LandXML parser

Ver 1.1

Taewook Kang
laputa99999@gmail.com

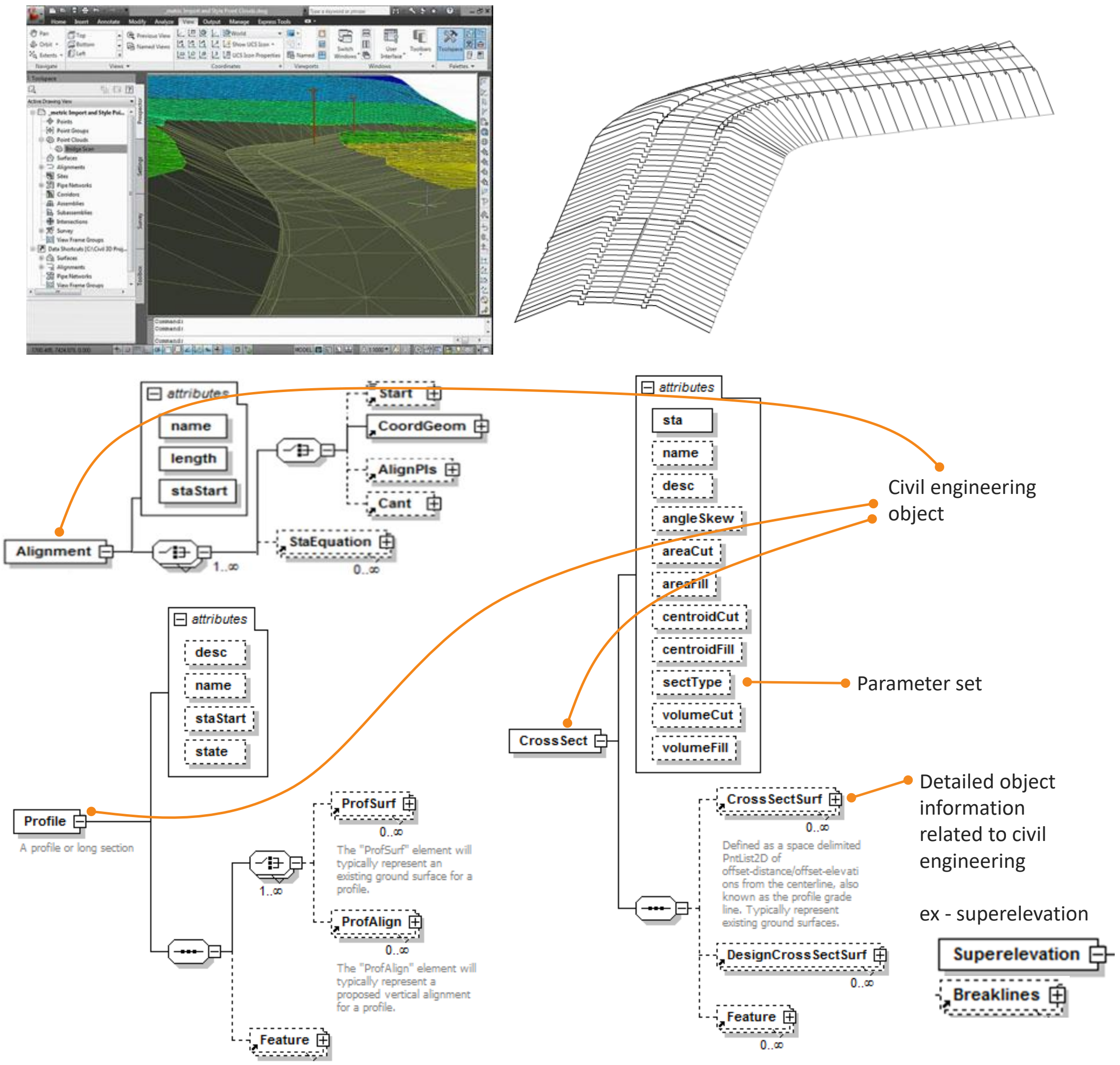
Date - 2025.3

Table of Contents

1. Overview

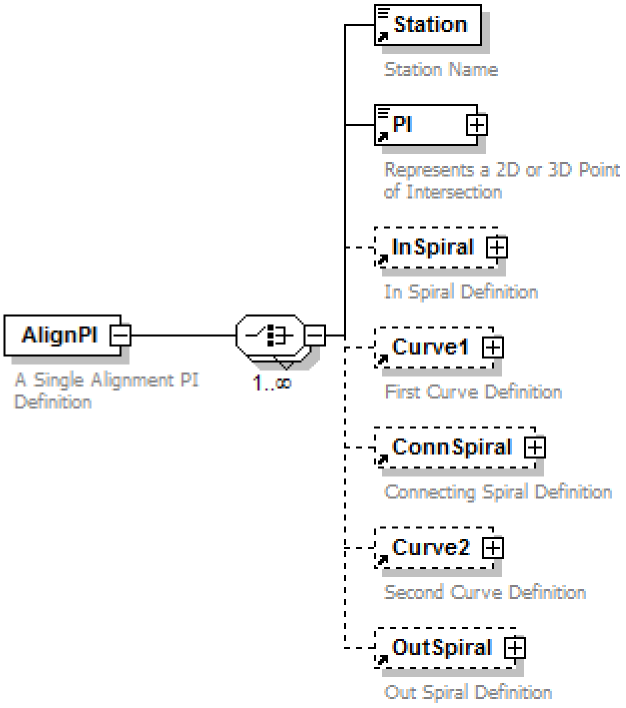
LandXML is an industry standard data exchange in the field of civil engineering developed by US DOT EAS-E and Autodesk. It was developed in 1999 at LandXML.org (http://www.landxml.org/) and is used as a civil engineering neutral format. there is.

This format includes all information related to civil engineering, that is, most models such as surveying, road design, complex site, and digital terrain model (DTM), and focuses on civil engineering object information interoperability, as shown in Figure 2.


attributes
name
length
staStart
Start
CoordGeom
AlignPIs
Cant
StaEquation
0..∞
Alignment
1..∞
attributes
sta
name
desc
angleSkew
areaCut
areaFill
centroidCut
centroidFill
sectType
volumeCut
volumeFill
Civil engineering object
Parameter set
CrossSect
attributes
desc
name
staStart
state
Profile
A profile or long section
ProfSurf
0..∞
The "ProfSurf" element will typically represent an existing ground surface for a profile.
1..∞
ProfAlign
0..∞
The "ProfAlign" element will typically represent a proposed vertical alignment for a profile.
Feature
CrossSectSurf
0..∞
Defined as a space delimited PntList2D of offset-distance/offset-elevations from the centerline, also known as the profile grade line. Typically represent existing ground surfaces.
DesignCrossSectSurf
0..∞
Feature
0..∞
Detailed object information related to civil engineering
ex - superelevation
Superelevation
Breaklines

LandXML representation object example

This format was developed as readable and extensible XML as follows. The following is the AlignPI structure, which is a control point that forms the shape of alignment. On the left is a graphical representation of AlignPI, and on the right is the XML file format in which it is actually saved. As shown in the figure, object information such as AlignPI consists of several nodes below it, in this case station, intersection (PI), entry spiral curve definition (InSpiral – in the case of road alignment, it becomes a clothoid), It consists of curve definition (Curve1, 2) and outspiral curve definition (OutSpiral).

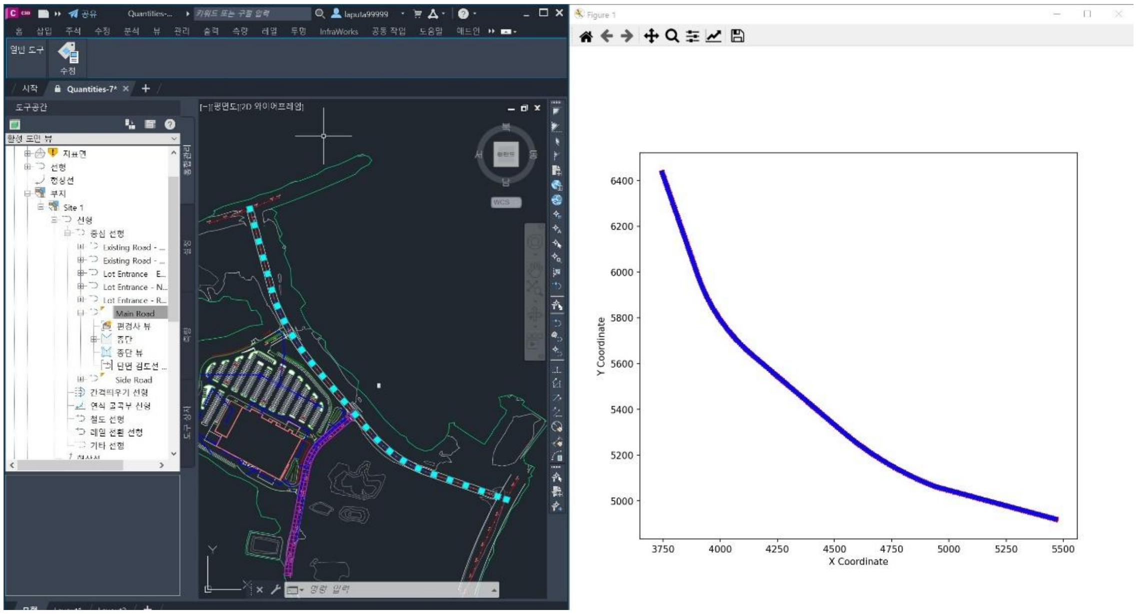


```
<xs:element name="AlignPI">
  <xs:annotation>
    <xs:documentation>A Single Alignment PI
Definition</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Station"/>
      <xs:element ref="PI"/>
      <xs:element ref="InSpiral"
minOccurs="0"/>
      <xs:element ref="Curve1"
minOccurs="0"/>
      <xs:element ref="ConnSpiral"
minOccurs="0"/>
      <xs:element ref="Curve2"
minOccurs="0"/>
      <xs:element ref="OutSpiral"
minOccurs="0"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

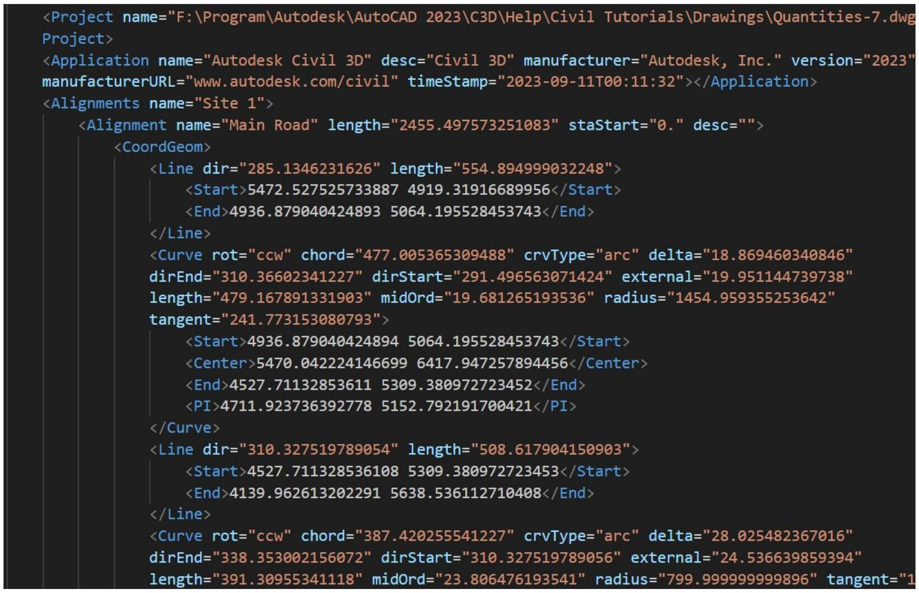
LandXML AlignPI format example

LandXML is a model that well expresses the civil engineering model shape. It is structured to well describe infrastructure engineering information such as roads, railways, and complexes, and includes many related concepts and attributes.

LandXML is basically created from basic design and detailed design modelers used in civil engineering. For this reason, various overseas civil engineering project management systems and software have essential functions to interpret and utilize this model.



Civil object model modeled in Autodesk Civil 3D and LandXML visualized after CGE analysis



```
<Project name="F:\Program\Autodesk\AutoCAD 2023\C3D\Help\Civil Tutorials\Drawings\Quantities-7.dwg
Project>
<Application name="Autodesk Civil 3D" desc="Civil 3D" manufacturer="Autodesk, Inc." version="2023"
manufacturerURL="www.autodesk.com/civil" timeStamp="2023-09-11T00:11:32"></Application>
<Alignments name="Site 1">
    <Alignment name="Main Road" length="2455.497573251083" staStart="0." desc="">
        <CoordGeom>
            <Line dir="285.1346231626" length="554.894999032248">
                <Start>5472.527525733887 4919.31916689956</Start>
                <End>4936.879040424893 5064.195528453743</End>
            </Line>
            <Curve rot="ccw" chord="477.005365309488" crvType="arc" delta="18.869460340846"
            dirEnd="310.36602341227" dirStart="291.496563071424" external="19.951144739738"
            length="479.167891331903" midOrd="19.681265193536" radius="1454.959355253642"
            tangent="241.773153080793">
                <Start>4936.879040424894 5064.195528453743</Start>
                <Center>5470.042224146699 6417.947257894456</Center>
                <End>4527.71132853611 5309.380972723452</End>
                <PI>4711.923736392778 5152.792191700421</PI>
            </Curve>
            <Line dir="310.327519789054" length="508.617904150903">
                <Start>4527.711328536108 5309.380972723453</Start>
                <End>4139.962613202291 5638.536112710408</End>
            </Line>
            <Curve rot="ccw" chord="387.420255541227" crvType="arc" delta="28.025482367016"
            dirEnd="338.353002156072" dirStart="310.327519789056" external="24.536639859394"
            length="391.30955341118" midOrd="23.806476193541" radius="799.999999999896" tangent="1
```

Part of the LandXML actual format

2. LandXML CGE Advantages

LandXML CGE is a calculation engine developed to obtain the necessary information by analyzing and geometrically numerically analyzing LandXML's Civil BIM parameter information model.

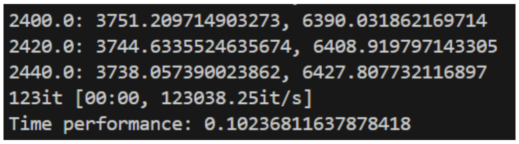
Since the core algorithm for analyzing the geometric structure of linear designs such as roads has been implemented, the development of diverse and flexible services is possible.

The advantages are as follows:

1. Since civil engineering design geometric parameters are used in the calculation as is, it is more accurate than the method of using an approximated DXF polyline. Since DXF approximates curved lines with line segments, it has problems such as the overall length not matching, so it cannot be used for engineering project management purposes.

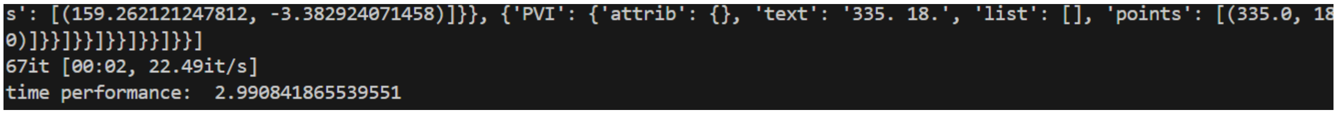
2. Fast civil engineering information model parser and CGE-based geometric analysis

High-speed calculation is possible by directly developing and embedding a civil engineering object numerical analysis algorithm in CGE. In the case of a 2,440 meter linear system, geometric structure calculation and 123 Station DB creation and storage (sqlite) takes approximately 0.102 seconds, 16783 cross-sectional object information calculation and DB creation and storage (MongoDB) takes 2.991 seconds, and 980 block object information calculation and DB creation takes approximately 0.102 seconds. Saving (MongoDB) takes 10.492 seconds (based on 12th Gen Intel(R) Core(TM) i9-12900H 2.90 GHz, 16G RAM)



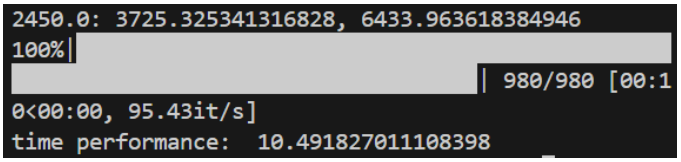
```
2400.0: 3751.209714903273, 6390.031862169714
2420.0: 3744.6335524635674, 6408.919797143305
2440.0: 3738.057390023862, 6427.807732116897
123it [00:00, 123038.25it/s]
Time performance: 0.10236811637878418
```

alignment statation position calculation performance



```
s': [(159.262121247812, -3.382924071458)]}}, {'PVI': {'attrib': {}, 'text': '335. 18.', 'list': [], 'points': [(335.0, 18
0)]}}]}}]}}]}}]}}]
67it [00:02, 22.49it/s]
time performance:  2.990841865539551
```

alignment chain(cross section) object data calculation performance



```
2450.0: 3725.325341316828, 6433.963618384946
100%|██████████████████████████████████████████
████████████████████████████████████| 980/980 [00:1
0<00:00, 95.43it/s]
time performance:  10.491827011108398
```

block object data calculation performance

3. Supports LandXML data parsing and interpretation calculations

4. Supports not only linear but also block creation and cross-sectional object analysis

5. Preservation of coordinate system

6. Easy-to-use API support

7. Numerical calculation API support, such as normal-linear distance calculation

8. Support database conversion

9. Supports various examples

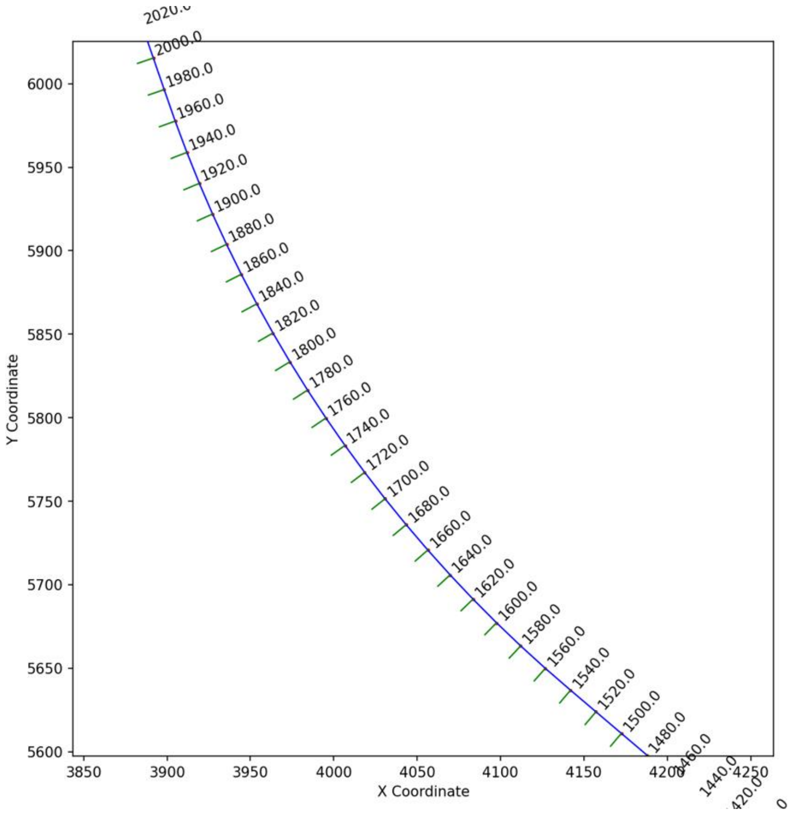
10. Support for web-based app development examples, etc.

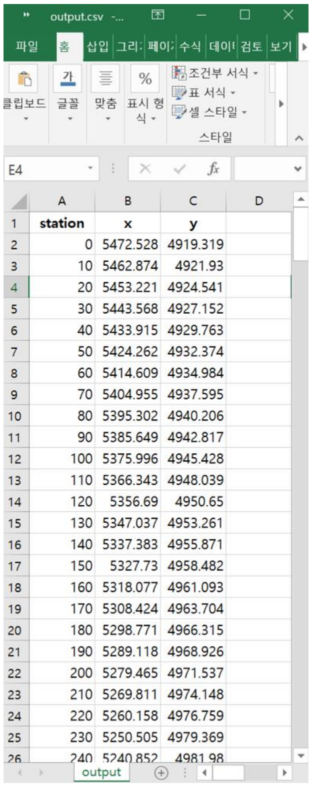
3. Functions

The following shows the basic functions that can be implemented using CGE.

3.1 Station analysis

The civil linear information model consists of parameters to define the geometry. For earthwork, crossing, and construction management, construction is managed on a station (distance from the point of construction) basis. CGE can interpret this.

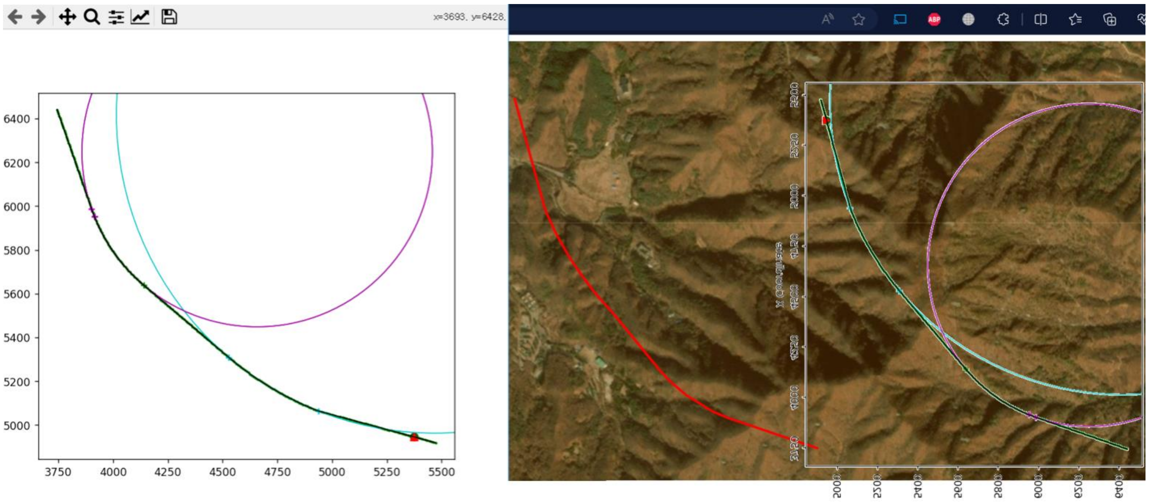




	A	B	C	D
1	station	x	y	
2	0	5472.528	4919.319	
3	10	5462.874	4921.93	
4	20	5453.221	4924.541	
5	30	5443.568	4927.152	
6	40	5433.915	4929.763	
7	50	5424.262	4932.374	
8	60	5414.609	4934.984	
9	70	5404.955	4937.595	
10	80	5395.302	4940.206	
11	90	5385.649	4942.817	
12	100	5375.996	4945.428	
13	110	5366.343	4948.039	
14	120	5356.69	4950.65	
15	130	5347.037	4953.261	
16	140	5337.383	4955.871	
17	150	5327.73	4958.482	
18	160	5318.077	4961.093	
19	170	5308.424	4963.704	
20	180	5298.771	4966.315	
21	190	5289.118	4968.926	
22	200	5279.465	4971.537	
23	210	5269.811	4974.148	
24	220	5260.158	4976.759	
25	230	5250.505	4979.369	
26	240	5240.852	4981.98	

3.2 Preservation of real coordinate system

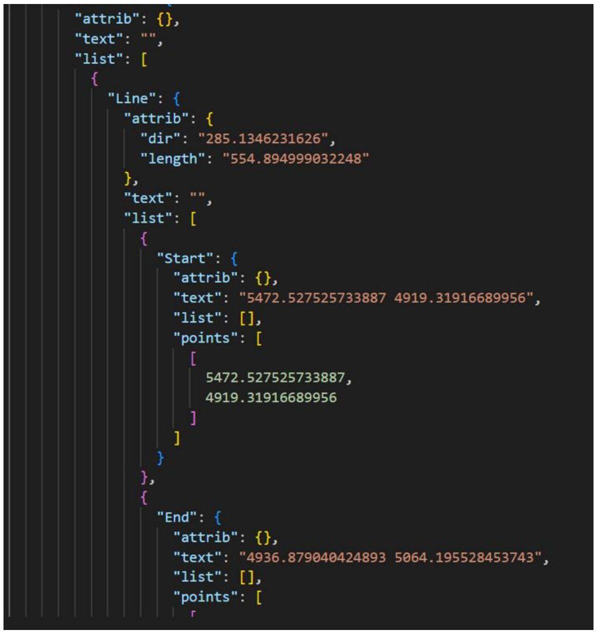
CGE preserves the real coordinate system (e.g. TM coordinate system, etc.). Therefore, there is no problem when overlaying other civil engineering objects based on absolute coordinates or converting other coordinate systems (e.g. GRS80, longitude and latitude, etc.).



Coordinate System Preservation

3.3 JSON Export

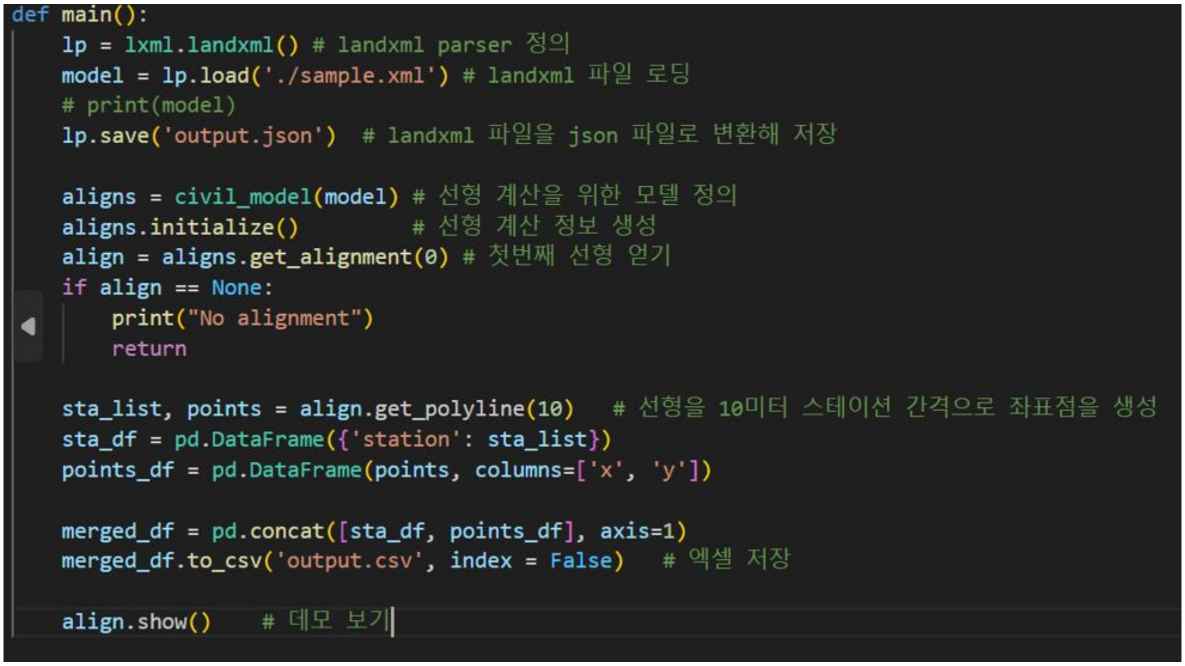
CGE parses LandXML and supports saving JSON files. Supported types include horizontal alignment, vertical alignment, cross section objects, object properties, and geometric design parameters.



```
"attrib": {},
"text": "",
"list": [
  {
    "Line": {
      "attrib": {
        "dir": "285.1346231626",
        "length": "554.894999032248"
      },
      "text": "",
      "list": [
        {
          "Start": {
            "attrib": {},
            "text": "5472.527525733887 4919.31916689956",
            "list": [],
            "points": [
              [
                5472.527525733887,
                4919.31916689956
              ]
            ]
          }
        },
        {
          "End": {
            "attrib": {},
            "text": "4936.879040424893 5064.195528453743",
            "list": [],
            "points": [
```

3.4 Python library API support

CGE supports an easy-to-use Python-based API for convenient service development.

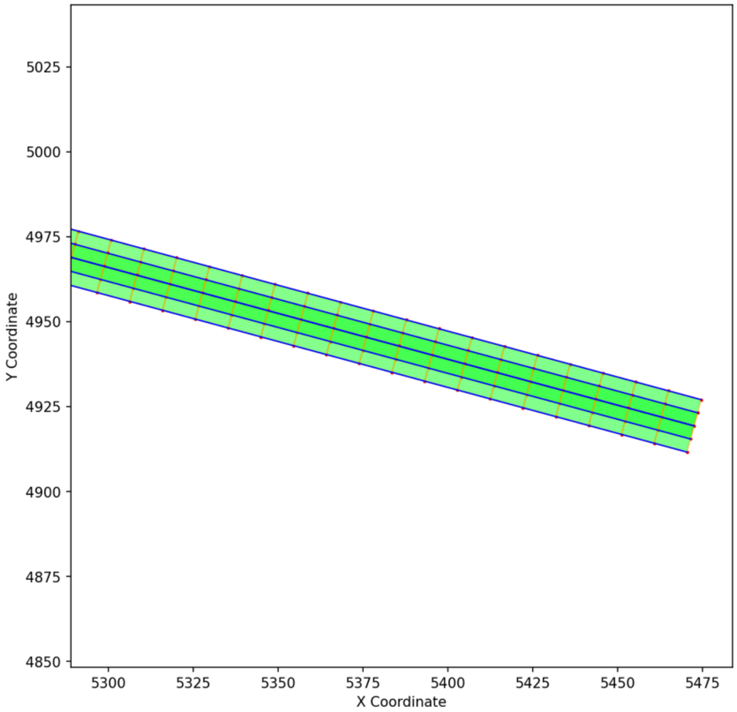


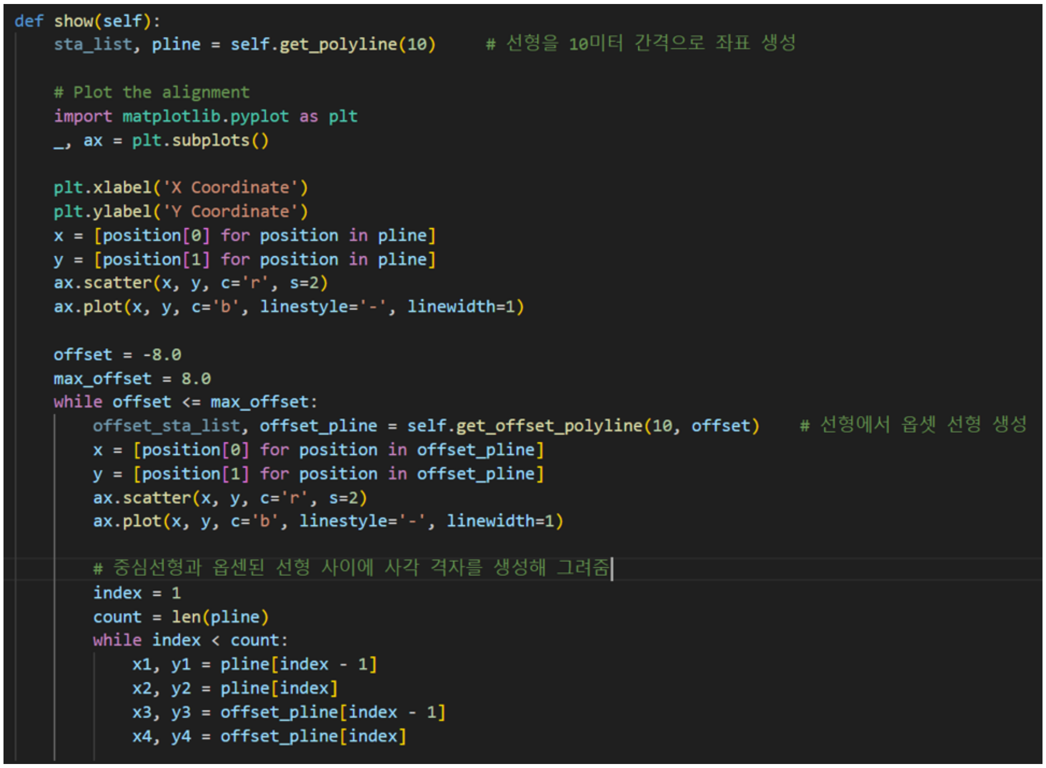
```python
def main():
    lp = lxml.landxml() # landxml parser 정의
    model = lp.load('./sample.xml') # landxml 파일 로딩
    # print(model)
    lp.save('output.json')  # landxml 파일을 json 파일로 변환해 저장

    aligns = civil_model(model) # 선형 계산을 위한 모델 정의
    aligns.initialize()         # 선형 계산 정보 생성
    align = aligns.get_alignment(0) # 첫번째 선형 얻기
    if align == None:
        print("No alignment")
        return

    sta_list, points = align.get_polyline(10)   # 선형을 10미터 스테이션 간격으로 좌표점을 생성
    sta_df = pd.DataFrame({'station': sta_list})
    points_df = pd.DataFrame(points, columns=['x', 'y'])

    merged_df = pd.concat([sta_df, points_df], axis=1)
    merged_df.to_csv('output.csv', index = False)   # 엑셀 저장

    align.show()    # 데모 보기
```

3.5 Information model visualization support

Provides the function to visualize geometric information interpreted in CGE.

Depending on the option, lane width, offset, perpendicular point, and curve parameter expressions are supported.

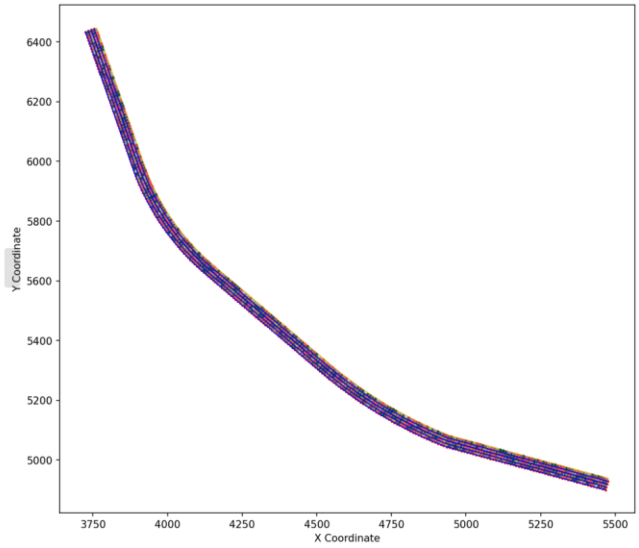


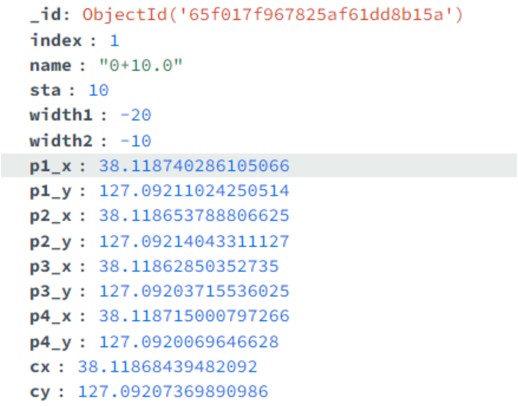


```
def show(self):
    sta_list, pline = self.get_polyline(10)     # 선형을 10미터 간격으로 좌표 생성

    # Plot the alignment
    import matplotlib.pyplot as plt
    _, ax = plt.subplots()

    plt.xlabel('X Coordinate')
    plt.ylabel('Y Coordinate')
    x = [position[0] for position in pline]
    y = [position[1] for position in pline]
    ax.scatter(x, y, c='r', s=2)
    ax.plot(x, y, c='b', linestyle='-', linewidth=1)

    offset = -8.0
    max_offset = 8.0
    while offset <= max_offset:
        offset_sta_list, offset_pline = self.get_offset_polyline(10, offset)    # 선형에서 옵셋 선형 생성
        x = [position[0] for position in offset_pline]
        y = [position[1] for position in offset_pline]
        ax.scatter(x, y, c='r', s=2)
        ax.plot(x, y, c='b', linestyle='-', linewidth=1)

        # 중심선형과 옵셋된 선형 사이에 사각 격자를 생성해 그려줌
        index = 1
        count = len(pline)
        while index < count:
            x1, y1 = pline[index - 1]
            x2, y2 = pline[index]
            x3, y3 = offset_pline[index - 1]
            x4, y4 = offset_pline[index]
```

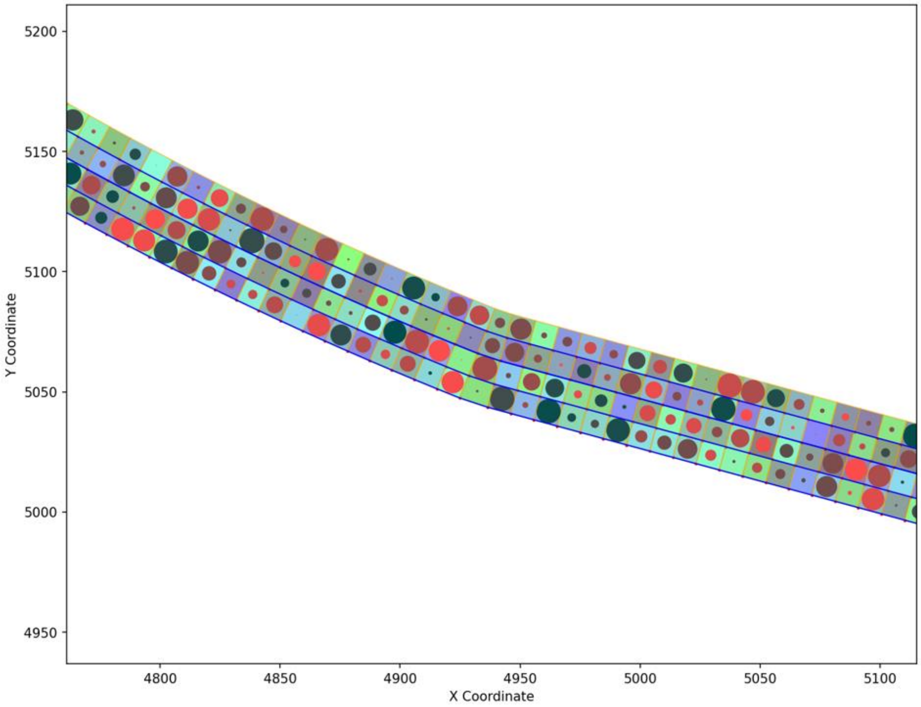
3.6 Block (grid) unit support to link external dataset

It supports self-implemented linear block geometric calculations without using a separate external library. Each block supports API to manage separate sensor data and obtain Bounding Box, Station, Width, etc.



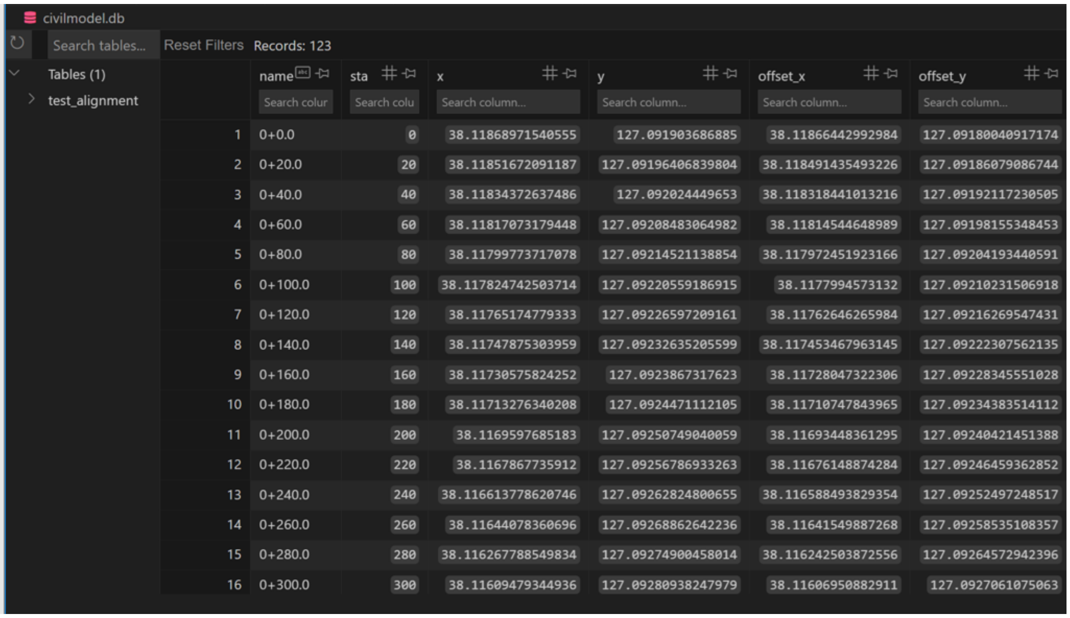


```
_id: ObjectId('65f017f967825af61dd8b15a')
index : 1
name : "0+10.0"
sta : 10
width1 : -20
width2 : -10
p1_x : 38.118740286105066
p1_y : 127.09211024250514
p2_x : 38.118653788806625
p2_y : 127.09214043311127
p3_x : 38.11862850352735
p3_y : 127.09203715536025
p4_x : 38.118715000797266
p4_y : 127.0920069646628
cx : 38.11868439482092
cy : 127.09207369890986
```



3.7 Excel Export

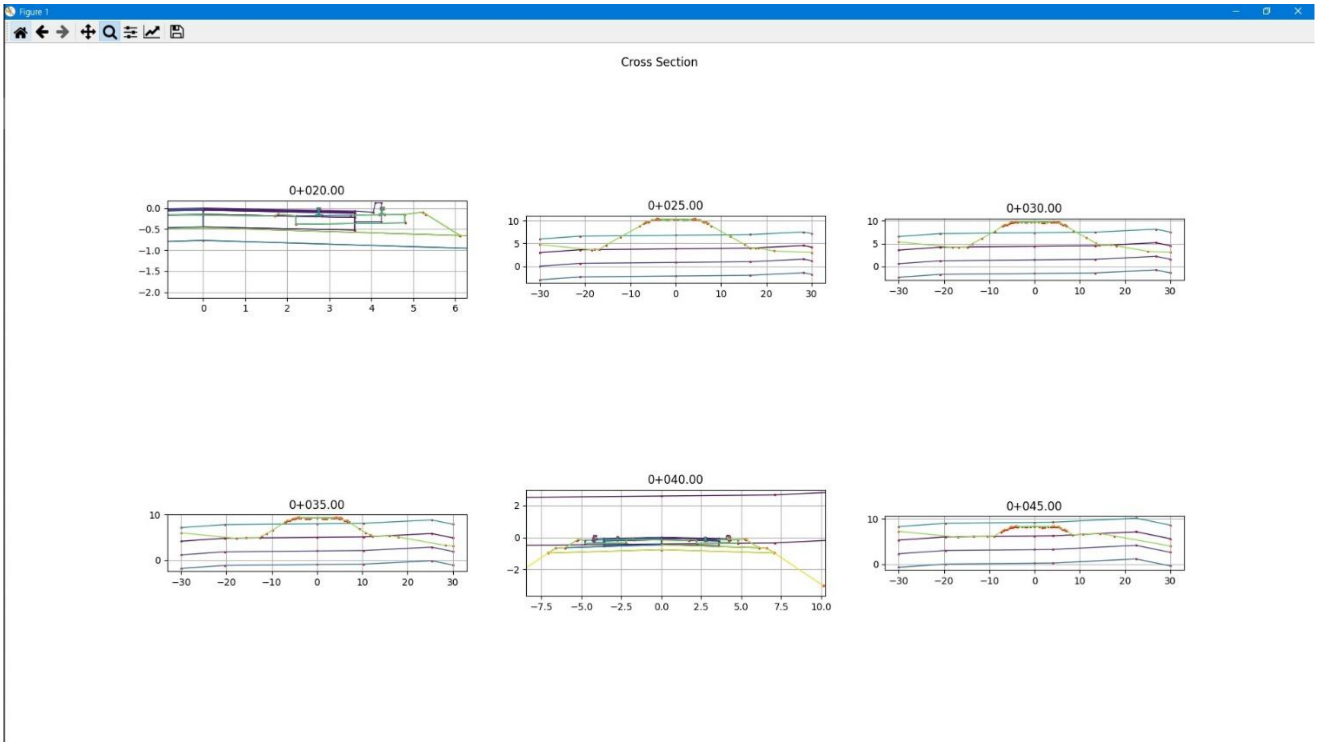
Supports Excel output for analyzed stations.



	name	sta	x	y	offset_x	offset_y
1	0+0.0	0	38.11868971540555	127.091903686885	38.11866442992984	127.09180040917174
2	0+20.0	20	38.11851672091187	127.09196406839804	38.118491435493226	127.09186079086744
3	0+40.0	40	38.11834372637486	127.092024449653	38.118318441013216	127.09192117230505
4	0+60.0	60	38.11817073179448	127.09208483064982	38.11814544648989	127.09198155348453
5	0+80.0	80	38.11799773717078	127.09214521138854	38.117972451923166	127.09204193440591
6	0+100.0	100	38.117824742503714	127.09220559186915	38.1177994573132	127.09210231506918
7	0+120.0	120	38.11765174779333	127.09226597209161	38.11762646265984	127.09216269547431
8	0+140.0	140	38.11747875303959	127.09232635205599	38.117453467963145	127.09222307562135
9	0+160.0	160	38.11730575824252	127.0923867317623	38.11728047322306	127.09228345551028
10	0+180.0	180	38.11713276340208	127.0924471112105	38.11710747843965	127.09234383514112
11	0+200.0	200	38.1169597685183	127.09250749040059	38.11693448361295	127.09240421451388
12	0+220.0	220	38.1167867735912	127.09256786933263	38.11676148874284	127.09246459362852
13	0+240.0	240	38.116613778620746	127.09262824800655	38.116588493829354	127.09252497248517
14	0+260.0	260	38.11644078360696	127.09268862642236	38.11641549887268	127.09258535108357
15	0+280.0	280	38.116267788549834	127.09274900458014	38.116242503872556	127.09264572942396
16	0+300.0	300	38.11609479344936	127.09280938247979	38.11606950882911	127.0927061075063

3.8 Cross-sectional object analysis support

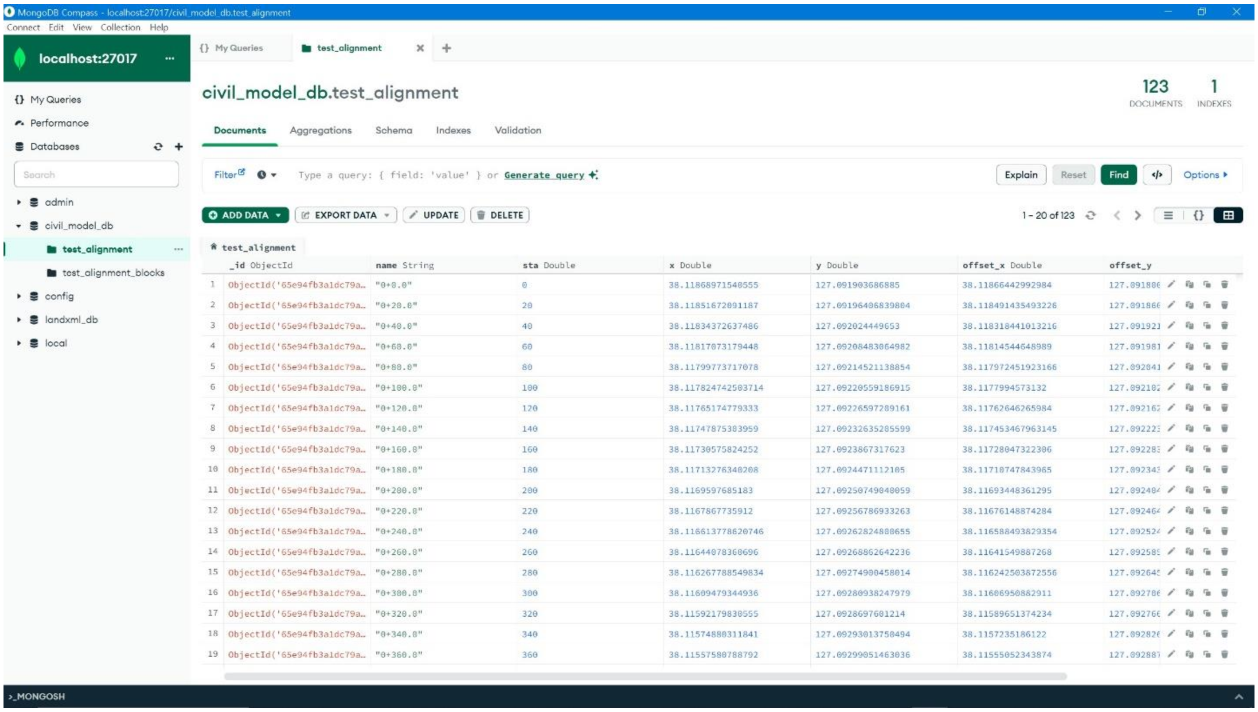
A crossing is a set of corridors made up of several parts. This can be interpreted and visualized.



3.9 Support for database output such as MongoDB

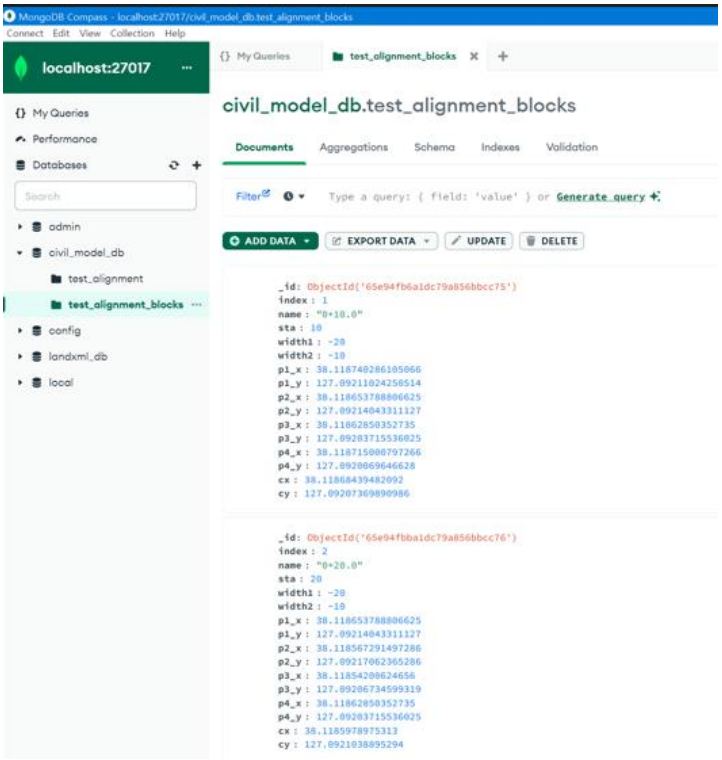
Supports database output such as MongoDB and sqlite.

Through this, web application development is possible.



test_alignment

	_id ObjectId	name String	sta Double	x Double	y Double	offset_x Double	offset_y
1	ObjectId('65e94fb3a1dc79a…	"0+0.0"	0	38.11868971540555	127.091903686885	38.11866442992984	127.091806
2	ObjectId('65e94fb3a1dc79a…	"0+20.0"	20	38.11851672091187	127.09196406839804	38.118491435493226	127.091866
3	ObjectId('65e94fb3a1dc79a…	"0+40.0"	40	38.11834372637486	127.092024449653	38.118318441013216	127.091921
4	ObjectId('65e94fb3a1dc79a…	"0+60.0"	60	38.11817073179448	127.09208483064982	38.11814544648989	127.091981
5	ObjectId('65e94fb3a1dc79a…	"0+80.0"	80	38.11799773717078	127.09214521138854	38.117972451923166	127.092041
6	ObjectId('65e94fb3a1dc79a…	"0+100.0"	100	38.117824742503714	127.09220559186915	38.1177994573132	127.092102
7	ObjectId('65e94fb3a1dc79a…	"0+120.0"	120	38.11765174779333	127.09226597209161	38.11762646265984	127.092162
8	ObjectId('65e94fb3a1dc79a…	"0+140.0"	140	38.11747875303959	127.09232635205599	38.117453467963145	127.092223
9	ObjectId('65e94fb3a1dc79a…	"0+160.0"	160	38.11730575824252	127.0923867317623	38.11728047322306	127.092283
10	ObjectId('65e94fb3a1dc79a…	"0+180.0"	180	38.11713276340208	127.0924471112105	38.11710747843965	127.092343
11	ObjectId('65e94fb3a1dc79a…	"0+200.0"	200	38.1169597685183	127.09250749040059	38.11693448361295	127.092404
12	ObjectId('65e94fb3a1dc79a…	"0+220.0"	220	38.1167867735912	127.09256786933263	38.11676148874284	127.092464
13	ObjectId('65e94fb3a1dc79a…	"0+240.0"	240	38.116613778620746	127.09262824800655	38.116588493829354	127.092524
14	ObjectId('65e94fb3a1dc79a…	"0+260.0"	260	38.11644078360696	127.09268862642236	38.11641549887268	127.092585
15	ObjectId('65e94fb3a1dc79a…	"0+280.0"	280	38.116267788549834	127.09274900458014	38.116242503872556	127.092645
16	ObjectId('65e94fb3a1dc79a…	"0+300.0"	300	38.11609479344936	127.09280938247979	38.11606950882911	127.092706
17	ObjectId('65e94fb3a1dc79a…	"0+320.0"	320	38.11592179830555	127.0928697601214	38.11589651374234	127.092766
18	ObjectId('65e94fb3a1dc79a…	"0+340.0"	340	38.11574880311841	127.09293013750494	38.1157235186122	127.092826
19	ObjectId('65e94fb3a1dc79a…	"0+360.0"	360	38.11557580788792	127.09299051463036	38.11555052343874	127.092887



```
_id: ObjectId('65e94fb6a1dc79a856bbcc75')
index : 1
name : "0+10.0"
sta : 10
width1 : -20
width2 : -10
p1_x : 38.118740286105066
p1_y : 127.09211024258514
p2_x : 38.118653788806625
p2_y : 127.09214043311127
p3_x : 38.11862850352735
p3_y : 127.09203715536025
p4_x : 38.118715080797266
p4_y : 127.0920069646628
cx : 38.11868439482092
cy : 127.09207369890986
```

```
_id: ObjectId('65e94fbba1dc79a856bbcc76')
index : 2
name : "0+20.0"
sta : 20
width1 : -20
width2 : -10
p1_x : 38.118653788806625
p1_y : 127.09214043311127
p2_x : 38.118567291497286
p2_y : 127.09217062365286
p3_x : 38.11854208624656
p3_y : 127.09206734599319
p4_x : 38.11862850352735
p4_y : 127.09203715536025
cx : 38.1185978975313
cy : 127.0921038895294
```

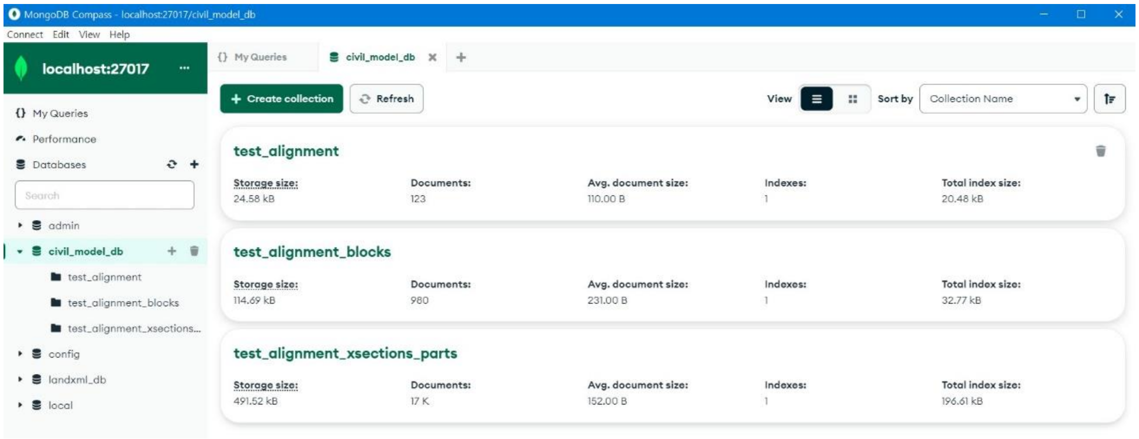
3.10 Basic Web App Example

A basic Web App example is provided so that you can use all functions. The examples are largely as follows.

1. How to create a database
2. How to link web map and linear CGE data
3. How to link web map and cross-sectional CGE data

This example provides CGE code for implementing the following sequence.

1. MongoDB Export from LandXML through CGE interpretation
2. Create a map on the web
3. Import CGE data from the web to MongoDB
4. Query CGE data from map in MongoDB
   A. CGE Alignment Data
   B. CGE Station data
   C. CGE Block data
5. Visualize queried data on map
6. Execute event function when clicking on CGE object
   A. CGE data query and display
   B. Crossing markings



test_alignment

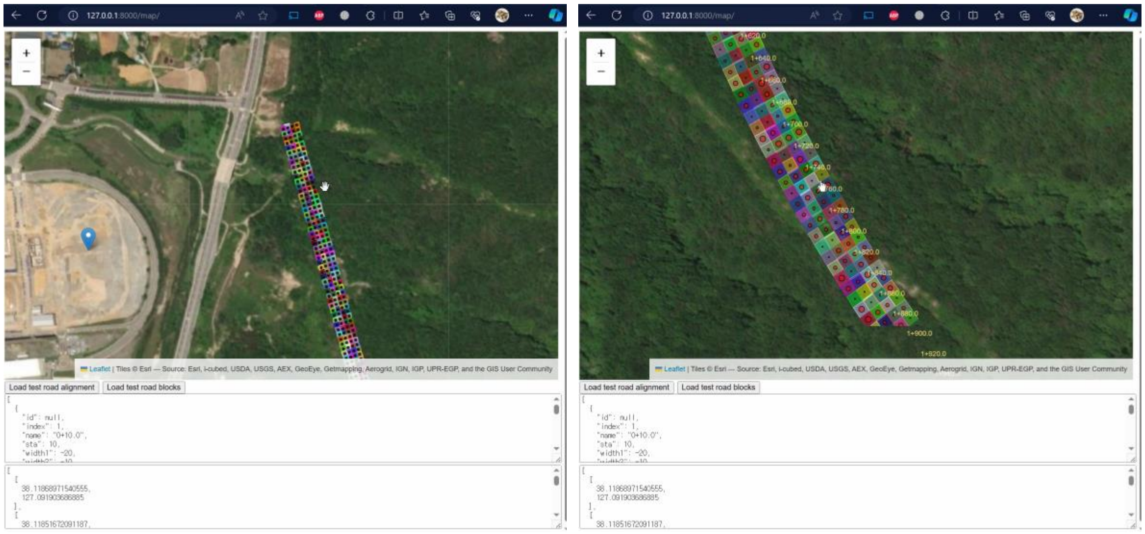
Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
24.58 kB	123	110.00 B	1	20.48 kB

test_alignment_blocks

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
114.69 kB	980	231.00 B	1	32.77 kB

test_alignment_xsections_parts

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
491.52 kB	17 K	152.00 B	1	196.61 kB


127.0.0.1:8000/map/
+
−
Leaflet | Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community
Load test road alignment
Load test road blocks
127.0.0.1:8000/map/
1+620.0
1+640.0
1+660.0
1+680.0
1+700.0
1+720.0
1+740.0
1+760.0
1+780.0
1+800.0
1+820.0
1+840.0
1+860.0
1+880.0
1+900.0
1+920.0
Leaflet | Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community
Load test road alignment
Load test road blocks

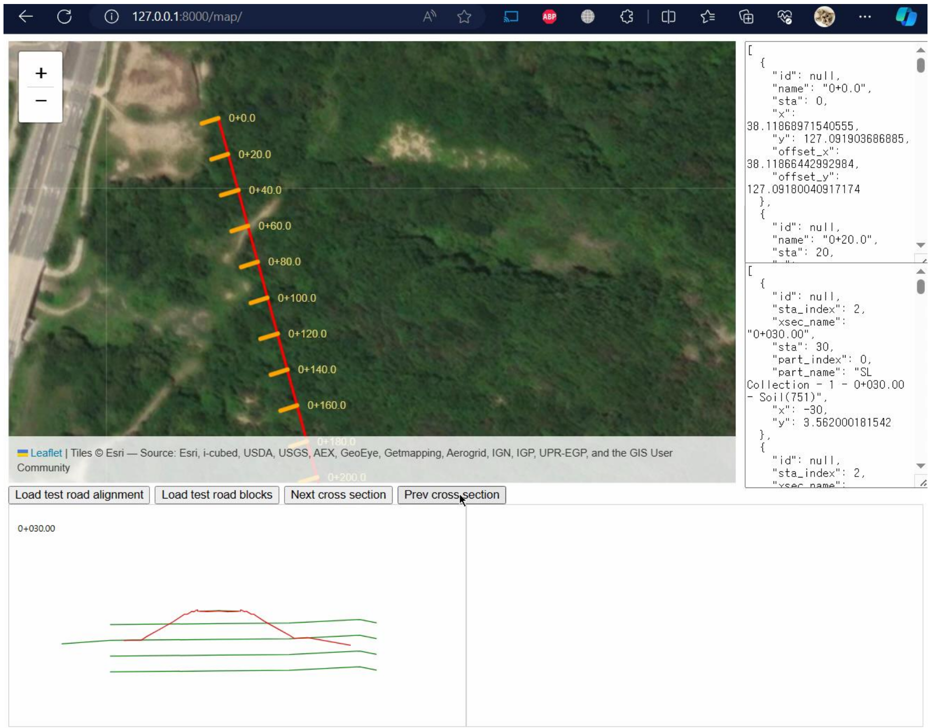
```
[
  {
    "id": null,
    "index": 1,
    "name": "0+10.0",
    "sta": 10,
    "width1": -20,
```

```
[
  [
    38.11868971540555,
    127.091903686885
  ],
  [
    38.11851672091187,
```

```
[
  {
    "id": null,
    "index": 1,
    "name": "0+10.0",
    "sta": 10,
    "width1": -20,
```

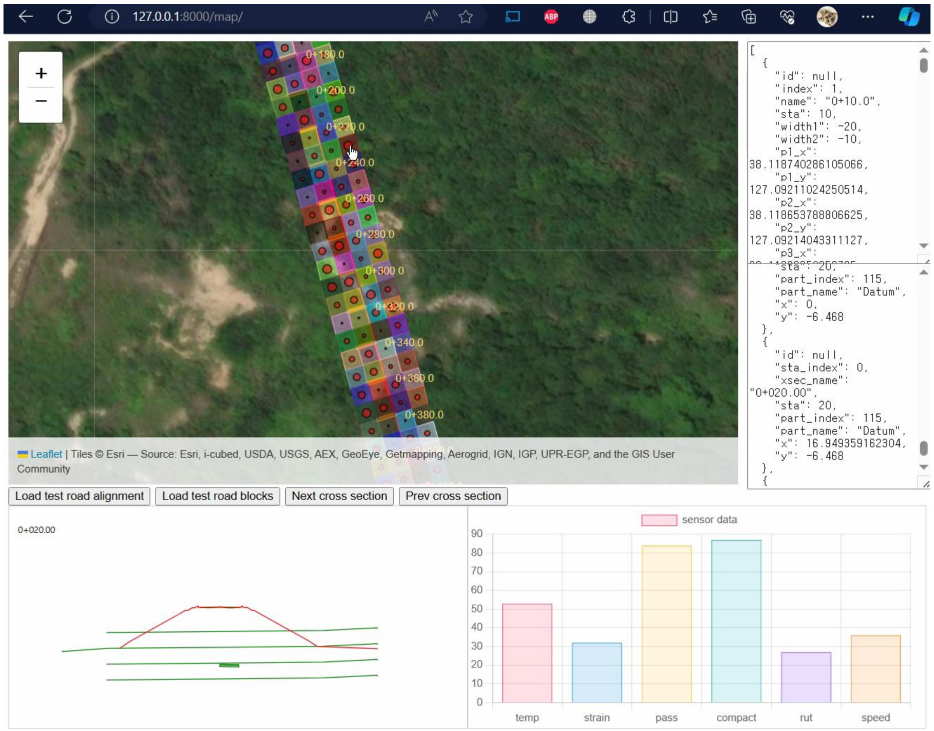
```
[
  [
    38.11868971540555,
    127.091903686885
  ],
  [
    38.11851672091187,
```

Alignment station, block map and data display



```
[
  {
    "id": null,
    "name": "0+0.0",
    "sta": 0,
    "x":
38.11868971540555,
    "y": 127.091903686885,
    "offset_x":
38.11866442992984,
    "offset_y":
127.09180040917174
  },
  {
    "id": null,
    "name": "0+20.0",
    "sta": 20,
```

```
[
  {
    "id": null,
    "sta_index": 2,
    "xsec_name":
"0+030.00",
    "sta": 30,
    "part_index": 0,
    "part_name": "SL
Collection - 1 - 0+030.00
- Soil(751)",
    "x": -30,
    "y": 3.562000181542
  },
  {
    "id": null,
    "sta_index": 2,
```



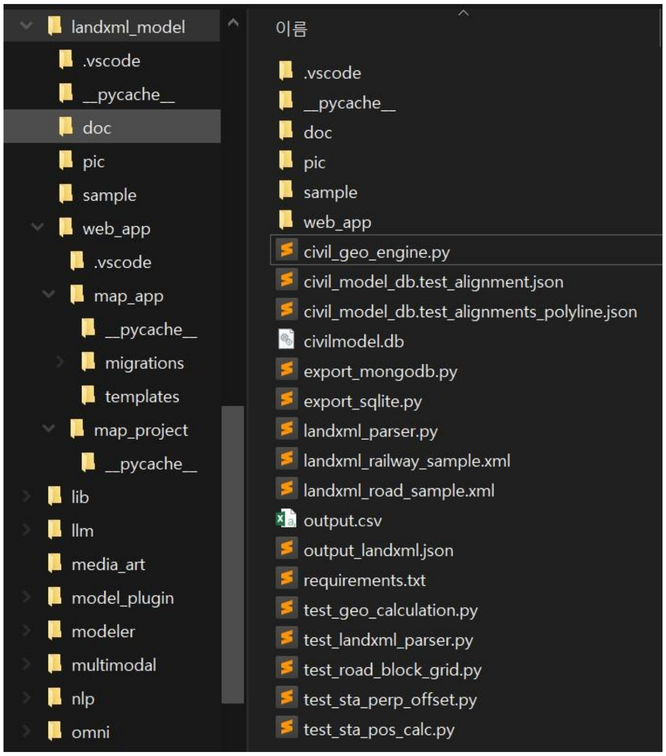
```
[
  {
    "id": null,
    "index": 1,
    "name": "0+10.0",
    "sta": 10,
    "width1": -20,
    "width2": -10,
    "p1_x":
38.118740286105066,
    "p1_y":
127.09211024250514,
    "p2_x":
38.118653788806625,
    "p2_y":
127.09214043311127,
    "p3_x":
```

```
    "sta": 20,
    "part_index": 115,
    "part_name": "Datum",
    "x": 0,
    "y": -6.468
  },
  {
    "id": null,
    "sta_index": 0,
    "xsec_name":
"0+020.00",
    "sta": 20,
    "part_index": 115,
    "part_name": "Datum",
    "x": 16.949359162304,
    "y": -6.468
  },
  {
```

Event handling on CGE traverse and object clicks

3.11 Various test examples

Provides various basic test examples for using CGE.



```
landxml_model
  .vscode
  __pycache__
  doc
  pic
  sample
  web_app
    .vscode
    map_app
      __pycache__
      migrations
      templates
    map_project
      __pycache__
lib
llm
media_art
model_plugin
modeler
multimodal
nlp
omni
```

```
이름
.vscode
__pycache__
doc
pic
sample
web_app
civil_geo_engine.py
civil_model_db.test_alignment.json
civil_model_db.test_alignments_polyline.json
civilmodel.db
export_mongodb.py
export_sqlite.py
landxml_parser.py
landxml_railway_sample.xml
landxml_road_sample.xml
output.csv
output_landxml.json
requirements.txt
test_geo_calculation.py
test_landxml_parser.py
test_road_block_grid.py
test_sta_perp_offset.py
test_sta_pos_calc.py
```

그림. 다양한 테스트 소스 예제