بسم الله الرحمن الرحیم

گزارش کار تمرین ۳ – بینایی کامپیوتر

عنوان تمرین :

پردازش در حوزه فرکانس

دانشجو : محمدداود وهاب رجائی

شماره دانشجویی : ۴۰۴۱۴۱۹۰۴۱

استاد محترم : دکتر علی جعفری

پاییز ۱۴۰۴

فهرست مطالب

مقدمه

هدف از این تمرین، آشنایی عملی با تبدیل فوریه (Fourier Transform) و پردازش تصویر در حوزه فرکانس است. در این تمرین، ما یاد می‌گیریم که چگونه ویژگی‌های یک تصویر (مانند نویز یا بافت) در فضای فرکانس نمایش داده می‌شوند و چگونه می‌توان با دستکاری طیف فرکانسی (Spectrum) ، تصویر نهایی را در فضای مکانی تغییر داد.

بخش اول : ایجاد نویز پریودیک روی تصویر X-Ray

تعریف مسئله

در این بخش، هدف این است که یک تصویر پزشکی (X-Ray) را به فضای فرکانس برده و با اعمال تغییراتی در طیف آن، نویزی ایجاد کنیم که در تصویر خروجی به صورت خطوط مورب Diagonal) (Stripesدیده شود.

الگوریتم و روش پیاده‌سازی

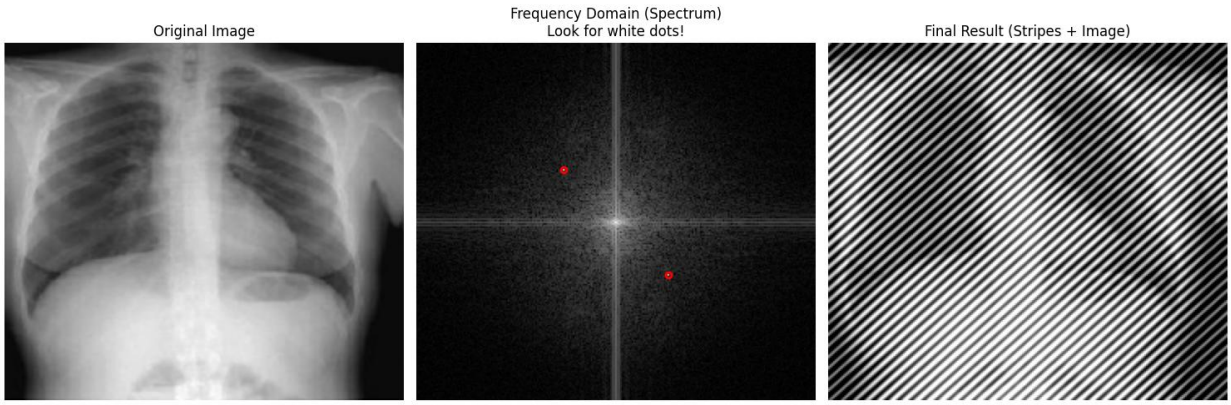
برای انجام این کار از زبان پایتون و کتابخانه OpenCV و NumPy استفاده شده است. مراحل الگوریتم به شرح زیر است :

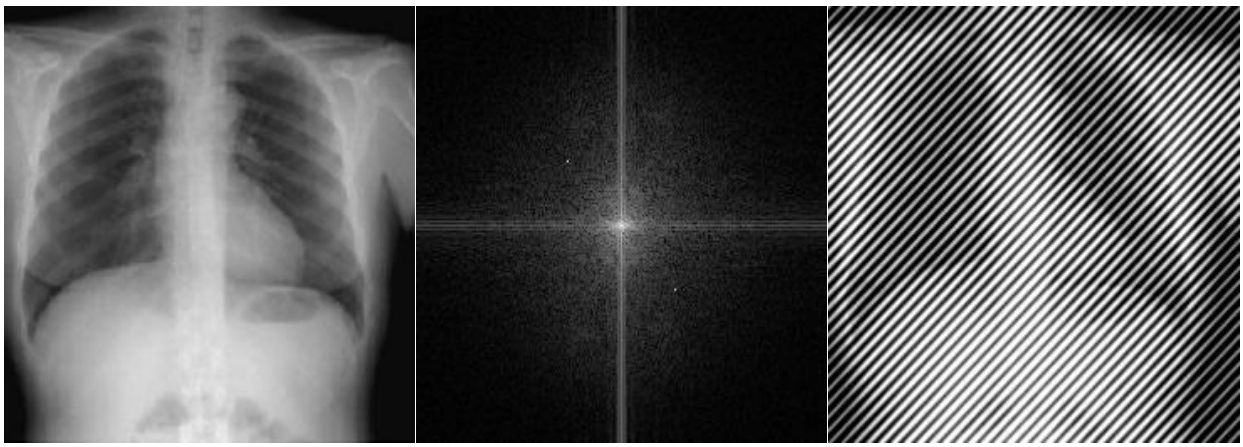
1. **خواندن تصویر :** تصویر ورودی به صورت Grayscale خوانده شده و نرمالایز می‌شود.
2. **تبدیل فوریه (FFT) :** با استفاده از np.fft.fft۲ تصویر به فضای فرکانس برده می‌شود و با fftshiftمولفه DC (فرکانس صفر) به مرکز ماتریس منتقل می‌شود.
3. **ایجاد نویز :** از آنجا که نویزهای پریودیک در فضای مکانی معادل نقاط روشن (Impulse) در فضای فرکانس هستند، دو نقطه متقارن نسبت به مرکز مختصات با شدت بالا در طیف فرکانسی ایجاد شد.
   - مختصات نقاط : (center_row ± ۳۲, center_col ± ۳۲)
   - این مختصات باعث ایجاد امواج مورب می‌شود.

4. **معکوس تبدیل فوریه (IFFT) :** پس از اعمال تغییرات، با استفاده از تبدیل معکوس، تصویر به فضای مکانی بازگردانده شد.

نتایج و تصاویر

در تصویر زیر، تصویر اصلی، طیف فرکانسی (با نقاط نویز مشخص شده) و تصویر خروجی را مشاهده می‌کنید :





تحلیل فنی

تصاویر در حوزه فرکانس بر اساس تغییرات شدت روشنایی تحلیل می‌شوند. نقاط نزدیک به مرکز نشان‌دهنده تغییرات کند (فرکانس پایین) و نقاط دور از مرکز نشان‌دهنده تغییرات سریع (جزئیات و لبه‌ها) هستند.

در اینجا ما دو نقطه (Peak) مصنوعی در فرکانس‌های میانی اضافه کردیم. طبق خاصیت تبدیل فوریه، یک جفت ضربه (Impulse) در فضای فرکانس معادل یک موج سینوسی (یا کسینوسی) در تمام فضای تصویر است.

- چون نقاط به صورت قطری نسبت به مرکز قرار دارند، موج‌ها نیز مورب ظاهر شدند.
- فاصله نقاط از مرکز، فرکانس (تراکم) خطوط را تعیین می‌کند؛ هرچه نقاط دورتر باشند، خطوط فشرده‌تر می‌شوند.

بخش دوم : انتقال بافت موج (Ripples) به تصویر دیگر

تعریف مسئله

هدف این بخش استخراج بافت "امواج آب" از یک تصویر خاکستری و اعمال آن روی یک تصویر زمینه سبز رنگ است، به طوری که گویی امواج روی آن سطح سبز ایجاد شده‌اند.

الگوریتم و روش پیاده‌سازی

برای انتقال بافت بدون انتقال رنگ خاکستری اصلی، باید از فیلترینگ فرکانسی استفاده کرد:

1. **پیش‌پردازش :**تصویر امواج (٣.png) هم‌اندازه تصویر سبز (٤.png) می‌شود.
2. **تبدیل فوریه :** تصویر امواج به فضای فرکانس برده می‌شود.

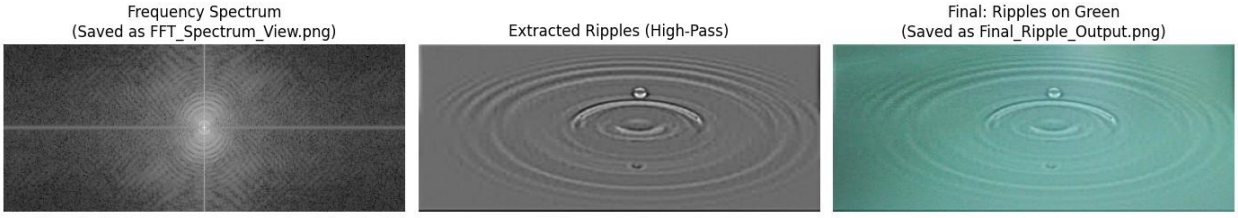
3. **فیلتر بالاگذر (High-Pass Filter)**

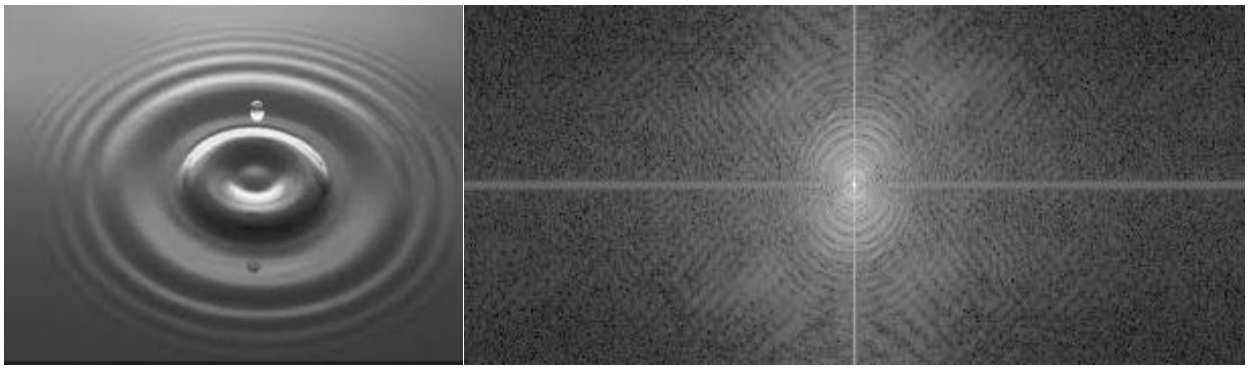
- اطلاعات روشنایی کلی و رنگ زمینه در فرکانس‌های پایین (نزدیک مرکز) قرار دارند.
- جزئیات لبه‌ها و امواج در فرکانس‌های بالا قرار دارند.
- یک ماسک گوسی معکوس ساخته شد تا فرکانس‌های پایین (مرکز) را حذف و فرکانس‌های بالا را عبور دهد.

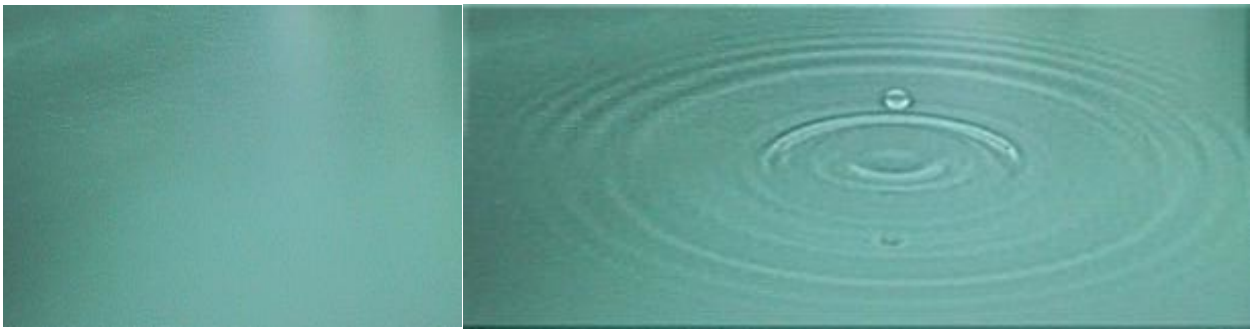
4. **بازسازی و ترکیب :**پس از اعمال فیلتر و بازگشت به فضای مکانی (IFFT) تنها "لبه‌های امواج" باقی ماندند. این لبه‌ها با ضریب تقویت (Gain) روی کانال‌های رنگی تصویر سبز جمع شدند.

نتایج و تصاویر

در تصویر زیر طیف فرکانسی، بافت استخراج شده و نتیجه نهایی ترکیب را مشاهده می‌کنید :







تحلیل فنی

چالش اصلی در این تمرین، جدا کردن "بافت موج" از "رنگ خاکستری پس‌زمینه" بود. اگر تصویر موج را مستقیماً روی تصویر سبز جمع می‌کردیم، تصویر تیره و خاکستری می‌شد.

با استفاده از فیلتر بالاگذر (High-Pass) ، مولفه DC و فرکانس‌های پایین (که مسئول روشنایی یکنواخت پس‌زمینه بودند) حذف شدند. نتیجه‌ی IFFT یک تصویر بود که میانگین روشنایی آن صفر است (شامل مقادیر مثبت و منفی برای لبه‌های موج).

وقتی این مقادیر به تصویر سبز اضافه شدند :

- مقادیر مثبت باعث روشن‌تر شدن سبز (قله موج).
- مقادیر منفی باعث تیره‌تر شدن سبز (قعر موج) این فرآیند باعث ایجاد افکت سه بعدی و طبیعی امواج روی سطح سبز شد.

نتیجه‌گیری کلی

این تمرین نشان داد که فضای فرکانس ابزاری قدرتمند برای دستکاری ویژگی‌های سراسری تصویر است. ما توانستیم با افزودن مولفه‌های فرکانسی، نویز ساختاری ایجاد کنیم و با حذف مولفه‌های فرکانس پایین، بافت را از پس‌زمینه جدا کرده و عملیات ترکیب تصاویر (Texture Synthesis) را انجام دهیم.

پیوست : کدهای پیاده‌سازی شده

در این بخش، کدهای پایتون که برای انجام بخش اول و دوم تمرین نوشته شده‌اند، آورده شده است.

کد مربوط به بخش اول ۱.py : ایجاد نویز روی تصویر X-Ray

```
import numpy as np
import matplotlib
try:
    matplotlib.use('TkAgg')
except:
    pass
import matplotlib.pyplot as plt
import cv2
import os

# --- 1. Load Image ---
img_clean = cv2.imread('1.png', cv2.IMREAD_GRAYSCALE)

if img_clean is None:
    print("Error: '1.png' not found.")
    exit()

# Normalize
img_float = img_clean.astype(np.float64) / 255.0
rows, cols = img_float.shape
center_r, center_c = rows // 2, cols // 2
# --- 2. FFT (Frequency Domain) ---
f_transform = np.fft.fft2(img_float)
f_shift = np.fft.fftshift(f_transform)

# --- 3. Add Periodic Noise ---
# Settings for diagonal stripes
u_noise = 32
v_noise = 32
noise_amplitude = (rows * cols) * 0.25  # 0.25 allows background
visibility
```

```
# Peak coordinates
p1 = (center_r + u_noise, center_c + v_noise)
p2 = (center_r - u_noise, center_c - v_noise)

# Add noise spikes
f_shift[p1] += noise_amplitude
f_shift[p2] += noise_amplitude

# --- 4. GENERATE SPECTRUM IMAGE (CRITICAL STEP) ---
# We calculate the Magnitude Spectrum to visualize the frequency
domain.
# Logarithm is used because the dynamic range is very high.
magnitude_spectrum = 20 * np.log(np.abs(f_shift) + 1)

# Normalize the spectrum to 0-255 to save it as a PNG image
spectrum_img = cv2.normalize(magnitude_spectrum, None, 0, 255,
cv2.NORM_MINMAX)
spectrum_img = spectrum_img.astype(np.uint8)

# --- 5. Inverse FFT ---
f_ishift = np.fft.ifftshift(f_shift)
img_noisy = np.fft.ifft2(f_ishift)
img_noisy = np.real(img_noisy)

# Clip and convert to uint8
img_noisy_final = np.clip(img_noisy, 0, 1) * 255
img_noisy_final = img_noisy_final.astype(np.uint8)

# --- 6. Display (Matplotlib) ---
plt.figure(figsize=(15, 6))
# Plot 1: Original
plt.subplot(1, 3, 1)
plt.imshow(img_clean, cmap='gray')
plt.title('Original Image')
plt.axis('off')

# Plot 2: Frequency Spectrum (The requested visualization)
plt.subplot(1, 3, 2)
plt.imshow(spectrum_img, cmap='gray')
plt.title('Frequency Domain (Spectrum)\nLook for white dots!')
```

```
plt.axis('off')
# Draw circles around the noise points to highlight them in the
plot
plt.plot(p1[1], p1[0], 'ro', markersize=5, fillstyle='none',
markeredgewidth=2)
plt.plot(p2[1], p2[0], 'ro', markersize=5, fillstyle='none',
markeredgewidth=2)

# Plot 3: Final Result
plt.subplot(1, 3, 3)
plt.imshow(img_noisy_final, cmap='gray')
plt.title('Final Result (Stripes + Image)')
plt.axis('off')

plt.tight_layout()

print("Displaying plot... (Close the window to finish script)")
plt.show() # This will pop up the window

# --- 7. Save Images ---
# Save the Frequency Spectrum
cv2.imwrite('Frequency_Spectrum.png', spectrum_img)
print("Saved: Frequency_Spectrum.png")

# Save the Final Result
cv2.imwrite('Final_Output_Stripes.png', img_noisy_final)
print("Saved: Final_Output_Stripes.png")
```

کد ۱.py تمرین بخش اول در پوشه ۱ پیوست شده است.

کد مربوط به بخش دوم ۲.py : انتقال بافت موج

```
import matplotlib
try:
    matplotlib.use('TkAgg')
except:
    pass

import matplotlib.pyplot as plt
import numpy as np
import cv2
import os

def main():
    # --- 1. Load Images ---
    print("Loading images...")
    # Source: Water Ripples (Grayscale for intensity)
    img_ripple_src = cv2.imread('3.png', cv2.IMREAD_GRAYSCALE)
    # Target: Green Background (Color)
    img_bg_color = cv2.imread('4.png', cv2.IMREAD_COLOR)

    if img_ripple_src is None or img_bg_color is None:
        print("Error: Could not find '3.png' or '4.png'.")
        return

    # --- 2. Resize Source to Match Target ---
    rows, cols, _ = img_bg_color.shape
    img_ripple_resized = cv2.resize(img_ripple_src, (cols, rows))

    # --- 3. FFT (Frequency Domain Transformation) ---
    # Convert to float
```

```
dft = np.fft.fft2(img_ripple_resized.astype(float))
dft_shift = np.fft.fftshift(dft)

# --- 4. Generate & Save Magnitude Spectrum (Task Requirement) ---
# We use log scale because the center values are huge compared to edges.
magnitude_spectrum = 20 * np.log(np.abs(dft_shift) + 1)

# Normalize to 0-255 for saving as an image
spectrum_img = cv2.normalize(magnitude_spectrum, None, 0, 255, cv2.NORM_MINMAX)
spectrum_img = spectrum_img.astype(np.uint8)

# Save the Spectrum Image immediately
cv2.imwrite('FFT_Spectrum_View.png', spectrum_img)
print(">> Saved: FFT_Spectrum_View.png")

# --- 5. Gaussian High-Pass Filter ---
# Create a Gaussian mask to remove low frequencies (lighting/shadows)
# and keep high frequencies (ripples/edges).
crow, ccol = rows // 2, cols // 2
y, x = np.ogrid[:rows, :cols]

# Calculate distance from center
distance = np.sqrt((x - ccol)**2 + (y - crow)**2)

# Sigma controls how much "blur" or "lighting" is removed.
sigma = 30
# High Pass Formula: 1 - Gaussian_Low_Pass
gaussian_mask = 1 - np.exp(-(distance**2) / (2 * (sigma**2)))

# Apply Filter
fshift_filtered = dft_shift * gaussian_mask
```

```
# --- 6. Inverse FFT (Back to Image) ---
f_ishift = np.fft.ifftshift(fshift_filtered)
img_back = np.fft.ifft2(f_ishift)
img_back = np.real(img_back)

# --- 7. Post-Processing & Blending ---
# 1. Smooth slightly to remove digital noise
img_back_smooth = cv2.GaussianBlur(img_back, (3, 3), 0)

# 2. Increase contrast (Gain) to make ripples visible on green
gain = 2.5
ripple_layer = img_back_smooth * gain

# 3. Add to Green Background
img_bg_float = img_bg_color.astype(np.float32)
final_image = np.zeros_like(img_bg_float)

# Apply ripple offset to all channels (Blue, Green, Red)
for i in range(3):
    final_image[:, :, i] = img_bg_float[:, :, i] + ripple_layer

# Clip values to valid 0-255 range
final_image = np.clip(final_image, 0, 255).astype(np.uint8)

# Save the Final Image
cv2.imwrite('Final_Ripple_Output.png', final_image)
print(">> Saved: Final_Ripple_Output.png")

# --- 8. Visualization (Matplotlib) ---
print("Displaying results... (Close the window to finish)")
```

```
    plt.figure(figsize=(14, 6))

    # Show Spectrum
    plt.subplot(1, 3, 1)
    plt.imshow(spectrum_img, cmap='gray')
    plt.title('Frequency Spectrum\n(Saved as FFT_Spectrum_View.png)')
    plt.axis('off')

    # Show Filtered Ripple Mask
    plt.subplot(1, 3, 2)
    plt.imshow(img_back_smooth, cmap='gray')
    plt.title('Extracted Ripples (High-Pass)')
    plt.axis('off')

    # Show Final Result
    plt.subplot(1, 3, 3)
    plt.imshow(cv2.cvtColor(final_image, cv2.COLOR_BGR2RGB))
    plt.title('Final: Ripples on Green\n(Saved as Final_Ripple_Output.png)')
    plt.axis('off')

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    main()
```

کد ۲.py تمرین بخش دوم در پوشه ۲ پیوست شده است.

با تشکر از نگاه پر مهر شما