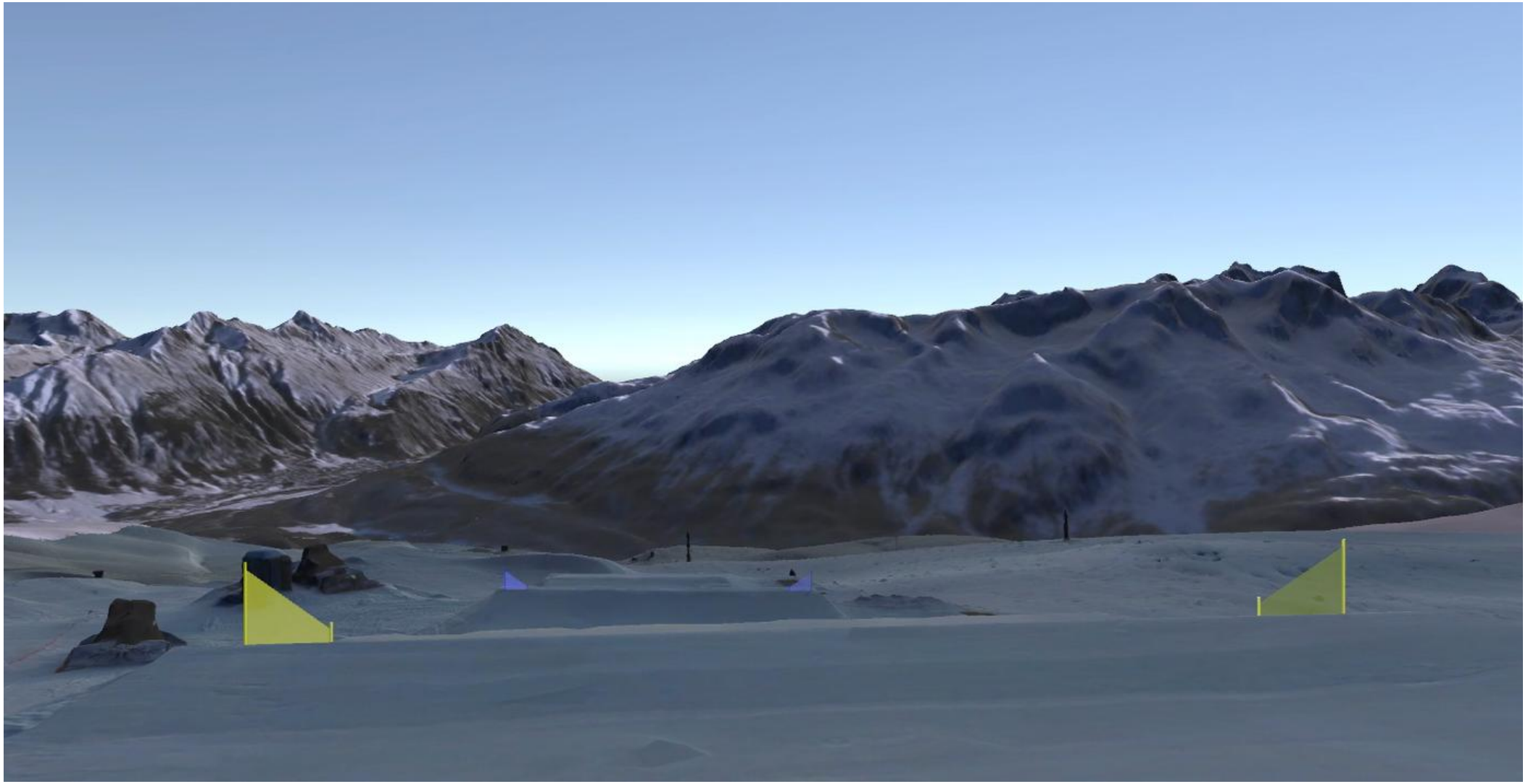
TRAJECTORY FILTERING AND JUMP MODELLING IN SKI CROSS VR-APPLICATION

Samuel Schwyn[1], Fabien Délèze[1], Sébastien Guillaume[1], Björn Bruhin[2]

[1] **INSIT, Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud (HEIG-VD), Yverdon-les-Bains, Switzerland**

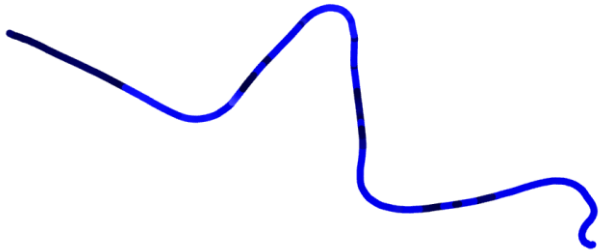
[2] **Swiss-Ski, Worblaufen, Switzerland**

Main goal : First person VR-App for ski cross



What we need

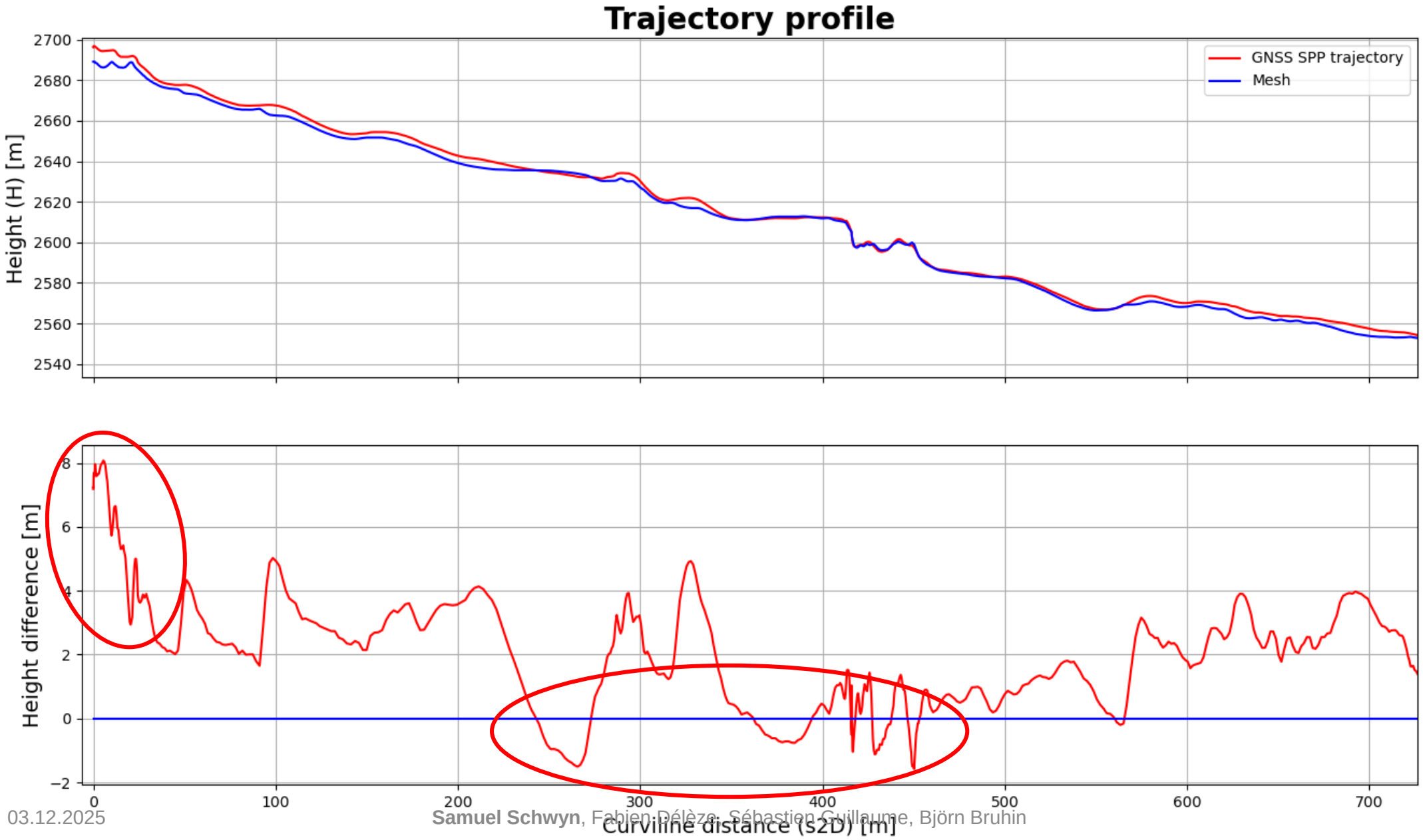
3D-model (high precision : ~0.1m)



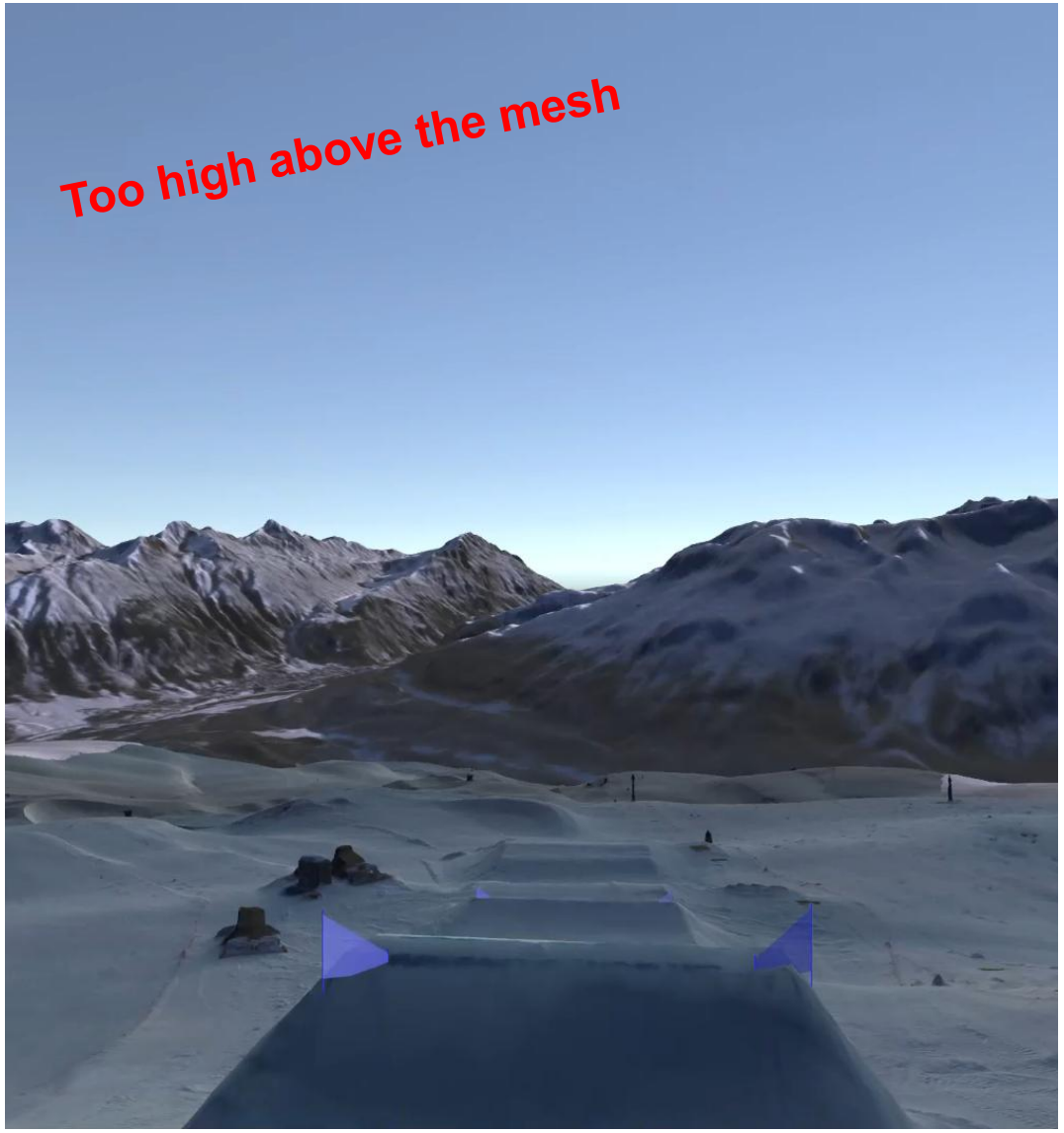
$$\mathbf{x}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}$$

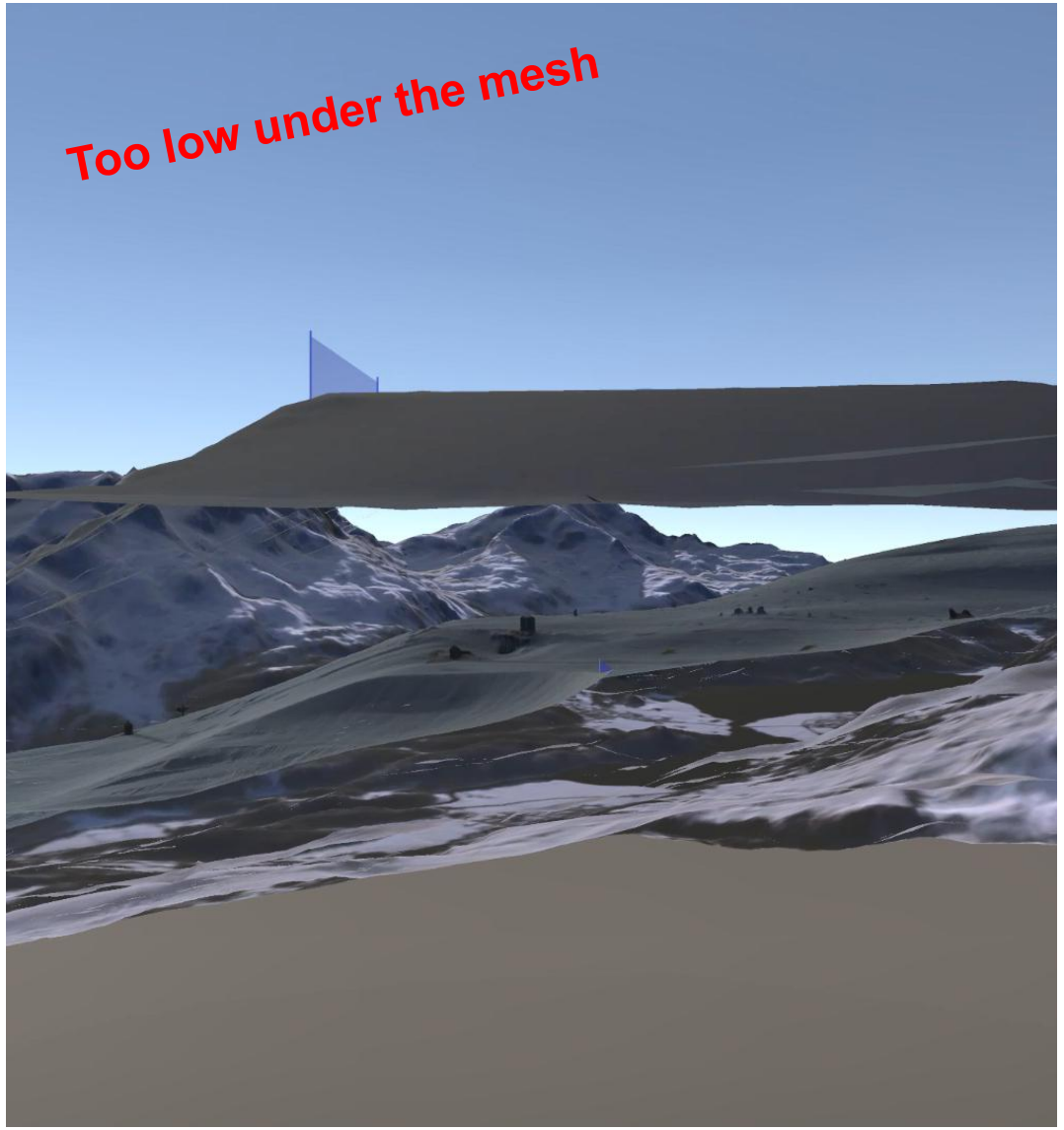
Trajectory (for example from GNSS)

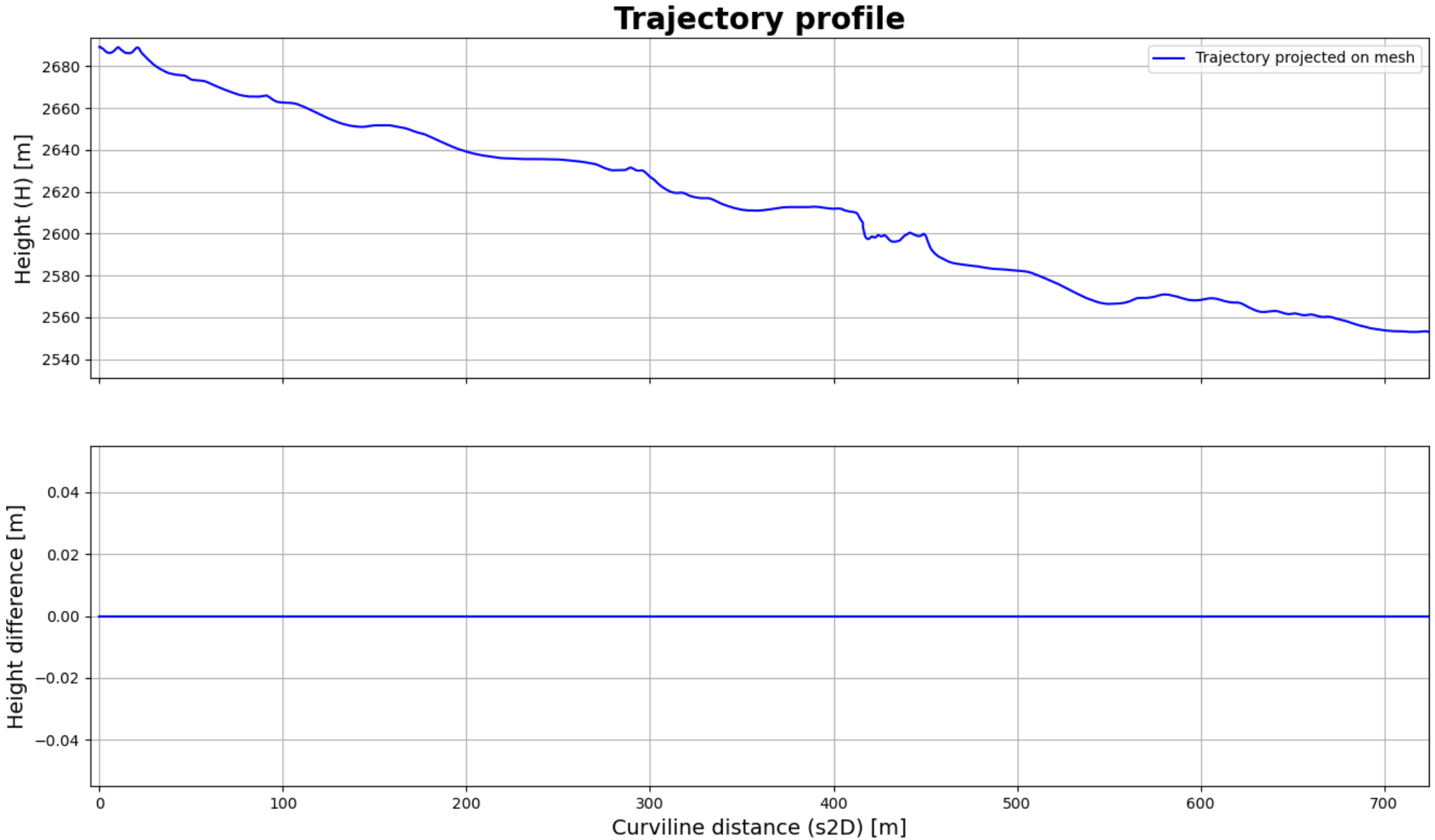
Single Point Positionning (SPP) :
Hz : ~2m V : ~6m


Trajectory profile
GNSS SPP trajectory
Mesh
Height (H) [m]
2700
2680
2660
2640
2620
2600
2580
2560
2540
Height difference [m]
8
6
4
2
0
−2
0
100
200
300
400
500
600
700
Curviline distance (s2D) [m]

Motivation for filtering

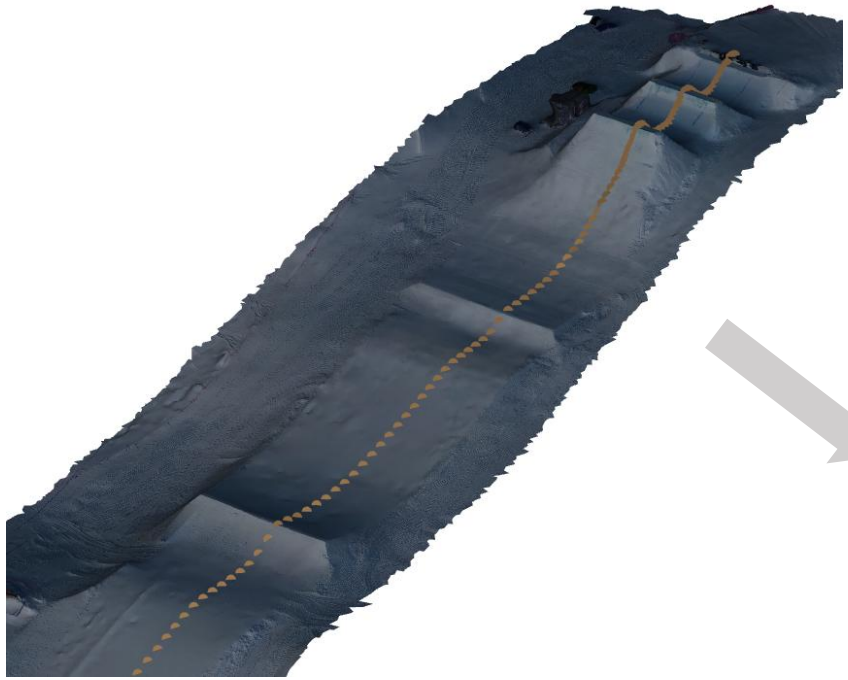




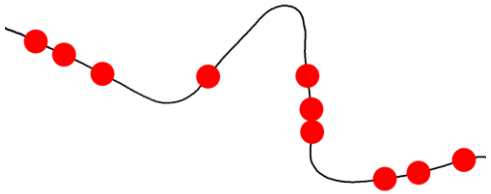

Trajectory profile
Trajectory projected on mesh
Height (H) [m]
2680
2660
2640
2620
2600
2580
2560
2540
Height difference [m]
0.04
0.02
0.00
−0.02
−0.04
0
100
200
300
400
500
600
700
Curviline distance (s2D) [m]

Projection SPP trajectory on mesh

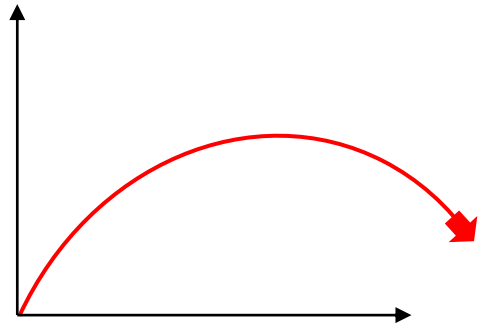
Fusion GNSS with ballistic model



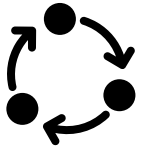
GNSS SPP
trajectory projected
on mesh



Jump positions



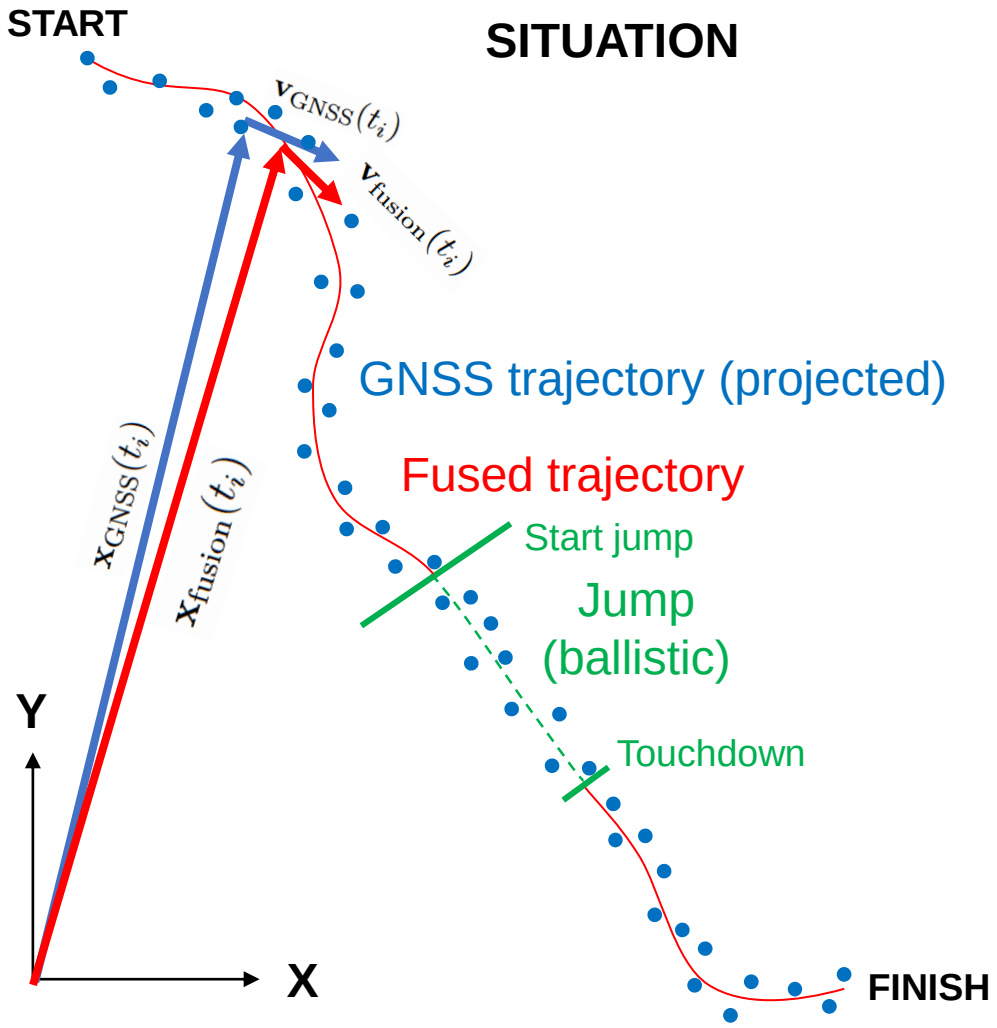
Ballistic model

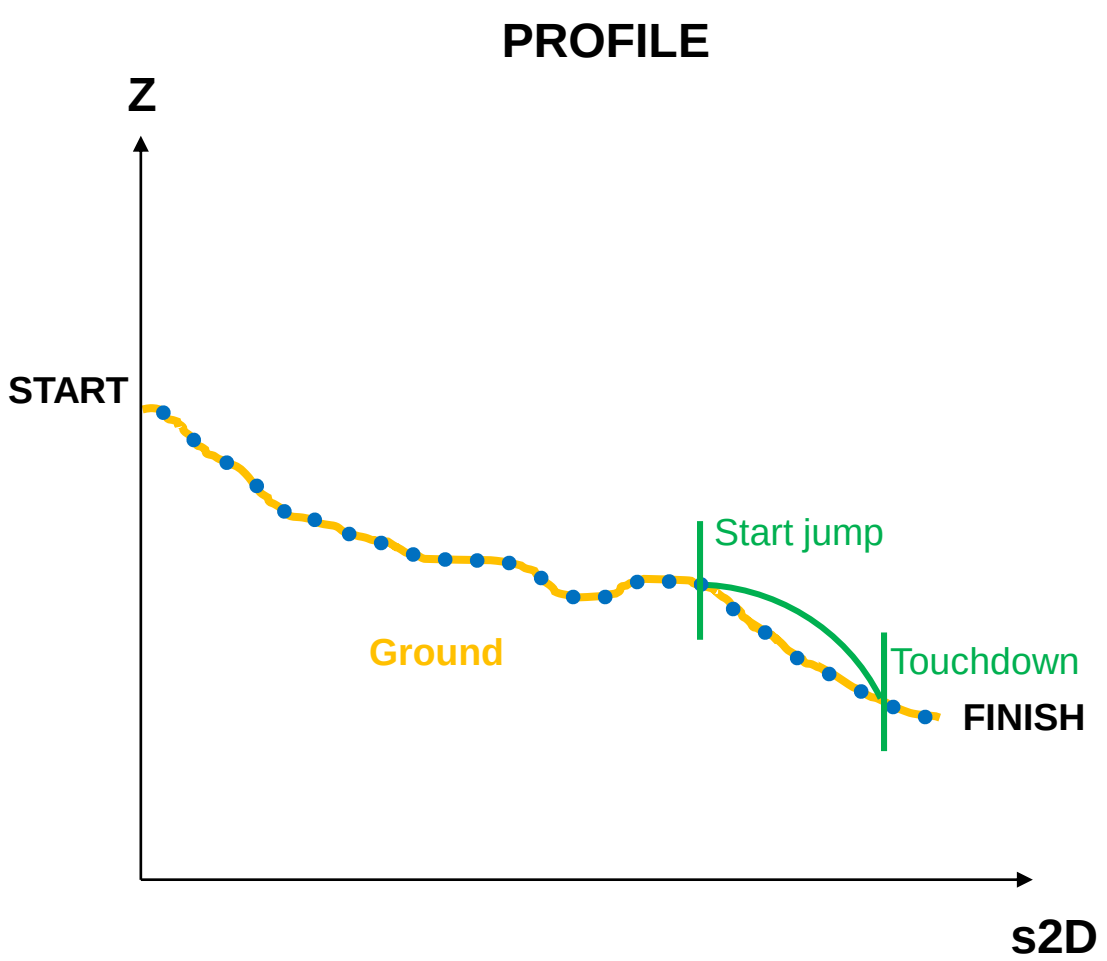


Fusion algorithm
(P-Controller)

Fused trajectory

Fusion GNSS with ballistic model



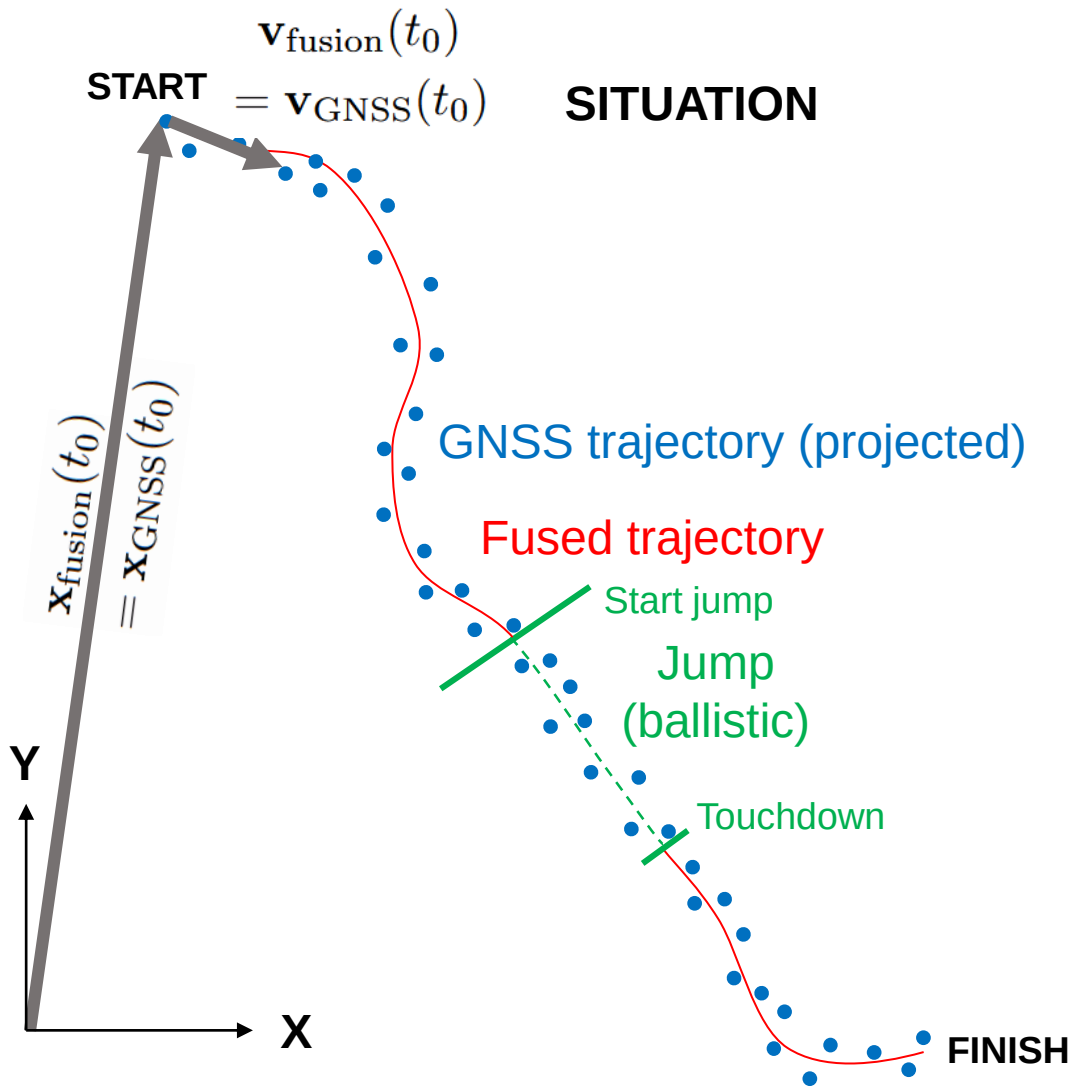


Fusion GNSS with ballistic model

Initial state vectors

$$\mathbf{x}_{\text{fusion}}(t_0) = \mathbf{x}_{\text{GNSS}}(t_0)$$
$$\mathbf{v}_{\text{fusion}}(t_0) = \mathbf{v}_{\text{GNSS}}(t_0)$$

$$\Delta t(t_0) = t_1 - t_0$$
$$\Delta \mathbf{v}(t_0) = \mathbf{0}$$



Fusion GNSS with ballistic model

Propagation of the state vectors on the ground

$$\mathbf{x}_{\text{fusion}}(t_{i+1}) = \mathbf{x}_{\text{fusion}}(t_i) + \mathbf{v}_{\text{fusion}}(t_i) \cdot \Delta t(t_i)$$

$$\mathbf{v}_{\text{fusion}}(t_{i+1}) = \mathbf{v}_{\text{fusion}}(t_i) + \Delta \mathbf{v}(t_i)$$

With :

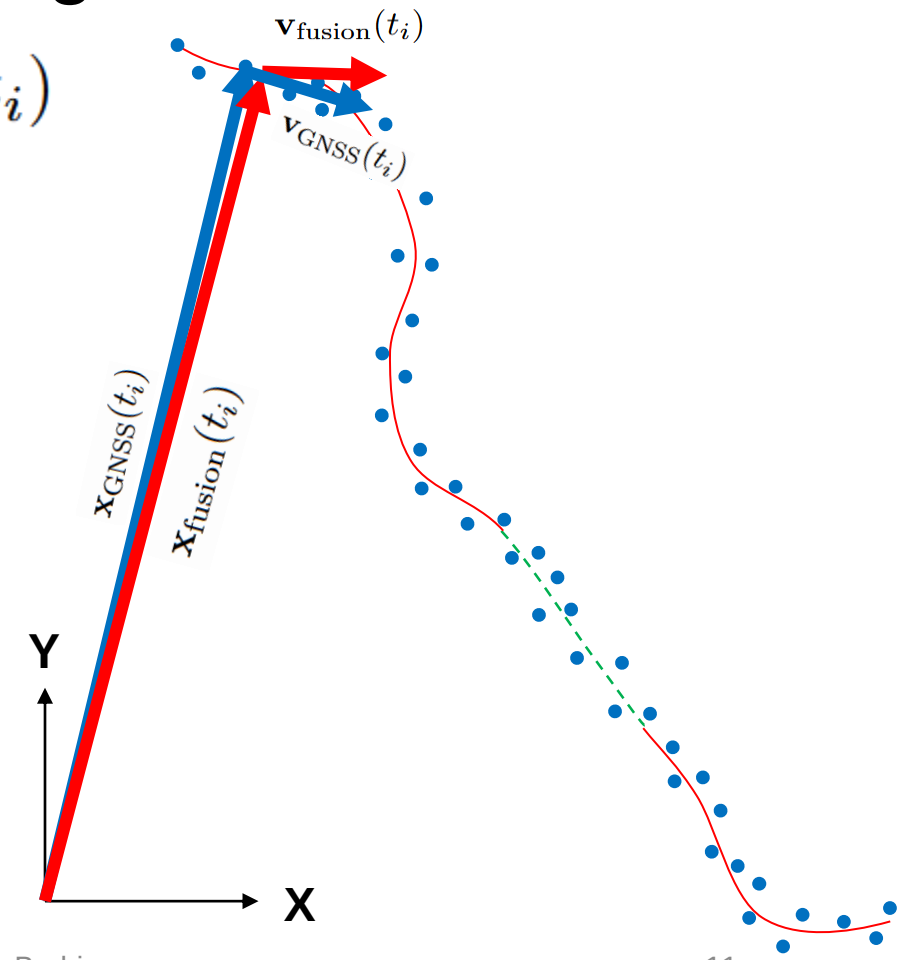
$$\Delta t(t_i) = t_{i+1} - t_i$$

simple P-Controller
for **position** and **velocity** matching

$$\Delta \mathbf{v}(t_i) = -k_{\mathbf{x}} \cdot \mathbf{e}_{\mathbf{x}}(t_i) - k_{\mathbf{v}} \cdot \mathbf{e}_{\mathbf{v}}(t_i)$$

$$\mathbf{e}_{\mathbf{x}}(t_i) = \mathbf{x}_{\text{GNSS}}(t_i) - \mathbf{x}_{\text{fusion}}(t_i)$$

$$\mathbf{e}_{\mathbf{v}}(t_i) = \mathbf{v}_{\text{GNSS}}(t_i) - \mathbf{v}_{\text{fusion}}(t_i)$$



Fusion GNSS with ballistic model

Propagation of the state vectors on the ground

$$\mathbf{x}_{\text{fusion}}(t_{i+1}) = \mathbf{x}_{\text{fusion}}(t_i) + \mathbf{v}_{\text{fusion}}(t_i) \cdot \Delta t(t_i)$$

$$\mathbf{v}_{\text{fusion}}(t_{i+1}) = \mathbf{v}_{\text{fusion}}(t_i) + \Delta \mathbf{v}(t_i)$$

With :

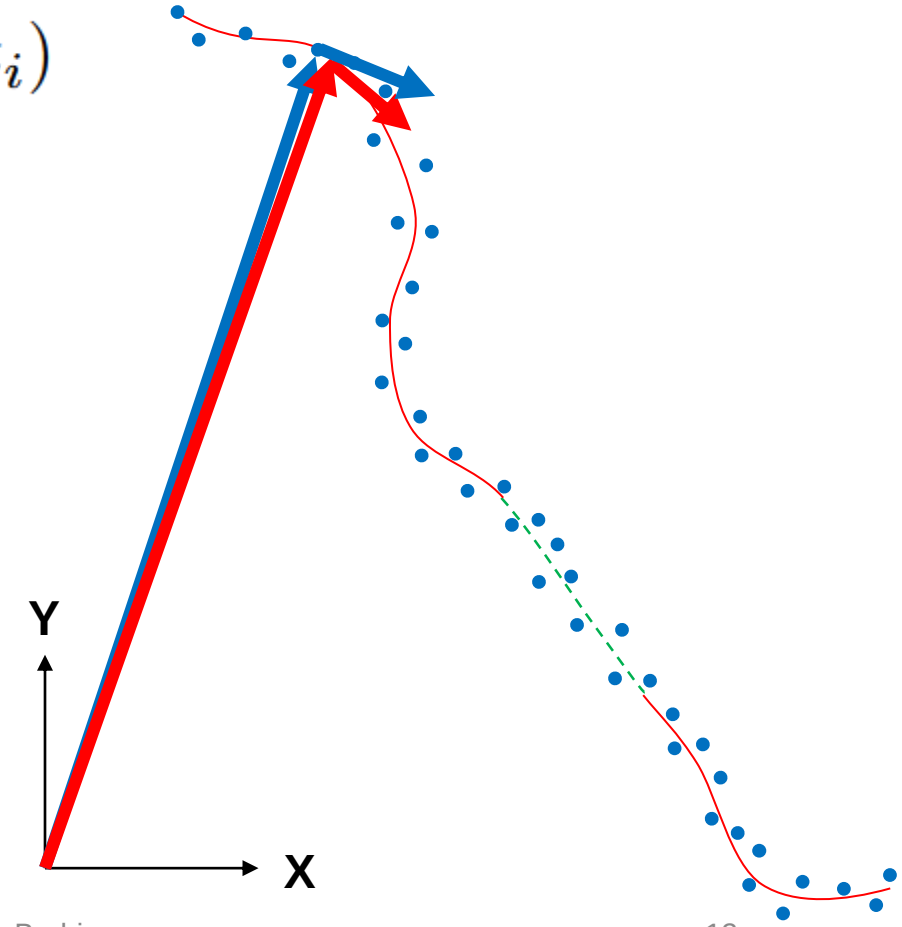
$$\Delta t(t_i) = t_{i+1} - t_i$$

simple P-Controller
for **position** and **velocity** matching

$$\Delta \mathbf{v}(t_i) = -k_{\mathbf{x}} \cdot \mathbf{e}_{\mathbf{x}}(t_i) - k_{\mathbf{v}} \cdot \mathbf{e}_{\mathbf{v}}(t_i)$$

$$\mathbf{e}_{\mathbf{x}}(t_i) = \mathbf{x}_{\text{GNSS}}(t_i) - \mathbf{x}_{\text{fusion}}(t_i)$$

$$\mathbf{e}_{\mathbf{v}}(t_i) = \mathbf{v}_{\text{GNSS}}(t_i) - \mathbf{v}_{\text{fusion}}(t_i)$$



Fusion GNSS with ballistic model

Propagation of the state vectors on the ground

$$\mathbf{x}_{\text{fusion}}(t_{i+1}) = \mathbf{x}_{\text{fusion}}(t_i) + \mathbf{v}_{\text{fusion}}(t_i) \cdot \Delta t(t_i)$$

$$\mathbf{v}_{\text{fusion}}(t_{i+1}) = \mathbf{v}_{\text{fusion}}(t_i) + \Delta \mathbf{v}(t_i)$$

With :

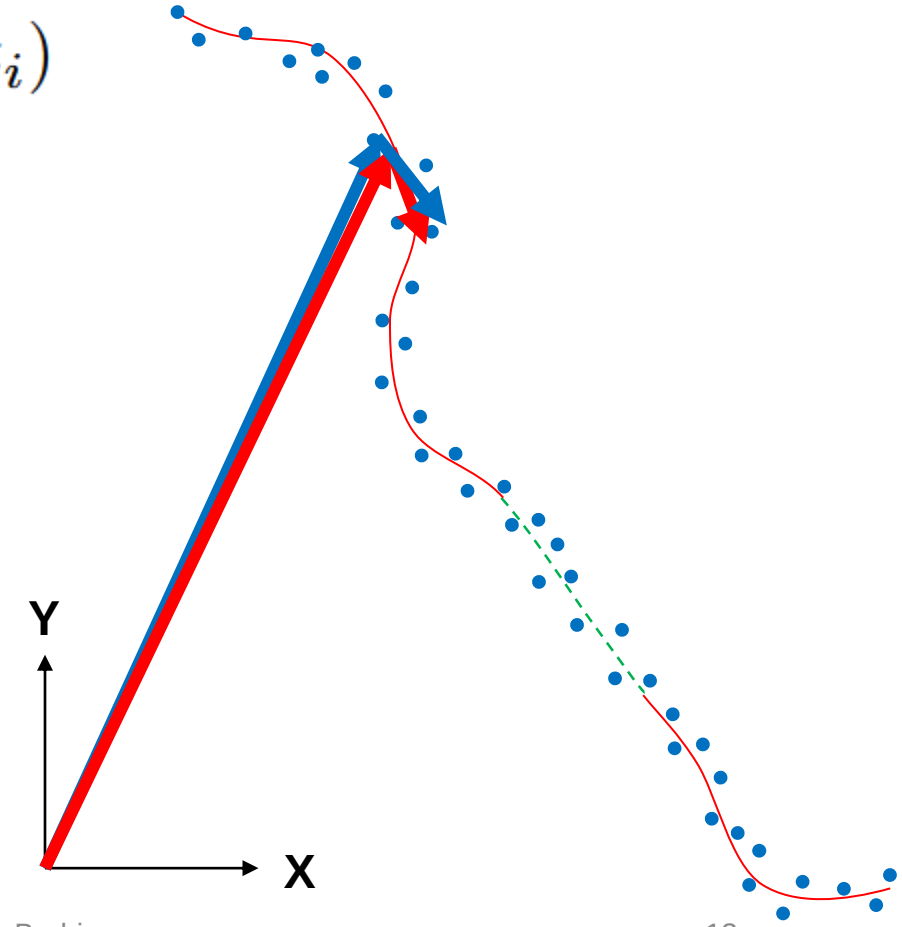
$$\Delta t(t_i) = t_{i+1} - t_i$$

simple P-Controller
for **position** and **velocity** matching

$$\Delta \mathbf{v}(t_i) = -k_{\mathbf{x}} \cdot \mathbf{e}_{\mathbf{x}}(t_i) - k_{\mathbf{v}} \cdot \mathbf{e}_{\mathbf{v}}(t_i)$$

$$\mathbf{e}_{\mathbf{x}}(t_i) = \mathbf{x}_{\text{GNSS}}(t_i) - \mathbf{x}_{\text{fusion}}(t_i)$$

$$\mathbf{e}_{\mathbf{v}}(t_i) = \mathbf{v}_{\text{GNSS}}(t_i) - \mathbf{v}_{\text{fusion}}(t_i)$$



Fusion GNSS with ballistic model

Propagation of the state vectors on the ground

$$\mathbf{x}_{\text{fusion}}(t_{i+1}) = \mathbf{x}_{\text{fusion}}(t_i) + \mathbf{v}_{\text{fusion}}(t_i) \cdot \Delta t(t_i)$$

$$\mathbf{v}_{\text{fusion}}(t_{i+1}) = \mathbf{v}_{\text{fusion}}(t_i) + \Delta \mathbf{v}(t_i)$$

With :

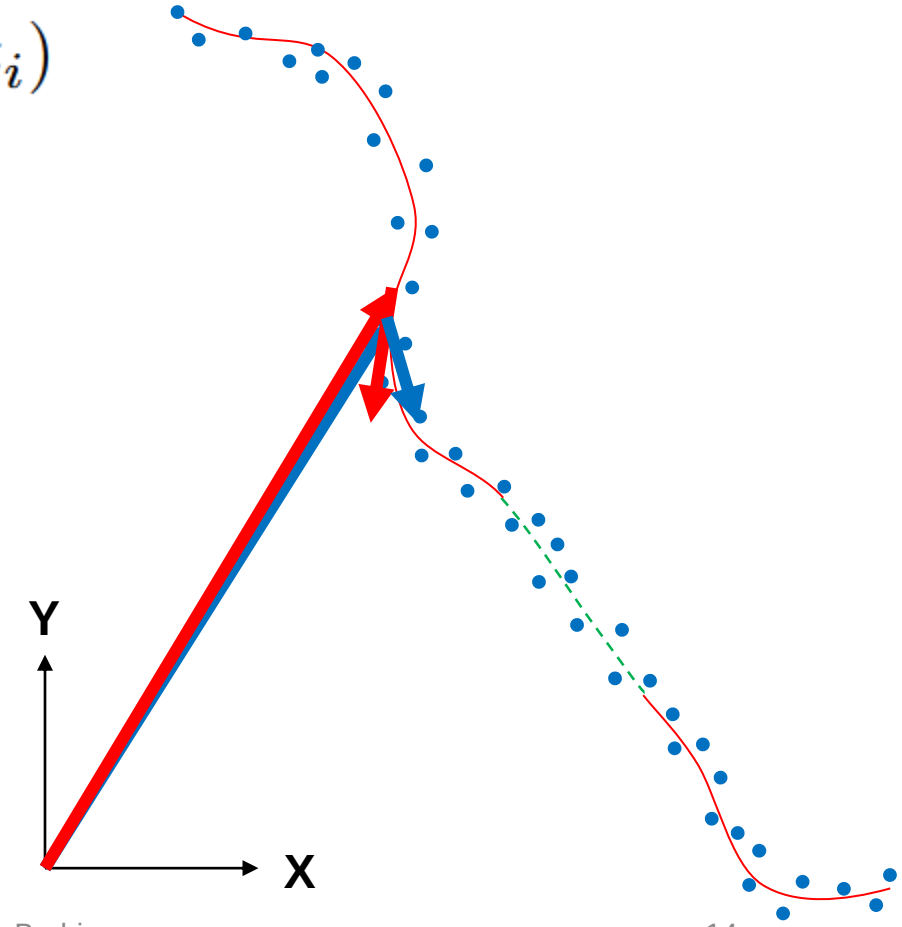
$$\Delta t(t_i) = t_{i+1} - t_i$$

simple P-Controller
for **position** and **velocity** matching

$$\Delta \mathbf{v}(t_i) = \boxed{-k_{\mathbf{x}} \cdot \mathbf{e}_{\mathbf{x}}(t_i) - k_{\mathbf{v}} \cdot \mathbf{e}_{\mathbf{v}}(t_i)}$$

$$\mathbf{e}_{\mathbf{x}}(t_i) = \mathbf{x}_{\text{GNSS}}(t_i) - \mathbf{x}_{\text{fusion}}(t_i)$$

$$\mathbf{e}_{\mathbf{v}}(t_i) = \mathbf{v}_{\text{GNSS}}(t_i) - \mathbf{v}_{\text{fusion}}(t_i)$$



Fusion GNSS with ballistic model

Propagation of the state vectors on the ground

$$\mathbf{x}_{\text{fusion}}(t_{i+1}) = \mathbf{x}_{\text{fusion}}(t_i) + \mathbf{v}_{\text{fusion}}(t_i) \cdot \Delta t(t_i)$$

$$\mathbf{v}_{\text{fusion}}(t_{i+1}) = \mathbf{v}_{\text{fusion}}(t_i) + \Delta \mathbf{v}(t_i)$$

With :

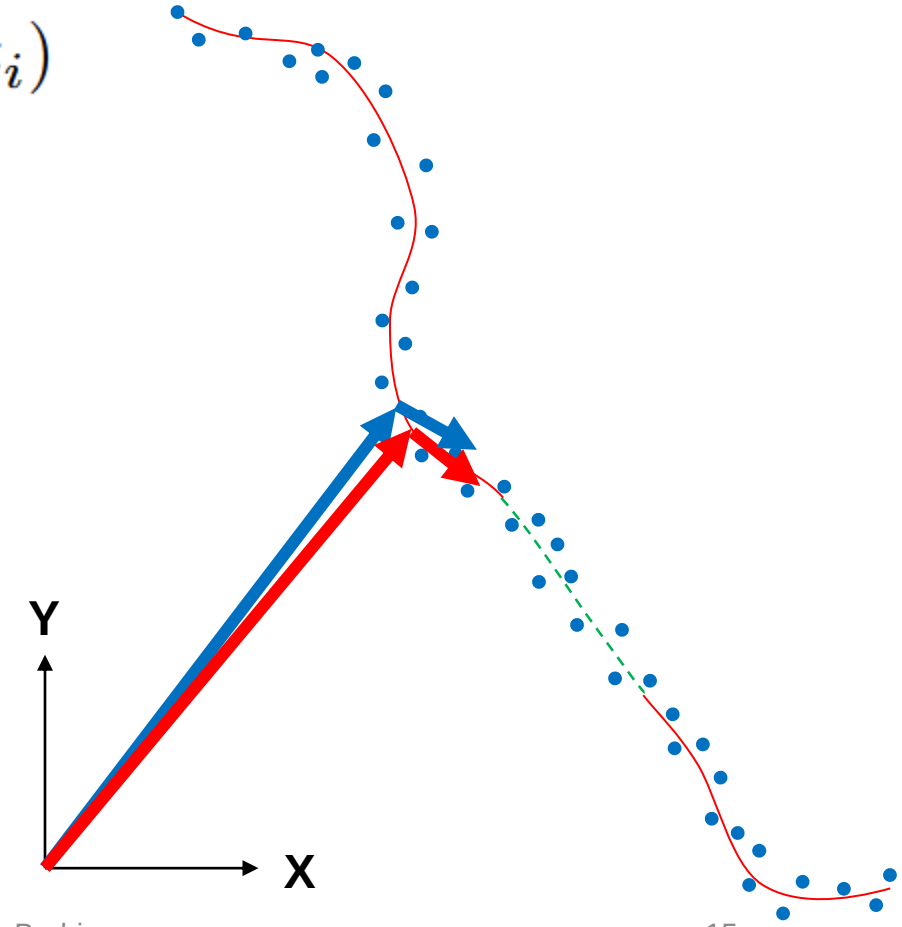
$$\Delta t(t_i) = t_{i+1} - t_i$$

simple P-Controller
for **position** and **velocity** matching

$$\Delta \mathbf{v}(t_i) = \boxed{-k_{\mathbf{x}} \cdot \mathbf{e}_{\mathbf{x}}(t_i) - k_{\mathbf{v}} \cdot \mathbf{e}_{\mathbf{v}}(t_i)}$$

$$\mathbf{e}_{\mathbf{x}}(t_i) = \mathbf{x}_{\text{GNSS}}(t_i) - \mathbf{x}_{\text{fusion}}(t_i)$$

$$\mathbf{e}_{\mathbf{v}}(t_i) = \mathbf{v}_{\text{GNSS}}(t_i) - \mathbf{v}_{\text{fusion}}(t_i)$$



Fusion GNSS with ballistic model

Initial conditions for ballistic integration (jump start)

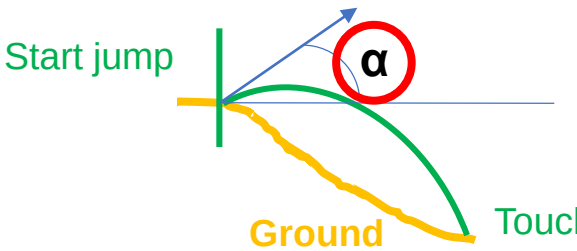
$$\mathbf{x}_{\text{fusion}}\left(t_{\text{jump}_0^j}\right) = \mathbf{x}_{\text{fusion}}\left(t_i\right)$$

$$\mathbf{v}_{\text{fusion}}\left(t_{\text{jump}_0^j}\right) = \mathbf{v}_{\text{fusion}}\left(t_i\right) + \Delta\mathbf{v}_{\text{impulse}^j}$$

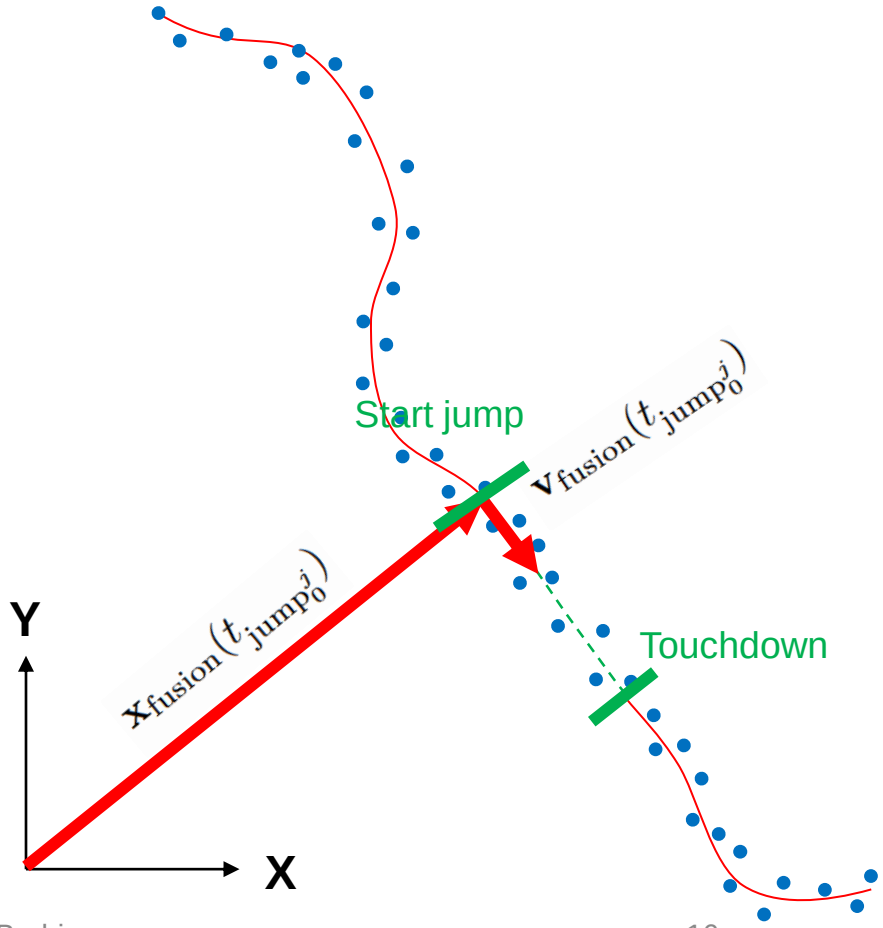
With :

$$\Delta\mathbf{v}_{\text{impulse}^j} = \begin{pmatrix} 0 \\ 0 \\ \alpha^{[\text{rad}]} \cdot |\mathbf{v}_{\text{fusion}}(t_i)| \end{pmatrix}$$

PROFILE



SITUATION



Fusion GNSS with ballistic model

Propagation of the state vectors in flight

$$\mathbf{x}_{\text{fusion}}(t_{i+1}) = \mathbf{x}_{\text{fusion}}(t_i) + \mathbf{v}_{\text{fusion}}(t_i) \cdot \Delta t(t_i)$$

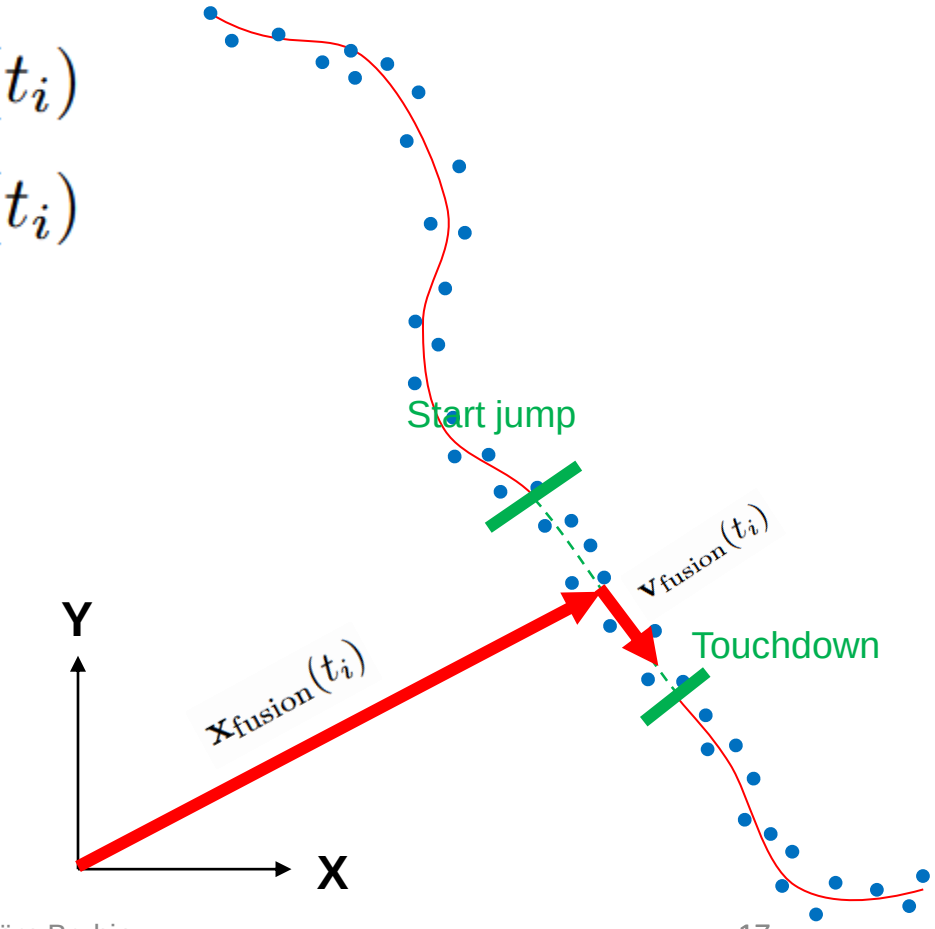
$$\mathbf{v}_{\text{fusion}}(t_{i+1}) = \mathbf{v}_{\text{fusion}}(t_i) + \mathbf{a}_{\text{fusion}}(t_i) \cdot \Delta t(t_i)$$

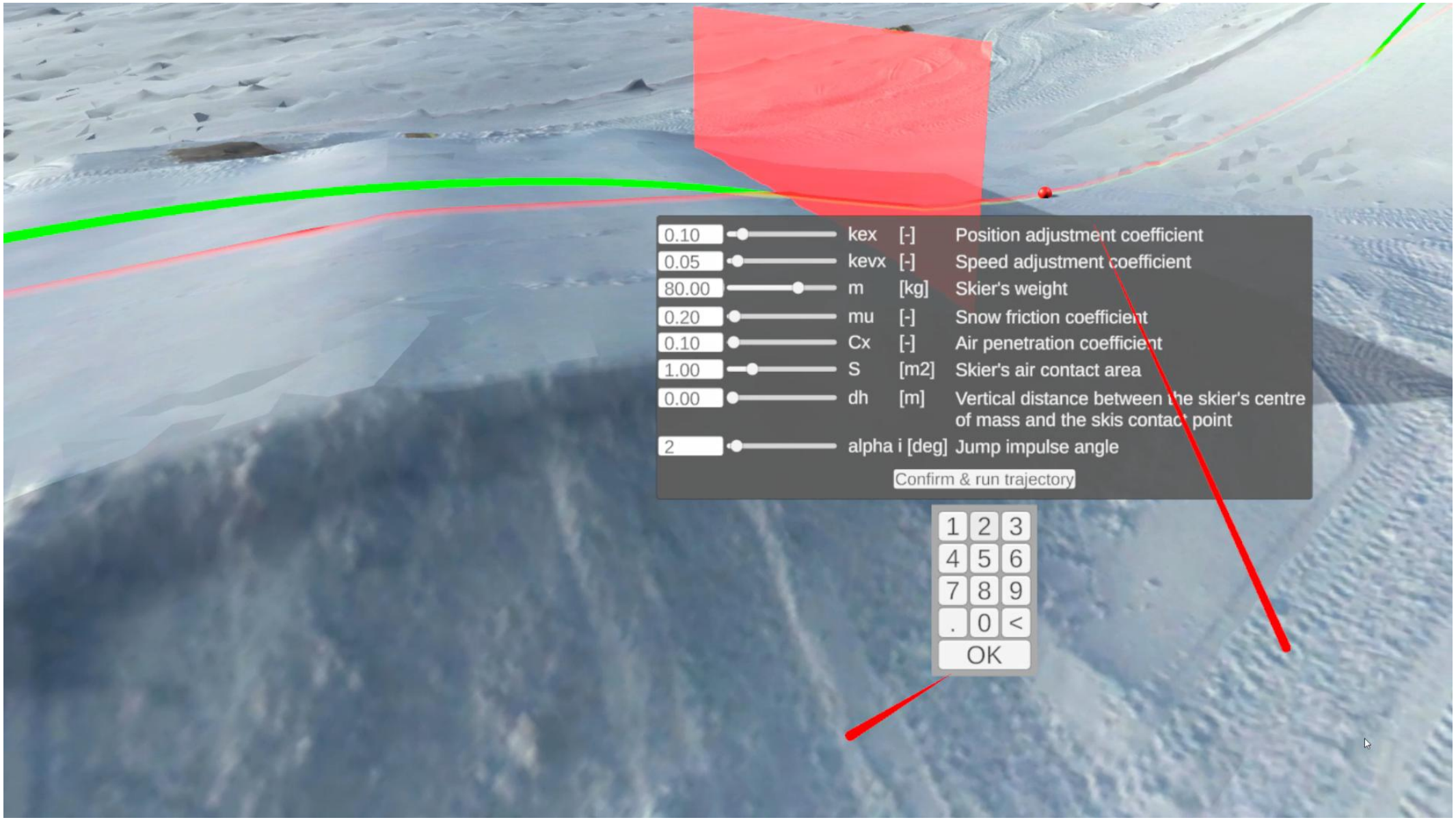
With :

$$\mathbf{a}_{\text{fusion}}(t_i) = \mathbf{g} + \frac{1}{m} \cdot \mathbf{F}_{\text{airdrag}}$$

$$\mathbf{F}_{\text{airdrag}} = -\frac{1}{2} \cdot \rho_{\text{air}} \cdot C_x \cdot S \cdot |\mathbf{v}_{\text{fusion}}(t_i)|^2 \cdot \frac{\mathbf{v}_{\text{fusion}}(t_i)}{|\mathbf{v}_{\text{fusion}}(t_i)|}$$

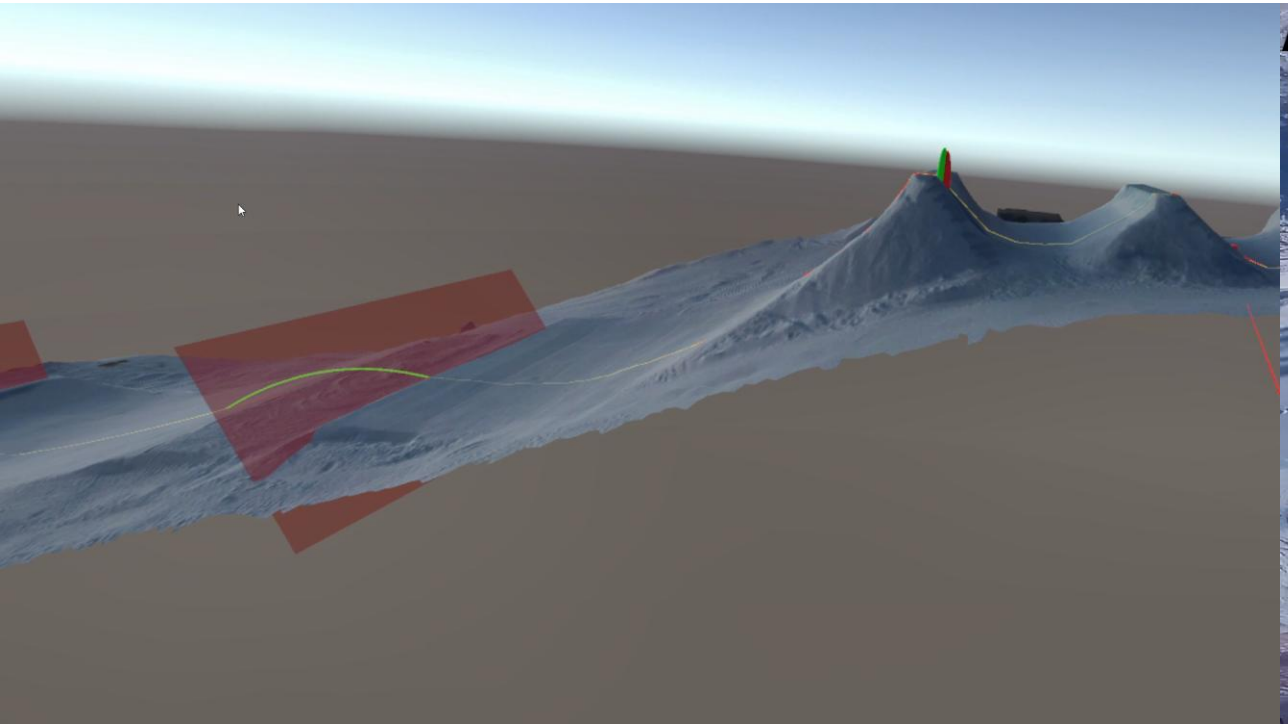
SITUATION




0.10 kex [-] Position adjustment coefficient
0.05 kevx [-] Speed adjustment coefficient
80.00 m [kg] Skier's weight
0.20 mu [-] Snow friction coefficient
0.10 Cx [-] Air penetration coefficient
1.00 S [m2] Skier's air contact area
0.00 dh [m] Vertical distance between the skier's centre of mass and the skis contact point
2 alpha i [deg] Jump impulse angle
Confirm & run trajectory
1 2 3
4 5 6
7 8 9
. 0 <
OK

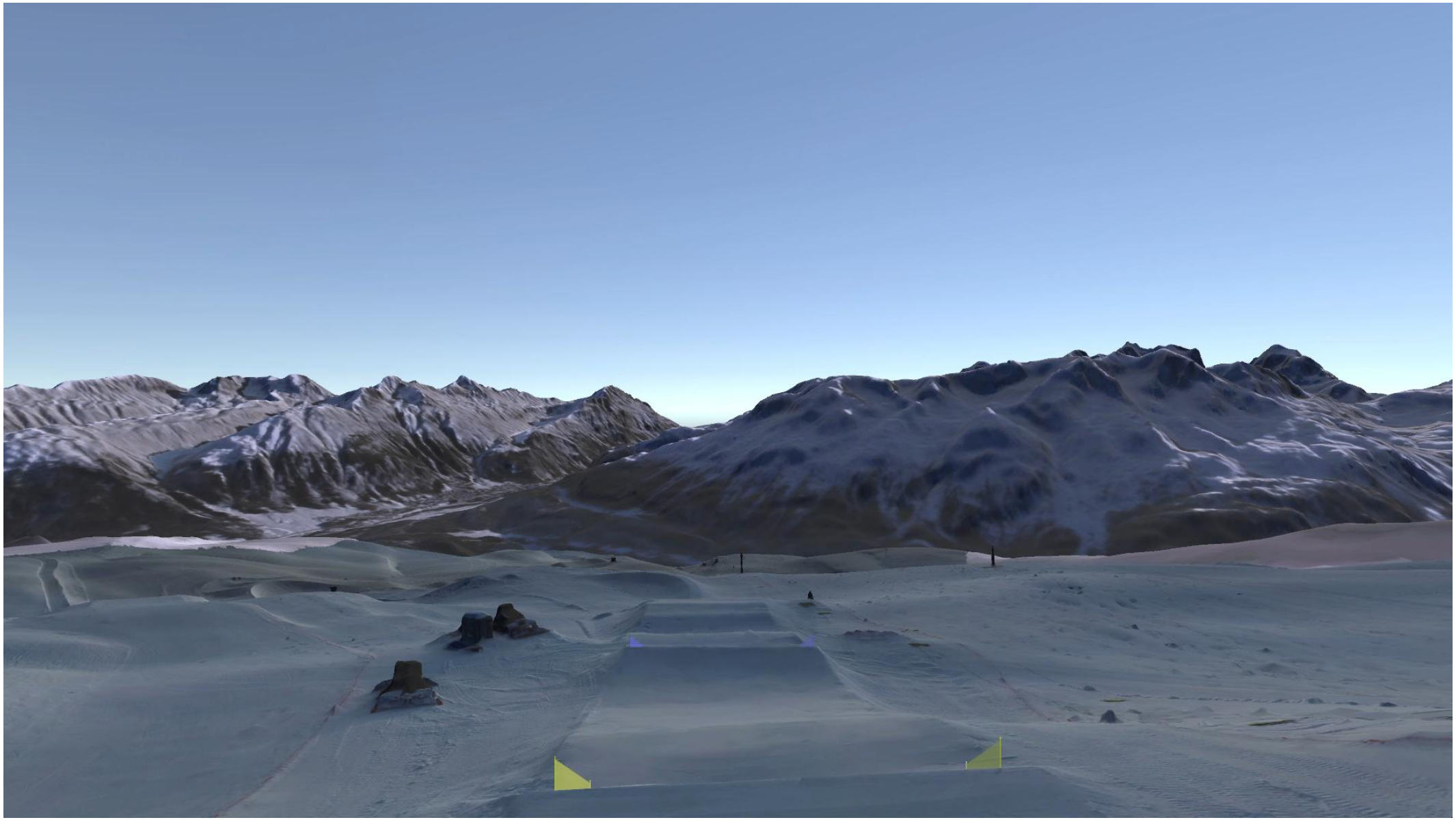

Trajectory profile
Mesh
Trajectory with jumps
Height (H) [m]
2680
2660
2640
2620
2600
2580
2560
2540
Height difference [m]
6
4
2
0
−2
−4
−6
0
100
200
300
400
500
600
700
Curviline distance (s2D) [m]

Visual check with video comparison



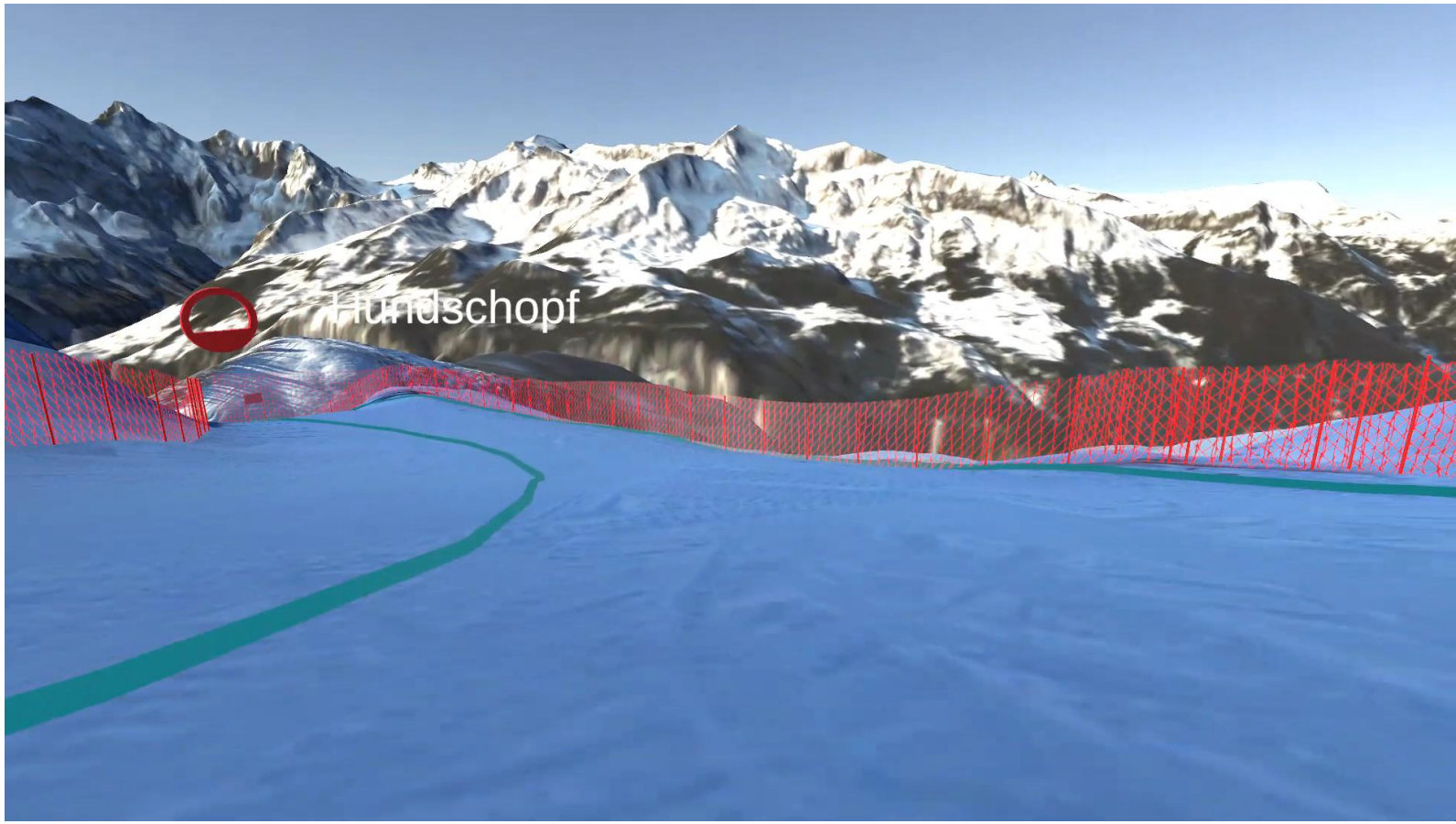
Data : HEIG-VD

Video : Swiss-Ski – Björn Bruhin



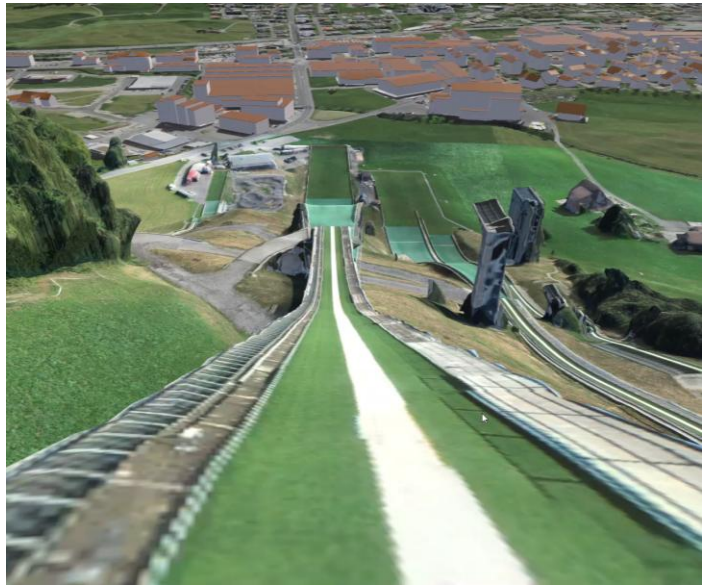
Conclusion

- Useful fusion-algorithm :
  - When trajectory with bad vertical component (GNSS SPP)
  - When no measured trajectory available
- Further usecases :
  - Alpine skiing
  - Ski jumping
  - …



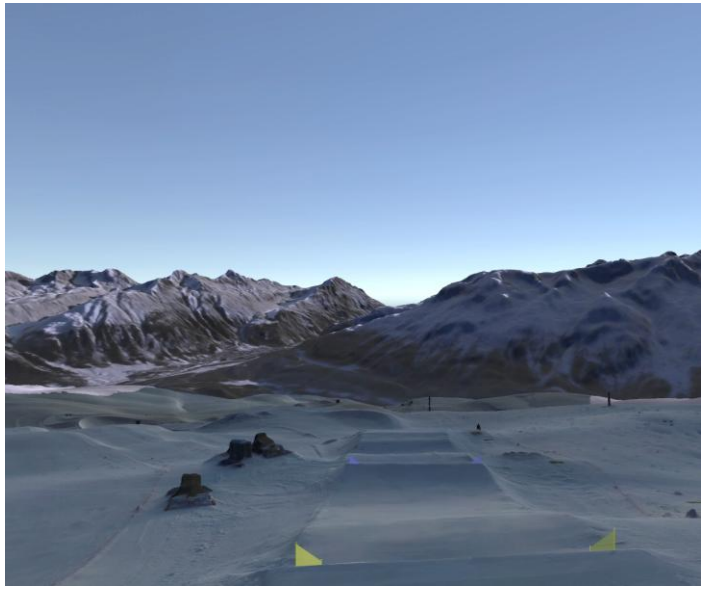
Thank you for your attention!

Einsiedeln



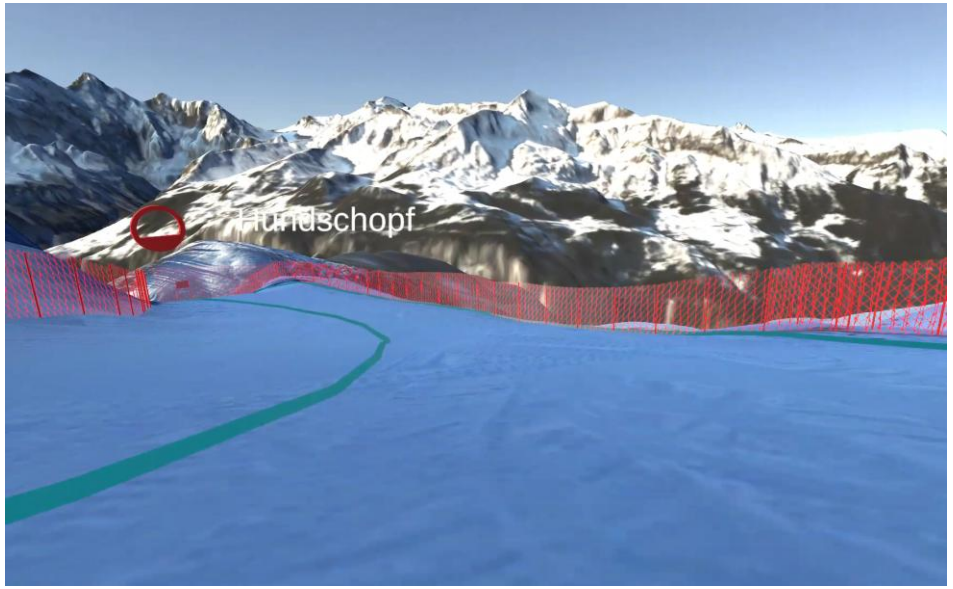
Data : HEIG-VD, swisstopo

St. Moritz



Data : HEIG-VD, swisstopo, Sentinel

Wengen



Data : HEIG-VD, swisstopo, Sentinel

samuel.schwyn@heig-vd.ch
fabien.deleze@heig-vd.ch
sebastien.guillaume@heig-vd.ch
bjoern.bruhin@swiss-ski.ch