SWOT

FPDEM: Algorithm status and plans

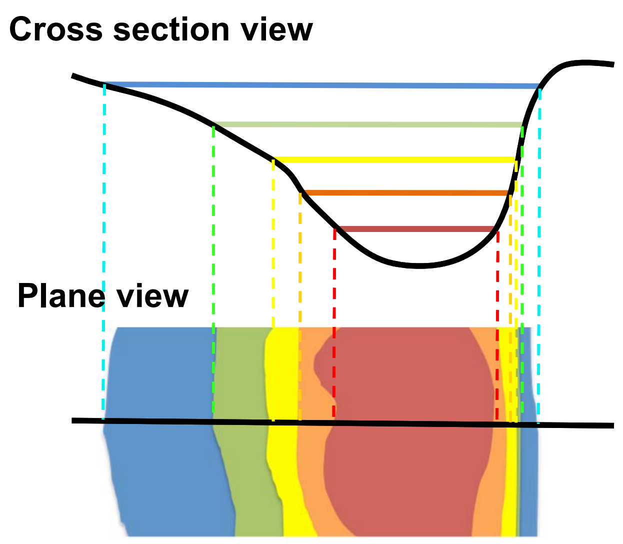
Damien Desroches, Emmanuelle Sarrazin

SWOT ST Meeting, Bordeaux
17-20 June 2019

Introduction

Principle

- Water edge pixel heights ("well done") are used to get the heights of the banks.
- For each date, the water edge pixels of a water body form an iso-elevation curve.
- With enough observations, the floodplain DEM is well sampled between the observed minimum and maximum water height and extent.



Produced after a sufficient accumulation of SWOT data (at least one year)

Format not defined yet:

- Raster, 3D DEM ?
- Possibility to include intermediate products as polygons

Algorithm

Data needed to test and validate

- Synthetic cases:
  - Create synthetic DEM
- Real cases:
  - Required simulations with various water heights

Input data

- Pixel cloud
  - Requested fields: ['classification', 'pixel_area', 'longitude', 'latitude', 'height', 'range_index', 'azimuth_index']
- PixcVec
  - Requested fields: ['range_index‘, 'azimuth_index‘, 'longitude_vectorproc', 'latitude_vectorproc', ’height_vectorproc']
- TBD : Introduce error and flag fields from pixel cloud product to select only “good pixels”

Algorithm (detailed version in back-up slides)

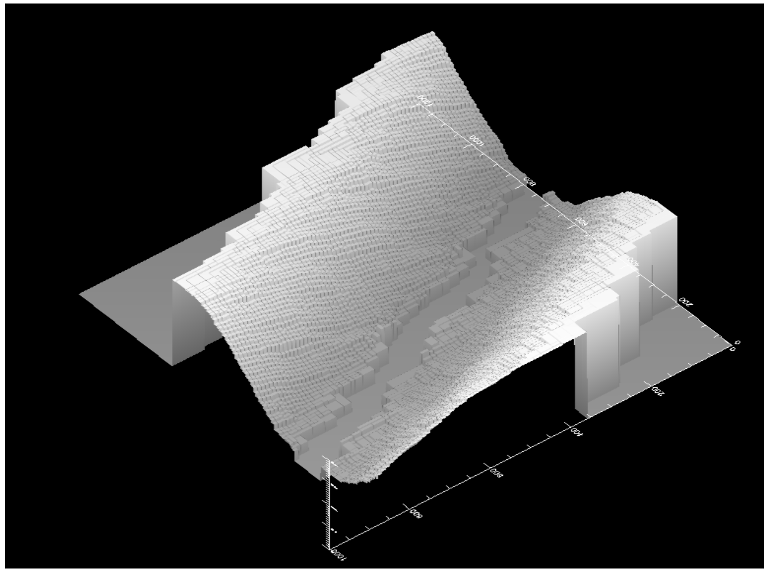
Step 1: Boundary extraction for each date

- Identify the polygon in utm coordinates that best captures a region with the alpha shape approach

➔ Provide a polygon for both lake and river (could be deliver as a product when floodplain DEM will be produced)

Step 2

- Aggregate results on all dates to get a global point cloud
- Two possible products:
  - Raster product => Raster creation
  - Mesh product => Surface reconstruction

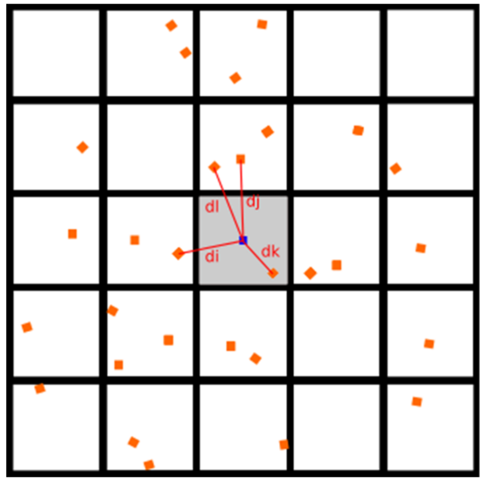


Vintage representation of flood-plain DEM (synthetic DEM as input)

Algorithms

Step 2: Raster creation

- Produce a raster of floodplain DEM height
- The floodplain DEM height is obtained by averaging the edge pixel heights from the different acquisitions that fall within the cell.
- To avoid holes in the DEM raster, the cell height is interpolated from pixels in the neighbor cells. The height of a cell is computed by weighting the pixel heights within a neighborhood based on their distance to the center of the interpolated cell.
- Implemented by using plyflatten (https://github.com/MISS3D/s2p.git)



Height interpolation in the DEM grid

Algorithms

Step 2: Surface reconstruction

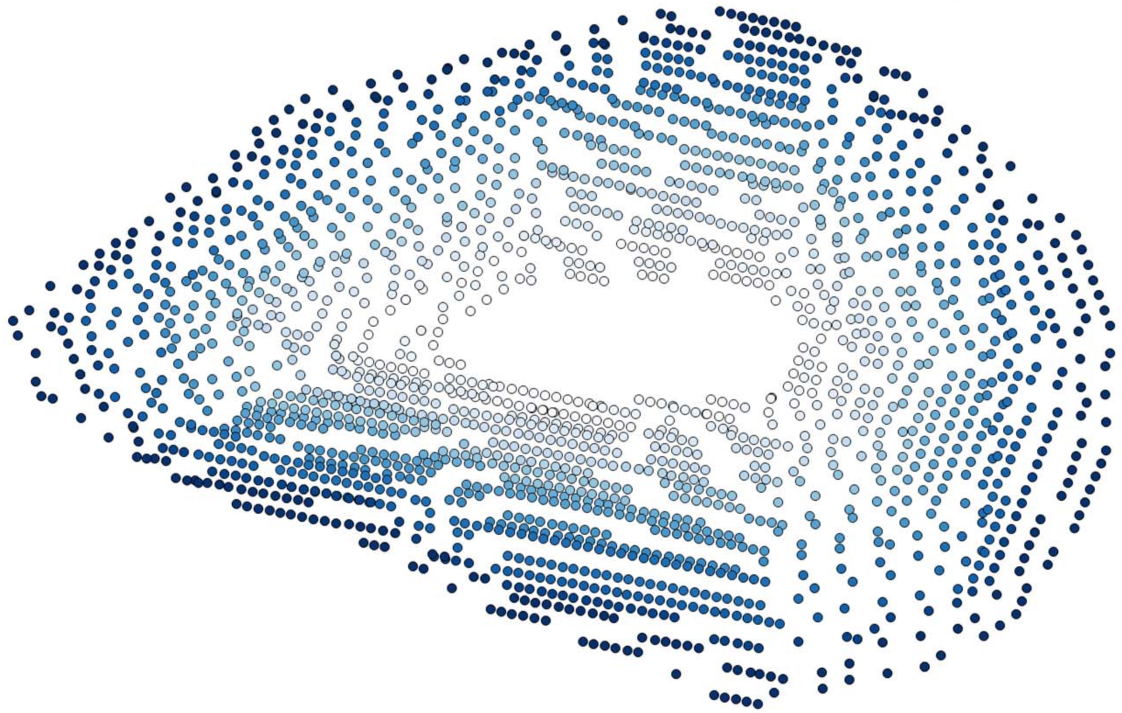
- Work in progress (not implemented yet)
- Requirements:
  - Filter point cloud
    - Remove outlier
    - Denoising point cloud
    - Remove redundant points
  - Compute normal vectors

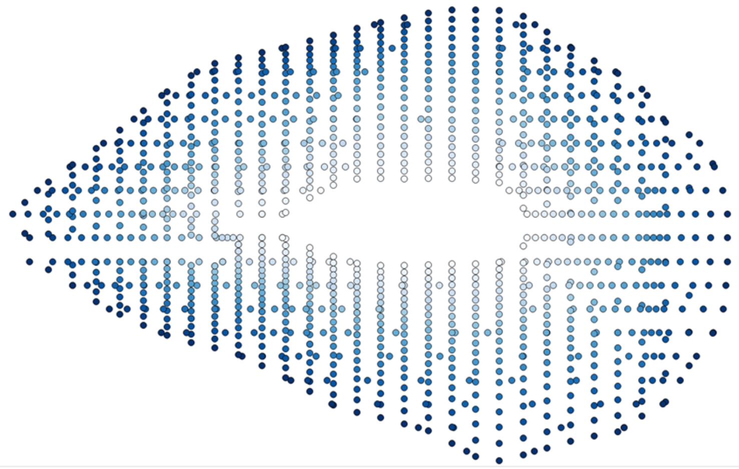
- Possible method for surface reconstruction: Poisson method

Results

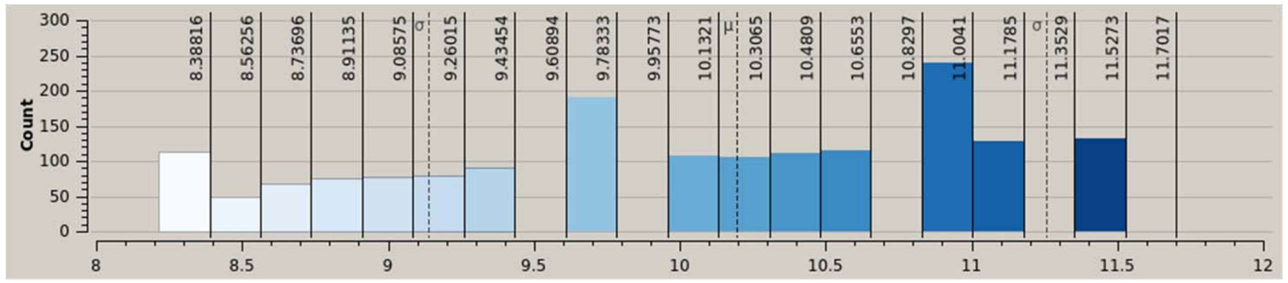
Simple lake with smooth banks

- With SWOT Hydrology toolbox
- 21 dates





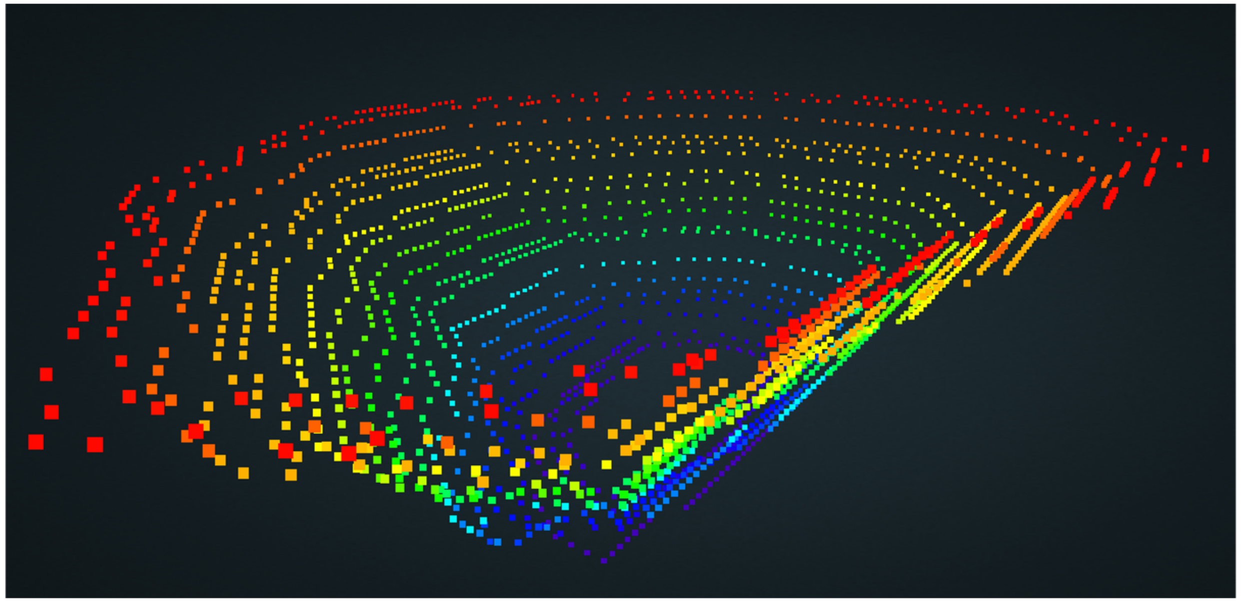
Ground truth



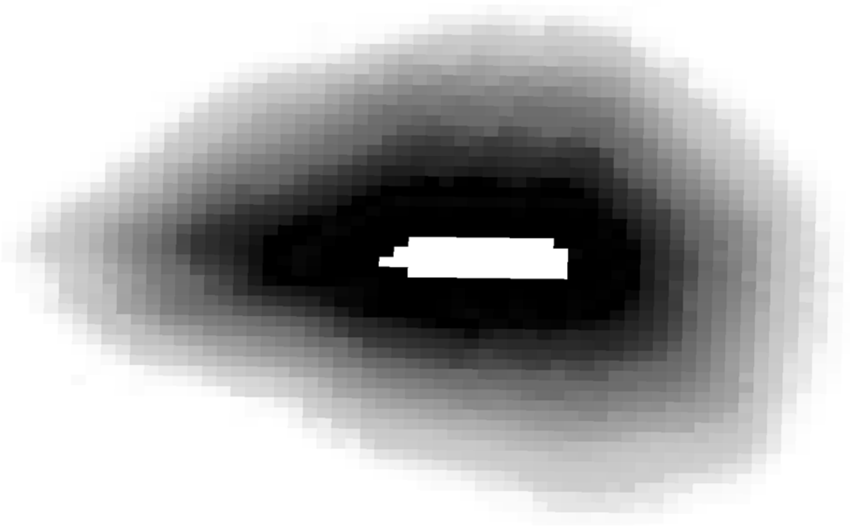
Results

Simple lake with smooth banks

- Results



3D point cloud visualization (Potree)

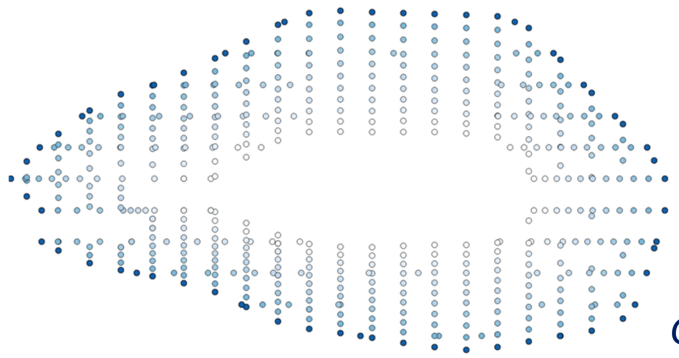


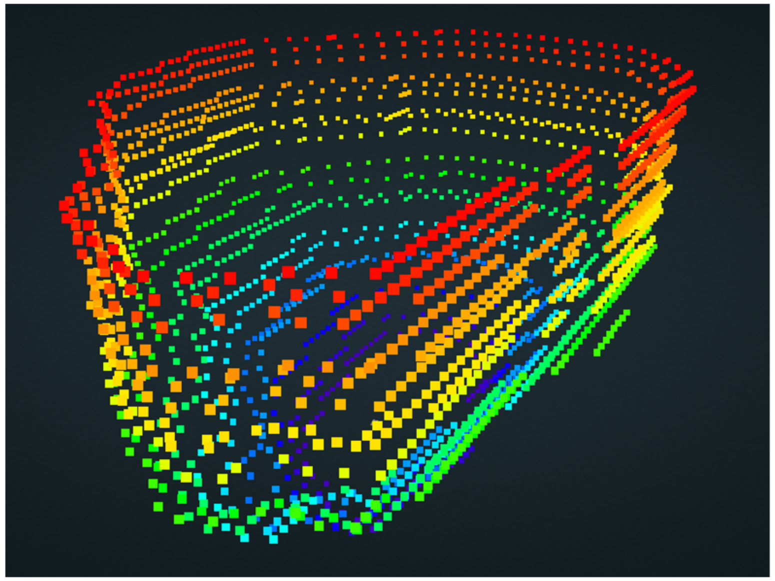
Raster product

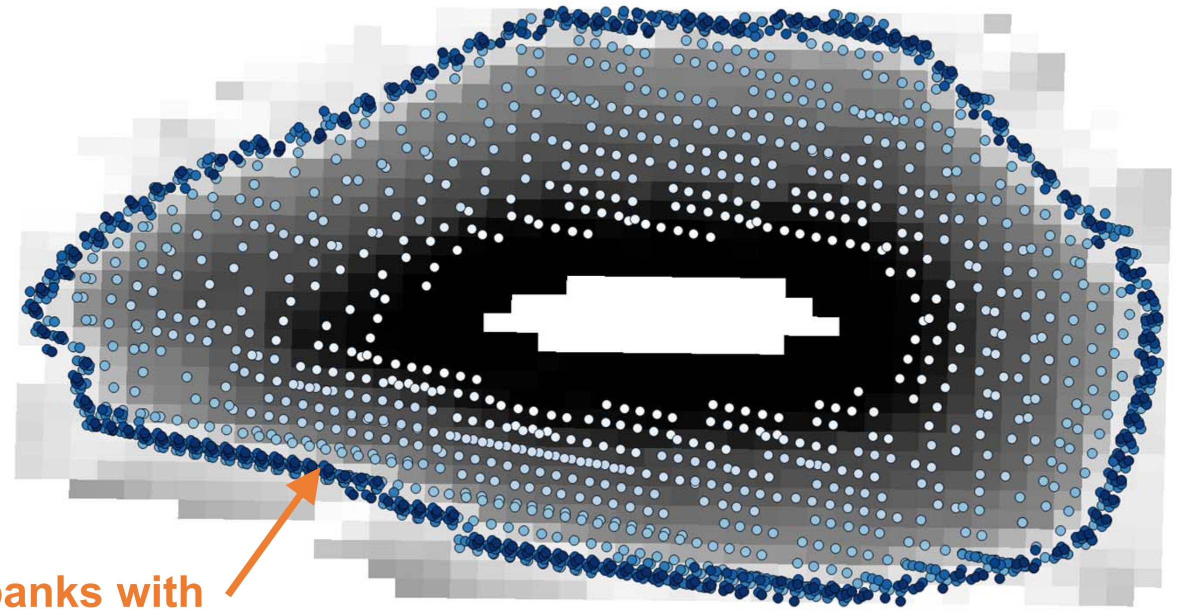
Results

Simple lake with steep banks

- With SWOT Hydrology toolbox


Ground truth



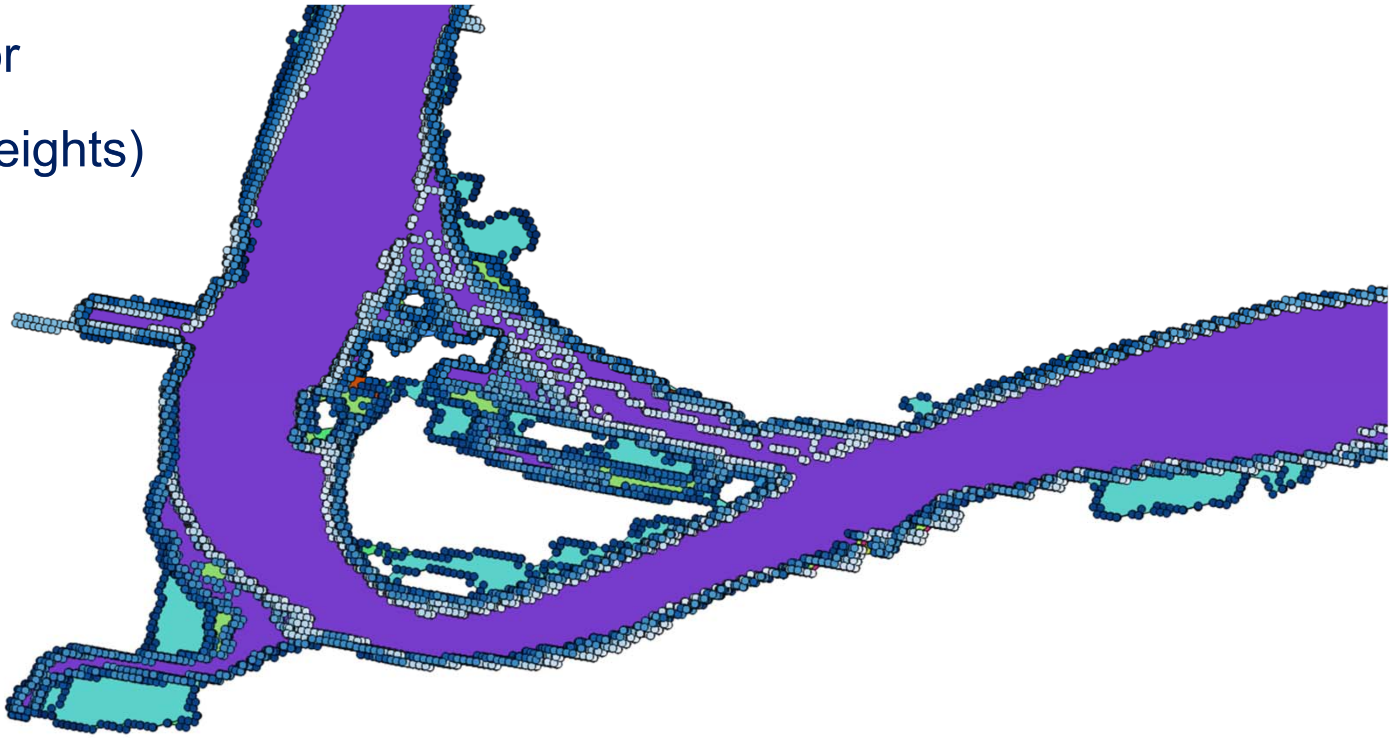


Height errors appear for steep banks with a raster product

Results

Po river

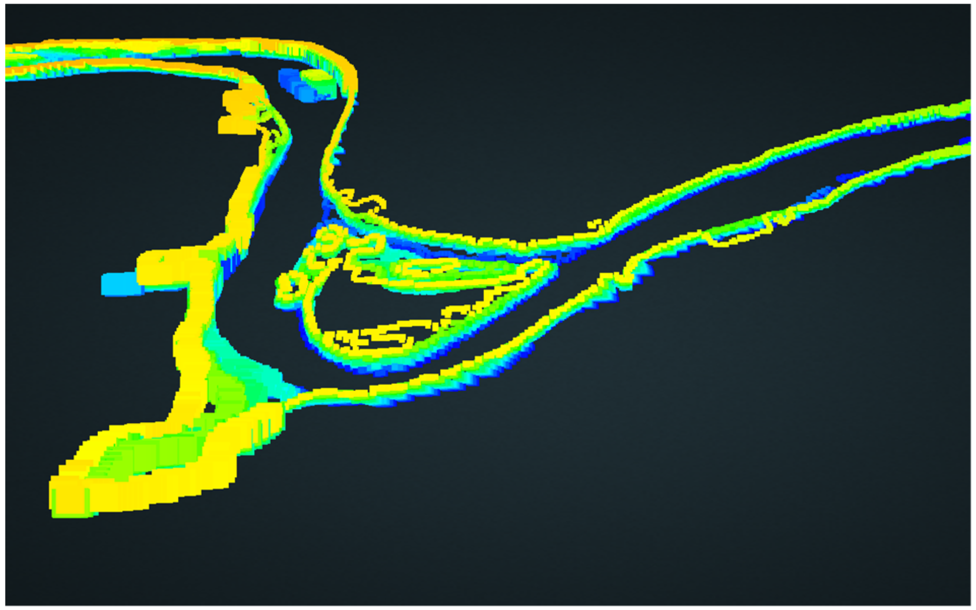
- SWOT HR simulator
- 36 dates (various heights)

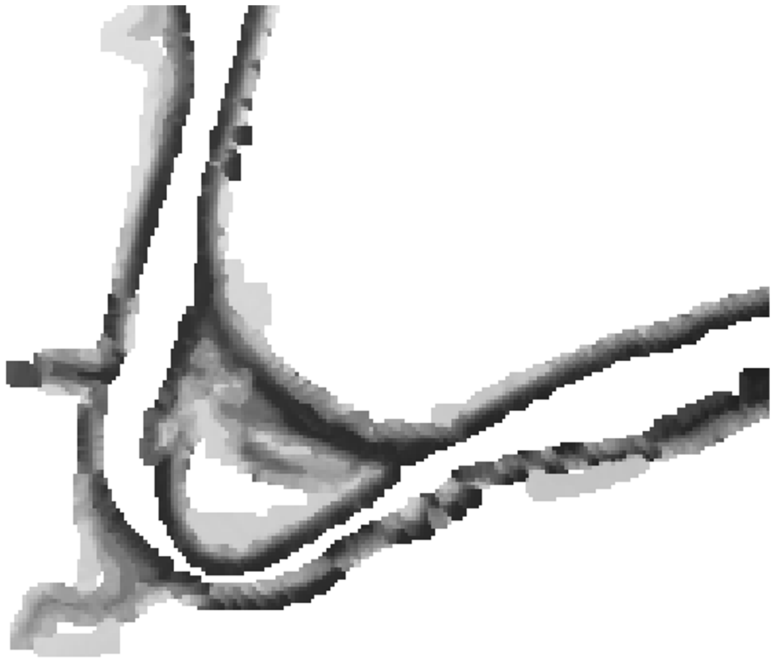


Results

Po river

- SWOT HR simulator
- 36 dates


3D point cloud visualization (Potree)


Raster product

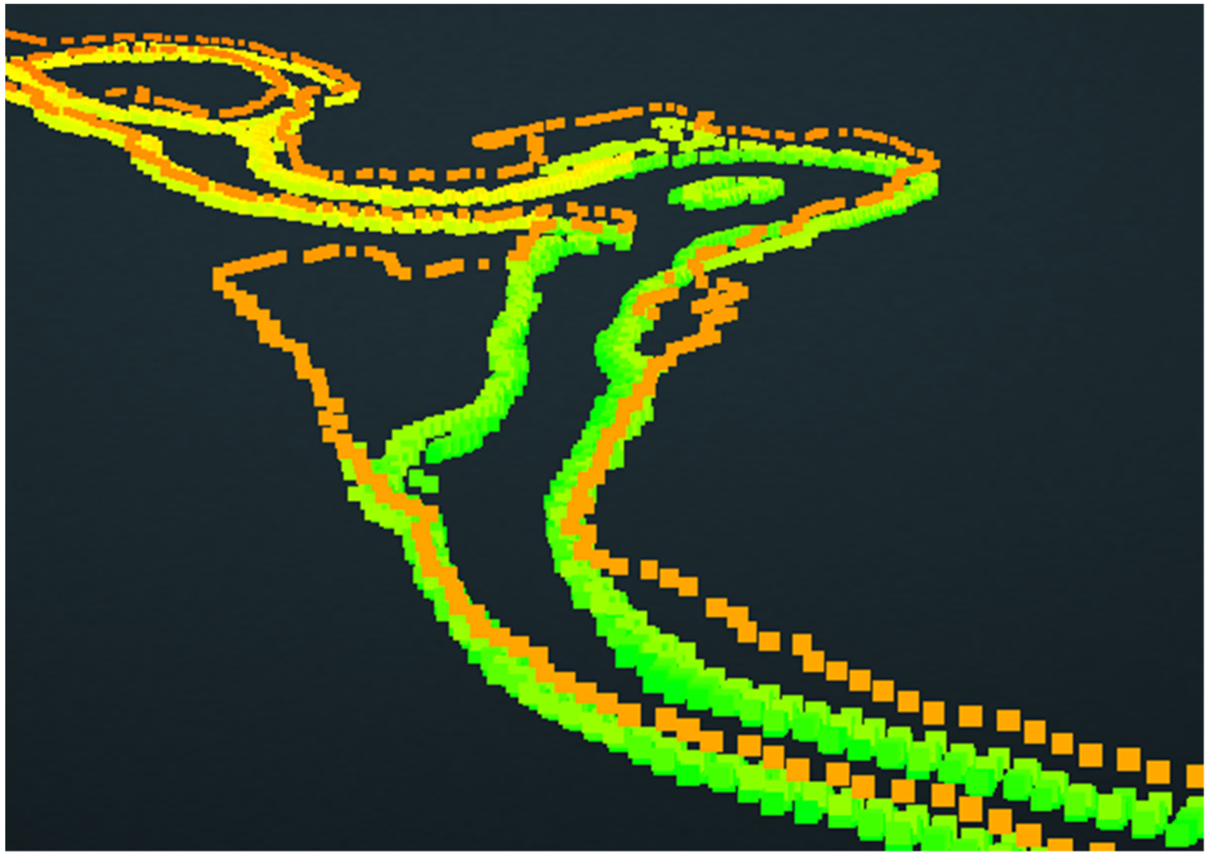
Results

Sacramento

- SWOT HR simulator
- 14 dates

- Not enough height variation

=> Difficult to produce the floodplain DEM

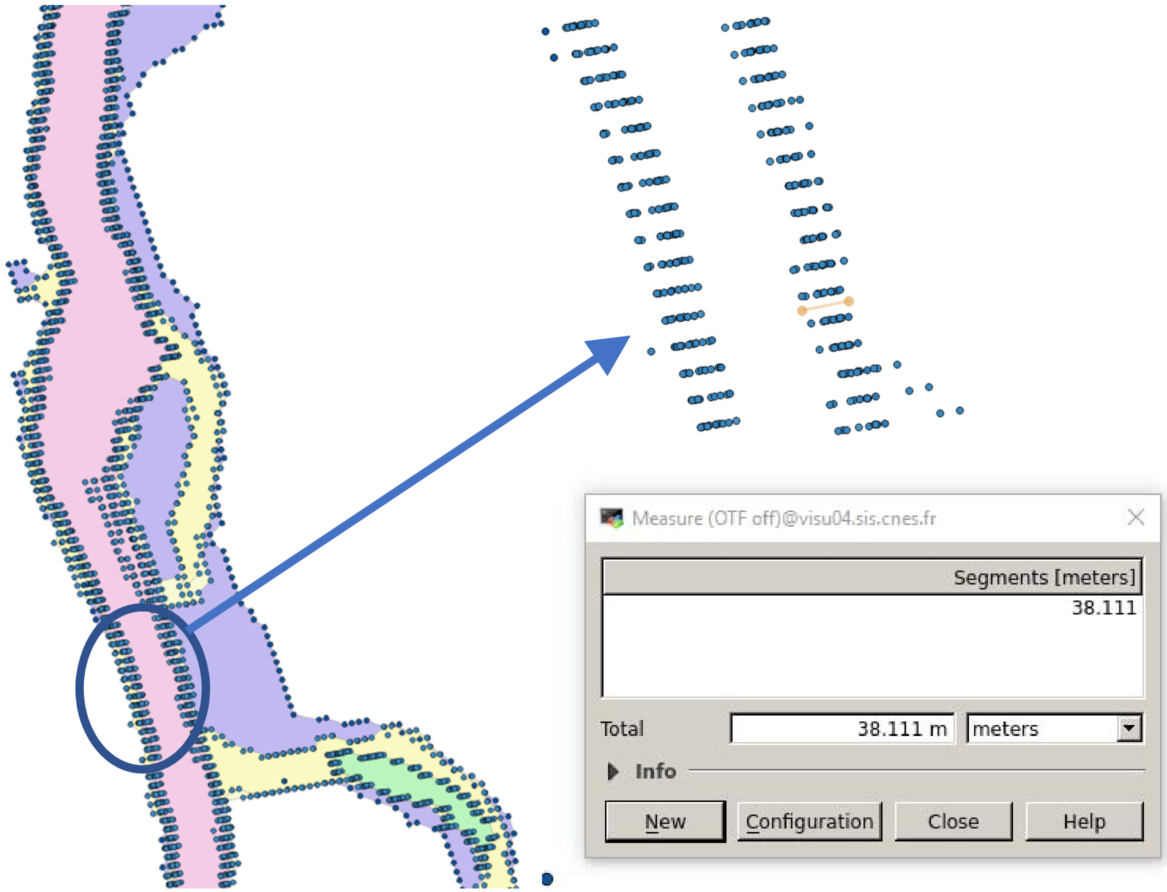


Results

Sacramento

- SWOT HR simulator
- 14 dates

=> About 30 m of shift variation for the boundary polygons



Measure (OTF off)@visu04.sis.cnes.fr

Segments [meters]
38.111

Total 38.111 m meters

Summary

Conclusions

- Need to have various heights to correctly extract floodplain DEM raster
  - Can be very sensitive to heights/areas errors (water detection, phase unwrapping, etc.)
  - Need to produce a quality flag (or uncertainties…) to identify where floodplain DEM makes sense
- Use boundary extraction for each date, each water body
  - Polygon product available for river in floodplain DEM product
- Prototype can handle large pixel clouds (date by date)
- Raster products are unable to catch some details (for example steep banks) => Mesh products ?

Summary

Perspectives

- Some possible improvements
  - Use of pixel flagging and complementary approaches to filter errors
  - Improve reconstruction of steep banks
  - Introduce adaptive grid into raster product (quadtree)
- Implement surface reconstruction algorithm
- Incorporate computed floodplain DEM in a global DEM like SRTM ?
- Produce stress tests using “large flooding” dataset using synthetic DEM generator

- Prototype code will be distributed as part of the SWOT Hydrology Toolbox (Open Source)
  - https://github.com/CNES/swot-hydrology-toolbox

Questions

(Forbidden question : How do you compute uncertainty and quality flag for floodplain DEM ?)

BACKUP

Algorithm

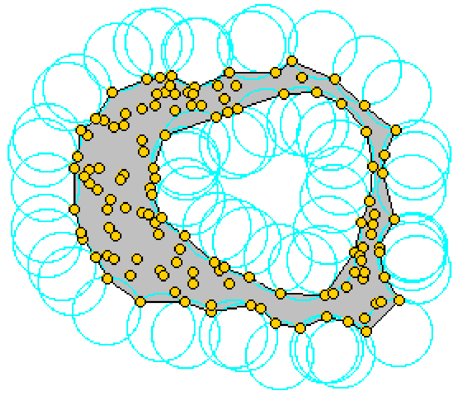
Step 1: Boundary extraction for each date

- Filter water pixels : 3 (Water near land), 4 (Water), 23 (Dark water near land) et 24 (Dark water)
- Convert to a water binary mask in SAR geometry
- Use connected-component labeling algorithm to detect connected regions and label them.
- Compute region areas and remove small regions (area threshold value to defined)
- Convert to utm coordinates
- Remove isolated points
  - Use cKDTree to compute the distance between points.
  - For each points compute the number of neighbors at a distance less than distance_min
  - Keep only points that have enough neighbors
- Identify the polygon in utm coordinates that best captures a region with the alpha shape approach
  - Concave polygon

Algorithm

Focus on alpha shape algorithm

- https://doc.cgal.org/latest/Alpha_shapes_2/index.html#Chapter_2D_Alpha_Shapes
  - Imagine a huge mass of ice-cream making up the space R^3 and containing the points as "hard" chocolate pieces. Using one of these sphere-formed ice-cream spoons, we carve out all parts of the ice-cream block we can reach without bumping into chocolate pieces, thereby even carving out holes in the inside (e.g. parts not reachable by simply moving the spoon from the outside). We will eventually end up with a (not necessarily convex) object bounded by caps, arcs and points. If we now straighten all "round" faces to triangles and line segments, we have an intuitive description of what is called the α-shape of S.
  - Alpha shapes depend on a parameter α. In the ice-cream analogy above, α is the squared radius of the carving spoon. A very small value will allow us to eat up all of the ice-cream except the chocolate points themselves. On the other hand, a huge value of α will prevent us even from moving the spoon between two points since it is too large. So we will never spoon up the ice-cream lying in the inside of the convex hull of S. Hence, the alpha shape becomes the convex hull of S as $\alpha \to \infty$.

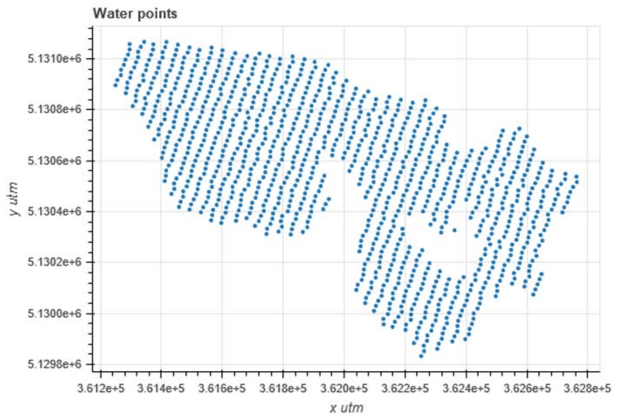


- Today two implementations
  - Full python implementation based on scipy.spatial.Delaunay for triangulation and shapely.ops.cascaded_union to aggregate polygons: Slow
  - CGAL: Very fast but GPL license

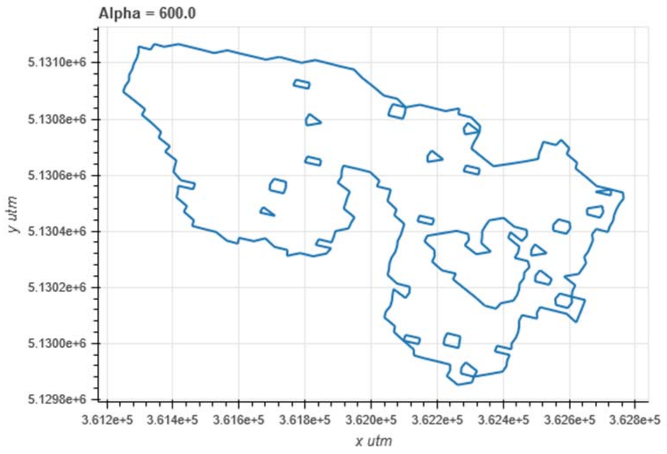
Algorithm

Focus on alpha shape algorithm

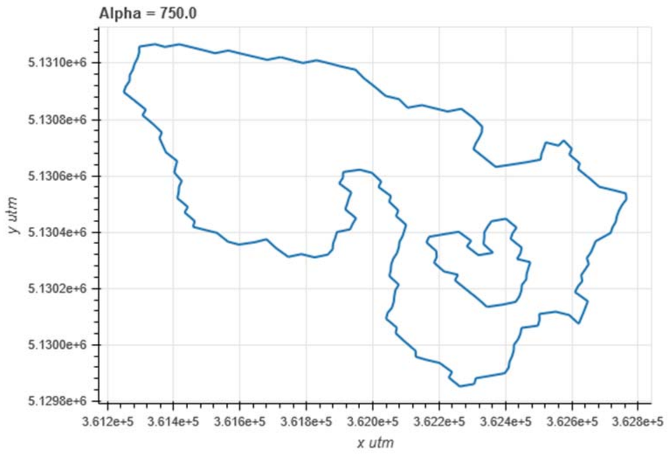
- Today two implementations
  - Full python implementation: Slow
  - CGAL: Very fast but GPL license
- Importance of alpha value



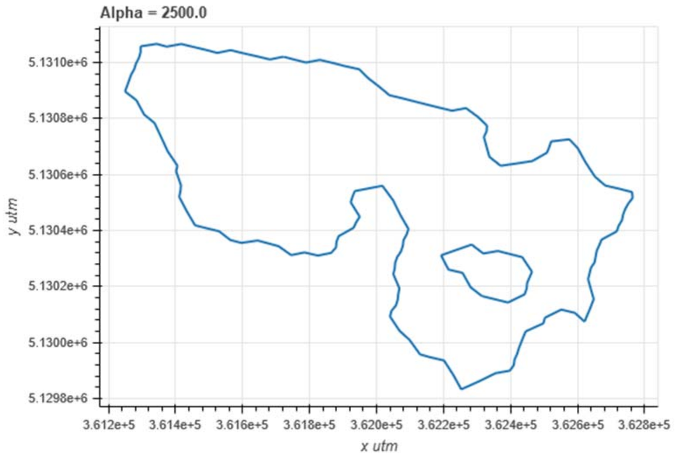
Pixel cloud



α = 600



α = 750

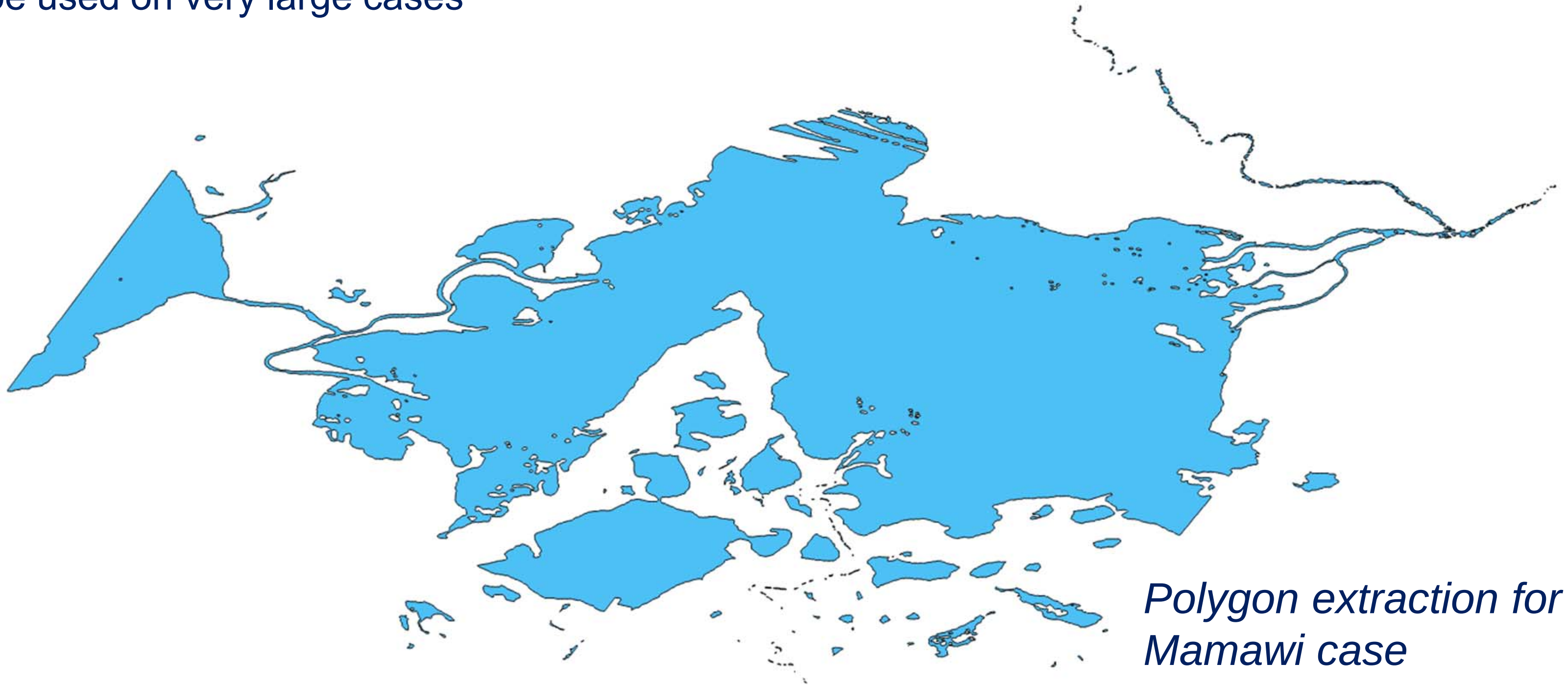


α = 2500

Algorithm

Focus on alpha shape algorithm

- Good performances of CGAL implementation
- Can be used on very large cases



Polygon extraction for Mamawi case