Vario-scale visualization of the AHN2 point cloud

P5 presentation

Student:	Jippe van der Maaden
Main mentor:	Prof.dr.ir. P.J.M. van Oosterom
Second mentor:	Dr.ir. B.M. Meijers
Examiner:	Dr. H.M.H. van der Heijden

TUDelft

What is a point cloud?

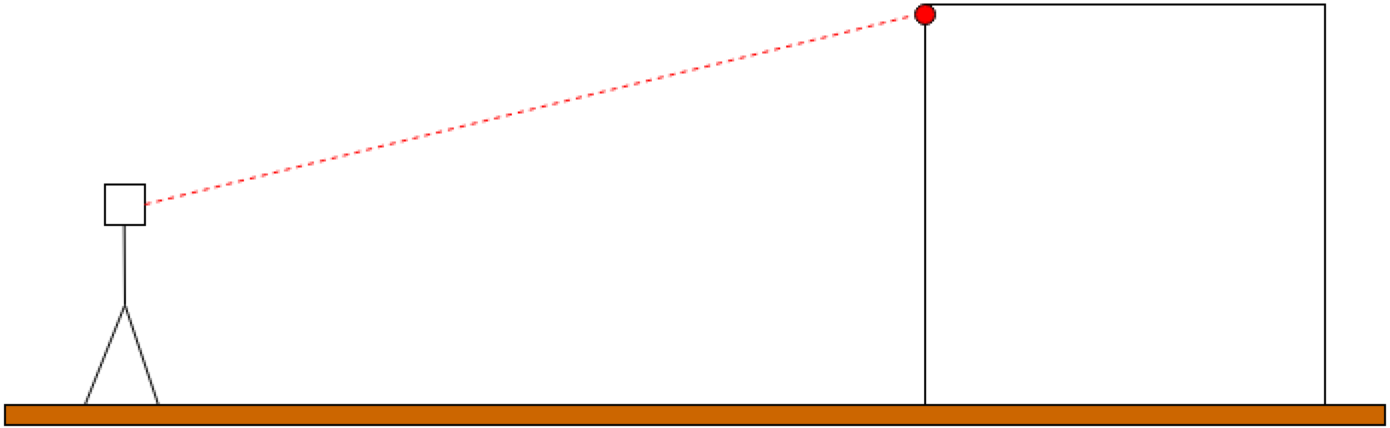
What is a point cloud?

- Collection of geo-referenced points

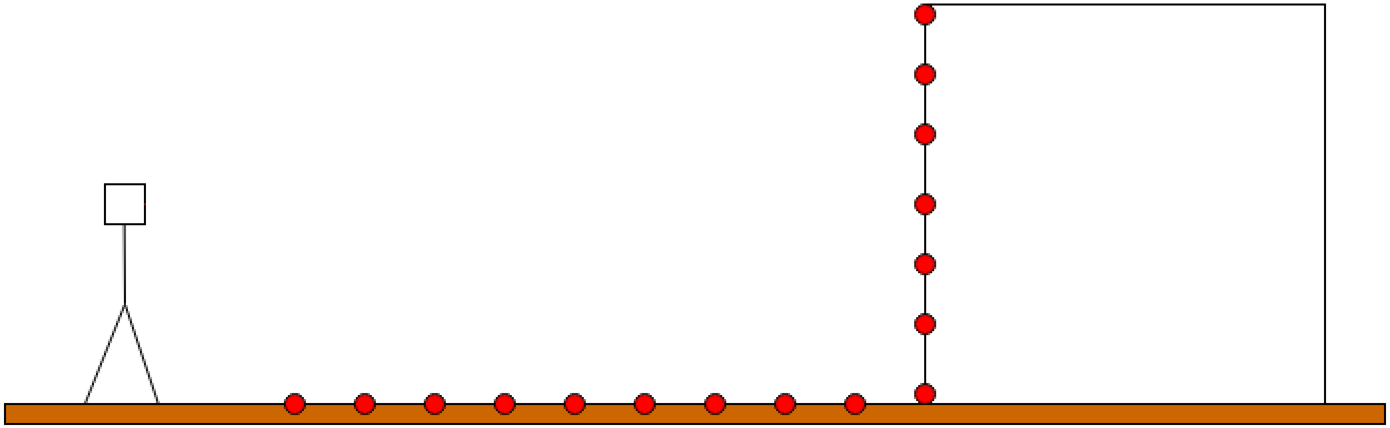
What is a point cloud?

- Collection of geo-referenced points
- Collected using LiDAR (Light Detection and Ranging)

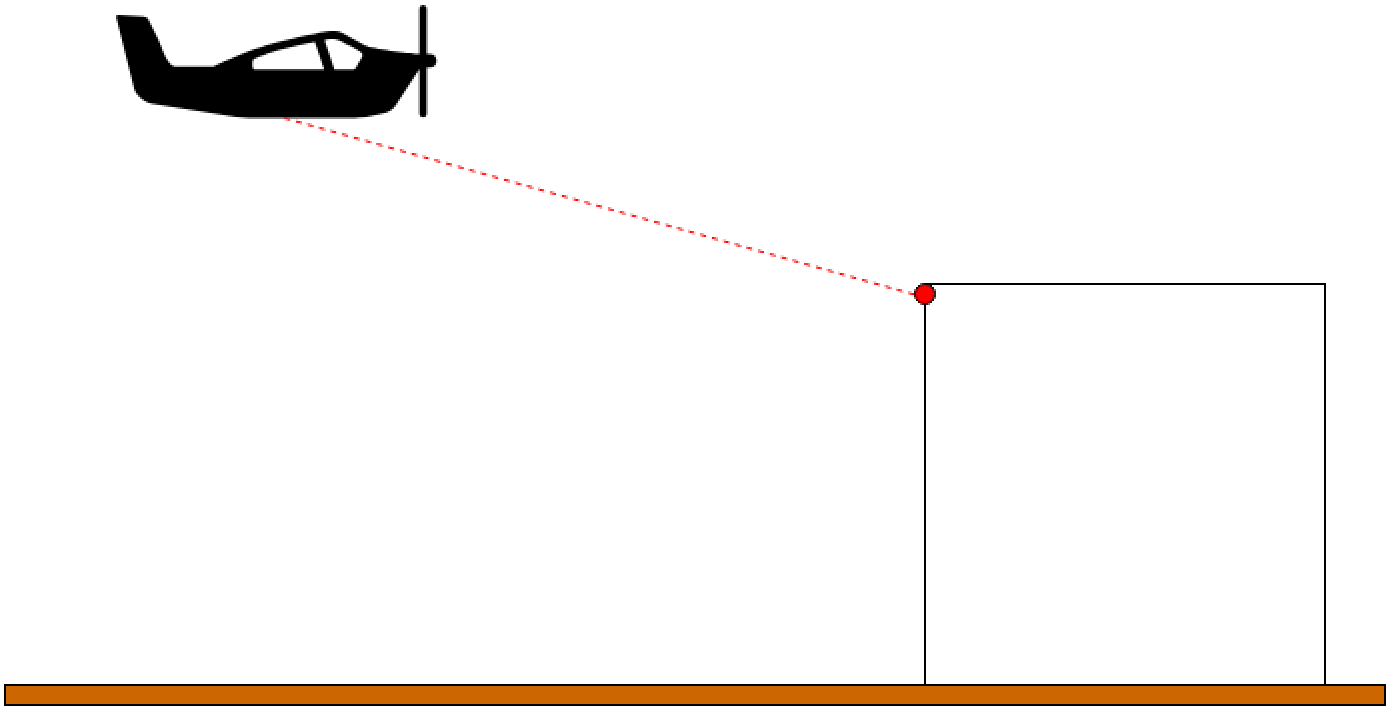
What is a point cloud?



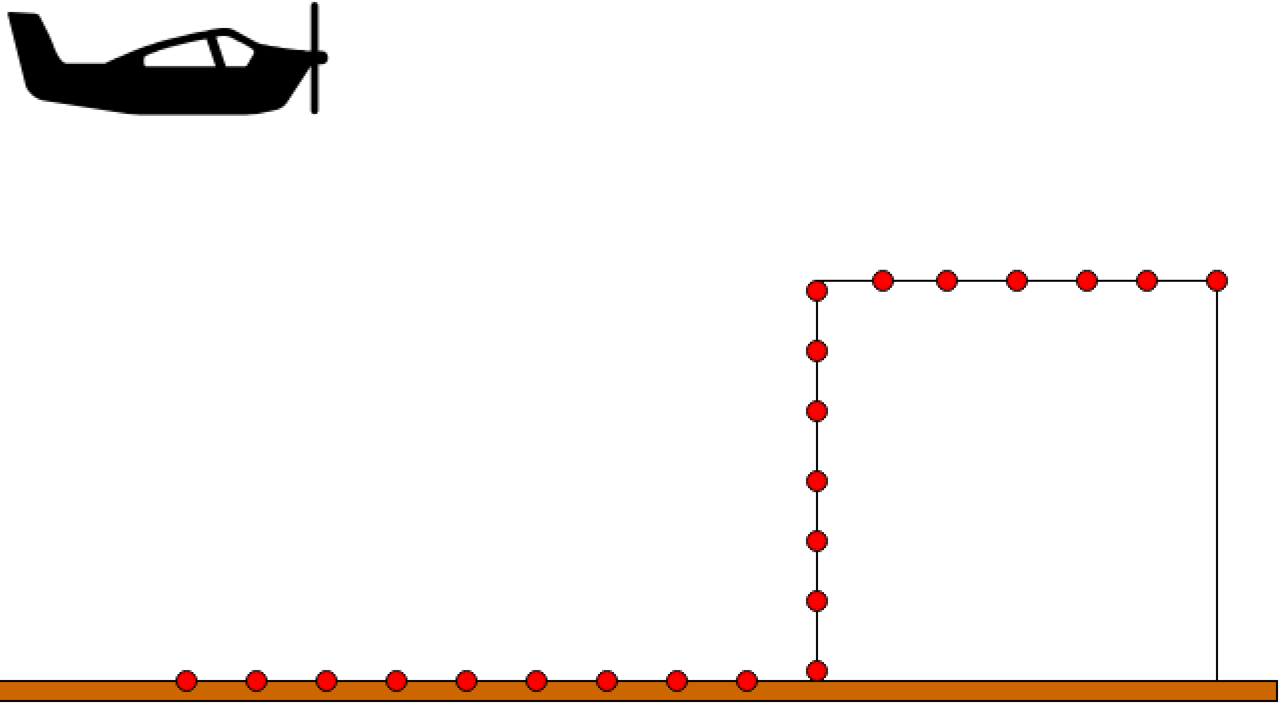
What is a point cloud?



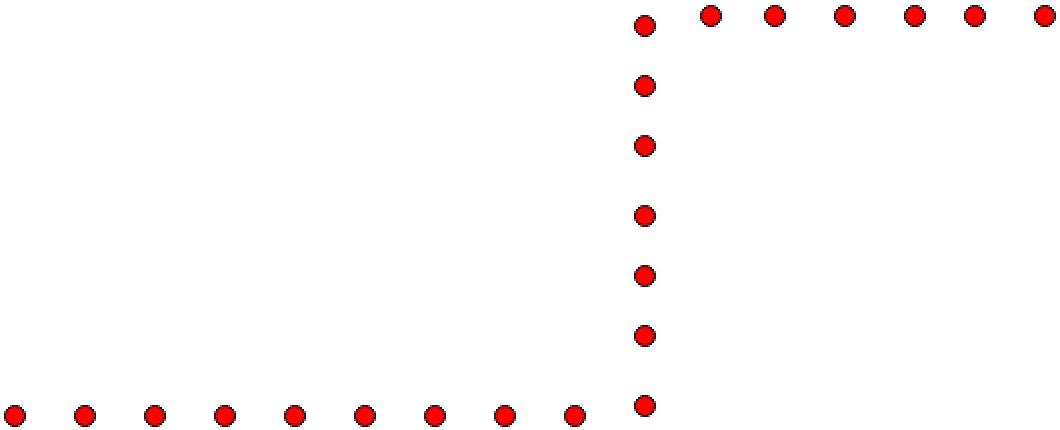
What is a point cloud?



What is a point cloud?



What is a point cloud?



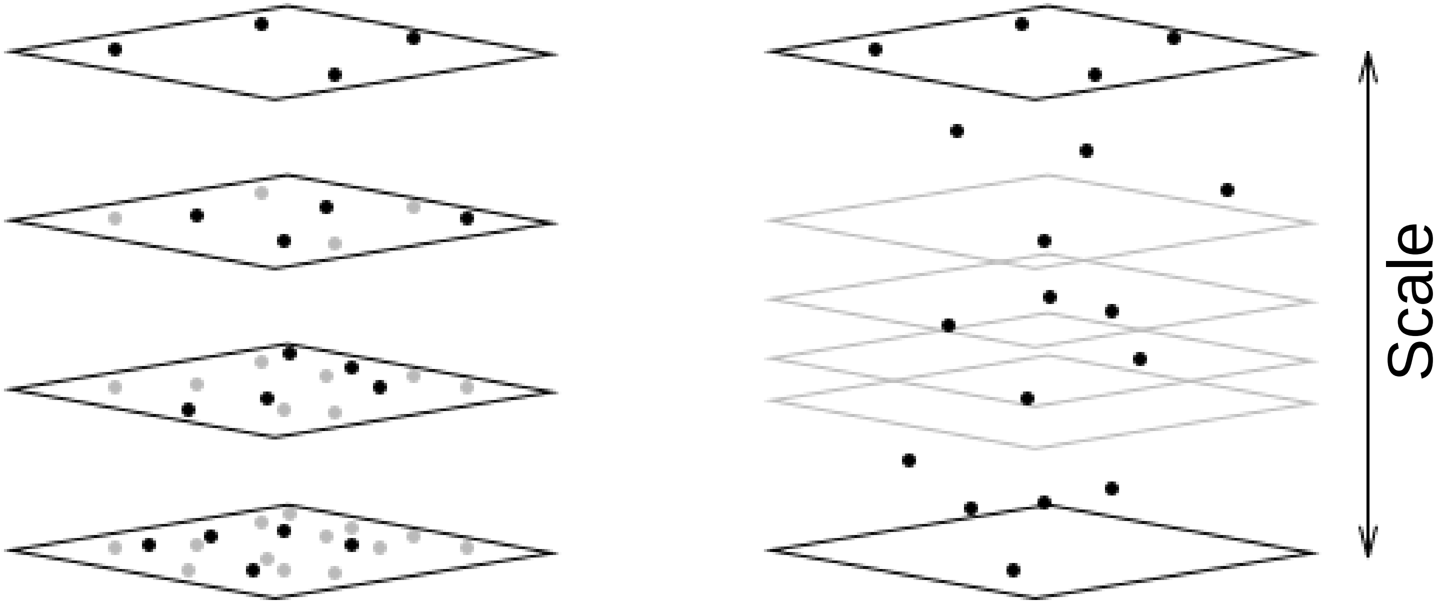
Thesis goal

- From discrete visualisation of massive point clouds, to the continious visualization of massive point clouds.

TU Delft

What is vario-scale visualization?

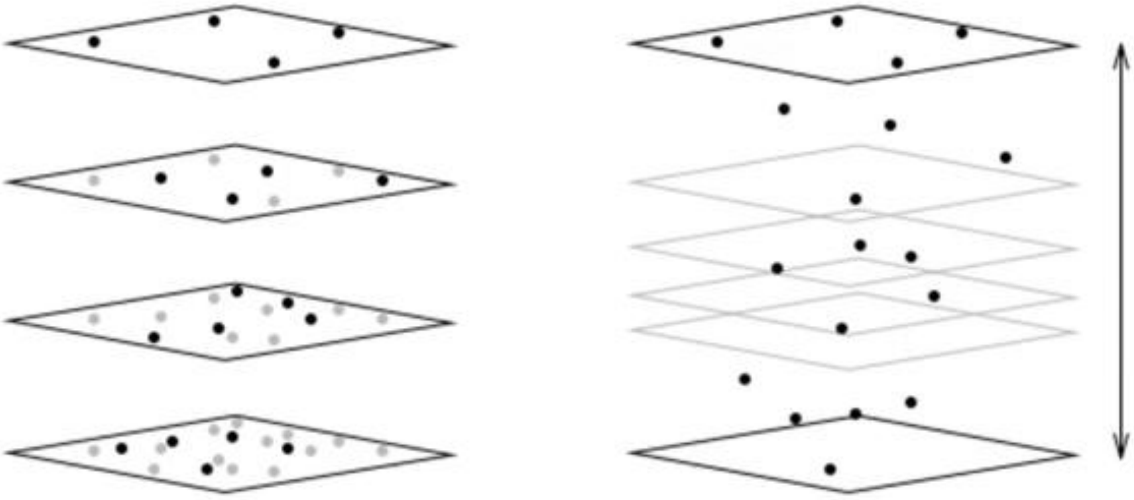
Vario-scale visualization



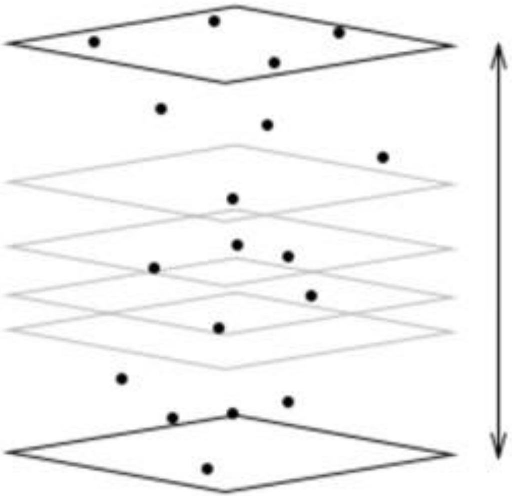
Discrete

Continuous

Discrete visualization

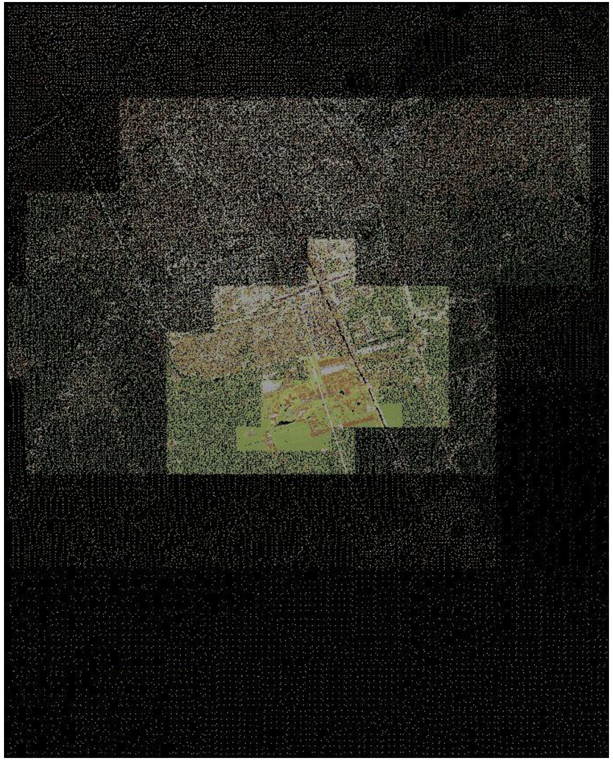


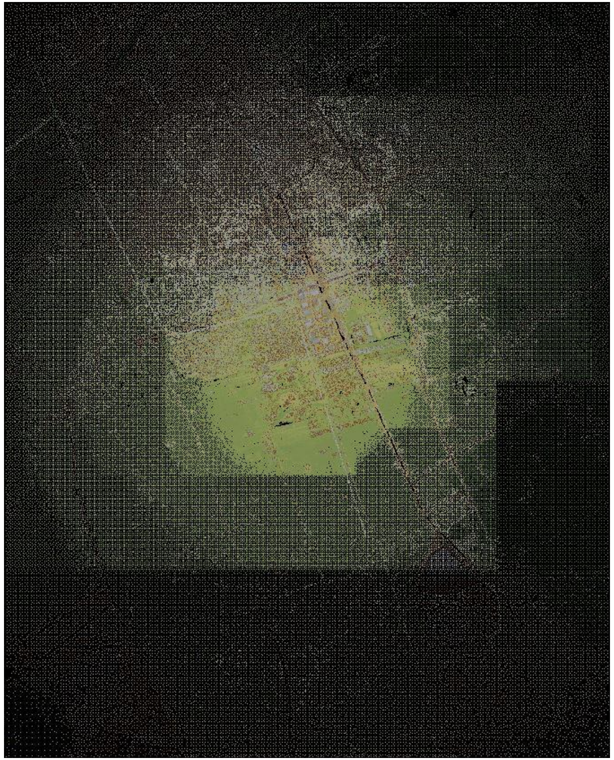
Continuous visualization



TU Delft

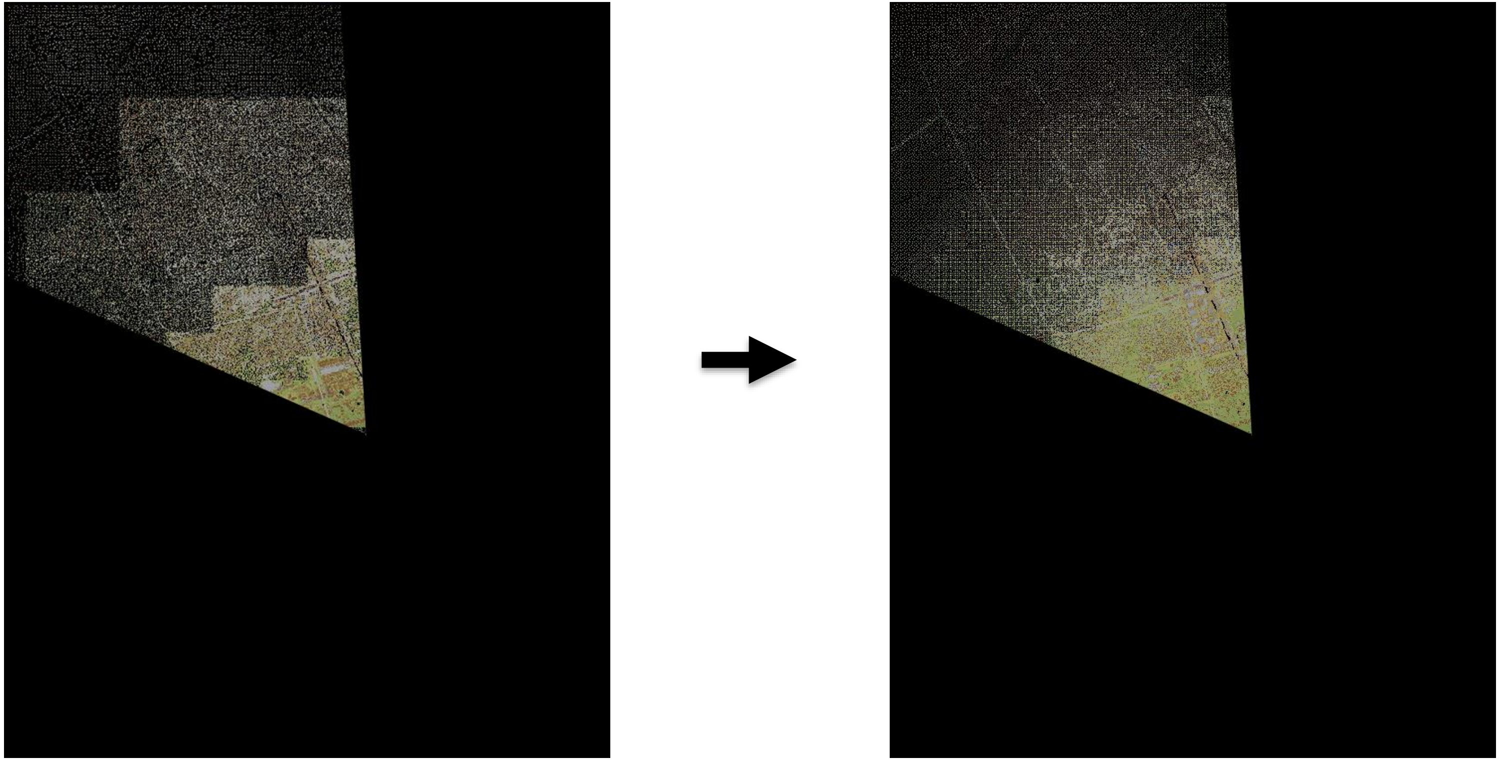
Top-down data set



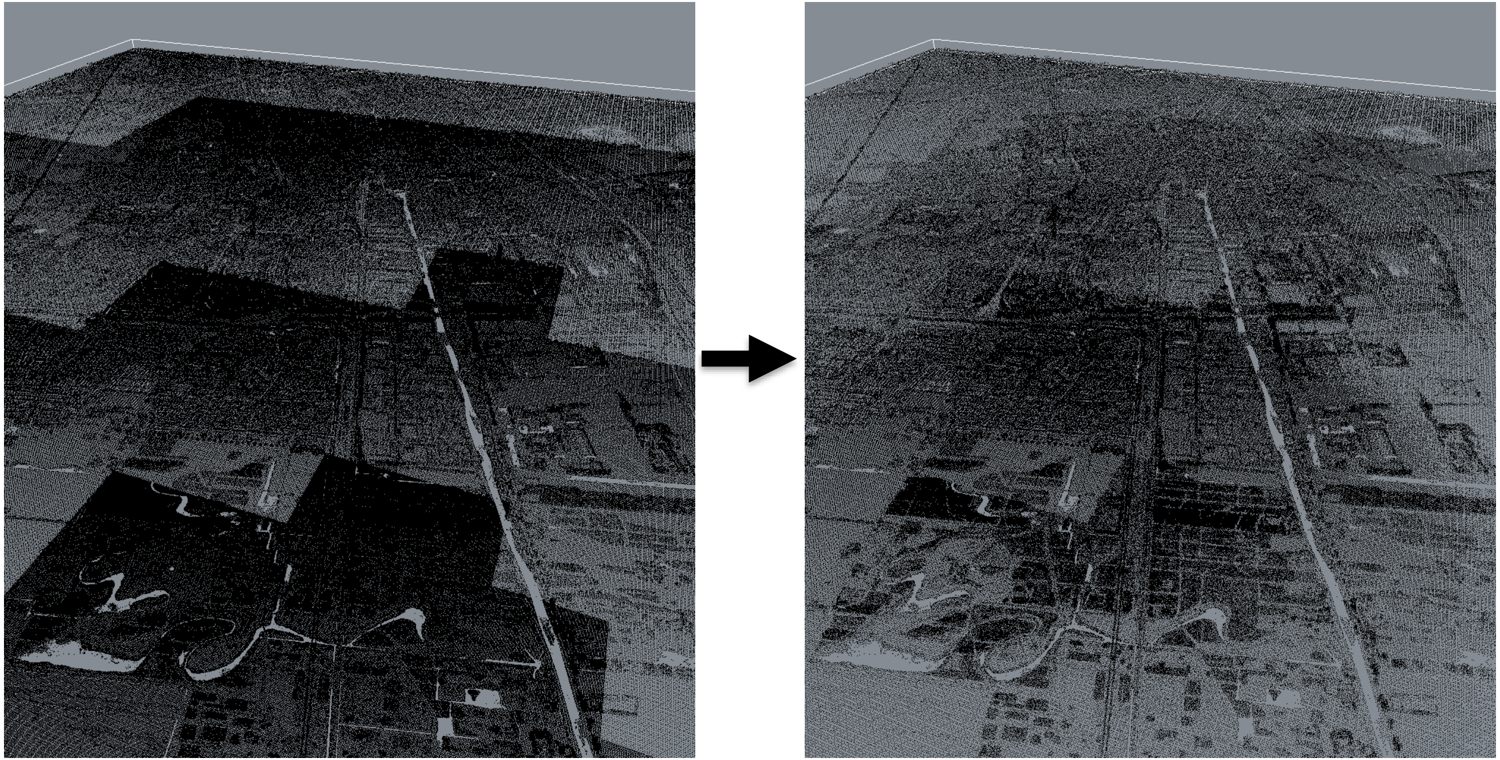


TU Delft

Top-down camera angle



Perspective



- Introduction
- Research
- Theory
- Implementation

TU Delft

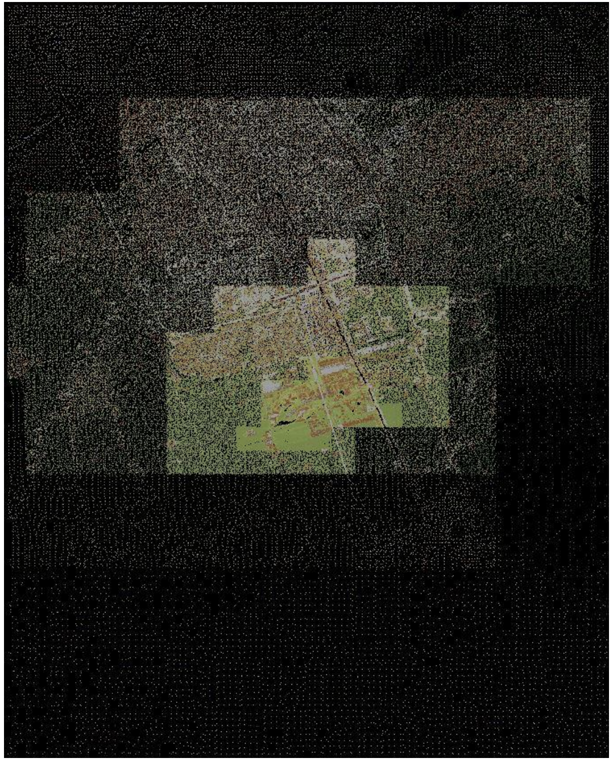
- Introduction
- Research
- Theory
- Implementation

What is the problem?

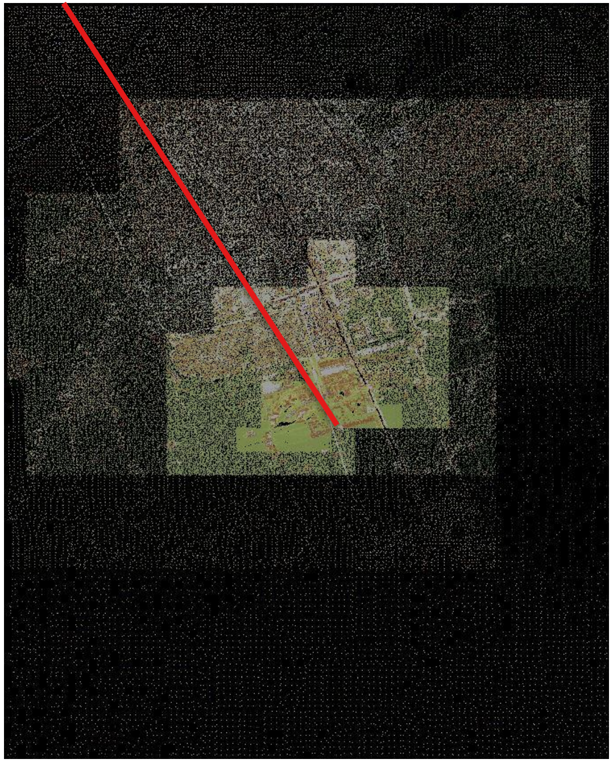
What is the problem?

Density jumps in web based point cloud visualization

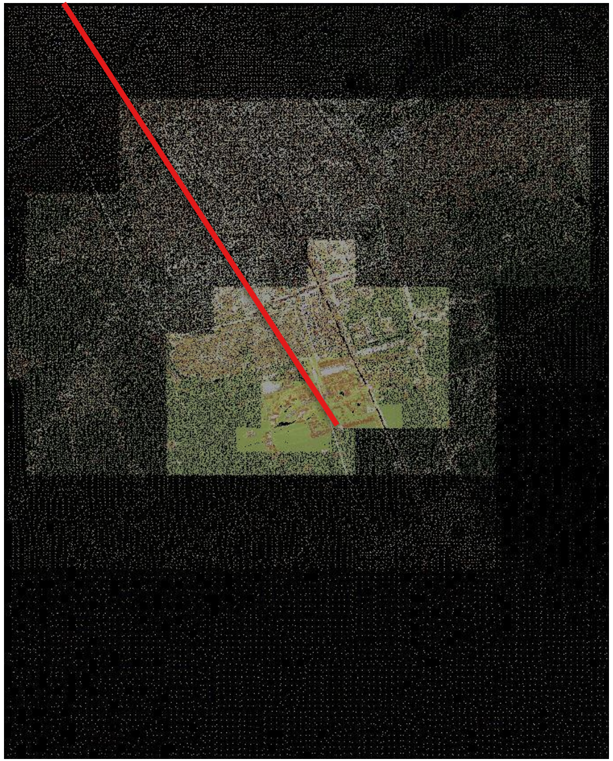
What is the problem?

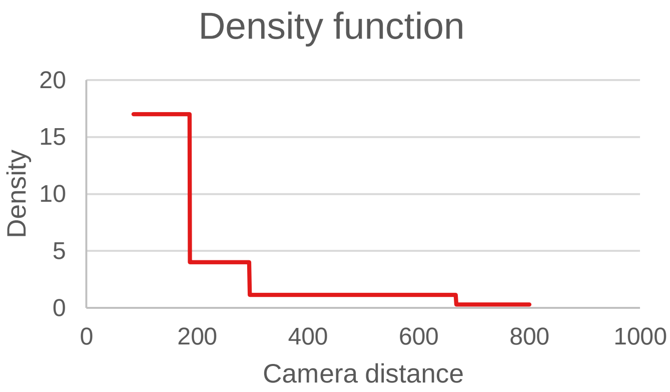


What is the problem?

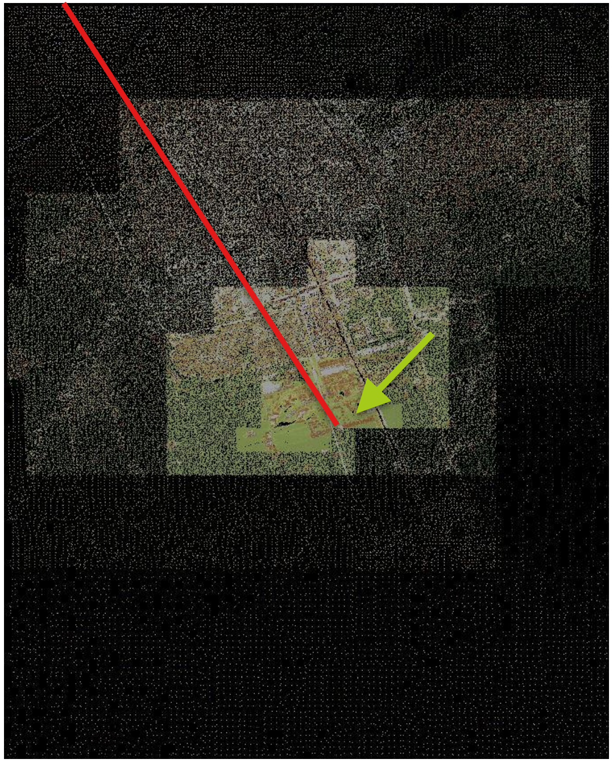


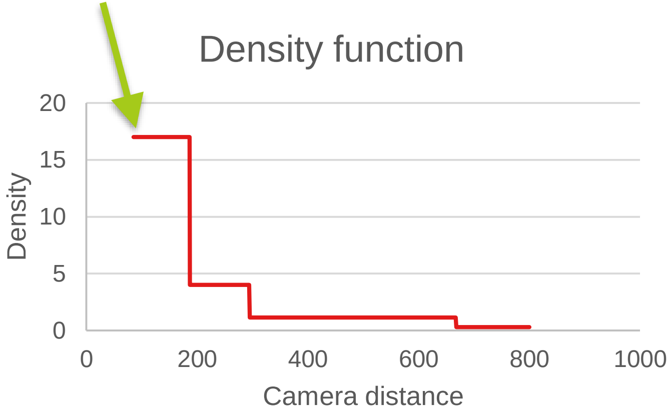
What is the problem?



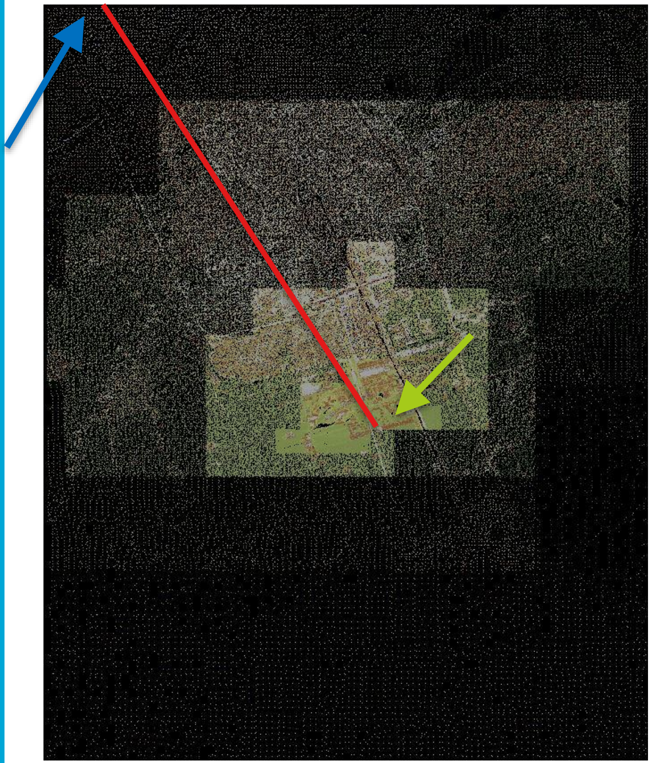


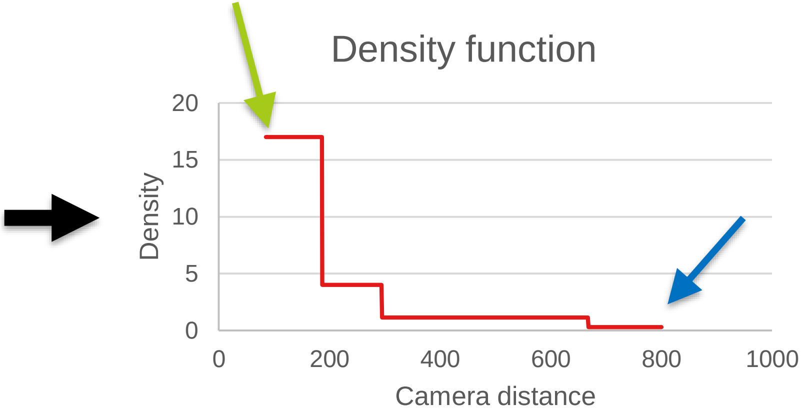
What is the problem?



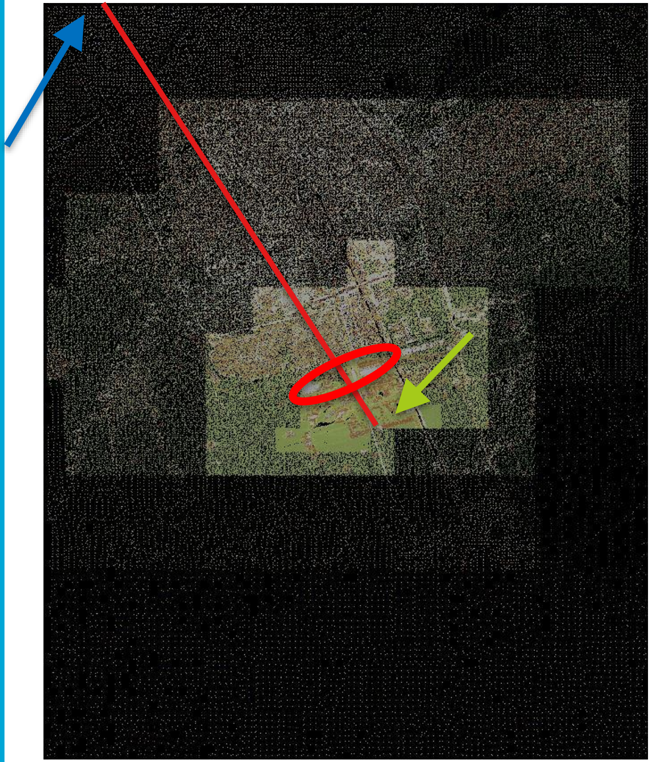


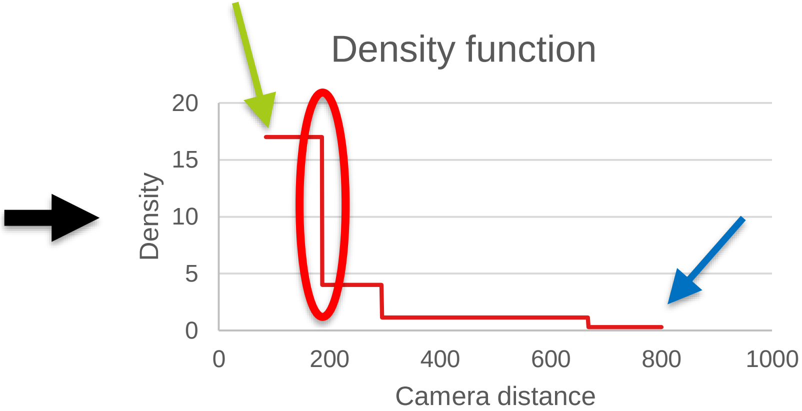
What is the problem?



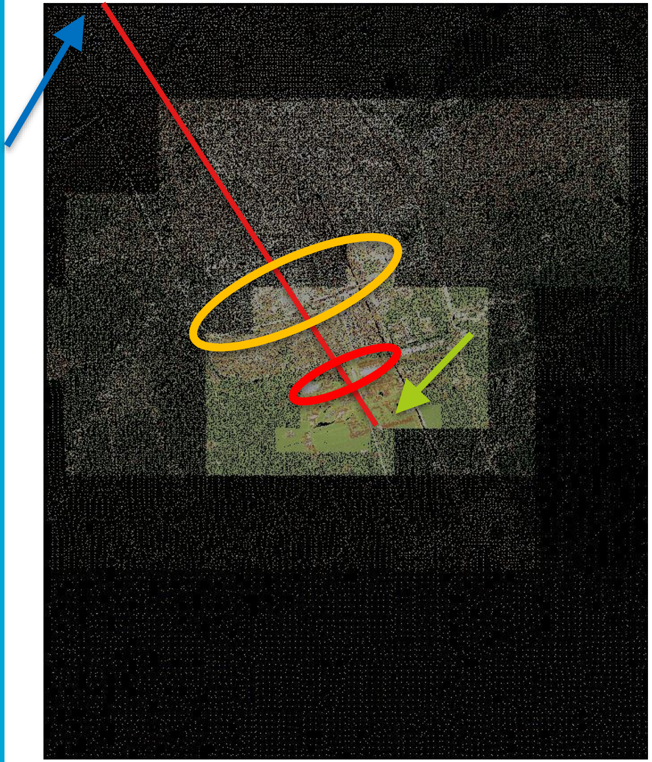


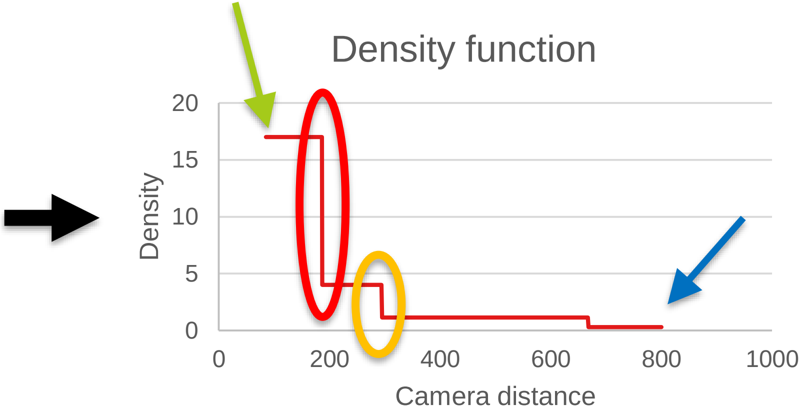
What is the problem?



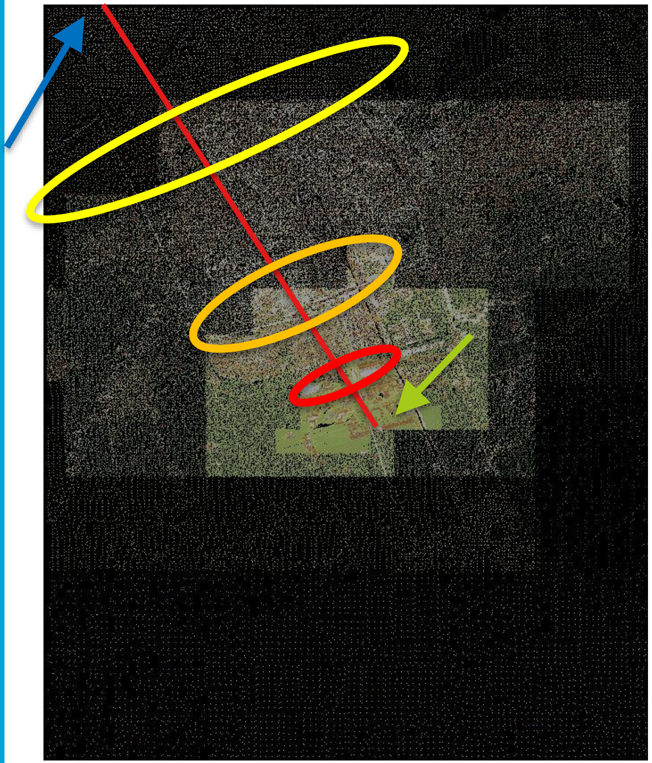


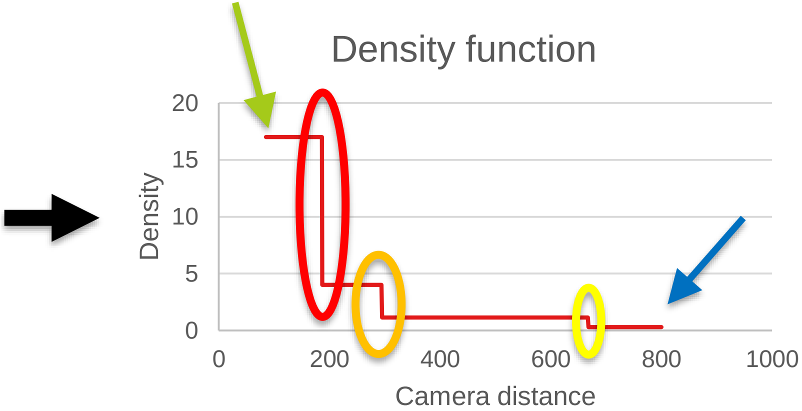
What is the problem?





What is the problem?





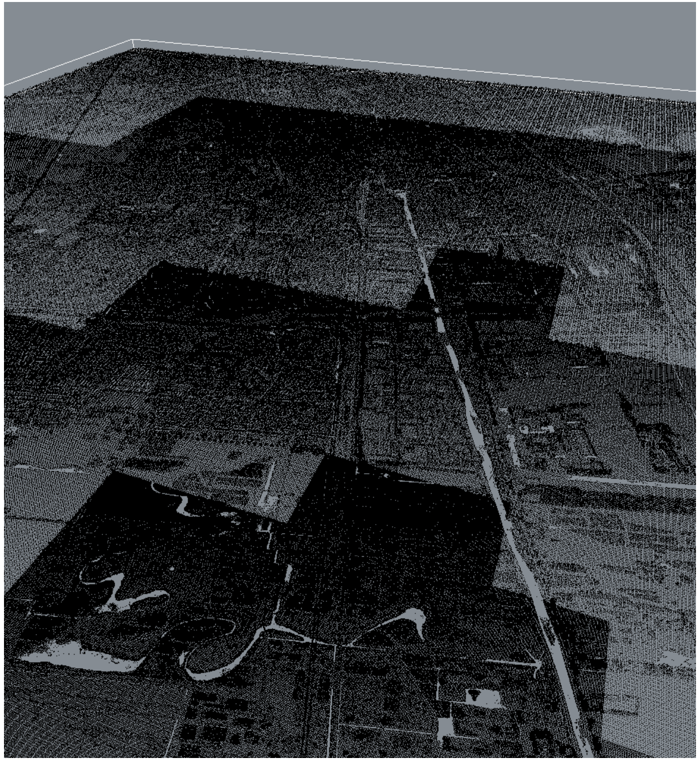
What is the problem?

- In different media these density jumps manifest themselves differently

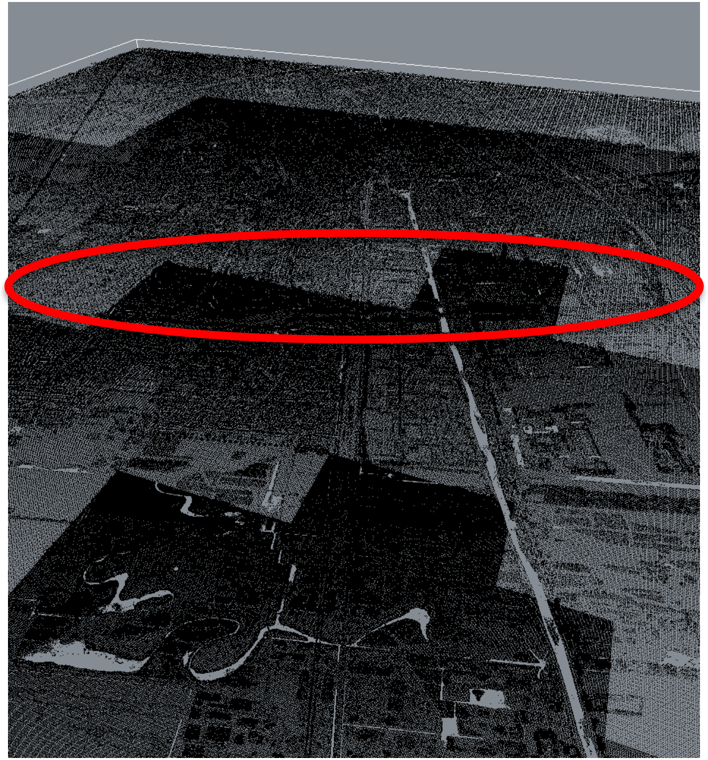
What is the problem?

- Single frame perspective

What is the problem?



What is the problem?



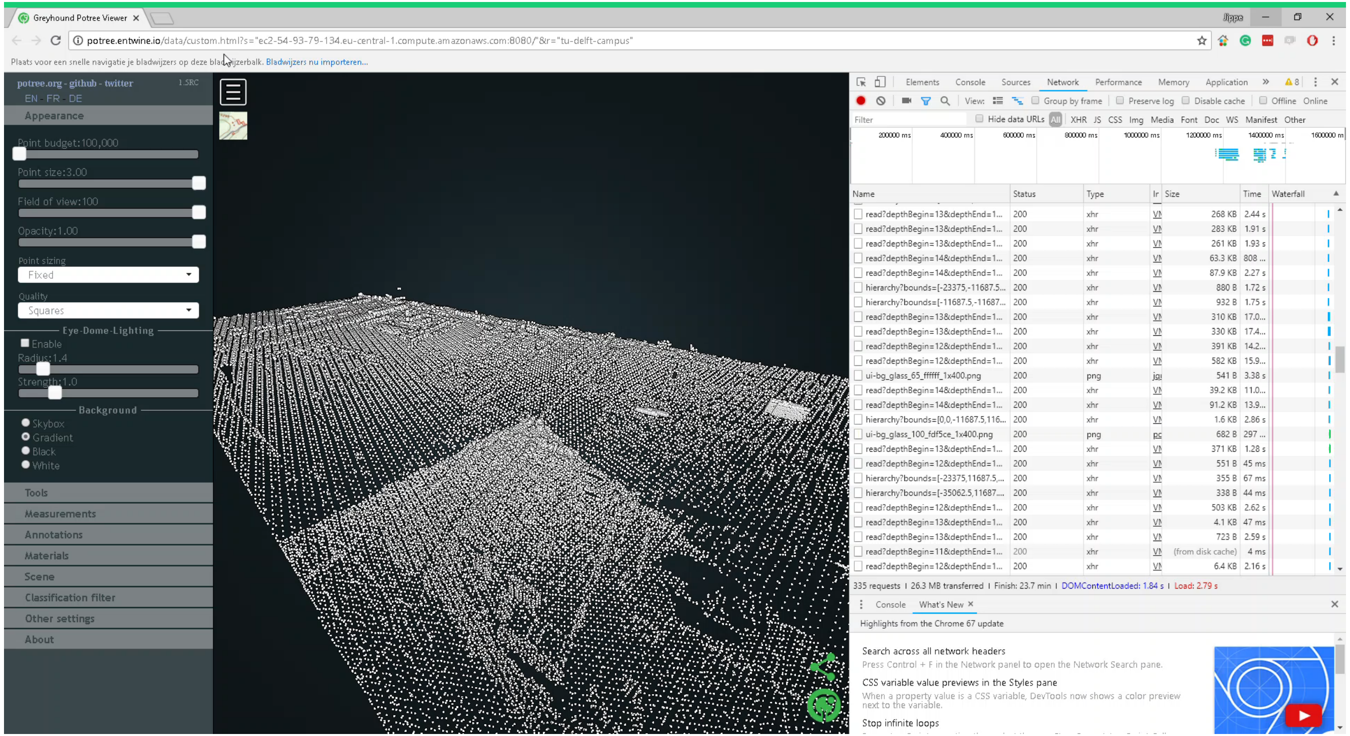
What is the problem?

What is the problem?

- Single frame perspective
- 30 fps web visualization

TUDelft

What is the problem?

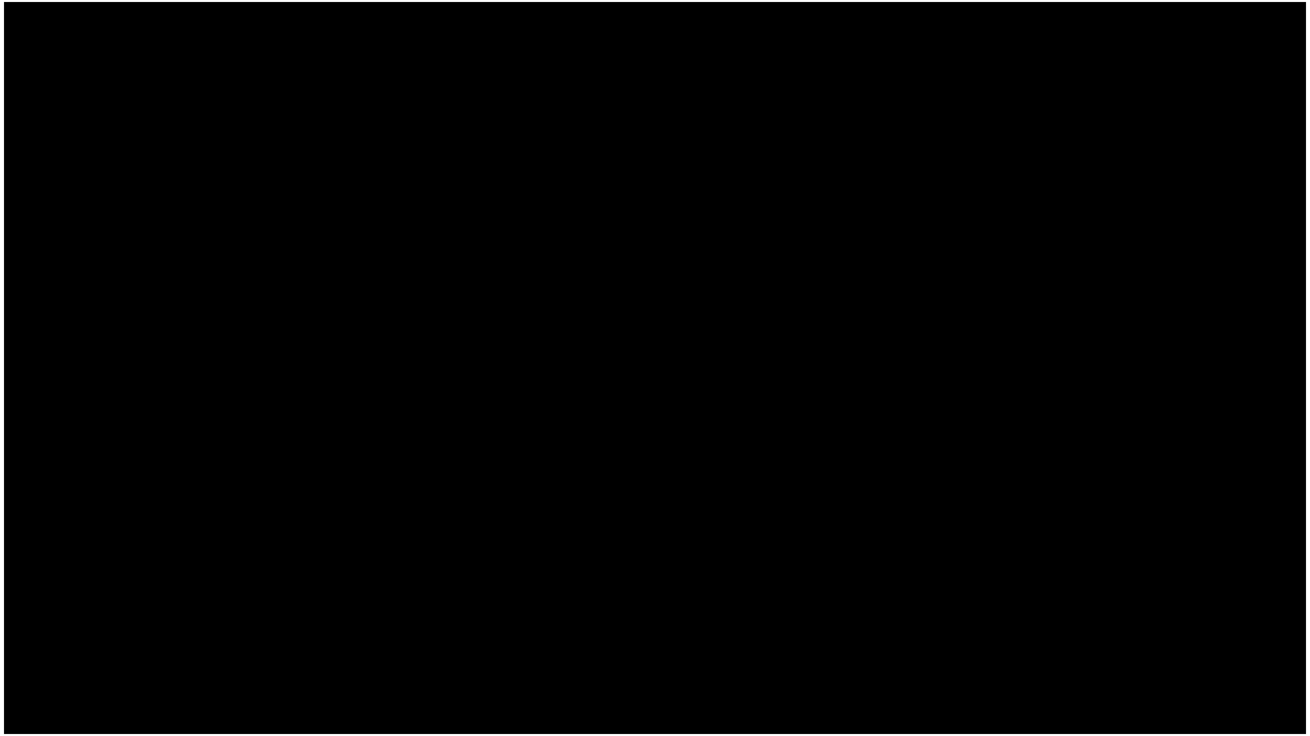


What is the problem?

- Single frame perspective
- 30 fps popping
- VR inconsistent loading

TU Delft

What is the problem?



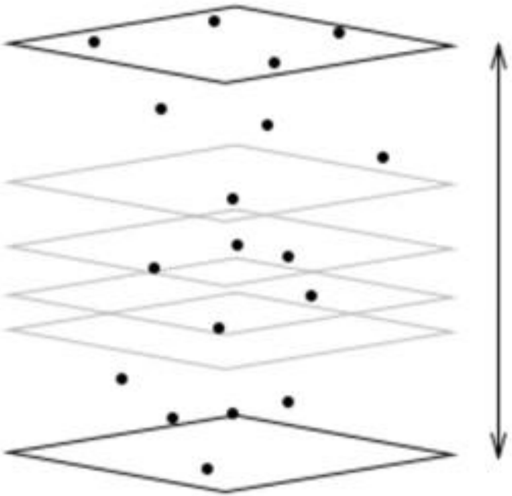
What is the solution?

TU Delft

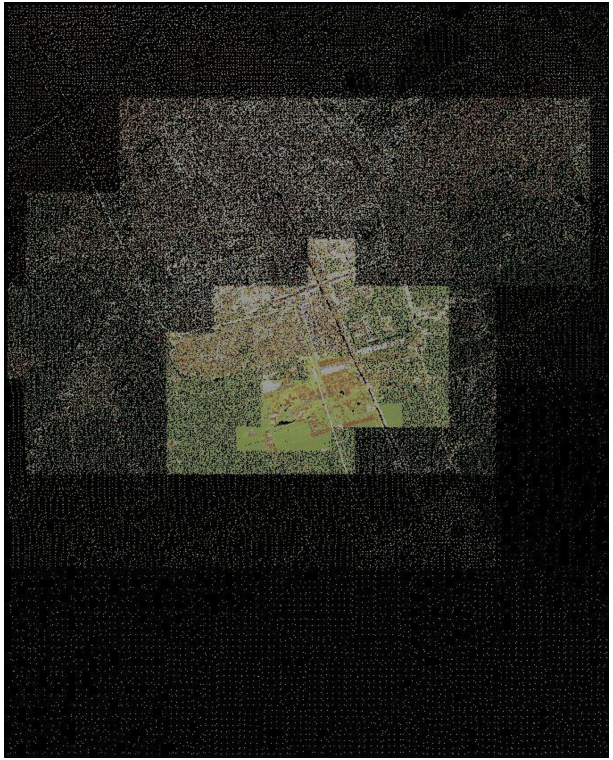
What is the solution?

Vario-scale visualization

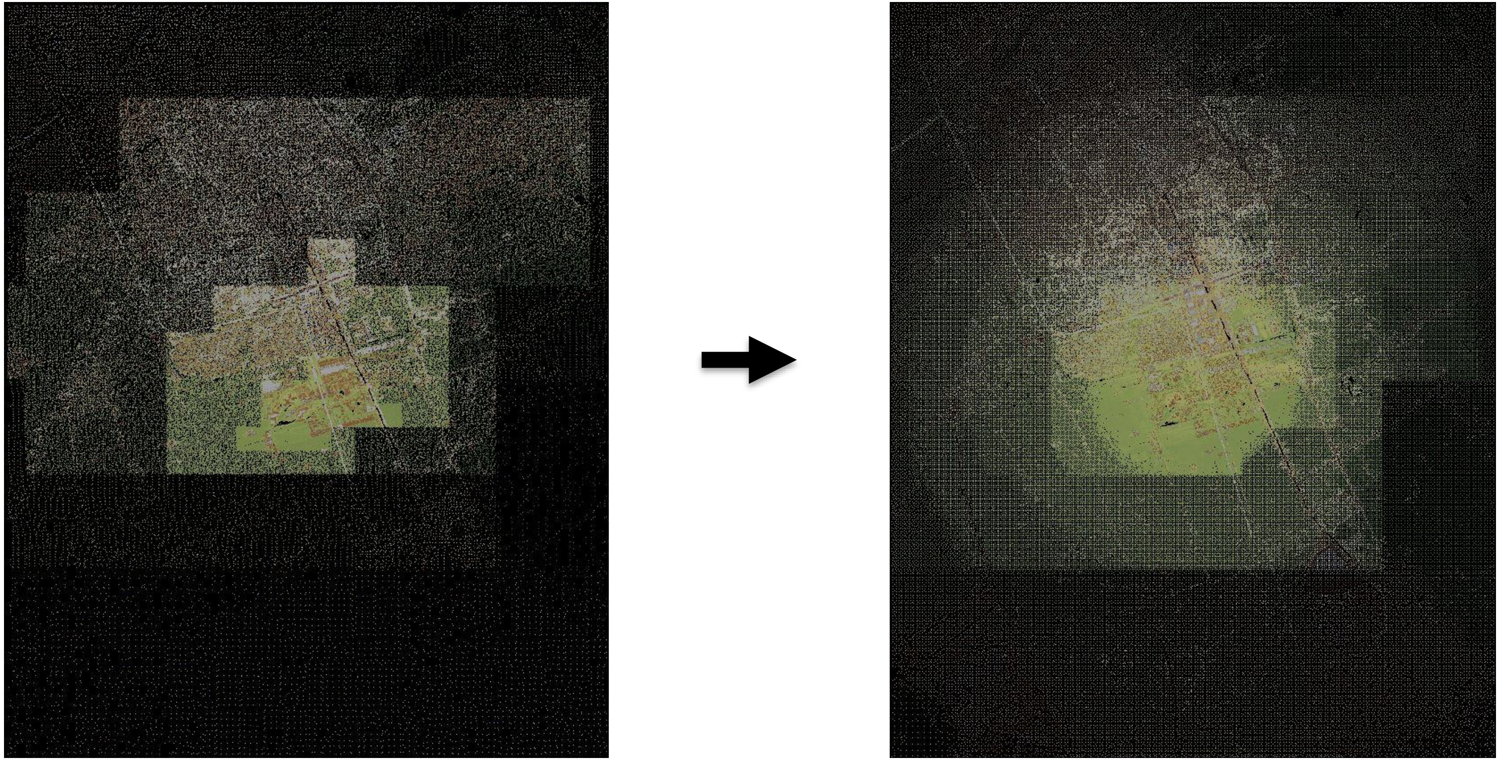
What is the solution?



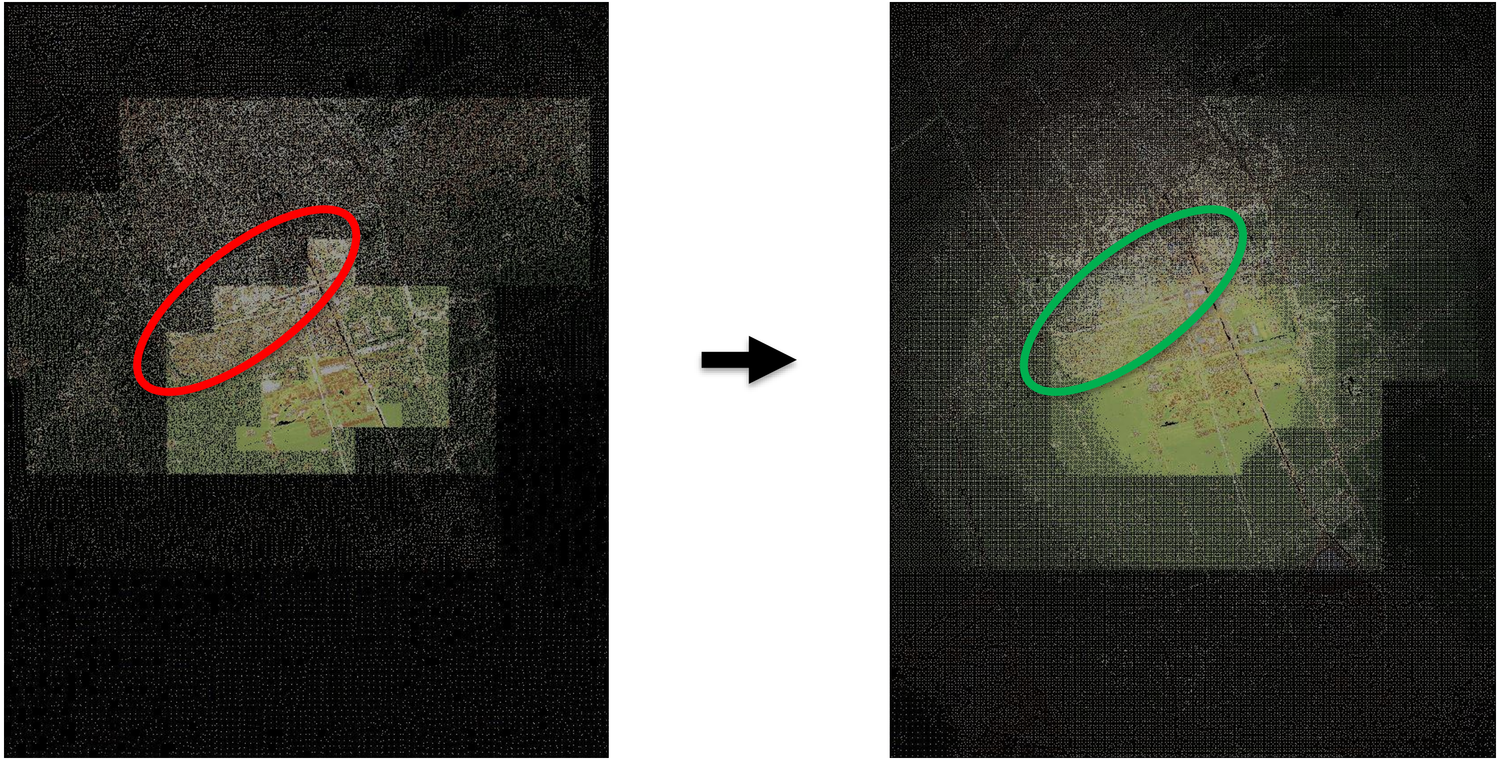
What is the proposed solution?



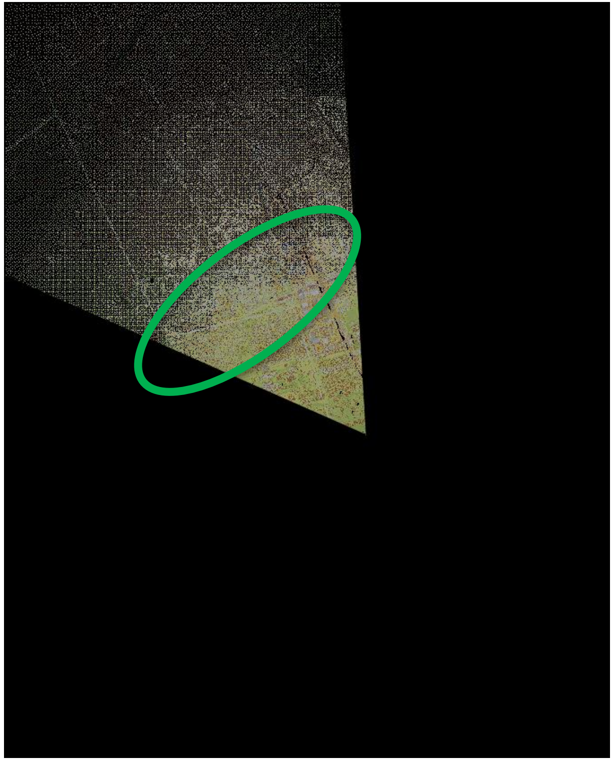
What is the proposed solution?



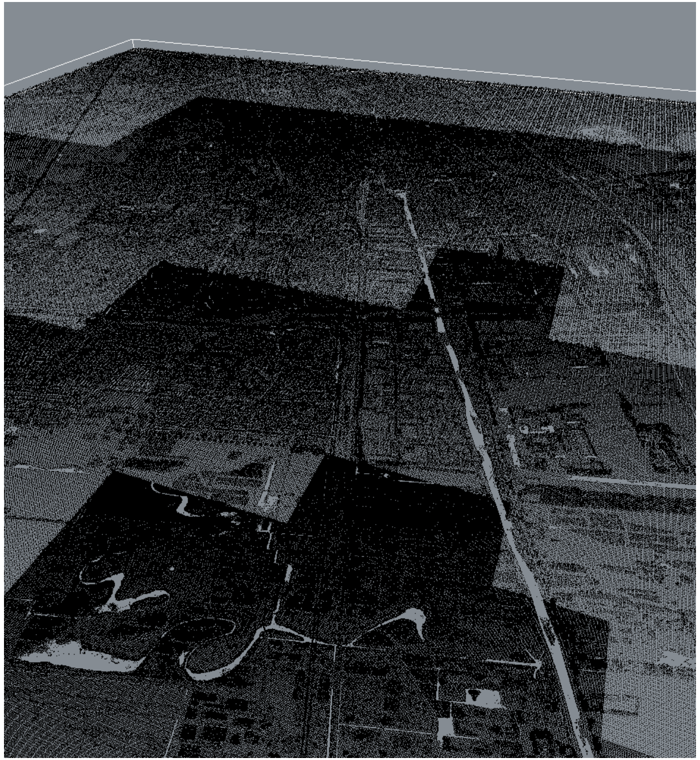
What is the proposed solution?



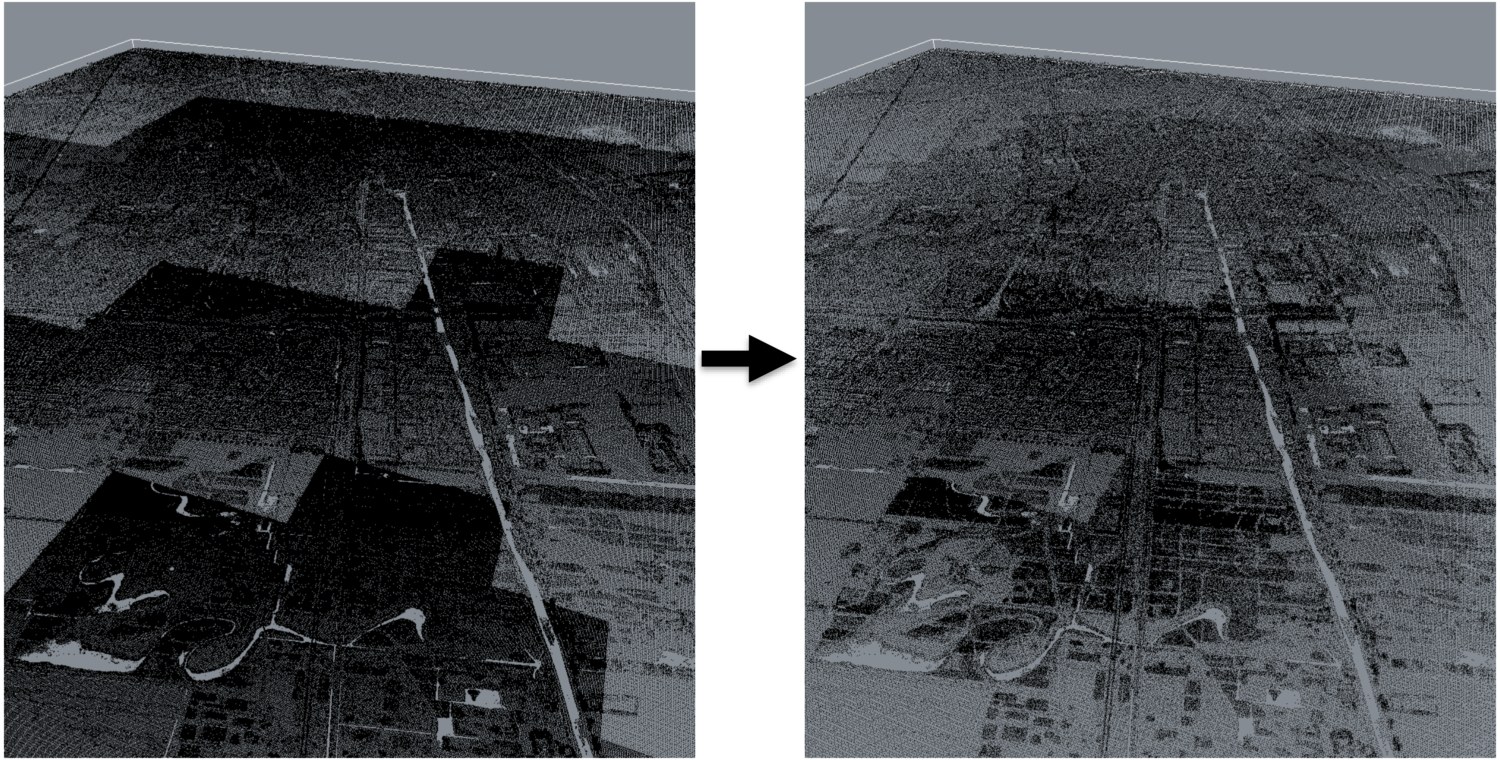
What is the proposed solution?



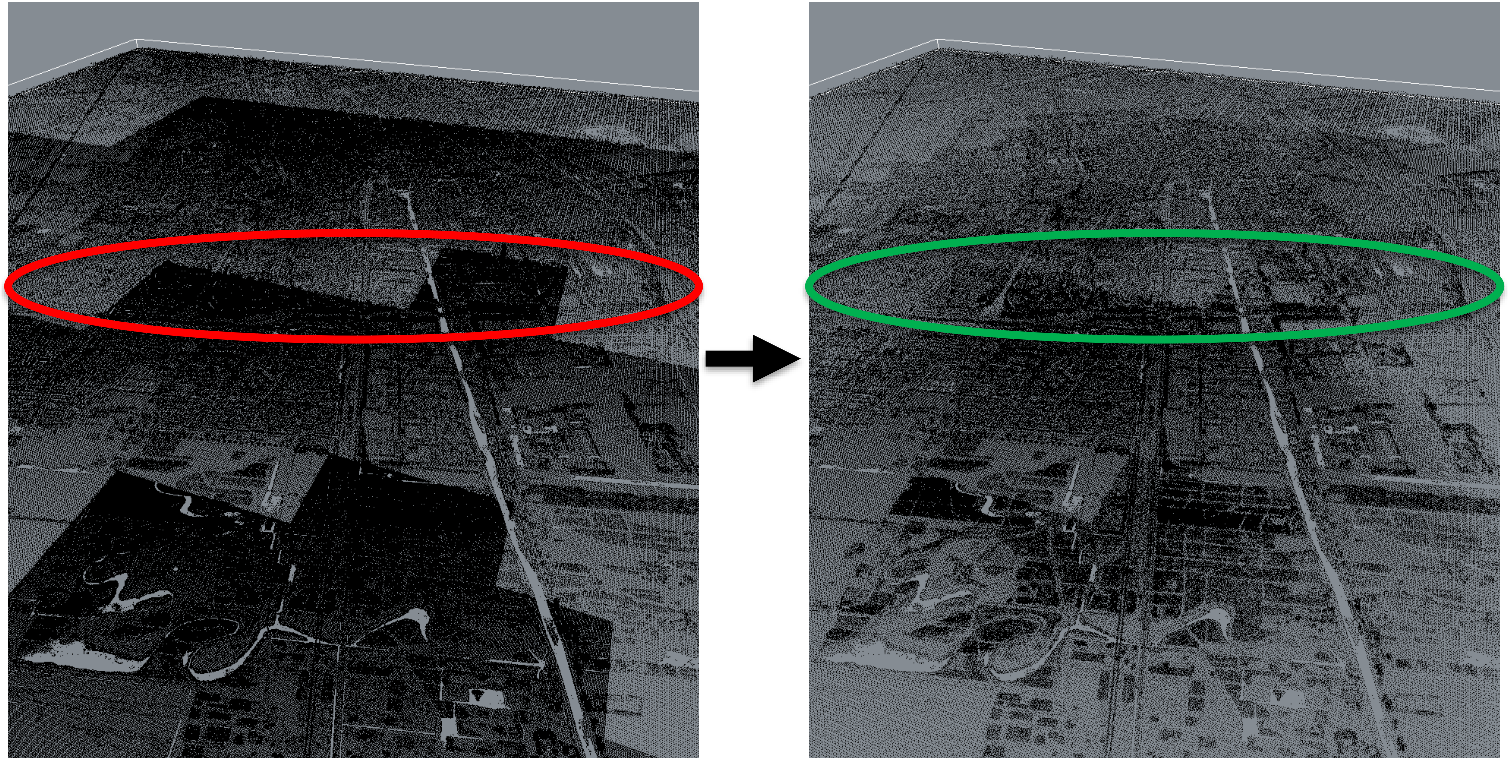
What is the proposed solution?



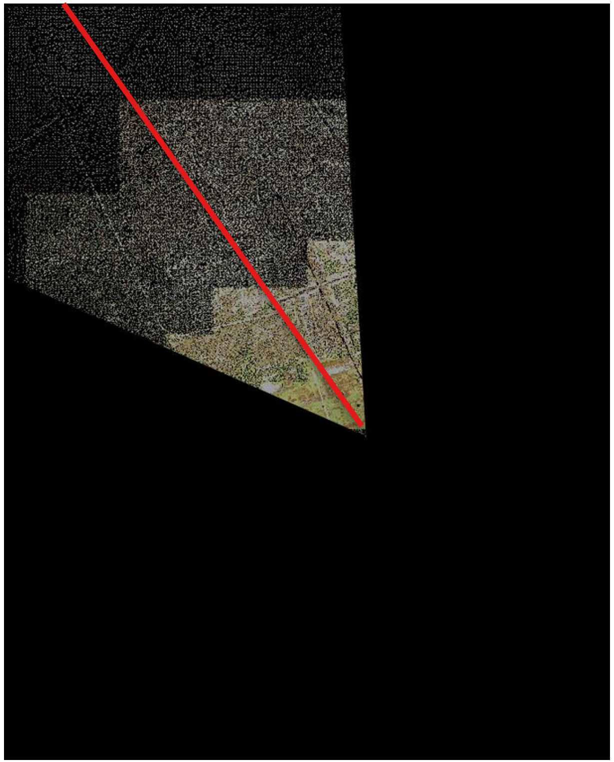
What is the proposed solution?



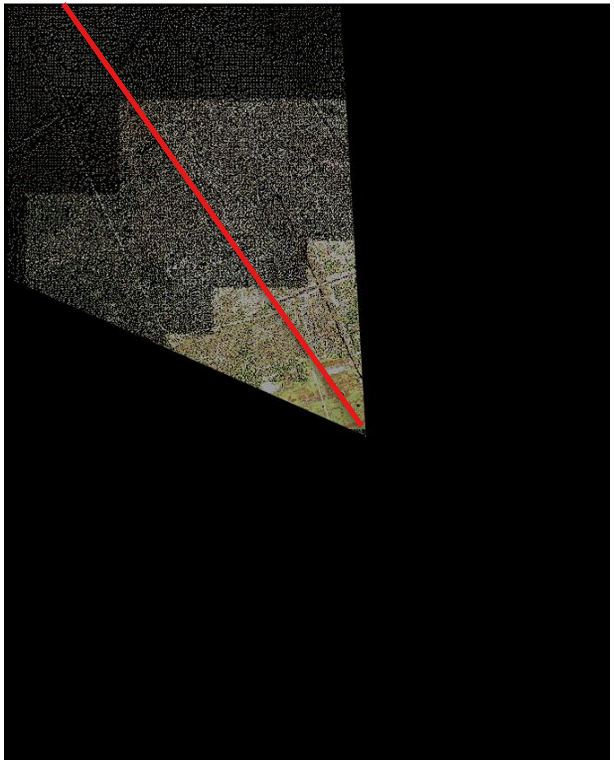
What is the proposed solution?

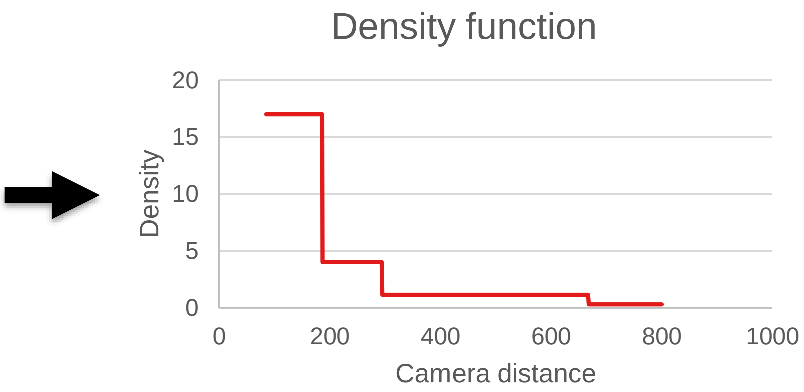


What is the proposed solution?

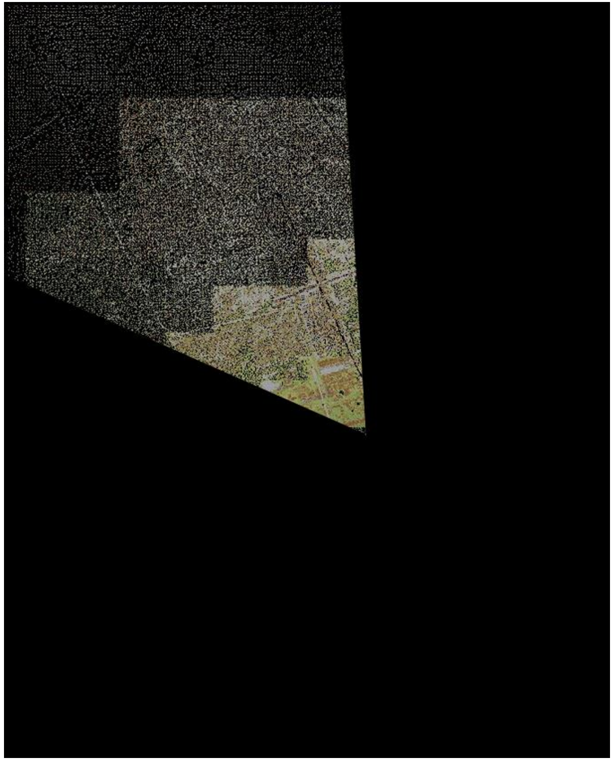


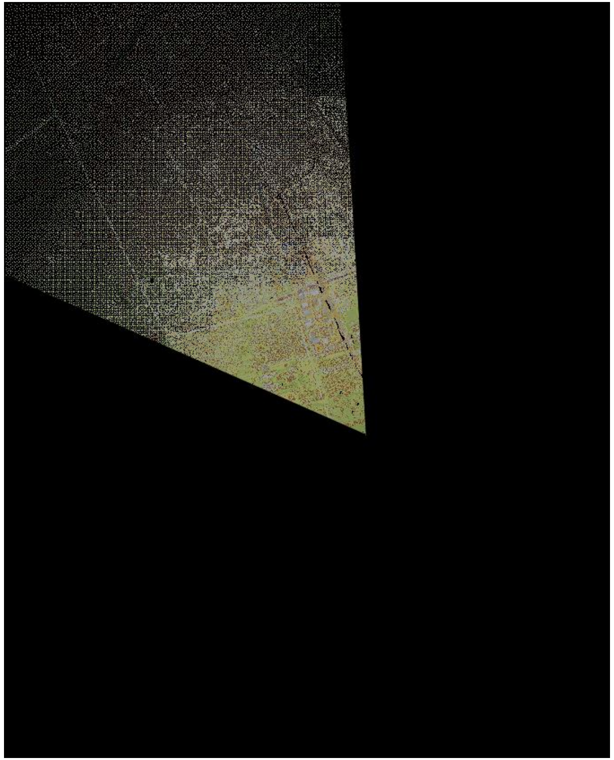
What is the proposed solution?



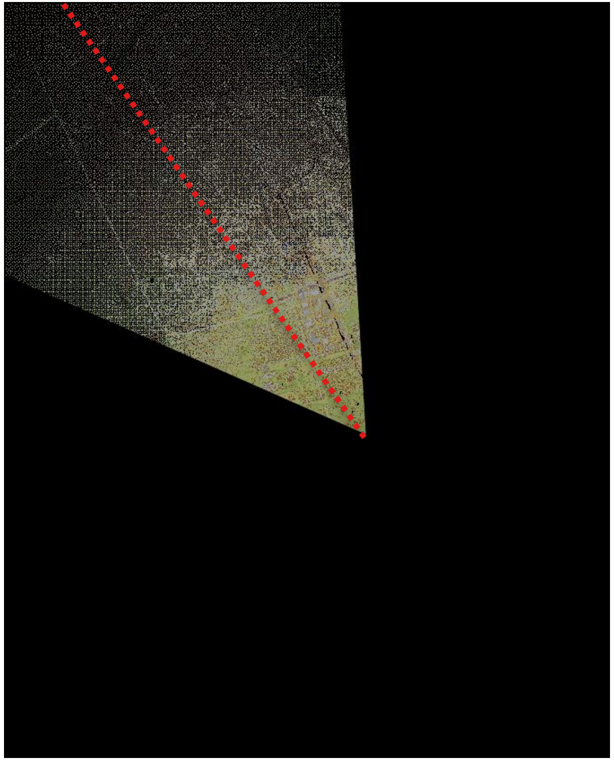


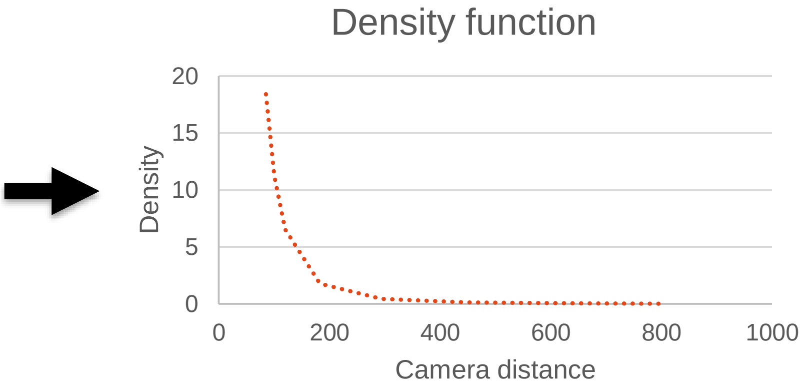
What is the proposed solution?





What is the proposed solution?





- Introduction
- Research
- Theory
- Implementation

Research objective

- To create a vario-scale visualization method for the AHN2 point cloud

TU Delft

Research question

To what extend can a vario-scale visualization method be created that eliminates density jumps from the web-based visualization of the AHN2 point cloud?

Supporting research questions

1. To what extent is the current body of research done on the vario-scale visualization of vector data sets relevant for the vario-scale visualization of point cloud data sets?

TU Delft

Supporting research questions

2. To what extend can a theoretical post-processing approach be created for vario-scale visualization of point cloud data sets?

Supporting research questions

3. Which point-cloud processing framework is best suited to create a proof-of-concept vario-scale visualization platform for the AHN2 point cloud?

TU Delft

Supporting research questions

4. To what extend can the theoretical approach be implemented in an existing point cloud web visualization framework?

TU Delft

Presentation

- Introduction
- Research
- Theory
- Implementation

TU Delft

Supporting research question

1. To what extent is the current body of research done on the vario-scale visualization of vector data sets relevant for the vario-scale visualization of point cloud data sets?

Theory – Vector data sets

2002 – Adaptive zooming (Cecconi and Galanda)

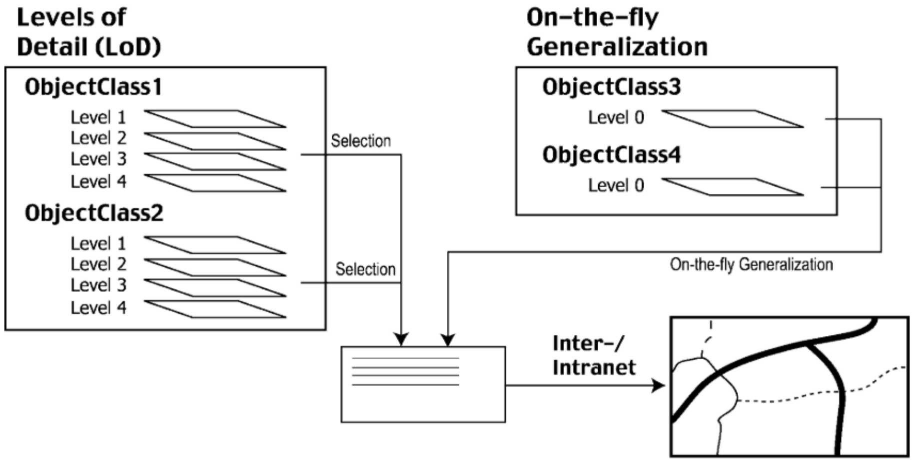
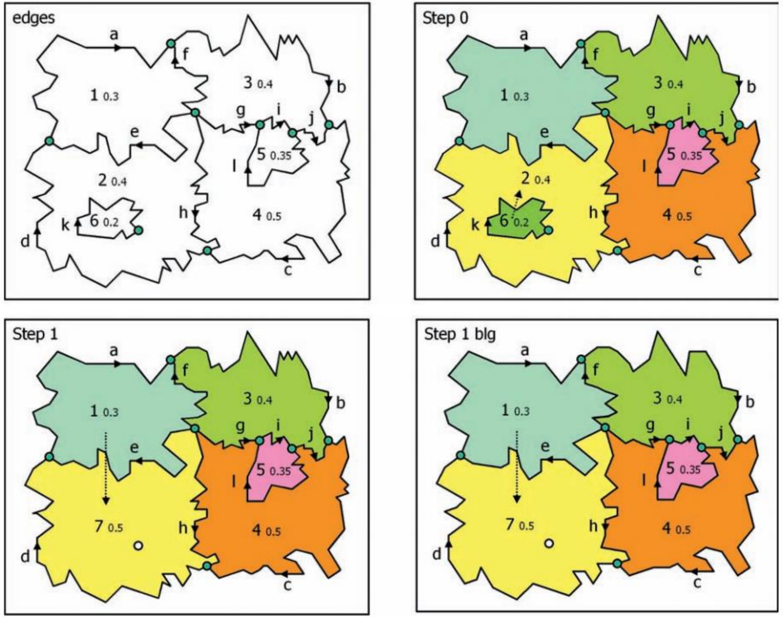


Figure 1: *Principle of adaptive zooming and web map generation based on LoD and on-the-fly generalization (Source: Cecconi and Galanda* [14]*).*

Theory – Vector data sets

2005 – Variable scale (van Oosterom)



Theory – Vector data sets

2005 – Variable scale (van Oosterom)

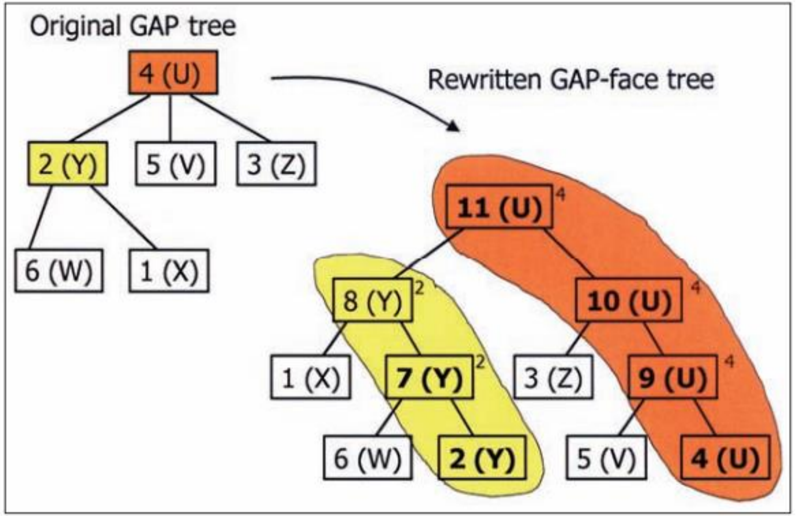


Figure 5. The classic GAP tree rewritten as the GAP-face tree (with a new object Id whenever a face changes and the old object Id appearing in a small font to the upper right of a node). The class is shown in brackets after the object Id.

Theory – Vector data sets

2005 – Variable scale (van Oosterom)

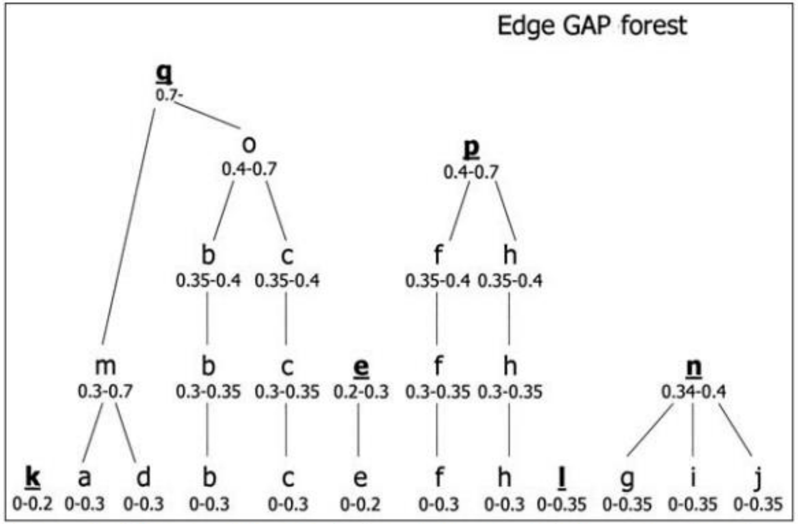
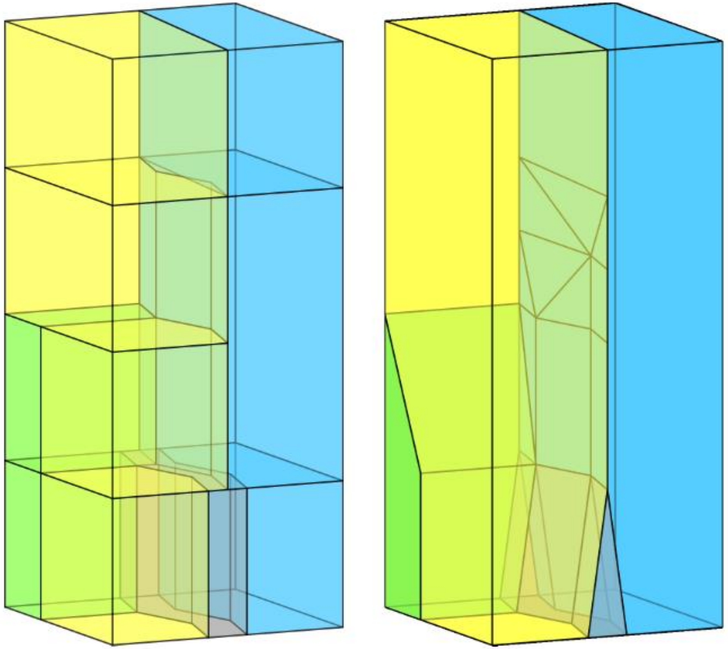


Figure 6. GAP-edge forest (with important ranges). Note that the edges shown in bold and the underlined letters k, q, e, p, l, and n are the roots of the different GAP-edge trees.

Theory – Vector data sets

2013 – Variable scale (Suba)



(a) SSC for classic tGAP (b) SSC for smooth tGAP

Theory – Point cloud data sets

1. To what extent is the current body of research done on the vario-scale visualization of vector data sets relevant for the vario-scale visualization of point cloud data sets?

TU Delft

Theory – Point cloud data sets

- 2002: Simplification by semantics
- 2005: Simplification by attributes (faces & edges)
- 2013: Continuous simplification in 3 dimensions (X, Y, Scale)

TU Delft

Theory – Point cloud data sets

- 2002: Simplification by semantics
- 2005: Simplification by attributes (faces & edges)
- 2013: Continuous simplification in 3 dimensions (X, Y, Scale)

TU Delft

Theory – Point cloud data sets

- 2002: Simplification by semantics
- 2005: Simplification by attributes (faces & edges)
- 2013: Continuous simplification in 3 dimensions (X, Y, Scale)

TU Delft

Theory – Point cloud data sets

- 2002: Simplification by semantics
- 2005: Simplification by attributes (faces & edges)
- 2013: Continuous simplification in 3 dimensions (X, Y, Scale)

TU Delft

Theory – Point cloud data sets

Simplification in 3 dimensions (X, Y, camera distance) using the density formula

TU Delft

Theory – Simplification in 3 dimensions

- Camera origin

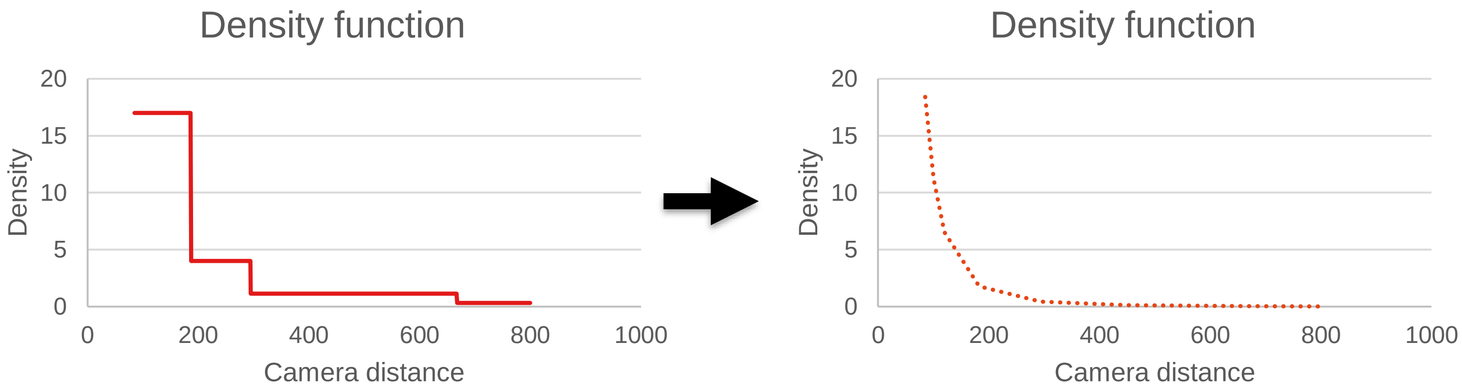
TU Delft

Theory – Simplification in 3 dimensions

- Camera origin
- Point distance from camera

Theory – Simplification in 3 dimensions

- Camera origin
- Point distance from camera



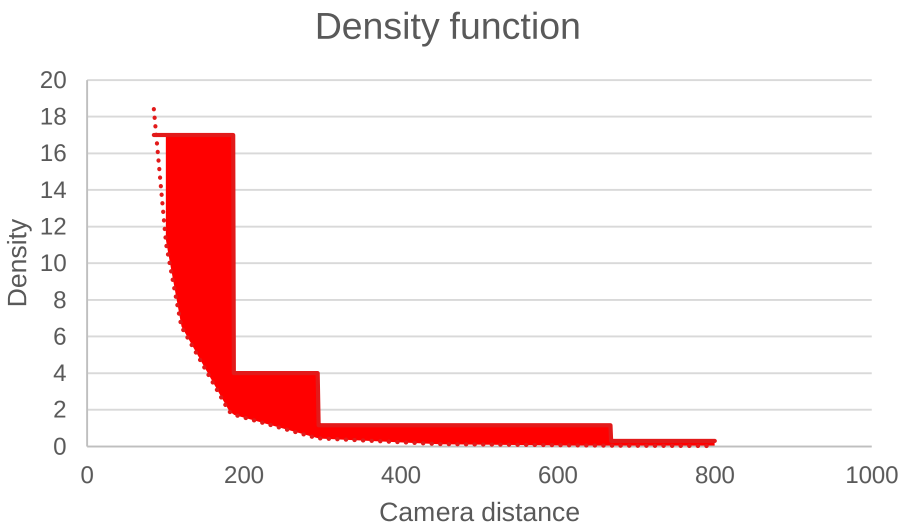
Theory – Simplification in 3 dimensions

- Perform a per-point evaluation to determine whether the point should be rendered or not

TU Delft

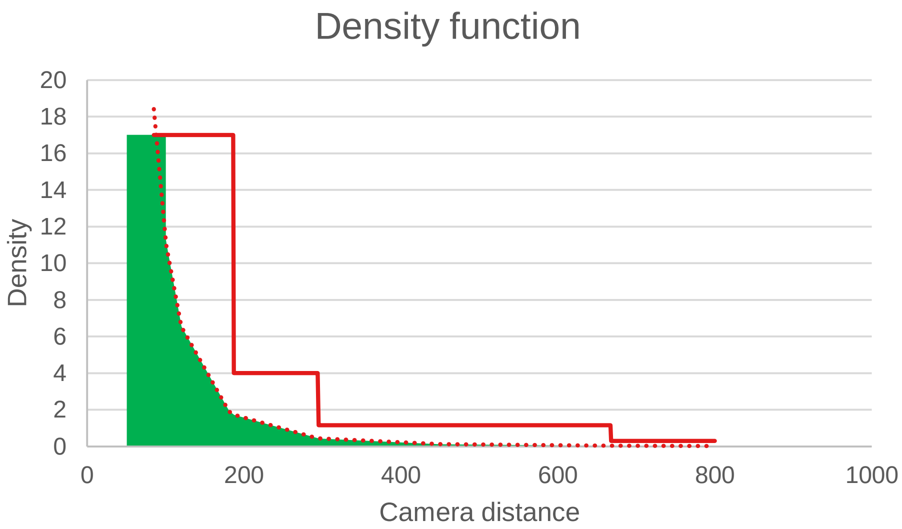
Theory – Simplification in 3 dimensions

- Perform a per-point evaluation



Theory – Simplification in 3 dimensions

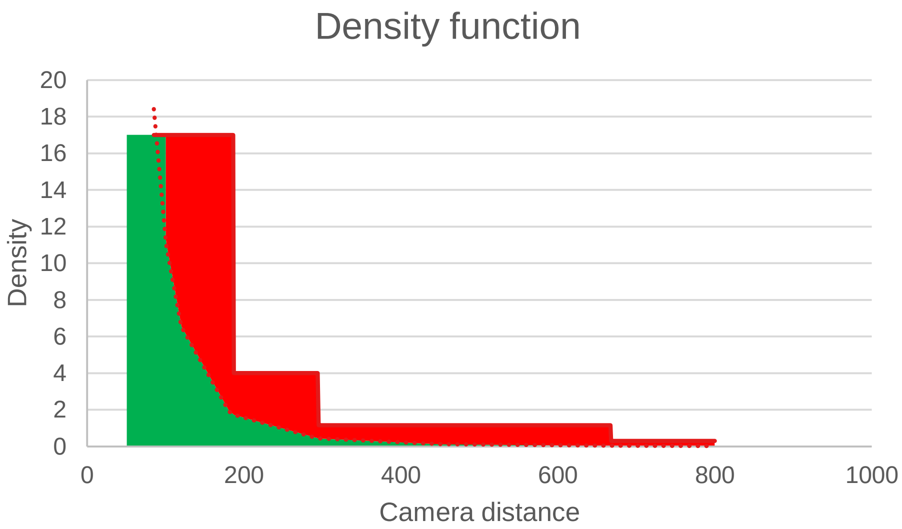
- Perform a per-point evaluation



TU Delft

Theory – density function

- We know what we want to keep and what we want to remove



Theory – density function

- We know what we want to keep and what we want to remove
- How do we know the density function?

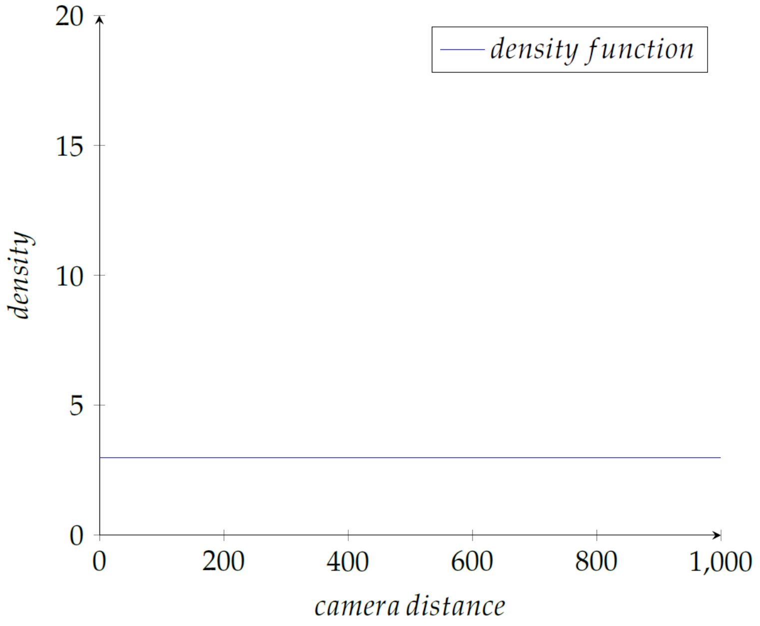
TUDelft

Theory – basic density function

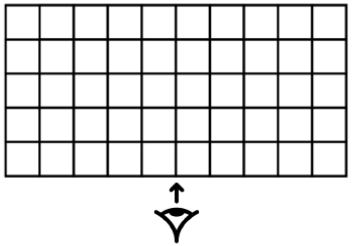
- Lets start with a basic example

Theory – basic density function

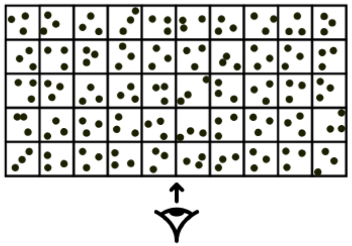
- Lets start with a basic example



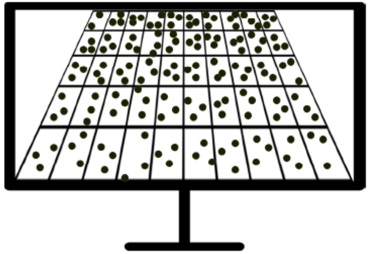
Theory – basic density function



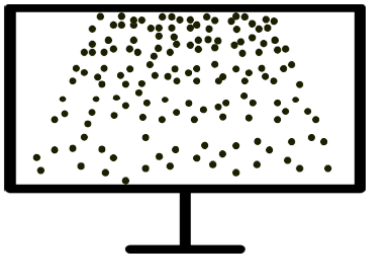
(a) 2D spatial extent of LiDAR data set, with density grid



(b) 2D spatial extents, with a density of 3 points per $n * n$ units



(c) Perspective translation of point cloud with a density of 3 points per $n * n$ units



(d) User's visualization of the point cloud

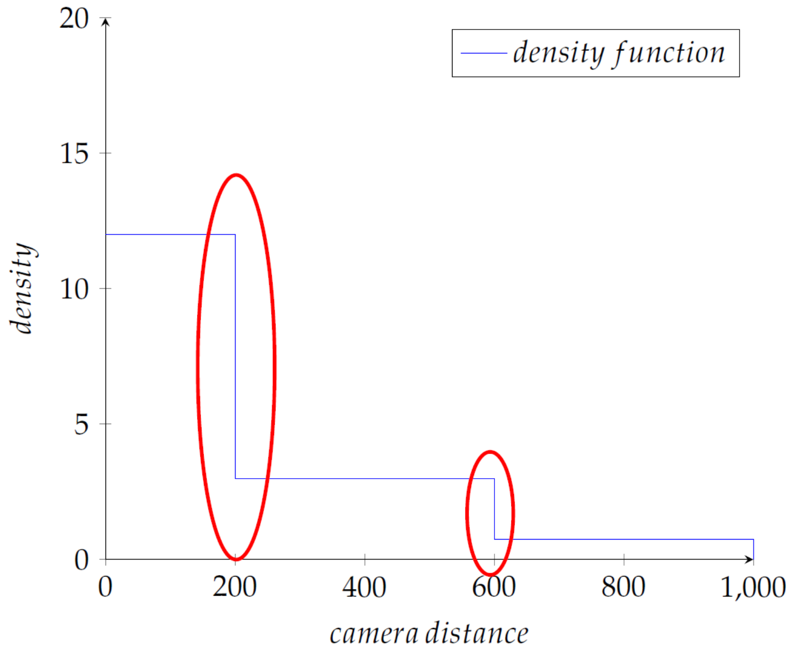
Theory – octree density function

- The current situation

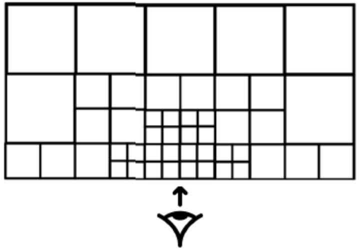
TU Delft

Theory – octree density function

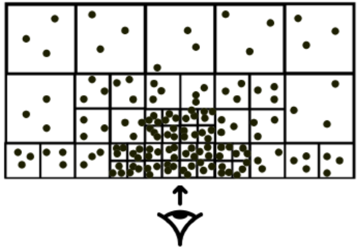
- The current situation



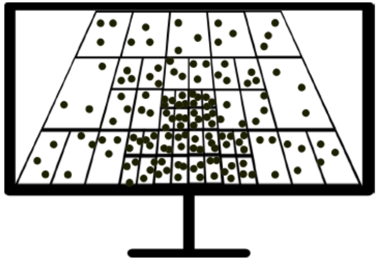
Theory – octree density function



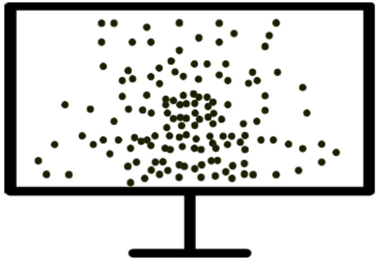
(a) 2D spatial extents, overlayed with an Octree selection grid



(b) 2D spatial extents, with an increased density closer to the camera

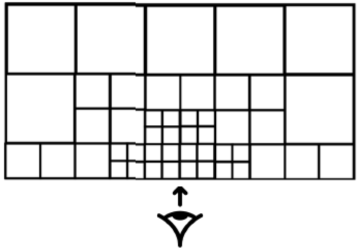


(c) Perspective translation of point cloud using octree selection grid

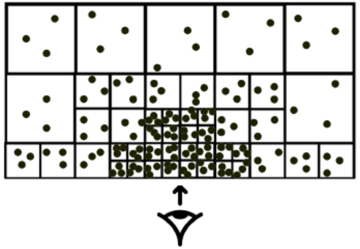


(d) User's visualization of the point cloud

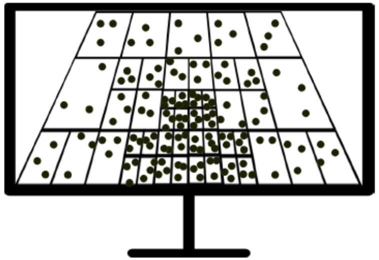
Theory – octree density function



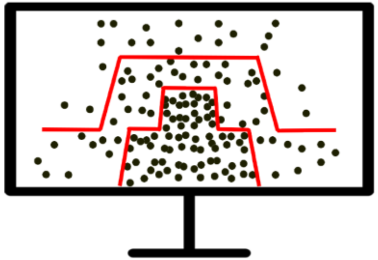
(a) 2D spatial extents, overlayed with an Octree selection grid



(b) 2D spatial extents, with an increased density closer to the camera



(c) Perspective translation of point cloud using octree selection grid



(d) User's visualization of the point cloud

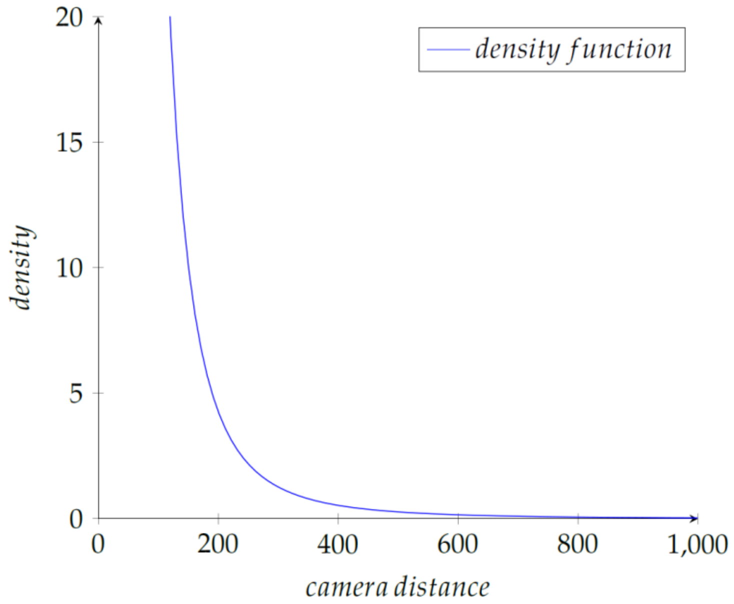
TUDelft

Theory – vario-scale density function

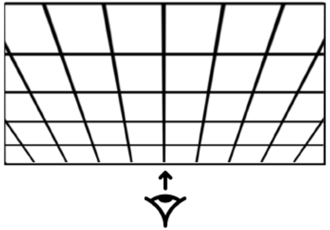
- The desired situation

Theory – vario-scale density function

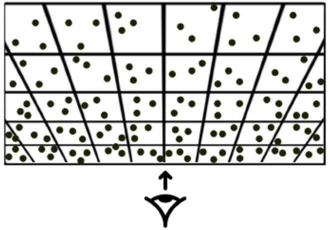
- The desired situation



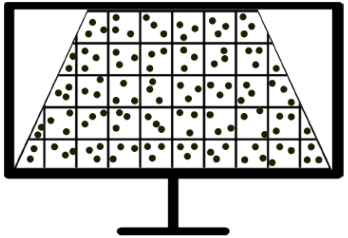
Theory – vario-scale density function



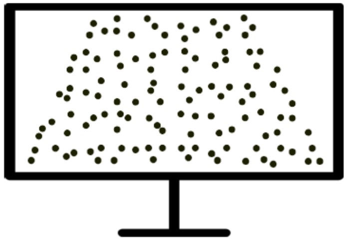
(a) 2D spatial extents, overlayed with the inverse density grid



(b) 2D spatial extents, with the inverse density grid filled with points



(c) Perspective translation of point cloud using the perspective matrix translation



(d) User's visualization of the point cloud

Theory – vario-scale density function

- The desired situation
- A global vario-scale function

TU Delft

Theory – vario-scale density function

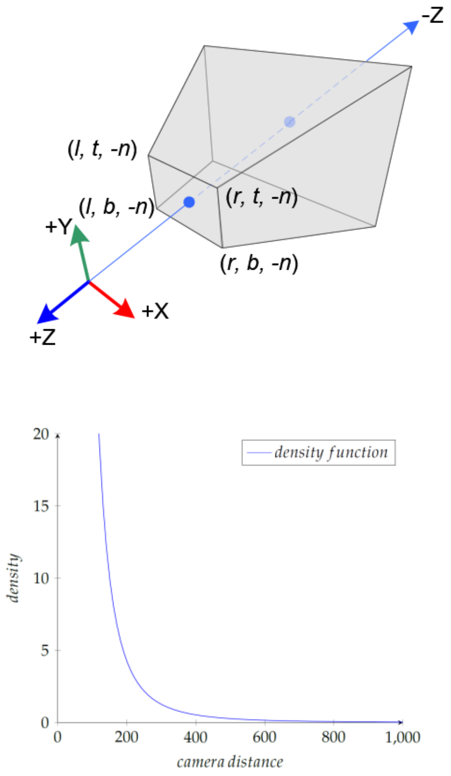
- The desired situation
- A global vario-scale function
- Dependant on a translation through the perspective matrix

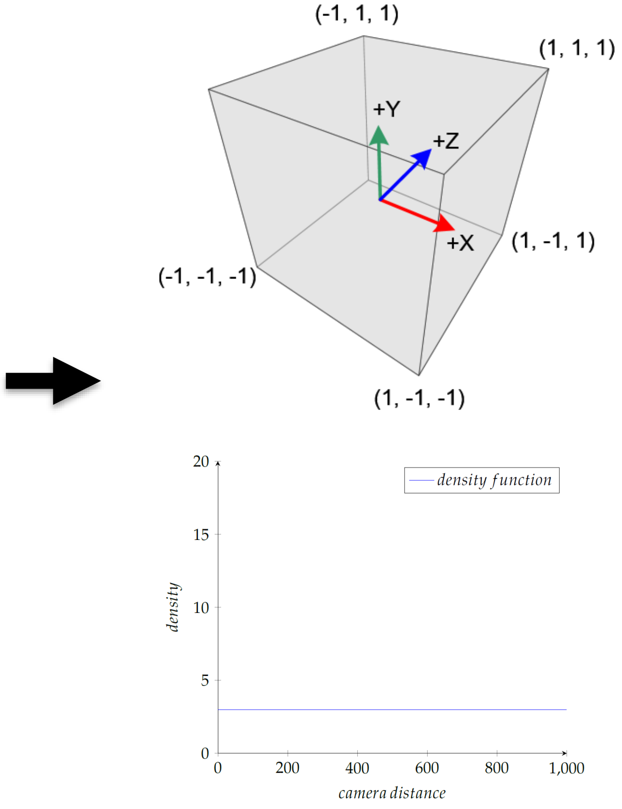
Theory – vario-scale density function

- The desired situation
- A global vario-scale function
- Dependant on a translation through the perspective matrix
- Dependant on
  - Near-plane
  - Far-plane
  - Field of View (FoV)

TU Delft

Theory – vario-scale density function





Theory – vario-scale density function

- Finding how the density is transformed through the perspective translation matrix is out of the scope of this thesis

Theory – vario-scale density function

- Finding how the density is transformed through the perspective translation matrix is out of the scope of this thesis
- The results presented are created by using a function specific to each frame

TUDelft

Theory

- We know the density function
- We know what to keep and what to remove
- How do we do this?

TU Delft

Supporting research question

2. To what extend can a theoretical post-processing approach be created for vario-scale visualization of point cloud data sets?

TU Delft

Different approaches

- Random point removal
- Filtering bands point removal
- Circle packing point removal

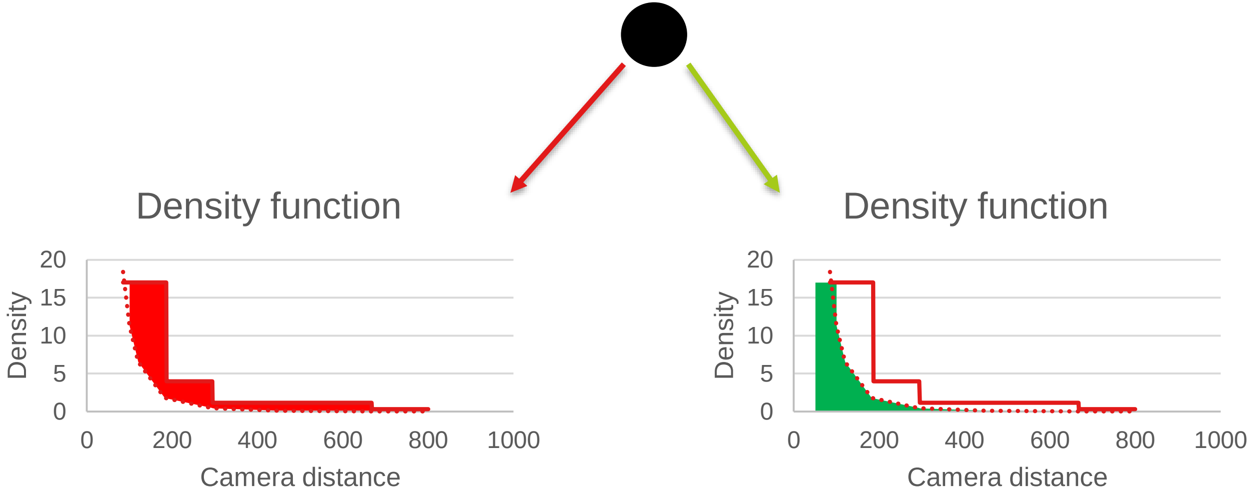
Random point removal

Random point removal

- Determine random value per point

Random point removal

- Determine random value per point
- Assed to threshold value



Random point removal

Algorithm 1 Random removal

```
import numpy as np
import random
import scipy

def random_removal(points, camera_parameters, density_function):
    """

    """
    camera_origin = camera_parameters.origin
    selected_points = set()
    kdtree = scipy.spatial.KDTree(points)

    for point in points:
        d = distance_from_camera(point, camera_origin)
        r = random(0,1)
        local_density = len(kdtree.query_ball_point(point, 0.5642))
        t = density_function(d) / local_density

        if: r => t:
            continue
        if r < t:
            selected_points.add(point)

    return selected_points
```

Random point removal

Algorithm 1 Random removal

```
import numpy as np
import random
import scipy

def random_removal(points, camera_parameters, density_function):
    """

    """
    camera_origin = camera_parameters.origin
    selected_points = set()
    kdtree = scipy.spatial.KDTree(points)

    for point in points:
        d = distance_from_camera(point, camera_origin)
        r = random(0,1)
        local_density = len(kdtree.query_ball_point(point, 0.5642))
        t = density_function(d) / local_density

        if: r => t:
            continue
        if r < t:
            selected_points.add(point)

    return selected_points
```

Random point removal

Algorithm 1 Random removal

```
import numpy as np
import random
import scipy

def random_removal(points, camera_parameters, density_function):
    """

    """
    camera_origin = camera_parameters.origin
    selected_points = set()
    kdtree = scipy.spatial.KDTree(points)

    for point in points:
        d = distance_from_camera(point, camera_origin)
        r = random(0,1)
        local_density = len(kdtree.query_ball_point(point, 0.5642))
        t = density_function(d) / local_density

        if: r => t:
            continue
        if r < t:
            selected_points.add(point)

    return selected_points
```

TUDelft

Random point removal - advantage

- Random selection

Random point removal - disadvantage

- Creation of global kd-tree to compute local density

Random point removal - disadvantage

- Creation of global kd-tree to compute local density
- Computationally heavy because of per point computations:
  - Distance to the camera
  - Random value
  - Local density
  - Threshold

TU Delft

Filtering bands point removal

TU Delft

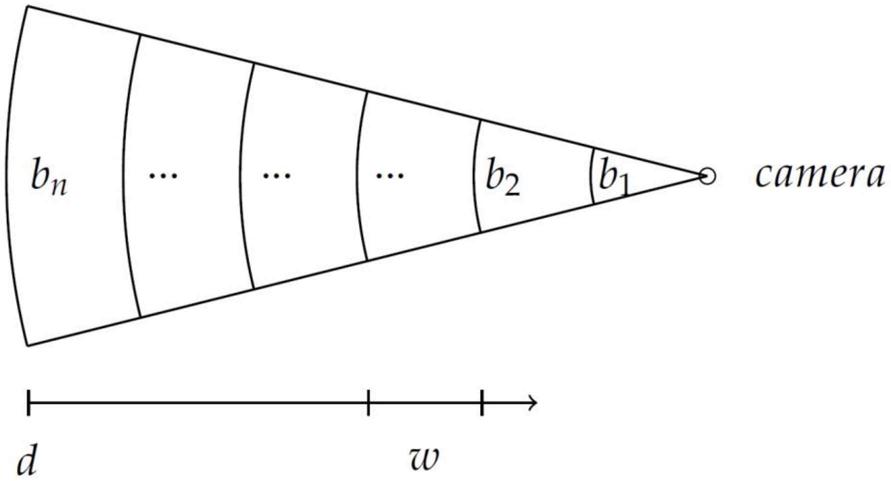
Filtering bands point removal

- Create bands extending outward from camera origin

TU Delft

Filtering bands point removal

- Create bands extending outward from camera origin



Filtering bands point removal

- Create bands extending outwards from camera origin
- Filter points on local density

TU Delft

Filtering bands point removal

Algorithm 2 Filtering bands

```
def filtering_bands(points,
                    camera_parameters,
                    density_function,
                    width = 1.0):
    """

    """
    distance = camera_parameters.distance
    origin = camera_parameters.origin
    selected_points = set()

    for i in range(distance / width):
        band_radius_center = i * width + 0.5 * width
        local_points_array = points_in_band(origin,
                                            band_radius_center,
                                            points)
        allowed_points = density_function(band_radius_center)

        selected_points.add(local_points_array[allowed_points]

    return selected_points
```

TU Delft

Filtering bands - advantage

- No local density calculation per point

Filtering bands - advantage

- No local density calculation per point
- Per band density calculation

Filtering bands - advantage

- No local density calculation per point
- Per band density calculation
- No global computation needed such as kd-tree building

Filtering bands - disadvantage

- (Small) discrete steps

Filtering bands - disadvantage

- (Small) discrete steps
- Requires band creation

Filtering bands - disadvantage

- (Small) discrete steps
- Requires band creation
- Per band clipping operation

```
for i in range(distance / width):
    band_radius_center = i * width + 0.5 * width
    local_points_array = points_in_band(origin,
                                        band_radius_center,
                                        points)
    allowed_points = density_function(band_radius_center)

    selected_points.add(local_points_array[allowed_points]

return selected_points
```

TU Delft

Circle packing point removal

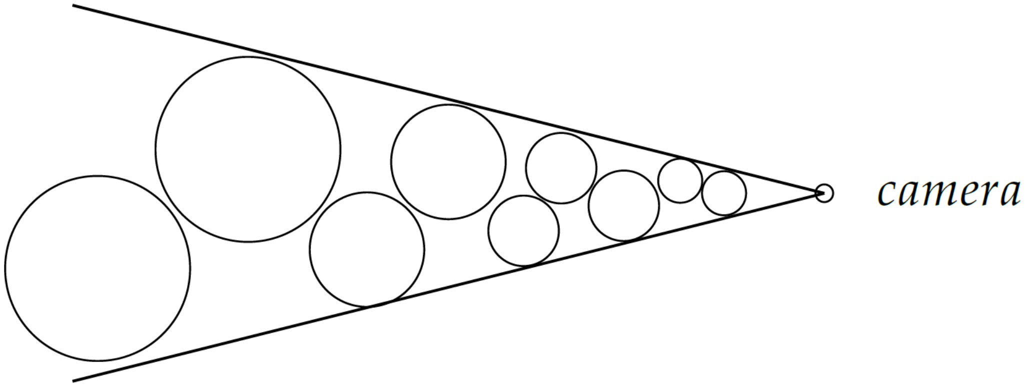
Circle packing point removal

- Determine region in which no two points can exist

TU Delft

Circle packing point removal

- Determine region in which no two points can exist



Circle packing point removal

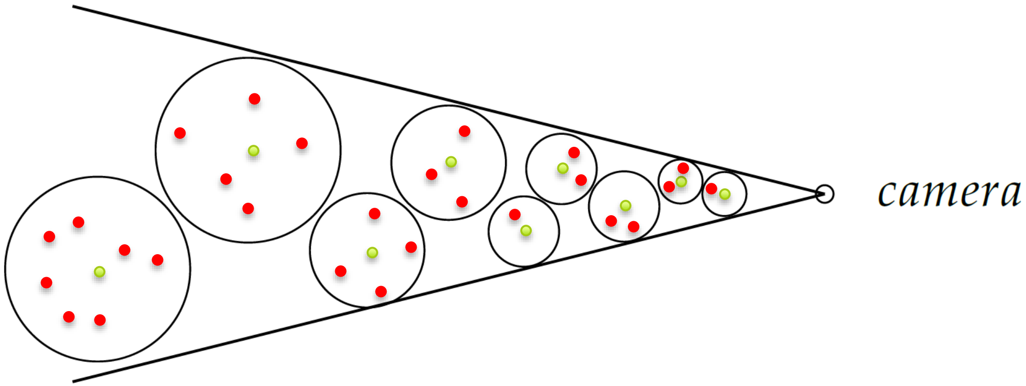
- Determine region in which no two points can exist
- Discard all neighbours in that region as potention points

TUDelft

Circle packing point removal

- Determine region in which no two points can exist
- Discard all neighbours in that region as potential points to evaluate
- Based on the circle packing study of arrangement of circles in a given surface, so that no overlap occurs

Circle packing point removal



- Based on the circle packing study of arrangement of circles in a given surface, so that no overlap occurs

TU Delft

Circle packing point removal

Algorithm 3 Circle packing

```
import scipy

def circle_packing(points, camera_parameters, density_function):
    """

    """
    used_points = [False] * len(points)
    selected_points = set()

    kdtree = scipy.spatial.KDTree(points)

    for j, point in enumerate(points):
        if used(j) == True:
            continue

        d = distance_from_camera(point, camera_parameters)
        r = radius_function(density_function(d))
        nn = kdtree.query_ball_point(point, r)

        for i in nn:
            used[i] = True

        selected_points.add(point)

    return selected_points
```

TU Delft

Circle packing - advantage

- Only computations needed for accepted points

Circle packing - advantage

- Only computations needed for accepted points

```
for j, point in enumerate(points):
    if used(j) == True:
        continue

    d = distance_from_camera(point, camera_parameters)
    r = radius_function(density_function(d))
    nn = kdtree.query_ball_point(point, r)

    for i in nn:
        used[i] = True

    selected_points.add(point)

return selected_points
```

TU Delft

Circle packing - disadvantage

- Requires the creation of a kd-tree for fast nn-search

TU Delft

Theoretic summary

- Determine the density function
- Eliminate points according to this density function
- Using one of the three described methods

TU Delft

Presentation

- Introduction
- Research
- Theory
- **Implementation**

TU Delft

Supporting research question

3. Which point-cloud processing framework is best suited to create a proof-of-concept vario-scale visualization platform for the AHN2 point cloud?

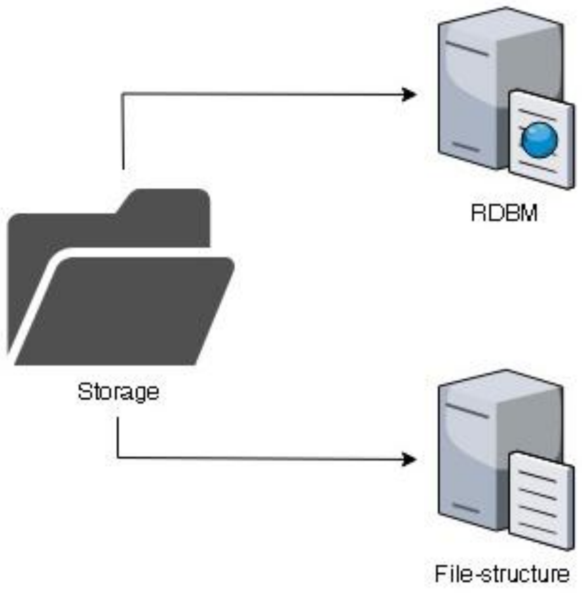
TU Delft

Web based visualization framework

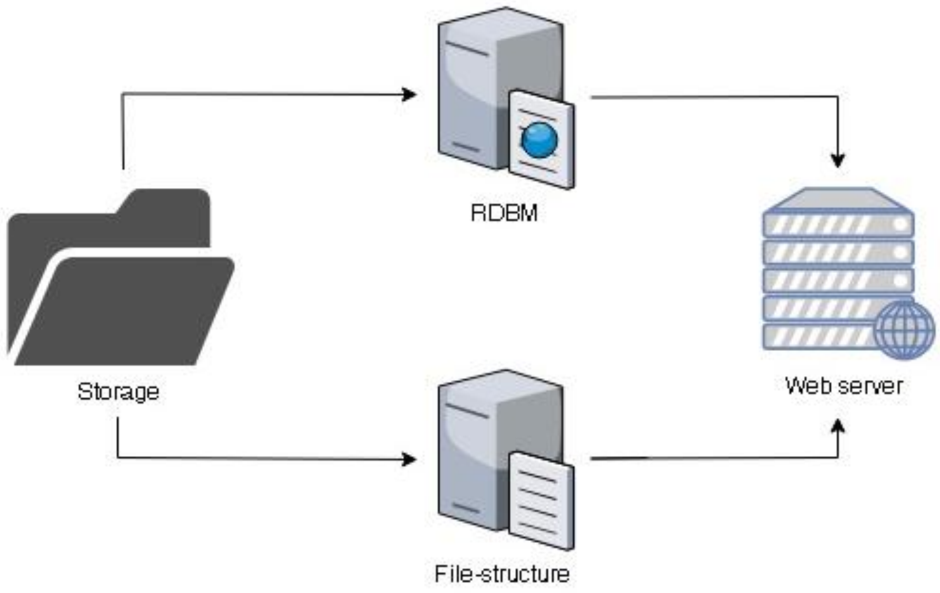
How does web based point cloud visualization currently work?

Web based visualization framework

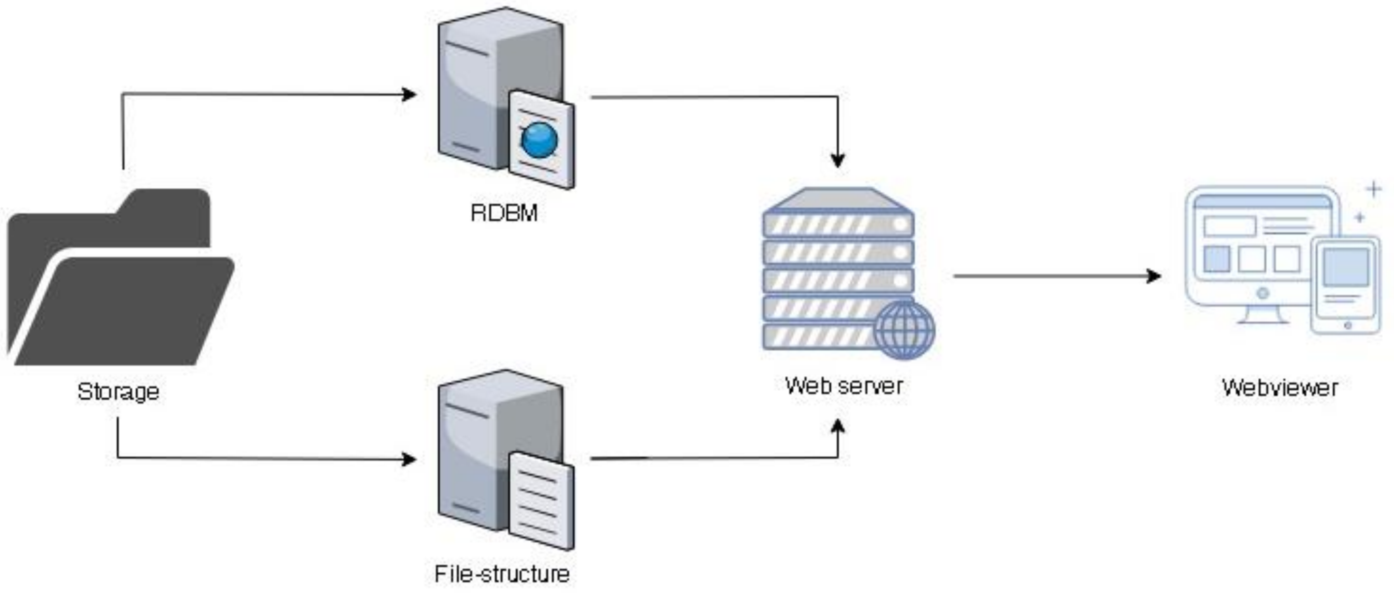
Web based visualization framework



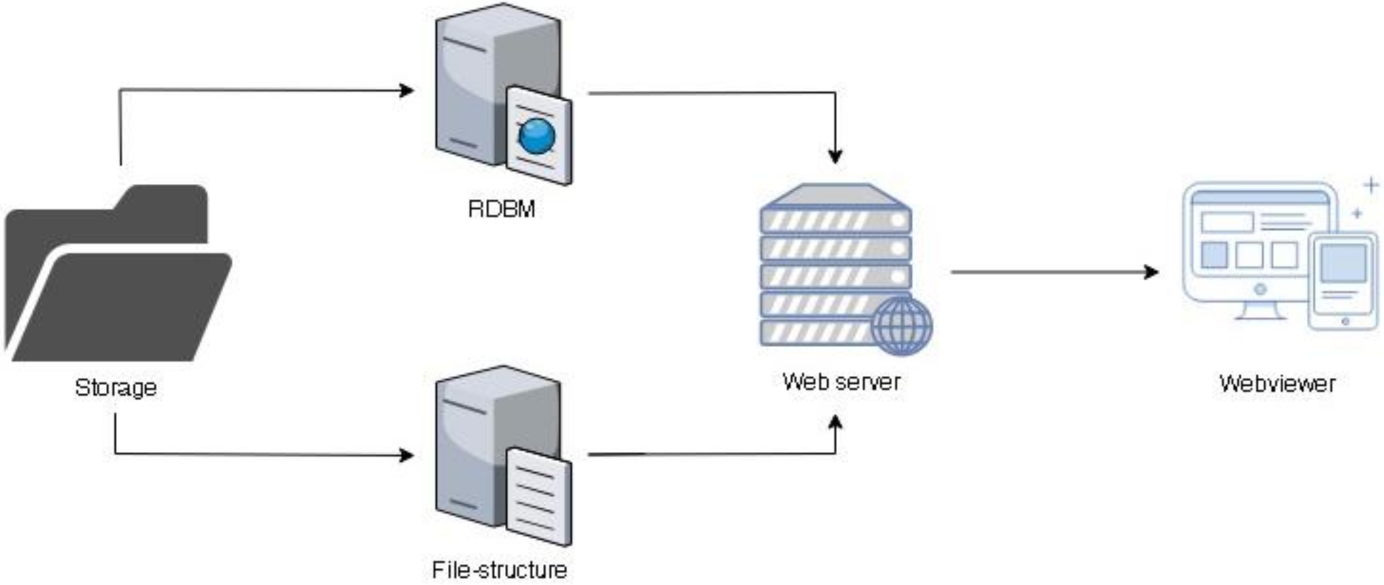
Web based visualization framework



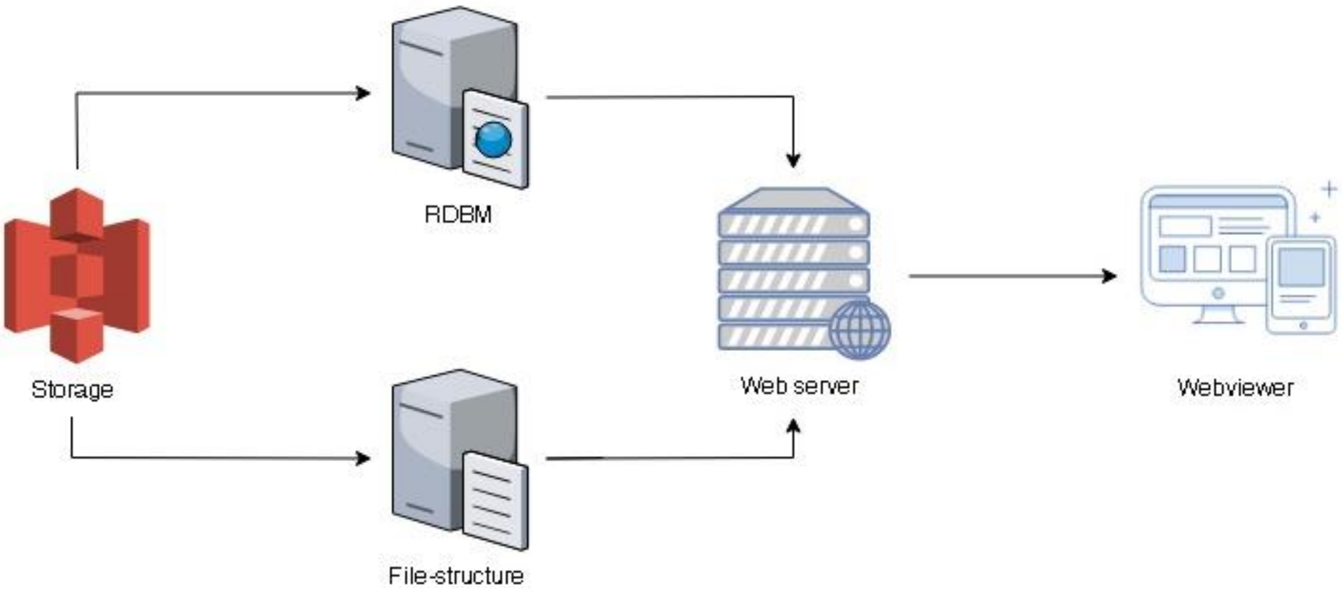
Web based visualization framework



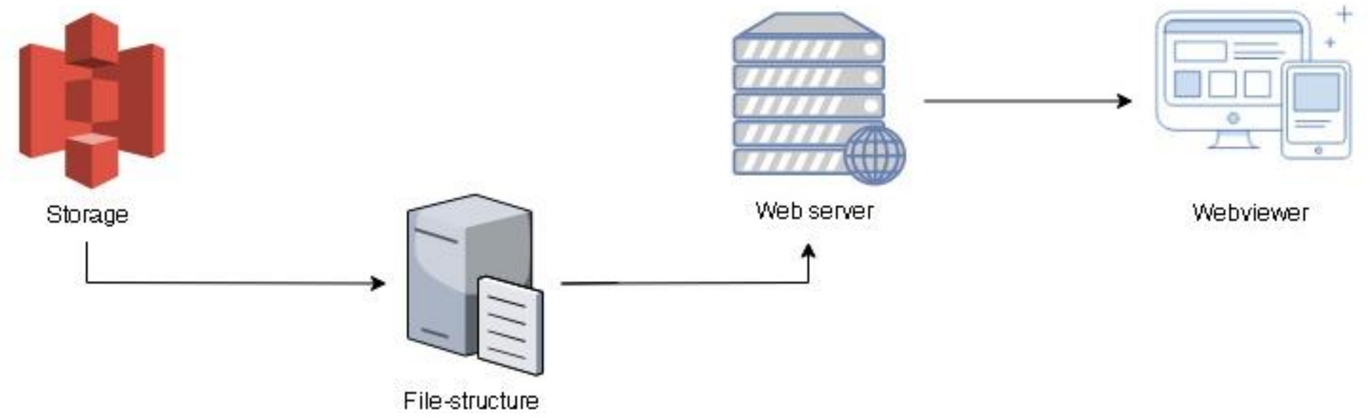
Chosen framework



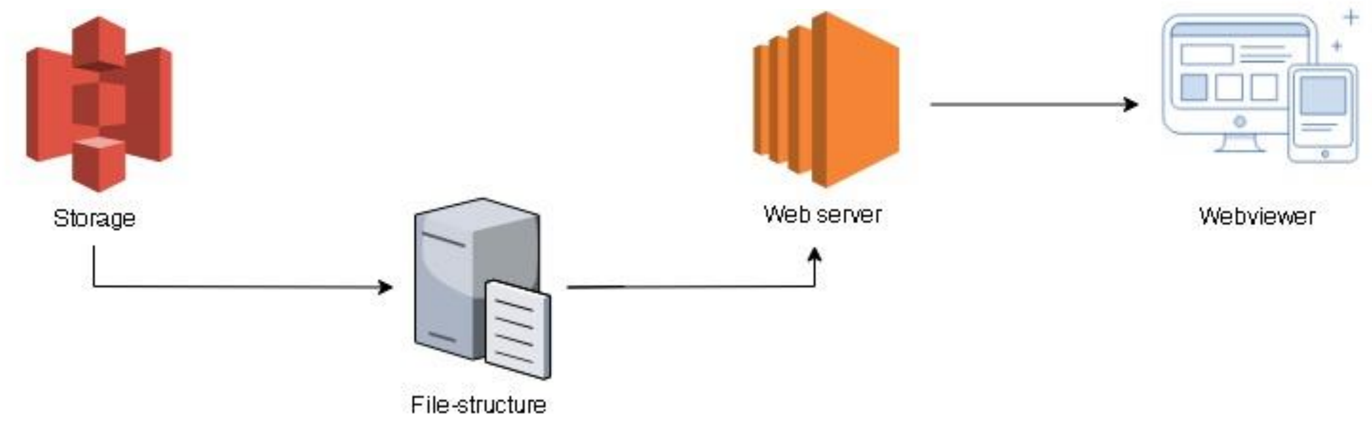
Chosen framework



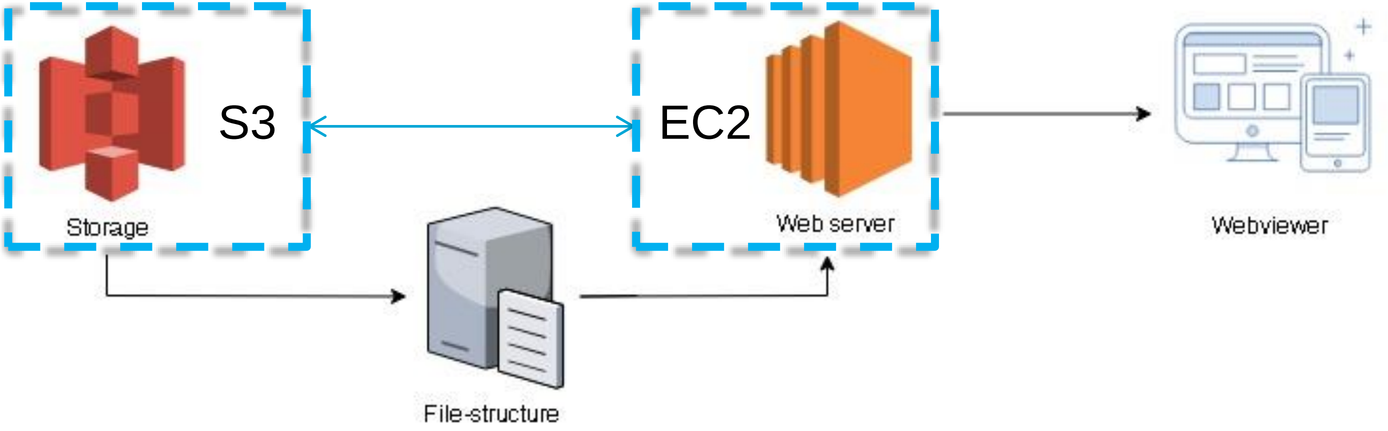
Chosen framework



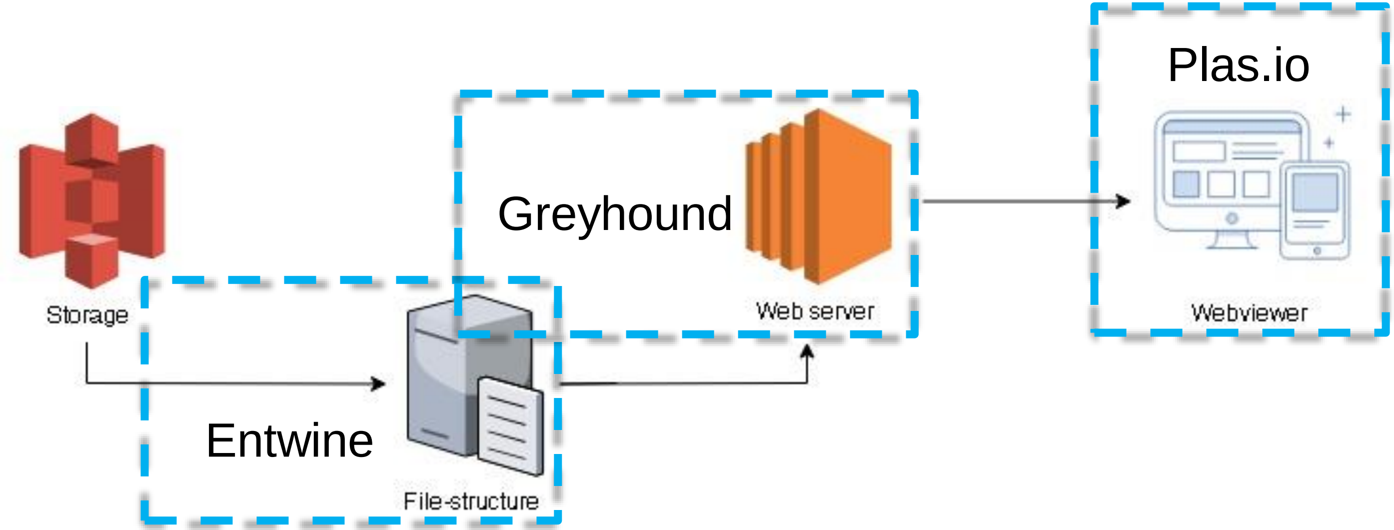
Chosen framework



Chosen framework



Chosen framework



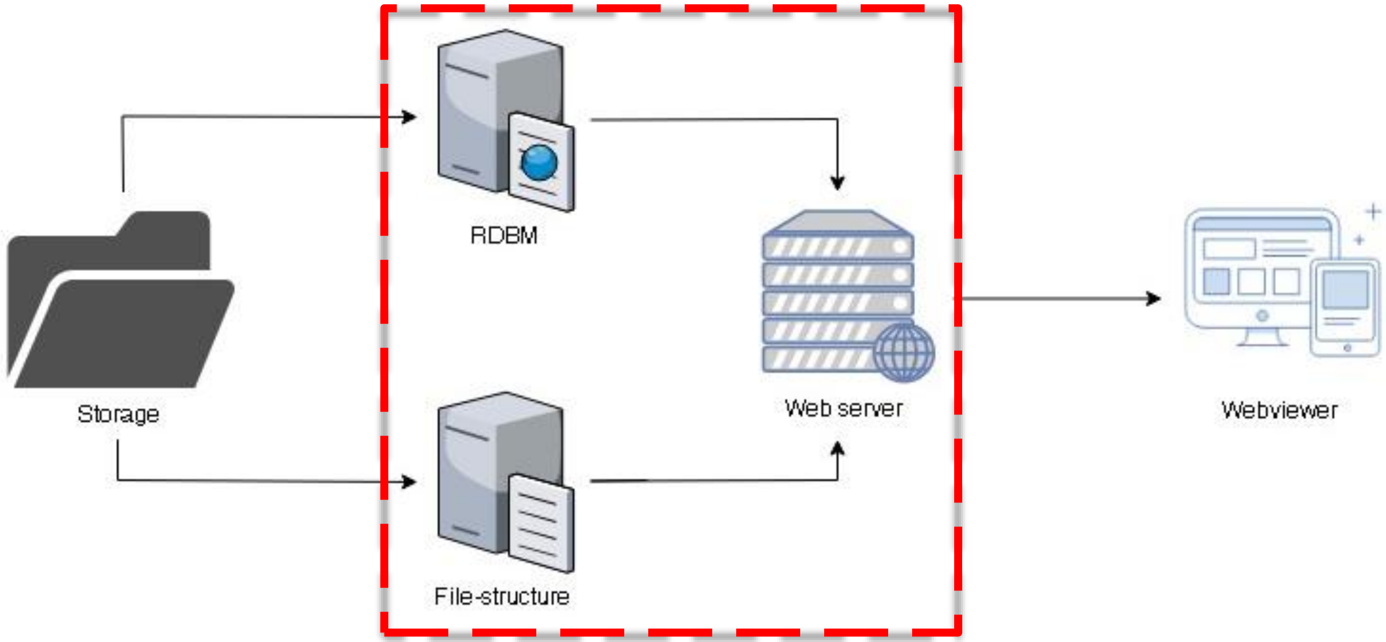
Supporting research questions

4. To what extent cant the theoretical approach be implemented in an existing point cloud web visualization framework?

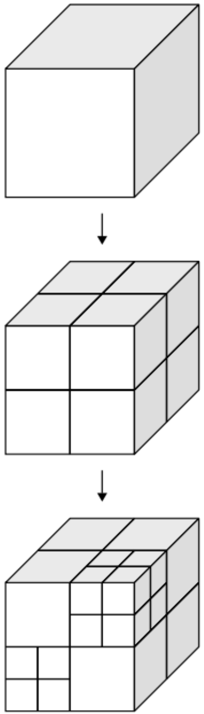
TU Delft

Where is the issue?

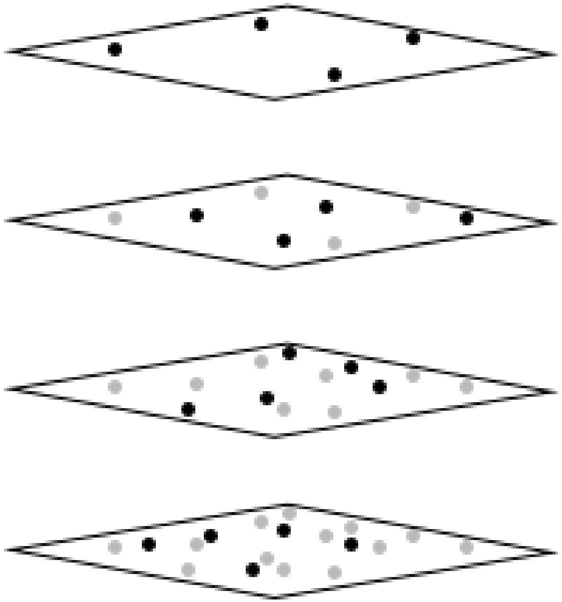
Where is the issue?

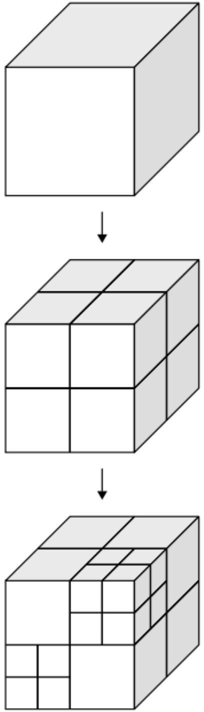


Where is the issue?

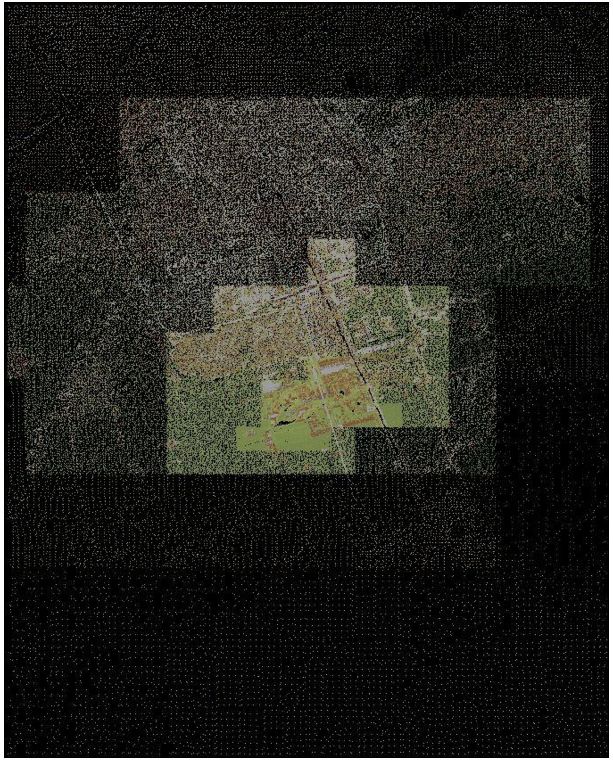


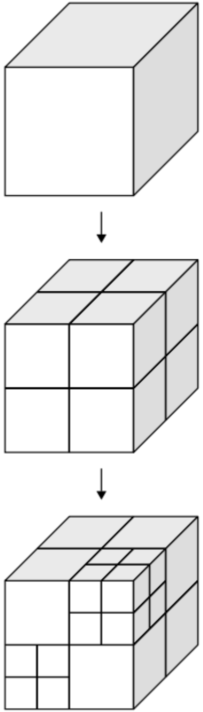
Where is the issue?



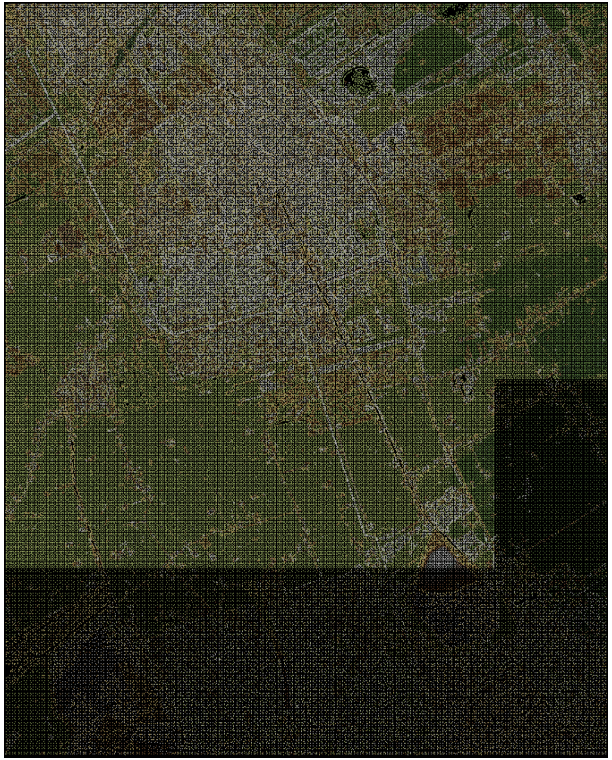


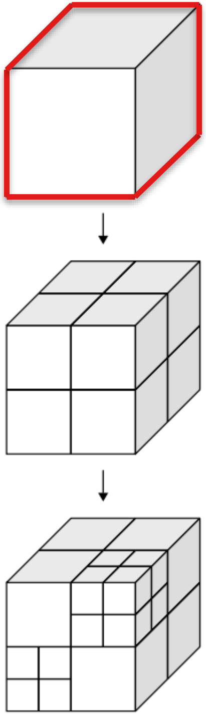
Where is the issue?



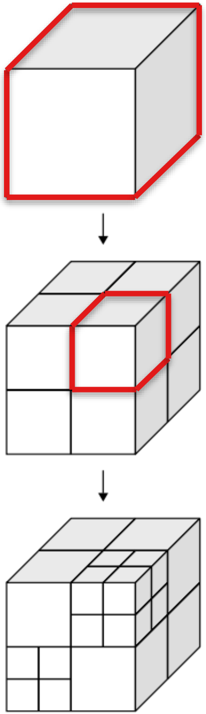


Where is the issue?

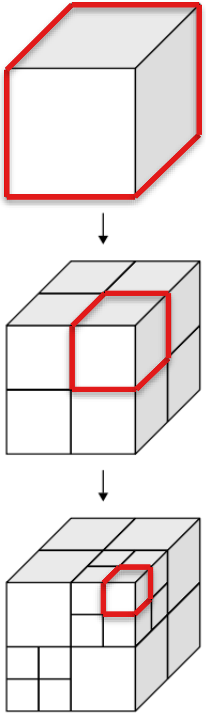




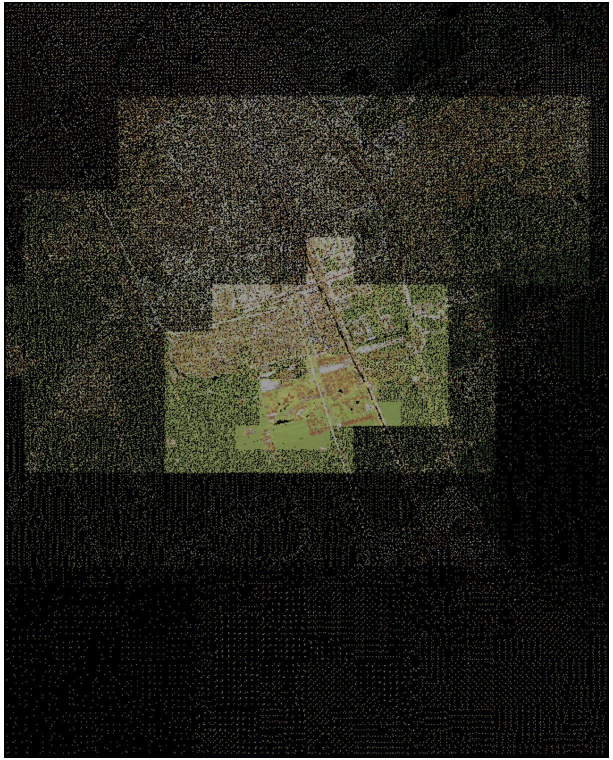
Where is the issue?

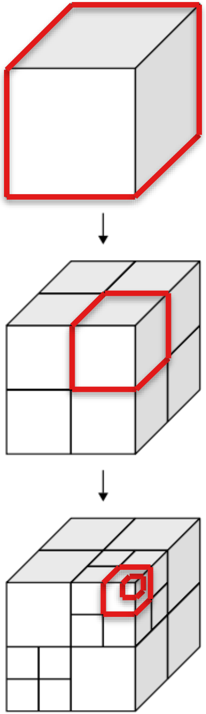


Where is the issue?

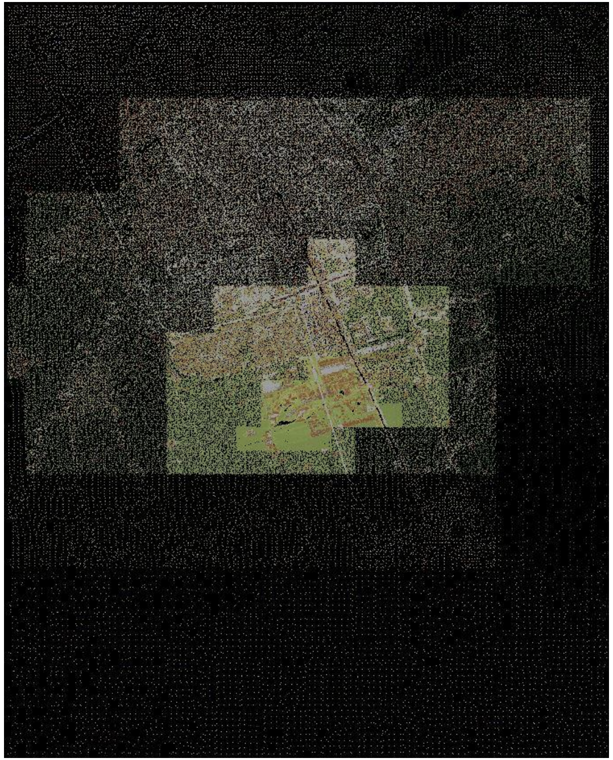


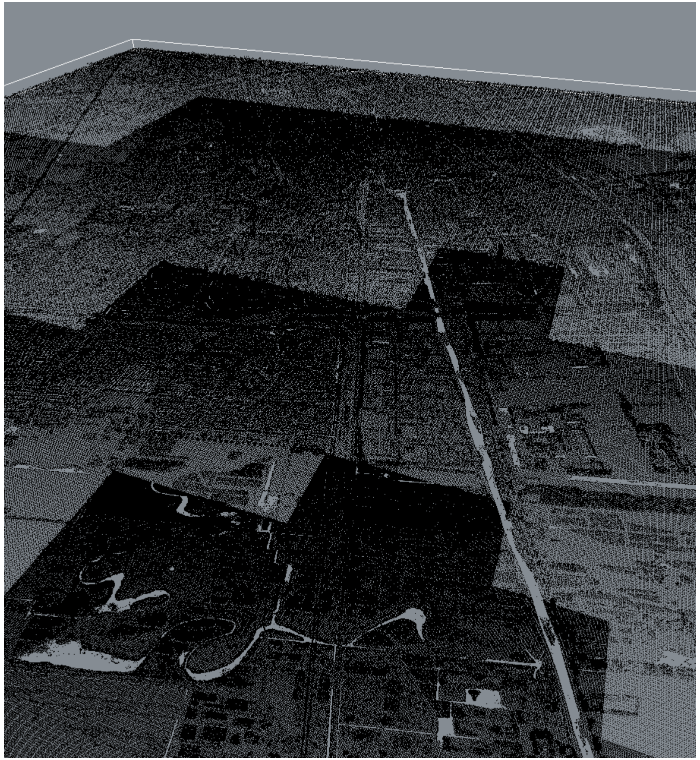
Where is the issue?





Where is the issue?

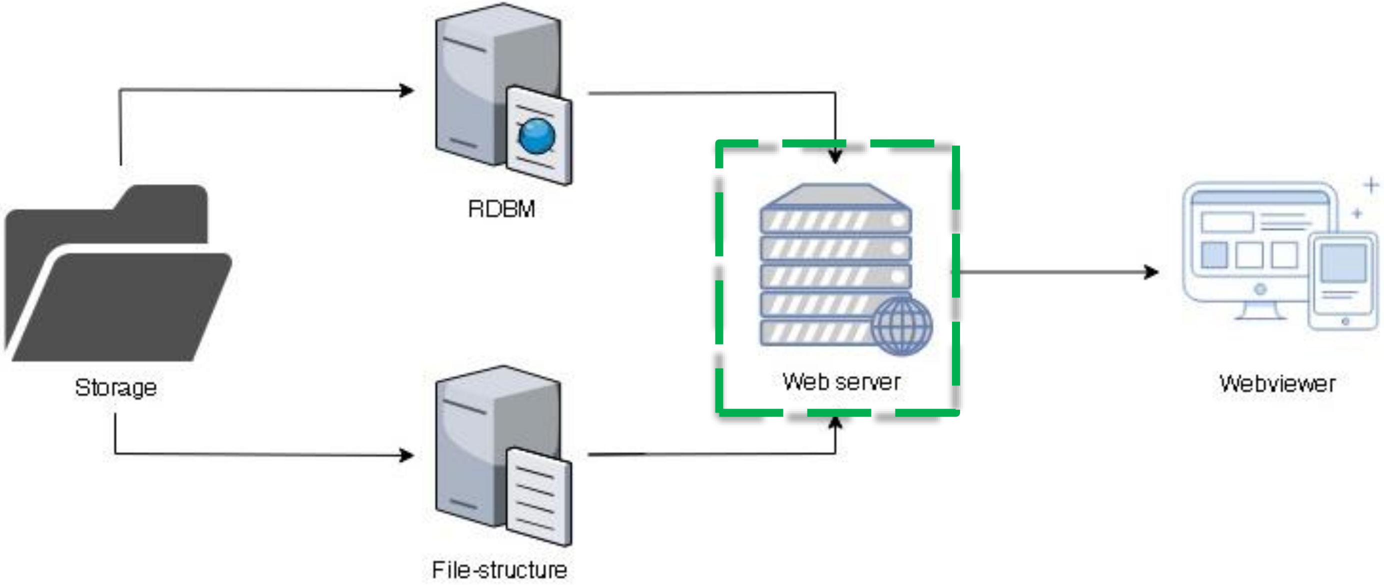




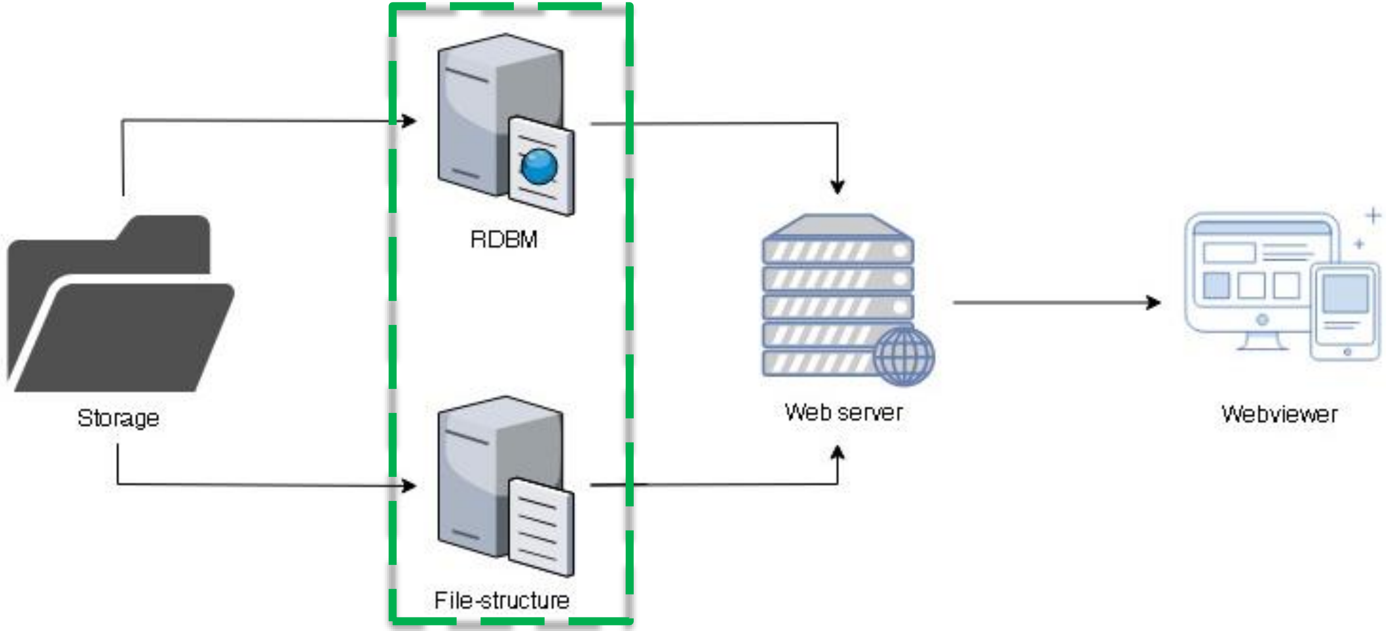
Where is the solution?

TU Delft

Where is the solution?



Why not here?



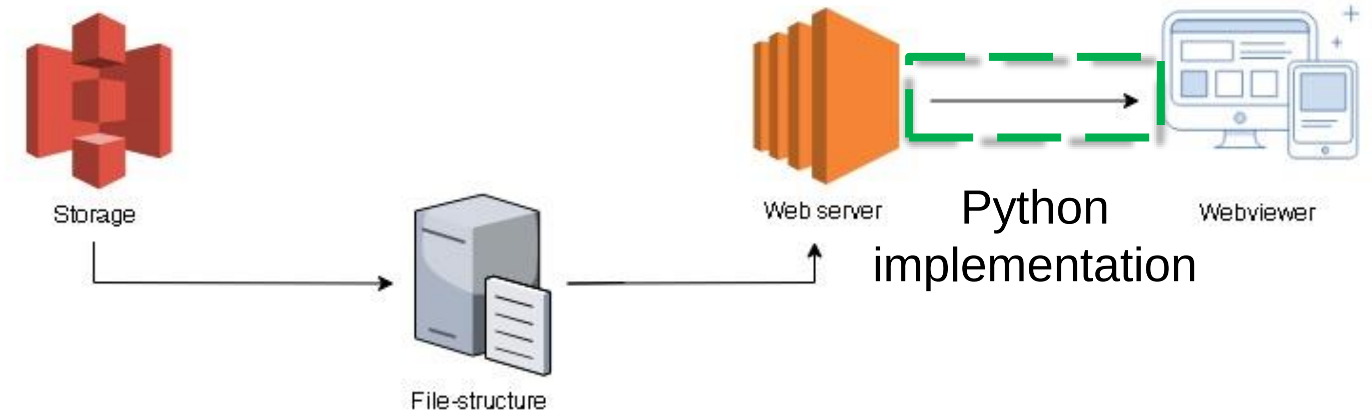
Why not here?

- Current octree index most efficient for web based querying

Why not here?

- Current octree index most efficient for web based querying
- Creating a new indexing method for vario-scale visualization of point clouds is outside of the scope of this thesis

Method


Storage
File-structure
Web server
Python implementation
Webviewer

Python implementation

- Circle packing method

TUDelft

Python implementation

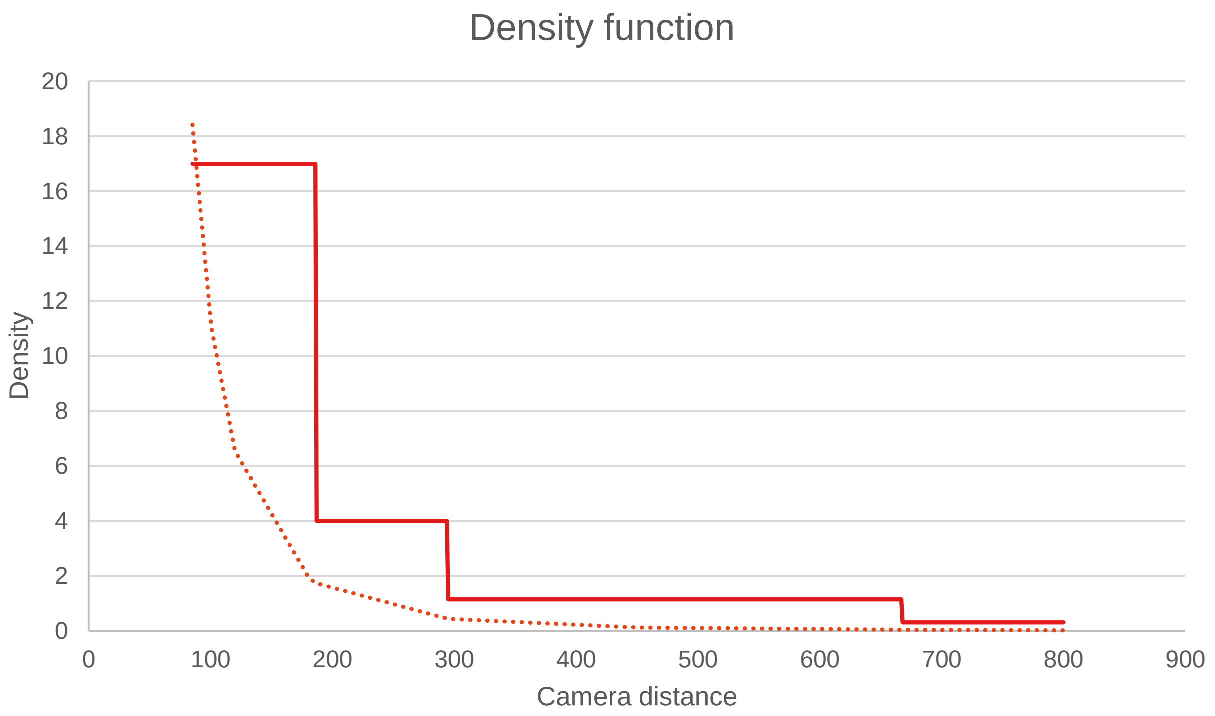
- Circle packing method
- Density function to match the frame’s density jumps

Python implementation

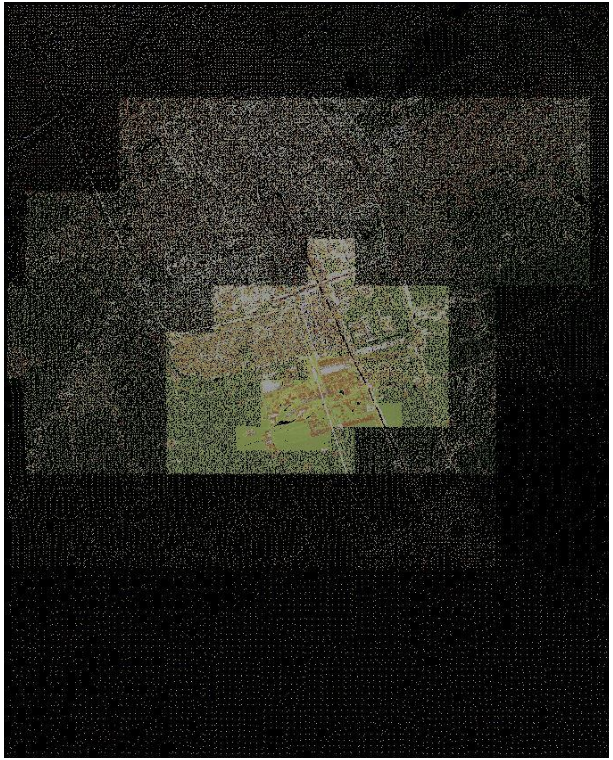
- Circle packing method
- Density function to match the frame's density jumps
- Density formula = 1/(0,7*(0,005*camera_distance)^3)

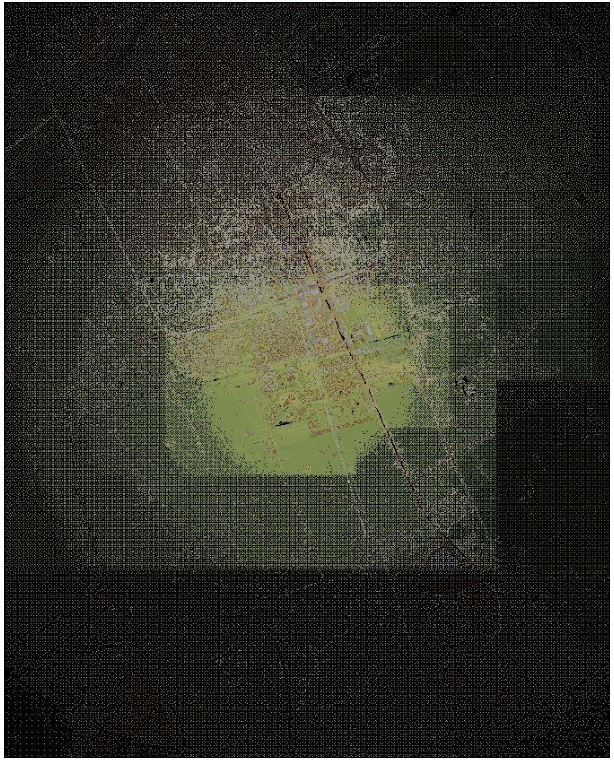
TUDelft

Results

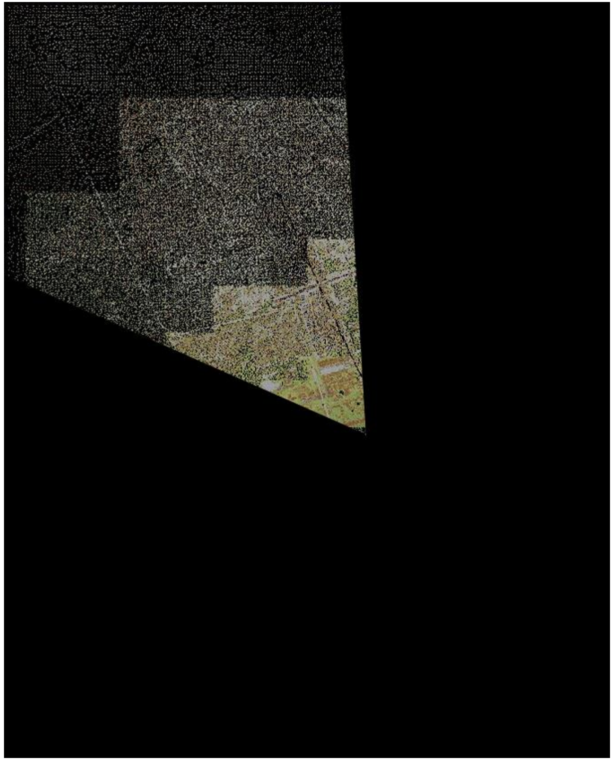

Density function
Density
Camera distance
0
2
4
6
8
10
12
14
16
18
20
0
100
200
300
400
500
600
700
800
900

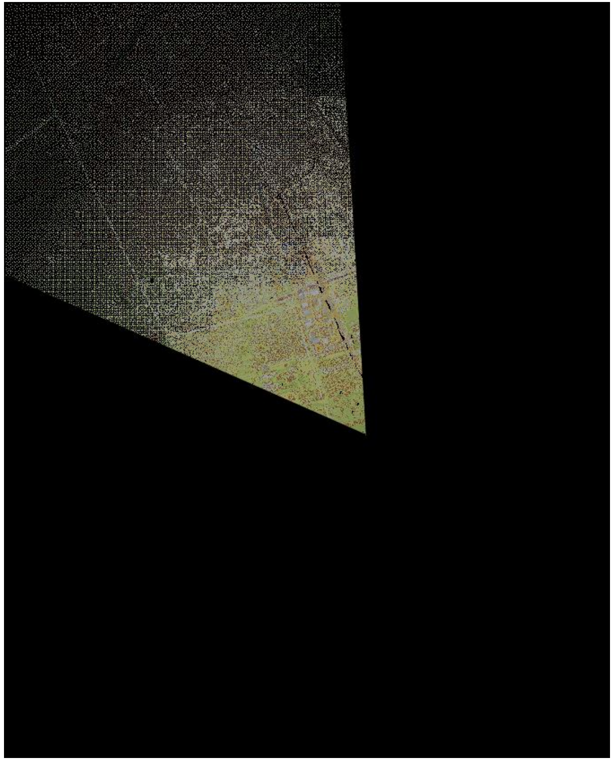
Results – top-down



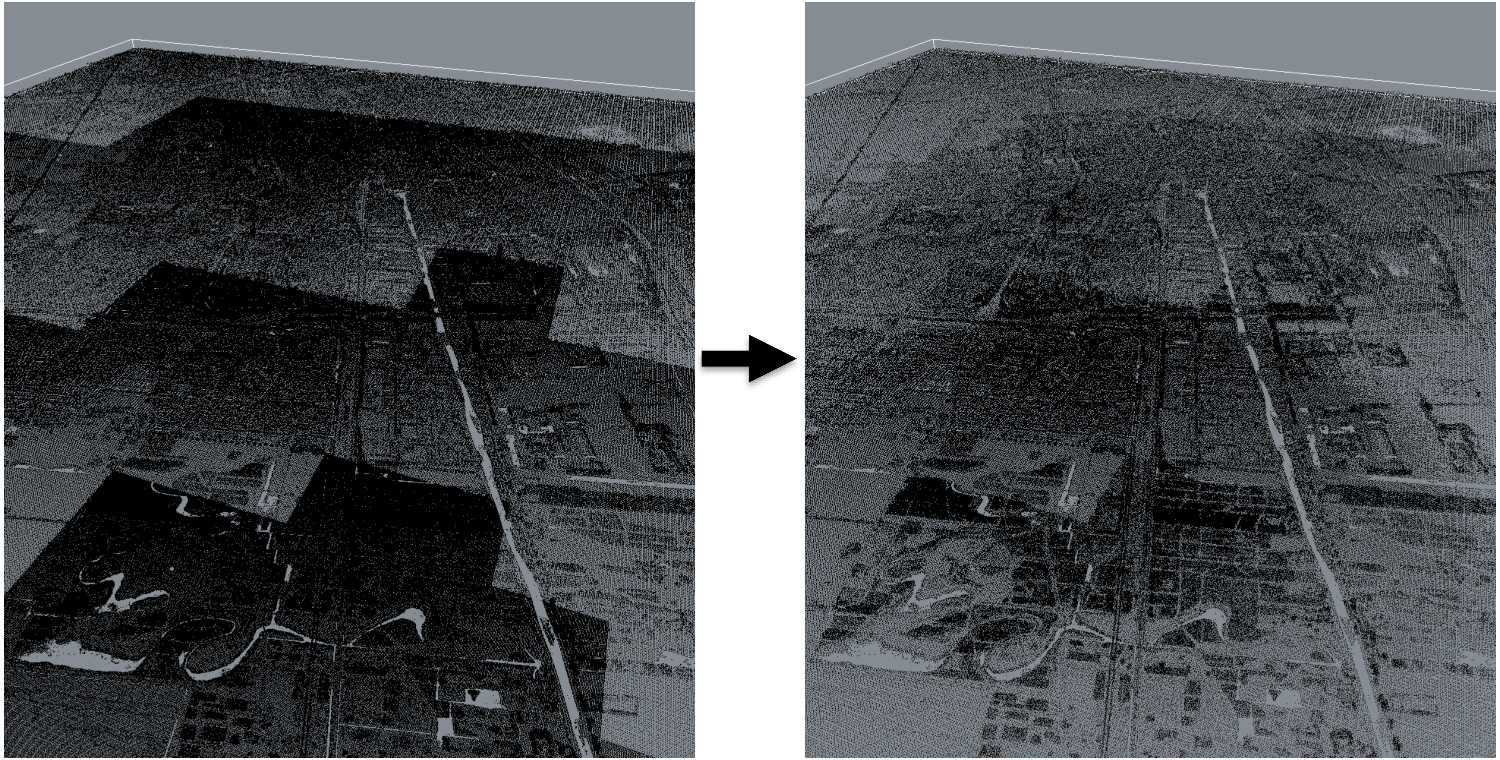


Results – top-down camera frustum

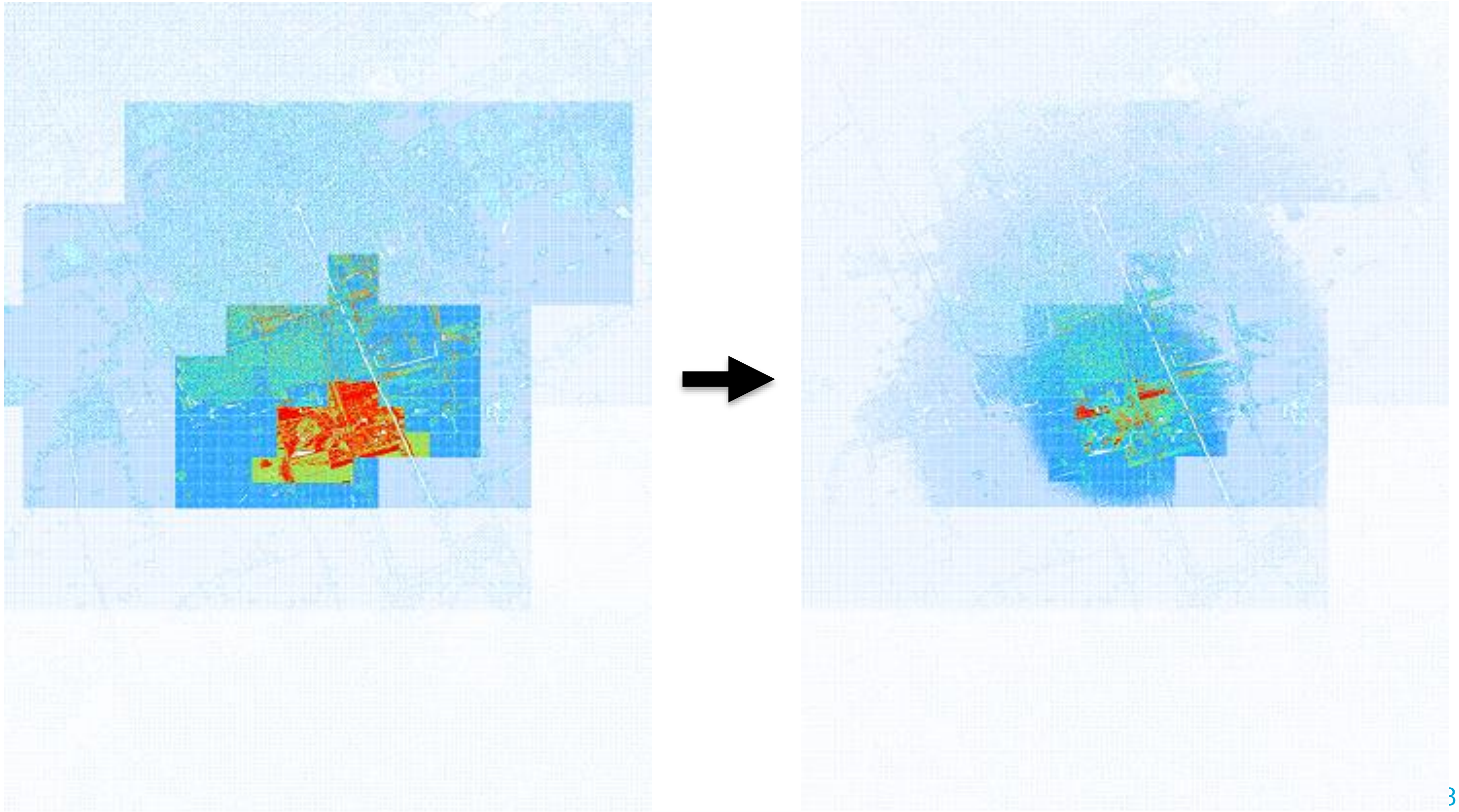




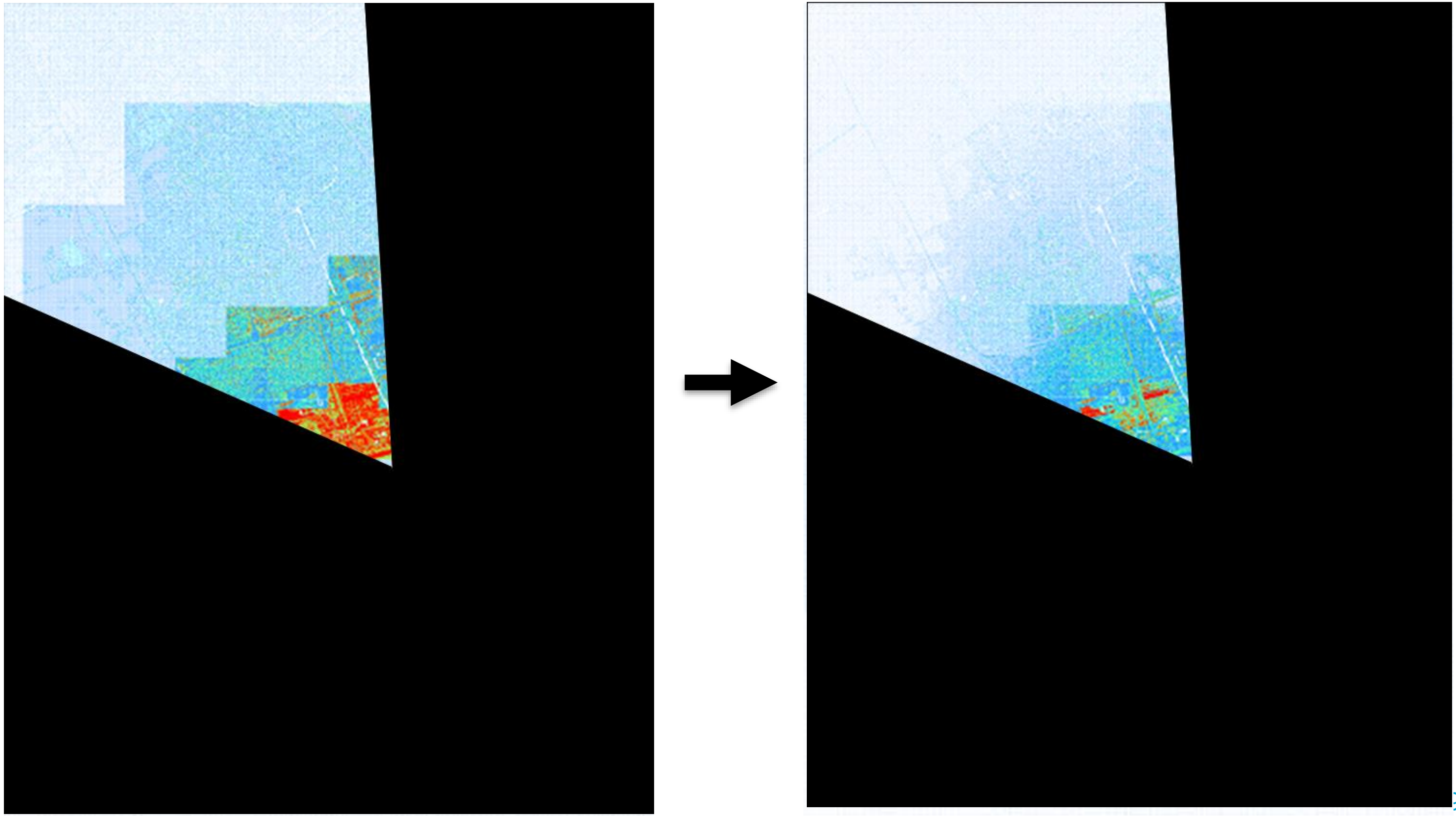
Results – top-down camera frustum



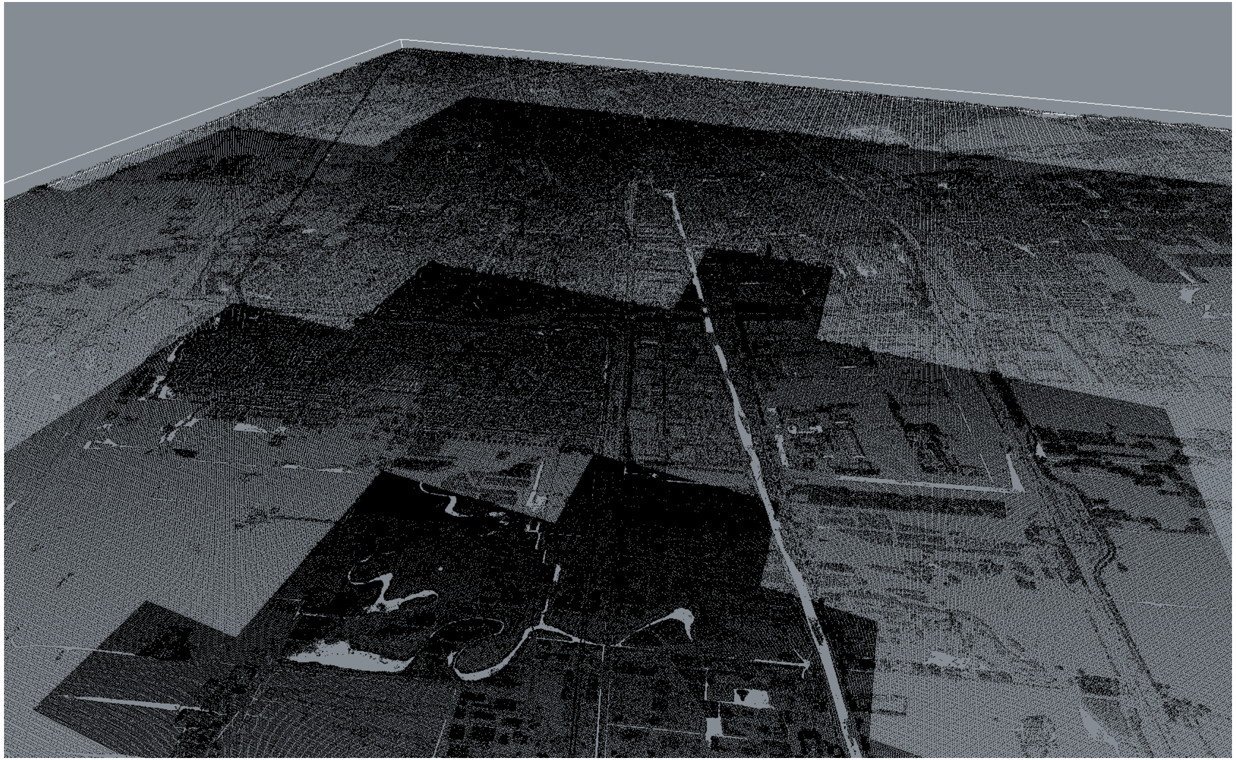
Results – density



Results – density camera frustum

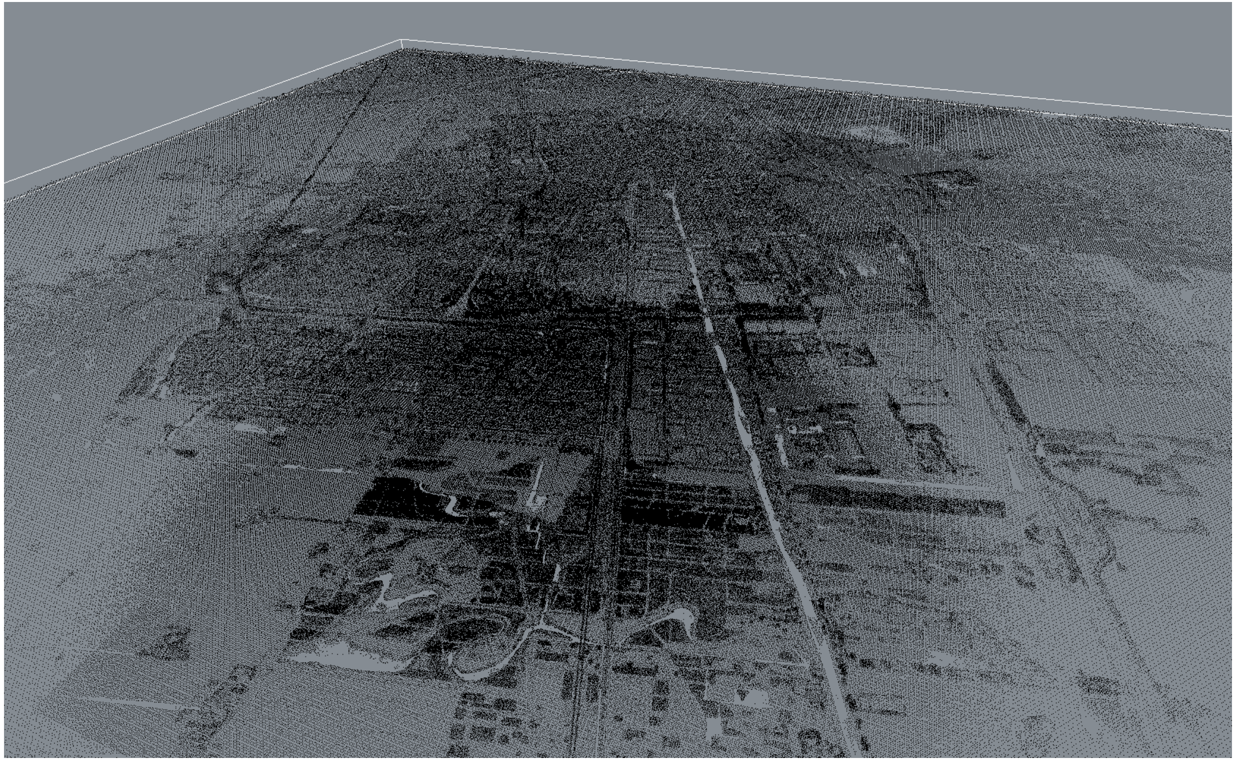


Results – perspective original

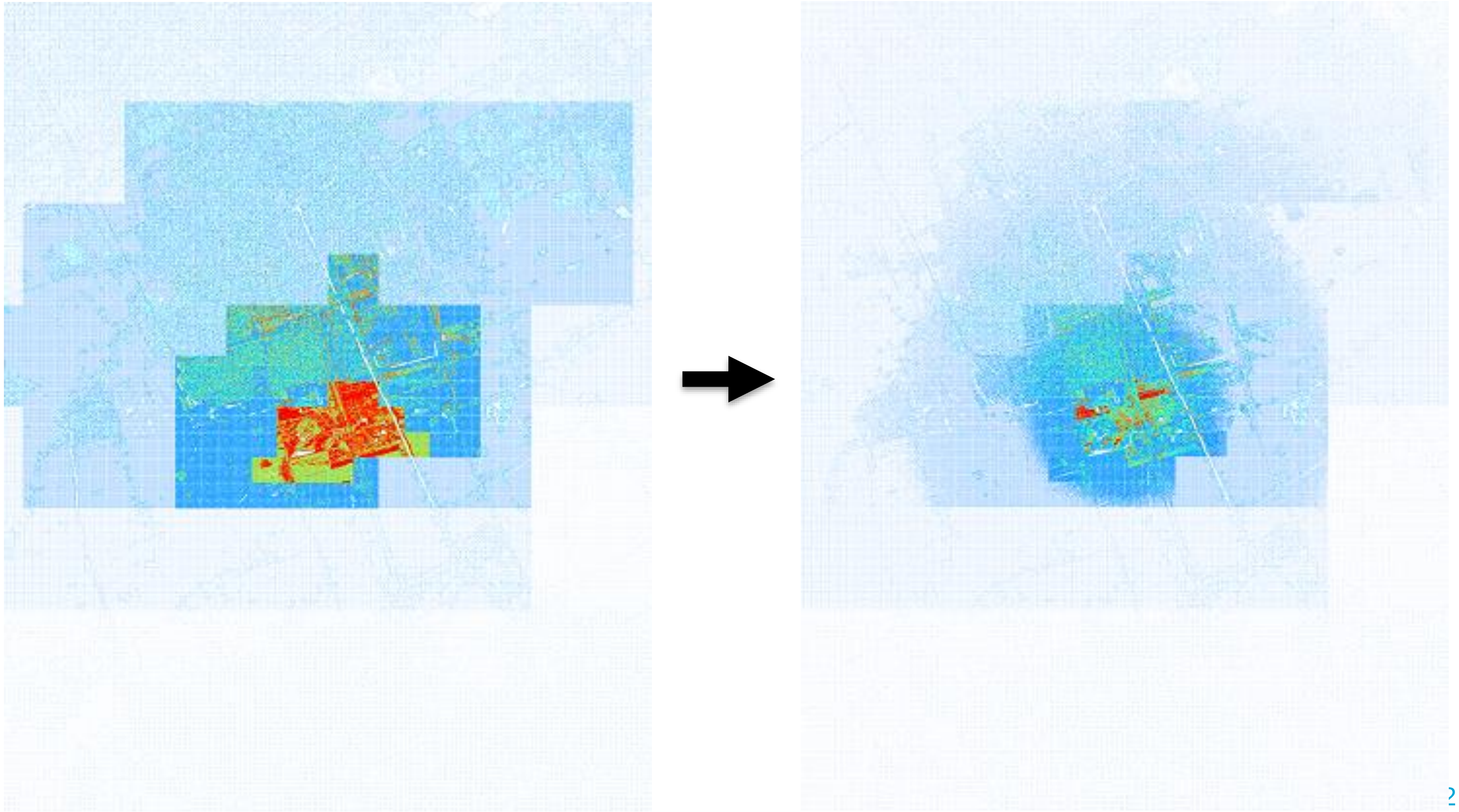


TU Delft

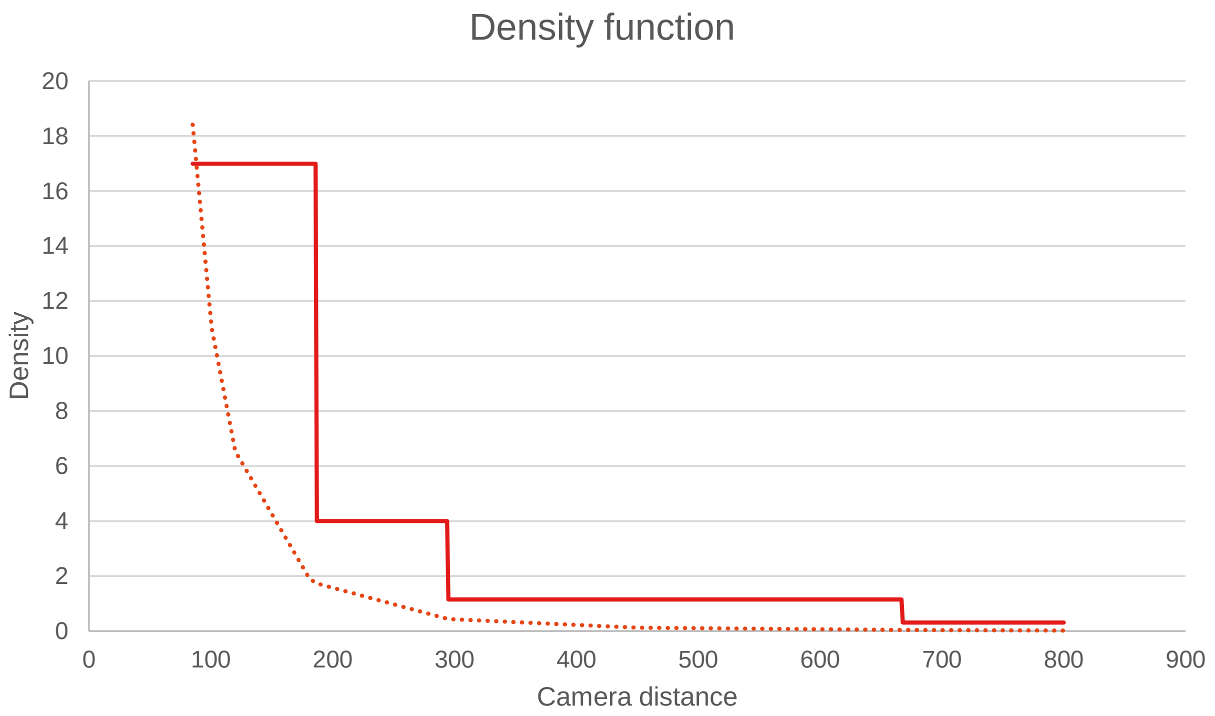
Results – perspective vario-scale



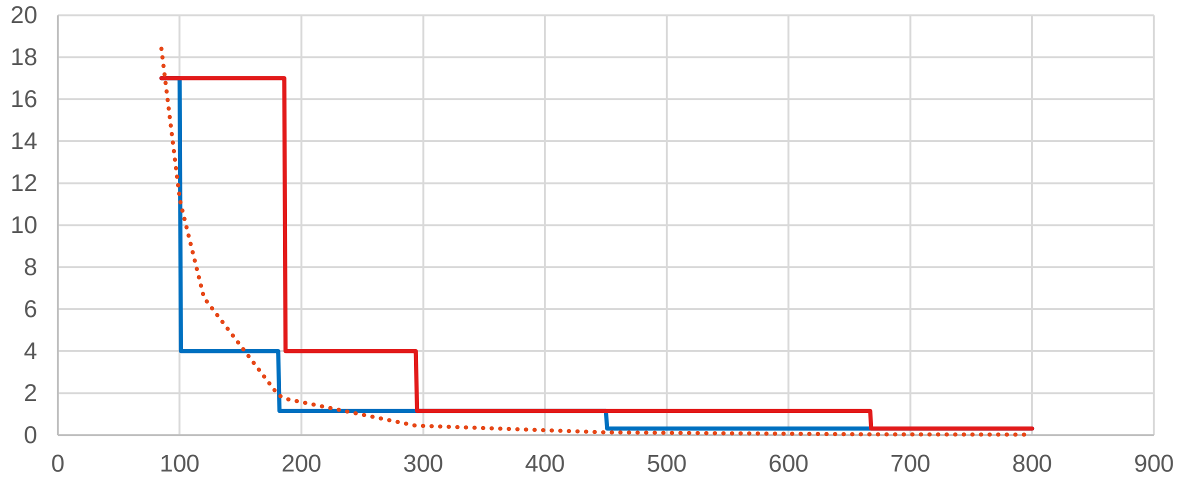
Results – analysis



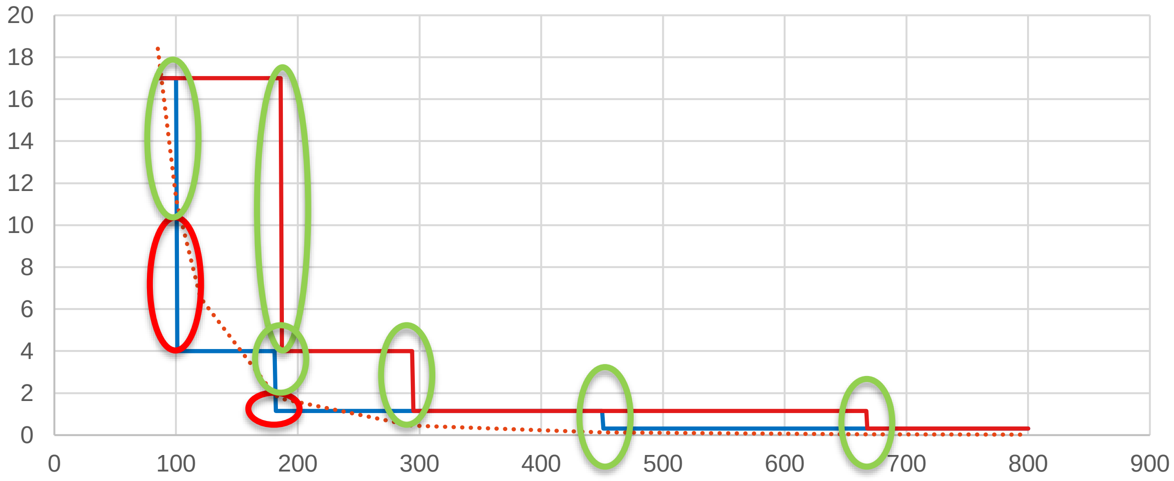
Results


Density function
Density
20
18
16
14
12
10
8
6
4
2
0
0
100
200
300
400
500
600
700
800
900
Camera distance

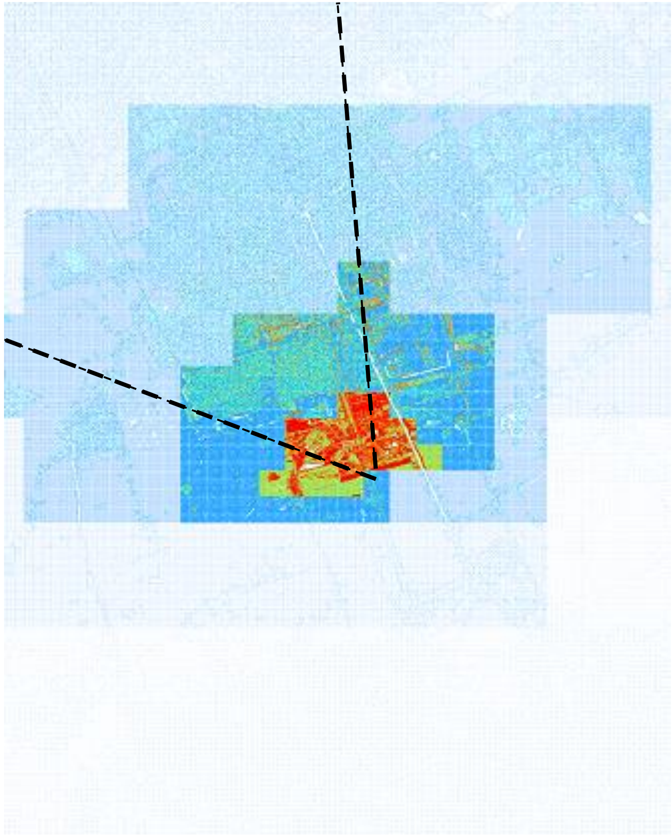
Results – analysis

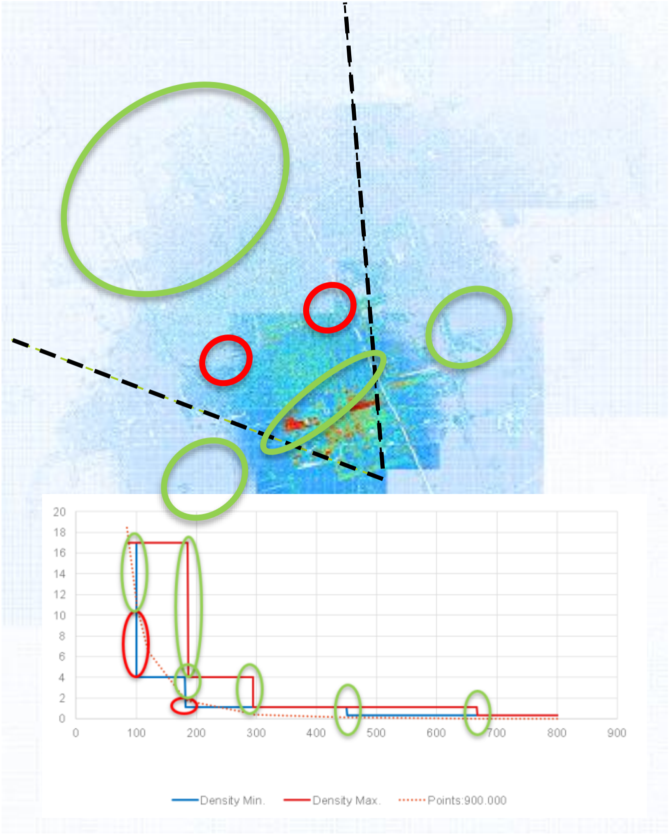


Results – analysis

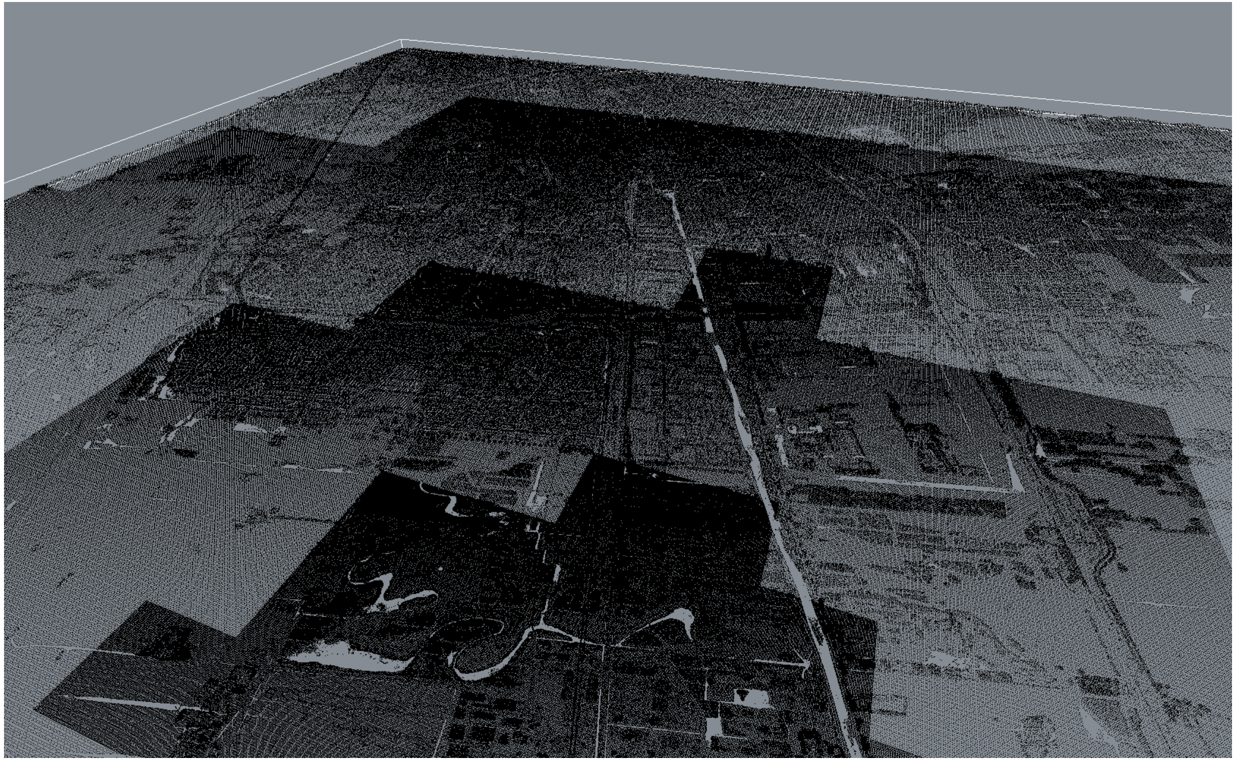


Results – analysis

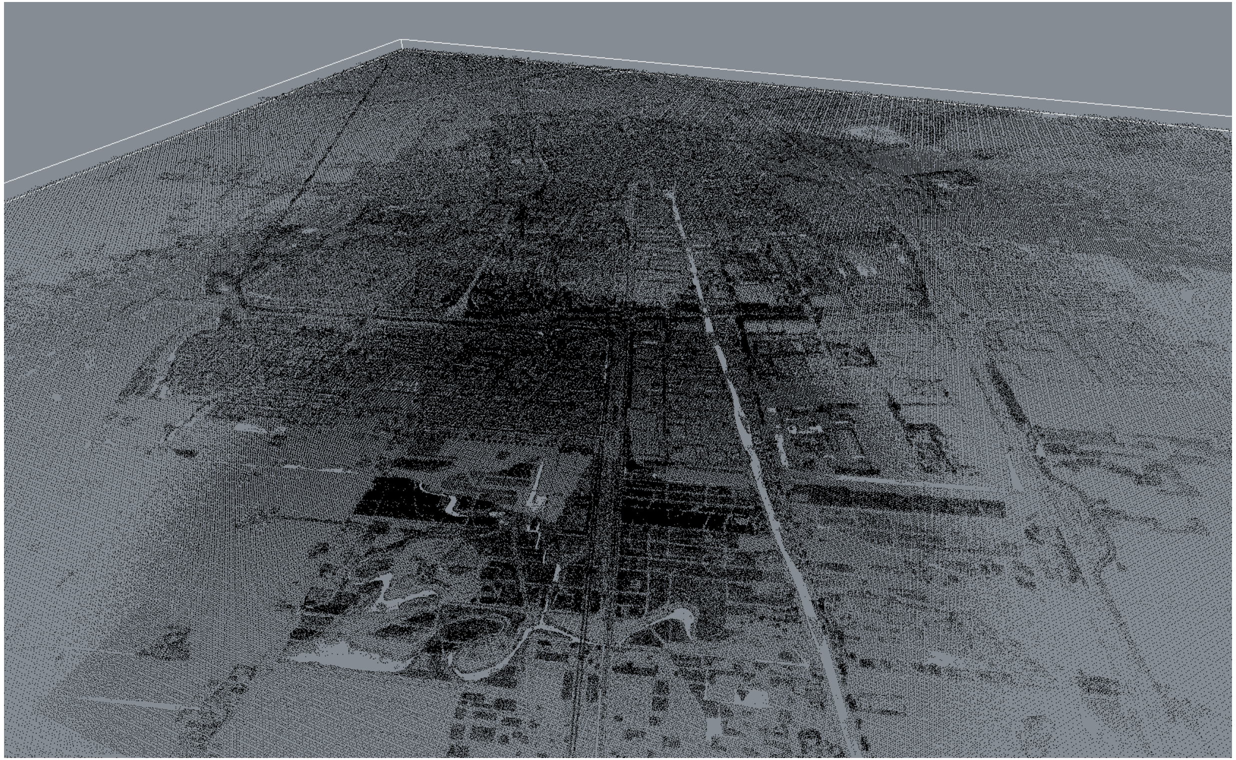




Results – perspective original

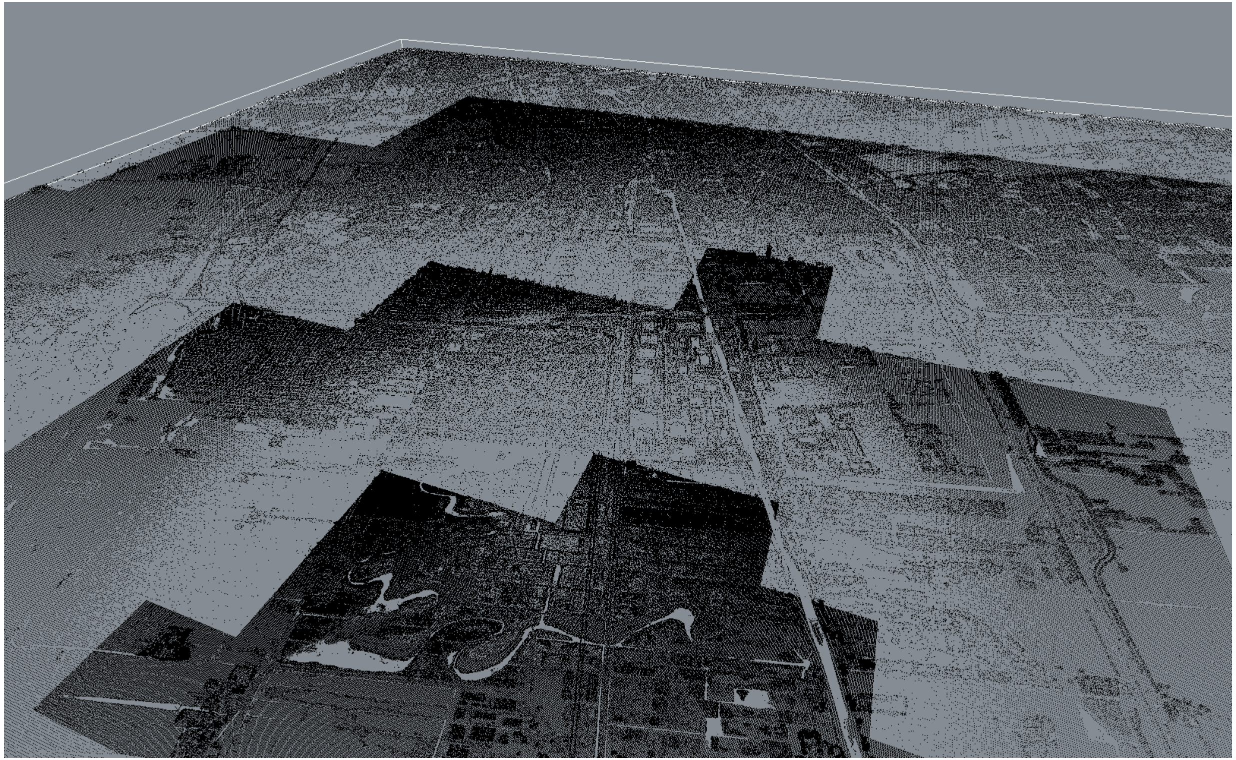


Results – perspective vario-scale



TU Delft

Results – removed points



Findings

- The theoretical importance of the perspective transformation matrix for vario-scale visualization

Findings

- The theoretical importance of the perspective transformation matrix for vario-scale visualization
- Circle packing method to set the upper limit for density in a data set, dependant on the distance from the camera origin

Findings

- The theoretical importance of the perspective transformation matrix for vario-scale visualization
- Circle packing method to set the upper limit for density in a data set, dependant on the distance from the camera origin
- Improvements in processing speed are needed for 30fps support

TU Delft

Future work

- 4D index implementation

Future work

- 4D index implementation
- Determine a global density formula relationship

Future work

- 4D index implementation
- Determine a global density formula relationship
- Practical implementation improvements
  - Implementation in Greyhound
  - GPU calculations
  - Addition in stead of subtraction
  - Process only affected octree blocks, not all

TU Delft

Conclusion

- Cicle packing method is used to enforce the density formula for vario-scale visualization

TU Delft

Conclusion

- Cicle packing method is used to enforce the density formula for vario-scale visualization
- It is possible to visualize the AHN2 data set in a vario-scale manner with existing web viewer architecture

TU Delft

Conclusion

- Cicle packing method is used to enforce the density formula for vario-scale visualization
- It is possible to visualize the AHN2 data set in a vario-scale manner with existing web viewer architecture
- The current solution is not fast enough for 30+ fps performance

P5 presentation

Student:	Jippe van der Maaden
Main mentor:	Prof.dr.ir. P.J.M. van Oosterom
Second mentor:	Dr.ir. B.M. Meijers
Examiner:	Dr. H.M.H. van der Heijden

TU Delft