ECE278C Imaging Systems

Lab 7: Geolocation

March 20, 2021

Student: Ivan Arevalo
Perm Number: 5613567
Email: ifa@ucsb.com

Department of Electrical and Computer Engineering, UCSB

0 Introduction

The main objective of this assignment is to combine range estimation and bearing-angle estimation for the geolocation of a single target in the active mode. We'll simulate a short baseline sensing device which consists of one centered transmitter and two receivers, one at each side. The distance from the transmitter to a receiver is $D = M\lambda_0$, where M is an integer. (Thus, the entire length of the device is 2D.) A sequence of 128 FMCW waveforms are transmitted, corresponding to the operating wavelengths,

$$\lambda_n = 128\lambda_0/(n + 128) \qquad n = 0, 1, 2, \ldots 127$$

The distance from the transmitter to the target is r_0, and the distance from the target to the first receiver is r_1. The 128-point data sequence collected at the location of the first receiver is

$$g_1(n) = Aexp(j2\pi(r_0 + r_1)/\lambda_n)$$

and at the second receiver, the data sequence is in the form

$$g_2(n) = Aexp(j2\pi(r_0 + r_2)/\lambda_n)$$

Given these two 128-point sequences, we will design and implement a simple technique to estimate the location of the target.

1 Intersecting Range Profiles

1.1 Intuition

The first and most simple method to estimate the geolocation of a single target is to compute the range profile of both receivers around the target area and observe where the range profiles intersect. By normalizing each range profile to unit magnitude and adding them together, we get our composite range profile of both receivers as shown in figure 1 and 2. We can then pick the location with highest magnitude in the positive y direction as our estimated target location

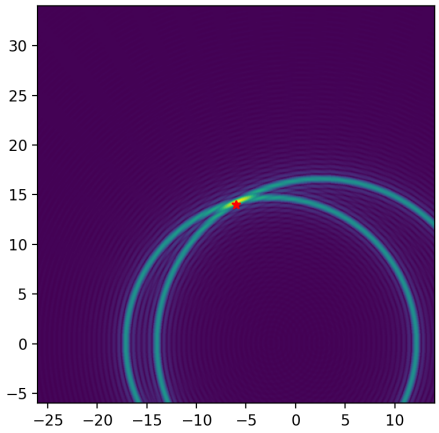

Figure 1: Intersecting Range Profiles for target at (-6,14) marked as red star

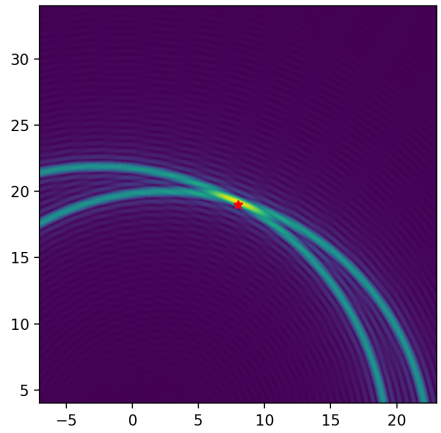

Figure 2: Intersecting Range Profiles for target at (8,19) marked as red star

1.2 Algorithm

The algorithm consist in estimating the range from each receiver's range profile and compute where the 2 corresponding circles with their respective radius and origin would intersect.

We'll generate the range profile of each receiver by taking the 2048-point FFT of the 128-point wave-field sample sequence $g_k(n)$ of each receiver. Next we plot the range profile where the x-axis is range vector from 0 to 128 and the y-axis is the FFT of $g_k(n)$ we computed previously. Given we are simulating an active mode system, we need to account for the round trip wave propagation and scale the range vector by $\frac{1}{2}$. Finally, we can get our estimated range by observing where the peak of our range profile occurs. For the same target locations as before, figure 3-6 show the corresponding range profiles and their peak locations.

Given our distance between our transmitter and receivers $D = 5$, the two receivers are located at (-5, 0) and (5, 0), the equations of each circle would be

$$(x+5)^2 + y^2 = (r_{left})^2$$
$$(x-5)^2 + y^2 = (r_{right})^2$$

subtracting bottom from bottom,

$$(x+5)^2 - (x-5)^2 = (r_{left})^2 - (r_{right})^2$$

simplifying the expression, we get the following linear function,

$$(x^2 + 10x + 25) - (x^2 - 10x + 25) = (r_{left})^2 - (r_{right})^2$$

$$20x = (r_{left})^2 - (r_{right})^2$$

$$x = \frac{(r_{left})^2 - (r_{right})^2}{20}$$

we then plug in x into either circle function to solve for y, and take the solution with a positive y.

$$(\frac{(r_{left})^2 - (r_{right})^2}{20} - 5)^2 + y^2 = (r_{right})^2$$

$$y^2 = (r_{right})^2 - (\frac{(r_{left})^2 - (r_{right})^2}{20} - 5)^2$$

$$y = \sqrt{(r_{right})^2 - (\frac{(r_{left})^2 - (r_{right})^2}{20} - 5)^2}$$

Hence our estimate target location would be,

$$\text{Target}(x, y) = \left(\frac{(r_{left})^2 - (r_{right})^2}{20}, \sqrt{(r_{right})^2 - (\frac{(r_{left})^2 - (r_{right})^2}{20} - 5)^2} \right)$$

1.3 Results

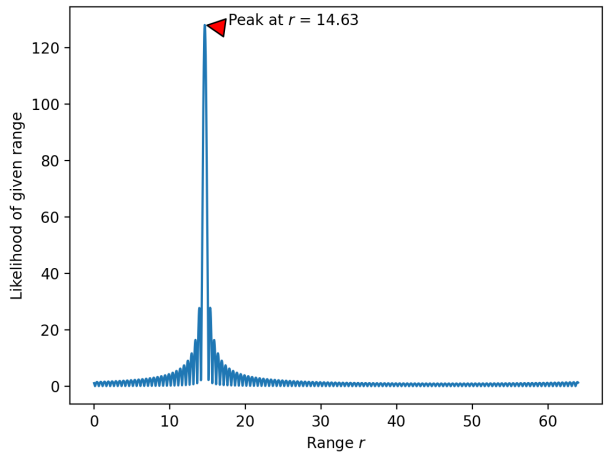

Figure 3: Range Profiles for left receiver target at (-6,14)

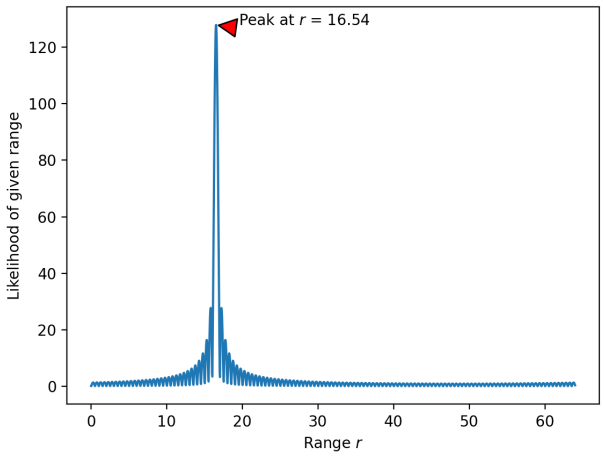

Figure 4: Range Profiles for right receiver target at (-6,14)

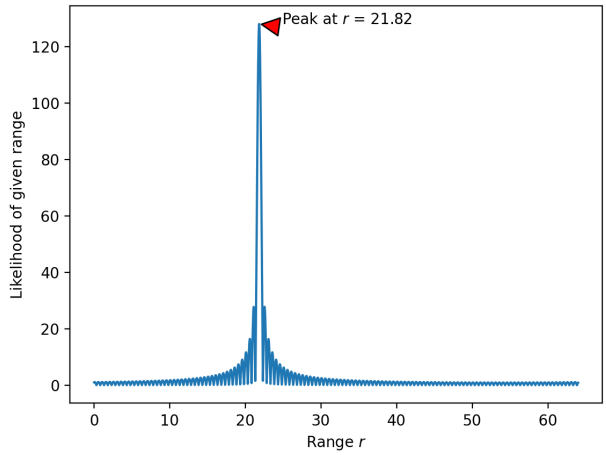

Figure 5: Range Profiles for left receiver target at (8,19)

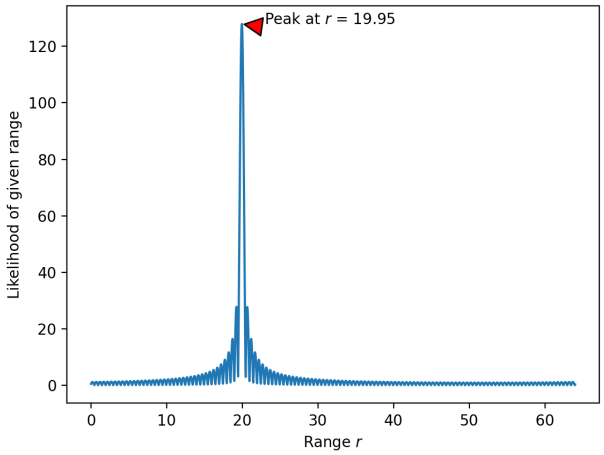

Figure 6: Range Profiles for right receiver target at (8,19)

From figures 3-6, we have that the left and right receiver ranges for target location $(-6, 14)$ are 14.63 and 16.34 respectively. Likewise, the left and right receiver ranges for target location $(8, 19)$ are 21.82 and 19.95 respectively.

This corresponds to following target location estimations

$$\text{Target at } (-6, 14) = \left(\frac{(16.34)^2 - (14.63)^2}{20}, \sqrt{(14.63)^2 - (\frac{(16.34)^2 - (14.63)^2}{20} - 5)^2} \right) = (-3.00, 14.49)$$

$$\text{Target at } (8,19) = \left(\frac{(19.95)^2 - (21.82)^2}{20}, \sqrt{(21.82)^2 - (\frac{(19.95)^2 - (21.82)^2}{20} - 5)^2} \right) = (3.91, 19.91)$$

These estimations are descent for the y range but not for the x range. After various experiments, I realized that this estimation scheme works best when the distance between the target location and receiver is at least 1 order of magnitude larger than the distance between receivers. Hence for the same target locations, I ran the experiment setting the distance between receivers to 2. These are the results,

$$\text{Target at } (-6,14) = \left(\frac{(16.34)^2 - (14.63)^2}{2}, \sqrt{(14.63)^2 - (\frac{(16.34)^2 - (14.63)^2}{2} - 1)^2} \right) = (5.72, 13.90)$$

$$\text{Target at } (8,19) = \left(\frac{(20.82)^2 - (20.44)^2}{2}, \sqrt{(20.44)^2 - (\frac{(20.82)^2 - (20.44)^2}{2} - 1)^2} \right) = (7.75, 19.62)$$

2 Range and Bearing Angle Estimation

Another approach to estimate the target location is to divide the problem into a range estimation problem and a bearing angle estimation problem from the point of view of the receiver.

2.1 Range Estimation

Since we learned that range estimation works best with a small distance between receiver compared to the distance between the receiver and target, we now set our distance between transmitter and receiver to $D = 1$. We repeat our previous step of gathering the sampled wave-field at each receiver $g_1(n), g_2(n)$. The main difference now, is that we want to estimate the range from the point of view of the transmitter which is at (0,0). To achieve this, we multiply the 2 sampled wave-fields at the receivers to get

$$g_r(n) = g_1(n)g_2(n) = A^2 exp(j2\pi(2r_0 + r_1 + r_2)/\lambda_n)$$

We now follow the same procedure as in section 1.2 to generate the range profile. This entails taking the 2048-FFT of g_r and plotting against the range vector. Notice that we are now estimating the range from the transmitter point of view as $(2r_0 + r_1 + r_2)/4$ so we will scale our range vector by $\frac{1}{4}$ instead. Finally, we can get our estimated range by observing where the peak of our range profile occurs. For the same target locations as before, figure 7 and 8 show the corresponding range profiles and their peak locations.

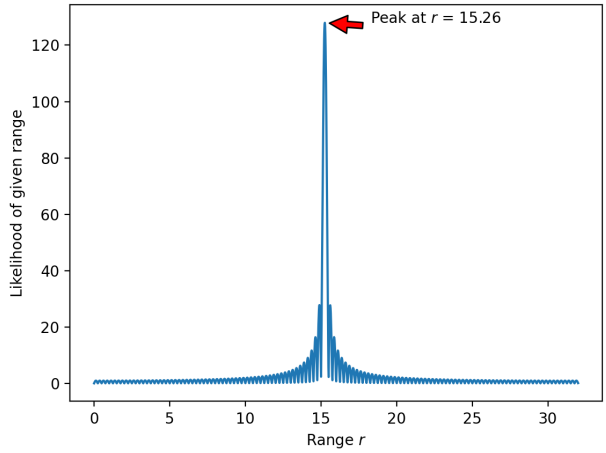

Figure 7: Range Profiles from transmitter with target at (-6,14)

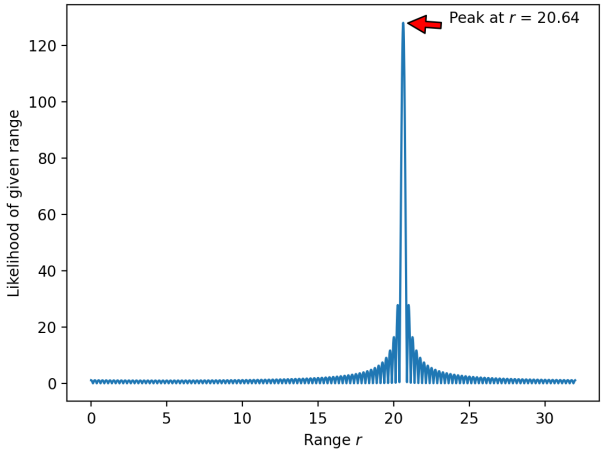

Figure 8: Range Profiles from transmitter with target at (8,19)

Observing figure 7 and 8, we can estimate that the range from transmitter point of view for target at (-6,14) and (8,19) are 15.26 and 20.64 respectively.

Furthermore, we can visualize these results by reconstructing the 2D range profile around the target areas as shown in figures 9 and 10.

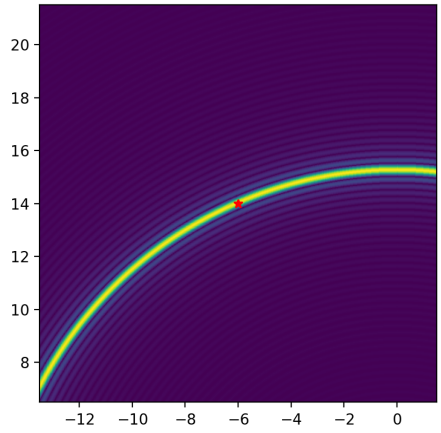

Figure 9: 2D Range Profile from transmitter with target at (-6,14) shown as red star

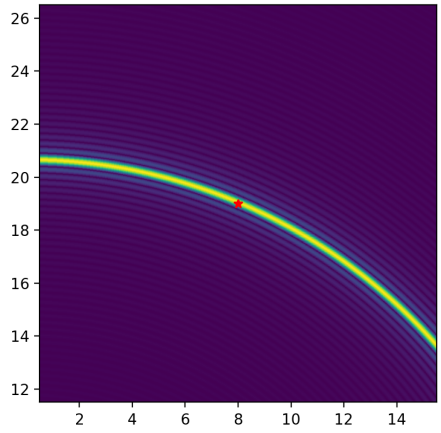

Figure 10: 2D Range Profile from transmitter with target at (8,19) shown as red star

We can see that our target locations lie within the circle generated by our transmitter range profiles. We now need a method of estimating the bearing angle of our target to pinpoint the target location along the range profile circle.

2.2 Bearing Angle Estimation

To estimate the bearing angle of our target we can use the difference between the distances from our receivers to the target $(r_2 - r_1)$. This will give us two possible solutions since there are two locations along a circle for which the difference between the distance from location on the circle to two points inside the circle are the same.

We can use the same method derived for range estimation by generating

$$g_a(n) = g_1(n)g_2^*(n) = A^2 exp(j2\pi(r_2 - r_1)/\lambda_n)$$

and plotting the shifted FFT of $g_a(n)$ with respect to a bearing angle vector. In order to get the correct bearing angle, we need to scale the bearing angle vector by $\frac{1}{2D}$, where D is the distance between the transmitter and receivers. We then need to window our observation for bearing-angle-vector values between $(-1, 1)$ and take the sin^{-1}(bearing angle vector) to get our bearing angle vector in radians which we can then map to degrees. Figure 11 and 12 show the bearing angle profile for our targets at (-6, 14) and (8,19) respectively.

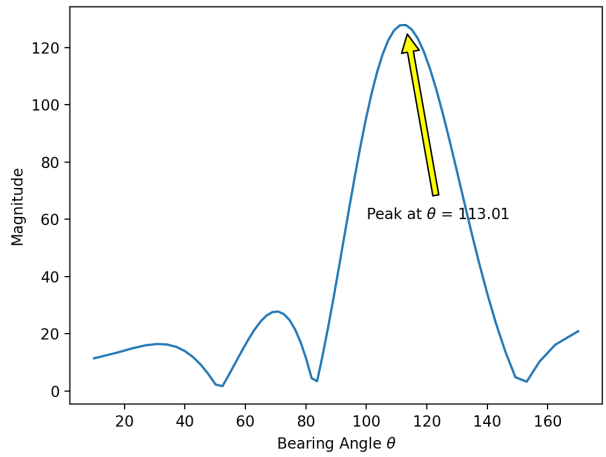

Figure 11: Bearing Angle Profile from transmitter with target at (-6,14)

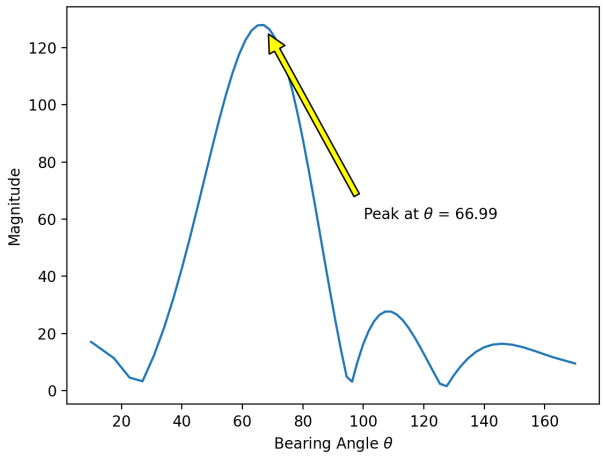

Figure 12: Bearing Angle Profile from transmitter with target at (8,19)

Figure 11 and 12 show that the peaks occur at $\theta = 113.01°$ and $\theta = 66.99°$ for targets at (-6, 14) and (8,19) respectively.

2.3 Geolocation

We can now combine our estimated range and bearing angle to pinpoint the location of our targets. Figure 13 and 14 are polar maps that show the range estimation in red and bearing angle estimation in blue for each of our targets.

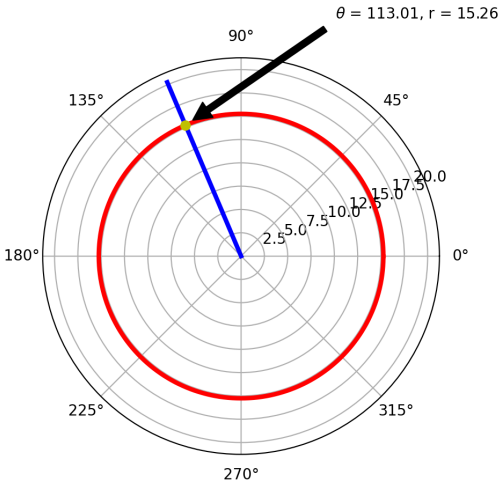

Figure 13: Polar plot of range and bearing angle estimation for target at (-6,14)

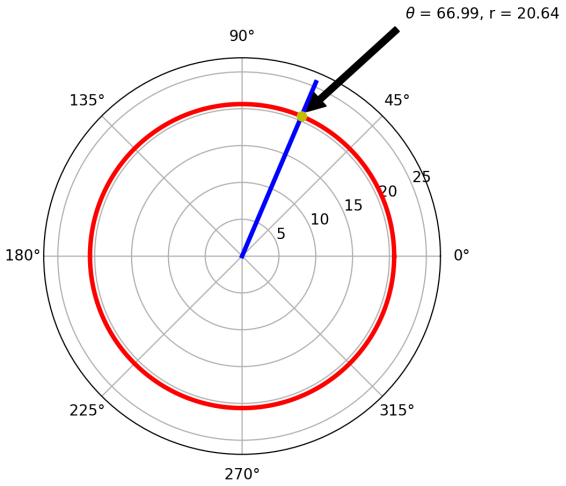

Figure 14: Polar plot of range and bearing angle estimation for target at (8,19)

Finally, we convert our polar coordinates to Cartesian coordinates by,

$$(x, y) = \big(r\cos(\theta), r\sin(\theta)\big)$$

The following table summarizes our results

Target Location	Estimated Location	$(\mathrm{x}_{\%error}, y_{\%error})$
$(-6, 14)$	$(-5.97, 14.05)$	$(-0.005, 0.003)$
$(8, 19)$	$(8.07, 19.00)$	$(0.00875, 0.00)$

3 Conclusion

The results from our range and bearing angle estimation method are very satisfactory with the caveat that the accuracy decreases as we choose points closer to the transmitter since we approximate that $r_0 = (2r_0 + r_1 + r_2)/4$. Furthermore, we have shown that this method has greater accuracy under the same simulation conditions that the intersection of profile ranges method outlined in section 1.

4 Code

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


class simulation(): # Single centerd transmitter, twin receivers
    def __init__(self, target_loc, distance_rx_t, num_freq, ref_wavelength=1):
        self.target_loc = target_loc
        self.distance_rx_tx = distance_rx_t
        self.num_freq = num_freq
        self.ref_wavelength = ref_wavelength
```

```
def calc_distance(self, coord_1, coord_2):
    return np.sqrt( (coord_1[0] - coord_2[0])**2 + (coord_1[1] - coord_2[1])**2)

def sim_data_collect(self, receiver_location):
    receiver_data = np.zeros(self.num_freq, dtype=np.complex)
    for i in range(self.num_freq):
        lam = self.num_freq * self.ref_wavelength / (i + self.num_freq)
        r_tx = self.calc_distance((0,0), self.target_loc)
        r_rx = self.calc_distance(receiver_location, self.target_loc)
        receiver_data[i] = np.exp(1j * 2 * np.pi * (r_tx + r_rx) / lam)
    return receiver_data

def compute_range_profile(self, rx_data, rx_loc, target_window, scale=2,
    plot_range=False, plot_image=False):

    mult = 16 # Multiplier for higher resolution
    fft_bin = np.fft.fft(rx_data, self.num_freq * mult)
    range_v = np.linspace(0, self.num_freq, len(fft_bin))/scale
    self.range_v = abs(fft_bin/np.max(abs(fft_bin)))*range_v
    range_peak_idx = np.where(abs(fft_bin) == np.max(abs(fft_bin)))[0][0]
    range_peak = range_v.ravel()[range_peak_idx]


    if plot_range:
        plt.figure()
        plt.plot(range_v, abs(fft_bin))
        plt.annotate(fr"Peak at $r$ = {round(range_peak,2)}", xy=(range_peak,
            abs(fft_bin).ravel()[range_peak_idx]),
                    xytext=(range_peak+3, abs(fft_bin).ravel()[range_peak_idx]),
                        arrowprops=dict(facecolor='red',
                                                shrink=0.1), xycoords="data", )
        plt.xlabel(r"Range $r$")
        plt.ylabel("Likelihood of given range")
        print(f"Estimated Range from {rx_loc}: {range_peak}")
        print(f"Actual Range from {rx_loc}:
            {np.linalg.norm((self.target_loc[0]-rx_loc[0])+(self.target_loc[1]-rx_loc[1])*1j)}\n")

    if plot_image:
        x_ax = np.arange(target_window[0,0], target_window[0,1], 0.25 / 2)
        y_ax = np.arange(target_window[1,0], target_window[1,1], 0.25 / 2)
        X, Y = np.meshgrid(x_ax, y_ax)
        Y = np.rot90(Y, 2)
        image = np.zeros(X.shape).ravel()

        for i, xy in enumerate(zip(X.ravel(), Y.ravel())):
            r = np.linalg.norm([xy[0] - rx_loc[0], xy[1] - rx_loc[1]]) * mult
            pxl_val = 0.5 * (fft_bin[int(np.floor(r))] + fft_bin[int(np.ceil(r))])
            image[i] = abs(pxl_val)

        image = image.reshape(X.shape)

        return image / np.max(abs(image)), range_peak
    else:
```

```
            return range_peak

    def bearing_angle(self, rx_data):
        mult = 16
        fft_bin = np.fft.fftshift(np.fft.fft(rx_data, self.num_freq * mult))
        angle_v = np.linspace(-128/2, 128/2, len(fft_bin))/(2*self.distance_rx_tx)

        phase_peak_idx = np.where(abs(fft_bin) == np.max(abs(fft_bin)))[0][0]
        phase = angle_v.ravel()[phase_peak_idx]
        angle = 90 - np.degrees(np.arcsin(phase))

        plt.figure()
        idx_min = np.argmin(abs(angle_v+1))
        idx_max = np.argmin(abs(angle_v-1))
        plt.plot(90 - np.degrees(np.arcsin(angle_v[idx_min:idx_max+1])),
            abs(fft_bin[idx_min:idx_max+1]))
        plt.annotate(fr"Peak at $\theta$ = {round(angle,2)}", xy=(angle,
            abs(fft_bin).ravel()[phase_peak_idx]),
                     xytext=(100, 60), arrowprops=dict(facecolor='yellow',
                                                       shrink=0.05), xycoords="data", )
        plt.xlabel(r"Bearing Angle $\theta$")
        plt.ylabel("Magnitude")

        return angle

def main():
    rxtx_d = 1 # Distance between transmitter and receivers
    target_loc = (-6,14)
    win_rad = 30
    num_freq = 128

    target_window = np.array([[2*target_loc[0]-win_rad, 2*target_loc[0]+win_rad],
        [2*target_loc[1]-win_rad, 2*target_loc[1]+win_rad]])

    sim = simulation(target_loc, rxtx_d, num_freq)
    g1 = sim.sim_data_collect((-rxtx_d,0))
    g2 = sim.sim_data_collect((rxtx_d,0))

    im1, _ = sim.compute_range_profile(g1, (-rxtx_d, 0), target_window, plot_image=True,
        plot_range=True)
    im2, _ = sim.compute_range_profile(g2, (rxtx_d, 0), target_window, plot_image=True,
        plot_range=True)

    # Intersection of range profile from both receivers
    plt.figure()
    max_point = np.argmax(abs(im1 + im2))
    max_mat = np.zeros(im1.shape)
    max_mat.ravel()[max_point] = 0 # set to 10 to highlight predicted intersection.
    plt.imshow(abs(im1+im2+max_mat), extent=[target_window[0,0]/2, target_window[0,1]/2,
        target_window[1,0]/2, target_window[1,1]/2])
    plt.plot(target_loc[0], target_loc[1], 'r*')

    # Range profile of g1(n)g2(n) from transmitter POV
    target_window = np.array([[4 * target_loc[0] - win_rad, 4 * target_loc[0] + win_rad],
```

```
                            [4 * target_loc[1] - win_rad, 4 * target_loc[1] + win_rad]])
    im_comp, range_comp= sim.compute_range_profile(g1 * g2, (0, 0), target_window,
        scale=4, plot_image=True, plot_range=True)
    plt.figure()
    plt.imshow(abs(im_comp), extent=[target_window[0,0]/4, target_window[0,1]/4,
        target_window[1,0]/4, target_window[1,1]/4])
    plt.plot(target_loc[0], target_loc[1], 'r*')
    print(f"Estimated Range: {range_comp}")
    print(f"Actual Range: {np.linalg.norm(target_loc[0]+target_loc[1]*1j)}")

    # Bearing angle estimation from g1(n)g2*(n) from transmitter POV
    angle = sim.bearing_angle(g1 * np.conjugate(g2))
    print(f"Estimated Angle: {angle}")
    print(f"Actual Angle: {np.degrees(np.arctan2(target_loc[1],target_loc[0]))}")

    # Create polar plot
    plt.figure()
    theta = np.arange(0, 2 * np.pi, 0.01)
    plt.polar(theta, range_comp*np.ones(theta.shape), 'r', lw=3)
    plt.polar([np.deg2rad(angle),np.deg2rad(angle)], [0,range_comp+5], 'b', lw=3)
    plt.plot(np.deg2rad(angle), range_comp, 'yo')
    plt.annotate(fr"$\theta$ = {round(angle, 2)}, r = {round(range_comp, 2)}",
                xy=(np.deg2rad(angle), range_comp), # theta, radius
                xytext=(0.65, 0.95), # fraction, fraction
                textcoords='figure fraction',
                arrowprops=dict(facecolor='black', shrink=0.05),
                horizontalalignment='left',
                verticalalignment='bottom',
                )

    plt.show()

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