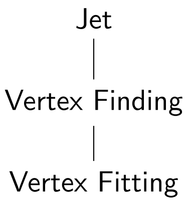
Toy SSVF

Lucio Derin

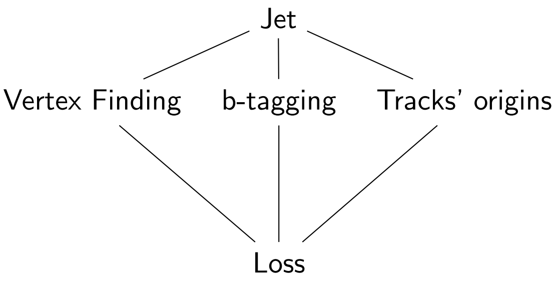
12/12/2022

Project description

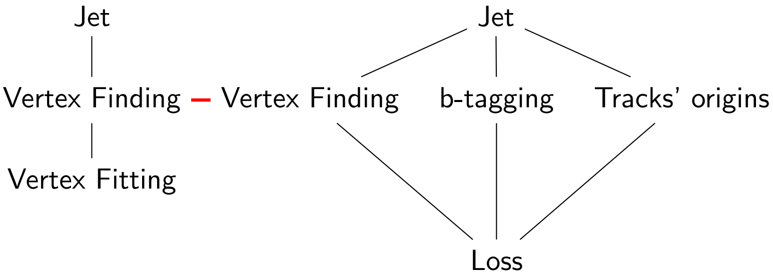
SSVF:



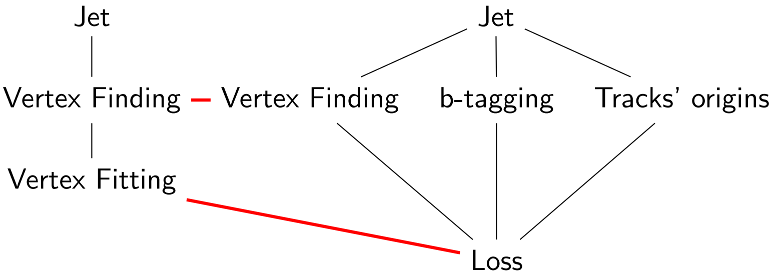
GN1:



Project description



Project description



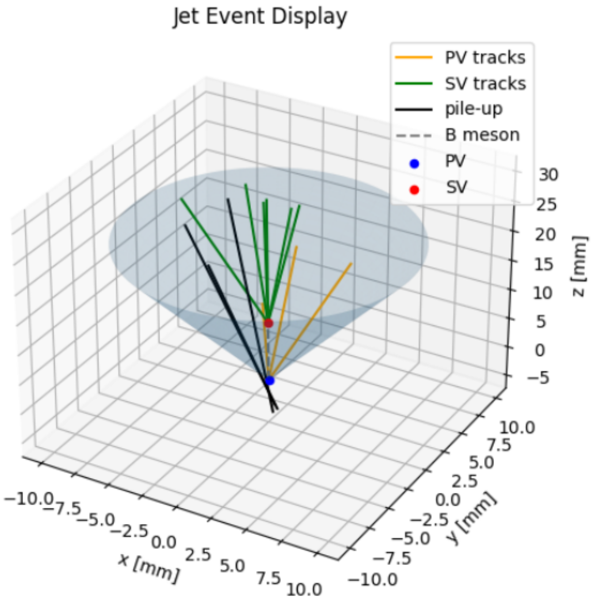
Step 0: understanding vertex finding (pseudo-code)

```
# List of coupled tracks (two-tracks vertex approximation)
couples = []
indices = [i for i in range(len(tracks))]
# While there are tracks to be coupled
while len(tracks)>1:
    # Select the track to be coupled
    i,t = indices.pop(0),tracks.pop(0)
    #selecting the closest track
    j = argmin(distances)
    # if tracks are close enough, couple them
    if distances[j] < threshold:
        tc = tracks.pop(j)
        indices.pop(j)
        couples.append([t,tc])
return couples
```

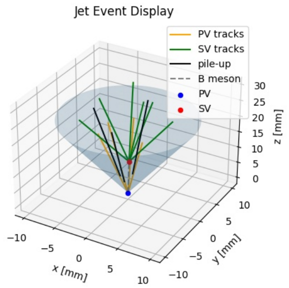
Step 0: understanding vertex fitting (pseudo-code)

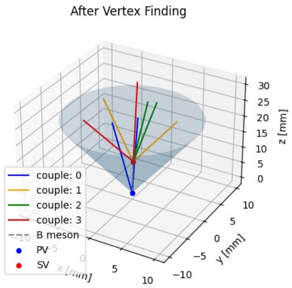
```
# fitting loop
while True:
    # Find the SV that minimizes the chi2
    # from the current selected tracks
    SVfitted = min(chi2(tracks))
    # If chi2 is good enough, break
    if chi2 < threshold:
        break
    # Else, reject the worst track
    tracks.pop(argmax(chi2s))
    # If there are less than 2 tracks,
    # no vertex has been found
    if len(tracks)<2:
        return None
return SVfitted
```

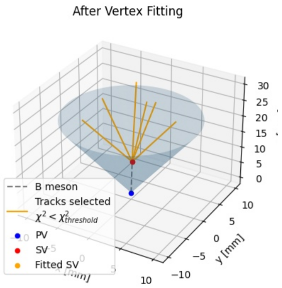
Testing the algorithm: Toy Jets



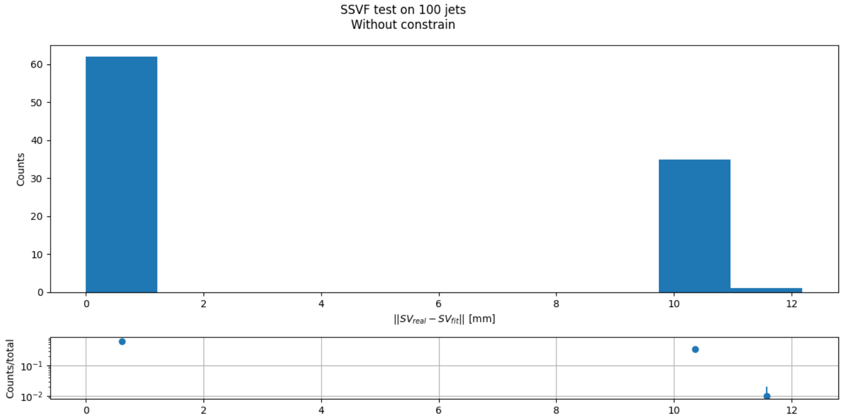
Testing the algorithm: SSVF on toy Jets



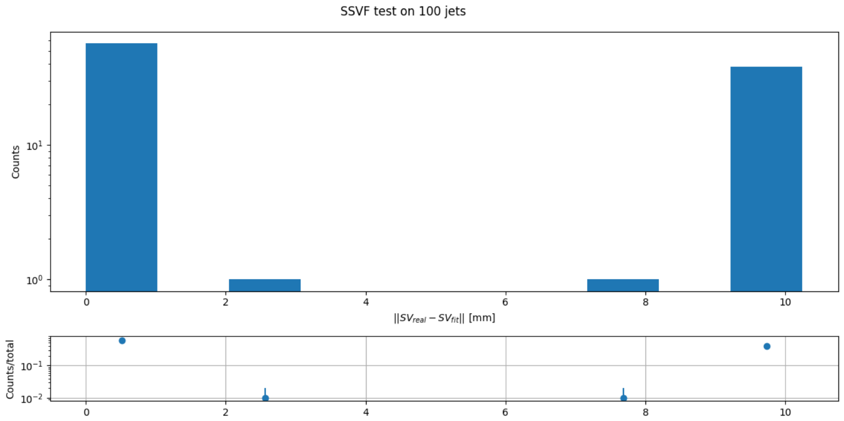




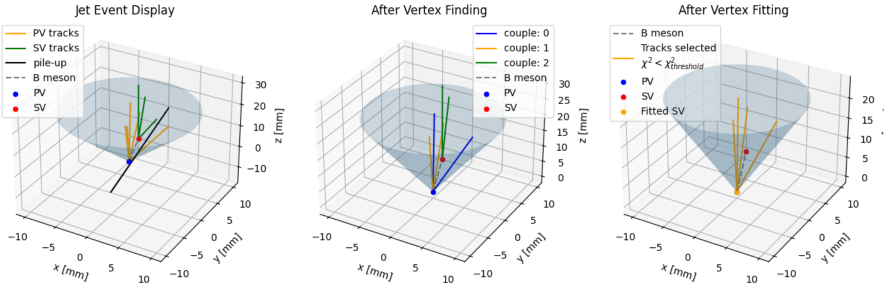
Testing the algorithm: SSVF errors on 100 Jets



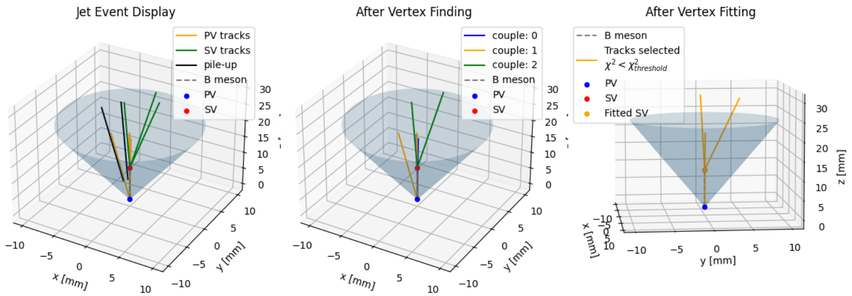
Testing the algorithm: SSVF errors on 100 Jets



Error 1: fit converges to PV



Error 2: tracks from PV are not rejected



Alternative algorithm: clustering-based vertex finding

SSVF:

- Needs PV's tracks rejection to avoid convergence to PV;
- $O(N^2)$

Alternative algorithm: clustering-based vertex finding

SSVF:

- Needs PV's tracks rejection to avoid convergence to PV;
- $O(N^2)$

Alternative algorithm: clustering-based vertex finding

C-SSVF based on K-Means clustering:

- Automatically clusters together only high IP tracks from SV;
- LLoyd implementation: $O(tkNd) \implies O(N)$

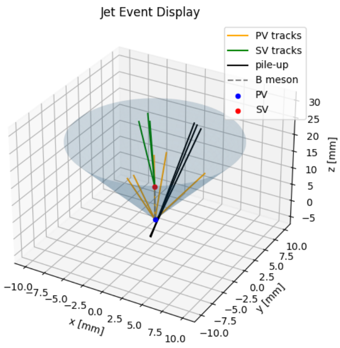
Alternative algorithm: clustering-based vertex finding

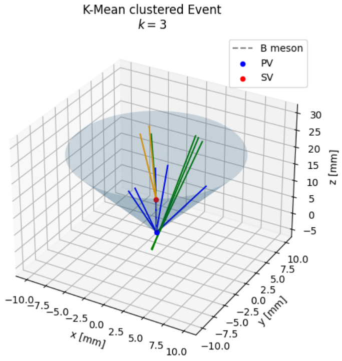
C-SSVF based on K-Means clustering:

- Automatically clusters together only high IP tracks from SV;
- LLoyd implementation: $O(tkNd) \implies O(N)$

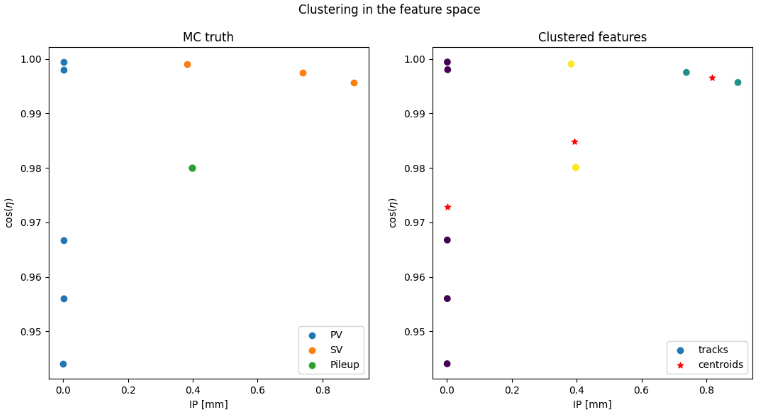
$$\min_{m_i \in \mathbb{R}^D} \sum_{i=1}^{n} \min_{j=1,K} |x_i - m_j|^2$$

Culstering based vertex finding

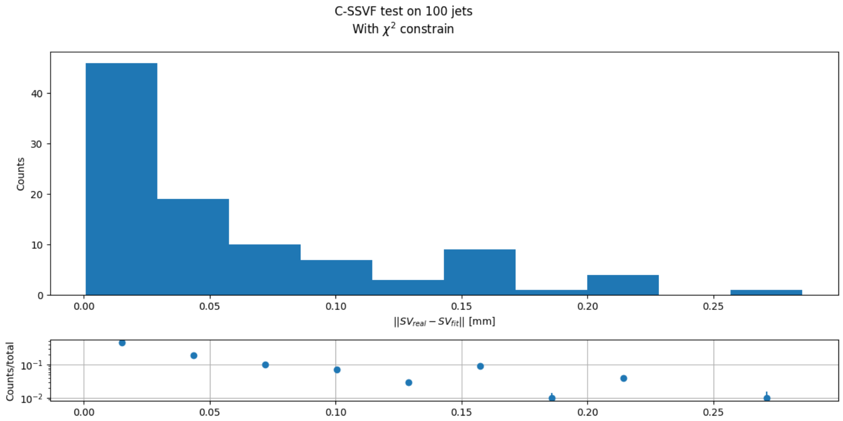




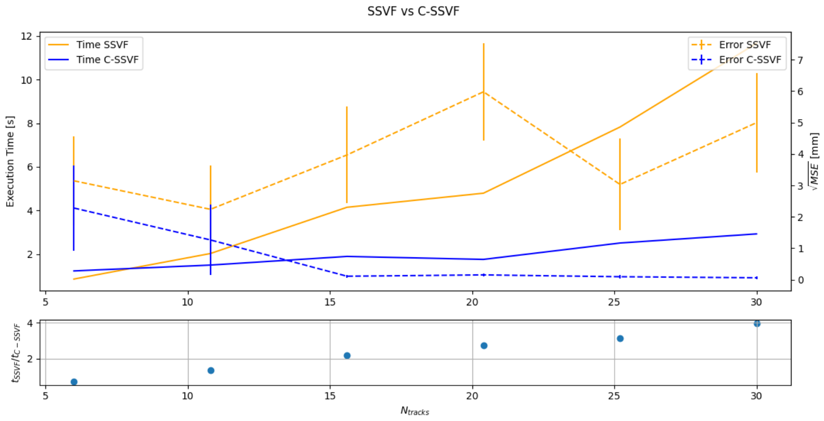
Culstering based vertex finding



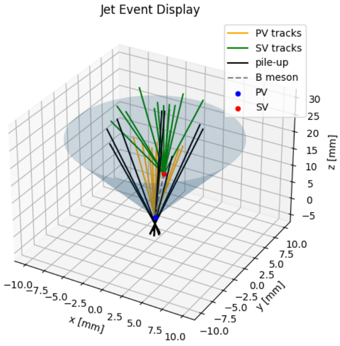
Testing the algorithm: C-SSVF errors on 100 Jets

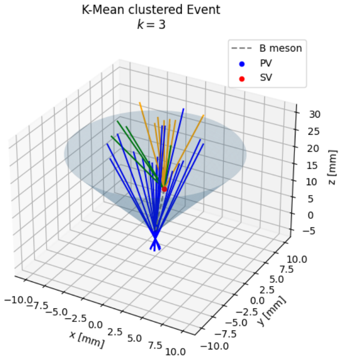


SSVF vs C-SSVF

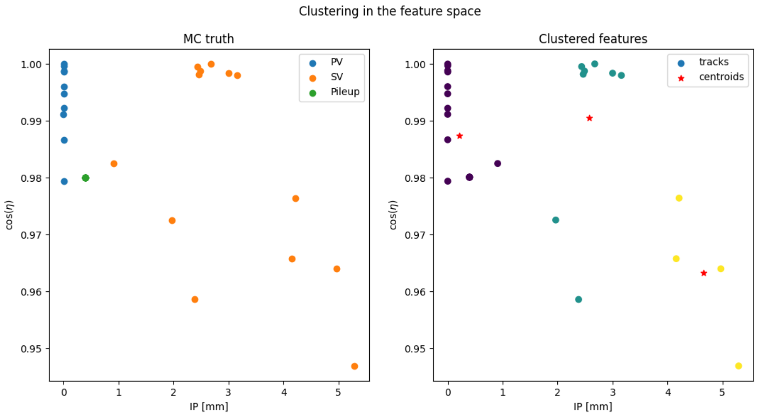


C-SSVF in dense jets





C-SSVF in dense jets



Next up

- Display real Jets;
- Apply SSVF to real Jets;
- Repeat all previous analysis on real Jets;
- ...

Next up

- Display real Jets;
- Apply SSVF to real Jets;
- Repeat all previous analysis on real Jets;
- ...

Next up

- Display real Jets;
- Apply SSVF to real Jets;
- Repeat all previous analysis on real Jets;
- ...

Next up

- Display real Jets;
- Apply SSVF to real Jets;
- Repeat all previous analysis on real Jets;
- ...