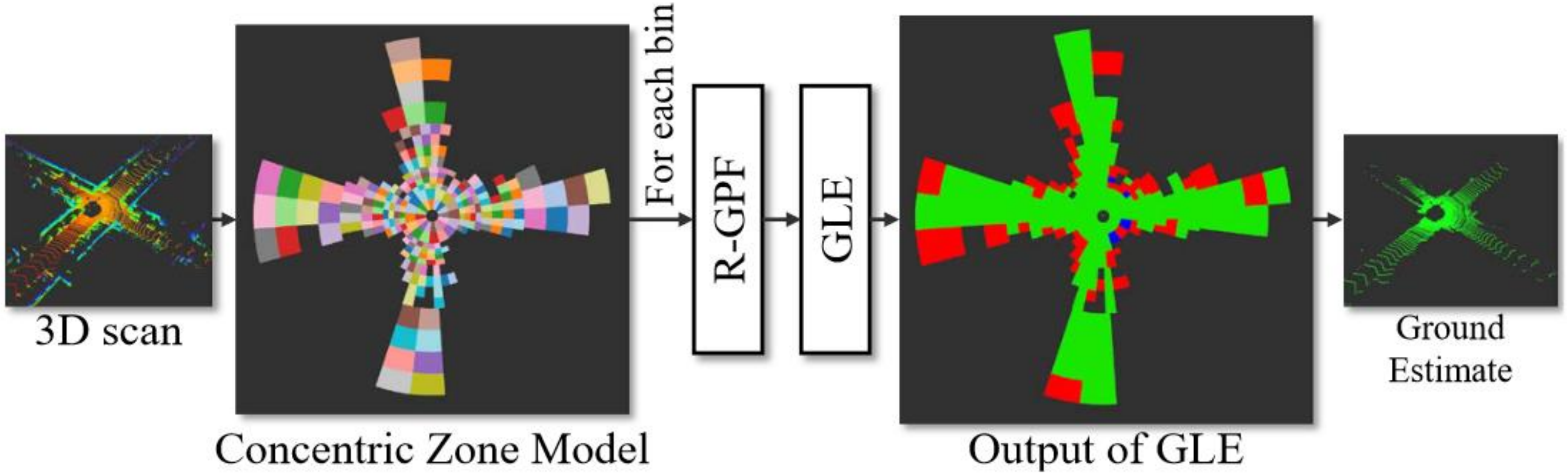
ELTE | FACULTY OF INFORMATICS

3D Point Cloud processing and analysis
Ground Plane Segmentation

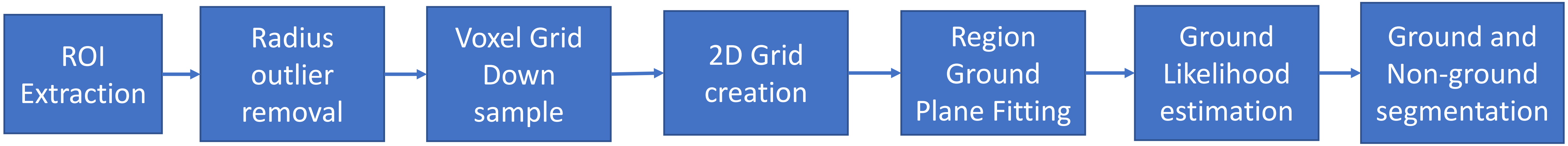
Massinissa Aouragh:
Faculty of Informatics, Department of Artificial Intelligence
Robert Bosch Kft
m2j7au@inf.elte.hu

Patch Work

- [Patchwork: Concentric Zone-based Region-wise Ground Segmentation with Ground Likelihood Estimation Using a 3D LiDAR Sensor]()
- [patchwork github]()



Path work like pipeline



Noise Removal

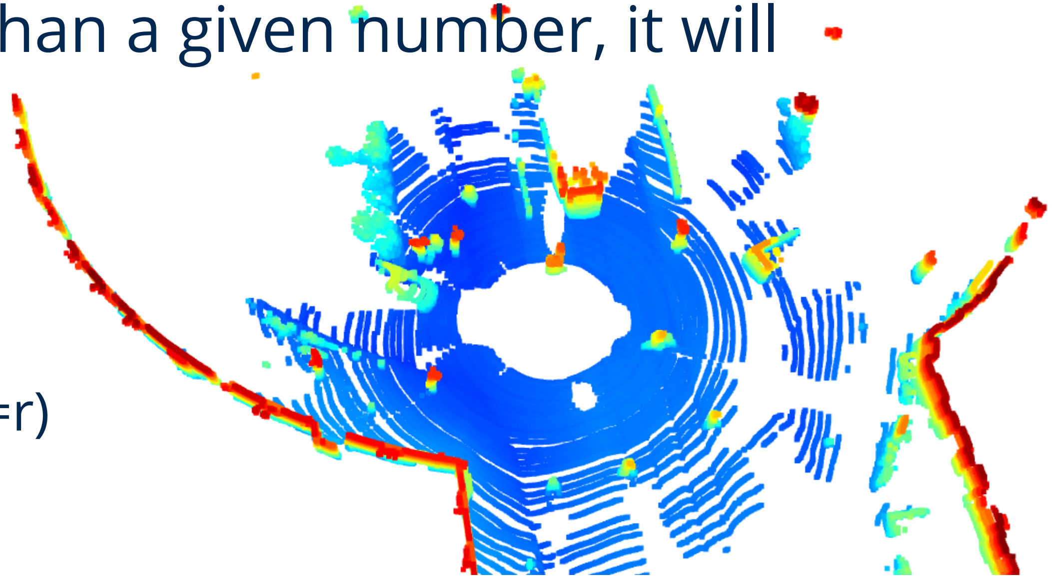
- Radius Outlier Removal:
  - Filters points based on the number of neighbors in a certain radius.
  - For each point we first find it neighbors with a certain radius
  - If the number of the neighbors k is less than a given number, it will be considered as outlier and removed

r = 300 mm

k_min = 10 points

Open 3D:

Pt.remove_radius_outlier(nb_points=k_min, radius=r)



Grid Down sample

- Min_point = Points.min(axis=0)
- Point_to_grid_indices = Points – Min_point / grid_size
- Sorting the Point_to_grid_indices
- Sample unique indices

2D Grid Creation

Given point cloud $p_1, p_2, \ldots, p_n$

$$x_{min} = min(x_1, x_2, \ldots, x_n)$$
$$x_{max} = max(x_1, x_2, \ldots, x_n)$$
$$y_{min} = min(y_1, y_2, \ldots, x_n)$$
$$y_{max} = max(y_1, y_2, \ldots, x_n)$$
$$z_{min} = min(z_1, z_2, \ldots, x_n)$$
$$z_{max} = max(z_1, z_2, \ldots, x_n)$$

$For\ voxel\ of\ size\ r\ we\ have:$

$$N_x = \lfloor \frac{(x_{max} - x_{min})}{r} \rfloor$$
$$N_y = \lfloor \frac{(y_{max} - y_{min})}{r} \rfloor$$
$$N_z = \lfloor \frac{(z_{max} - z_{min})}{r} \rfloor$$

$Voxel\ Index:$

$$i_x = \lfloor \frac{(x - x_{min})}{r} \rfloor$$
$$i_y = \lfloor \frac{(y - y_{min})}{r} \rfloor$$
$$i_z = \lfloor \frac{(z - z_{min})}{r} \rfloor$$
$$i = i_x + i_y * N_x + i_z * N_x * N_y$$

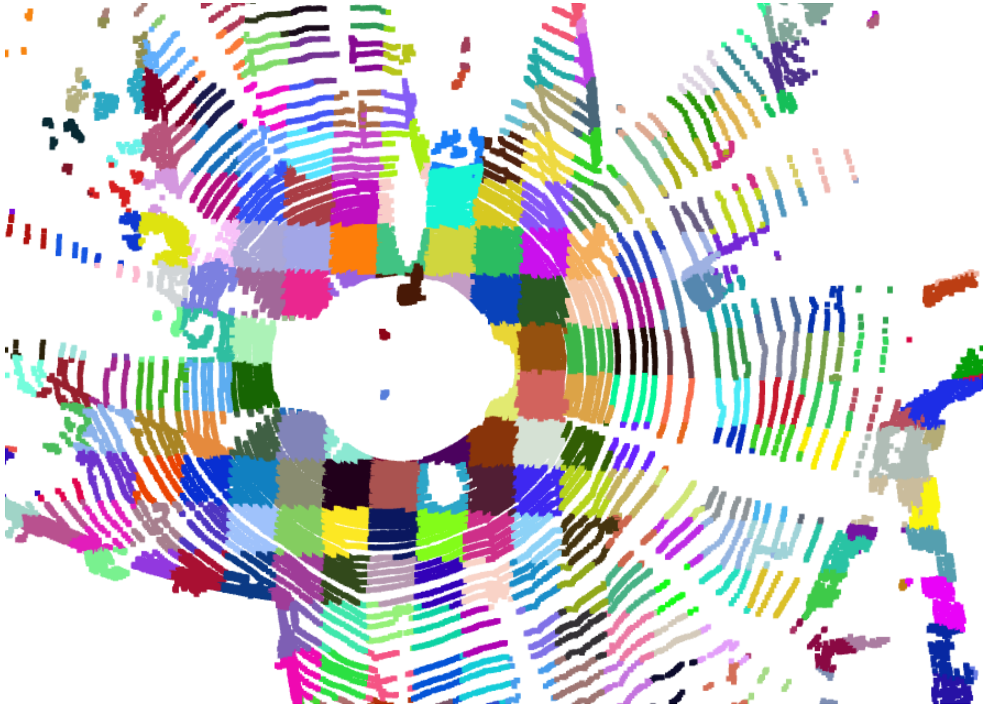
$$x_{min} = min(x_1, x_2, \ldots, x_n)$$
$$x_{max} = max(x_1, x_2, \ldots, x_n)$$
$$y_{min} = min(y_1, y_2, \ldots, x_n)$$
$$y_{max} = max(y_1, y_2, \ldots, x_n)$$

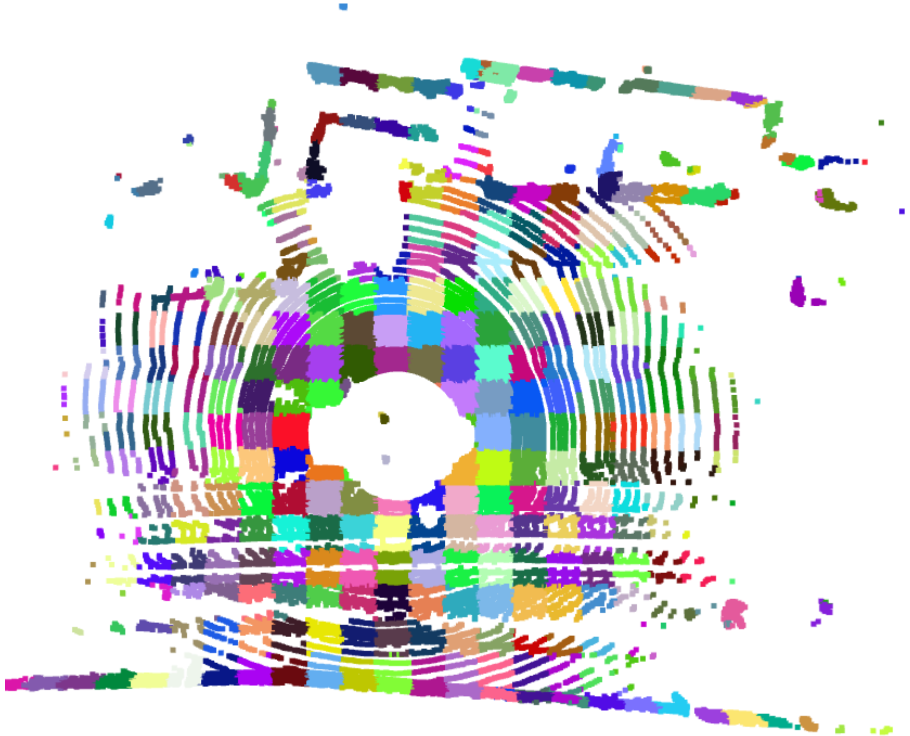
$$N_x = \lfloor \frac{(x_{max} - x_{min})}{r} \rfloor$$
$$N_y = \lfloor \frac{(y_{max} - y_{min})}{r} \rfloor$$

$$i_x = \lfloor \frac{(x - x_{min})}{r} \rfloor$$
$$i_y = \lfloor \frac{(y - y_{min})}{r} \rfloor$$

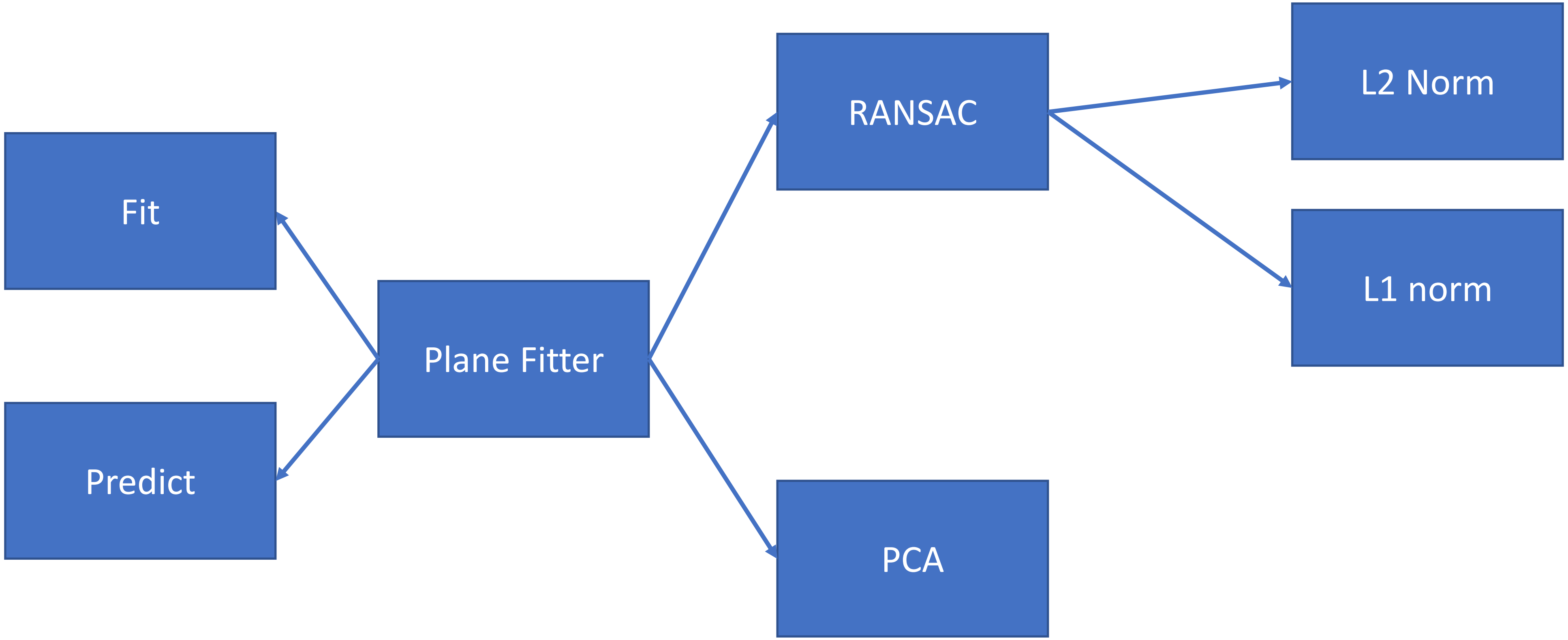
$$i = i_x + i_y * N_x$$

2D Grid

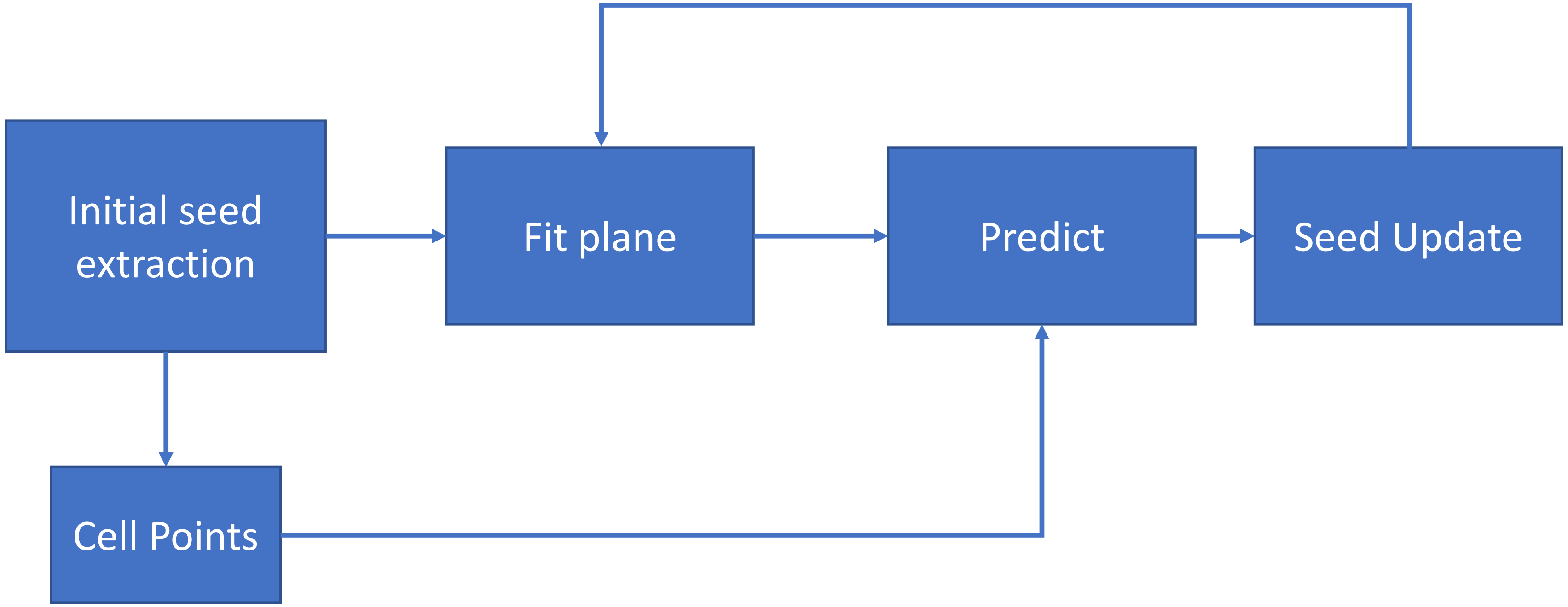




Plane fitter



PCA Plane Extraction



Initial seed selection

- Sort cell points based on height (z)
- Take first k points (min ~30)
- Calculate mean height of the first k points
- Choose points that satisfy Z < mean Height + distance threshold (100~120 mm)

PCA fitting

- Compute mean and covariance of the seed points
- Compute the eigenvalues and eigenvectors of the covariance matrix
- The eigenvector that corresponds to the smallest eigenvalue represents the surface normal
- Normalize the normal vector (Note that the z attribute need to be positive n[2] = abs(n[2]))
- Use the mean to calculate the **d coefficient** (ax + by + cz + **d** =0) **d = -n . mean**

PCA Fitting

- Calculate point to plane distances in the cell using the estimated plane normal and d coefficient
- Calculate the mean squared distances of the points as model error
- Update the seed points with points that satisfy distance < distance threshold
- Choose model with the smallest error

Plane estimation

Uprightness:

$the\ angle\ \theta\ between\ the\ normal\ and\ the\ refrence\ vector\ z = [0,0,1]$

$n\ .\ z = ||n|| * ||z|| * \cos(\theta)$

$$\cos(\theta) = \frac{n\ .z}{||n|| * ||z||}$$

$||n|| = ||z|| = 1 \rightarrow n.z = 0 * a + 0 * b + 1 * c$

$\cos(\theta) = C$, C needs to be as close to 1 as possible (min ~0.85)

Plane estimation

Flatness:

the proportion of the variance by the smallest eigenvalue

$$flatness = \frac{\lambda_1}{\sum \lambda\_i}$$

Since the flatness mean the proportion of the variance for a flat plane it needs to be small (~0.0005)

Plane estimation

Height condition:

Mean height in cell < height threshold

Final conditions:

(Predicted distances < threshold)

and

Uprightness

and

(Flatness or **(Mean height in cell < height threshold))**