Alignment

Why are this col of interest

    #################################

    # Read Vehicle and Resim data

    #################################

    cols\_of\_interest = [

        "scan\_index",

        "vacs\_boresight\_az\_nominal",

        "vacs\_boresight\_az\_kf\_internal",

        "vacs\_boresight\_az\_estimated",

        "vacs\_boresight\_el\_nominal",

        "vacs\_boresight\_el\_kf\_internal",

        "vacs\_boresight\_el\_estimated",

    ]

What is this line doing?

    kpis\_align = {'result1':

                   {'numerator': number\_of\_scans\_with\_matching\_az\_misalign,

                    'denominator': number\_of\_scans\_in\_both\_real\_and\_sim,

                    'value': round((number\_of\_scans\_with\_matching\_az\_misalign / number\_of\_scans\_in\_both\_real\_and\_sim) \* 100, 2)},

               'result2':

                   {'numerator': number\_of\_scans\_with\_matching\_el\_misalign,

                    'denominator': number\_of\_scans\_in\_both\_real\_and\_sim,

                    'value': round((number\_of\_scans\_with\_matching\_el\_misalign / number\_of\_scans\_in\_both\_real\_and\_sim) \* 100, 2)},

               }

Why are we doing it one byb one manually ? # Manually add traces for line plots

    fig\_line.add\_trace(

        go.Scatter(

            x=scan\_index\_list,

            y=az\_misalign\_est\_real\_list,

            mode="lines",

            name="Veh Az Est",

            line=dict(color="blue"),

        ),

        row=1,

        col=1,

    )

    fig\_line.add\_trace(

        go.Scatter(

            x=scan\_index\_list,

            y=az\_misalign\_ref\_real\_list,

            mode="lines",

            name="Veh Az Ref",

            line=dict(color="blue"),

            opacity=0.2,

        ),

        row=1,

        col=1,

    )

    fig\_line.add\_trace(

        go.Scatter(

            x=scan\_index\_list,

            y=az\_misalign\_est\_sim\_list,

            mode="lines",

            name="Sim Az Estd",

            line=dict(color="red"),

        ),

        row=1,

        col=1,

    )

    fig\_line.add\_trace(

        go.Scatter(

            x=scan\_index\_list,

            y=az\_misalign\_ref\_sim\_list,

            mode="lines",

            name="Sim Az Ref",

            line=dict(color="red"),

            opacity=0.2,

        ),

        row=1,

        col=1,

    )

    fig\_line.update\_yaxes(title\_text="Azimuth misalignment", row=1, col=1)

    fig\_line.update\_xaxes(title\_text="Scan Index", row=1, col=1)

why are we taking this csv ? can we use other csv stream

 '\_UDP\_GEN7\_DYNAMIC\_ALIGNMENT\_STREAM.csv'

Tracker

Why this col of interest : and also why are the

cols\_of\_interest = cols\_of\_interest + [item for i in range(max\_number\_of\_data) for item in (f"trkID\_{i}", f"vcs\_xposn\_{i}", f"vcs\_yposn\_{i}", f"vcs\_xvel\_{i}", f"vcs\_yvel\_{i}", f"vcs\_heading\_{i}", f"len1\_{i}", f"len2\_{i}", f"wid1\_{i}", f"wid2\_{i}", f"f\_moving\_{i}")]

after finding difference bet sim and real and seeing if it passes threshold and then incrementing match count

# Apply the matching function row-wise

    merged\_df[['mov\_trk\_match\_count', 'all\_trk\_match\_count']] = merged\_df.apply(match\_tracks, axis=1, result\_type="expand")

    merged\_df["same\_num\_of\_valid\_trks"] = (merged\_df["num\_valid\_trks\_veh"] != 0) & (merged\_df["num\_valid\_trks\_veh"] == merged\_df["num\_valid\_trks\_sim"])

    merged\_df["same\_num\_of\_valid\_mov\_trks"] = (merged\_df["num\_valid\_mov\_trks\_veh"] != 0) & (merged\_df["num\_valid\_mov\_trks\_veh"] == merged\_df["num\_valid\_mov\_trks\_sim"])

    merged\_df['matching\_pct\_all\_trks'] = merged\_df['all\_trk\_match\_count']/merged\_df['num\_valid\_trks\_veh']

    merged\_df['matching\_pct\_mov\_trks'] = merged\_df['mov\_trk\_match\_count']/merged\_df['num\_valid\_mov\_trks\_veh']

    scan\_index\_list = merged\_df['scan\_index'].tolist()

    accuracy\_all\_trks\_list = merged\_df['matching\_pct\_all\_trks'].tolist()

    accuracy\_mov\_trks\_list = merged\_df['matching\_pct\_mov\_trks'].tolist()

is this for caculaing accury for sure? Using confusion mtx

    kpis\_trks = {'result1':

                   {'numerator': num\_of\_SI\_with\_same\_num\_of\_trks,

                    'denominator': num\_of\_same\_SI\_in\_veh\_and\_sim,

                    'value': round((num\_of\_SI\_with\_same\_num\_of\_trks / num\_of\_same\_SI\_in\_veh\_and\_sim) \* 100, 2) if (num\_of\_same\_SI\_in\_veh\_and\_sim != 0) else None},

               'result2':

Why this csv can’t use single csv or data as I need to use hdf?

\_UDP\_GEN7\_ROT\_OBJECT\_STREAM.csv

Why subplot and also used cdn that’s fine

  # Create subplots for accuracy v/s scanindex:

    #################################

    fig\_line = sp.make\_subplots(rows=2, cols=1, horizontal\_spacing=0.04, vertical\_spacing=0.2)

    # Manually add traces for line plots

    fig\_line.add\_trace(func\_line(scan\_index\_list, accuracy\_all\_trks\_list), row=1, col=1)

    fig\_line.update\_yaxes(title\_text="All tracks accuracy", row=1, col=1)

    fig\_line.update\_xaxes(title\_text="Scan Index", row=1, col=1)

    fig\_line.add\_trace(func\_line(scan\_index\_list, accuracy\_mov\_trks\_list), row=2, col=1)

    fig\_line.update\_yaxes(title\_text="Moving tracks accuracy", row=2, col=1)

    fig\_line.update\_xaxes(title\_text="Scan Index", row=2, col=1)

    # Update layout for plot

    fig\_line.update\_layout(height=800, width=1250, title\_text="Accuracy v/s scanindex", showlegend=False)

    fig\_line.update\_traces(marker\_color='red')

    fig\_line.update\_xaxes(zeroline=False, showgrid=False, type='category')

    fig\_line.update\_yaxes(zeroline=False, showgrid=False)

    fig\_line.show()

    line\_plot\_html = pio.to\_html(fig\_line, full\_html=False, include\_plotlyjs='cdn')

detection

why rdd index is taken after scanindex is matched

 # Step 2: Compute matched (rdd1\_rindx, rdd1\_dindx) pairs

    def count\_rindx\_dindx\_matches(row):

        """Count matching (rindx, dindx) pairs between veh and sim."""

        num\_detect\_veh = int(row['rdd1\_num\_detect\_veh'])

        num\_detect\_sim = int(row['rdd1\_num\_detect\_sim'])

why are me doing this after reed d and r index matched

# Step 3: Compute matches within thresholds for rdd2\_range and rdd2\_range\_rate

    def count\_range\_matches(row):

        """Count matches for rdd2\_range and rdd2\_range\_rate within thresholds."""

        num\_detect\_veh = int(row['rdd1\_num\_detect\_veh'])

        num\_detect\_sim = int(row['rdd1\_num\_detect\_sim'])

        #if(num\_detect\_veh < num\_detect\_sim):

        #    print(num\_detect\_veh, num\_detect\_sim)

        veh\_pairs = list(zip(row[rindx\_cols\_veh[:num\_detect\_veh]], row[dindx\_cols\_veh[:num\_detect\_veh]]))

        sim\_pairs = list(zip(row[rindx\_cols\_sim[:num\_detect\_sim]], row[dindx\_cols\_sim[:num\_detect\_sim]]))

        sim\_data = dict(zip(sim\_pairs, zip(row[range\_cols\_sim[:num\_detect\_sim]], row[range\_rate\_cols\_sim[:num\_detect\_sim]])))

what is this line doing:

  merged\_df['range\_rangerate\_matches'] = merged\_df.apply(count\_range\_matches, axis=1)

    merged\_df['same\_num\_of\_RDD1\_detections'] = merged\_df['rdd1\_num\_detect\_veh'] == merged\_df['rdd1\_num\_detect\_sim']

    merged\_df['matching\_pct\_rindx\_dindx\_pairs'] = merged\_df['matched\_rindx\_dindx\_pairs']/merged\_df['rdd1\_num\_detect\_veh']

    merged\_df['matching\_pct\_range\_range

after above line is result that confusion matrix thing

why is this done :

    #################################

    # AF Stream matching

    #################################

    print("Extracting rdd1\_rindx and rdd1\_dindx values from veh\_rdd\_df...")

    #################################

    # Extract and append rdd1\_rindx and rdd1\_dindx values from veh\_rdd\_df to veh\_det\_df

    # based on rdd\_idx columns of veh\_det\_df

    #################################

    # Generate a dictionary for the new columns 'rdd1\_rindx' and 'rdd1\_dindx' with `None` values

    new\_data = {f"rdd1\_rindx\_{i}": None for i in range(max\_number\_of\_data)}

    new\_data.update({f"rdd1\_dindx\_{i}": None for i in range(max\_number\_of\_data)})

    # Create a new DataFrame with these columns

    new\_columns\_df = pd.DataFrame(new\_data, index=veh\_det\_df.index)

    # Concatenate the new columns to the existing DataFrame

    veh\_det\_df = pd.concat([veh\_det\_df, new\_columns\_df], axis=1)

    # Iterate through each row of veh\_det\_df

    for idx, row in veh\_det\_df.iterrows():

        scan\_index = row['scan\_index']

        # Extract corresponding row in veh\_rdd\_df

        rdd\_row = veh\_rdd\_df[veh\_rdd\_df['scan\_index'] == scan\_index]

        # If a matching row is found in veh\_rdd\_df

        if not rdd\_row.empty:

            # Iterate over each rdd\_idx column in veh\_det\_df

            for i in range(len([col for col in veh\_det\_df.columns if col.startswith('rdd\_idx')])):

                rdd\_idx = row[f'rdd\_idx\_{i}']

                # Assign the values from rdd1\_rindx and rdd1\_dindx based on rdd\_idx

                veh\_det\_df.at[idx, f'rdd1\_rindx\_{i}'] = rdd\_row[f'rdd1\_rindx\_{rdd\_idx}'].values[0]

                veh\_det\_df.at[idx, f'rdd1\_dindx\_{i}'] = rdd\_row[f'rdd1\_dindx\_{rdd\_idx}'].values[0]

    # Display the updated veh\_det\_df

    #print(veh\_det\_df)

    #################################

    print("Extracting rdd1\_rindx and rdd1\_dindx values from sim\_rdd\_df...")

    #################################

    # Extract and append rdd1\_rindx and rdd1\_dindx values from sim\_rdd\_df to sim\_det\_df

    # based on rdd\_idx columns of sim\_det\_df

    #################################

    new\_columns\_df = pd.DataFrame(new\_data, index=sim\_det\_df.index)

    # Concatenate the new columns to the existing DataFrame

    sim\_det\_df = pd.concat([sim\_det\_df, new\_columns\_df], axis=1)

    # Iterate through each row of veh\_det\_df

    for idx, row in sim\_det\_df.iterrows():

        scan\_index = row['scan\_index']

        # Extract corresponding row in veh\_rdd\_df

        rdd\_row = sim\_rdd\_df[sim\_rdd\_df['scan\_index'] == scan\_index]

        # If a matching row is found in veh\_rdd\_df

        if not rdd\_row.empty:

            # Iterate over each rdd\_idx column in veh\_det\_df

            for i in range(len([col for col in sim\_det\_df.columns if col.startswith('rdd\_idx')])):

                rdd\_idx = row[f'rdd\_idx\_{i}']

                # Assign the values from rdd1\_rindx and rdd1\_dindx based on rdd\_idx

                sim\_det\_df.at[idx, f'rdd1\_rindx\_{i}'] = rdd\_row[f'rdd1\_rindx\_{rdd\_idx}'].values[0]

                sim\_det\_df.at[idx, f'rdd1\_dindx\_{i}'] = rdd\_row[f'rdd1\_dindx\_{rdd\_idx}'].values[0]

    # Display the updated veh\_det\_df

    #print(sim\_det\_df)

    #################################

    print("Merging the dataframes...")

    #################################

    # Merge Vehicle and Resim data

    #################################

    result\_df = pd.merge(veh\_det\_df, sim\_det\_df, on='scan\_index', how='inner', suffixes=('\_veh', '\_sim'))

    if (max\_num\_of\_si\_to\_process != 0):

        result\_df = result\_df.iloc[:max\_num\_of\_si\_to\_process]

    num\_of\_same\_SI\_in\_veh\_and\_sim\_af = result\_df.shape[0]

    num\_of\_SI\_with\_same\_num\_of\_dets\_af = result\_df[result\_df['num\_af\_det\_veh'] <= result\_df['num\_af\_det\_sim']].shape[0]

    base\_columns = ['scan\_index', 'num\_af\_det\_veh', 'num\_af\_det\_sim']

    repeated\_columns = ['rdd\_idx', 'rdd1\_rindx', 'rdd1\_dindx', 'ran', 'vel', 'theta', 'phi', 'f\_single\_target', 'f\_superres\_target', 'f\_bistatic']

    selected\_columns\_real = [f'{col}\_{i}\_veh' for col in repeated\_columns for i in range(max\_number\_of\_data)]

    selected\_columns\_sim = [f'{col}\_{i}\_sim' for col in repeated\_columns for i in range(max\_number\_of\_data)]

    selected\_columns = base\_columns + selected\_columns\_real + selected\_columns\_sim

    final\_df = result\_df[selected\_columns]

    #final\_df['scan\_index'] = final\_df['scan\_index'].astype(int)

    # print(final\_df.head())

    #################################

After this again above threshold and other result things are their

And agin mannuly adding why ?

   # Manually add traces for scatter plots

    # Range error against detection properties

    row\_num = 1

    rang = [ele[rng\_idx] for ele in ran\_diff\_list]

    fig\_scatter.add\_trace(func\_scatter(rang, ran\_diffs), row=row\_num, col=1)

    fig\_scatter.update\_yaxes(title\_text="Range error", row=row\_num, col=1)

    fig\_scatter.update\_xaxes(title\_text="Range", row=row\_num, col=1)

    vel = [ele[vel\_idx] for ele in ran\_diff\_list]

    fig\_scatter.add\_trace(func\_scatter(vel, ran\_diffs), row=row\_num, col=2)

    fig\_scatter.update\_xaxes(title\_text="Range Rate", row=row\_num, col=2)

    theta = [ele[theta\_idx] for ele in ran\_diff\_list]

    fig\_scatter.add\_trace(func\_scatter(theta, ran\_diffs), row=row\_num, col=3)

    fig\_scatter.update\_xaxes(title\_text="Azimuth", row=row\_num, col=3)

    phi = [ele[phi\_idx] for ele in ran\_diff\_list]

    fig\_scatter.add\_trace(func\_scatter(phi, ran\_diffs), row=row\_num, col=4)

    fig\_scatter.update\_xaxes(title\_text="Elevation", row=row\_num, col=4)

    single\_target = [ele[single\_target\_idx] for ele in ran\_diff\_list]

    fig\_scatter.add\_trace(func\_scatter(single\_target, ran\_diffs), row=row\_num, col=5)

    fig\_scatter.update\_xaxes(title\_text="Single Target", row=row\_num, col=5)

    superres\_target = [ele[superres\_target\_idx] for ele in ran\_diff\_list]

    fig\_scatter.add\_trace(func\_scatter(superres\_target, ran\_diffs), row=row\_num, col=6)

    fig\_scatter.update\_xaxes(title\_text="Superres Target", row=row\_num, col=6)

    bistatic = [ele[bistatic\_idx] for ele in ran\_diff\_list]

    fig\_scatter.add\_trace(func\_scatter(bistatic, ran\_diffs), row=row\_num, col=7)

    fig\_scatter.update\_xaxes(title\_text="Bistatic", row=row\_num, col=7)

1. Will all the name will be similar because the code is tightly coupled with name if name changes to customer then this code might not work