

# Structured input to HIJSON

## Towards indoor mapping and IoT modeling \*

Alberto Paoluzzi

December 1, 2015

### Abstract

This module aims to prototype the process of entering geometric data representing a complex building, to provide them explicit semantic and a hierarchical model. The generated LAR structures [DPS14, PDFJ15] are finally output to HIJSON format, an experimental data format extending GEOJSON for applications of indoor mapping and the Internet-of-Things. In HIJSON a strongly simplified building model, yet sufficient for useful purposes, accomodates the knowledge concerning the use models of the building and the set of interior devices and their connection.

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Implementation</b>	<b>2</b>
2.1	Input of geometry from external drawings . . . . .	2
2.2	Emulation of interactive graphics input . . . . .	2
2.3	Structural operations . . . . .	4
<b>3</b>	<b>Exporting the library</b>	<b>4</b>
<b>4</b>	<b>Test examples</b>	<b>5</b>
4.1	Example of a complex building model: part definition . . . . .	5
4.2	Example of a complex building model: integration . . . . .	10

---

\*This document is part of the *Linear Algebraic Representation with CoChains* (LAR-CC) framework [CL13]. December 1, 2015

# 1 Introduction

## 2 Implementation

### 2.1 Input of geometry from external drawings

**File input and computation of cellular complex** The modeling process starts with the input of a `.svg` file, the W3C standard for 2D vector graphics on the web. The file must only contain (by now) `<rect>` and `<line>` graphics primitives. The geometric data generate the partition of the plane induced by an arrangement of intersecting lines. The arrangement of (fragmented) lines is finally transformed into a 2D *LAR model*, made by the triple  $V, FV, EV$  of vertices, faces and edges. We notice the the LAR input is normalized by the `larFromLines` function: all vertices in  $V$  are transformed in the standard plane interval  $[0, 1]^2$ .

```
<SVG file input and computation of cellular complex 2a>  $\equiv$ 
    """ File input and computation of cellular complex """
    def svg2lar(filename):
        lines = svg2lines(filename)
        larModel = larFromLines(lines)
        V, FV, EV = larModel
        return larModel
     $\diamond$ 
```

Macro referenced in [4b](#).

### 2.2 Emulation of interactive graphics input

In this section two functions are given to emulate two graphics input primitives, respectively.

**Emulation of input from “selection box”** The function accepts as input the LAR model  $V, FV, EV$  and a `queryBox` given in normalized coordinates. It returns `vertexSubset`  $\subseteq V$ , `faceSubset`  $\subseteq FV$ , `edgeSubset`  $\subseteq EV$ .

```
<Emulation of input from “selection box” 2b>  $\equiv$ 

    """ Emulation of input from “selection box” over a LAR normalized representation """
    from scipy import spatial

    def subComplexInBox(V, FV, EV, queryBox):
        (xmin, ymin), (xmax, ymax) = queryBox
        if xmin > xmax: xmin, xmax = xmax, xmin
        if ymin > ymax: ymin, ymax = ymax, ymin
        vdict = dict([(vcode(vert), k) for k, vert in enumerate(V)])
```

```

vertexSubset = [vdict[vcode((x,y))]] for x,y in V if xmin<=x<=xmax and ymin<=y<=ymax]
edgeSubset = [e for e,edge in enumerate(EV) if all([v in vertexSubset for v in edge])]
faceSubset = [f for f,face in enumerate(FV) if all([v in vertexSubset for v in face])]
return vertexSubset,faceSubset,edgeSubset

if __name__=="__main__":
    selectBox = ((0.45, 0.45), (0.65, 0.75))
    vertexSubset,faceSubset,edgeSubset = subComplexInBox(V,FV,EV,selectBox)
    #VIEW(EXPLODE(1.2,1.2,1.2)(MKPOLLS((V,[EV[e] for e in edgeSubset])) + [
        #COLOR(RED)(MK(selectBox[0])), COLOR(RED)(MK(selectBox[1]))]))
    #VIEW(EXPLODE(1.2,1.2,1.2)(MKPOLLS((V,[FV[f] for f in faceSubset])) + [
        #COLOR(RED)(MK(selectBox[0])), COLOR(RED)(MK(selectBox[1]))]))

```

◇

Macro referenced in 4b.

**Emulation of “pick” input over a LAR normalized representation** The function accepts as input the LAR model  $V, FV, EV$ , the incidence relation  $FE$ , that provides for each face the list of incident edges, and a  $queryPoint$  given in normalized coordinates. It returns  $vertexSubset \subseteq V$ ,  $faceSubset \subseteq FV$ ,  $edgeSubset \subseteq EV$ .

⟨ Emulation of “pick” input 3a ⟩ ≡

```

""" Emulation of ‘pick’ input over a LAR normalized representation """
def subComplexAroundPoint(V,FV,EV,FE,queryPoint):
    tree = spatial.cKDTree(V)
    pts = np.array([queryPoint])
    dist,closestVertex = tree.query(pts)
    VF = invertRelation(FV)
    closestFaces = VF[closestVertex]

    for face in closestFaces:
        faceEdges = [EV[e] for e in FE[face]]
        classify = pointInPolygonClassification((V,faceEdges))
        if classify(queryPoint) == "p_in":
            break
    vertexSubset = FV[face]
    edgeSubset = [EV[e] for e in FE[face]]
    faceSubset = [face]
    return vertexSubset,faceSubset,edgeSubset

if __name__=="__main__":
    FE = crossRelation(FV,EV)
    queryPoint = (0.6,0.58)
    vertexSubset,faceSubset,edgeSubset = subComplexAroundPoint(V,FV,EV,FE,queryPoint)
    ##VIEW(EXPLODE(1.2,1.2,1.2)(MKPOLLS((V,[EV[e] for e in FE[faceSubset[0]]])) + [
        #COLOR(RED)(MK(queryPoint))]) )

```

◇

Macro referenced in 4b.

**From LAR chain to colored HPCs** The function `cells2hpcs` is used to assign a given `k` color to the cells of a chain (cell subset) of a LAR model `V,FV`. The function returns a list of HPC (colored) objects.

```
⟨ From LAR chain to colored HPCs 3b ⟩ ≡  
    """ From LAR chain to colored HPCs """  
    def cells2hpcs(V,FV,cells,k):  
        colors = [RED, GREEN, BLUE, CYAN, MAGENTA, YELLOW, WHITE, PURPLE, BROWN]  
        return AA(COLOR(colors[k]))(MKPOLs((V,[FV[f] for f in cells])))  
    ◇
```

Macro referenced in 4b.

## 2.3 Structural operations

### From chains to structures

```
⟨ From chains to structures 4a ⟩ ≡  
    """ From chains to structures """  
    def chain2structs(V,FV,EV,FE):  
        def chain2structs0(args):  
            if args == ([],[],[]): return  
            chain,chainName,classtype = args  
            struct = []  
            for cell in chain:  
                vs = [V[v] for v in FV[cell]]  
                vdict = dict([[vcode(vert),k] for k,vert in enumerate(vs)])  
                facetEdges = [[V[v] for v in EV[e]] for e in FE[cell]]  
                ev = [(vdict[vcode(v1)], vdict[vcode(v2)]) for v1,v2 in facetEdges]  
                fv = [range(len(vs))]  
                shape = vs,fv,ev  
                struct += [ Struct([ shape ], name=None, category="room" ) ]  
            out = Struct( struct, name=chainName, category=classtype )  
            return out  
        return chain2structs0  
    ◇
```

Macro referenced in 4b.

## 3 Exporting the library

```
"larlib/larlib/hijson.py" 4b ≡
```

```

""" Module for Structured input to HIJSON """
from larlib import *
from copy import copy
DEBUG = False

⟨SVG file input and computation of cellular complex 2a⟩
⟨Emulation of input from “selection box” 2b⟩
⟨Emulation of “pick” input 3a⟩
⟨From LAR chain to colored HPCs 3b⟩
⟨From chains to structures 4a⟩
◇

```

## 4 Test examples

### 4.1 Example of a complex building model: part definition

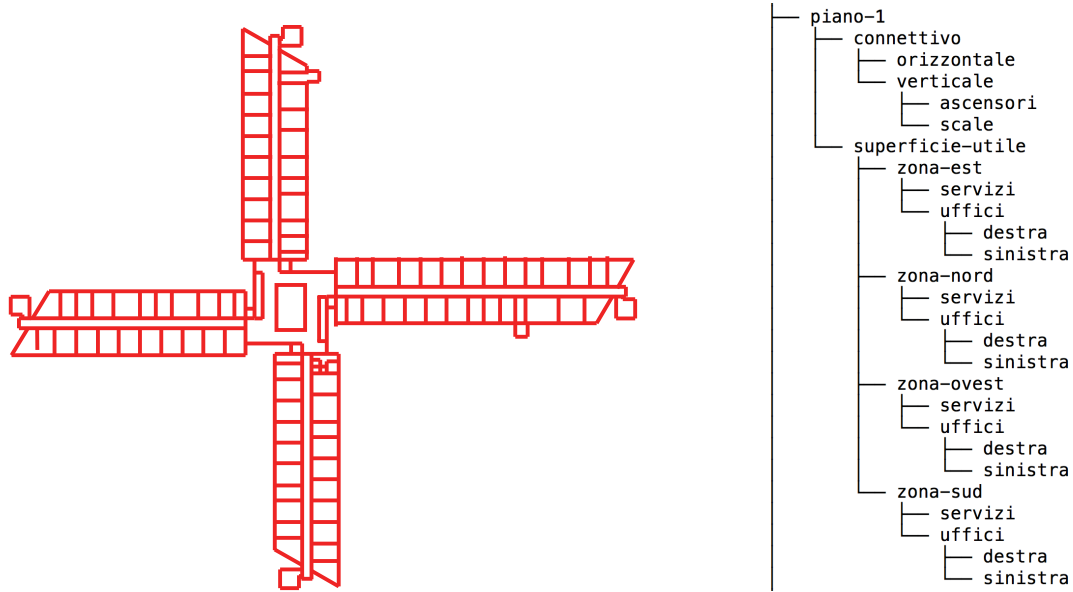


Figure 1: The input SVG drawing of the typical floor layout.

### Visualization of cell numbering in a 2D complex

```

⟨Visualization of cell numbering in a 2D complex 5a⟩ ≡
    """ Visualization of cell numbering in a 2D complex """
    VV = AA(LIST)(range(len(V)))
    submodel = STRUCT(MKPOLS((V,EV)))

```

```
#VIEW(larModelNumbering(1,1,1)(V,[VV,EV,FV[:-1]],submodel,2.5))
```

◇

Macro referenced in 10b.

## Ala nord (HPCs)

⟨Ala nord 5b⟩ ≡

```
FE = crossRelation(FV,EV)
chainsToStruct = chain2structs(V,FV,EV,FE)

""" Ala nord """
boxes = [0 for k in range(64)]
point = [0 for k in range(64)]
boxes[0] = array([[0.431, 0.607], [0.474, 0.91]])*scaleFactor #[V[k] for k in [39,208]]
boxes[1] = array([[0.416, 0.657], [0.372, 0.953]])*scaleFactor #[V[k] for k in [162,39]]
boxes[2] = array([[0.416, 0.627], [0.431, 0.986]])*scaleFactor #[V[k] for k in [206,247]]
boxes[3] = array([[0.431, 0.607], [0.448, 0.627]])*scaleFactor #[V[k] for k in [39,7]]
boxes[4] = array([[0.431, 0.91], [0.494, 0.929]])*scaleFactor #[V[k] for k in [213,234]]
boxes[5] = array([[0.431, 0.97], [0.466, 1.0]])*scaleFactor #[V[k] for k in [58,88]]
boxes[27] = array([[0.416, 0.627], [0.372, 0.657]])*scaleFactor #[V[k] for k in [110,82]]

point[0] = array([0.394, 0.9625])*scaleFactor #CCOMB([V[k] for k in [190,197]])
point[1] = array([0.4525, 0.9325])*scaleFactor #CCOMB([V[k] for k in [166,159]])

piano1_superficieUtile_zonaNord_uffici_destra = subComplexInBox(V,FV,EV,boxes[0])[1]
piano1_superficieUtile_zonaNord_uffici_sinistra = subComplexInBox(V,FV,EV,boxes[1])[1]
piano1_connettivo_orizzontale_zonaNord = subComplexInBox(V,FV,EV,boxes[2])[1]
piano1_connettivo_verticale_zonaNord_ascensore = subComplexInBox(V,FV,EV,boxes[3])[1]
piano1_connettivo_verticale_zonaNord_ascensore += subComplexInBox(V,FV,EV,boxes[4])[1]
piano1_connettivo_verticale_zonaNord_scale = subComplexInBox(V,FV,EV,boxes[5])[1]
piano1_superficieUtile_zonaNord_servizi = subComplexAroundPoint(V,FV,EV,FE,point[0])[1]
piano1_superficieUtile_zonaNord_servizi += subComplexAroundPoint(V,FV,EV,FE,point[1])[1]
piano1_superficieUtile_zonaNord_servizi += subComplexInBox(V,FV,EV,boxes[27])[1]

piano1N = [piano1_superficieUtile_zonaNord_uffici_destra, piano1_superficieUtile_zonaNord_uffici
```

◇

Macro defined by 5b, 6a.

Macro referenced in 10b.

⟨Ala nord 6a⟩ ≡

```
""" Ala nord """
piano1N_nomi = ["piano1_superficieUtile_zonaNord_uffici_destra", "piano1_superficieUtile_zonaNord_uffici_sinistra"]
piano1N_categorie = ["uffici","uffici","corridoi","ascensori","scale","servizi"]
p1N = zip(piano1N,piano1N_nomi,piano1N_categorie)
piano1_zonaNord = Struct(AA(chainsToStruct)(p1N),"piano1_zonaNord","ala")
#VIEW(SKEL_1(STRUCT(MKPOLS(struct2lar(piano1_zonaNord))))))
```

```
nord = CAT([cells2hpcs(V,FV,chain,k) for k,chain in enumerate(piano1N)])
#VIEW(EXPLODE(1.2,1.2,1.2)(nord))
◇
```

Macro defined by [5b](#), [6a](#).  
Macro referenced in [10b](#).

## Ala est (HPCs)

⟨Ala est 6b⟩ ≡

```
""" Ala est """
boxes[6] = array([[0.019, 0.533], [0.376, 0.577]])*scaleFactor #[V[k] for k in [241,29]]
boxes[7] = array([[0.07, 0.474], [0.343, 0.518]])*scaleFactor #[V[k] for k in [264,148]]
boxes[8] = array([[0.013, 0.518], [0.376, 0.533]])*scaleFactor #[V[k] for k in [22,63]]
boxes[9] = array([[0.376, 0.533], [0.39, 0.549]])*scaleFactor #[V[k] for k in [63,92]]
boxes[10] = array([[0.001, 0.474], [0.07, 0.518]])*scaleFactor #[V[k] for k in [263,265]]
boxes[11] = array([[0.343, 0.474], [0.376, 0.518]])*scaleFactor #[V[k] for k in [84,149]]

point[2] = array([0.015, 0.5535])*scaleFactor #CCOMB([V[k] for k in [228,14]])

piano1_superficieUtile_zonaEst_uffici_destra = subComplexInBox(V,FV,EV,boxes[6])[1]
piano1_superficieUtile_zonaEst_uffici_sinistra = subComplexInBox(V,FV,EV,boxes[7])[1]
piano1_connettivo_orizzontale_zonaEst = subComplexInBox(V,FV,EV,boxes[8])[1]
piano1_connettivo_verticale_zonaEst_ascensore = subComplexInBox(V,FV,EV,boxes[9])[1]
piano1_connettivo_verticale_zonaEst_scale = subComplexAroundPoint(V,FV,EV,FE,point[2])[1]
piano1_superficieUtile_zonaEst_servizi = subComplexInBox(V,FV,EV,boxes[10])[1]
piano1_superficieUtile_zonaEst_servizi += subComplexInBox(V,FV,EV,boxes[11])[1]

piano1E = [piano1_superficieUtile_zonaEst_uffici_destra, piano1_superficieUtile_zonaEst_uffici.
◇
```

Macro defined by [6b](#), [7a](#).  
Macro referenced in [10b](#).

⟨Ala est 7a⟩ ≡

```
""" Ala est """
piano1E_nomi = ["piano1_superficieUtile_zonaEst_uffici_destra", "piano1_superficieUtile_zonaEs
piano1E_categorie = ["uffici","uffici","corridoi","ascensori","scale","servizi"]
p1E = zip(piano1E,piano1E_nomi, piano1E_categorie)
piano1_zonaEst = Struct(AA(chainsToStruct)(p1E), "piano1_zonaEst", "ala")
#VIEW(SKEL_1(STRUCT(MKPOLS(struct2lar(piano1_zonaEst))))))

est = CAT([cells2hpcs(V,FV,chain,k) for k,chain in enumerate(piano1E)])
#VIEW(EXPLODE(1.2,1.2,1.2)(est + nord))
◇
```

Macro defined by [6b](#), [7a](#).  
Macro referenced in [10b](#).

## Ala sud (HPCs)

⟨ Ala sud 7b ⟩ ≡

```
"" Ala sud ""
boxes[12] = array([[0.467, 0.138], [0.423, 0.476]])*scaleFactor #[V[k] for k in [252,47]]
boxes[13] = array([[0.482, 0.145], [0.525, 0.445]])*scaleFactor #[V[k] for k in [241,126]]
boxes[14] = array([[0.482, 0.476], [0.467, 0.116]])*scaleFactor #[V[k] for k in [254,232]]
boxes[15] = array([[0.449, 0.476], [0.467, 0.493]])*scaleFactor #[V[k] for k in [40,237]]
boxes[16] = array([[0.431, 0.101], [0.467, 0.131]])*scaleFactor #[V[k] for k in [259,2]]
boxes[17] = array([[0.482, 0.445], [0.525, 0.476]])*scaleFactor #[V[k] for k in [155,248]]
boxes[18] = array([[0.525, 0.104], [0.482, 0.145]])*scaleFactor #[V[k] for k in [111,241]]

piano1_superficieUtile_zonaSud_uffici_destra = subComplexInBox(V,FV,EV,boxes[12])[1]
piano1_superficieUtile_zonaSud_uffici_sinistra = subComplexInBox(V,FV,EV,boxes[13])[1]
piano1_connettivo_orizzontale_zonaSud = subComplexInBox(V,FV,EV,boxes[14])[1]
piano1_connettivo_verticale_zonaSud_ascensore = subComplexInBox(V,FV,EV,boxes[15])[1]
piano1_connettivo_verticale_zonaSud_scale = subComplexInBox(V,FV,EV,boxes[16])[1]
piano1_superficieUtile_zonaSud_servizi = subComplexInBox(V,FV,EV,boxes[17])[1]
piano1_superficieUtile_zonaSud_servizi += subComplexInBox(V,FV,EV,boxes[18])[1]

piano1S = [piano1_superficieUtile_zonaSud_uffici_destra, piano1_superficieUtile_zonaSud_uffici.
```

Macro defined by 7b, 8a.  
Macro referenced in 10b.

⟨ Ala sud 8a ⟩ ≡

```
"" Ala sud ""
piano1S_nomi = ["piano1_superficieUtile_zonaSud_uffici_destra", "piano1_superficieUtile_zonaSu
piano1S_categorie = ["uffici","uffici","corridoi","ascensori","scale","servizi"]
p1S = zip(piano1S,piano1S_nomi, piano1S_categorie)
piano1_zonaSud = Struct(AA(chainsToStruct)(p1S), "piano1_zonaSud", "ala")
#VIEW(SKEL_1(STRUCT(MKPOLS(struct2lar(piano1_zonaSud))))))

sud = CAT([cells2hpcs(V,FV,chain,k) for k,chain in enumerate(piano1S)])
#VIEW(EXPLODE(1.2,1.2,1.2)(est + nord + sud))
```

Macro defined by 7b, 8a.  
Macro referenced in 10b.

## Ala ovest (HPCs)

⟨ Ala ovest 8b ⟩ ≡

```
"" Ala ovest ""
boxes[19] = array([[0.521, 0.526], [0.963, 0.568]])*scaleFactor #[V[k] for k in [169,202]]
boxes[20] = array([[0.555, 0.584], [0.955, 0.627]])*scaleFactor #[V[k] for k in [12,23]]
boxes[21] = array([[0.521, 0.568], [0.985, 0.584]])*scaleFactor #[V[k] for k in [209,204]]
```



```

boxes[22] = array([[0.506, 0.551], [0.521, 0.568]])*scaleFactor #[V[k] for k in [89,209]]
boxes[23] = array([[0.808, 0.504], [0.828, 0.526]])*scaleFactor #[V[k] for k in [270,77]]
boxes[24] = array([[0.955, 0.584], [0.997, 0.627]])*scaleFactor #[V[k] for k in [220,24]]
boxes[25] = array([[0.521, 0.584], [0.555, 0.627]])*scaleFactor #[V[k] for k in [11,144]]
boxes[26] = array([[1.0, 0.533], [0.97, 0.568]])*scaleFactor #[V[k] for k in [233,201]]

piano1_superficieUtile_zonaOvest_uffici_destra = subComplexInBox(V,FV,EV,boxes[19])[1]
piano1_superficieUtile_zonaOvest_uffici_sinistra = subComplexInBox(V,FV,EV,boxes[20])[1]
piano1_connettivo_orizzontale_zonaOvest = subComplexInBox(V,FV,EV,boxes[21])[1]
piano1_connettivo_verticale_zonaOvest_ascensore = subComplexInBox(V,FV,EV,boxes[22])[1]
piano1_connettivo_verticale_zonaOvest_ascensore += subComplexInBox(V,FV,EV,boxes[23])[1]
piano1_superficieUtile_zonaOvest_servizi = subComplexInBox(V,FV,EV,boxes[24])[1]
piano1_superficieUtile_zonaOvest_servizi += subComplexInBox(V,FV,EV,boxes[25])[1]
piano1_connettivo_verticale_zonaOvest_scale = subComplexInBox(V,FV,EV,boxes[26])[1]

```

```

piano10 = [piano1_superficieUtile_zonaOvest_uffici_destra, piano1_superficieUtile_zonaOvest_uf
◇

```

Macro defined by [8b](#), [9a](#).  
Macro referenced in [10b](#).

```

⟨Ala ovest 9a⟩ ≡
    """ Ala ovest """
    piano10_nomi = ["piano1_superficieUtile_zonaOvest_uffici_destra", "piano1_superficieUtile_zonaOvest_uffici_sinistra"]
    piano10_categorie = ["uffici","uffici","corridoi","ascensori","scale","servizi"]
    p10 = zip(piano10,piano10_nomi, piano10_categorie)
    piano1_zonaOvest = Struct(AA(chainsToStruct)(p10), "piano1_zonaOvest", "ala")
    #VIEW(SKEL_1(STRUCT(MKPOLS(struct2lar(piano1_zonaOvest))))))

    ovest = CAT([cells2hpcs(V,FV,chain,k) for k,chain in enumerate(piano10)])
    #VIEW(EXPLODE(1.2,1.2,1.2)(est + nord + sud + ovest))
◇

```

Macro defined by [8b](#), [9a](#).  
Macro referenced in [10b](#).

## Centro stella (HPCs)

```

⟨Centro stella 9b⟩ ≡
    """ Centro stella """
    piano1_connettivo_orizzontale_centroStella = [2]
    piano1_connettivo_verticale_centroStella_scale = [15,26]

    piano1C = [[],[],piano1_connettivo_orizzontale_centroStella,[], piano1_connettivo_verticale_centroStella_scale]
    centro = CAT([cells2hpcs(V,FV,chain,k) for k,chain in enumerate(piano1C)])
◇

```

Macro defined by [9bc](#).  
Macro referenced in [10b](#).

```

⟨ Centro stella 9c ⟩ ≡
    """ Centro stella """
    piano1C_nomi = [[],[],"piano1_connettivo_orizzontale_centroStella", [], "piano1_connettivo_ver"
    piano1C_categorie = [[],[],"corridoi",[], "ascensori"]
    p1C = zip(piano1C,piano1C_nomi, piano1C_categorie)
    piano1_centroStella = Struct(AA(chainsToStruct)(p1C), "piano1_centroStella", "centro")
    #VIEW(SKEL_1(STRUCT(MKPOLS(struct2lar(piano1_centroStella))))))

    #VIEW(EXPLODE(1.2,1.2,1.2)(est + nord + sud + ovest + centro))
    #VIEW(STRUCT(est + nord + sud + ovest + centro))
    ◇

```

Macro defined by [9bc](#).  
Macro referenced in [10b](#).

## 4.2 Example of a complex building model: integration

### Assembly of parts (HPCs)

```

⟨ Assemblaggio 10a ⟩ ≡
    """ Assemblaggio """
    p1 = p1N + p1S + p1E + p1O + p1C

    piano1_nomi = ["piano1_zonaNord", "piano1_zonaEst", "piano1_zonaSud", "piano1_zonaOvest", "pia
    piano1_categorie = ["ala","ala","ala","ala","centro"]
    piano1 = Struct(AA(chainsToStruct)(p1), "piano1", "level")

    #VIEW(SKEL_1(STRUCT(MKPOLS(struct2lar(piano1))))))

    V,boundaryEdges = structBoundaryModel(piano1)
    drawing = mkSignedEdges((V,boundaryEdges))
    #VIEW(drawing)

    polylines = boundaryModel2polylines((V,boundaryEdges))
    #VIEW(STRUCT(AA(POLYLINE)(polylines)))

    print boundaryPolylines(piano1)
    ◇

```

Macro referenced in [10b](#).

### Assembling floor layout generation

```

⟨ Assembling floor layout generation 10b ⟩ ≡
    """ Assembling floor layout generation """
    ⟨ Visualization of cell numbering in a 2D complex 5a ⟩
    ⟨ Ala nord 5b, ... ⟩

```

```

<Ala est 6b, ... >
<Ala sud 7b, ... >
<Ala ovest 8b, ... >
<Centro stella 9b, ... >
<Assemblaggio 10a>
◇

```

Macro referenced in 11ab.

```

<Tower example initialization 10c> ≡
    """ make the model of a layout floor """

    scaleFactor = 83.333

    filename = "test/svg/inters/plan.svg"
    larModel = svg2lar(filename)
    larModel = larApply(s(scaleFactor,scaleFactor))(larModel)
    V,FV,EV = larModel
    FV[2] += FV[71]      # for now :o)
◇

```

Macro referenced in 11ab.

```

"test/py/hijson/test01.py" 11a ≡
    """ make the model of a layout floor """
    from larlib import *

    <Tower example initialization 10c>
    <Assembling floor layout generation 10b>

    primoPiano = AA(list)([CAT(AA(S1)(p1N)),CAT(AA(S1)(p1E)),CAT(AA(S1)(p1S)),
                           CAT(AA(S1)(p1O)),CAT(AA(S1)(p1C))])
    primoPiano_nomi = ["piano1_zonaNord","piano1_zonaEst","piano1_zonaSud","piano1_zonaOvest","piano1_zonaCentro"]
    primoPiano_categorie = ["ala","ala","ala","ala","centro"]
    pianoPrimo = zip(primoPiano, primoPiano_nomi, primoPiano_categorie)
    piano_1 = Struct( AA(chainsToStruct)(pianoPrimo), "piano1", "level" )

    piano_1_3D = embedStruct(1)(piano_1,"3D")
    iot3d.printStruct2GeoJson("./",piano_1_3D)

    print "piano_1_3D =",piano_1_3D.category
    print "piano_1_3D.body[0] =",piano_1_3D.body[0].category
    print "piano_1_3D.body[0].body[0] =",piano_1_3D.body[0].body[0].category
◇

```

"test/py/hijson/test02.py" 11b ≡

```

""" make the 2.5 model of a building tower """
from larlib import *

< Tower example initialization 10c >
< Assembling floor layout generation 10b >

pianoNord3D = embedStruct(1)(piano1_zonaNord,"3D")
pianoEst3D = embedStruct(1)(piano1_zonaEst,"3D")
pianoSud3D = embedStruct(1)(piano1_zonaSud,"3D")
pianoOvest3D = embedStruct(1)(piano1_zonaOvest,"3D")
pianoCentro3D = embedStruct(1)(piano1_centroStella,"3D")
torreNord = Struct(4*[pianoNord3D,t(0,0,3)], "torreNord", "edificio")
torreEst = Struct(7*[pianoEst3D,t(0,0,3)], "torreEst", "edificio")
torreSud = Struct(7*[pianoSud3D,t(0,0,3)], "torreSud", "edificio")
torreOvest = Struct(7*[pianoOvest3D,t(0,0,3)], "torreOvest", "edificio")
torreCentro = Struct(7*[pianoCentro3D,t(0,0,3)], "torreCentro", "edificio")
torre = Struct([torreNord,torreEst,torreSud,torreOvest,torreCentro],"torre", "edificio")
V,FV,EV = struct2lar(torre)
#VIEW(EXPLODE(1.2,1.2,1.2)(MKPOLs((V,EV))))
#VIEW(STRUCT(MKPOLs((V,FV))))
◇

```

## References

- [CL13] CVD-Lab, *Linear algebraic representation*, Tech. Report 13-00, Roma Tre University, October 2013.
- [DPS14] Antonio DiCarlo, Alberto Paoluzzi, and Vadim Shapiro, *Linear algebraic representation for topological structures*, Comput. Aided Des. **46** (2014), 269–274.
- [PDFJ15] Alberto Paoluzzi, Antonio DiCarlo, Francesco Furiani, and Miroslav Jirik, *Cad models from medical images using lar*, Computer-Aided Design and Applications **13** (2015), To appear.