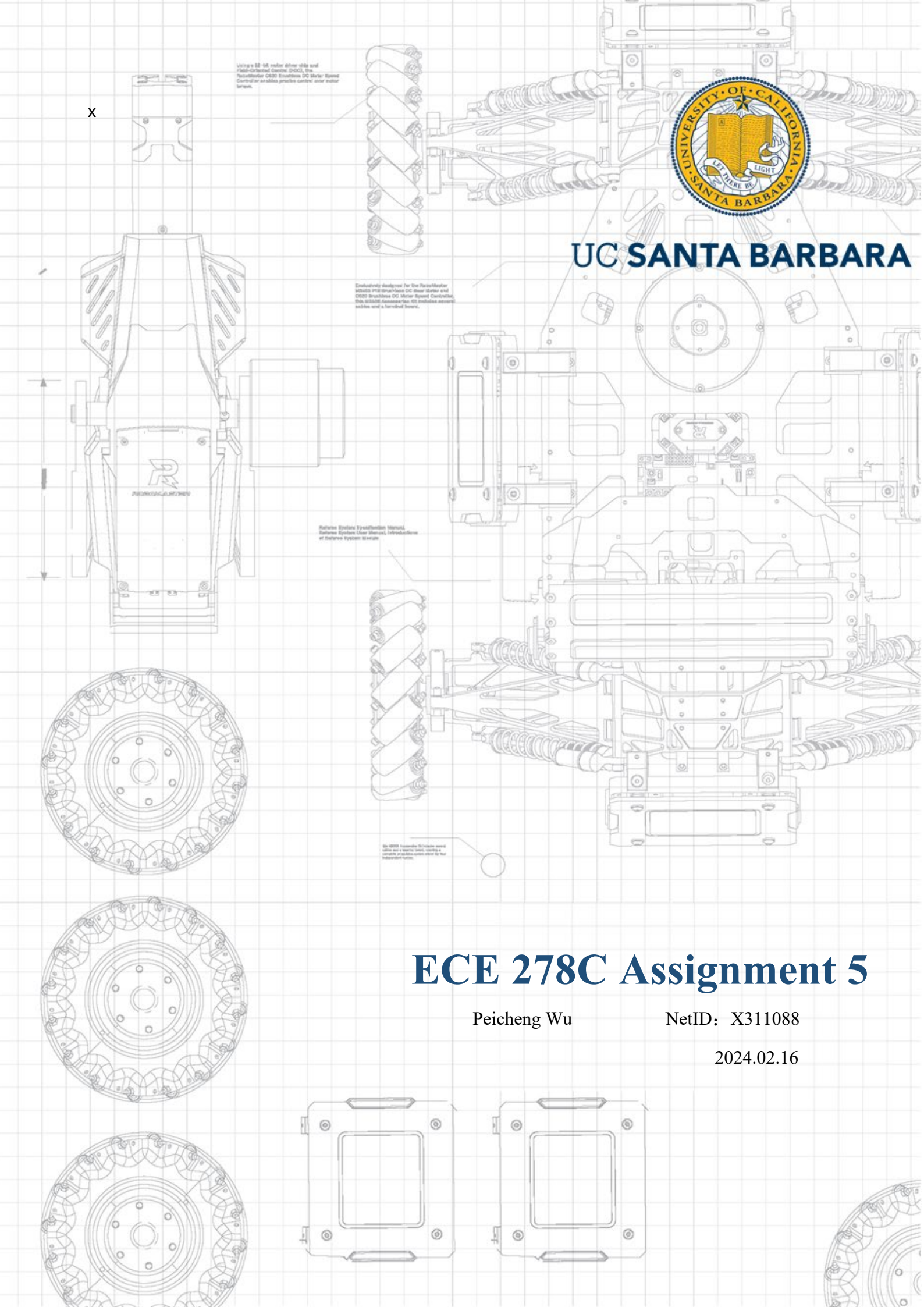
x



UC SANTA BARBARA

ECE 278C Assignment 5

Peicheng Wu NetID：X311088

2024.02.16

Catalog

1. Platform: VScode

Visual Studio Code, often referred to as VSCode, is a free and open-source code editor developed by Microsoft. It's renowned for its lightweight design, speed, and robust capabilities. Available for Windows, macOS, and Linux, VSCode offers built-in Git integration, an integrated terminal, and a debugger. Its power lies in its extensibility, with a vast marketplace of extensions that add support for various programming languages, debuggers, and tools. The editor's appearance and behavior are highly customizable, and its Intellisense feature provides smart code completions. Thanks to its open-source nature, it has a vibrant community that continually contributes to its development and enhancement.

2. Question

Consider a single centered point source,

	scatters	*scatter locations*
1	(x_1, y_1)	$(0, +15\lambda)$
2	(x_2, y_2)	$(-12\lambda, -9\lambda)$
3	(x_3, y_3)	$(+12\lambda, -9\lambda)$

The horizontal receiver aperture is organized in the form of a centered linear receiver array with a span of 60λ (from x= -30λ to x = +30λ). This horizontal linear receiver array is placed at

$$y = y_0 + 60\,\lambda$$

With quarter-wavelength spacing (λ/4) spacing, there are 241 complex wavefield data samples in total over the 60λ-long linear aperture.

we move up to the **multi-frequency (wideband) operating mode**, we collect wavefield samples over a wide range of spectrum corresponding to 64 different wavelengths, in the form:

$$\lambda_n = 64\lambda_0/(n+32) \qquad n = 1,2,\ldots 64$$

- By repeating the range estimation procedure, produce the 241 range profiles over the source region.
- Superimpose the 241 sub-range profiles sequentially and observe the convergence to images
- Produce the spectra of the complex range profiles $\hat{S}n(x, y)$, n = 1, 2, …. 241
- Compile a 241-frame video of the range-profile sequence $\hat{S}n(x, y)$.
- Compile a 241-frame video of the spectra of the range-profile sequence.

3. Solutions

In the field we have equation:

$$g(x,y,z) = s(x,y,z) * h(s,y,z)$$

And now we just assume source as the impulse function:

$$s(x,y,z) = \delta(x,y,z)$$

Thus, for each receiver, we can write the received signal

$$g(x,y,z) = \frac{1}{j\lambda r} exp\left(\frac{j2\pi r}{\lambda}\right)$$

According the backward propagation:

$$\dot{s}(x,y,z) = g(x,y,z) * h^*(x,y,z)$$

$$= \iiint \left[\frac{1}{j\lambda r} exp\left(\frac{j2\pi r}{\lambda}\right)\right]\left[\frac{-1}{j\lambda r'} exp\left(\frac{-j2\pi r'}{\lambda}\right)\right] dx' dy' dz'$$

$$= \iiint \frac{1}{\lambda^2 r r'} exp\left(\frac{j2\pi\left(r - r'\right)}{\lambda}\right) dx' dy' dz'$$

Where r means the distance from source scatter to the receiver and r' means the distance from source region point to the receiver.

So we can just write the equation of the reconstructed image:

$$\dot{s}(x,y,z) = \iiint \frac{1}{\lambda^2 r r'} exp\left(\frac{j2\pi\left(r - r'\right)}{\lambda}\right) dx' dy' dz'$$

Then we can use this equation calculate all the points in the source region.

Also, we can write the **full phase only term** equation:

$$\dot{s_k}(x,y,z) = \iiint exp\left(\frac{j2\pi r}{\lambda}\right) \times exp\left(\frac{j2\pi r'}{\lambda}\right) dx'dy'dz'$$

So for the complex wave, we have:

$$\dot{s_n}(x,y,z) = \sum_{k=1}^{n} \dot{s_k}(x,y,z) \qquad n = 1,2,\ldots 241$$

4. Results

4.1 Reconstructed images

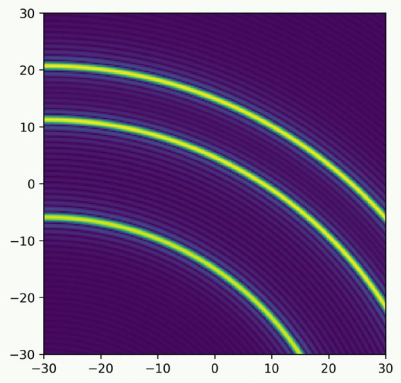


Fig.1 Reconstructed image in time domain

The image has been constructed by linear receivers. It can show that three source scatters locations on three different radius circles. When the receiver is on the center. We can see two circles because there are two scatters have same y.

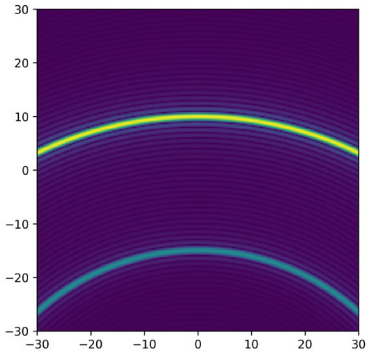


Fig.2 Reconstructed image by center receiver

Also, we can stack these images:

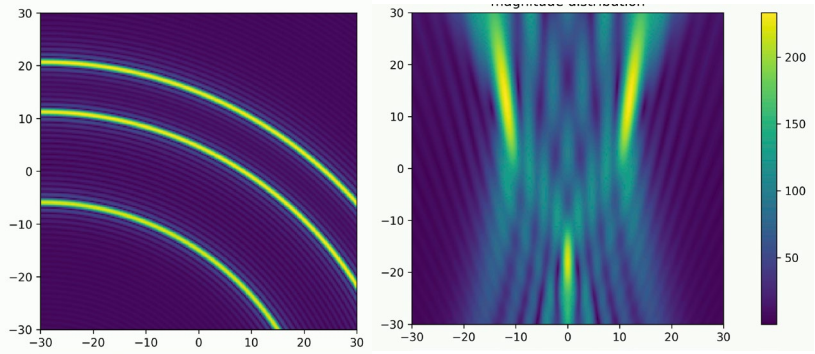


Fig.3 Reconstructed image stacked by different receivers

Fig.4 Reconstructed image stacked by different frequency waves

In this part, we can see receivers and frequencies both can enhance the accuracy of reconstructed imags. But they are different, by adding receivers can pinpoint the location of scatter on the ring and by adding different frequency waves can pinpoint the depth information.

4.2 FFT

For this part we just see how frequency domain changed by adding receivers:

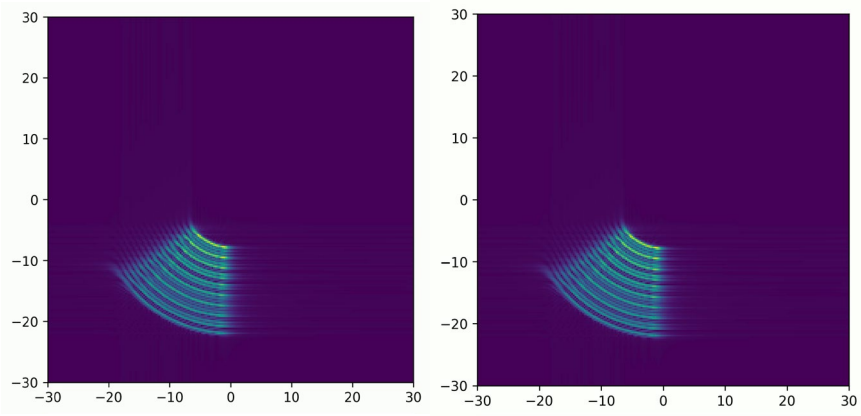


Fig.5 Frequency domain of reconstruction image

Fig.6 Stacked frequency domain of reconstruction image

We can see the fig.5, each receiver can get a wide range of directions of wavs. However, some of them will cancel each other. Finally, it will show the angle in the assignment 2 and 3 which depends on the width of receivers and the distance from receivers to center of source region.

5. Appendix

```python
import numpy as np
import cupy as cp
import matplotlib.pyplot as plt
from scipy.fft import fft2, fftshift
import argparse
from tqdm import tqdm

def generate_receivers_locations(lambda_0, y_offset, span):
    delta_x = lambda_0 / 4
    num_points = int(span / delta_x) + 1
    x_positions = np.linspace(-span / 2, span / 2, num_points)
    scatter_locations = [(x, y_offset) for x in x_positions]
    # print(scatter_locations)
    return scatter_locations

def generate_source_region(lambda_0, X_length, Y_length):
    delta_x = lambda_0 / 4
    num_points_x = int(X_length / delta_x) + 1
    num_points_y = int(Y_length / delta_x) + 1
```

```
    x_positions = np.linspace(-X_length / 2, X_length / 2, num_points_x)
    y_positions = np.linspace(-Y_length / 2, Y_length / 2, num_points_y)
    source_locations = [(x, y) for y in y_positions for x in x_positions]
    print(source_locations)
    return source_locations

def calculate(source_scatter):

    lambda_0 = 1
    X_length = 60 * lambda_0
    Y_length = 60 * lambda_0
    y_offset = 60 * lambda_0
    span = 60 * lambda_0
    source_locations = generate_source_region(lambda_0, X_length, Y_length)
    receivers_locations = generate_receivers_locations(lambda_0, y_offset, span)

    num_points_x = int(X_length / (lambda_0 / 4)) + 1
    num_points_y = int(Y_length / (lambda_0 / 4)) + 1
    h_whole = np.zeros((num_points_y, num_points_x), dtype=complex)
    each_scatter = np.zeros((num_points_y, num_points_x), dtype=complex)
    temp = np.zeros((num_points_y, num_points_x), dtype=complex)
    complex_wave = []
    num = 0
    for x_g, y_g in tqdm((receivers_locations)):
        for n in range(64):
            lambda_n = 64*lambda_0/(32+n)
            for x_cc, y_cc in source_scatter:
                for idx, (x_s, y_s) in enumerate(source_locations):
                    y_idx = idx // num_points_x
                    x_idx = idx % num_points_x
                    r_g = np.sqrt((x_g - x_cc)**2 + (y_g - y_cc)**2)
                    r_s = np.sqrt((x_s - x_g)**2 + (y_s - y_g)**2)
                    h = np.exp(1j * 2 * np.pi * (r_g) / lambda_n) * np.exp(-1j * 2 * np.pi
* (r_s) / lambda_n)
                    h_whole[y_idx, x_idx] = h
                each_scatter += h_whole
                h_whole = np.zeros((num_points_y, num_points_x), dtype=complex)
            temp += each_scatter
            each_scatter = np.zeros((num_points_y, num_points_x), dtype=complex)
        np.save(f'F:/desktop/results/complex_wave_{num}.npy', temp)
        num += 1
        complex_wave.append(temp)
        temp = np.zeros((num_points_y, num_points_x), dtype=complex)
    return complex_wave


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='Different question in the
assignment')
    args = parser.parse_args()
```

```
    source_scatter = [(0, 15), (-12, -9), (12, -9)]

    complex_wave = calculate(source_scatter)
    np.save('F:/desktop/results/complex_wave.npy', complex_wave)
```

```
import numpy as np
import cupy as cp
import matplotlib.pyplot as plt
from scipy.fft import fft2, fftshift
import argparse
from tqdm import tqdm

file_path = 'F:/desktop/results/complex_wave.npy'


parser = argparse.ArgumentParser(description="choose time domain or frequency domain")
group = parser.add_mutually_exclusive_group()  # 创建互斥组
group.add_argument('--time_domain', action='store_true', help='show the magnitude')
group.add_argument('--fre_domain', action='store_true', help='show the FFT')
parser.add_argument('--stack', action='store_true', help='whether stack the output')
args = parser.parse_args()


complex_wave = np.load(file_path)
temp = np.zeros((241, 241), dtype=complex)
stack = np.zeros((241, 241), dtype=complex)
i = 0

for cw in tqdm(complex_wave):

    if args.time_domain:
        temp = cw
        if args.stack:
            stack += temp
            temp = stack
    elif args.fre_domain:
        fft = fftshift(fft2(cw))
        temp = fft
        if args.stack:
            stack += temp
            temp = stack

    plt.figure(figsize=(12, 5))
    plt.imshow(abs(temp), extent=(-30, 30, -30, 30))
    plt.title('')
    # plt.colorbar()
```

```
    plt.savefig(f'F:\\Peicheng Wu\\HUST\\3+1\\24Winter\\ECE
278C\\assignment5\\results\\fre_domain_stack\\complex_wave_{i}.png', dpi=300,
bbox_inches='tight')
    plt.close()
    i += 1
```

You can also find the code on the github: UCSB-ECE-278C/assignment3 at main · percyance/UCSB-ECE-278C (github.com)

When you want to run: please type python assignment2.py --Q1 --phase_only

Q1 means one target, Q2 means three targets

phase_only and full_phase_only is choose different term.