mask.

For deconvolution, all you are required to do is apply a gaussian kernel (gk) to your input image (in the FD/FFT):

```
gk = cv2.getGaussianKernel(21,5)
gk = gk * gk.T

def ft(im, newsize=None):
    dft = np.fft.fft2(np.float32(im),newsize)
    return np.fft.fftshift(dft)

def ift(shift):
    f_ishift = np.fft.ifftshift(shift)
    img_back = np.fft.ifft2(f_ishift)
    return np.abs(img_back)

imf = ft(im, (im.shape[1],im.shape[1])) # make sure sizes match
gkf = ft(gk, (im.shape[1],im.shape[1])) # so we can multiple easily
imconvf = imf * gkf

# now for example we can reconstruct the blurred image from its FT
blurred = ift(imconvf)
```

Using these simple helper functions I provided you can probably now see how to go the other way around, given an already convolved image, to do deconvolution (use division instead of multiplication).

Note: please use the following image instead of `blurred2.png`: blurred2.exr
To load it (as float32 single-channel):

```
a  = cv2.imread("blurred2.exr", cv2.IMREAD_ANYCOLOR | cv2.IMREAD_ANYDEPTH)
```

Laplacian Pyramid Blending

This tutorial will tell you everything you need to know about LPB:
http://docs.opencv.org/3.2.0/dc/dff/tutorial_py_pyramids.html
It translates very literally to C++.

Make sure you make images rectangular and equal size:

```
# make images rectangular
A = A[:,:A.shape[0]]
B = B[:A.shape[0],:A.shape[0]]
```

Submission Guidelines
These are the images and skeleton code packet you should be using to solve this assignment:
HW1Filters.zip
Inputs are numbered:

- input1.jpg -- for hist-eq,
- input2.png -- for LPF and HPF, blurred2.png -- for de-conv, and
- input3A.jpg, input3B.jpg -- for LPB.

Please submit working code (cpp or py) and results numbered as follows:

- output1.png -- for hist-eq,
- output2LPF.png, output2HPF.png, output2deconv.png -- for FT assignment, and
- output3.png -- for LPB

Read the instructions in the skeleton codes carefully and only change the functions allowed to be changed. Keep the output functions intact.
Put all result images into a folder named "Results" and all source files into a folder called "Source". Put both folders and other things you would like to send in a master folder named after your FirstName_LastName_SBUID,
e.g. Frodo_Baggins_11483269 , zip this file and upload it through blackboard as Frodo_Baggins_11483269.zip
If this is still not clear, download and read the submission-instructions under general resources.

Note that we are using an automatic reading system. Failing to comply to these requirements may cause a delay grading your homework. And a wrong order of result images will result in losing a lot of points. It is highly recommended to test your program
under VM before submission. Codes not working normally under VM will also lead to points off.

Submit on Blackboard, **Due 9/21 9:00am.**
#pin

hw1 week3

Updated 11 hours ago by Roy Shilkrot and Fan Wang

followup discussions *for lingering questions and comments*

[✓ Resolved] Unresolved

Alex Scarlatos 3 days ago
For the histogram equalization, opencv has this tutorial: http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_histograms/py_histogram_equalization/py_histogram_equalization.html

Is it okay if we use the code they provide there, as long as we understand & comment it?

Roy Shilkrot 3 days ago You may use that (or any other) tutorial, yes, as long as you achieve the goal and understand what you're doing.

[✓ Resolved] Unresolved

Anonymous 2 days ago
Regarding deconvolution:

It seems as though the code snippet provided does implement blurring. Here is an executable version of the snippet:

```
import os
import sys
import cv2
import numpy as np

im = cv2.imread('img.png', 0)

gk = cv2.getGaussianKernel(21,5)
gk = gk * gk.T

def ft(im, newsize=None):
    dft = np.fft.fft2(np.float32(im),newsize)
    return np.fft.fftshift(dft)

def ift(shift):
    f_ishift = np.fft.ifftshift(shift)
    img_back = np.fft.ifft2(f_ishift)
    return np.abs(img_back)

imf = ft(im, (im.shape[0],im.shape[1])) # make sure sizes match
gkf = ft(gk, (im.shape[0],im.shape[1])) # so we can multiple easily
imconvf = imf * gkf

# now for example we can reconstruct the blurred image from its FT
blurred = ift(imconvf)

cv2.imshow('image',blurred)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

The only part I modified was making the dimensions correct in the ft() call.

Deconvolution results in noise using this kernel and method (division and the theorems).

Can we please have clarification on this issue?

Anonymous 2 days ago Sorry, I meant to say that the code snippet **does not implement blurring** but made a typo.

Anonymous 1 day ago Deconvolution seems to work on an arbitrary image but not on the given image.

I believe this is due to the typo in the code fragment above which should be:

```
imf = ft(im, (im.shape[0],im.shape[1])) # make sure sizes match
gkf = ft(gk, (im.shape[0],im.shape[1])) # so we can multiple easily
```

Please confirm and provide us with the correct image.

Roy Shilkrot 11 hours ago The correct blurred image for deconvolution is now attached.

[✓ Resolved] Unresolved

Anurag Arora 1 day ago
I think we have to refer Szeliski's section 3.1.4 for Histograms, not "3.4.1".

[✓ Resolved] Unresolved

Anonymous 1 day ago
Is anyone able to run the file main.py? It says that we are only allowed to change some specific functions but the main.py uses some additional queries such as "sys.argv" which I cannot find anywhere.

@TAs: please provide instructions how we can run main.py.

Xiaofei Sun 1 day ago For me, if I want to run for Question 3, I run

```
python main.py 3 input3A.jpg  input3B.jpg ./
```

Fan Wang 1 day ago There is a function called help_message() inside the scripts. It is meant to explain how you can pass in the parameters and run the program. There is not much codes inside those scripts. Please read them carefully.

[✓ Resolved] Unresolved

Dibyajyoti 1 day ago hi Prof, in order to perform deconvolution we need first use the helper function provided above to perform gaussian blur then use the known filter (Gaussian)above to deconvolve in the function. Could you please confirm if I got this correct

Harsh Trivedi 19 hours ago I believe we only need to do deconvolution on given image (blurred2.png) assuming that blurred image was obtained by convolution using cv2.getGaussianKernel(21,5) kernel. However, am not sure. If it is not the case then, what is blurred2.png given for?

Harshad's Untwale 12 hours ago Should we output the deconvoluted image as 8 bit image or 24 bit image ?

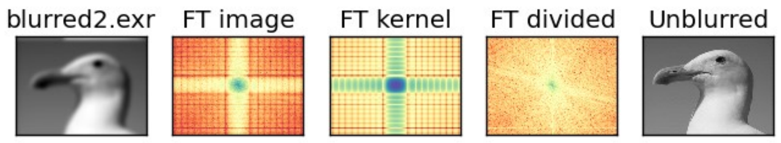
Dibyajyoti 12 hours ago yeah, I agree..i tried to deconvolution it, its leading some weird output!
So, I'll proceed as I understand, convolute the image then deconvolute using same Gaussian filter

Roy Shilkrot 11 hours ago See the latest addition as to which image to use as blurred input.

Dibyajyoti 11 hours ago prof, just to confirm that the kernel applied to the image in is with size=21 and sigma=5? Because dividing the blurred image with this filter isn't yielding anything valid

Roy Shilkrot 11 hours ago Just to confirm you are dividing in the frequency domain - right?

This is my result on the new input image 'blurred2.exr':



Dibyajyoti 11 hours ago yes, I'm operating in freq domain, even treating for zeros before division.
But I don't have a "blurred2.exr" in the zip attached above, is it from blackboard link?

Roy Shilkrot 11 hours ago It's now attached here on Piazza, in the instructions.

Dibyajyoti 10 hours ago opencv read call isn't reading exr files. I believe would need openEXR lib to read and proceed

Anonymous 10 hours ago Can you try using what is mentioned above in the instructions. The following seems to work for me
a = cv2.imread("blurred2.exr", cv2.IMREAD_ANYCOLOR | cv2.IMREAD_ANYDEPTH)

Also, the result is now as expected by using the .exr image

Dibyajyoti 4 hours ago yes it works now, only issue i'm concerned is the write back function.
The write function creates a file a 4KB, which I'm doubtful about as the original exr is 290K

But the imshow option seems to be working, are there any flags to be set while writing floating point numbers?

Anonymous 3 hours ago Check the type of the image after fft. If its anything else than unint8, you need to convert it.

Dibyajyoti 3 hours ago its a floating because the input in floating, but just cant convrt by multiplying 255 and range normalization. It ends up being a white image at the end

Dibyajyoti 3 hours ago writing out a numpy matrix along with the image file, I checked (converting it jpg using some tool) the 4K exr file written but there seems significant loss

Timothy Zhang 2 hours ago I have the same problem. cv2.imshow() and matplotlib both display the correct image but cv2.imwrite() expects uint8 valued arrays and there doesn't seem to be a clean conversion in Python that I can find...

Resolved [✓ Unresolved]

Anonymous 20 hours ago Regarding - "Your image is color, so split it to it's channels with cv2.split(), and work on each channel."
Without splitting can we use channels attribute in calchist ?

hist = cv2.calcHist([gray_img],[0],None,[256],[0,256])

Resolved [✓ Unresolved]

Dibyajyoti 2 hours ago Laplacian Pyramid: as per open cv doc pyrUp() does the upsampling then bluring opposed to what is mentioned in the slides and the algo. I m not sure if things change lot: as we are supposed to Downsample->Blur->Upsamples->subtract from just above layer of Gaussian pyramid
instead: we are Downsample->Upsamples->blur->subtract from just above layer of Gaussian pyramid

could someone please confirm if this is correct

[✓ Resolved] Unresolved

Anonymous 42 minutes ago
hi prof,
Do we need to convert RGB into YUV before histogram equalization or can we directly execute it on RGB?