گروه كامپيوتر - دانشكده مهندسي - دانشگاه فردوسي مشهد

مستندات گزارش تمرین پنجم

درس مبانی بینایی کامپیوتر

**ارائه ‌دهنده:**

**صالح شیروانی**

**استاد درس:**

**دکتر امیرحسین طاهری نیا**

**پاییز1401**

چکیده

در این تمرین قصد داریم به بررسی هرم ها بپردازیم. هرم های مهمی همچون گوسین، لاپلاسین و موجک را پیاده خواهیم کرد و به مقایسه خروجی ها می‌پردازیم و در نهایت نقش مهم هرم ها را در پردازش تصویر بررسی خواهیم کرد. همچنین در بخشی از تمرین به بررسی تبدیل موجک خواهیم پرداخت و خواهیم دید که این تبدیل چه اطلاعاتی را در اختیار ما قرار خواهد داد..

Technical Description

**5- Wavelet**

5-1- Pyramid

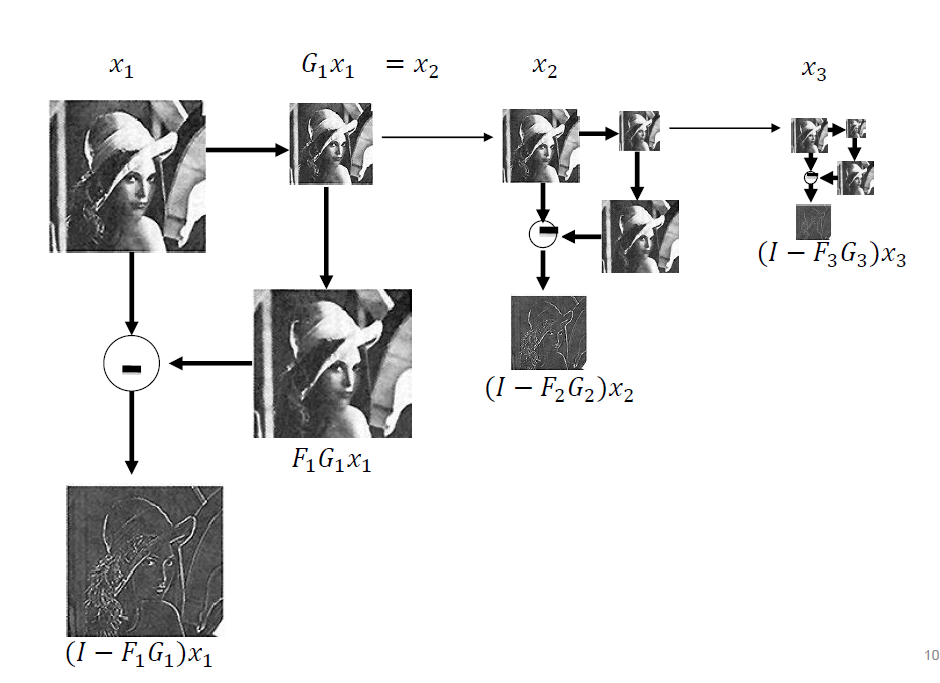
5-1-1- در این تمرین از ما خواسته شده تا هرم گوسین و لاپلاسین (DOG) را پیاده سازی کنیم. پیاده سازی این دو هرم در پایتون بسیار ساده است.

برای پیاده کردن هرم گوسین از تابع pyrDown در ماژول cv2 استفاده می‌کنیم. این تابع روی تصویر فیلتر گوسین اعمال می‌کند و سپس سایز تصویر را نصف می‌کند. این دقیقا همان چیزیست که برای پیاده سازی هرم گوسین نیاز داریم.

هرم لاپلاسین را با توجه به هرم گوسینی که حساب کردیم به دست می‌آوریم. برای به دست آوردن تصویر سطح ام در هرم لاپلاسین ابتدا تصویر موجود در سطح هرم گوسین را upscale می‌کنیم و سپس تفاضل این تصویر را با تصویر موجود در سطح هرم گوسین حساب می‌کنیم. برای upscale کردن تصویر از تابع pyrUp موجود در ماژول cv2 بهره می‌بریم.

نکته ای که باید به آن توجه کرد این است که در مواقعی که ابعاد تصویر عددی فرد است وقتی تصویر downscale می‌شود بخشی از اطلاعات از بین می‌رود و در نتیجه وقتی این تصویر upscale می‌شود ابعاد تصویر upscale شده با تصویر اولیه یکسان نخواهد بود و عملا نمی‌شود تفاضل دو تصویر را حساب کرد. برای حل این مشکل به تصویری که تعداد سطر یا ستون کمتری دارد خودمان سطر یا ستون اضافه می‌کنیم.

تصویر زیر نحوه محاسبه هرم لاپلاسین را نشان می‌دهد:

**

5-1-2- در gaussian smoothing، separability به خاصیت یک کرنل 2 بعدی اشاره دارد که می‌تواند به دو کرنل 1 بعدی جدا شود. یک کرنل برای smoothing در امتداد محور y و یک کرنل برای smoothing در امتداد محور x. یعنی می‌توان smoothing را به صورت جداگانه در راستای افقی و عمودی روی تصویر اعمال کرد. این یعنی به جای اینکه یک فیلتر دو بعدی در تصویر convolve شود میتوان دو تا فیلتر یک بعدی را با تصویر convolve کرد. این ویژگی می‌تواند به میزان قابل توجهی هزینه محاسباتی را به خصوص زمانی که سایز تصویر بزرگ است، کاهش دهد. این کاهش در محاسبات می‌تواند کمک زیادی به gaussian smoothing کند.

همچنین می‌توان این دو فیلتر یک بعدی را به صورت موازی به تصویر اعمال کرد تا سرعت را افزایش داد.

Pseudo-code طراحی شده به صورت زیر است: (این pseudo-code در فایل ipynb تمرین که کد های مربوط به تمرینات در آن نوشته شده است نیز در دسترس است.)

Input: image, levels, sigma

Output: pyramid

pyramid = [image]

for i = 1 to levels:

    blurred\_x = separableGaussianBlur(image, sigma)

    blurred\_xy = separableGaussianBlur(blurred\_x, sigma)

    image = downSample(blurred\_xy)

    pyramid.append(image)

    sigma = sqrt(2) \* sigma

return pyramid

function separableGaussianBlur(image, sigma)

    kernel = create1DGaussianKernel(sigma)

    blurred\_x = convolve(image, kernel)

    blurred\_xy = convolve(blurred\_x, transpose(kernel))

    return blurred\_xy

function downSample(image)

    return image resized by half

5-1-3- حداکثر تعداد سطحی که approximation pyramid می‌تواند داشته باشد به صورت زیر است:

*اگرحداکثر تعداد سطح (level) های هرم برایر با l باشد داریم:*

*بنابراین حداکثر تعداد سطحی که هرم approximation می‌تواند داشته باشد برابر با j+1 است.*

*تعداد کل پیکسل های موجود در هرم به صورت زیر است:*

*همواره این مقدار بزرگتر یا مساوی تعداد پیکسل ها در تصویر اصلی است. (تعداد پیکسل های تصویر اصلی برابر است)*

*به دلیل تناظری که بین prediction pyramid و approximation pyramid وجود دارد می‌توان ادعا کرد حداکثر تعداد سطحی که prediction pyramid می‌تواند داشته باشد برابربا حداکثر تعداد سطوح approximation pyramid است. بنابراین حداکثر تعداد سوطح prediction pyramid برابر با j+1 سطح است.*

*تعداد کل پیکسل ها در prediction pyramid برابر است با:*

*دلیلش این است که در prediction pyramid تعداد پیکسل ها در هر سطح برابر با تعداد پیکسل ها در تصویر اصلی یعنی است.*

*بنابراین این مقدار همواره بزرگتر یا مساوی تعداد پیکسل ها در تصویر اصلی است.*

*با این وجود استفاده ازprediction pyramid فواید زیر را در پی دارد:*

* *کارایی محاسباتی: با نمایش تصویر در سطوح چندگانه جزئیات، prediction pyramid امکان پردازش کارآمد تصاویر را به خصوص در هنگام برخورد با تصاویر بزرگ فراهم می کنند.*
* *اندازه داده کاهش می‌یابد: از آنجایی که approximation pyramid پیش بینی تنها باقیمانده بین تصویر اصلی و پیش بینی آن تصویر را در سطح قبلی ذخیره می کند، در مقایسه با ذخیره تصویر کامل در هر سطح، میزان داده مورد نیاز برای نمایش تصویر را کاهش می دهد.*
* *نمایش بهتر تصویر: با نمایش تصاویر در سطوح مختلف جزئیات، هرم های prediction نمایش جامع تری از یک تصویر ارائه می دهند و تجزیه و تحلیل و پردازش تصویر را آسان تر می کنند.*

*همچنین برخی فواید استفاده از approximation pyramid به صورت زیر است:*

* *فشرده سازی تصویر: می توان برای بهبود فشرده سازی تصویر با فشرده سازی تقریب در هر سطح از هرم استفاده کرد.*
* *بازنمایی تصویر: approximation pyramid با نمایش تصاویر در سطوح مختلف جزئیات، نمایش جامع تری از یک تصویر ارائه می دهند و تجزیه و تحلیل و پردازش تصویر را آسان تر می کنند.*
* *فیلتر کردن تصویر: هرم های approximation را می توان برای فیلتر کردن کارآمد تصویر استفاده کرد، زیرا فیلتر کردن را می توان در چندین سطح از جزئیات انجام داد که هرکدام دارای وضوح کمتری نسبت به سطح قبلی هستند.*

*5-1-4- در این تمرین باید approximation pyramid و prediction pyramid پیاده سازی کنیم. طبق صورت سوال باید از averaging filter با سایز برای approximation pyramid استفاده کنیم که برای این کار از تابع blur از ماژول cv2 استفاده شده است. این تابع یک box filter روی تصویر اعمال می‌کند. سپس تصویر را downscale می‌کنیم و هرم approximation را می‌سازیم.*

*برای ساخت هرم prediction در واقع باید هرم لاپلاسین را پیاده کنیم که نحوه پیاده سازی این هرم در سوال 1-1-5 به طور کامل شرح داده شد و اینجا هم از همان روش استفاده می‌کنیم. تنها نکته ای که باقی می‌ماند این است که گفته شده در این سوال از pixel replication برای درونیابی استفاده شود که این کار را به کمک تابع resize از cv2 انجام می‌دهیم. این تابع یک پارامتر به نام interpolation دارد که روش درونیابی را مشخص می‌کند.*

*برای pixel replication مقدار cv2.INTER\_NEAREST را به عنوان آرگومان برای این تابع ارسال می‌کنیم.*

*5-1-5- در این تمرین باید تبدیل موجک روی تصویر اعمال کنیم. برای اینکار از تابع dwt2 از ماژول pywt (pyWavelet) استفاده می‌کنیم. طبق صورت سوال باید برای تبدیل موجک از فیلتر Haar استفاده کنیم. برای این منظور رشته ‘haar’ را به عنوان آرگومان برای dwt2 ارسال می‌کنیم.*

*خروجی این تابع یک تاپل است که فرمت (approx., (HL,LH,HH)) را دارد. که approx. تصویر approximation و HL، تصویر HL است که حاوی لبه های افقی تصویر اصلی می‌باشد و LH، تصویرLH و در نهایت HH، تصویر HH می‌باشد.*

*HH: horizontal high pass & vertical high pass*

*LH: horizontal low pass & vertical high pass*

*HL: horizontal high pass & vertical low pass*

**Results**

**5- Wavelet**

5-1- Pyramid

5-1-1-

def get\_down\_gussian\_pyramid(*image*, *NOLevels*):

    layer = *image*.copy()

    pyramid = []

    for i in range(*NOLevels*):

        pyramid.append(layer)

*# pyrDown() decreases the size to half.*

        layer = cv2.pyrDown(layer)

    return pyramid

number\_of\_pyramid\_levels = 5

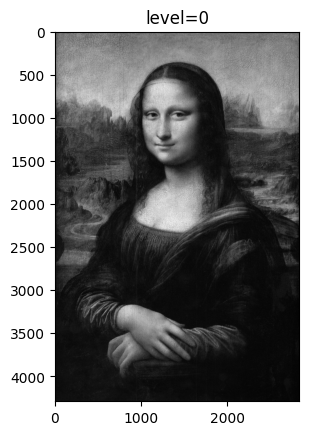
pyramid = get\_down\_gussian\_pyramid(mona\_lisa, number\_of\_pyramid\_levels)

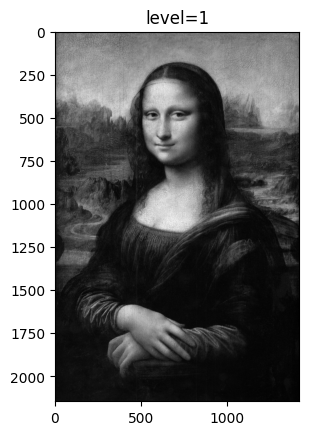
for i in range(number\_of\_pyramid\_levels):

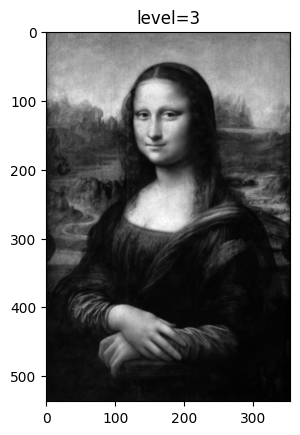
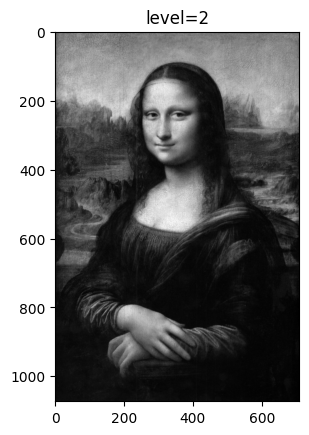
    plot\_image(pyramid[i], f'level={i}')

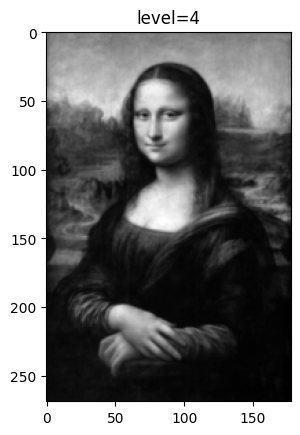
خروجی این تمرین به صورت زیر است:

Gaussian pyramid









Laplacian pyramid

def get\_laplacian\_pyramid(*image*, *NOLevels*):

    img = *image*.copy()

*# Create the Gaussian pyramid*

    gaussian\_pyramid = get\_down\_gussian\_pyramid(img, *NOLevels*)

*# Create the Laplacian pyramid*

    last\_level\_of\_gaussian = gaussian\_pyramid[*NOLevels* - 1]

    laplacian\_pyramid = [last\_level\_of\_gaussian]

    for i in range(*NOLevels* - 1, 0, -1):

        expanded\_img = cv2.pyrUp(gaussian\_pyramid[i])

        img = gaussian\_pyramid[i-1].copy()

*# check if number of rows are not equal*

        if img.shape[0] != expanded\_img.shape[0]:

            img = cv2.copyMakeBorder(

                img, 0, 1, 0, 0, cv2.BORDER\_CONSTANT, *value*=(0, 0, 0))

*# check if number of columns are not equal*

        if img.shape[1] != expanded\_img.shape[1]:

            img = cv2.copyMakeBorder(

                img, 0, 0, 0, 1, cv2.BORDER\_CONSTANT, *value*=(0, 0, 0))

        laplacian = cv2.subtract(img, expanded\_img)

        laplacian\_pyramid.append(laplacian)

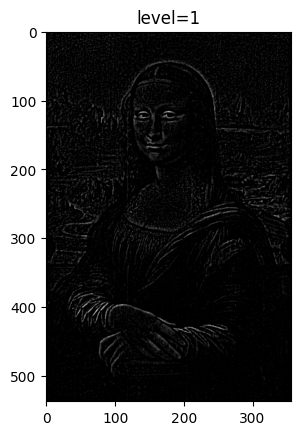
    return laplacian\_pyramid

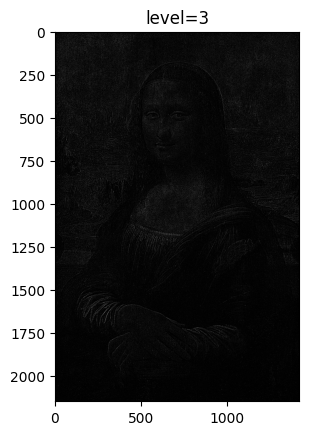
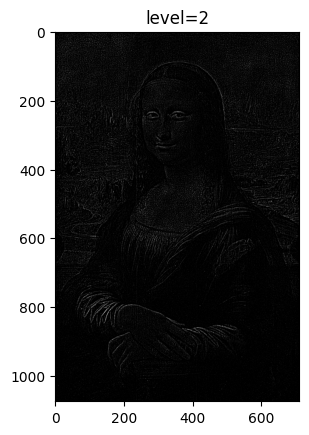
number\_of\_pyramid\_levels = 5

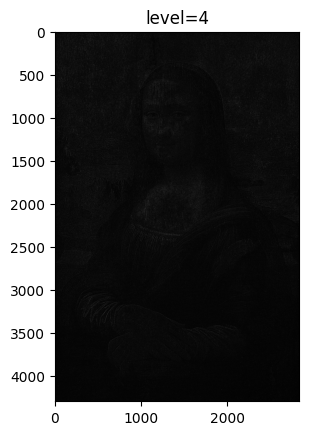
pyramid = get\_laplacian\_pyramid(mona\_lisa, number\_of\_pyramid\_levels)

for i in range(number\_of\_pyramid\_levels):

    plot\_image(pyramid[i], f'level={i}')







*تمامی این تصاویر در پیوست در اختیار شما قرار گرفته است. در صورت پایین بودن کیفیت عکس های بالا می‌توانید از تصاویر ذخیره شده استفاده نمایید.*

*هرم های گاوسی ابزار مفیدی در پردازش تصویر هستند، زیرا به شما امکان می دهند تصاویر را به روش های مختلف دستکاری کنید و تصاویر را در مقیاس های مختلف نشان دهید. آنها همچنین جزء اساسی چندین الگوریتم بینایی کامپیوترهستند، از جمله تشخیص اشیا، دید استریو، و image blendimg و ...*

*با توجه به این تمرین یکی از کاربرد های مهم هرم لاپلاسین در فشرده سازی تصاویر است. زیرا این هرم جزئیات با فرکانس بالای تصویر را دور می‌اندازد. این دور انداختن جزئیات با فرکانس بالا کمک زیادی به حذف نویز از تصویر نیز می‌کند. همچنین با توجه به تصاویر بالا واضح است که می‌توان از هرم لاپلاسین برای شارپ کردن تصویر و افزایش وضوح آن استفاده کرد.*

*5-1-2- همانطور که در شرح فنی مساله گذشت* می‌توان smoothing را به صورت جداگانه در راستای افقی و عمودی روی تصویر اعمال کرد. این یعنی به جای اینکه یک فیلتر دو بعدی در تصویر convolve شود میتوان دو تا فیلتر یک بعدی را با تصویر convolve کرد. این ویژگی می‌تواند به میزان قابل توجهی هزینه محاسباتی را به خصوص زمانی که سایز تصویر بزرگ است، کاهش دهد. این کاهش در محاسبات می‌تواند کمک زیادی به gaussian smoothing کند.

همچنین می‌توان این دو فیلتر یک بعدی را به صورت موازی به تصویر اعمال کرد تا سرعت را افزایش داد.

Pseudo-code خواسته شده برای این تمرین در قسمت شرح فنی (technical description) مربوط به این مساله در همین گزارش و همچنین در نوتبوک مربوط به کد های مساله در دسترس است.

5-1-3- محاسبات و توضیحات مربوط به این تمرین به طور کامل در قسمت شرح فنی (technical description) مربوط به این مساله در همین گزارش نوشته شده است.

با توجه به توضیحات مطرح شده در شرح فنی، حتی زمانی که مجموع پیکسل های موجود در هرم از تعداد پیکسل های تصویر اصلی بیشتر است استفاده از هرم می‌تواند فواید مهمی همچون فشرده سازی تصویر، کارایی محاسباتی، بازنمایی بهتر تصویر و... را به ارمغان آورد.

5-1-4-

img = lena.copy()

*# Create a list to hold the pyramids*

approx\_pyramid = []

pred\_residual\_pyramid = []

*# Add the original image to the list*

*# approx\_pyramid.append(img)*

*# pred\_residual\_pyramid.append(img)*

*# Set the number of levels*

levels = 3

*# Create the pyramids*

for i in range(levels):

*# Down-sample the current image using 2x2 averaging*

    current\_img=cv2.blur(*src*=img,*ksize*=(2,2))

    current\_img = current\_img[::2, ::2]

    approx\_pyramid.append(current\_img)

*# Up-sample the previous level of the approximation pyramid using pixel replication*

    upsampled\_img = cv2.resize(

        approx\_pyramid[i], (img.shape[1], img.shape[0]), *interpolation*=cv2.INTER\_NEAREST)

*# Predict the current image by subtracting the upsampled version of the previous level*

    prediction = cv2.subtract(img, upsampled\_img)

*# Add the prediction to the current level of the approximation pyramid*

    pred\_residual\_pyramid.append(prediction)

    img = current\_img.copy()

titles = []

for i in range(levels+1):

    titles.append('prediction, level={}'.format(i))

plt.figure(*figsize*=(20, 20))

subplot(levels, 1, pred\_residual\_pyramid, titles)

titles = []

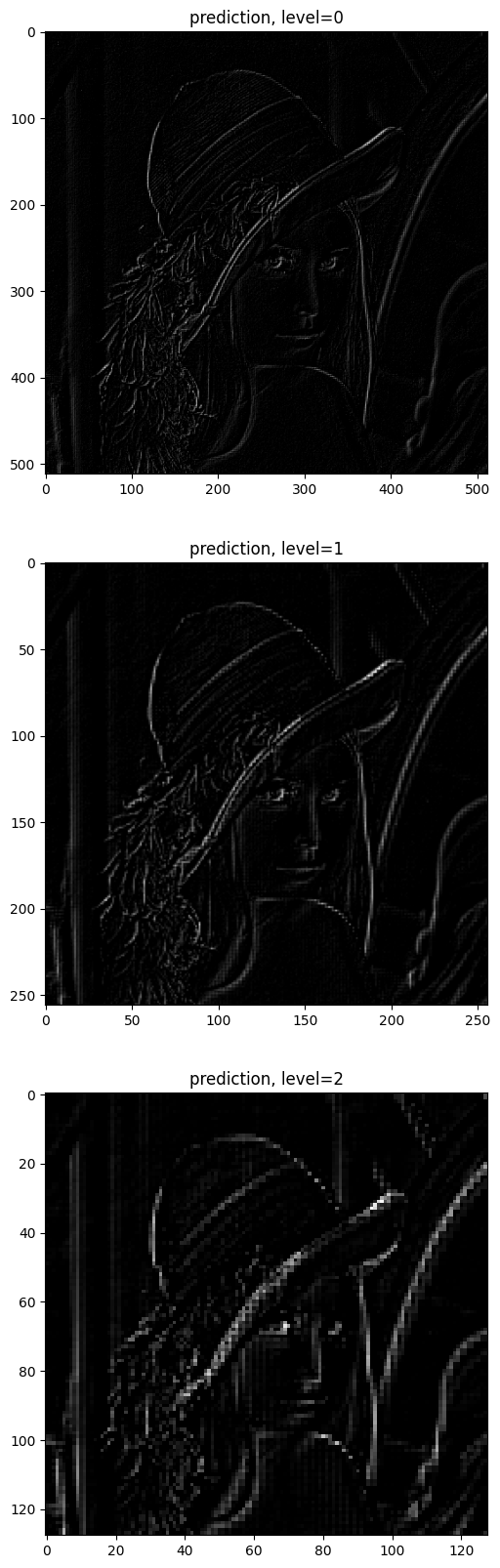
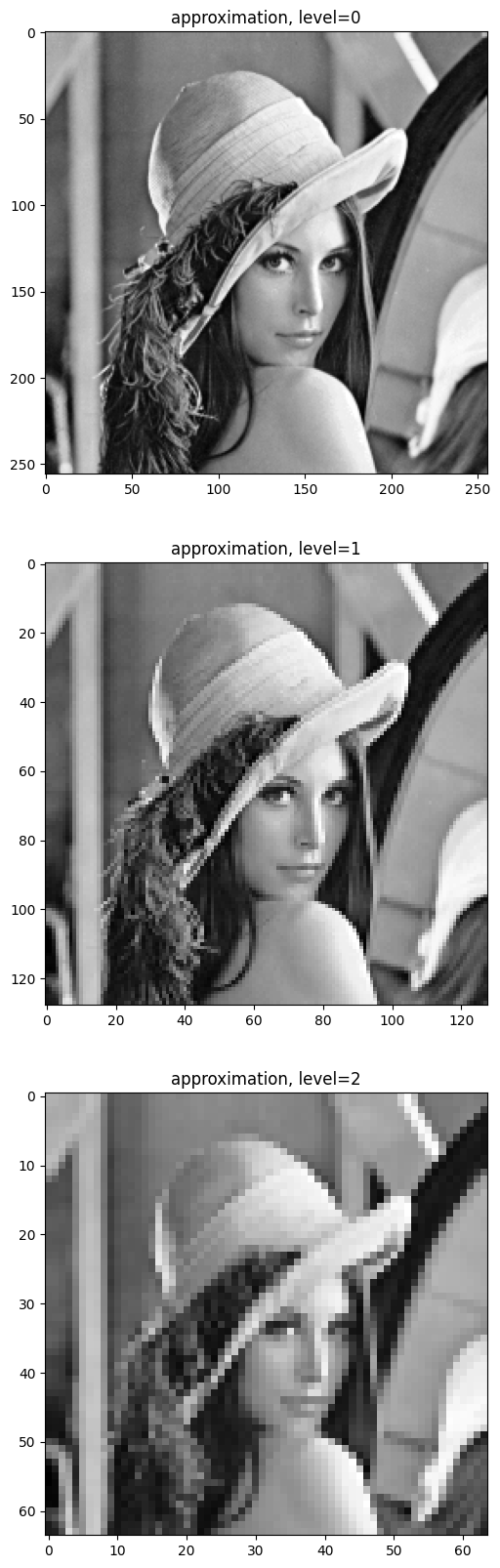
for i in range(levels+1):

    titles.append('approximation, level={}'.format(i))

plt.figure(*figsize*=(20, 20))

subplot(levels, 1, approx\_pyramid, titles)

تصاویر خروجی حاصل از این تمرین در صفحه بعد آمده است.



5-1-5-

def get\_wavelet\_pyramid(*image*, *NOLevels*):

    pyramid = []

    for i in range(*NOLevels*):

*# Perform the wavelet transform*

        coeffs = pywt.dwt2(*image*, 'haar')

*# The output is a tuple of (cA, (cHL, cLH, cHH))*

*# cA is the approximation coefficients*

*# cHL is the horizontal detail coefficients*

*# cLH is the vertical detail coefficients*

*# cHH is the diagonal detail coefficients*

        pyramid.append(coeffs)

*image* = coeffs[0].copy()

    return pyramid

number\_of\_pyramid\_levels = 3

pyramid = get\_wavelet\_pyramid(lena, number\_of\_pyramid\_levels)

for i in range(number\_of\_pyramid\_levels):

    (approx, (HL, LH, HH)) = pyramid[i]

    coeffs = [approx, HL, LH, HH]

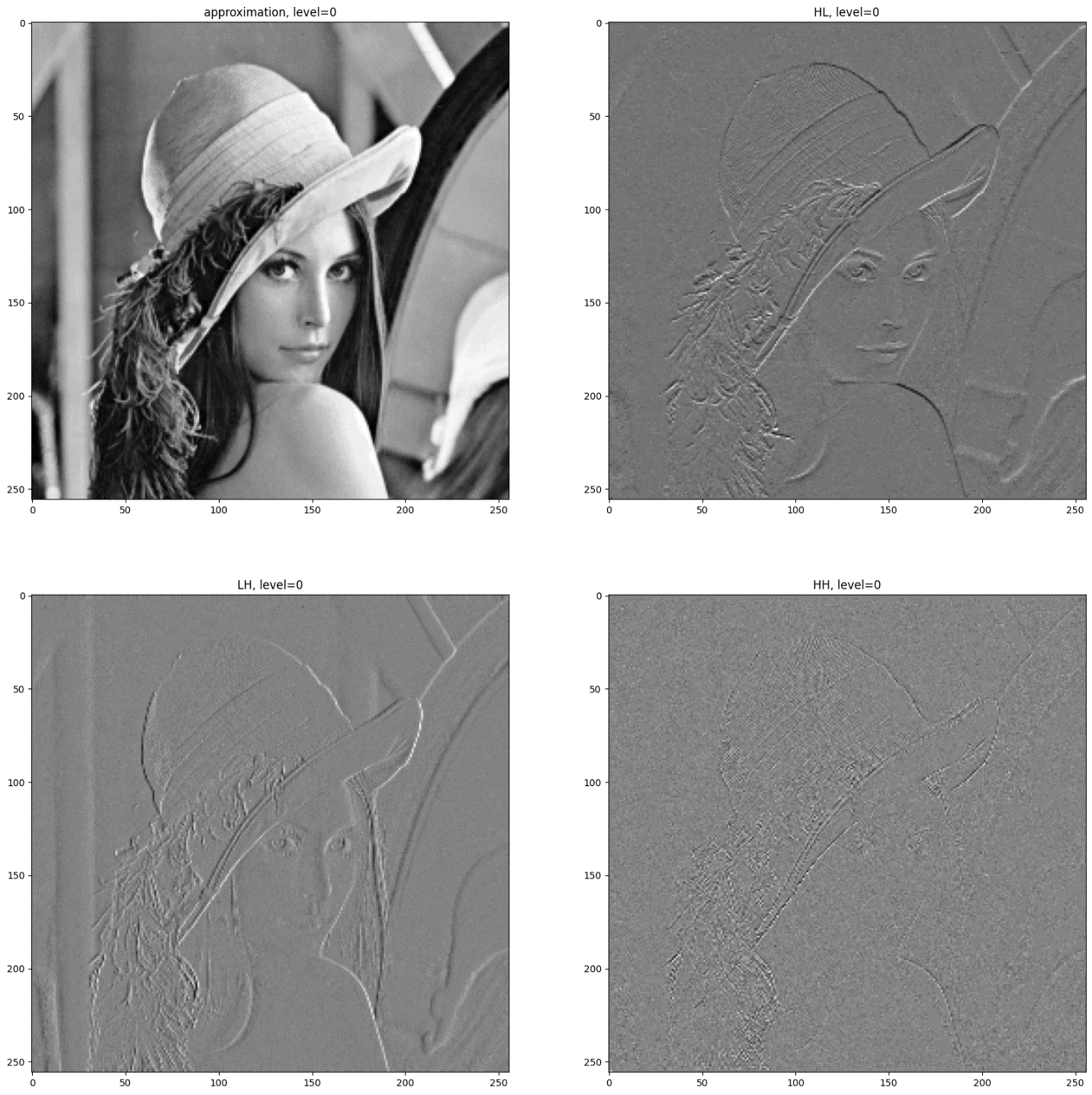
    titles = [f'approximation, level={i}',

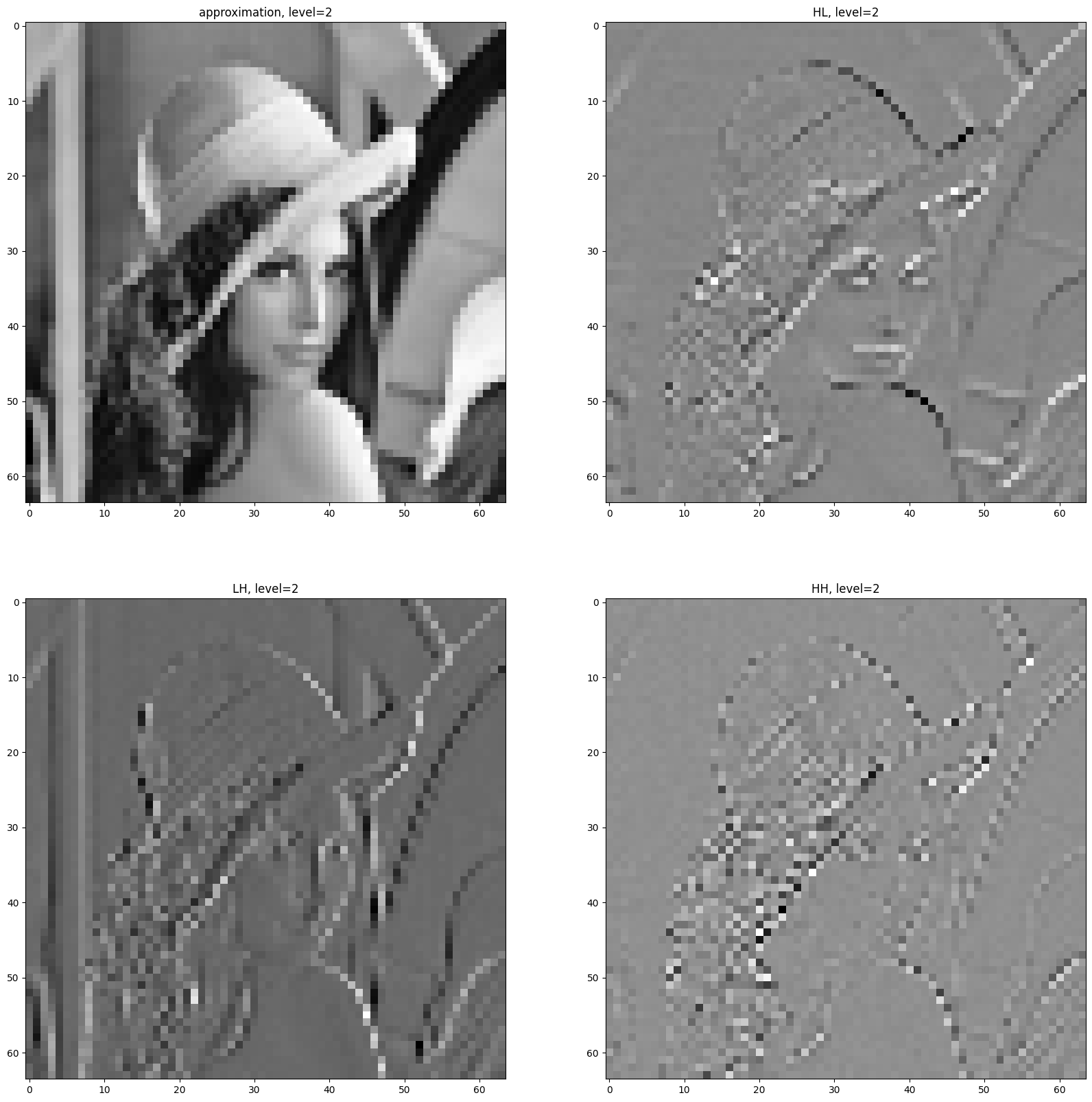
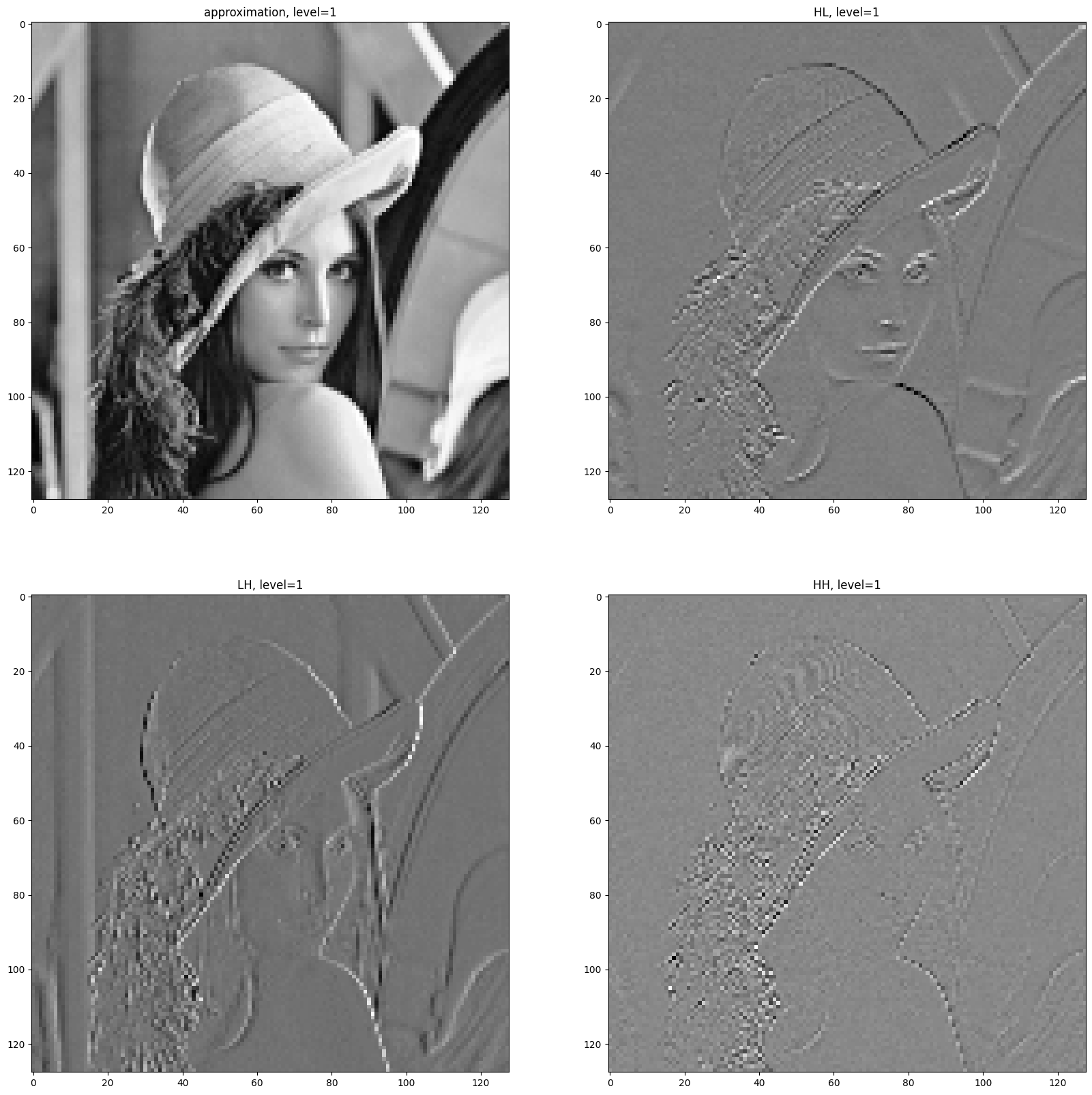
              f'HL, level={i}', f'LH, level={i}', f'HH, level={i}']

    plt.figure(*figsize*=(20, 20))

    subplot(2, 2, coeffs, titles)

تصاویر خروجی مربوط به این تمرین به صورت زیر است:



**

*در سوال قبل (4-1-5) به محاسبه approximation pyramid و prediction pyramid پرداختیم. در این سوال این هرم هارا به کمک تبدیل موجک به دست آوردیم. با توجه به خروجی، هرم موجک لبه های افقی، عموی و قطری را به صورت جداگانه به ما می‌دهد.*

*هرم لاپلاس (prediction residual pyramid) به دست آمده در سوال قبل را می توان برای فشرده سازی تصویر با دور انداختن ضرایب فرکانس بالا و بازسازی تصویر از ضرایب فرکانس پایین استفاده کرد. هرم موجک به دست آمده در این سوال نیز می تواند برای فشرده سازی تصویر با دور انداختن ضرایب با بزرگی های کوچک و بازسازی تصویر از ضرایب باقی مانده استفاده شود.*

***پیوست***

* *کدها و تصاویر مربوط به این تمرین در*[*GitHub*](https://github.com/saleh-sh/HW7) *در دسترس است.*