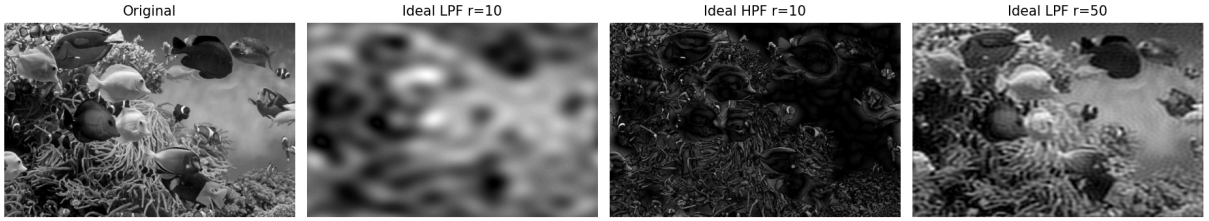
Image Processing Homework 3

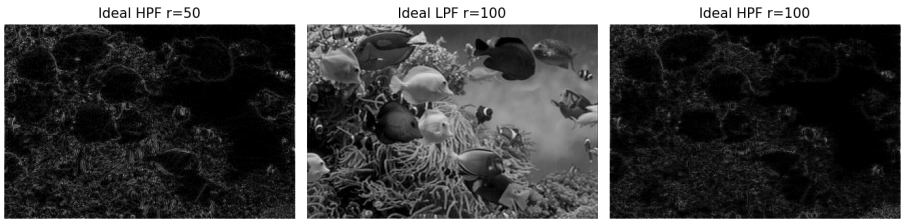
1. Write a program to enhance the “fish.jpg” and “cabin.jpg” images in frequency domain using the following filters:
    a. Notch filters
        i. Low-pass filter with radius, r equals to 10, 50, and 100, respectively.
        ii. High-pass filter with radius, r equals to 10, 50, and 100, respectively.
    b. Gaussian filters
        i. Low-pass filter with cutoff, $D0$ equals to 10, 50, and 100, respectively.
        ii. High-pass filter with cutoff, $D0$ equals to 10, 50, and 100, respectively.

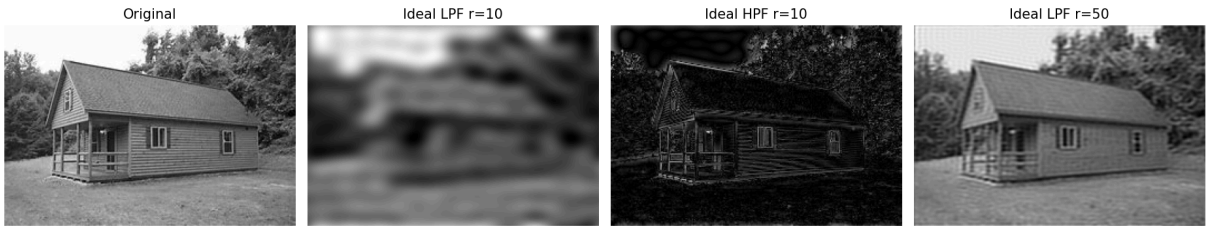
code

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

def make_ideal_filter(shape, r, filter_type="low"):
    rows, cols = shape
    crow, ccol = rows//2, cols//2
    u = np.arange(rows)
    v = np.arange(cols)
    U, V = np.meshgrid(u - crow, v - ccol, indexing='ij')
    D = np.sqrt(U**2 + V**2)
    if filter_type == "low":
        H = np.float32(D <= r)
    else:  # high-pass
        H = np.float32(D > r)
    return H

def make_gaussian_filter(shape, D0, filter_type="low"):
    rows, cols = shape
    crow, ccol = rows//2, cols//2
    u = np.arange(rows)
    v = np.arange(cols)
    U, V = np.meshgrid(u - crow, v - ccol, indexing='ij')
    D2 = U**2 + V**2
    if filter_type == "low":
        H = np.exp(-D2 / (2 * (D0**2)))
    else: # high-pass
        H = 1 - np.exp(-D2 / (2 * (D0**2)))
    return H.astype(np.float32)


def apply_filter(img, H):
    f = np.fft.fft2(img)
    fshift = np.fft.fftshift(f)
    G = fshift * H
    f_ishift = np.fft.ifftshift(G)
    img_back = np.fft.ifft2(f_ishift)
    return np.abs(img_back)

def show_results(img, filtered_list, titles):
    plt.figure(figsize=(15,8))
    plt.subplot(2, len(filtered_list)//2+1, 1)
    plt.imshow(img, cmap='gray')
    plt.title("Original")
    plt.axis("off")
    for i, f_img in enumerate(filtered_list):
        plt.subplot(2, len(filtered_list)//2+1, i+2)
        plt.imshow(f_img, cmap='gray')
        plt.title(titles[i])
        plt.axis("off")
    plt.tight_layout()
    plt.show()


fish = cv2.imread("fish.jpg", 0)
cabin = cv2.imread("cabin.jpg", 0)

# notch filter
for img, name in [(fish,"Fish"), (cabin,"Cabin")]:
    results = []
    titles = []
    for r in [10,50,100]:
        H_low = make_ideal_filter(img.shape, r, "low")
        H_high = make_ideal_filter(img.shape, r, "high")
        results.append(apply_filter(img, H_low))
        results.append(apply_filter(img, H_high))
        titles.append(f"Ideal LPF r={r}")
        titles.append(f"Ideal HPF r={r}")
    show_results(img, results, titles)

# gaussian filter
for img, name in [(fish,"Fish"), (cabin,"Cabin")]:
    results = []
    titles = []
    for D0 in [10,50,100]:
        H_low = make_gaussian_filter(img.shape, D0, "low")
        H_high = make_gaussian_filter(img.shape, D0, "high")
        results.append(apply_filter(img, H_low))
        results.append(apply_filter(img, H_high))
        titles.append(f"Gaussian LPF D0={D0}")
        titles.append(f"Gaussian HPF D0={D0}")
    show_results(img, results, titles)
```

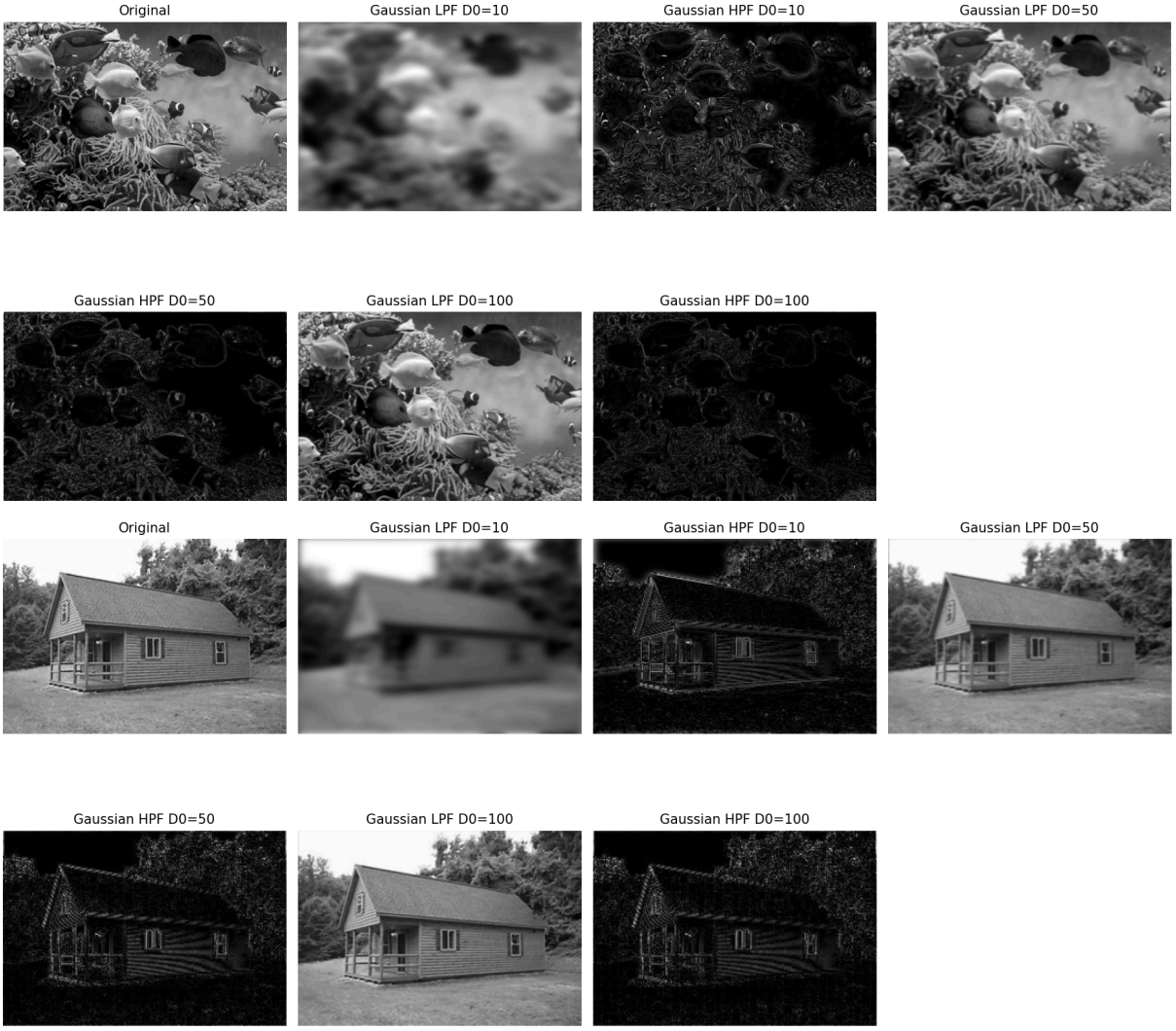
result North filter:







result Gaussian:



3. Remove periodic noise from the given images using any of the filters that you think are the most suitable ones to be used.

code:

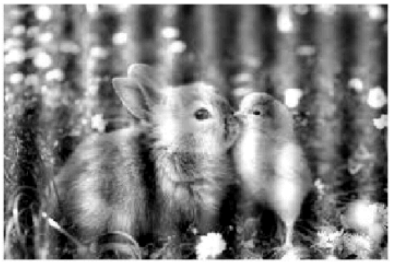
```python
# 3.Remove periodic noise
# ใช้ Fourier Transform (FFT) → ดู spectrum หาแบบ noisy
def fourier_spectrum(img):
    f = np.fft.fft2(img)
    fshift = np.fft.fftshift(f)
    spectrum = np.log(1 + np.abs(fshift)) ทำให้เห็น spectrum ชัดขึ้น
    return fshift, spectrum

def show_spectrum(img, title="Spectrum"):
    _, spectrum = fourier_spectrum(img)
    plt.figure(figsize=(5,5))
    plt.imshow(spectrum, cmap='gray')
    plt.title(title)
    plt.axis("off")
    plt.show()

# Inverse FFT → กลับมาเป็นภาพที่ลบสัญญาณที่ผิดปกติออก
def apply_notch(img, mask):
    fshift, _ = fourier_spectrum(img)
    # apply mask
    G = fshift * mask
    # inverse FFT
    f_ishift = np.fft.ifftshift(G)
    img_back = np.abs(np.fft.ifft2(f_ishift))
    return img_back

def axis_mask(shape, axis='hori', points=[], band=1):
    mask = np.ones(shape, np.uint8)
    for point in points:
        if axis == 'hori':
            mask[point-band:point+band,:] = 0
        elif axis == 'vert':
            mask[:,point-band:point+band] = 0
    return mask

def circle_mask(shape, center, r):
    mask = np.ones(shape, np.uint8)
    for c in center:
        cv2.circle(mask, c, r, 0,  1)
    return mask

original = cv2.imread("animals.jpg", 0)
diag = cv2.imread("animals_diag_noise.jpg", 0)
hori = cv2.imread("animals_hori_noise.jpg", 0)
vert = cv2.imread("animals_vert_noise.jpg", 0)

# # เลือก horiL จาก spectrum ที่เห็นแถบสว่าง
# diag_spectrum = show_spectrum(diag, "Diagonal Noise Spectrum") #yx(106, 189)(244, 180)
# hori_spectrum = show_spectrum(hori, "Horizontal Noise Spectrum") # 154 134
# vert_spectrum = show_spectrum(vert, "Vertical Noise Spectrum") # 206 226

mask_hori = axis_mask(hori.shape, 'hori', points=[134,154], band=1)
mask_vert = axis_mask(vert.shape, 'vert', points=[206,226], band=1)
mask_diag = axis_mask(hori.shape, 'hori', points=[180,244], band=2)*axis_mask(hori.shape, 'vert', points=[107,189,35,90,341,390], band=2)

# Diagonal noise
diag_restored = apply_notch(diag, mask_diag)
# diag_restored_spectrum = show_spectrum(diag_restored, "Restored Diagonal Spectrum")

# 🔹 Horizontal noise → แถบแนวนอน
hori_restored = apply_notch(hori, mask_hori)
# hori_restored_spectrum = show_spectrum(hori_restored, "Restored Horizontal Spectrum")

# 🔹 Vertical noise → แถบแนวตั้ง
vert_restored = apply_notch(vert, mask_vert)
# vert_restored_spectrum = show_spectrum(vert_restored, "Restored Vertical Spectrum")

plt.figure(figsize=(12,10))

plt.subplot(3,3,1); plt.imshow(diag, cmap='gray'); plt.title("Diagonal Noise"); plt.axis("off")
plt.subplot(3,3,2); plt.imshow(hori, cmap='gray'); plt.title("Horizontal Noise"); plt.axis("off")
plt.subplot(3,3,3); plt.imshow(vert, cmap='gray'); plt.title("Vertical Noise"); plt.axis("off")

plt.subplot(3,3,4); plt.imshow(diag_restored, cmap='gray'); plt.title("Restored Diagonal"); plt.axis("off")
plt.subplot(3,3,5); plt.imshow(hori_restored, cmap='gray'); plt.title("Restored Horizontal"); plt.axis("off")
plt.subplot(3,3,6); plt.imshow(vert_restored, cmap='gray'); plt.title("Restored Vertical"); plt.axis("off")

plt.subplot(3,3,7); plt.imshow(original, cmap='gray'); plt.title("Original"); plt.axis("off")

plt.tight_layout()
plt.show()
```

Result :

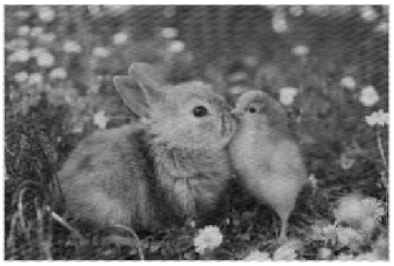
Diagonal Noise

Horizontal Noise

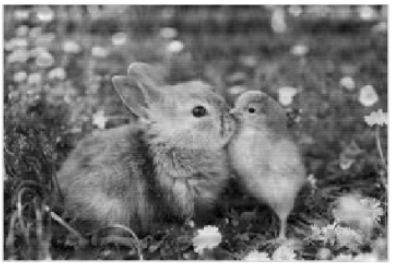
Vertical Noise



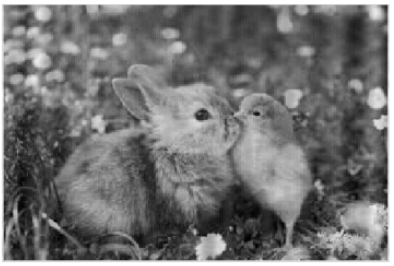
Restored Diagonal



Restored Horizontal



Restored Vertical



Original