Computational Graphics: Lecture 13

Alberto Paoluzzi

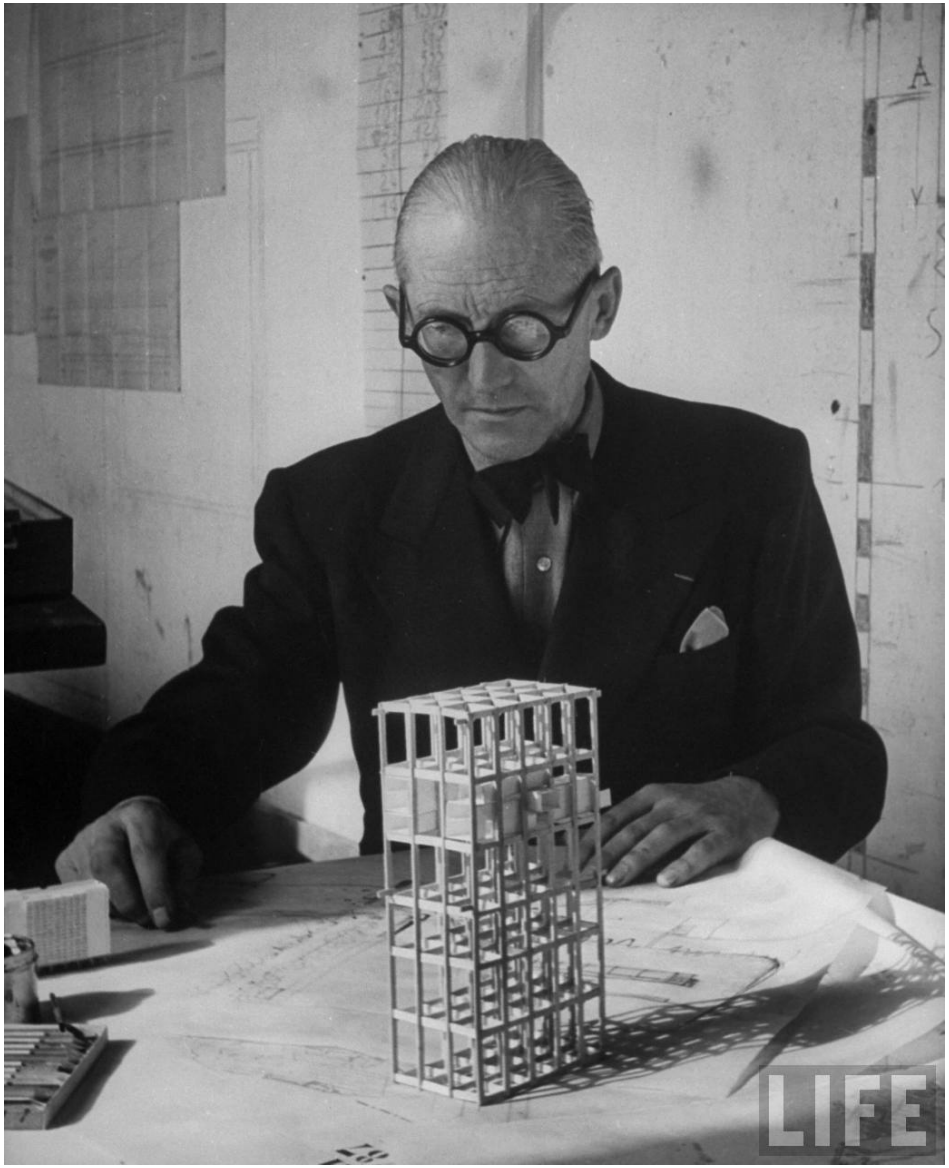
Mon, Apr 11, 2015

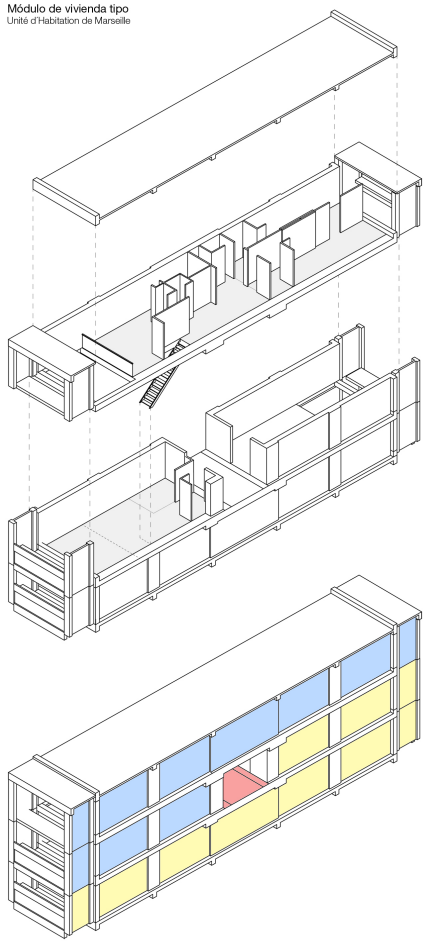
Outline: Modeling: "Unit'e d'habitation Marseille"

1. Building Backbone
2. 2D data input
3. 3D modeling

Building Backbone

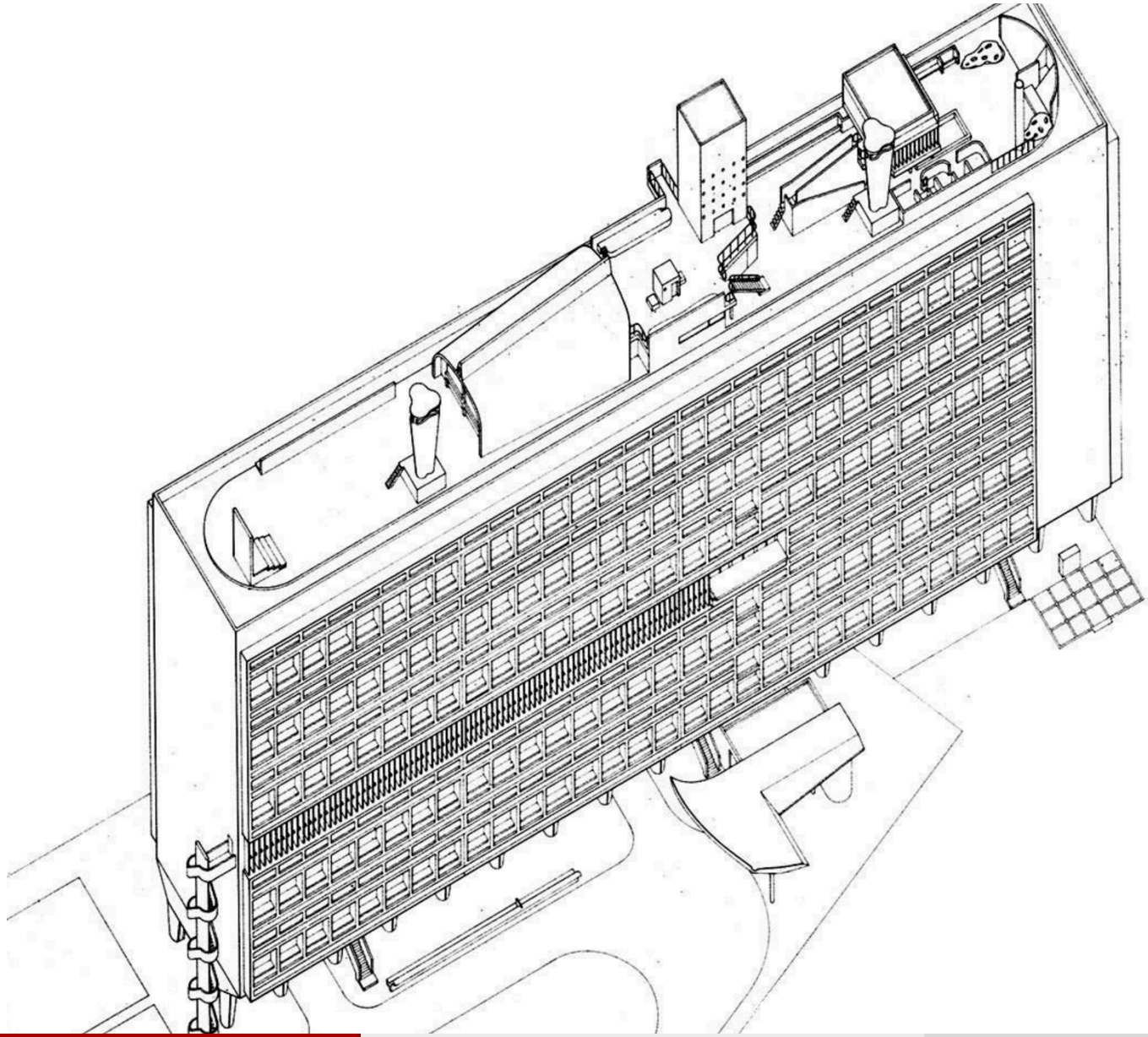
The main idea





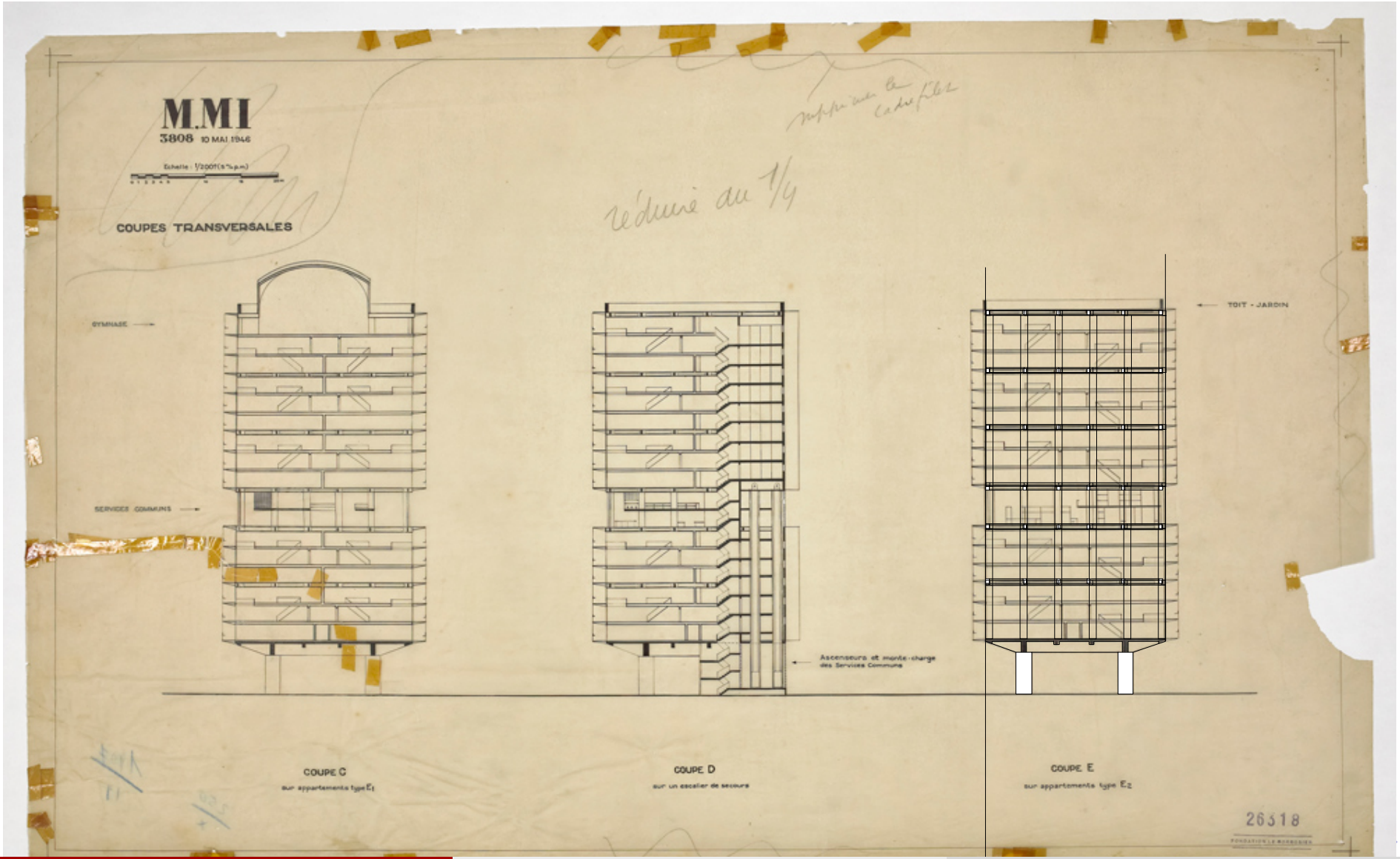
Two duplex apartments and the interior street

The main idea



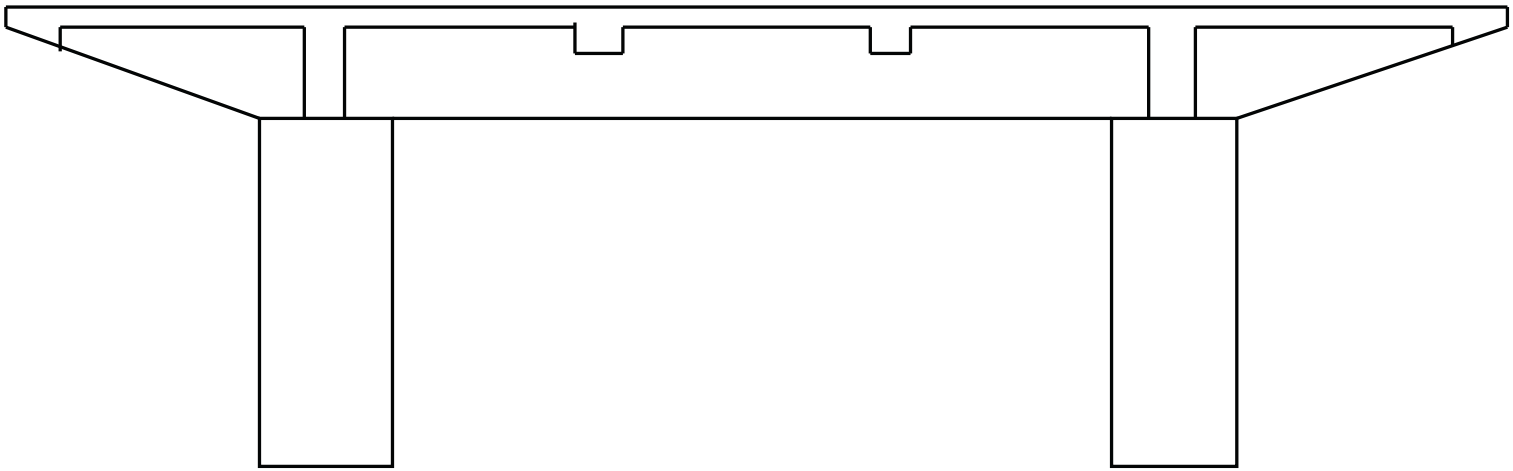
The main idea

Building sections — reference for model measurement



2D data input

Input of pilotis (base-structure)



```
""" Input pilotis (struttura-base) """
filename = "struttura-base.svg"
lines = svg2lines(filename)
VIEW(STRUCT(AA(POLYLINE)(lines)))
```

Cellular 2-complex

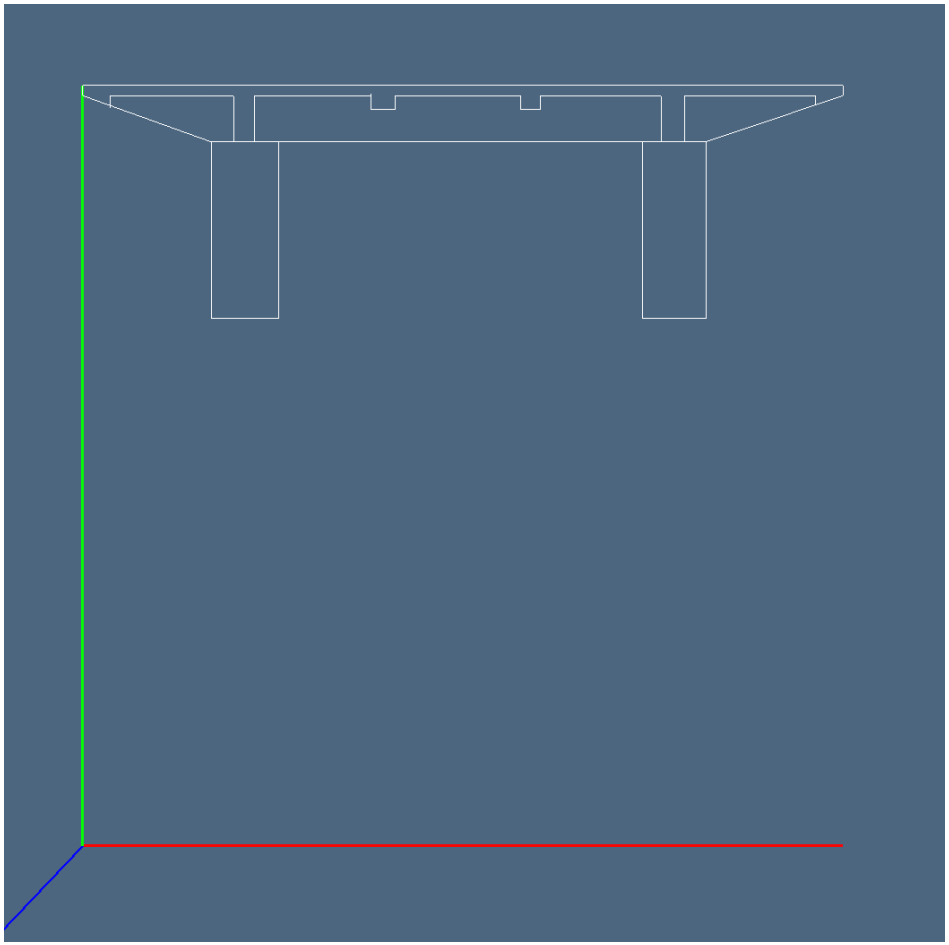


Figure 1:Line drawing in normalized space

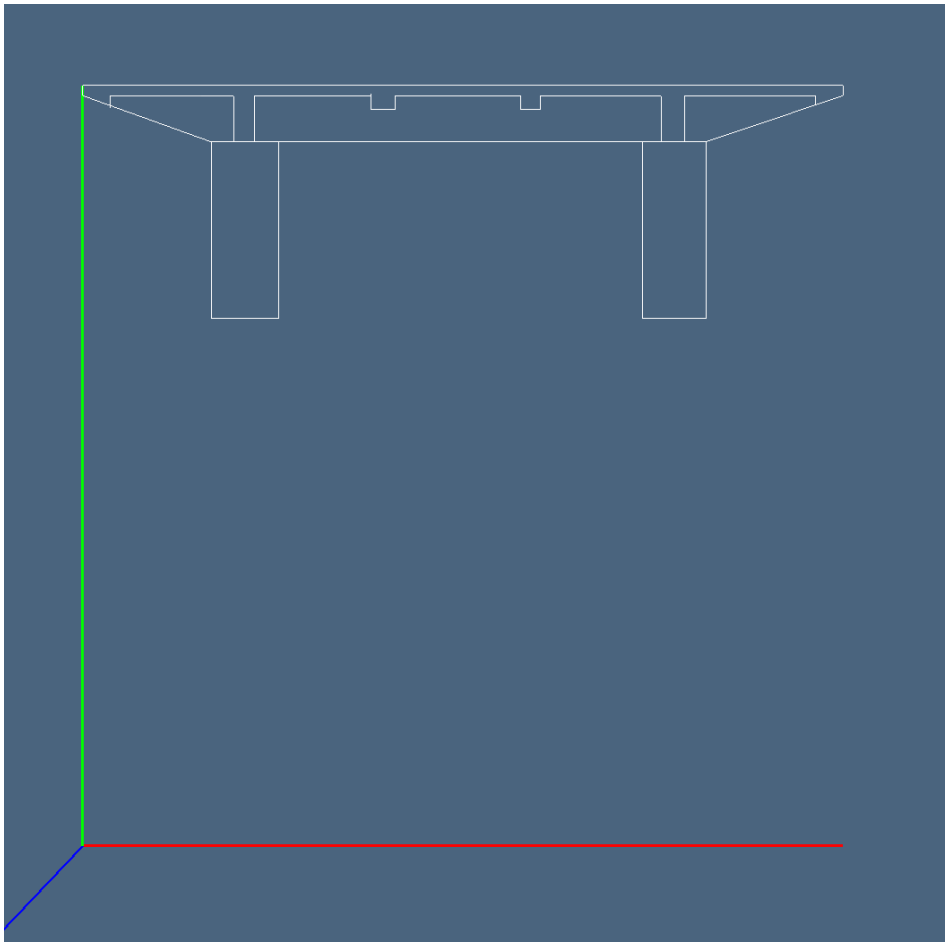


Figure 2:Cellular2-complex

Cellular 2-complex

```
""" Cellular complex of 2D pilotis """

V,FV,EV,polygons = larFromLines(lines)
VIEW(EXPLODE(1.2,1.2,1)(MKPOLS((V,EV)) + AA(MK)(V)))
VIEW(EXPLODE(1.2,1.2,1)(MKTRIANGLES((V,FV,EV))))
VIEW(STRUCT(MKTRIANGLES((V,FV,EV)) + AA(MK)(V)))
```

Cellular 2-complex

Figure 3:Exploded 2-cells

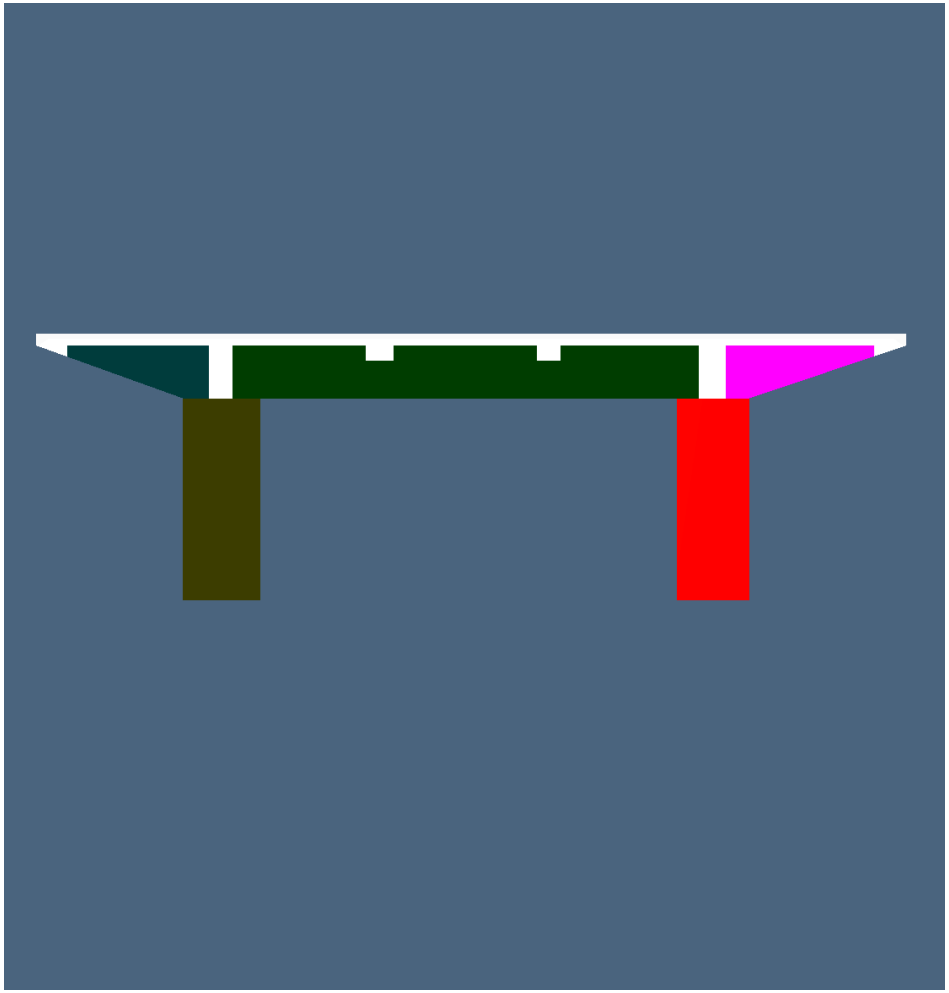


Figure 4:Colored 2-cells

Cellular 2-complex

```
colors = [CYAN,MAGENTA,WHITE,RED,YELLOW,GREEN,GRAY,ORANGE,
    BLACK,BLUE,PURPLE,BROWN]

VIEW(STRUCT([ COLOR(colors[k%12])(cell)    # HPC value
    for k,cell in enumerate( MKTRIANGLES((V,FV,EV))) ]))
```

Remark

`MKTRIANGLES((V,FV,EV))` requires a LAR model as triple

Cellular 2-complex

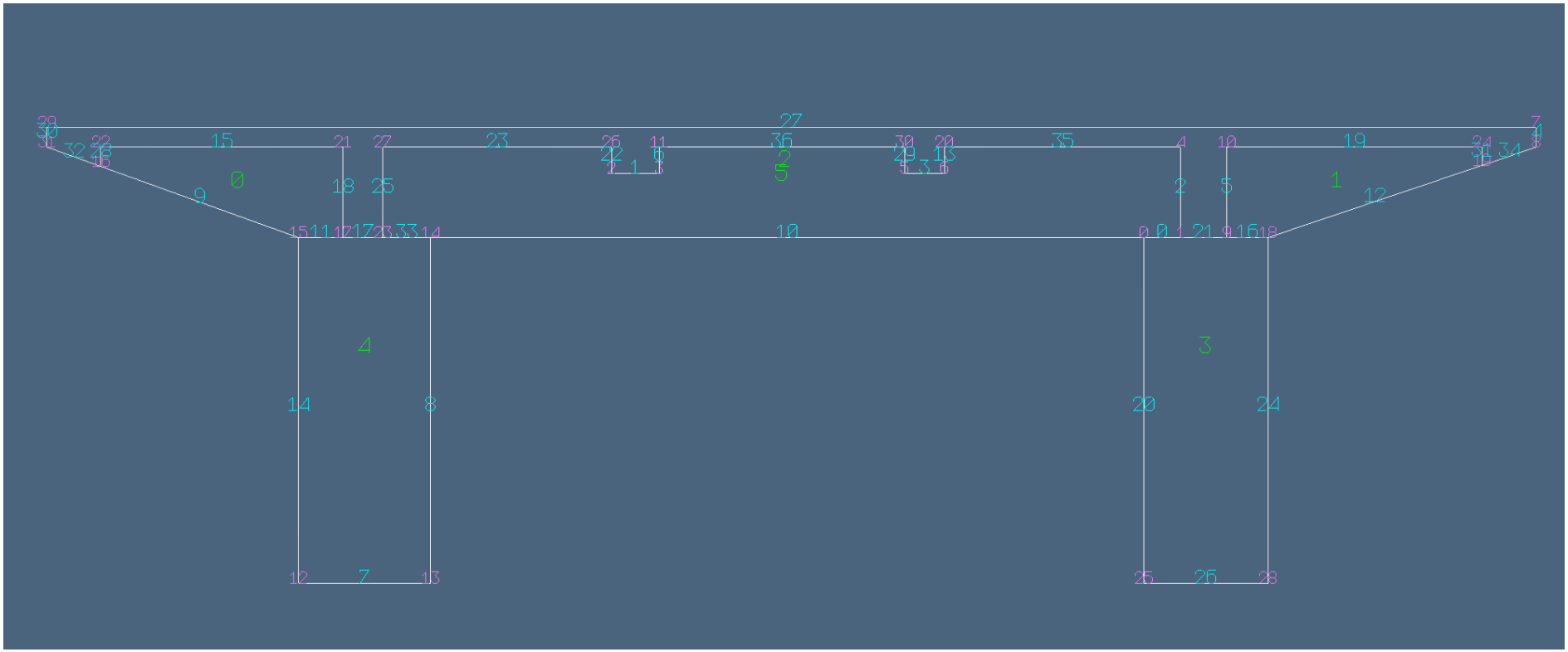


Figure 5:Color numbering of various skeletons

```
VV = AA(LIST)(range(len(V)))
submodel = STRUCT(MKPOLS((V,EV)))
VIEW(larModelNumbering(1,1,1)(V,[VV,EV,FV],submodel,0.1))
```

Scaling to true measures (in meters: m)

- n of spans between structure frames $= 16$
- n of structure frames $= 17$
- span width $= 2 \times 4.8\,m = 9.6\,m$
- balcony cantilever $= 1.2\,m$
- basement width $= 24\,m - 2 \times 1.2\,m = 21.6\,m$

Scaling to true measures

```
""" model edge of maximum length """
assert EV[27] == [29, 7]
assert V[29], V[7] == ([0.0, 1.0], [1.0, 1.0])

""" Scaling to true measures (in meters) """
basementScaling = 21.6
W = ((mat(V) - V[12]) * basementScaling).tolist()
```

`W` new (scaled) vertex list

Remark

notice the use of Scipy `matrix` class (look for!)

Definition of semantics of some 2-subcomplexes

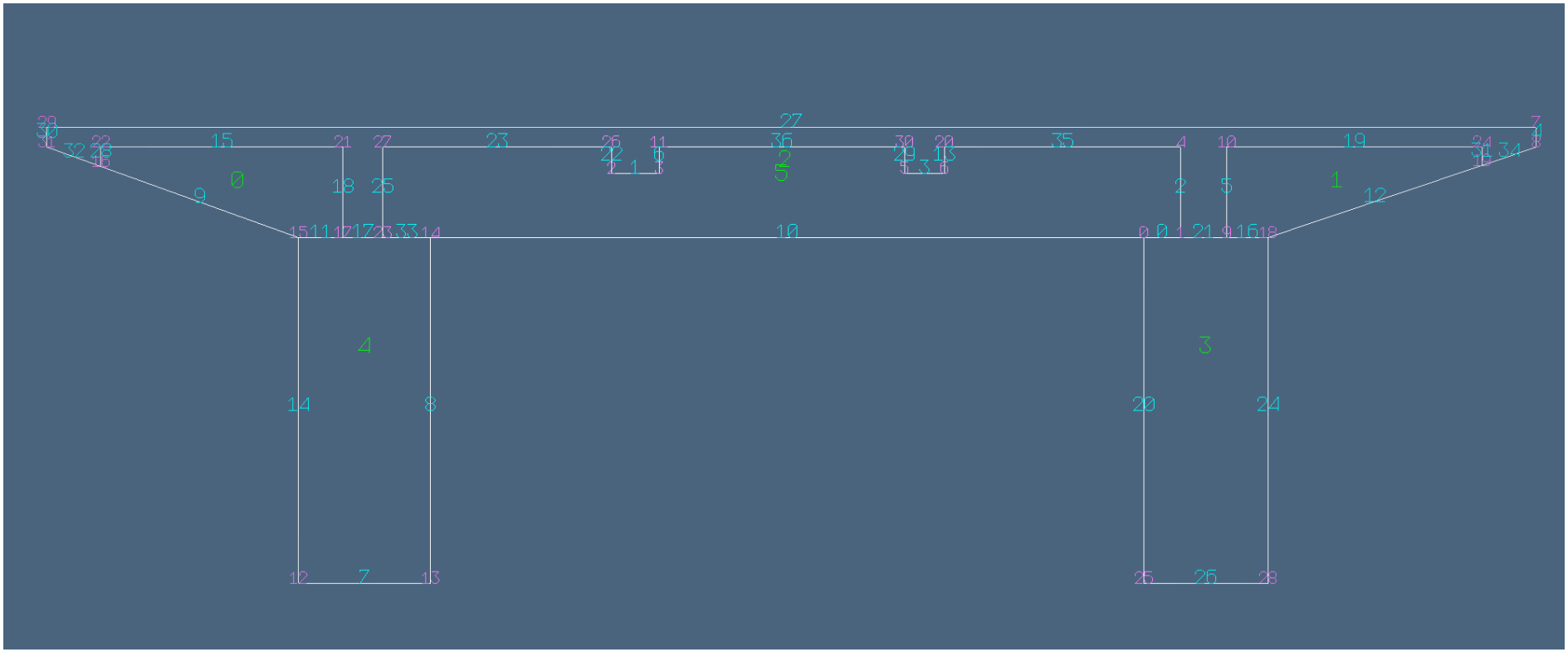


Figure 6:notice the indices of 2-cells (see below)

```
""" Definition of semantics of some 2-subcomplexes """
pillar = (W,[FV[k] for k in [0,1,3,4,5]])
slab = (W,[FV[k] for k in [2]])
VIEW(EXPLODE(1.2,1.2,1)(MKPOLS(pillar)+MKPOLS(slab)))
```

pillars and slabs

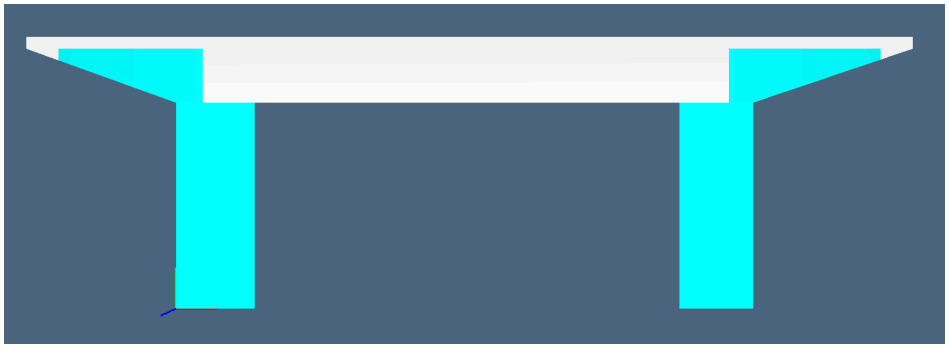


Figure 7:pillar (cyan) and slab (white). The central white color is a bug :-(

```
""" color rendering """
VIEW(STRUCT([ COLOR(CYAN)(STRUCT(MKPOLS(pillar))),
              STRUCT(MKPOLS(slab)) ]))
```

pillars and slabs

```
pillarPattern = 17*[1.2,-8.4]
VIEW(STRUCT(MKPOLS(larQuote1D(pillarPattern))))
pillars = larModelProduct([pillar,larQuote1D(pillarPattern)])
VIEW(STRUCT(MKPOLS(pillars)))

slabPattern = 16*[1.2,8.4]+[1.2]
slabs = larModelProduct([slab,larQuote1D(slabPattern)])
VIEW(STRUCT(MKPOLS(slabs)))

VIEW(STRUCT(MKPOLS(pillars)+MKPOLS(slabs)))
```

pillars and slabs

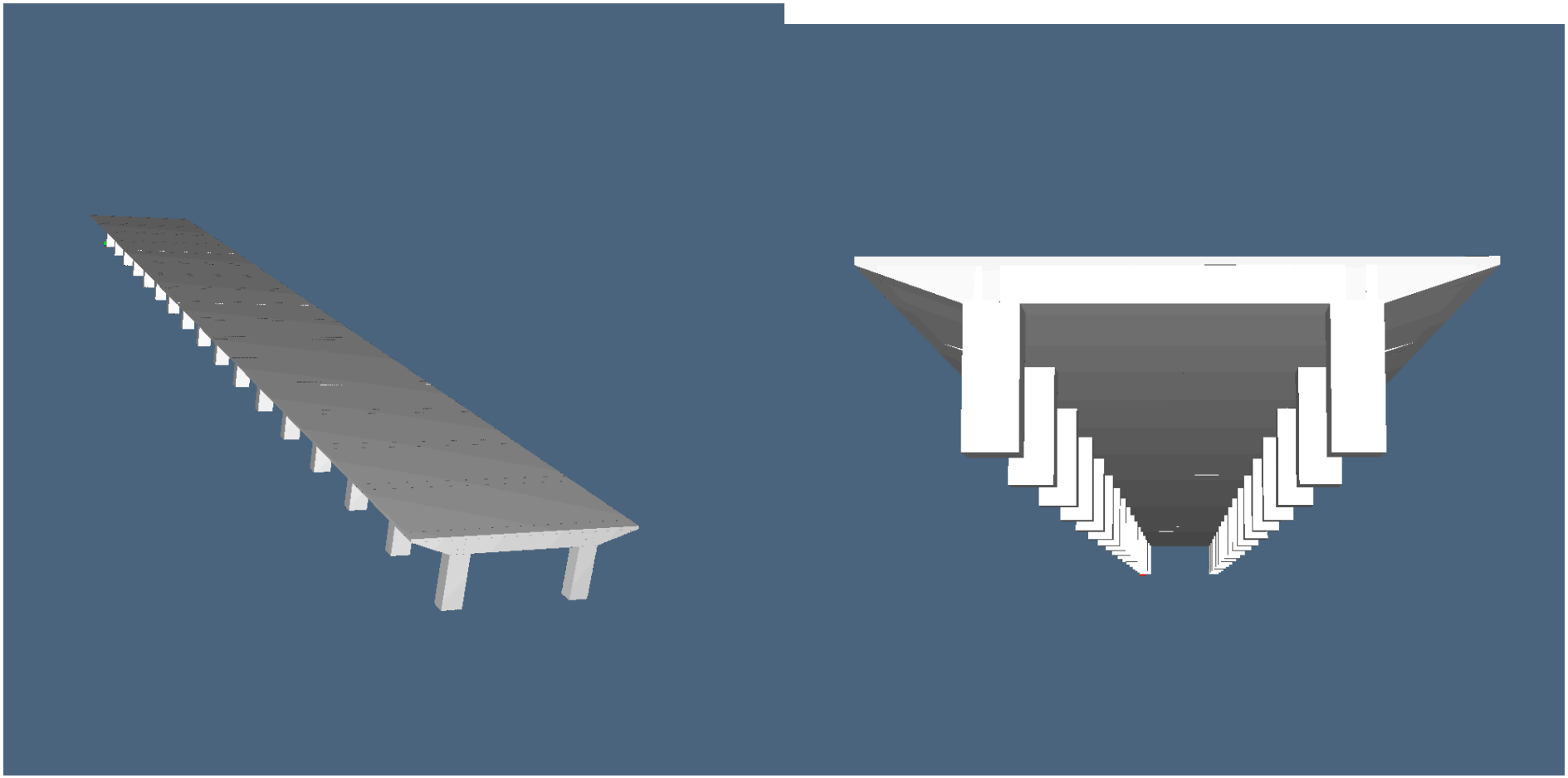


Figure 8:Repetition of basement in world coordinates

Unit frame in normalized space

```
""" Unit frame in normalized space """
filename = 'travePilastri.svg'
linesTravePilastri = svg2lines(filename)
VIEW(STRUCT(AA(POLYLINE)(linesTravePilastri)))
P,FP,EP,polygons = larFromLines(linesTravePilastri)
PP = AA(LIST)(range(len(P)))
submodel = STRUCT(MKPOLS((P,EP)))
VIEW(larModelNumbering(1,1,1)(P,[PP,EP,FP],submodel,0.075))

pilastri = (P,[FP[k] for k in [0,12,13,14,15,16]])
traveLongitudinale = (P,[FP[k] for k in [6,7,8,9,10,11]])
traviTrasversali = (P,[FP[k] for k in [1,2,3,4,5]])
VIEW(STRUCT(MKPOLS(pilastri)+MKPOLS(traveLongitudinale)
    + MKPOLS(traviTrasversali)))
VIEW(EXPLODE(1.2,1.2,1)(MKPOLS(pilastri)+MKPOLS(traveLongitudinale)
    + MKPOLS(traviTrasversali)))
```

Unit frame in normalized space

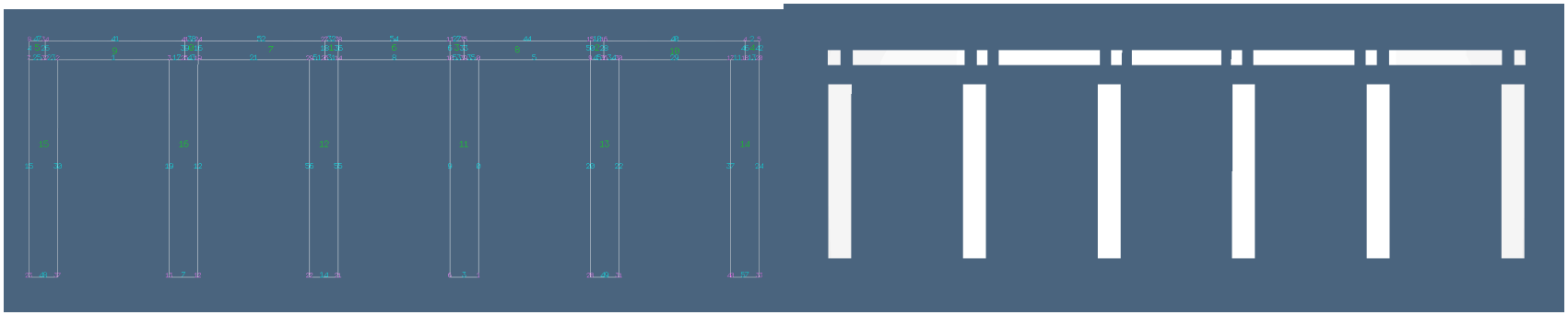


Figure 9:2-cells of a single-storey multi-span frame

Remark

This approach was abandoned via numerical problems (no vertex snap)

Grid for multi-storey frame (MUCH better solution)

```
""" Grid for multi-storey frame """

filename = 'grid.svg'
linesGrid = svg2lines(filename)
VIEW(STRUCT(AA(POLYLINE)(linesGrid)))
```

Grid for multi-storey frame

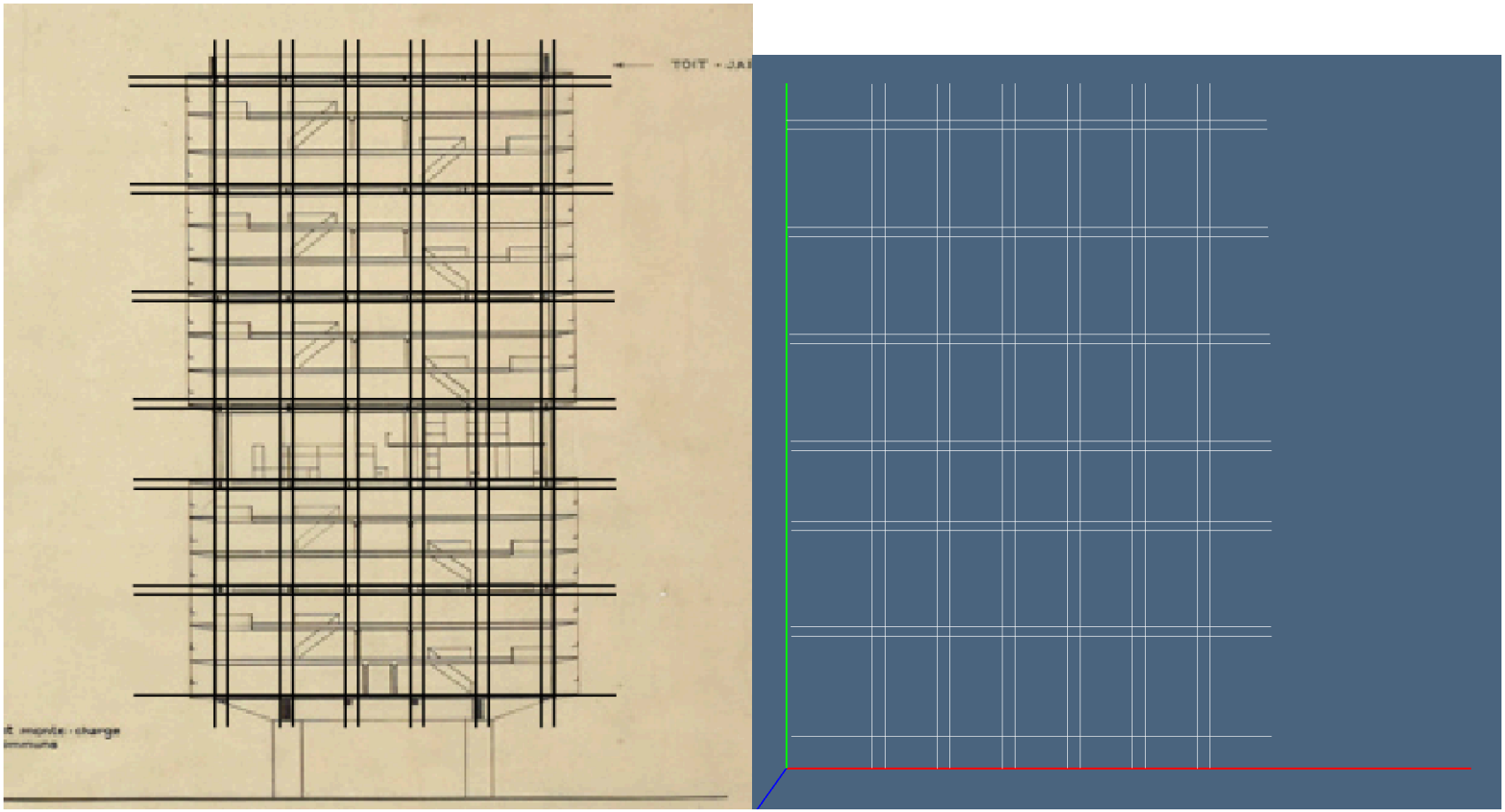
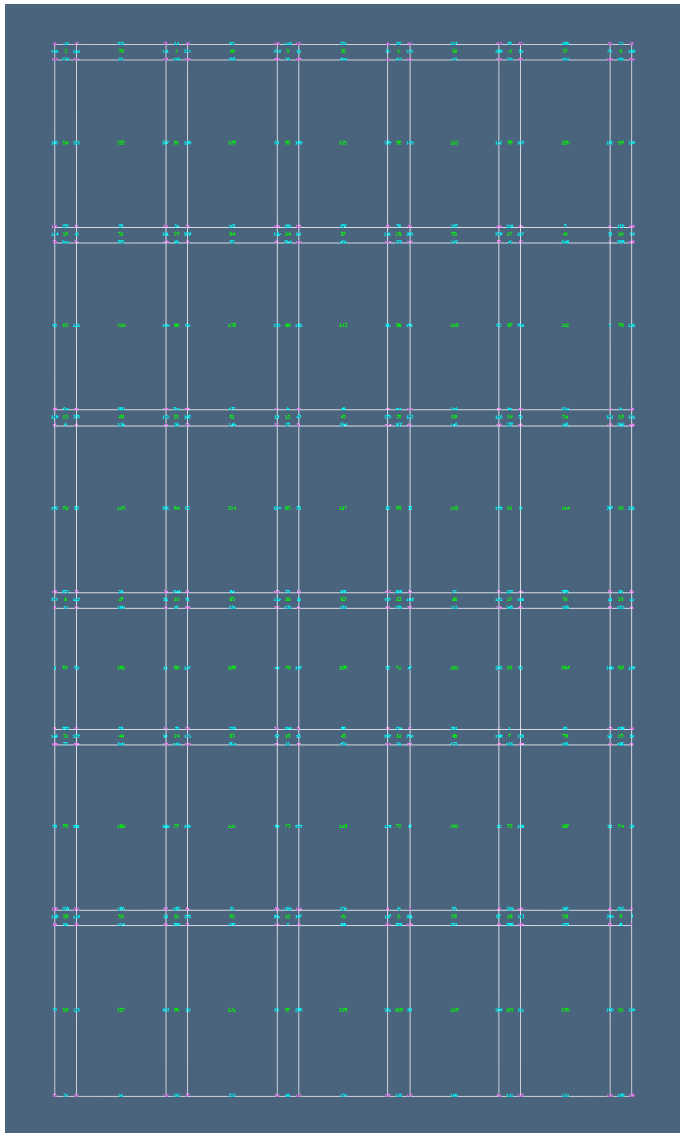


Figure 10:Line grid derived from the building section drawings

Grid for multi-storey frame (MUCH better solution)

```
""" Grid for multi-storey frame """
P,FP,EP,polygons = larFromLines(linesGrid)
PP = AA(LIST)(range(len(P)))
submodel = STRUCT(MKPOLS((P,EP)))
VIEW(larModelNumbering(1,1,1)(P,[PP,EP,FP],submodel,0.025))
```

Grid for multi-storey frame (MUCH better solution)



Transformation to world coordinates

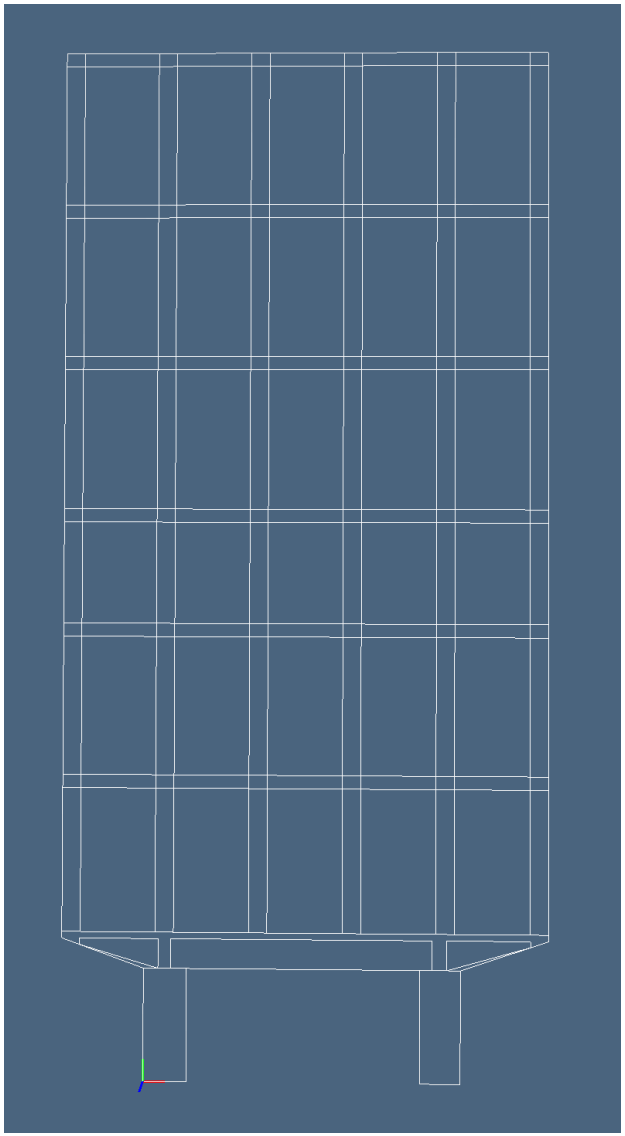
linear unit = Meter

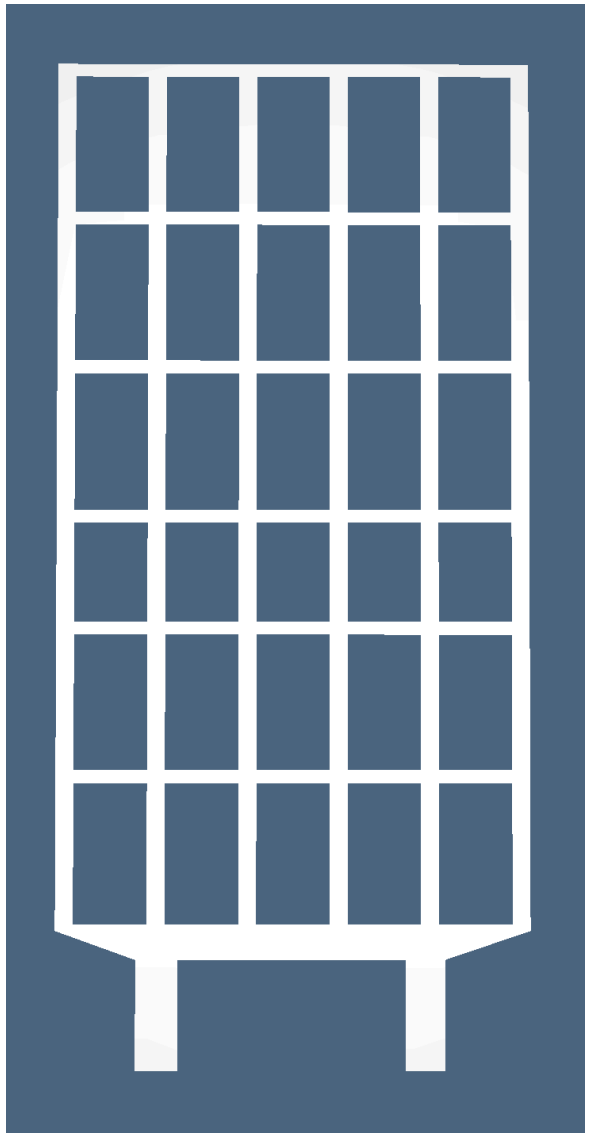
```
""" Transformation to world coordinates """

Delta_x_telaio = P[130][0]-P[63][0]
scale_telaio = delta_x_basement/Delta_x_telaio
X = ((mat(P)-P[63]) * scale_telaio + W[29]).tolist()
VIEW(SKEL_1(STRUCT(MKPOLS((X,FP)) + MKPOLS(pillar)
                   + MKPOLS(slab))))
```

Removal of empty cells from structural frame

```
""" Removal of empty cells from structural frame """

campate = sorted([127,131,129,128,130,107,108,110,111,109,
102,106,105,103,104,
114,115,117,124,116,119,125,113,118,112,120,123,121,126,122])

campate == range(102,132)
len(FP) == 132
telaio = (X,[FP[k] for k in range(102)])
VIEW(STRUCT(MKPOLS(telaio)+MKPOLS(pillar)+MKPOLS(slab)))
VIEW(EXPLODE(1.2,1.2,1)(MKPOLS(telaio)+MKPOLS(pillar)+MKPOLS(slab)))
```

Removal of empty cells from structural frame

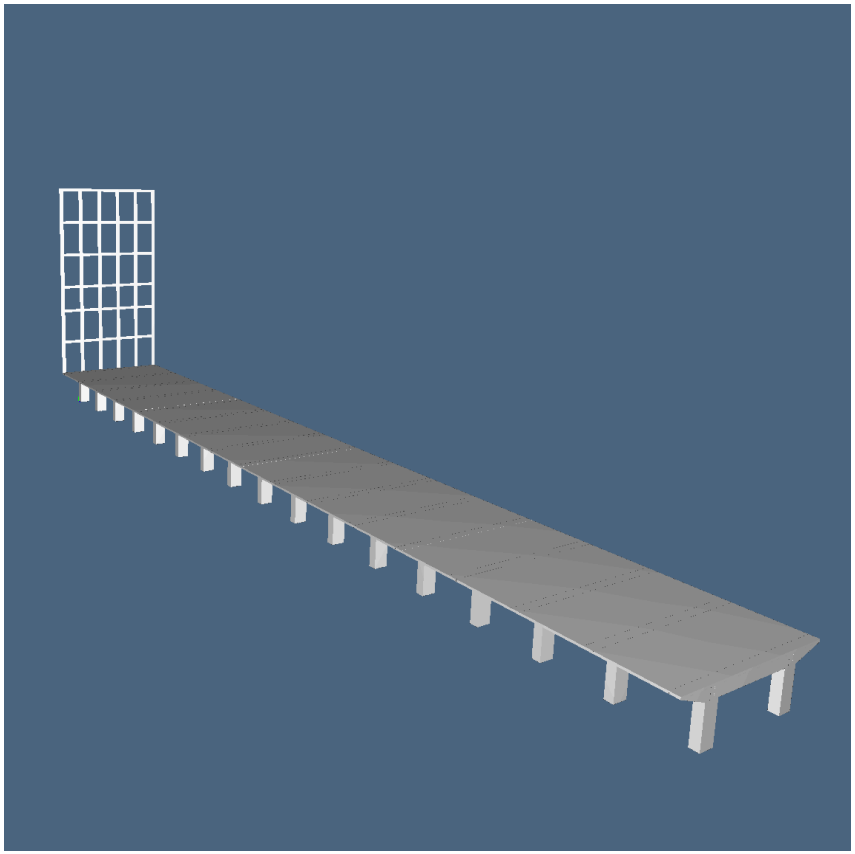




3D modeling

Construction of 3D structural frame

```
""" Positioning of 3D structural frame """
VIEW(STRUCT(MKPOLS(telaio)+MKPOLS(pillars)+MKPOLS(slabs)))
```



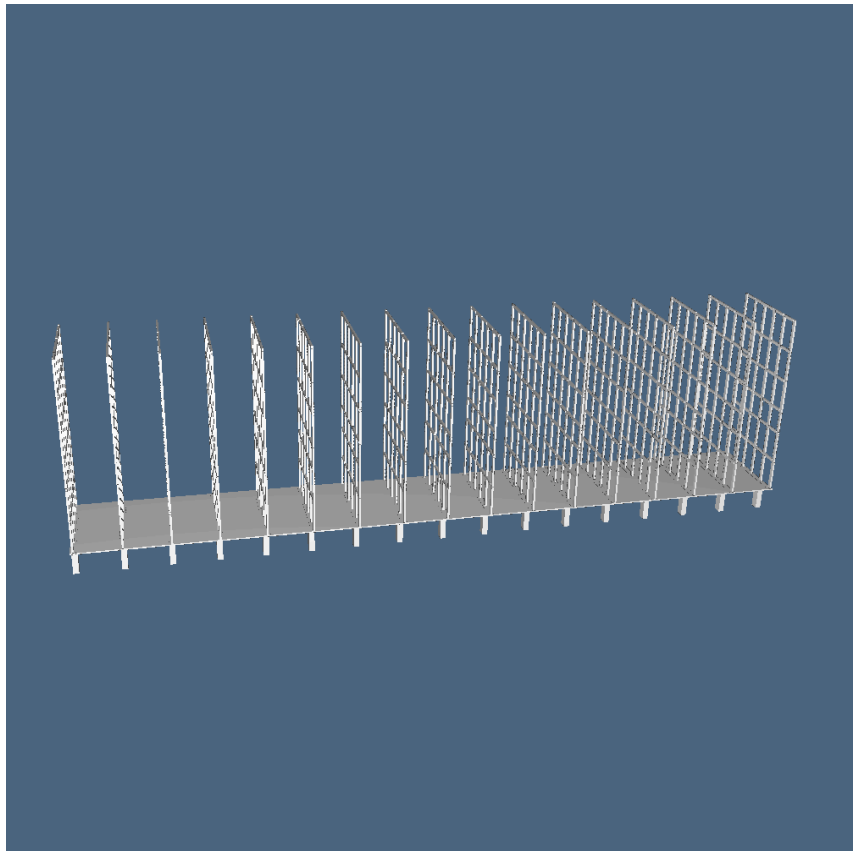
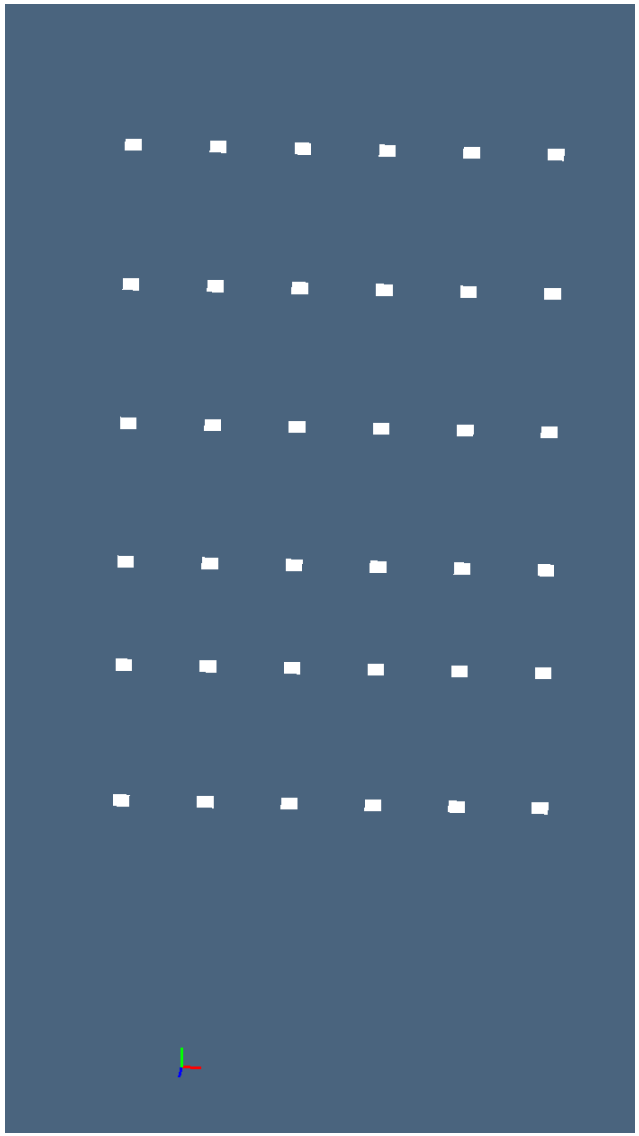


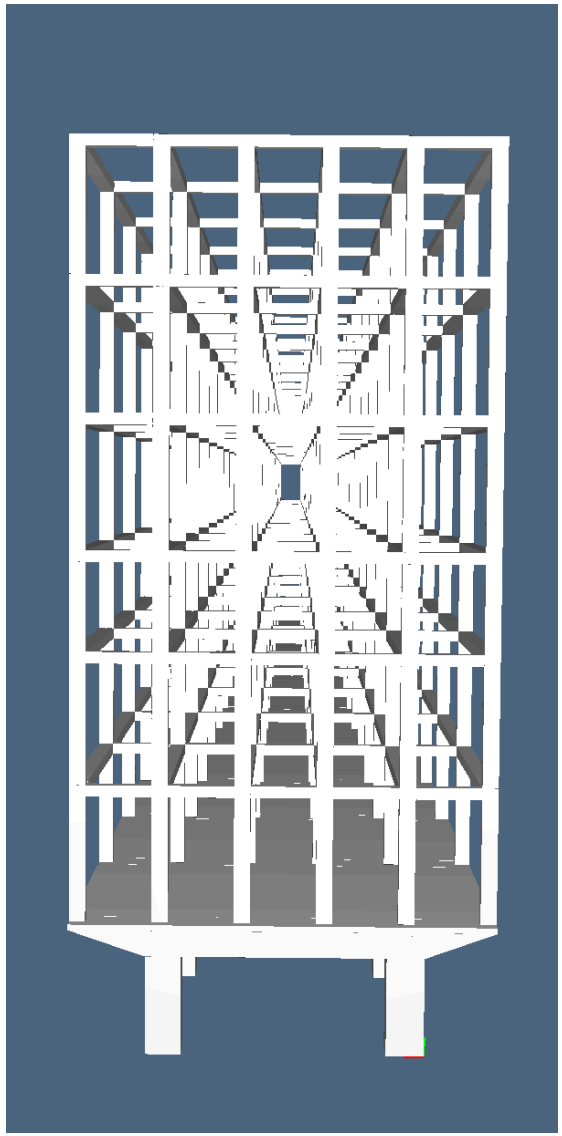
Figure 13:Positioning of initial 2D section

Construction of 3D structural frame

```
""" Construction of 3D structural frame """

concreteFramePattern = 16*[-0.4,0.4,-0.4, -8.4]+[-0.4,0.4,-0.4]
concreteFrame = larModelProduct([telaio,
                                 larQuote1D(concreteFramePattern)])
VIEW(STRUCT(MKPOLS(concreteFrame)+MKPOLS(pillars)+MKPOLS(slabs)))

crossBeams = sorted([30,22,12,6,20,8,15,7,32,16,24,31,9,18,28,
    33,17,25,19,14,35,13,21,11,26,23,34,29,27,10,0,4,2,1,5,3])

assert range(36) == sorted(crossBeams)
```

Modeling the crossBeams

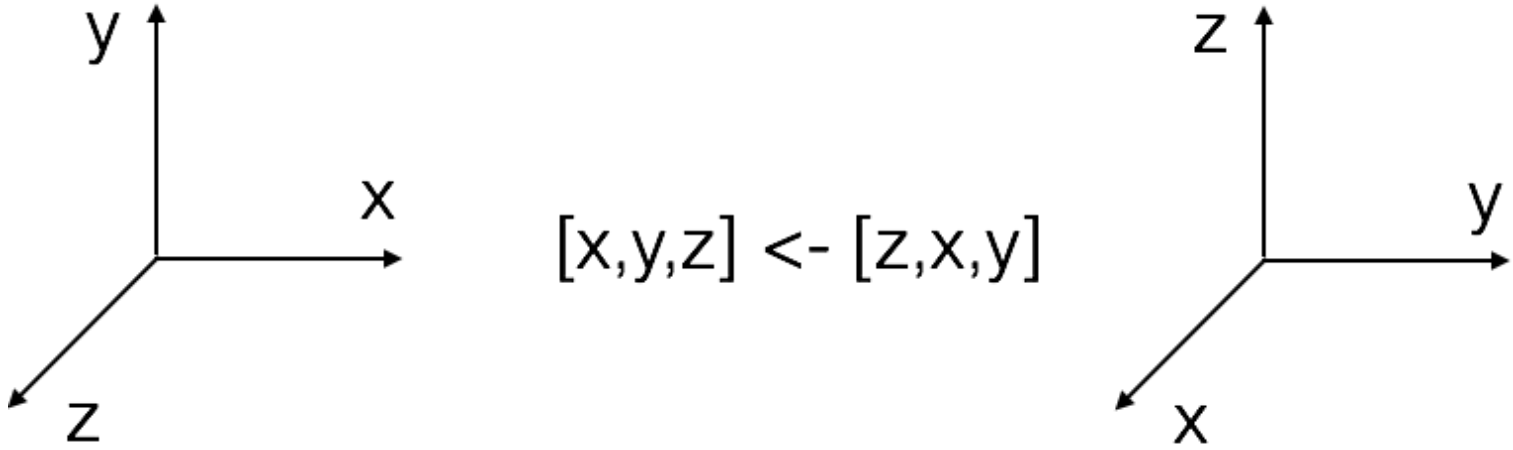




Modeling the crossBeams

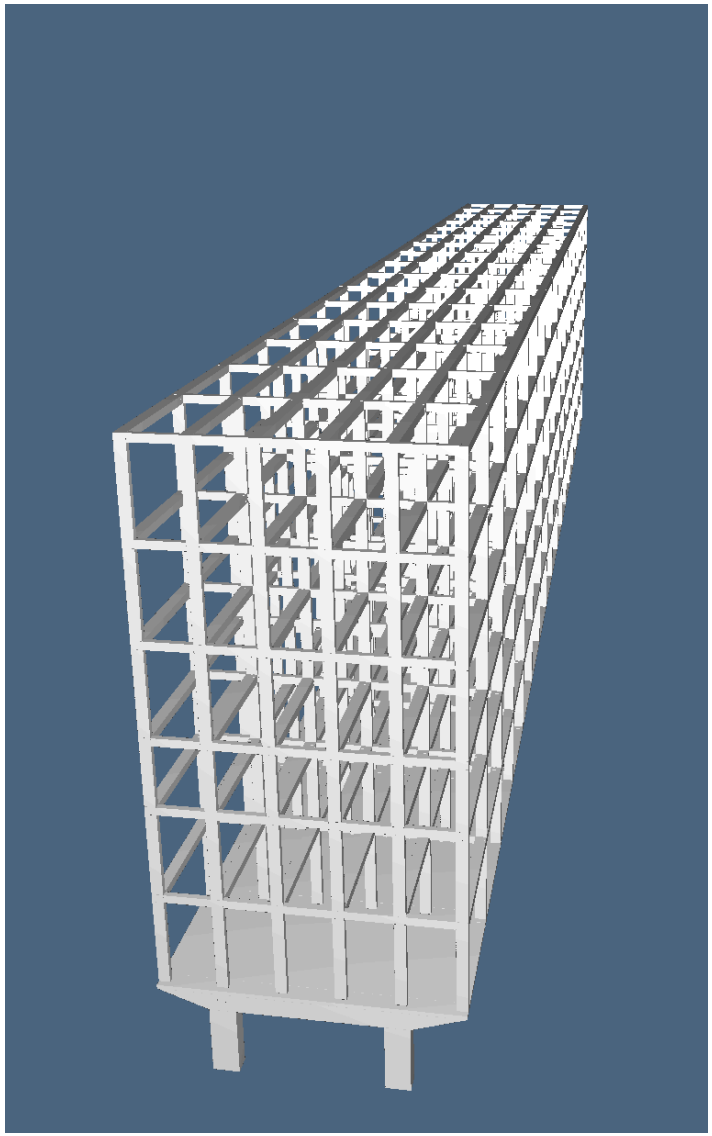
```
crossBeam = (X,[FP[k] for k in range(36)])
VIEW(STRUCT(MKPOLS(crossBeam)))

concreteCrossBeamsPattern = [-0.4]+16*[0.8,8.8]+[0.4]
concreteCrossBeams = larModelProduct([crossBeam,
                        larQuote1D(concreteCrossBeamsPattern)])
VIEW(STRUCT(MKPOLS(concreteFrame) + MKPOLS(pillars)
            + MKPOLS(slabs) + MKPOLS(concreteCrossBeams)))
```

Rotation to engineering coordinates




```
""" Rotation to engineering coords (via permutation) """

frame = struct2lar(Struct([ pillars,concreteFrame,slabs,
                            concreteCrossBeams ]))
Q,CQ = frame
frame = AA(CONS([S3,S1,S2]))(Q),CQ    # (see above)
VIEW(STRUCT(MKPOLS(frame) ))
```

Rotation to engineering coordinates



Rotation to engineering coordinates

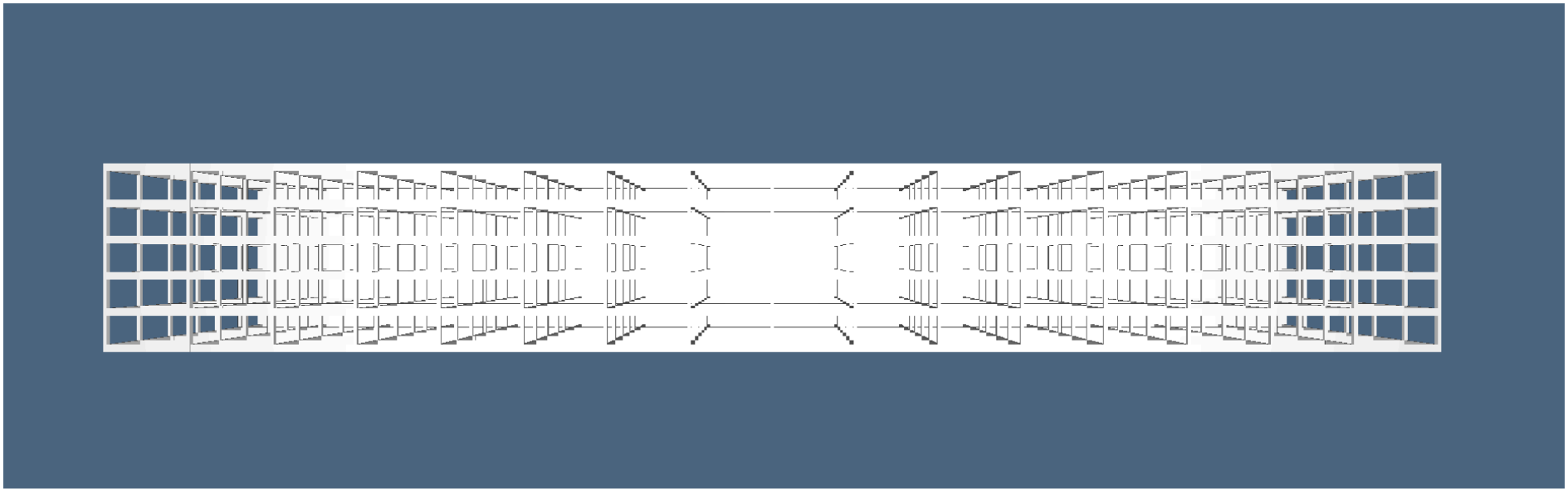


Figure 16:View of building frame from above

Input of building entrance

```
""" Input of building entrance """

filename = "entree.lines"
lines = lines2lines(filename)
VIEW(STRUCT(AA(POLYLINE)(lines)))

Y,FY,EY,polygons = larFromLines(lines)
VIEW(STRUCT(MKPOLS((Y,EY))))
VIEW(STRUCT(MKTRIANGLES((Y,FY,EY))))

YY = AA(LIST)(range(len(Y)))
submodel = STRUCT(MKPOLS((Y,EY)))
VIEW(larModelNumbering(1,1,1)(Y,[YY,EY,FY],submodel,0.25))
```

Input of building entrance

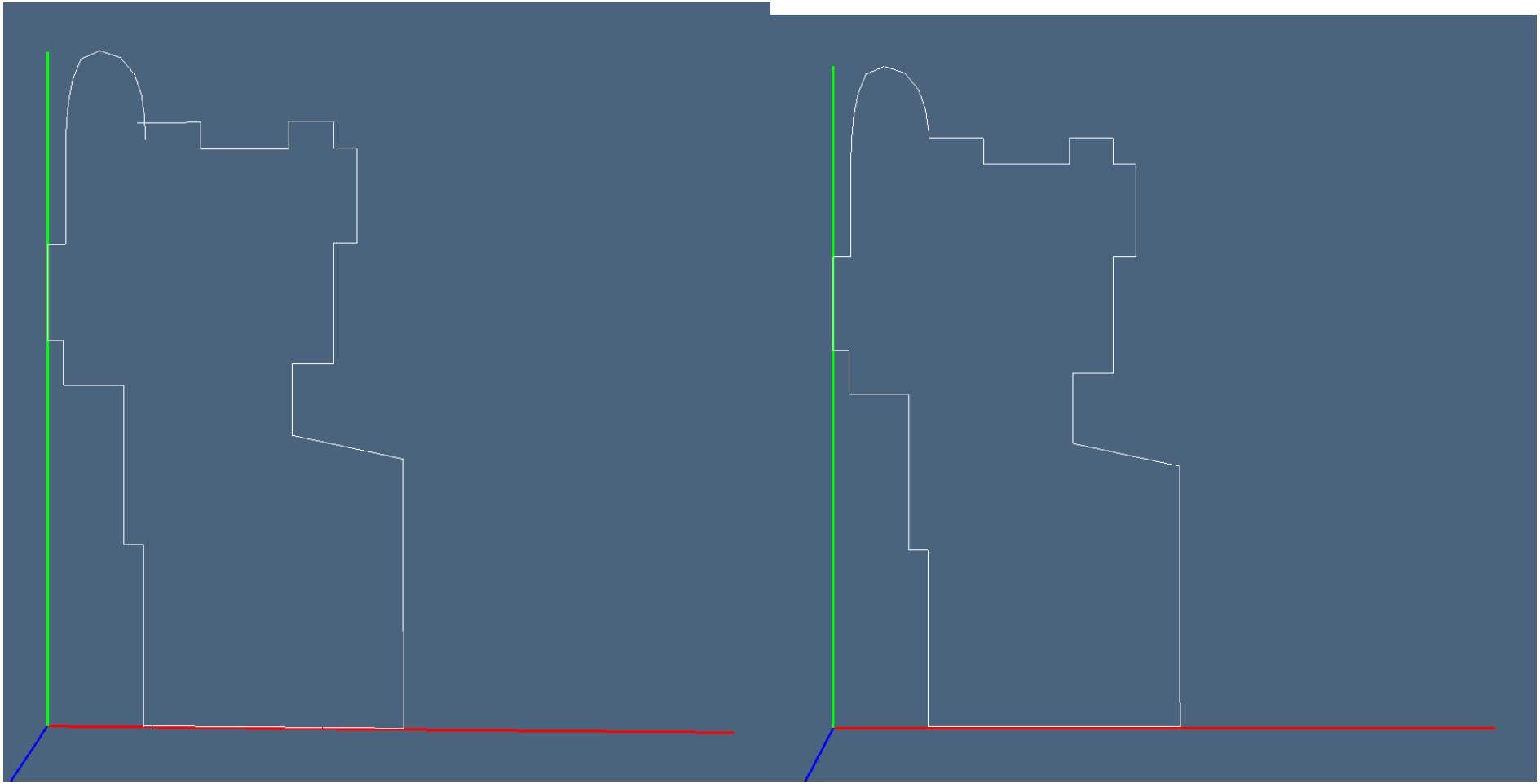


Figure 17:example caption

Input of building entrance

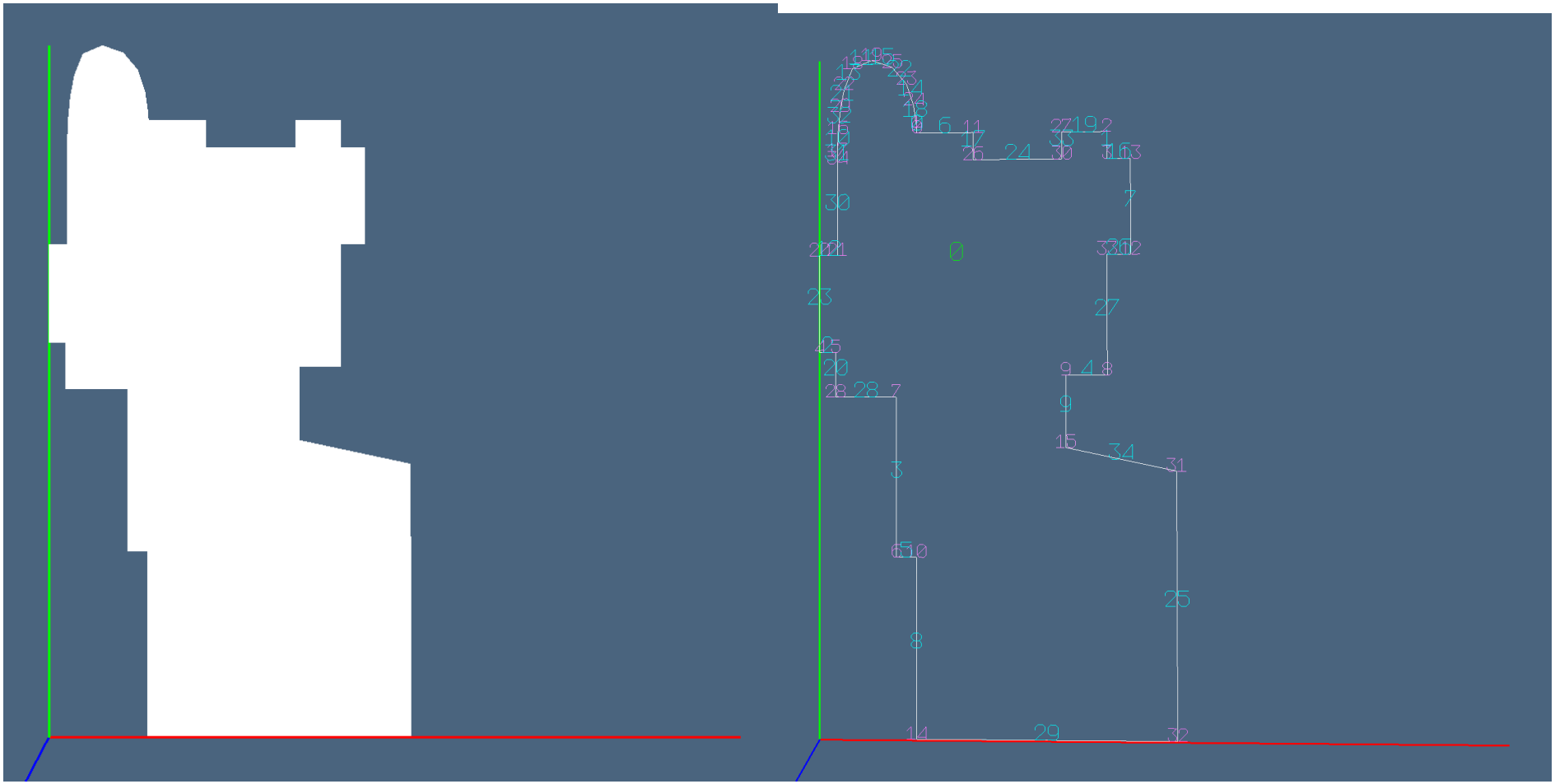


Figure 18:example caption

Entrance scaling and positioning

```
""" entrance scaling and positioning """

assert EY[29] == [32, 14]
scaling = 9./(Y[32][0] - Y[14][0])*1.5
Z = ((mat(Y)-Y[14])*scaling + [8*9.6+.8,-3.63]).tolist()

entree = (Z,FY,EY)
VIEW(STRUCT(MKPOLS(frame)+MKTRIANGLES(entree)))
```

Entrance scaling and positioning

```
entranceSurface = entree[0],entree[2]
stairCasePattern = 8*[6.0]
stairCase = larModelProduct([entranceSurface,
                larQuote1D(stairCasePattern)])
VIEW(STRUCT(MKPOLS(frame)+MKPOLS(stairCase)))
```

Entrance scaling and positioning

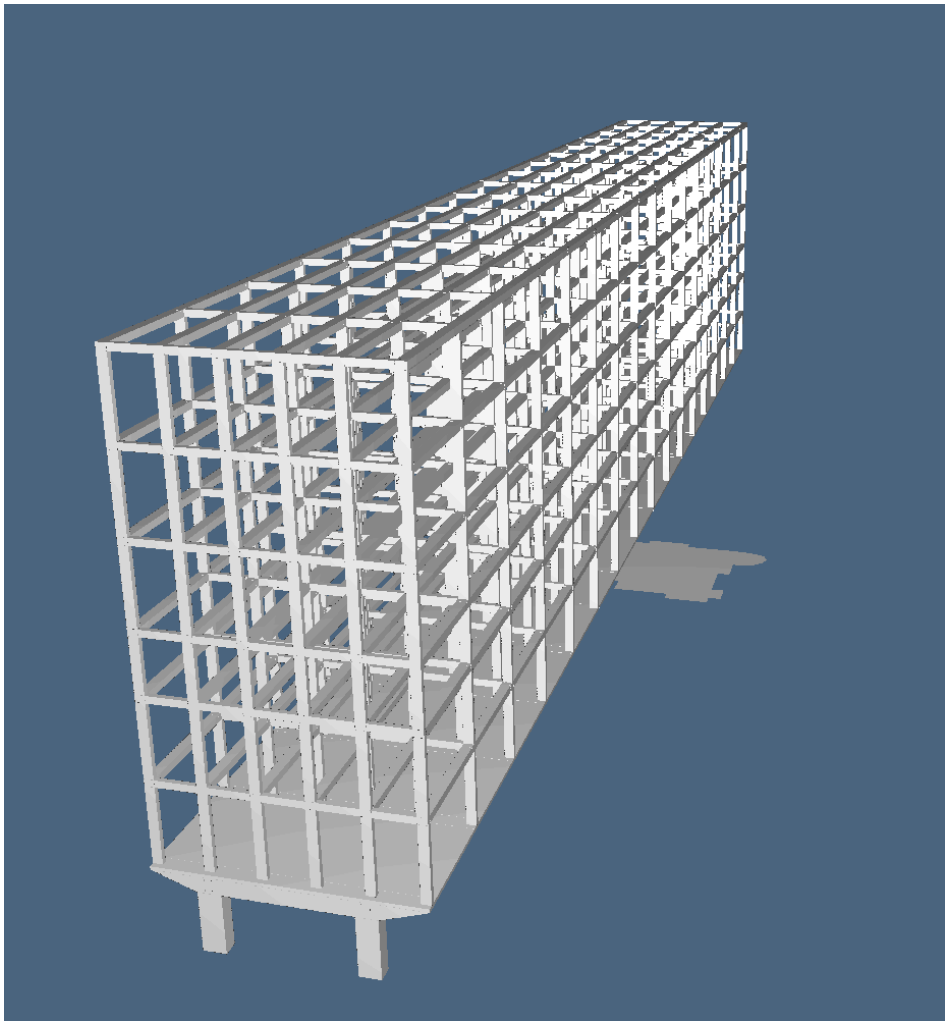


Figure 19:2D entrance correctly positioned

Entrance scaling and positioning

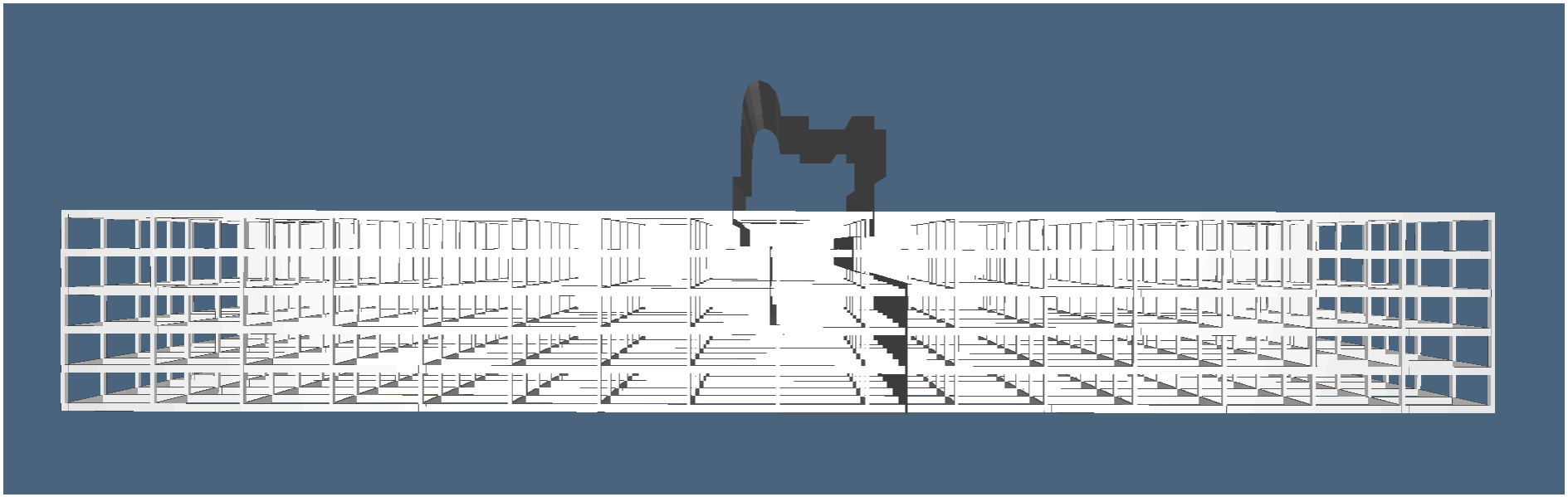


Figure 20:View from above

Entrance scaling and positioning

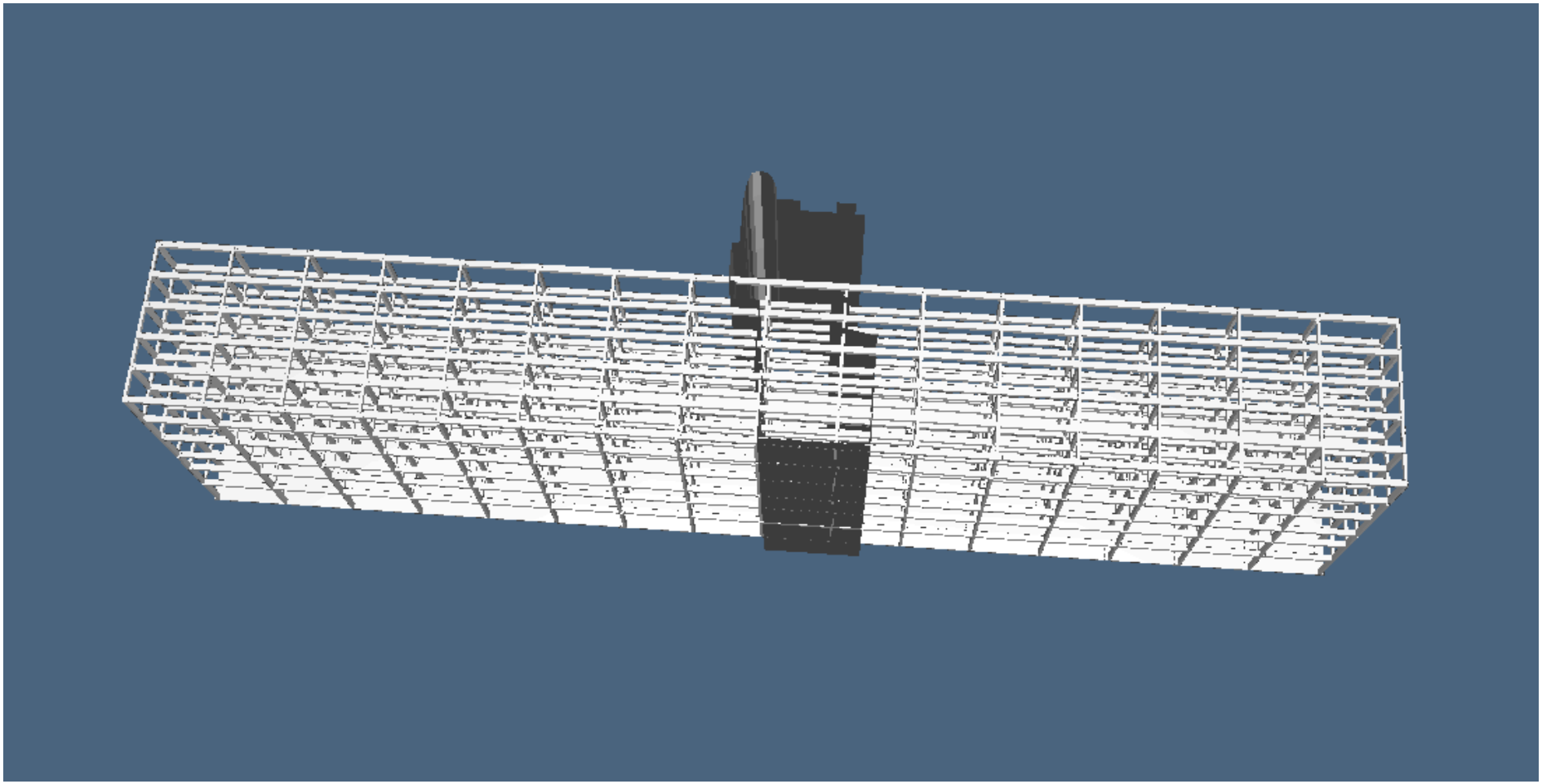


Figure 21:View from above