SSVF algorithm pseudo-code

Step 1: vertex finding

```
# List of coupled tracks (two-tracks vertex approximation)
couples = []
indices = [i for i in range(len(tracks))]
# While there are have 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 tracks
    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 2: vertex fitting

```
# 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
```

Tracks distances calculations

Let the track line be represented in a parametric form:

$$r : \begin{cases} x = x_0 + \cos(\varphi)\sin(\theta)t \\ y = y_0 + \sin(\varphi)\sin(\theta)t \\ z = z_0 + \cos(\theta)t \end{cases}$$

where $\boldsymbol{P} = (x_0, y_0, z_0)^\top$ is the origin of the track and

$$\hat{\boldsymbol{v}} = (\cos(\varphi)\sin(\theta), \sin(\varphi)\sin(\theta), \cos(\theta))^\top$$

is the track versor.

Distance between two tracks

The minimum distance between two lines is the length of the only segment that is orthogonal to both of them. This segment is the solution of the following system:

$$\begin{cases} \Delta\boldsymbol{P} \cdot \hat{r}_1 = 0 \\ \Delta\boldsymbol{P} \cdot \hat{r}_2 = 0 \end{cases}$$

where $\Delta\boldsymbol{P}$ is the vectorial difference between the parametric points of the two lines, which are:

$$\begin{array}{c} Pr_1 = (x_0 + at, y_0 + bt, z_0 + ct) \\ Pr_2 = (x_0^{'} + a^{'}s, y_0^{'} + b^{'}s, z_0^{'} + c^{'}s) \end{array}$$

Thus the system can be written as:

$$\begin{cases} \Delta\boldsymbol{P}_0 \cdot \hat{\boldsymbol{r}}_1 + (\hat{\boldsymbol{r}}_1 \cdot \hat{\boldsymbol{r}}_2)s - \hat{\boldsymbol{r}}_1^2 t = 0 \\ \Delta\boldsymbol{P}_0 \cdot \hat{\boldsymbol{r}}_2 + \hat{\boldsymbol{r}}_2^2 s - (\hat{\boldsymbol{r}}_1 \cdot \hat{\boldsymbol{r}}_2)t = 0 \end{cases}$$

Where $\Delta\boldsymbol{P}_0$ is the vector that connects the tracks' origins.
The system is solved by:

$$\begin{cases} t^* = -\frac{1}{\sin^2(\alpha)}\left[\cos(\alpha)(\Delta \boldsymbol{P}_0 \cdot \hat{\boldsymbol{r}}_2) - \Delta \boldsymbol{P}_0 \cdot \hat{\boldsymbol{r}}_1\right] \\ s^* = \cos(\alpha)t^* - \Delta \boldsymbol{P}_0 \cdot \hat{\boldsymbol{r}}_2 \end{cases}$$

By plugging the values of t^* and s^* in the parametric representation of the lines one finds the lines' closest points. The distance between the two points is the minimum distance between the lines.

Distance between a track and a point

The distance between a line and a point $\boldsymbol{P}_d = (x_d, y_d, z_d)^\top$ can be evaluated in two steps. The first step is finding the plane orthogonal to the line that contains the point.
A general plane is given by:

$$\pi : ax + by + cz + d = 0$$

A plane orthogonal to the line must have its versor parallel to the line's one: $\hat{\boldsymbol{\pi}} = c\hat{\boldsymbol{v}}$. WLOG, $c = 1$.
So a general plane orthogonal to the line is given by:

$$v_0x + v_1y + v_2z + d = 0$$

Now, the plane must contain $\boldsymbol{P}_d$ so:

$$d^* = -(v_0x_d + v_1y_d + v_2z_d)$$

The line's point that's closest to $\boldsymbol{P}_d$ is the intersection between the plane and the line. By substituting the parametric representation of the line in the plane equation:

$$v_0x_0 + v_1y_0 + v_2z_0 + v_0^2t + v_1^2t + v_2^2t + d^* = 0$$

Finally:

$$t^* = -(\frac{v_0x_0 + v_1y_0 + v_0z_2}{v_0^2 + v_1^2 + v_2^2} + d^*)$$

The distance is the distance between $\boldsymbol{P}_d$ and the parametric representation of the line evaluated at $t = t^*$.