SEQUENCE DIAGRAM

@startuml

actor User

participant Main

participant CVFilter

participant CSVReader

User -> Main: Start

Main -> CSVReader: read\_measurements\_from\_csv(file\_path)

CSVReader --> Main: measurements

Main -> CVFilter: initialize\_filter()

loop for each measurement\_group

Main -> CVFilter: process\_measurement\_group()

alt First measurement

CVFilter -> CVFilter: initialize\_filter\_state()

else Second measurement

CVFilter -> CVFilter: initialize\_filter\_state(second)

else Further measurements

CVFilter -> CVFilter: predict\_step()

CVFilter -> CVFilter: form\_clusters()

CVFilter -> CVFilter: generate\_hypotheses()

CVFilter -> CVFilter: select\_best\_hypothesis()

alt Best hypothesis found

CVFilter -> CVFilter: update\_step()

end

end

end

Main -> Main: plot\_results()

User <-- Main: End

@enduml

Mermaid live editor

sequenceDiagram

participant User

participant Main

participant CVFilter

participant CSVReader

User ->> Main: Start

Main ->> CSVReader: read\_measurements\_from\_csv(file\_path)

CSVReader -->> Main: measurements

Main ->> CVFilter: initialize\_filter()

loop for each measurement\_group

Main ->> CVFilter: process\_measurement\_group()

alt First measurement

CVFilter ->> CVFilter: initialize\_filter\_state()

else Second measurement

CVFilter ->> CVFilter: initialize\_filter\_state(second)

else Further measurements

CVFilter ->> CVFilter: predict\_step()

CVFilter ->> CVFilter: form\_clusters()

CVFilter ->> CVFilter: generate\_hypotheses()

CVFilter ->> CVFilter: select\_best\_hypothesis()

alt Best hypothesis found

CVFilter ->> CVFilter: update\_step()

end

end

end

Main ->> Main: plot\_results()

User -->> Main: End

A class diagram will help illustrate the structure of your code, showing the classes, their attributes, methods, and the relationships between them. Here's a breakdown of the classes and their relationships in your script:

### Classes and Relationships

1. \*\*CVFilter\*\*:

- \*\*Attributes\*\*:

- Sf, Pf, Sp, Pp, plant\_noise, H, R, Meas\_Time, prev\_Time, Q, Phi, Z, Z1, Z2, first\_rep\_flag, second\_rep\_flag, gate\_threshold

- \*\*Methods\*\*:

- \_\_init\_\_, initialize\_filter\_state, predict\_step, update\_step, gating

2. \*\*Main Function\*\*:

- \*\*Functions\*\*:

- main, form\_measurement\_groups, read\_measurements\_from\_csv, chi\_square\_clustering, form\_clusters, generate\_hypotheses, compute\_hypothesis\_likelihood, jpda, sph2cart, cart2sph, cart2sph2

### PlantUML Code

```plantuml

@startuml

class CVFilter {

- np.ndarray Sf

- np.ndarray Pf

- np.ndarray Sp

- np.ndarray Pp

- float plant\_noise

- np.ndarray H

- np.ndarray R

- float Meas\_Time

- float prev\_Time

- np.ndarray Q

- np.ndarray Phi

- np.ndarray Z

- np.ndarray Z1

- np.ndarray Z2

- bool first\_rep\_flag

- bool second\_rep\_flag

- float gate\_threshold

\_\_init\_\_()

initialize\_filter\_state(x, y, z, vx, vy, vz, time)

predict\_step(current\_time)

update\_step(Z)

gating(Z)

}

class Main {

main()

form\_measurement\_groups(measurements, max\_time\_diff)

read\_measurements\_from\_csv(file\_path)

chi\_square\_clustering(Z, kalman\_filter)

form\_clusters(measurements, kalman\_filter)

generate\_hypotheses(clusters)

compute\_hypothesis\_likelihood(hypothesis, kalman\_filter)

jpda(clusters, kalman\_filter)

sph2cart(az, el, r)

cart2sph(x, y, z)

cart2sph2(x, y, z, filtered\_values\_csv)

}

Main --> CVFilter : Uses

@enduml

```

### Mermaid Code

```mermaid

classDiagram

class CVFilter {

- np.ndarray Sf

- np.ndarray Pf

- np.ndarray Sp

- np.ndarray Pp

- float plant\_noise

- np.ndarray H

- np.ndarray R

- float Meas\_Time

- float prev\_Time

- np.ndarray Q

- np.ndarray Phi

- np.ndarray Z

- np.ndarray Z1

- np.ndarray Z2

- bool first\_rep\_flag

- bool second\_rep\_flag

- float gate\_threshold

\_\_init\_\_()

initialize\_filter\_state(x, y, z, vx, vy, vz, time)

predict\_step(current\_time)

update\_step(Z)

gating(Z)

}

class Main {

main()

form\_measurement\_groups(measurements, max\_time\_diff)

read\_measurements\_from\_csv(file\_path)

chi\_square\_clustering(Z, kalman\_filter)

form\_clusters(measurements, kalman\_filter)

generate\_hypotheses(clusters)

compute\_hypothesis\_likelihood(hypothesis, kalman\_filter)

jpda(clusters, kalman\_filter)

sph2cart(az, el, r)

cart2sph(x, y, z)

cart2sph2(x, y, z, filtered\_values\_csv)

}

Main --> CVFilter : Uses

```

### Instructions for Visualization

#### PlantUML

1. Go to [PlantText](https://www.planttext.com/).

2. Copy the PlantUML code provided above.

3. Paste the code into the PlantText editor.

4. Click "Refresh" to generate the diagram.

#### Mermaid

1. Go to [Mermaid Live Editor](https://mermaid-js.github.io/mermaid-live-editor/).

2. Copy the Mermaid code provided above.

3. Paste the code into the Mermaid editor.

4. The diagram will be generated automatically.

FLOWCHART

Great choice! A flowchart will effectively illustrate the sequence of operations and decision points in your code. Let's break down the main components and structure the flowchart accordingly.

Here is a high-level description of the functionality and structure of your code:

1. \*\*Initialization and Data Reading\*\*:

- Initialize the Kalman filter (`CVFilter`).

- Read measurements from a CSV file.

2. \*\*Measurement Processing\*\*:

- Group measurements based on time difference.

- Iterate through each measurement group.

3. \*\*Filter Initialization and Update\*\*:

- Initialize or update the Kalman filter state.

- Predict the next state.

- Form clusters from measurements.

- Generate hypotheses and select the best one.

- Update the filter state based on the best hypothesis.

4. \*\*Plotting\*\*:

- Plot the range, azimuth, and elevation against time.

I'll create a flowchart using PlantUML and Mermaid to represent this flow. First, here is the PlantUML code:

### PlantUML Code

```plantuml

@startuml

start

:Initialize CVFilter;

:Read measurements from CSV;

:Form measurement groups;

foreach measurement\_group in measurement\_groups

:Process each measurement group;

if (First measurement?) then (yes)

:Initialize filter state;

else (no)

if (Second measurement?) then (yes)

:Initialize filter state with second measurement;

else (no)

:Predict next state;

:Form clusters;

:Generate hypotheses;

:Select best hypothesis;

if (Best hypothesis found?) then (yes)

:Update filter state;

endif

endif

endif

endforeach

:Plot range vs. time;

:Plot azimuth vs. time;

:Plot elevation vs. time;

stop

@enduml

```

### Mermaid Code

```mermaid

graph TD

A[Initialize CVFilter] --> B[Read measurements from CSV]

B --> C[Form measurement groups]

C --> D{Process each measurement group}

D --> E{First measurement?}

E -->|Yes| F[Initialize filter state]

E -->|No| G{Second measurement?}

G -->|Yes| H[Initialize filter state with second measurement]

G -->|No| I[Predict next state]

I --> J[Form clusters]

J --> K[Generate hypotheses]

K --> L[Select best hypothesis]

L --> M{Best hypothesis found?}

M -->|Yes| N[Update filter state]

M -->|No| D

N --> D

D --> O[Plot range vs. time]

O --> P[Plot azimuth vs. time]

P --> Q[Plot elevation vs. time]

```

### Instructions for Visualization

#### PlantUML

1. Go to [PlantText](https://www.planttext.com/).

2. Copy the PlantUML code provided above.

3. Paste the code into the PlantText editor.

4. Click "Refresh" to generate the diagram.

#### Mermaid

1. Go to [Mermaid Live Editor](https://mermaid-js.github.io/mermaid-live-editor/).

2. Copy the Mermaid code provided above.

3. Paste the code into the Mermaid editor.

4. The diagram will be generated automatically.