pohsun_40183452

September 17, 2025

1 EECS 442 PS2: Frequency and Signal Processing

Please provide the following information (e.g. Aparajito Saha, rajsaha):

Po-Hsun Chang, pohsun

Important: after you download the .ipynb file, please name it ___<your_uniqname>_<your_umid>.ipynb___ before you submit it to Canvas.

Example: rajsaha_00000000.ipynb.

2 Setup

Run the following code to import the modules you'll need. After your finish the assignment, remember to run all the cells and save the notebook to your local machine as a `.ipynb` file for Canvas submission.

[2]:
```python
import numpy as np, matplotlib as mpl, matplotlib.pyplot as plt, os
import urllib.request
import math
from PIL import Image
import scipy.ndimage # For image filtering
from scipy import signal
from scipy import fft
import imageio # For loading images
import ssl
ssl._create_default_https_context = ssl._create_unverified_context
```

3 Problem 2.1: Image blending

3.1 (a) Reconstructing an image

In this problem, we construct a Laplacian pyramid with 4 levels. We use the Laplacian pyramid to reconstruct the original image.

[3]:
```python
img_url = 'https://mapproxy.studentlife.umich.edu/images/
  ↪bob-and-betty-beyster-building/XP2S3440.jpg'
with open('bbb.jpg', 'wb') as out:
  out.write(urllib.request.urlopen(img_url).read())
```

```
img = Image.open('bbb.jpg')
img = img.resize((400,200))
img=np.array(img)

plt.figure()
plt.title('Input image')
plt.axis('off')
plt.imshow(img)
```

```
[3]: <matplotlib.image.AxesImage at 0x7b36d3f30650>
```

```
[4]: img_url = 'https://mapproxy.studentlife.umich.edu/images/
      ↪bob-and-betty-beyster-building/XP2S3440.jpg'
     with open('bbb.jpg', 'wb') as out:
       out.write(urllib.request.urlopen(img_url).read())

     img = Image.open('bbb.jpg')
     img = img.resize((1440,808))
     img.size
```

```
[4]: (1440, 808)
```

```
[5]: def pyramid_upsample(img, kernel_size=(5,5)):
       """
       Upsamples the given pyramid image.
       Input:
         - img: an image of shape M x N x C
```

```
    - kernel_size: a tuple representing the shape of the 2D kernel
  Returns:
    - upsampled: an image represented as an array of shape 2M x 2N x C
  """
  ###########################################################################
  # TODO: Implement pyramid upsampling.                                     #
  ###########################################################################

  # Insert zeros between pixels
  upsampled = np.zeros((img.shape[0] * 2, img.shape[1] * 2, img.shape[2]))
  upsampled[::2, ::2, :] = img

  # Apply Gaussian filter (sigma=1)
  radius = (kernel_size[0]//2, kernel_size[1]//2, 0)
  upsampled = scipy.ndimage.gaussian_filter(upsampled, sigma=1, radius=radius)␣
↪* 4
  ###########################################################################
  #                              END OF YOUR CODE                           #
  ###########################################################################
  return upsampled

def pyramid_downsample(img, kernel_size=(5,5)):
  """
  Downsamples the given pyramid image.
  Input:
    - img: an image of shape M x N x C
    - kernel_size: a tuple representing the shape of the 2D kernel
  Returns:
    - downsampled: an image of shape M/2 x N/2 x C
  """
  ###########################################################################
  # TODO: Implement pyramid downsampling.                                   #
  ###########################################################################

  # Apply Gaussian filter (sigma=1)
  radius = (kernel_size[0]//2, kernel_size[1]//2, 0)
  blurred_img = scipy.ndimage.gaussian_filter(img, sigma=1, radius=radius)

  # discard every other pixel
  downsampled = np.zeros((img.shape[0] // 2, img.shape[1] // 2, 3))
  downsampled = blurred_img[::2, ::2, :]
  ###########################################################################
  #                              END OF YOUR CODE                           #
  ###########################################################################
  return downsampled

def gen_gaussian_pyramid(img, num_levels):
  """
  Generates an entire Gaussian pyramid.
  Input:
    - img: an image of shape M x N x C
    - num_levels: number of levels in the Gaussian pyramid
  Return:
    - gp: list, the generated levels (imgs) of the Gaussian pyramid
  """
  ##############################################################################
  # TODO: Construct a Gaussian pyramid given a base image `img`.               #
  ##############################################################################
  gp = []
  for i in range(num_levels):
    if i == 0:
      gp = [img]
    else:
      gp.append(pyramid_downsample(gp[i-1]))
  ##############################################################################
  #                              END OF YOUR CODE                              #
  ##############################################################################
  return gp

def gen_laplacian_pyramid(gp, num_levels):
  """
  Generates an entire Laplacian pyramid.
  Input:
    gp: list, levels of a Gaussian pyramid
  Return:
    lp: list, the generated levels (imgs) of the Laplacian pyramid
  """
  ##############################################################################
  # TODO: Construct a Laplacian pyramid given a base Gaussian pyramid `gp`.    #
  ##############################################################################
  lp = [gp[-1]]
  for i in range(num_levels-1, 0, -1):
    level = (gp[i - 1] - pyramid_upsample(gp[i]))
    lp.append(level)

  ##############################################################################
  #                              END OF YOUR CODE                              #
  ##############################################################################
  return lp

def reconstruct_img(lp):
  """
  Reconstructs an image using a laplacian pyramid.
  Input:
    lp: list, levels (imgs) of a Laplacian pyramid
  Return:
    recon_img: reconstructed image
  """
  recon_img = lp[0]
  for i in range(1, len(lp)):
    ###########################################################################
    # TODO: For each level, reconstruct the image from the Laplacian pyramid. #
    ###########################################################################
    recon_img = np.clip(pyramid_upsample(recon_img) + lp[i], 0, 255)
    ###########################################################################
    #                              END OF YOUR CODE                           #
    ###########################################################################

  return recon_img
```

[6]:
```
# Run above functions and visualize results on bbb.jpg

img = np.array(img).astype('float')

num_levels = 4 # including the original img
gp = gen_gaussian_pyramid(img, num_levels)
lp = gen_laplacian_pyramid(gp, num_levels)
recon_img = reconstruct_img(lp)

print("Gaussian pyramid:")
for i, level in enumerate(gp):
  print("Level {}:".format(i), level.shape)
  plt.figure(figsize=[plt.rcParams.get('figure.figsize')[0]/(2**i),plt.rcParams.
 ↪get('figure.figsize')[1]/(2**i)])
  plt.title("Level {}:".format(i))
  plt.axis('off')
  level=(level-np.amin(level))/(np.amax(level)-np.amin(level))
  plt.imshow(level)


print("Laplacian pyramid:")
for i, level in enumerate(lp):
  print("Level {}:".format(i), level.shape)
  level_plot=level.clip(0, 255).astype('uint8')
  plt.figure(figsize=[plt.rcParams.get('figure.figsize')[0]/
 ↪(2**(num_levels-i-1)),plt.rcParams.get('figure.figsize')[1]/
 ↪(2**(num_levels-i-1))])
  plt.title("Level {}:".format(i))
  plt.axis('off')
  plt.imshow(level_plot)
```

```
img=img.astype(np.uint8)
plt.figure()
plt.title("Original Image")
plt.axis('off')
plt.imshow(img)

recon_img=recon_img.astype(np.uint8)
plt.figure()
plt.title("Reconstructed Image")
plt.axis('off')
plt.imshow(recon_img)
```

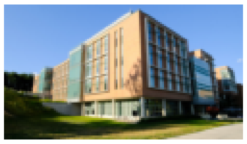
```
Gaussian pyramid:
Level 0: (808, 1440, 3)
Level 1: (404, 720, 3)
Level 2: (202, 360, 3)
Level 3: (101, 180, 3)
Laplacian pyramid:
Level 0: (101, 180, 3)
Level 1: (202, 360, 3)
Level 2: (404, 720, 3)
Level 3: (808, 1440, 3)
```

```
[6]: <matplotlib.image.AxesImage at 0x7b36c549b110>
```

Level 0:

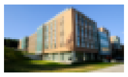
Level 1:

Level 2:

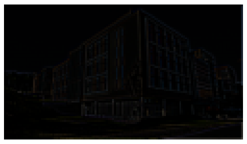


Level 3:

Level 0:

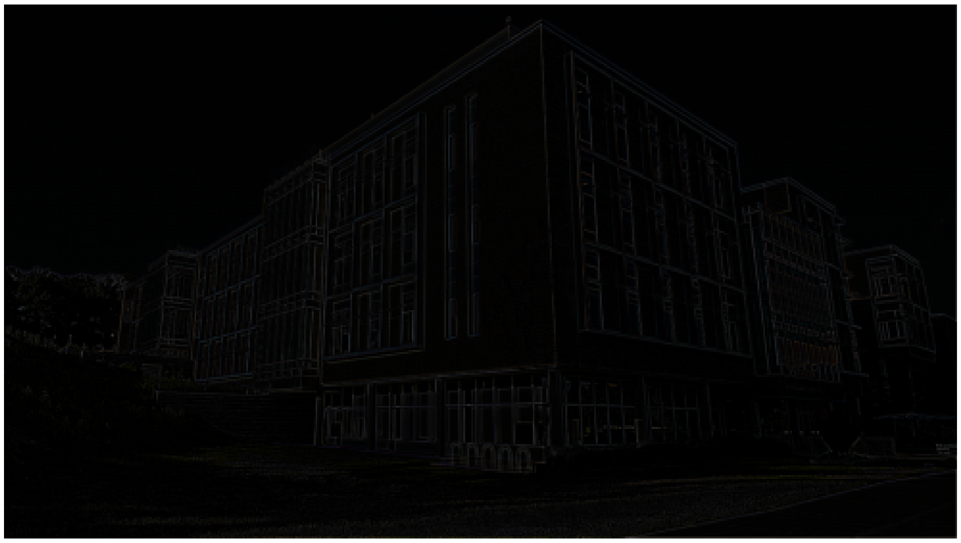


Level 1:



Level 2:

Level 3:



Original Image

Reconstructed Image

3.2 (b) Blending two images

In this problem, we blend an image of an orange and an apple.

[9]:
```python
# Download the `orange.jpg` and `apple.jpg` images.
base_url = 'https://www.eecs.umich.edu/courses/eecs442/fa24/ims'
for name in ['orange.jpg', 'apple-ps2.jpg']:
  with open(name, 'wb') as out:
    url = os.path.join(base_url, name)
    out.write(urllib.request.urlopen(url).read())

# Resize images and create the image mask.
img1 = Image.open('apple-ps2.jpg')
img2 =Image.open('orange.jpg')

width,height=img1.size
new_width,new_height=width//2,height//2

img1 = img1.resize((new_width,new_height))
img1=np.array(img1)

img2 = img2.resize((new_width,new_height))
img2=np.array(img2)

mask = np.zeros_like(img1)
mask[:,:mask.shape[1]//2, :] = 1
```

[10]:
```
def pyramid_blend(img1, img2, mask, num_levels=6):
    """
    This function produces the Laplacian pyramid blend of two images.
    Input:
      - img1: N x M x C uint8 array image
      - img2: N x M x C uint8 array image
      - mask: N x M array, all elements are either 0s or 1s
      - num_levels: int, height of the pyramid
    Return:
      - img_blend: the blended image, an N x M x C uint8 array
    """
    # Build Gaussian pyramids for img1, img2, and mask
    gp1, gp2, gpm = gen_gaussian_pyramid(img1, num_levels),␣
  ↪gen_gaussian_pyramid(img2, num_levels), gen_gaussian_pyramid(mask,␣
  ↪num_levels)

    # Build Laplacian pyramids for img1 and img2
    lp1, lp2, lpm = gen_laplacian_pyramid(gp1, num_levels),␣
  ↪gen_laplacian_pyramid(gp2, num_levels), gen_laplacian_pyramid(gpm,␣
  ↪num_levels)

    ############################################################################
    # TODO: Construct the Laplacian pyramid and use it to blend the images.    #
    ############################################################################

    blended_pyramid = []
    for i, (l1, l2) in enumerate(zip(lp1, lp2)):
        m = gpm[len(gpm)-1-i]

        blended = l1 * m + l2 * (1 - m)
        blended_pyramid.append(blended)

    # Reconstruct the blended image from the blended pyramid
    img_blend = reconstruct_img(blended_pyramid)

    ###############################################################################
    #                              END OF YOUR CODE                               #
    ###############################################################################

    return img_blend
```

Please describe the difference between the blended images as the number of levels in the Laplacian pyramid varies: how does the result change as we increase the number of levels?

Answer: As the number of levels in the Laplacian pyramid increases, the blending becomes smoother, with a more gradual transition at the boundaries, which reduces the visibility of sharp edges in the blended regions.

[11]:
```
# Display images and mask

# Plotting image 1 - Apple
plt.figure()
plt.title('Image 1')
plt.axis('off')
plt.imshow(img1)

# Plotting image 2 - Orange
plt.figure()
plt.title('Image 2')
plt.axis('off')
plt.imshow(img2)

# Plotting the mask
plt.figure()
plt.title('Mask')
plt.axis('off')
plt.imshow(mask * 255)
```

[11]:
```
<matplotlib.image.AxesImage at 0x7b36ba4f0b90>
```

Image 1

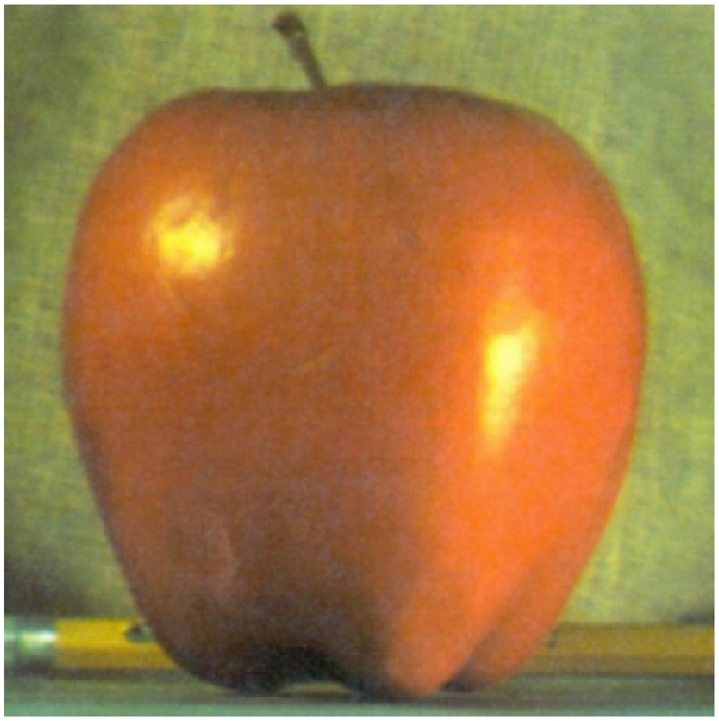
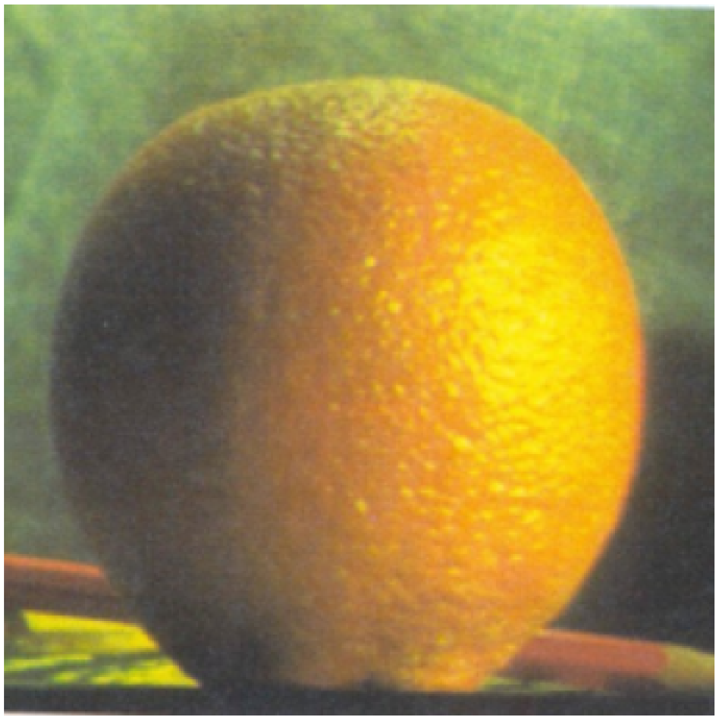
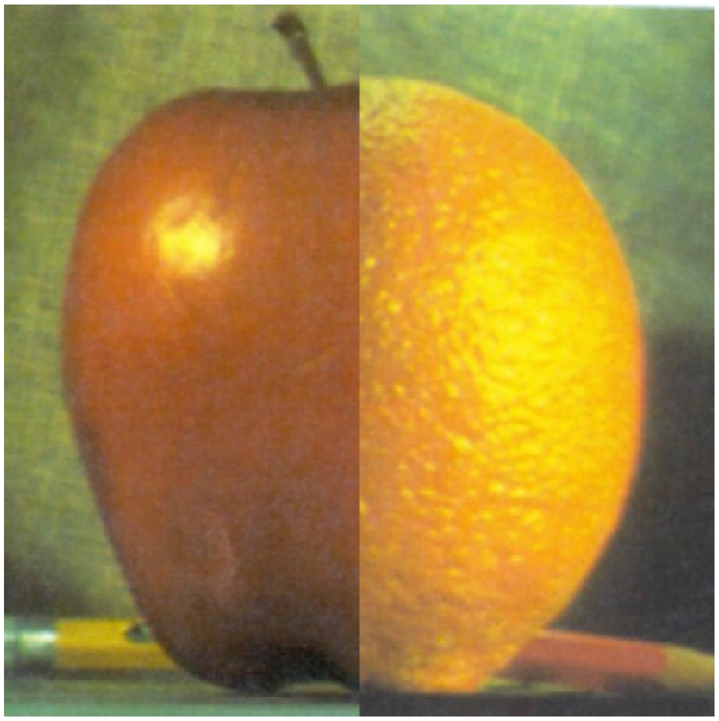


Image 2

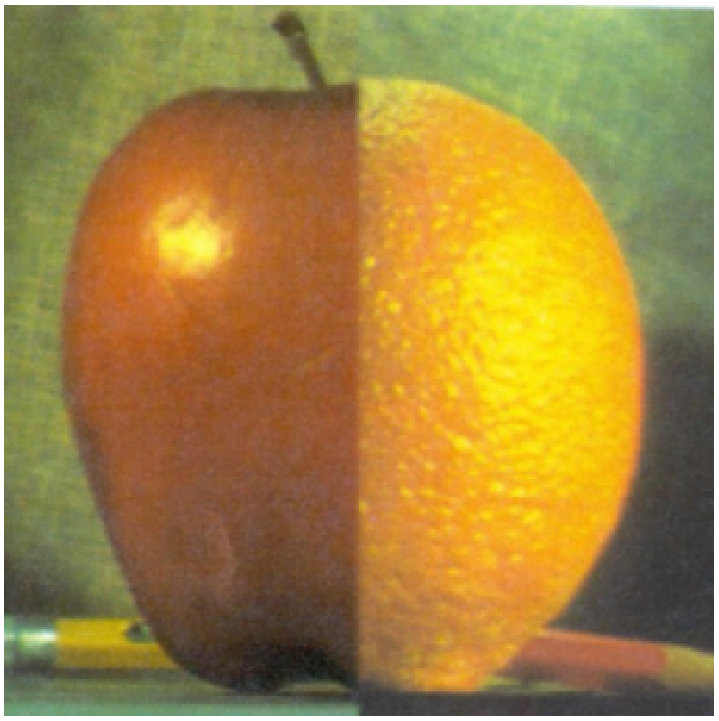


```
[12]: # Visualize Laplacian pyramid blend by varying number of levels
      for n_l in range(1, 7):
        img_blend = pyramid_blend(img1.astype(float), img2.astype(float), mask.
       ↪astype(float), num_levels=n_l)
        img_blend=img_blend.astype(np.uint8)
        plt.figure()
        plt.title('Number of levels = {}'.format(n_l))
        plt.axis('off')
        plt.imshow(img_blend)
```

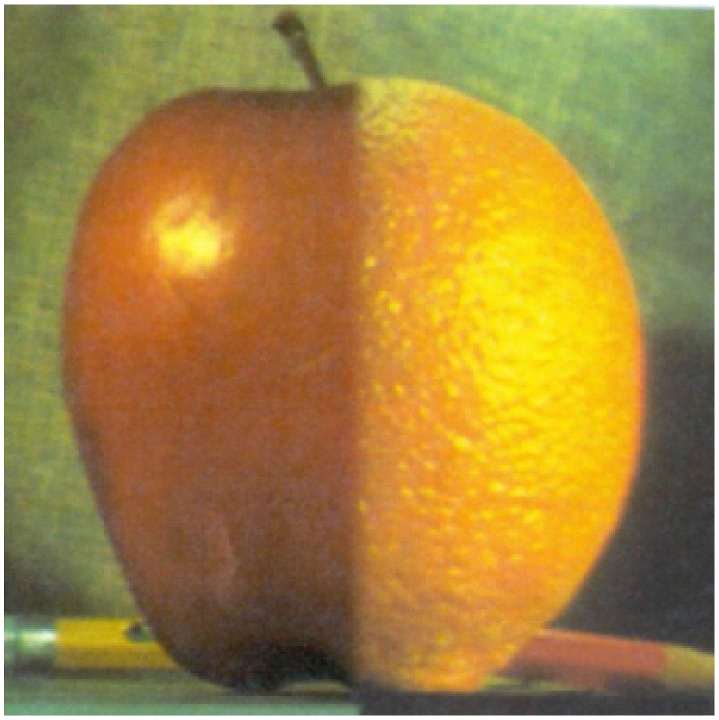
Number of levels = 1



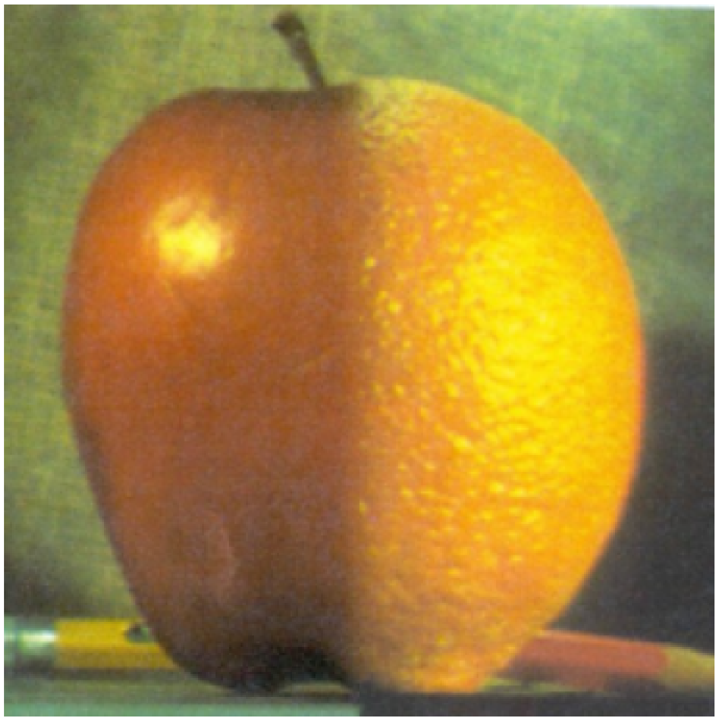
Number of levels = 2



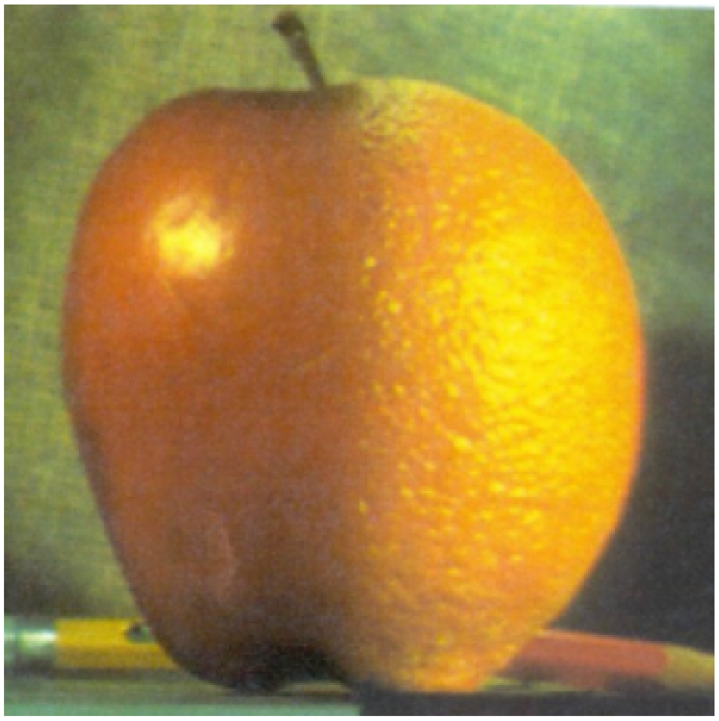
Number of levels = 3

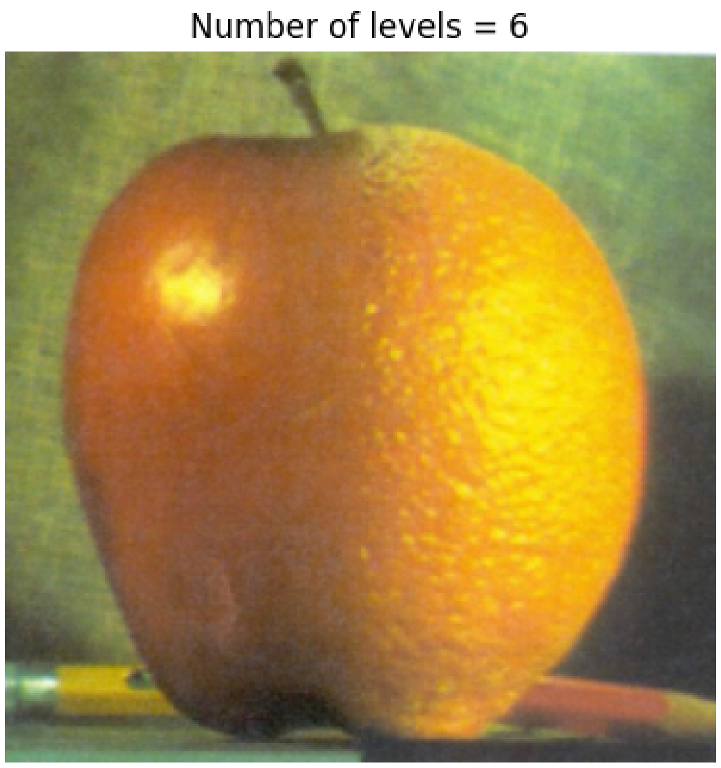


Number of levels = 4



Number of levels = 5





4 Problem 2.2: 2D DFT and the Convolution Theorem

(a) Please match the images in the left column of Figure 2 to their corresponding spectrum visualization in the right column.

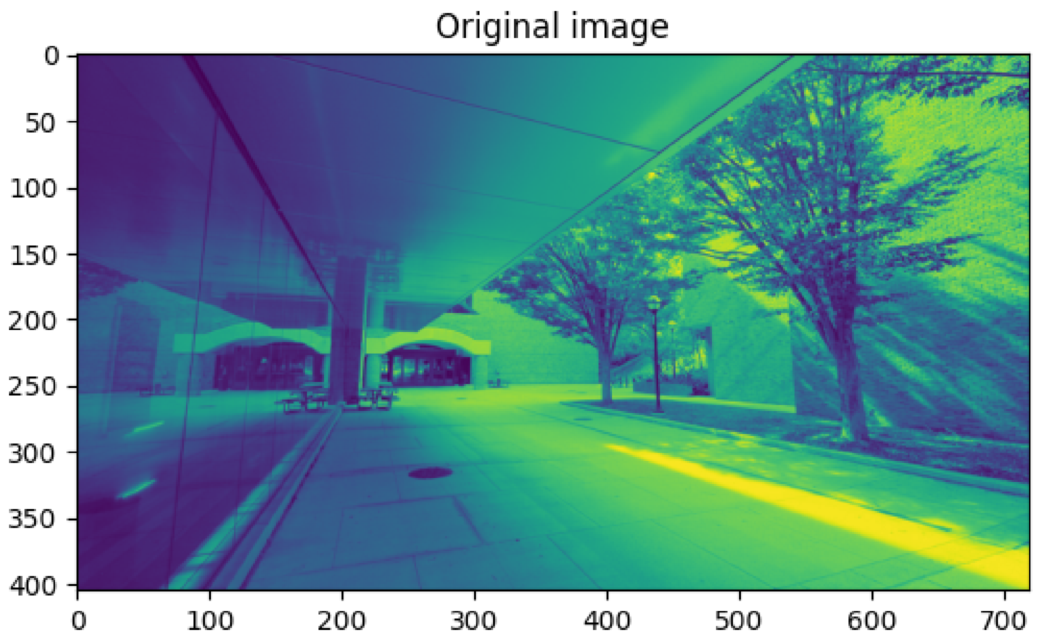
Answer: 1:E, 2:C, 3:D, 4:A, 5:B

(b) In this problem, we compare the output of a Gaussian filter through i) direct convolution in the spatial domain and ii) multiplication in the frequency domain.

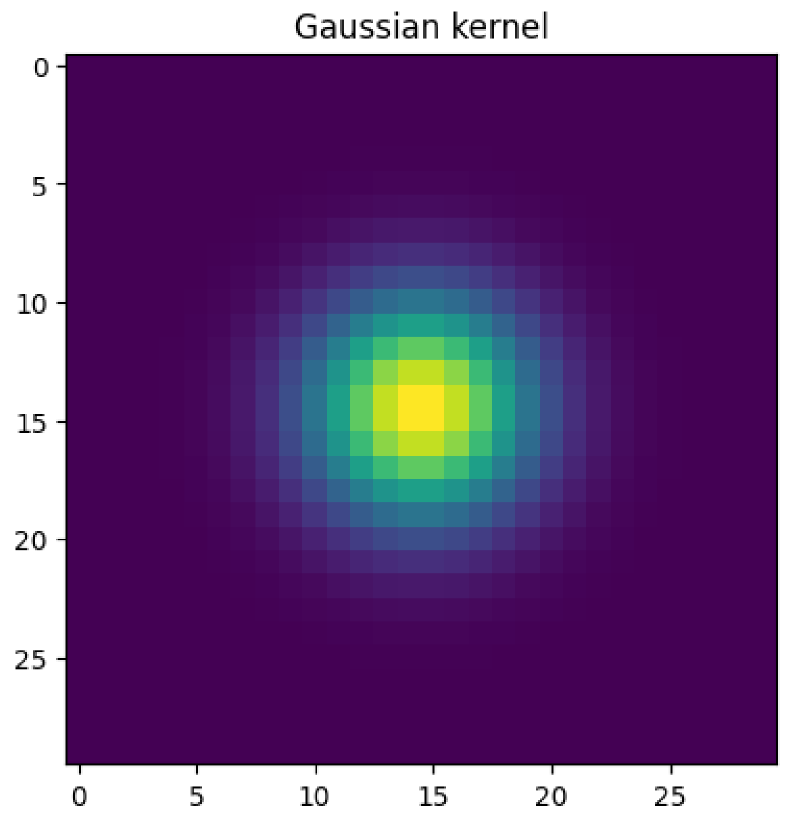
```
[13]: img_url = 'https://mapproxy.studentlife.umich.edu/images/
       ↪electrical-engineering-and-computer-science-bld/XP2S3935.jpg'
      with open('eecs.jpg', 'wb') as out:
        out.write(urllib.request.urlopen(img_url).read())


      img=Image.open('eecs.jpg').convert('L') #Here, we are converting the RGB image␣
       ↪into a grayscale
      width, height=img.size
      new_width, new_height = width//2, height//2
```

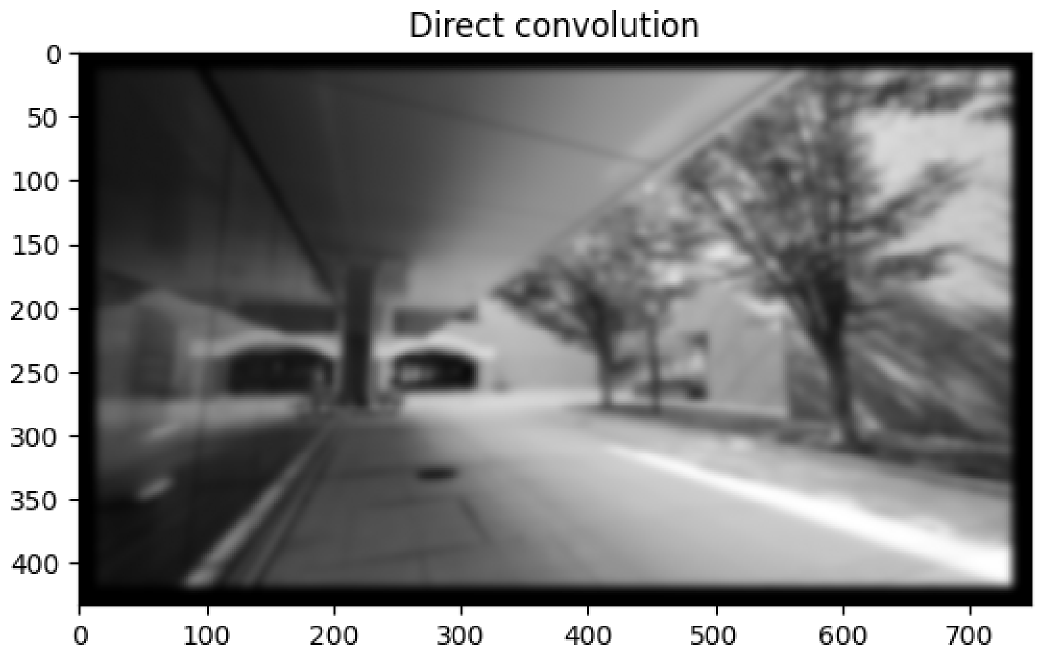
```
img = img.resize((new_width, new_height))
img = np.array(img)


plt.figure()
plt.imshow(img)
plt.title('Original image')
plt.show()

# Make a 2-D kernel Gaussian kernel
t = np.linspace(-10, 10, 30)
bump = np.exp(-0.1*t**2)
bump /= np.trapz(bump) # normalize the integral to 1
kernel = bump[:, np.newaxis] * bump[np.newaxis, :]
plt.imshow(kernel)
plt.title('Gaussian kernel')
plt.show()

# Convolution in spatial domain
#############################################################################
# TODO: Convolve the gaussian filter with the image in the spatial domain.  #
#############################################################################
img1 = signal.convolve2d(img, kernel)
#############################################################################
#                              END OF YOUR CODE                             #
#############################################################################
plt.figure()
plt.title('Direct convolution')
plt.imshow(img1, cmap='gray')
plt.show()

# Multiplication in frequency domain
# This shape needs to be passed into the fft2 function
shape = (img.shape[0]+kernel.shape[0]-1, img.shape[1]+kernel.shape[1]-1)
#############################################################################
# TODO: Use the convolution theorem to convolve the provided image with     #
#       a Gaussian filter.                                                  #
#############################################################################
# Perform FFT on the image and kernel
img_fft = fft.fft2(img, shape)
kernel_fft = fft.fft2(kernel, shape)

# Multiply in the frequency domain and perform inverse FFT
img2 = fft.ifft2(img_fft * kernel_fft).real

#############################################################################
```

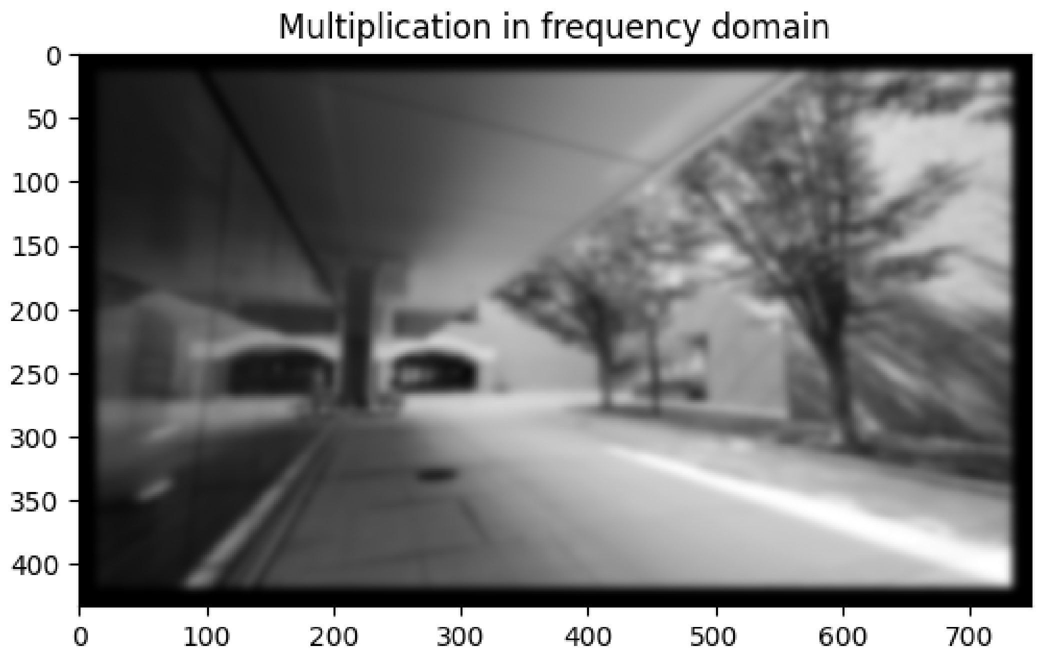
```
#                              END OF YOUR CODE                              #
##############################################################################
plt.figure()
plt.title('Multiplication in frequency domain')
plt.imshow(img2, cmap='gray')
plt.show()
```



```
/tmp/ipython-input-1727537708.py:22: DeprecationWarning: `trapz` is deprecated.
Use `trapezoid` instead, or one of the numerical integration functions in
`scipy.integrate`.
  bump /= np.trapz(bump) # normalize the integral to 1
```


Gaussian kernel
0
5
10
15
20
25
0
5
10
15
20
25


Direct convolution
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600
700

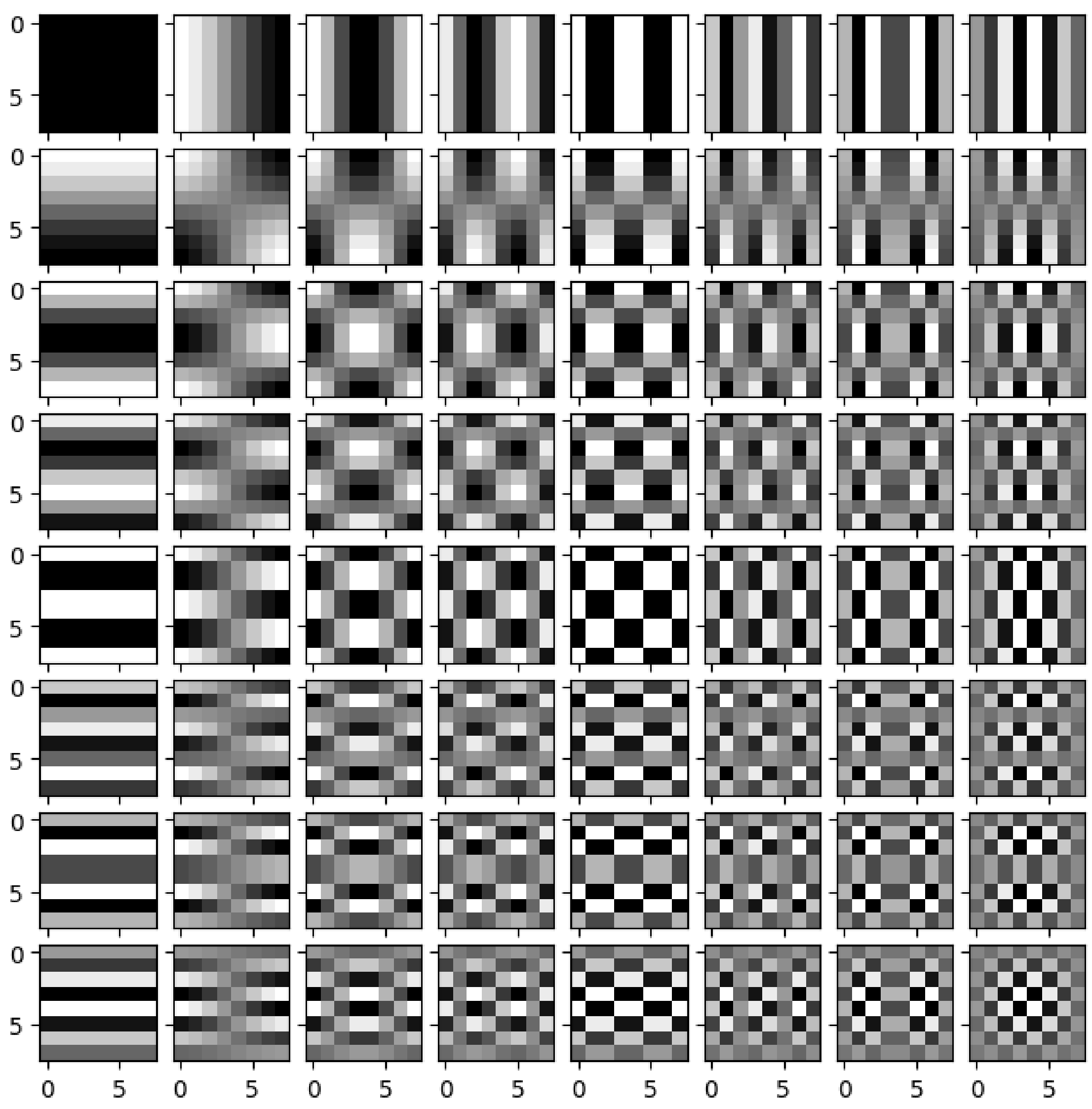


5 Problem 2.3 JPEG: Image Compression

5.1 (a) Basis for 2D Discrete Cosine Transform

[14]:
```
import numpy as np

def build_2D_DCT_basis(Nx, Ny):
  #Create 2D coordinate grid
  X = np.arange(Nx)[None,:].repeat(Ny,axis=0)
  Y = np.arange(Ny)[:,None].repeat(Nx,axis=1)

  DCT_basis = np.zeros([Ny,Nx,Ny,Nx]) # First two indices correspond to␣
↪frequency components (ky,kx), last two indices correspond to spatial␣
↪location (y,x).
  for kx in range(Nx):
    for ky in range(Ny):
      ␣
↪########################################################################
      # TODO: Fill in DCT_basis as defined in the problem set.                ␣
↪  #
      ␣
↪########################################################################
      DCT_basis[ky, kx] = (np.cos(np.pi * (2*X + 1) * kx / (2*Nx)) * np.cos(np.
↪pi * (2*Y + 1) * ky / (2*Ny)))
```

```
    ␣
↪#############################################################################
      #                              END OF YOUR CODE                          ␣
↪   #
    ␣
↪#############################################################################
  return DCT_basis

# Build the DCT basis and visualize it.
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import ImageGrid

Nx, Ny = 8, 8
DCT_basis = build_2D_DCT_basis(Nx, Ny)
fig = plt.figure(figsize=(8., 8.))
grid = ImageGrid(fig, 111,  # similar to subplot(111)
                 nrows_ncols=(Ny, Nx),  # creates a grid of axes
                 axes_pad=0.1,  # pad between axes in inch.
                 )
for ax, im in zip(grid, DCT_basis.reshape(-1,Ny,Nx)):
    # Iterating over the grid returns the Axes.
    ax.imshow(im, cmap = 'gray')
plt.show()
```



5.2 (b) 2D Discrete Cosine Transform

```
[15]: def DCT_2D(image_patch):
        """
        2D DCT transform of image patch of shape (H,W)
        """
        H,W = image_patch.shape
        DCT_basis = build_2D_DCT_basis(W,H)
        DCT_coeffcients = np.zeros([H,W])

        for h in range(H):
          for w in range(W):
```

```
        norm_const_h = 1/np.sqrt(H) if h == 0 else np.sqrt(2/H)
        norm_const_w = 1/np.sqrt(W) if w == 0 else np.sqrt(2/W)
        DCT_coeffcients[h,w] = norm_const_h * norm_const_w * (DCT_basis[h, w] *␣
  ↪image_patch).sum()
  return DCT_coeffcients

def IDCT_2D(image_spectrum):
  """
  2D inverse DCT trasform.
  """
  H,W = image_spectrum.shape
  DCT_basis = build_2D_DCT_basis(W,H)
  image = np.zeros([H,W])
  norm_matrix = 1/np.sqrt(H*W) * np.ones([H,W])
  norm_matrix[1:,0] *= np.sqrt(2)
  norm_matrix[0,1:] *= np.sqrt(2)
  norm_matrix[1:,1:] *= 2

  for h in range(H):
    for w in range(W):
      norm_const_h = 1/np.sqrt(H) if h == 0 else np.sqrt(2/H)
      norm_const_w = 1/np.sqrt(W) if w == 0 else np.sqrt(2/W)
      image += norm_const_h * norm_const_w * (DCT_basis[h, w] *␣
  ↪image_spectrum[h,w])
  return image

image_patch = np.random.rand(8,8)
# Verify above gives same result as using scipy fftpack
from scipy import fftpack
def dct_2d(image):
    return fftpack.dct(fftpack.dct(image.T, norm='ortho').T, norm='ortho')
def idct_2d(image):
    return fftpack.idct(fftpack.idct(image.T, norm='ortho').T, norm='ortho')
print(np.abs(DCT_2D(image_patch) - dct_2d(image_patch)).sum())
print(np.abs(IDCT_2D(image_patch) - idct_2d(image_patch)).sum())
```

```
3.1787072973799013e-14
2.256875242245826e-14
```

5.3 (c) JPEG Image Compression

[17]:
```
#Download the image to be compressed
!wget -O original-ps2.png "https://www.eecs.umich.edu/courses/eecs442/fa24/ims/
  ↪original-ps2.png"
```

```
--2025-09-17 01:43:53--
https://www.eecs.umich.edu/courses/eecs442/fa24/ims/original-ps2.png
```

```
Resolving www.eecs.umich.edu (www.eecs.umich.edu)… 141.212.113.116
Connecting to www.eecs.umich.edu (www.eecs.umich.edu)|141.212.113.116|:443…
connected.
ERROR: cannot verify www.eecs.umich.edu's certificate, issued by ‘CN=InCommon
RSA Server CA 2,O=Internet2,C=US’:
  Unable to locally verify the issuer's authority.
To connect to www.eecs.umich.edu insecurely, use `--no-check-certificate'.
```

5.3.1 Utility functions

We will first define some utility functions. You don't need to implement anything in this block or read all of its details. The only important functions in this part you may want to look at to understand better are `quantize` and `load_quantization_table`.

[18]:
```python
# We will first define some utility functions. You don't need to implement␣
↪anything in this block or read all of its details.
# The only important functions in this part you may want to look at to␣
↪understand better are 'quantize' and 'load_quantization_table'.
# Code in this part is based on https://github.com/ghallak/jpeg-python
from queue import PriorityQueue
class HuffmanTree:
  """
  Class for creating Huffman Table, given an input array. The huffman encodes
  input data based on their appearance frequency,
  i.e., assign a shorter code for more frequently appearing values.
  """
  def __init__(self, arr):
      q = PriorityQueue()
      # calculate frequencies and insert them into a priority queue
      for val, freq in self.__calc_freq(arr).items():
          q.put(self.__Node.init_leaf(val, freq))
      while q.qsize() >= 2:
          u = q.get()
          v = q.get()
          q.put(self.__Node.init_node(u, v))
      self.__root = q.get()
      # dictionaries to store huffman table
      self.__value_to_bitstring = dict()

  def value_to_bitstring_table(self):
      if len(self.__value_to_bitstring.keys()) == 0:
          self.__create_huffman_table()
      return self.__value_to_bitstring

  def __create_huffman_table(self):
      def tree_traverse(current_node, bitstring=''):
          if current_node is None:
              return
```

```
            if current_node.is_leaf():
                self.__value_to_bitstring[current_node.value] = bitstring
                return
            tree_traverse(current_node.left_child, bitstring + '0')
            tree_traverse(current_node.right_child, bitstring + '1')

        tree_traverse(self.__root)

    def __calc_freq(self, arr):
        freq_dict = dict()
        for elem in arr:
            if elem in freq_dict:
                freq_dict[elem] += 1
            else:
                freq_dict[elem] = 1
        return freq_dict

    class __Node:
        def __init__(self, value, freq, left_child, right_child):
            self.value = value
            self.freq = freq
            self.left_child = left_child
            self.right_child = right_child

        @classmethod
        def init_leaf(cls, value, freq):
            return cls(value, freq, None, None)

        @classmethod
        def init_node(cls, left_child, right_child):
            freq = left_child.freq + right_child.freq
            return cls(None, freq, left_child, right_child)

        def is_leaf(self):
            return self.value is not None

        def __eq__(self, other):
            stup = self.value, self.freq, self.left_child, self.right_child
            otup = other.value, other.freq, other.left_child, other.right_child
            return stup == otup

        def __nq__(self, other):
            return not (self == other)

        def __lt__(self, other):
            return self.freq < other.freq
```

```
      def __le__(self, other):
          return self.freq < other.freq or self.freq == other.freq

      def __gt__(self, other):
          return not (self <= other)

      def __ge__(self, other):
          return not (self < other)

def load_quantization_table(component):
    # Quantization tables determine how much DCT coefficents are divided during
    # quantization. Larger value means it is more likely to be a 0 after␣
 ↪division.
    # Setting large values like 100000 means that frequency component is set to␣
 ↪0 for sure.
    if component == 'dc_only':
        q = np.array([[1, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000]])
    elif component == 'first_3':
        q = np.array([[1, 1, 100000, 100000, 100000, 100000, 100000, 100000],
                      [1, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000]])
    elif component == 'first_6':
        q = np.array([[1, 1, 1, 100000, 100000, 100000, 100000, 100000],
                      [1, 1, 100000, 100000, 100000, 100000, 100000, 100000],
                      [1, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000]])
    elif component == 'first_10':
        q = np.array([[1, 1, 1, 1, 100000, 100000, 100000, 100000],
                      [1, 1, 1, 100000, 100000, 100000, 100000, 100000],
                      [1, 1, 100000, 100000, 100000, 100000, 100000, 100000],
                      [1, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000],
                      [100000, 100000, 100000, 100000, 100000, 100000, 100000,␣
 ↪100000]])
    #q1 and q2 are some more realistic quantization values. You can try these,␣
 ↪but no need to report any result from using 'q1' or 'q2'.
    elif component == 'q1':
        q = np.array([[2, 2, 2, 2, 3, 4, 5, 6],
                      [2, 2, 2, 2, 3, 4, 5, 6],
                      [2, 2, 2, 2, 4, 5, 7, 9],
                      [2, 2, 2, 4, 5, 7, 9, 12],
                      [3, 3, 4, 5, 8, 10, 12, 12],
                      [4, 4, 5, 7, 10, 12, 12, 12],
                      [5, 5, 7, 9, 12, 12, 12, 12],
                      [6, 6, 9, 12, 12, 12, 12, 12]])

    elif component == 'q2':
        q = np.array([[16, 11, 10, 16, 24, 40, 51, 61],
                      [12, 12, 14, 19, 26, 58, 60, 55],
                    [14, 13, 16, 24, 40, 57, 69, 56],
                    [14, 17, 22, 29, 51, 87, 80, 62],
                    [18, 22, 37, 56, 68, 109, 103, 77],
                    [24, 35, 55, 64, 81, 104, 1113, 92],
                    [49, 64, 78, 87, 103, 121, 120, 101],
                    [72, 92, 95, 98, 112, 100, 103, 99]])
    else:
        raise ValueError((
            "Unrecognized compoents,'{comp}' was found").format(comp=component))
    return q

def zigzag_points(rows, cols):
    # constants for directions
    UP, DOWN, RIGHT, LEFT, UP_RIGHT, DOWN_LEFT = range(6)

    # move the point in different directions
    def move(direction, point):
        return {
            UP: lambda point: (point[0] - 1, point[1]),
            DOWN: lambda point: (point[0] + 1, point[1]),
            LEFT: lambda point: (point[0], point[1] - 1),
            RIGHT: lambda point: (point[0], point[1] + 1),
            UP_RIGHT: lambda point: move(UP, move(RIGHT, point)),
            DOWN_LEFT: lambda point: move(DOWN, move(LEFT, point))
        }[direction](point)

    # return true if point is inside the block bounds
    def inbounds(point):
        return 0 <= point[0] < rows and 0 <= point[1] < cols
    # start in the top-left cell
    point = (0, 0)
    # True when moving up-right, False when moving down-left
    move_up = True
    for i in range(rows * cols):
        yield point
        if move_up:
            if inbounds(move(UP_RIGHT, point)):
                point = move(UP_RIGHT, point)
            else:
                move_up = False
                if inbounds(move(RIGHT, point)):
                    point = move(RIGHT, point)
                else:
                    point = move(DOWN, point)
        else:
            if inbounds(move(DOWN_LEFT, point)):
                point = move(DOWN_LEFT, point)
            else:
                move_up = True
                if inbounds(move(DOWN, point)):
                    point = move(DOWN, point)
                else:
                    point = move(RIGHT, point)

def bits_required(n):
    n = abs(n)
    result = 0
    while n > 0:
        n >>= 1
        result += 1
    return result

def binstr_flip(binstr):
    # check if binstr is a binary string
    if not set(binstr).issubset('01'):
        raise ValueError("binstr should have only '0's and '1's")
    return ''.join(map(lambda c: '0' if c == '1' else '1', binstr))

def uint_to_binstr(number, size):
    return bin(number)[2:][-size:].zfill(size)

def int_to_binstr(n):
    if n == 0:
        return ''
    binstr = bin(abs(n))[2:]
    # change every 0 to 1 and vice verse when n is negative
    return binstr if n > 0 else binstr_flip(binstr)

def flatten(lst):
    return [item for sublist in lst for item in sublist]


#Utility functions for encoding and writing to files below
def quantize(block, component):
    q = load_quantization_table(component)
    return (block / q).round().astype(np.int32)

def block_to_zigzag(block):
    return np.array([block[point] for point in zigzag_points(*block.shape)])

def run_length_encode(arr):
    # determine where the sequence is ending prematurely
    last_nonzero = -1
    for i, elem in enumerate(arr):
        if elem != 0:
            last_nonzero = i

    # each symbol is a (RUNLENGTH, SIZE) tuple
    symbols = []

    # values are binary representations of array elements using SIZE bits
    values = []

    run_length = 0

    for i, elem in enumerate(arr):
        if i > last_nonzero:
            symbols.append((0, 0))
            values.append(int_to_binstr(0))
            break
        elif elem == 0 and run_length < 15:
            run_length += 1
        else:
            size = bits_required(elem)
            symbols.append((run_length, size))
            values.append(int_to_binstr(elem))
            run_length = 0
    return symbols, values

def write_to_file(filepath, dc, ac, blocks_count, tables):
    try:
        f = open(filepath, 'w')
    except FileNotFoundError as e:
        raise FileNotFoundError(
                "No such directory: {}".format(
                    os.path.dirname(filepath))) from e
    #f.write('1')
    for table_name in ['dc_y', 'ac_y']:

        # 16 bits for 'table_size'
        f.write(uint_to_binstr(len(tables[table_name]), 16))

        for key, value in tables[table_name].items():
            if table_name == 'dc_y':
                # 4 bits for the 'category'
                # 4 bits for 'code_length'
                # 'code_length' bits for 'huffman_code'
                f.write(uint_to_binstr(key, 4))
                f.write(uint_to_binstr(len(value), 4))
                f.write(value)
            else:
                # 4 bits for 'run_length'
                # 4 bits for 'size'
                # 8 bits for 'code_length'
                # 'code_length' bits for 'huffman_code'
                f.write(uint_to_binstr(key[0], 4))
                f.write(uint_to_binstr(key[1], 4))
                f.write(uint_to_binstr(len(value), 8))
                f.write(value)

    # 32 bits for 'blocks_count'
    f.write(uint_to_binstr(blocks_count, 32))
    for b in range(blocks_count):
        for c in range(1):
            category = bits_required(dc[b, c])
            symbols, values = run_length_encode(ac[b, :, c])

            dc_table = tables['dc_y']
            ac_table = tables['ac_y']

            f.write(dc_table[category])
            f.write(int_to_binstr(dc[b, c]))
            #bit_count_ac = 0
            for i in range(len(symbols)):
                f.write(ac_table[tuple(symbols[i])])
                f.write(values[i])
                #bits_written = len(ac_table[tuple(symbols[i])]) +␣
↪len(values[i])
                #bit_count_ac += bits_written
            #print('Block {}:{} bits'.format(b,bit_count_ac))
    f.close()

def HuffmanEncode_im(filepath, im):
    """Huffman encoding the gray scale image directly.

    Args:
        filepath : [Location to save the encoded file]
        im ([numpy array]): Shape (H,W)
    """
    im = im.flatten()
    pixel_count = len(im)
    f = open(filepath,'w')
    H_Tree = HuffmanTree(im.flatten())
    H_table = H_Tree.value_to_bitstring_table()

    # 16 bits for 'table_size'
    f.write(uint_to_binstr(len(H_table), 16))
    for key, value in H_table.items():
            # 4 bits for the 'category'
            # 4 bits for 'code_length'
            # 'code_length' bits for 'huffman_code'
            f.write(uint_to_binstr(key, 4))
            f.write(uint_to_binstr(len(value), 4))
            f.write(value)

    # 32 bits for 'pixel_count'
    f.write(uint_to_binstr(pixel_count, 32))
    for item in im:
        f.write(H_table[item])
    f.close()

def write_as_binary(string_file_path, output_file_path):

    f = open(string_file_path,'r')
    binary_string = f.read()
    f_binary = open(output_file_path,mode='wb')
    #Remainder bits are not written for simplicity.
    for i in range(len(binary_string) // 8):
        byte = bytes([int(binary_string[8*i:8*i+8],2)])
        f_binary.write(byte)

    f.close()
    f_binary.close()


# Utility functions for decode
class JPEGFileReader:
    TABLE_SIZE_BITS = 16
    BLOCKS_COUNT_BITS = 32

    DC_CODE_LENGTH_BITS = 4
    CATEGORY_BITS = 4

    AC_CODE_LENGTH_BITS = 8
    RUN_LENGTH_BITS = 4
    SIZE_BITS = 4

    def __init__(self, filepath):
        self.__file = open(filepath, 'r')

    def read_int(self, size):
        if size == 0:
            return 0
        # the most significant bit indicates the sign of the number
        bin_num = self.__read_str(size)
        if bin_num[0] == '1':
            return self.__int2(bin_num)
        else:
            return self.__int2(binstr_flip(bin_num)) * -1

    def read_dc_table(self):
        table = dict()

        table_size = self.__read_uint(self.TABLE_SIZE_BITS)
        for _ in range(table_size):
            category = self.__read_uint(self.CATEGORY_BITS)
            code_length = self.__read_uint(self.DC_CODE_LENGTH_BITS)
            code = self.__read_str(code_length)
            table[code] = category
        return table

    def read_ac_table(self):
        table = dict()

        table_size = self.__read_uint(self.TABLE_SIZE_BITS)
        for _ in range(table_size):
            run_length = self.__read_uint(self.RUN_LENGTH_BITS)
            size = self.__read_uint(self.SIZE_BITS)
            code_length = self.__read_uint(self.AC_CODE_LENGTH_BITS)
            code = self.__read_str(code_length)
            table[code] = (run_length, size)
        return table

    def read_blocks_count(self):
        return self.__read_uint(self.BLOCKS_COUNT_BITS)

    def read_huffman_code(self, table):
        prefix = ''
        while prefix not in table:
            prefix += self.__read_char()
        return table[prefix]

    def __read_uint(self, size):
        if size <= 0:
            raise ValueError("size of unsigned int should be greater than 0")
        return self.__int2(self.__read_str(size))

    def __read_str(self, length):
        return self.__file.read(length)

    def __read_char(self):
        return self.__read_str(1)

    def __int2(self, bin_num):
        return int(bin_num, 2)


def read_image_file(filepath):
    reader = JPEGFileReader(filepath)

    tables = dict()
    for table_name in ['dc_y', 'ac_y']:
        if 'dc' in table_name:
            tables[table_name] = reader.read_dc_table()
        else:
            tables[table_name] = reader.read_ac_table()

    blocks_count = reader.read_blocks_count()

    dc = np.empty((blocks_count, 1), dtype=np.int32)
    ac = np.empty((blocks_count, 63, 1), dtype=np.int32)

    for block_index in range(blocks_count):
        for component in range(1):
            dc_table = tables['dc_y']
            ac_table = tables['ac_y']

            category = reader.read_huffman_code(dc_table)
            dc[block_index, component] = reader.read_int(category)
            cells_count = 0

            while cells_count < 63:
                run_length, size = reader.read_huffman_code(ac_table)

                if (run_length, size) == (0, 0):
                    while cells_count < 63:
                        ac[block_index, cells_count, component] = 0
                        cells_count += 1
                else:
                    for i in range(run_length):
                        ac[block_index, cells_count, component] = 0
                        cells_count += 1
                    if size == 0:
                        ac[block_index, cells_count, component] = 0
                    else:
                        value = reader.read_int(size)
                        ac[block_index, cells_count, component] = value
                    cells_count += 1
    return dc, ac, tables, blocks_count


def zigzag_to_block(zigzag):
    # assuming that the width and the height of the block are equal
    rows = cols = int(math.sqrt(len(zigzag)))

    if rows * cols != len(zigzag):
        raise ValueError("length of zigzag should be a perfect square")

    block = np.empty((rows, cols), np.int32)

    for i, point in enumerate(zigzag_points(rows, cols)):
        block[point] = zigzag[i]

    return block

def dequantize(block, component):
    q = load_quantization_table(component)
    return block * q
```

5.3.2 Encoding and Decoding

[ ]:
```
# Encoding

# Path to the input image file to be compressed.
input_file = './original-ps2.png'
# Path for the encoded code file.
output_file = './encoded_as_string'

###############################################################################
# TODO: Specify the quantization table to use here.                           #
# Your options: 'dc_only','first_3', 'first_6', 'first_10'                    #
###############################################################################
quantization_table = load_quantization_table('first_10')
###############################################################################
#                               END OF YOUR CODE                              #
###############################################################################

image = Image.open(input_file)
npmat = np.array(image, dtype=np.int16)[:,:,None]

rows, cols = npmat.shape[0], npmat.shape[1]

# block size: 8x8
if rows % 8 == cols % 8 == 0:
    blocks_count = rows // 8 * cols // 8
else:
    raise ValueError(("the width and height of the image should both be␣
 ↪mutiples of 8"))

# dc is the top-left cell of the block, ac are all the other cells
dc = np.empty((blocks_count, 1), dtype=np.int32)
ac = np.empty((blocks_count, 63, 1), dtype=np.int32)
block_index = 0

for i in range(0, rows, 8):
    for j in range(0, cols, 8):
        for k in range(1):
            # split 8x8 block and center the data range on zero
            # [0, 255] --> [-128, 127]
            block = npmat[i:i+8, j:j+8, k] - 128
            dct_matrix = DCT_2D(block)
            quant_matrix = quantize(dct_matrix,quantization_table)
            zz = block_to_zigzag(quant_matrix)

            dc[block_index, k] = zz[0]
            ac[block_index, :, k] = zz[1:]

        block_index += 1

H_DC_Y = HuffmanTree(np.vectorize(bits_required)(dc[:, 0]))
H_AC_Y = HuffmanTree(flatten(run_length_encode(ac[i, :, 0])[0] for i in␣
 ↪range(blocks_count)))

tables = {'dc_y': H_DC_Y.value_to_bitstring_table(), 'ac_y': H_AC_Y.
 ↪value_to_bitstring_table()}

write_to_file(output_file, dc, ac, blocks_count, tables)
write_as_binary(output_file, 'encoded_as_binary')

print('Finished encoding and writing data to file.')


# Printing the original image size and compressed file size
file_name = "original-ps2.png"
file_stats = os.stat(file_name)
size_ori = file_stats.st_size / (1024)
print('Orignal image size is {:.0f} kB'.format(size_ori))
file_name = "encoded_as_binary"
file_stats = os.stat(file_name)
size_compressed = file_stats.st_size / (1024)
print('Compressed file size is {:.0f} kB'.format(size_compressed))
print('Compression ratio is {:.2f}'.format(size_compressed/size_ori))

# Decoding

# Path to the encoded as string file.
encoded_string_file = 'encoded_as_string'

dc, ac, tables, blocks_count = read_image_file(encoded_string_file)
# Assuming that the block is a 8x8 square
block_side = 8

# Assuming that the image height and width are equal
image_side = int(math.sqrt(blocks_count)) * block_side
blocks_per_line = image_side // block_side
npmat = np.empty((image_side, image_side, 1), dtype=np.uint8)

for block_index in range(blocks_count):
    i = block_index // blocks_per_line * block_side
    j = block_index % blocks_per_line * block_side
    for c in range(1):
        zigzag = [dc[block_index, c]] + list(ac[block_index, :, c])
        quant_matrix = zigzag_to_block(zigzag)
        dct_matrix = dequantize(quant_matrix, quantization_table)
        block = IDCT_2D(dct_matrix)
        npmat[i:i+8, j:j+8, c] = (block + 128).clip(0,255)

print('Finished decoding file.')
plt.figure(figsize=(8,8))
plt.imshow(npmat[:,:,0],cmap='gray')
plt.title('Decoded_image')
image = Image.fromarray(npmat[:,:,0], 'L')
image.save('Decoded.png', compress_level=0)
```

Direct Huffman encoding the image, without DCT.

```
[ ]: Direct_encoding_save_name = 'Direct_im_huffman_encoded_as_string'
HuffmanEncode_im(Direct_encoding_save_name, npmat[:,:,0])
write_as_binary(Direct_encoding_save_name,␣
  ↪'Direct_im_huffman_encoded_as_binary')

file_name = 'Direct_im_huffman_encoded_as_binary'
file_stats = os.stat(file_name)
size_direct = file_stats.st_size / (1024)
print('Direct Huffman coded file size is {:.0f} kB'.format(size_direct))
```

6 Convert Notebook to PDF

```
[ ]: # generate pdf
     # Please provide the full path of the notebook file below
     # Important: make sure that your file name does not contain spaces!
     import os
     notebookpath = '/content/drive/My Drive/EECS 442 - Computer Vision/Hw/hw2/
      ↪pohsun_40183452.ipynb' # Ex: notebookpath = '/content/drive/My Drive/Colab␣
      ↪Notebooks/EECS 442 Fall 2024 - PS2.ipynb'
     drive_mount_point = '/content/drive/'
     from google.colab import drive
     drive.mount(drive_mount_point)
     file_name = notebookpath.split('/')[-1]
     get_ipython().system("apt update && apt install texlive-xetex␣
      ↪texlive-fonts-recommended texlive-generic-recommended")
     get_ipython().system("pip install pypandoc")
     get_ipython().system("apt-get install texlive texlive-xetex texlive-latex-extra␣
      ↪pandoc")
     get_ipython().system("jupyter nbconvert --to PDF {}".format(notebookpath.
      ↪replace(' ', '\\ ')))
     from google.colab import files
     files.download(notebookpath.split('.')[0]+'.pdf')
```

```
Mounted at /content/drive/
Hit:1 https://cli.github.com/packages stable InRelease
Get:2 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease
[3,632 B]
Hit:3 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64
InRelease
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:5 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:6 https://r2u.stat.illinois.edu/ubuntu jammy InRelease [6,555 B]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:9 https://r2u.stat.illinois.edu/ubuntu jammy/main all Packages [9,278 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages
[3,311 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages
[1,581 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [3,638
kB]
Get:13 https://r2u.stat.illinois.edu/ubuntu jammy/main amd64 Packages [2,799 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages
[5,665 kB]
Hit:15 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Hit:16 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy
InRelease
```

```
Hit:17 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease
Fetched 26.7 MB in 3s (8,583 kB/s)
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
38 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: Skipping acquire of configured file 'main/source/Sources' as
repository 'https://r2u.stat.illinois.edu/ubuntu jammy InRelease' does not seem
to provide it (sources.list entry misspelt?)
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
E: Unable to locate package texlive-generic-recommended
Collecting pypandoc
  Downloading pypandoc-1.15-py3-none-any.whl.metadata (16 kB)
Downloading pypandoc-1.15-py3-none-any.whl (21 kB)
Installing collected packages: pypandoc
Successfully installed pypandoc-1.15
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
The following additional packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono
  fonts-texgyre fonts-urw-base35 libapache-pom-java
  libcmark-gfm-extensions0.29.0.gfm.3 libcmark-gfm0.29.0.gfm.3
  libcommons-logging-java libcommons-parent-java libfontbox-java libgs9
  libgs9-common libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java
  libptexenc1 libruby3.0 libsynctex2 libteckit0 libtexlua53 libtexluajit2
  libwoff1 libzzip-0-13 lmodern pandoc-data poppler-data preview-latex-style
  rake ruby ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0
  rubygems-integration t1utils teckit tex-common tex-gyre texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-recommended texlive-pictures texlive-plain-generic tipa
  xfonts-encodings xfonts-utils
Suggested packages:
  fonts-noto fonts-freefont-otf | fonts-freefont-ttf libavalon-framework-java
  libcommons-logging-java-doc libexcalibur-logkit-java liblog4j1.2-java
  texlive-luatex pandoc-citeproc context wkhtmltopdf librsvg2-bin groff ghc
  nodejs php python libjs-mathjax libjs-katex citation-style-language-styles
  poppler-utils ghostscript fonts-japanese-mincho | fonts-ipafont-mincho
  fonts-japanese-gothic | fonts-ipafont-gothic fonts-arphic-ukai
  fonts-arphic-uming fonts-nanum ri ruby-dev bundler debhelper gv
  | postscript-viewer perl-tk xpdf | pdf-viewer xzdec
  texlive-fonts-recommended-doc texlive-latex-base-doc python3-pygments
  icc-profiles libfile-which-perl libspreadsheet-parseexcel-perl
  texlive-latex-extra-doc texlive-latex-recommended-doc texlive-pstricks
  dot2tex prerex texlive-pictures-doc vprerex default-jre-headless tipa-doc
The following NEW packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono
  fonts-texgyre fonts-urw-base35 libapache-pom-java
  libcmark-gfm-extensions0.29.0.gfm.3 libcmark-gfm0.29.0.gfm.3
  libcommons-logging-java libcommons-parent-java libfontbox-java libgs9
  libgs9-common libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java
  libptexenc1 libruby3.0 libsynctex2 libteckit0 libtexlua53 libtexluajit2
  libwoff1 libzzip-0-13 lmodern pandoc pandoc-data poppler-data
  preview-latex-style rake ruby ruby-net-telnet ruby-rubygems ruby-webrick
  ruby-xmlrpc ruby3.0 rubygems-integration t1utils teckit tex-common tex-gyre
  texlive texlive-base texlive-binaries texlive-fonts-recommended
  texlive-latex-base texlive-latex-extra texlive-latex-recommended
  texlive-pictures texlive-plain-generic texlive-xetex tipa xfonts-encodings
  xfonts-utils
0 upgraded, 58 newly installed, 0 to remove and 38 not upgraded.
Need to get 202 MB of archives.
After this operation, 728 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-droid-fallback all
1:6.0.1r16-1.1build1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-lato all 2.0-2.1
[2,696 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 poppler-data all
0.4.11-1 [2,171 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tex-common all 6.17
[33.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-urw-base35 all
20200910-1 [6,367 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9-common
all 9.55.0~dfsg1-0ubuntu5.12 [753 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libidn12 amd64
1.38-4ubuntu1 [60.0 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy/main amd64 libijs-0.35 amd64
0.35-15build2 [16.5 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/main amd64 libjbig2dec0 amd64
0.19-3build2 [64.7 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9 amd64
9.55.0~dfsg1-0ubuntu5.12 [5,031 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libkpathsea6
amd64 2021.20210626.59705-1ubuntu0.2 [60.4 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/main amd64 libwoff1 amd64
1.0.2-1build4 [45.2 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy/universe amd64 dvisvgm amd64
2.13.1-1 [1,221 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy/universe amd64 fonts-lmodern all
2.004.5-6.1 [4,532 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-noto-mono all
20201225-1build1 [397 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy/universe amd64 fonts-texgyre all
20180621-3.1 [10.2 MB]
```

```
Get:17 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libapache-pom-java
all 18-1 [4,720 B]
Get:18 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcmark-
gfm0.29.0.gfm.3 amd64 0.29.0.gfm.3-3 [115 kB]
Get:19 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcmark-gfm-
extensions0.29.0.gfm.3 amd64 0.29.0.gfm.3-3 [25.1 kB]
Get:20 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcommons-parent-
java all 43-1 [10.8 kB]
Get:21 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcommons-logging-
java all 1.2-2 [60.3 kB]
Get:22 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libptexenc1
amd64 2021.20210626.59705-1ubuntu0.2 [39.1 kB]
Get:23 http://archive.ubuntu.com/ubuntu jammy/main amd64 rubygems-integration
all 1.18 [5,336 B]
Get:24 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby3.0 amd64
3.0.2-7ubuntu2.11 [50.1 kB]
Get:25 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby-rubygems
all 3.3.5-2ubuntu1.2 [228 kB]
Get:26 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby amd64 1:3.0~exp1
[5,100 B]
Get:27 http://archive.ubuntu.com/ubuntu jammy/main amd64 rake all 13.0.6-2 [61.7
kB]
Get:28 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby-net-telnet all
0.1.1-2 [12.6 kB]
Get:29 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby-webrick
all 1.7.0-3ubuntu0.2 [52.5 kB]
Get:30 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby-xmlrpc all
0.3.2-1ubuntu0.1 [24.9 kB]
Get:31 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libruby3.0
amd64 3.0.2-7ubuntu2.11 [5,114 kB]
Get:32 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libsynctex2
amd64 2021.20210626.59705-1ubuntu0.2 [55.6 kB]
Get:33 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libteckit0 amd64
2.5.11+ds1-1 [421 kB]
Get:34 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtexlua53
amd64 2021.20210626.59705-1ubuntu0.2 [120 kB]
Get:35 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtexluajit2
amd64 2021.20210626.59705-1ubuntu0.2 [267 kB]
Get:36 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libzzip-0-13 amd64
0.13.72+dfsg.1-1.1 [27.0 kB]
Get:37 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-encodings all
1:1.0.5-0ubuntu2 [578 kB]
Get:38 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-utils amd64
1:7.7+6build2 [94.6 kB]
Get:39 http://archive.ubuntu.com/ubuntu jammy/universe amd64 lmodern all
2.004.5-6.1 [9,471 kB]
Get:40 http://archive.ubuntu.com/ubuntu jammy/universe amd64 pandoc-data all
2.9.2.1-3ubuntu2 [81.8 kB]
```

```
Get:41 http://archive.ubuntu.com/ubuntu jammy/universe amd64 pandoc amd64
2.9.2.1-3ubuntu2 [20.3 MB]
Get:42 http://archive.ubuntu.com/ubuntu jammy/universe amd64 preview-latex-style
all 12.2-1ubuntu1 [185 kB]
Get:43 http://archive.ubuntu.com/ubuntu jammy/main amd64 t1utils amd64
1.41-4build2 [61.3 kB]
Get:44 http://archive.ubuntu.com/ubuntu jammy/universe amd64 teckit amd64
2.5.11+ds1-1 [699 kB]
Get:45 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tex-gyre all
20180621-3.1 [6,209 kB]
Get:46 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 texlive-
binaries amd64 2021.20210626.59705-1ubuntu0.2 [9,860 kB]
Get:47 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-base all
2021.20220204-1 [21.0 MB]
Get:48 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-fonts-
recommended all 2021.20220204-1 [4,972 kB]
Get:49 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-base
all 2021.20220204-1 [1,128 kB]
Get:50 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-
recommended all 2021.20220204-1 [14.4 MB]
Get:51 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive all
2021.20220204-1 [14.3 kB]
Get:52 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libfontbox-java all
1:1.8.16-2 [207 kB]
Get:53 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libpdfbox-java all
1:1.8.16-2 [5,199 kB]
Get:54 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-pictures
all 2021.20220204-1 [8,720 kB]
Get:55 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-extra
all 2021.20220204-1 [13.9 MB]
Get:56 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-plain-
generic all 2021.20220204-1 [27.5 MB]
Get:57 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tipa all 2:1.3-21
[2,967 kB]
Get:58 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-xetex all
2021.20220204-1 [12.4 MB]
Fetched 202 MB in 4s (45.0 MB/s)
Extracting templates from packages: 100%
Preconfiguring packages …
Selecting previously unselected package fonts-droid-fallback.
(Reading database … 126435 files and directories currently installed.)
Preparing to unpack …/00-fonts-droid-fallback_1%3a6.0.1r16-1.1build1_all.deb
…
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1build1) …
Selecting previously unselected package fonts-lato.
Preparing to unpack …/01-fonts-lato_2.0-2.1_all.deb …
Unpacking fonts-lato (2.0-2.1) …
Selecting previously unselected package poppler-data.
```

```
Preparing to unpack …/02-poppler-data_0.4.11-1_all.deb …
Unpacking poppler-data (0.4.11-1) …
Selecting previously unselected package tex-common.
Preparing to unpack …/03-tex-common_6.17_all.deb …
Unpacking tex-common (6.17) …
Selecting previously unselected package fonts-urw-base35.
Preparing to unpack …/04-fonts-urw-base35_20200910-1_all.deb …
Unpacking fonts-urw-base35 (20200910-1) …
Selecting previously unselected package libgs9-common.
Preparing to unpack …/05-libgs9-common_9.55.0~dfsg1-0ubuntu5.12_all.deb …
Unpacking libgs9-common (9.55.0~dfsg1-0ubuntu5.12) …
Selecting previously unselected package libidn12:amd64.
Preparing to unpack …/06-libidn12_1.38-4ubuntu1_amd64.deb …
Unpacking libidn12:amd64 (1.38-4ubuntu1) …
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack …/07-libijs-0.35_0.35-15build2_amd64.deb …
Unpacking libijs-0.35:amd64 (0.35-15build2) …
Selecting previously unselected package libjbig2dec0:amd64.
Preparing to unpack …/08-libjbig2dec0_0.19-3build2_amd64.deb …
Unpacking libjbig2dec0:amd64 (0.19-3build2) …
Selecting previously unselected package libgs9:amd64.
Preparing to unpack …/09-libgs9_9.55.0~dfsg1-0ubuntu5.12_amd64.deb …
Unpacking libgs9:amd64 (9.55.0~dfsg1-0ubuntu5.12) …
Selecting previously unselected package libkpathsea6:amd64.
Preparing to unpack …/10-libkpathsea6_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libkpathsea6:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libwoff1:amd64.
Preparing to unpack …/11-libwoff1_1.0.2-1build4_amd64.deb …
Unpacking libwoff1:amd64 (1.0.2-1build4) …
Selecting previously unselected package dvisvgm.
Preparing to unpack …/12-dvisvgm_2.13.1-1_amd64.deb …
Unpacking dvisvgm (2.13.1-1) …
Selecting previously unselected package fonts-lmodern.
Preparing to unpack …/13-fonts-lmodern_2.004.5-6.1_all.deb …
Unpacking fonts-lmodern (2.004.5-6.1) …
Selecting previously unselected package fonts-noto-mono.
Preparing to unpack …/14-fonts-noto-mono_20201225-1build1_all.deb …
Unpacking fonts-noto-mono (20201225-1build1) …
Selecting previously unselected package fonts-texgyre.
Preparing to unpack …/15-fonts-texgyre_20180621-3.1_all.deb …
Unpacking fonts-texgyre (20180621-3.1) …
Selecting previously unselected package libapache-pom-java.
Preparing to unpack …/16-libapache-pom-java_18-1_all.deb …
Unpacking libapache-pom-java (18-1) …
Selecting previously unselected package libcmark-gfm0.29.0.gfm.3:amd64.
Preparing to unpack …/17-libcmark-gfm0.29.0.gfm.3_0.29.0.gfm.3-3_amd64.deb …
Unpacking libcmark-gfm0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
```

```
Selecting previously unselected package libcmark-gfm-
extensions0.29.0.gfm.3:amd64.
Preparing to unpack …/18-libcmark-gfm-
extensions0.29.0.gfm.3_0.29.0.gfm.3-3_amd64.deb …
Unpacking libcmark-gfm-extensions0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
Selecting previously unselected package libcommons-parent-java.
Preparing to unpack …/19-libcommons-parent-java_43-1_all.deb …
Unpacking libcommons-parent-java (43-1) …
Selecting previously unselected package libcommons-logging-java.
Preparing to unpack …/20-libcommons-logging-java_1.2-2_all.deb …
Unpacking libcommons-logging-java (1.2-2) …
Selecting previously unselected package libptexenc1:amd64.
Preparing to unpack …/21-libptexenc1_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libptexenc1:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package rubygems-integration.
Preparing to unpack …/22-rubygems-integration_1.18_all.deb …
Unpacking rubygems-integration (1.18) …
Selecting previously unselected package ruby3.0.
Preparing to unpack …/23-ruby3.0_3.0.2-7ubuntu2.11_amd64.deb …
Unpacking ruby3.0 (3.0.2-7ubuntu2.11) …
Selecting previously unselected package ruby-rubygems.
Preparing to unpack …/24-ruby-rubygems_3.3.5-2ubuntu1.2_all.deb …
Unpacking ruby-rubygems (3.3.5-2ubuntu1.2) …
Selecting previously unselected package ruby.
Preparing to unpack …/25-ruby_1%3a3.0~exp1_amd64.deb …
Unpacking ruby (1:3.0~exp1) …
Selecting previously unselected package rake.
Preparing to unpack …/26-rake_13.0.6-2_all.deb …
Unpacking rake (13.0.6-2) …
Selecting previously unselected package ruby-net-telnet.
Preparing to unpack …/27-ruby-net-telnet_0.1.1-2_all.deb …
Unpacking ruby-net-telnet (0.1.1-2) …
Selecting previously unselected package ruby-webrick.
Preparing to unpack …/28-ruby-webrick_1.7.0-3ubuntu0.2_all.deb …
Unpacking ruby-webrick (1.7.0-3ubuntu0.2) …
Selecting previously unselected package ruby-xmlrpc.
Preparing to unpack …/29-ruby-xmlrpc_0.3.2-1ubuntu0.1_all.deb …
Unpacking ruby-xmlrpc (0.3.2-1ubuntu0.1) …
Selecting previously unselected package libruby3.0:amd64.
Preparing to unpack …/30-libruby3.0_3.0.2-7ubuntu2.11_amd64.deb …
Unpacking libruby3.0:amd64 (3.0.2-7ubuntu2.11) …
Selecting previously unselected package libsynctex2:amd64.
Preparing to unpack …/31-libsynctex2_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libsynctex2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libteckit0:amd64.
Preparing to unpack …/32-libteckit0_2.5.11+ds1-1_amd64.deb …
```

```
Unpacking libteckit0:amd64 (2.5.11+ds1-1) …
Selecting previously unselected package libtexlua53:amd64.
Preparing to unpack …/33-libtexlua53_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libtexlua53:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libtexluajit2:amd64.
Preparing to unpack
…/34-libtexluajit2_2021.20210626.59705-1ubuntu0.2_amd64.deb …
Unpacking libtexluajit2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libzzip-0-13:amd64.
Preparing to unpack …/35-libzzip-0-13_0.13.72+dfsg.1-1.1_amd64.deb …
Unpacking libzzip-0-13:amd64 (0.13.72+dfsg.1-1.1) …
Selecting previously unselected package xfonts-encodings.
Preparing to unpack …/36-xfonts-encodings_1%3a1.0.5-0ubuntu2_all.deb …
Unpacking xfonts-encodings (1:1.0.5-0ubuntu2) …
Selecting previously unselected package xfonts-utils.
Preparing to unpack …/37-xfonts-utils_1%3a7.7+6build2_amd64.deb …
Unpacking xfonts-utils (1:7.7+6build2) …
Selecting previously unselected package lmodern.
Preparing to unpack …/38-lmodern_2.004.5-6.1_all.deb …
Unpacking lmodern (2.004.5-6.1) …
Selecting previously unselected package pandoc-data.
Preparing to unpack …/39-pandoc-data_2.9.2.1-3ubuntu2_all.deb …
Unpacking pandoc-data (2.9.2.1-3ubuntu2) …
Selecting previously unselected package pandoc.
Preparing to unpack …/40-pandoc_2.9.2.1-3ubuntu2_amd64.deb …
Unpacking pandoc (2.9.2.1-3ubuntu2) …
Selecting previously unselected package preview-latex-style.
Preparing to unpack …/41-preview-latex-style_12.2-1ubuntu1_all.deb …
Unpacking preview-latex-style (12.2-1ubuntu1) …
Selecting previously unselected package t1utils.
Preparing to unpack …/42-t1utils_1.41-4build2_amd64.deb …
Unpacking t1utils (1.41-4build2) …
Selecting previously unselected package teckit.
Preparing to unpack …/43-teckit_2.5.11+ds1-1_amd64.deb …
Unpacking teckit (2.5.11+ds1-1) …
Selecting previously unselected package tex-gyre.
Preparing to unpack …/44-tex-gyre_20180621-3.1_all.deb …
Unpacking tex-gyre (20180621-3.1) …
Selecting previously unselected package texlive-binaries.
Preparing to unpack …/45-texlive-
binaries_2021.20210626.59705-1ubuntu0.2_amd64.deb …
Unpacking texlive-binaries (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package texlive-base.
Preparing to unpack …/46-texlive-base_2021.20220204-1_all.deb …
Unpacking texlive-base (2021.20220204-1) …
Selecting previously unselected package texlive-fonts-recommended.
Preparing to unpack …/47-texlive-fonts-recommended_2021.20220204-1_all.deb …
```

```
Unpacking texlive-fonts-recommended (2021.20220204-1) …
Selecting previously unselected package texlive-latex-base.
Preparing to unpack …/48-texlive-latex-base_2021.20220204-1_all.deb …
Unpacking texlive-latex-base (2021.20220204-1) …
Selecting previously unselected package texlive-latex-recommended.
Preparing to unpack …/49-texlive-latex-recommended_2021.20220204-1_all.deb …
Unpacking texlive-latex-recommended (2021.20220204-1) …
Selecting previously unselected package texlive.
Preparing to unpack …/50-texlive_2021.20220204-1_all.deb …
Unpacking texlive (2021.20220204-1) …
Selecting previously unselected package libfontbox-java.
Preparing to unpack …/51-libfontbox-java_1%3a1.8.16-2_all.deb …
Unpacking libfontbox-java (1:1.8.16-2) …
Selecting previously unselected package libpdfbox-java.
Preparing to unpack …/52-libpdfbox-java_1%3a1.8.16-2_all.deb …
Unpacking libpdfbox-java (1:1.8.16-2) …
Selecting previously unselected package texlive-pictures.
Preparing to unpack …/53-texlive-pictures_2021.20220204-1_all.deb …
Unpacking texlive-pictures (2021.20220204-1) …
Selecting previously unselected package texlive-latex-extra.
Preparing to unpack …/54-texlive-latex-extra_2021.20220204-1_all.deb …
Unpacking texlive-latex-extra (2021.20220204-1) …
Selecting previously unselected package texlive-plain-generic.
Preparing to unpack …/55-texlive-plain-generic_2021.20220204-1_all.deb …
Unpacking texlive-plain-generic (2021.20220204-1) …
Selecting previously unselected package tipa.
Preparing to unpack …/56-tipa_2%3a1.3-21_all.deb …
Unpacking tipa (2:1.3-21) …
Selecting previously unselected package texlive-xetex.
Preparing to unpack …/57-texlive-xetex_2021.20220204-1_all.deb …
Unpacking texlive-xetex (2021.20220204-1) …
Setting up fonts-lato (2.0-2.1) …
Setting up fonts-noto-mono (20201225-1build1) …
Setting up libwoff1:amd64 (1.0.2-1build4) …
Setting up libtexlua53:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up libijs-0.35:amd64 (0.35-15build2) …
Setting up libtexluajit2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up libfontbox-java (1:1.8.16-2) …
Setting up rubygems-integration (1.18) …
Setting up libzzip-0-13:amd64 (0.13.72+dfsg.1-1.1) …
Setting up fonts-urw-base35 (20200910-1) …
Setting up poppler-data (0.4.11-1) …
Setting up tex-common (6.17) …
update-language: texlive-base not installed and configured, doing nothing!
Setting up libjbig2dec0:amd64 (0.19-3build2) …
Setting up libteckit0:amd64 (2.5.11+ds1-1) …
Setting up libapache-pom-java (18-1) …
Setting up ruby-net-telnet (0.1.1-2) …
Setting up xfonts-encodings (1:1.0.5-0ubuntu2) …
Setting up t1utils (1.41-4build2) …
Setting up libidn12:amd64 (1.38-4ubuntu1) …
Setting up fonts-texgyre (20180621-3.1) …
Setting up libkpathsea6:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up ruby-webrick (1.7.0-3ubuntu0.2) …
Setting up libcmark-gfm0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
Setting up fonts-lmodern (2.004.5-6.1) …
Setting up libcmark-gfm-extensions0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
Setting up fonts-droid-fallback (1:6.0.1r16-1.1build1) …
Setting up pandoc-data (2.9.2.1-3ubuntu2) …
Setting up ruby-xmlrpc (0.3.2-1ubuntu0.1) …
Setting up libsynctex2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up libgs9-common (9.55.0~dfsg1-0ubuntu5.12) …
Setting up teckit (2.5.11+ds1-1) …
Setting up libpdfbox-java (1:1.8.16-2) …
Setting up libgs9:amd64 (9.55.0~dfsg1-0ubuntu5.12) …
Setting up preview-latex-style (12.2-1ubuntu1) …
Setting up libcommons-parent-java (43-1) …
Setting up dvisvgm (2.13.1-1) …
Setting up libcommons-logging-java (1.2-2) …
Setting up xfonts-utils (1:7.7+6build2) …
Setting up libptexenc1:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up pandoc (2.9.2.1-3ubuntu2) …
Setting up texlive-binaries (2021.20210626.59705-1ubuntu0.2) …
update-alternatives: using /usr/bin/xdvi-xaw to provide /usr/bin/xdvi.bin
(xdvi.bin) in auto mode
update-alternatives: using /usr/bin/bibtex.original to provide /usr/bin/bibtex
(bibtex) in auto mode
Setting up lmodern (2.004.5-6.1) …
Setting up texlive-base (2021.20220204-1) …
/usr/bin/ucfr
/usr/bin/ucfr
/usr/bin/ucfr
/usr/bin/ucfr
mktexlsr: Updating /var/lib/texmf/ls-R-TEXLIVEDIST…
mktexlsr: Updating /var/lib/texmf/ls-R-TEXMFMAIN…
mktexlsr: Updating /var/lib/texmf/ls-R…
mktexlsr: Done.
tl-paper: setting paper size for dvips to a4:
/var/lib/texmf/dvips/config/config-paper.ps
tl-paper: setting paper size for dvipdfmx to a4:
/var/lib/texmf/dvipdfmx/dvipdfmx-paper.cfg
tl-paper: setting paper size for xdvi to a4: /var/lib/texmf/xdvi/XDvi-paper
tl-paper: setting paper size for pdftex to a4: /var/lib/texmf/tex/generic/tex-
ini-files/pdftexconfig.tex
Setting up tex-gyre (20180621-3.1) …
Setting up texlive-plain-generic (2021.20220204-1) …
Setting up texlive-latex-base (2021.20220204-1) …
Setting up texlive-latex-recommended (2021.20220204-1) …
Setting up texlive-pictures (2021.20220204-1) …
Setting up texlive-fonts-recommended (2021.20220204-1) …
Setting up tipa (2:1.3-21) …
Setting up texlive (2021.20220204-1) …
Setting up texlive-latex-extra (2021.20220204-1) …
Setting up texlive-xetex (2021.20220204-1) …
Setting up rake (13.0.6-2) …
Setting up libruby3.0:amd64 (3.0.2-7ubuntu2.11) …
Setting up ruby3.0 (3.0.2-7ubuntu2.11) …
Setting up ruby (1:3.0~exp1) …
Setting up ruby-rubygems (3.3.5-2ubuntu1.2) …
Processing triggers for man-db (2.10.2-1) …
Processing triggers for mailcap (3.70+nmu1ubuntu1) …
Processing triggers for fontconfig (2.13.1-4.2ubuntu5) …
Processing triggers for libc-bin (2.35-0ubuntu3.8) …
/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtcm.so.1 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc_proxy.so.2 is not a symbolic
link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtcm_debug.so.1 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_level_zero_v2.so.0 is not a
symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_opencl.so.0 is not a symbolic
link

/sbin/ldconfig.real: /usr/local/lib/libur_loader.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_5.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libhwloc.so.15 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_level_zero.so.0 is not a
symbolic link

/sbin/ldconfig.real: /usr/local/lib/libumf.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link
```