

000	054
001	055
002	056
003	057
004	058
005	059
006	060
007	061
008	062
009	063
010	064
011	065
012	066
013	067
014	068
015	069
016	070
017	071
018	072
019	073
020	074
021	075
022	076
023	077
024	078
025	079
026	080
027	081
028	082
029	083
030	084
031	085
032	086
033	087
034	088
035	089
036	090
037	091
038	092
039	093
040	094
041	095
042	096
043	097
044	098
045	099
046	100
047	101
048	102
049	103
050	104
051	105
052	106
053	107

Computational tools and file format for virtual interactive indoor mapping

Anonymous cvm submission

Paper ID ****

Abstract

This paper introduces **FIVE** (Framework for Indoor Virtual Environments) and **HIJSON** (Hierarchical Interactive JSON), respectively a web toolkit for indoor mapping applications and a novel cartographic document format. Client-side **FIVE** applications, entirely based on web technologies, rely on **HIJSON** documents produced server-side using **LAR**, a novel representation scheme for topology and geometry.

An *interactive indoor mapping* environment is a virtual reconstruction of a physical indoor space, where the user may interact with virtual objects, experienced in the actual position they occupy in the real world. Our approach outlines a specialized and evoluted 3D *user interface* giving a glimpse of a section of the real world, that the user can handle intuitively. Furthermore, the virtual indoor environment API provides a platform where many different applications can rely upon. Accessible via web browsers from any kind of device, several applications may coexist on this platform. IoT monitoring, realtime multi-person tracking, and cross-storey user navigation, are already implemented using an automatic search for all valid walkable routes, and taking into account both architectural obstacles and furniture.

The **HIJSON** format is used to represent any geometry of the indoor space of complex buildings, capturing their hierarchical structure, a complete representation of their topology, and all the objects (either smart or not) contained inside. Such textual representation allows the **FIVE** framework to offer a web environment in which the user is presented with 2D or 3D models to navigate. With respect to current cartographic formats, **HIJSON** suggests four major enhancements: (a) exposes a hierarchical structure; (b) uses local metric coordinate systems; (c) may import external geometric models; (d) accepts semantic extensions. The semantic extensions supported by the **FIVE** architecture encapsulate the details about communication protocols, rendering style, and exchanged and displayed information, allowing the **HIJSON** format to be extended with any sort of models of objects, sensors or behaviors.

1. Introduction

An *interactive indoor mapping* environment consists of a virtual reconstruction of a physical indoor space, in which the user can move around and interact with virtual objects, that are found in the same position they actually occupy in the real world. Such an interactive indoor mapping can be thought as a specialized and very evoluted *user interface* capable of giving a glimpse of a section of the real world that the user can handle in a natural and intuitive way. Such a reconstructed virtual indoor environment can be considered a general platform where many different applications can rely upon. Both promising and already well explored ICT applications may find in *virtual indoor mapping* the perfect context to be integrated into.

In particular, for environments with massive presence of sensor-equipped (or “smart”) objects, which realize the so-called *IoT* (Internet of Things), the interactive indoor mapping represents an ideal integrated interface for IoT monitoring systems. To be specific, it can be the container of indoor navigation systems, giving the user, to be routed across an indoor environment, the opportunity to interact with objects along the suggested paths. Furthermore, in conjunction with the advancements in the field of user indoor location, whose efforts are nowadays focused to realize an integration of positioning systems like GNSS (Global Navigation Satellite system), Wi-Fi, Bluetooth and LTE (Long Term Evolution), to support continuos outdoor/indoor navigation by means of integration of technologies, it represents the most natural interface to perform realtime access monitoring and multi-person tracking. This kind of evoluted user interface is provided by the **FIVE** Web Framework, whose design choice and implementation details represent the main contribution of this paper.

To enable such an interactive mapping platform it is of the utmost importance to set up a descriptive representation of the indoor environment. A remarkable contribution of this work is also the definition of the **HIJSON** document format, which provides exactly this required descriptive representation. This description belongs to the field of indoor cartography, which as digital evolution of plain floor plans, has arrived to arouse the interest of big players like Google, that has integrated indoor plans of specific locations of in-

108	terest [11] into Google Maps. In general, it can be considered “of interest” — such to justify and motivate indoor cartographic applications — both public or commercial places of vast dimensions, as for example airports, train stations, shopping malls, and also private buildings subject to strict access protocols, like warehouses, logistic centers, data centers, etc.	162
109		163
110		164
111		165
112		166
113		167
114		168
115		169
116	Moreover this paper quickly outlines the generation of geometric data of a complex building, to provide both an explicit semantic and a hierarchical model of indoor spaces. LAR, a general representation for geometric and solid modeling is used for this purpose. The generated LAR structures are exported to HIJSON format, extending GEOJSON for indoor mapping and the Internet-of- Things. A convenient way to extend the representation capabilities of IoT <i>smart objects</i> is also mentioned as semantic extensions, that affects both document format and the web framework, and can be easily collected in a public repository.	170
117		171
118		172
119		173
120		174
121		175
122		176
123		177
124		178
125		179
126		180
127	This work, jointly developed by Sogei S.p.A., an ICT company fully owned by Italian Ministry of Economy and Finance, and the CVDLAB (Computational Visual Design Laboratory) of the “Roma Tre” University, is inspired by the necessities of Sogei itself, which runs one of the largest data center of Europe, so requiring very strict access control policies, which include the recording and the real-time interaction with man/machine maintenance scenarios. Support for this interactive framework, where realtime awareness of the maintainer position inside the data center helps to reduce intervention times and to increase safety and security, has been chosen as case study of interactive indoor mapping, based on the proposed indoor cartographic format.	181
128		182
129		183
130		184
131		185
132		186
133		187
134		188
135		189
136		190
137		191
138		192
139		193
140		194
141		195
142		196
143		197
144		198
145		199
146		200
147		201
148		202
149		203
150		204
151		205
152		206
153	The remainder of this document is organized as follows. In Section 2 we provide an overview of the state of the art in the field of indoor document standards and related applications. Section 3 is devoted to present the FIVE Web Framework, focusing on its architecture and supported applications, while Section 4 introduces the HIJSON format specifically defined to describe for indoor environments. Section 5 reports about the software toolkit developed to handle the new document format. In Section 6 it is depicted the operative workflow adopted to realize the mapping of an hospital. Finally, Section 7 proposes some conclusive remarks and future developments.	207
154		208
155	<h2>2. Related work</h2>	209
156	The virtual indoor mapping is a quite young field of research resulting from a mix of interdisciplinary knowledge. Thus significative works are not known to the authors at the moment. Counterwise, research on the cartographic representation of indoor environments is extensive and heterogeneous with respect to the strategies applied. Different information sources are used, and accuracy of the produced so-	210
157		211
158		212
159		213
160		214
161		215

216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269

ment mapping. IndoorJSON is compliant with GeoJSON syntax, supporting all GeoJSON geometry types.

Customization with respect to GeoJSON format is obtained exploiting particular properties to correctly define the indoor elements: `level` (which describes which storey contains the feature) and `geomType` (which identifies the category of the object). A number of non-mandatory indoor properties is also defined: `accessible` (which describes if an element is walkable or not), `connector` (which defines if the element is a connection between two storeys) `direction` (which describes the direction of the connection: both ways, only up, only down).

A syntax validator is provided by *indoor.io*, but the commercial nature of this project limits the number of tools available to deal with this format.

3. FIVE Web Framework

234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321

FIVE, acronym of *Framework for Indoor mapping in Virtual Environments*, provides a customizable, and scalable web framework realizing an virtual indoor mapping platform in which multiple applications can convenient resides on at the same time: IoT, monitoring, realtime multi-person tracking and cross-storey user navigation.

Expandability and customizability derive from both design choices and HIJSON format inherent characteristics, i.e. the possibility of semantic extensions (see 4). Scalability is directly borrowed from technologies used for software development: *JavaScript* language, using *Node.js*, in particular *Express.js* as backend framework, exploiting the power of *WebSocket* protocol through the *Socket.io* library.

Being supported by the *web-as-a-platform*, the framework exposes also an high availability: it is so simple to use as to visit a website, both from desktop or mobile devices, without explicit requirements to install any software package from proprietary stores—access to which is often denied from business devices.

The FIVE Web Framework deeply relies on HIJSON Toolkit (see 5) and offers an all-inclusive client/server architecture with a convenient and highly interactive user interface, leaving aside the specific indoor positioning system and the IoT sensors to deal with. A robust application interface is provided and described in the following section.

3.1. Applications

The Framework has been designed with focus on two different kind of users: the *Explorer* and the *Supervisor*. They have different requirements and are likely equipped with different devices: while the *Supervisor* monitors the indoor environment through a desktop workstation, the *Explorer* has a smartphone available and needs to be routed across the building.

In both cases, the web platform ensures a perfect alignment with the BYOD (Bring Your Own Device) approach,

nowadays often supported by companies that encourage employees to use personal devices.

3.1.1 IoT monitoring

An *IoT monitoring application* consists of an interface showing to the user, in a single, integrated and centralized way, the information collected from all the smart objects modelled in the HIJSON document. IoT monitoring application provides bidirectional communication, since the interface let the user receive information coming from smart objects while allowing him to send commands to them.

As the name itself may suggest, it is an activity specifically performed by a *Supervisor* user, but it can be also suitable to be deployed for the *Explorer* user, since she can take advantage of the interactive information coming from the surroundings objects while she moves across the indoor environment.

Monitoring different smart objects may require different ways to visualize and/or send data and commands. Modularity and extendibility of the application respond superbly to these requirements, by providing for each class of objects a different interface of visualization and interaction, as a result of the polymorphism principles introduced by the HIJSON Class. In particular, the user interface is characterized by a dual-display mode, that allows the user to see at the same time a 2D map that gives an overall glance in a simplified plan, and a 3D virtual environment to navigate into, as shown in Figure 1.

Alongside with typical smart objects, suitable to deal with like thermostats, where the user can read the room temperature and turn the heating on/off, other kinds of objects, that are not properly considered “smart”, can be integrated into the FIVE environment. It is the case, for example, of fire extinguishers, that are able to show the date of their last check, stored in a database.

3.1.2 Realtime multi-person tracking

Realtime multi-person tracking allows a *Supervisor* to monitor the current real position of people inside the building. This kind of task can be useful for several reasons, including security, logistics or to supervise composite operative workflows. Each device equipped with the *Explorer* application is in charge of locating itself, interacting with the indoor positioning system, and notifying the current position in continuos mode. Evidence of the people position is given to the *Supervisor* both into a 2D map and an immersive 3D virtual environment (see Figure 1).

3.1.3 Cross-storey user navigation

The FIVE Framework also provides the capability to give directions to *Explorer* users that must move across the in-

324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377

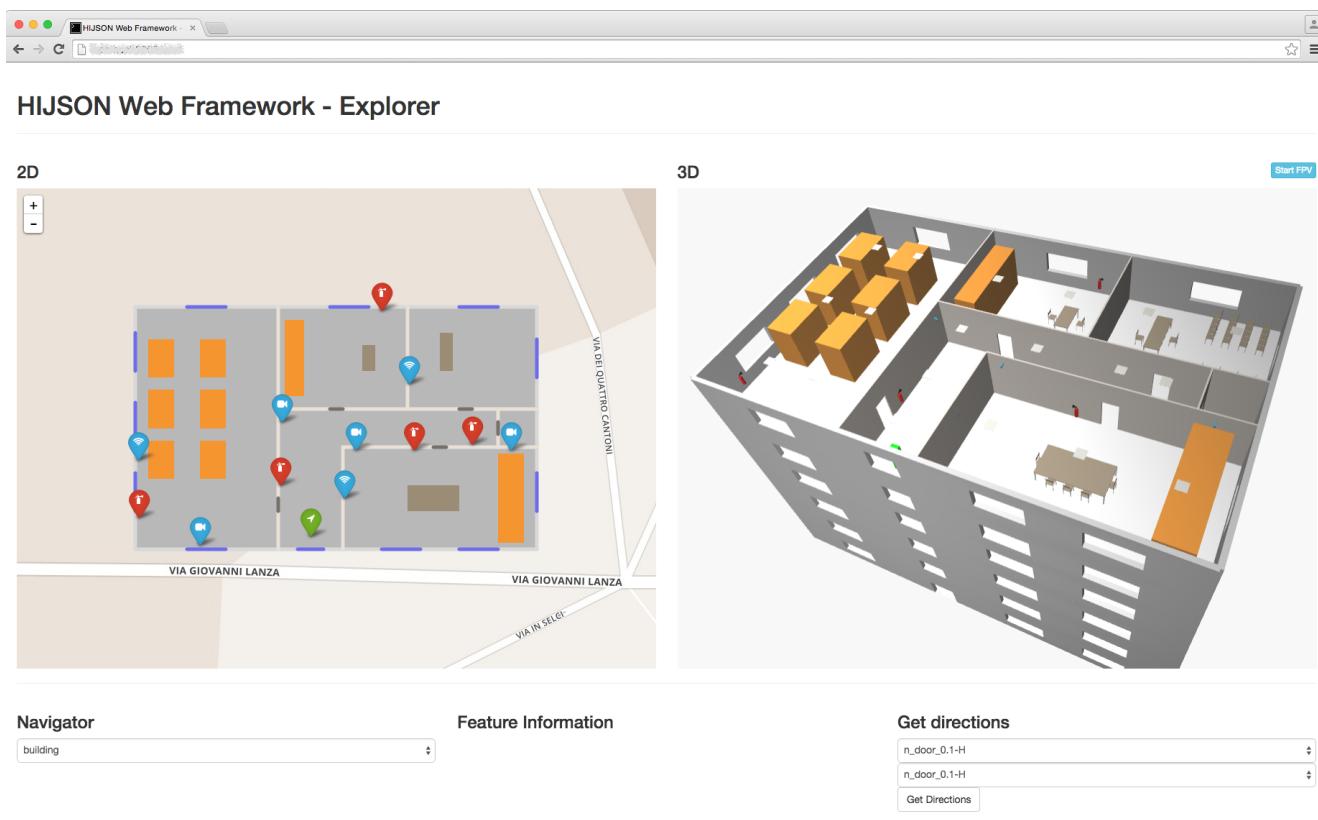


Figure 1: FIVE Web Framework UI

door environment. The user specifies a starting and an ending point and the system provides him with a valid connection path. This feature strongly rely on the graph of paths generated by the Toolkit, so starting and ending points must be nodes of the graph. *Connection nodes* are introduced to represent stairs or elevators, enabling cross-storey paths to be computed. Since paths can span more than one storey, the most effective way to display them to the user is to show the connection nodes visualized in one or more 2D maps.

3.2. Architecture

Like the vast majority of the web based applications, the Framework exposes an overall architecture that is inherently *client/server*. In particular, two different types of possible clients are identifiable, one for each different kind of users: the *Supervisor* client and the *Explorer* client. Both of them connect to the same server.

The indoor space described by the input HIJSON document is processed by the server via the processing pipeline. After that, any connecting *Explorer* client, presumably via a mobile device, will be provided with the information to perform cross-storey navigation of the building, while re-

porting the user position to the server. The server will feed any connecting *Supervisor* client with users positions, along with data from sensor-equipped objects present in the environment, achieving both IoT monitoring and realtime multi-person tracking.

3.2.1 Server Architecture

An architectural scheme of the framework is provided in Figure 2. A web server module is responsible for listening to connecting clients. Each client connection is handled by the web server module providing all the required resources and then by opening a WebSocket channel, in order to have both *Explorer* and/or *Supervisor* communication protocol data flow within. In particular, the multi-person tracking module receives position data from *Explorer* clients. It aggregates and sends these information to connected *Supervisor* clients through the WebSocket channel, using a simple but reliable protocol described later. Independence from particular IoT sensor equipment communication protocols is achieved introducing a smart object proxy module. This one is defined in the HI-

378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431

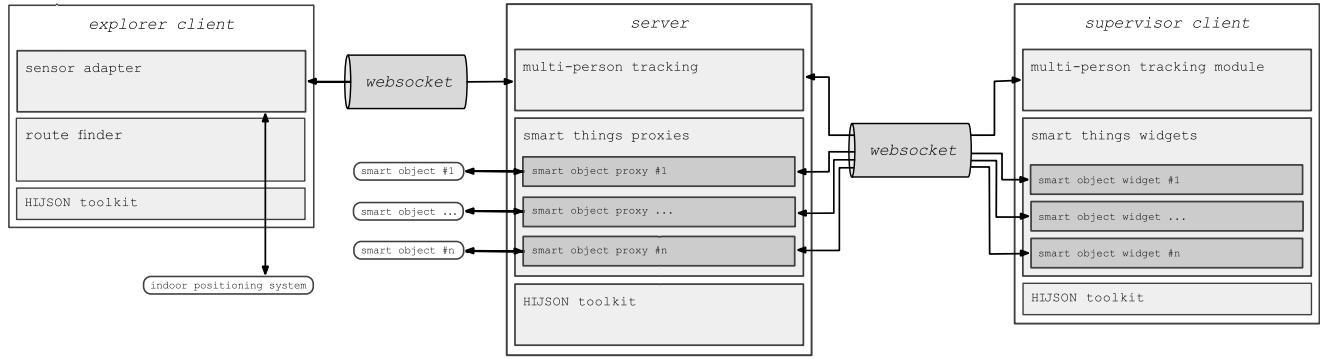
432
433
434
435
436
437
438
439
440
441
442
443

Figure 2: FIVE Web Framework architecture

444

445
446
447 JSON Class and is obtained via the `getProxy()` method
448 for each smart object modelled as previously described.
449450
451

3.2.2 Explorer client architecture

452
453 The *Explorer* client architecture is generally deployed on
454 a mobile device, which is usually supplied to a user who
455 needs to be routed across the indoor mapped environment.
456 The *sensor adapter* module encapsulates the commu-
457 nication logic with the indoor positioning system. The pres-
458 ence of this module ensures independence from particular
459 technologies, so allowing the *Explorer* client to rely on dif-
460 ferent indoor positioning systems (Wi-Fi, Bluetooth, LTE,
461 etc.).462 Every time a *sensor adapter* observes a perceptible
463 modification in user position, it sends the new position
464 information to the server through the single opened Web-
465 Socket, spawning a simple message with the syntax de-
466 scribed in listing 1.467
468

```
currentPosition = {
    coordinates: [x, y],
    levelId: level-ID
}
```

469
470
471 Listing 1: Example of message sent by the *Explorer* client
472 to the server.
473474
475 Relevant information includes, beside current coor-
476 dinates, the indication of the storey of the possibly multilevel
477 building the user is in.478 It is to remark that, when not outflanked by the introduc-
479 tion of an external server, the problem of communication
480 between positioning system and the low level APIs of the
481 browser is left to positioning system itself or to whom is
482 in charge of specific deployments of the FIVE Web Frame-
483 work. The *smart object widget* module, being in
484 common with the *Supervisor* client, will be discussed in the
485 next section.486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539

3.2.3 Supervisor client architecture

The architecture of *Supervisor* client includes two modules. The first one, named *multi-person tracking module*, is responsible to receive through the WebSocket, from the server information about *explorers* of the environment, showing them in the user interface. The second module, named *smart object widget*, communicates with the server to propose the user realtime information about sensor-equipped objects in the environment. Data passes through the single WebSocket opened between the server and every *Supervisor* client. Relying on a naive but effective communication protocol, each *smart object* widget exchanges data only with its corresponding *smart object proxy* on the server. To ensure the data are posted only when the user requires the information relative to a specific *smart object*, a *widget lifecycle protocol* is implemented. This one is based on 4 event triggers `on_before_show`, `on_show`, `on_before_hide`, `on_hide`, as suggested by their names. When the user requires information about a *smart object*, its *widget* has to be rendered, and `on_before_show` the server is notified to connect via relative proxy to the sensor. Once connected, the server begin to send data via WebSocket. Received data are shown through the *widget* to the user. When done, the `on_before_hide` event of the *widget* is triggered, a notification is sent to the server announcing to stop sending data, and the proxy closes the connection to the sensor. *Widget lifecycle protocol* ensures that only required data are sent from the server to the client.

4. The HIJSON format

To support the interactive indoor mapping platform, a novel format of cartographic documents has been defined: it has been named *HIJSON* (Hierarchical Indoor JSON). It is based upon ideas and design principles collected from previous formats and identifies four critical improvements with respect to them: it exposes a *hierarchical structure*,

540
 541 uses *metric local coordinate system*, may import *external*
 542 *geometric models* and accepts *semantic extensions*. Further-
 543 more, geometrical and topological data can be conveniently
 544 imported and represented via LAR (see A.1), an advanced
 545 representation scheme, allowing to deal with *hyperlinked*
 546 *geometric models*.

547
Hierarchical structure The HIJSON format allows for
 548 hierarchical description of indoor spaces. The introduction
 549 of a hierarchical structure establishes a parent-child relation
 550 between entities of the model, reflecting a container-
 551 contained relationship. This directly implies a neater rep-
 552 resentation than the plain linear structure adopted by Geo-
 553 JSON, being a perfect analogy of objects contained (i.e.
 554 placed) into spaces.

555 Therefore, an organized arrangement of spaces is al-
 556 lowed by HIJSON, via logical (or even physical) grouping:
 557 concepts like building wings, sections, storeys, depart-
 558 ments, etc. can be directly introduced, in order to reflect
 559 into the document structure the actual logical or physical
 560 divisions, categories or relationships among the modelled
 561 spaces.

562 Hierarchical structures are common in computer graph-
 563 ics, where they are used as scene graphs. This accordance
 564 of underlying structures really simplifies 3D rendering algo-
 565 rithms of HIJSON documented environments. Furthermore,
 566 the container-contained relation enables a recurring use of
 567 local reference frames.

568
Metric local coordinate system Supported by the hierar-
 569 chical underlying structure, the HIJSON document format
 570 allows the use of local coordinate systems. Hence the shape
 571 of all elements can be conveniently modelled using local co-
 572 ordinates, and then placed in the right position with respect
 573 to the position of the parent (or container) element applying
 574 a rotation, followed by a translation transformation.

575 Another substantial advantage is represented by the
 576 adoption of a metric reference frame, consequently simpli-
 577 fying the compilation of the document, either manually gen-
 578 erated or produced by software tools. Just remember that
 579 the GeoJSON coordinates are geographical, a pairs of (ab-
 580 solute) latitude and longitude angles, like the ones provided
 581 by GNSS systems. This kind of coordinates are certainly
 582 not particularly user friendly, when positioning a smart de-
 583 vice or a furniture element within a specific building room.

584 The HIJSON document format is specially designed to
 585 guarantee the user to be routed seamlessly from outdoor to
 586 indoor and vice versa. Even if indoor geometries are entered
 587 in a local metric coordinate system, continuos outdoor/in-
 588 door navigation is ensured through the processing pipeline
 589 detailed below.

590 Semantic extensions Semantic extensions make the HI-
 591 JSON format extendible and customizable, that is able to
 592 adequately respond to any need of objects representation.
 593 To define a semantic extension means to allow the HIJSON
 594 document to model an object previously not covered, or
 595 even to modify the behavior of a comprised one. Semantic
 596 extensions are to be defined both as HIJSON format syntax
 597 and as HIJSON Toolkit source code. In particular it is nec-
 598 essary to define respectively a new HIJSON Element and a
 599 new HIJSON Class, as specified below.

4.1. Structure and syntax

A HIJSON document is composed by a configuration section, followed by one or more FeatureCollections, containing the actual data.

Listing 2 shows a simplified HIJSON document, devoid of punctual details, to make clear to the reader the overall document structure.

```
{
  "config": {
    // ...
  },
  "data": [
    // ...
    {
      "id": "architecture",
      "type": "FeatureCollection",
      "features": [
        // ...
      ],
      "id": "furniture_1",
      "type": "FeatureCollection",
      "features": [
        // ...
      ],
      "id": "furniture_2",
      "type": "FeatureCollection",
      "features": [
        // ...
      ]
    }
  ]
}
```

Listing 2: Example of HIJSON document.

The configuration includes parameters and settings needed for building representation in the form of a JSON Object. One of the core information in this section is defined by the correspondence between three points of the local coordinate system and three points of the real world, expressed in geographical coordinates. This is needed to ensure a seamlessly passage from local to geographical coordinate system and vice versa.

After the configuration part, the document includes a list of FeatureCollection. An example of FeatureCollection is given in listing 3.

```

648
649 {
650   "id": "architecture",
651   "type": "FeatureCollection",
652   "features": [
653     // ...
654     {
655       "type": "Feature",
656       "id": "room_0.1",
657       "geometry": {
658         "type": "Polygon",
659         "coordinates": [
660           [ [0, 0], [11, 0], [11, 19], [0,
661             19] ]
662         ],
663         "properties": {
664           "class": "room",
665           "parent": "level_0",
666           "description": "Office of Mr. Smith",
667           "tVector": [10, 20, 0],
668           "rVector": [0, 0, 90]
669         }
670       },
671     // ...
672   ]
673 }
```

Listing 3: Example of FeatureCollection.

Each element of the list is given in the form of a GeoJSON FeatureCollection, that contains an arbitrary number of HIJSON Elements. Each FeatureCollection imposes a logical relationship that can be used to group together related HIJSON Elements. Since HIJSON Elements adhere to the GeoJSON format, each FeatureCollection results compliant with GeoJSON syntax and then accepted by any GeoJSON validator. As detailed below, the HIJSON format introduces some additional rules that allow the adoption of this format for indoor representation.

4.1.1 HIJSON Element

Dealing with indoor environments, there are essentially two classes of objects that is necessary to represent. They are (a) architectural elements, like a room, a corridor, a wall, etc. and (b) furnishings, intended in a broad sense, such as to contain both furniture, like a desk or a chair, and/or “smart objects”, like an IP-cam or a connected thermostat.

A HIJSON Element defines a GeoJSON compliant syntax to describe both geometry and properties of an object. It represents the atomic component of a HIJSON document. It would be a best practice to group together related JSON Element using FeatureCollections: several classification strategies can be applied, for example by grouping the elements by storey or even by room. Alternatively, since the furnishings are more likely to change

than the architectural components of a building, these two different kinds of elements can be isolated in different FeatureCollections.

The hierarchical structure of the document gives visible form to the capability of HIJSON Elements to have children elements. A unique ID is mandatory for every HIJSON Element.

Three Geometry types can be used here: Point, LineString and Polygon. The choice of the Geometry type to be associated to a HIJSON Element implicitly defines the category of the element: Point is used for furnishings, LineString for walls and doors, while Polygon may describe levels and rooms.

The Geometry coordinates are expressed in metres, by convention starting at the bottom-left corner of the element, whose position is used to set-up the origin of a local coordinate frame. Unlike GeoJSON, where all properties are optional, in HIJSON some strict requirements are imposed, and some attributes are mandatories: a) class (representing the element category, used to instantiate the appropriate *HIJSON Class*), b) parent (containing the ID of the parent of the element), c) d) tVector (representing the translation relative to the parent element, expressed in metres), e) rVector (representing the rotation relative to the parent element, expressed in nonagesimal degrees).

Specific classes may require the mandatory presence of other properties. For example, the classes internal_wall and external_wall that define the internal partitions and the external envelope, respectively, require a connections array, containing the IDs of the adjacent elements. This information is used by the connector children of the element (e.g. doors) to identify the areas linked together.

Given the nature of the GeoJSON format from which HIJSON derives, the elements are represented by their 2D shape, like on a planimetry. The property height was introduced to assign a value to the height of the object, intended as a third dimension.

A description property can provide further information about the element. Arbitrary optional fields can be added without restrictions, in order to enrich and extend the expressivity of the representation, or simply for the sake of documentation.

5. HIJSON Toolkit

The HIJSON Toolkit is a software module that implements common operations and transformations on HIJSON documents. Written in *JavaScript* language, this software module has been built to be deployed in the web environment. It is *modular* and entirely *isomorphic*, i.e. can run on the server as well as on every client. Working in the web environment, the Toolkit benefits of the “fertility” commonly concerning the software development in this field:

for example, it takes advantage of libraries and frameworks such as *React*, “the JavaScript library for building user interfaces” by Facebook, and as *Three.js*, the current de-facto standard to deal with *WebGL* technologies.

The Toolkit executes the instantiation and extension logic of a HIJSON document, and provides a multistage transformation pipeline that, according to the requirements, can be used either entirely or only in part.

5.1. Processing pipeline

The HIJSON processing pipeline implements the sequence of preliminary transformations that have to be applied to a HIJSON document before any further operation. It is not strictly required to complete each stage of the pipeline: the exit stage depends on the specific use case.

The application of the transformation pipeline has a double aim. The first one consists in generating the graph of valid paths among all the interesting elements. The second objective is the generation of one *GeoJSON* document for each storey of the building described by the HIJSON document. In this way a bidimensional layout can be provided for every level of the building, and visualized through any compliant *GeoJSON* viewer.

The HIJSON processing pipeline is composed by six elaboration stages, denoted as *validation*, *georeferencing*, *parsing*, *graph paths generation*, *2D layers generation*, *marshalling*. The pipeline of transformations and the output of each stage are shown in Figure 3.

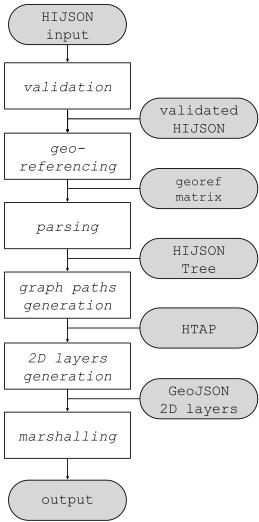


Figure 3: HIJSON processing pipeline

1. *validation* - The first one is the validation stage. In order to begin with the effective transformations the input HIJSON document must be compliant with both the syntax format and the structural requirements. In

the case the validation stage fails, processing aborts and does not continue to following stages; instead if this stage successes, then the output for the next stage is a validated HIJSON.

2. *georeferencing* - In the second stage, in order to allow for continuous outdoor/indoor navigation, the system needs to compute the georeferencing matrix, a linear operator able to transform local coordinates into global coordinates (world coordinate system — latitude and longitude angles) and vice versa. This task is accomplished by solving a linear system obtained from information contained in HIJSON configuration part and precisely from the correspondence of three real world points to three points included into the HIJSON document.
3. *parsing* - The parsing stage takes the validated and georeferenced HIJSON as its input, that as illustrated before can be thought of as a list of HIJSON Elements, parses them and produces an instance of the HIJSON Tree. The HIJSON Tree is an object in memory representing the hierarchical structure of the building described by the HIJSON document.
4. *graph of paths generation* - The fourth stage is in charge of the generation of the graph of paths. The algorithm to achieve such a goal is introduced in Section 5.1.1. The graph of paths can be used to compute valid directions between pairs of points of interest inside the building model. Once the graph of paths has been computed, the input HIJSON Tree is augmented with paths information, becoming what has been called an HTAP (HIJSON Tree Augmented with Paths). Augmentation always takes place in the form of an addition of leaf nodes as children of a specific element (e.g. “room”).
5. *2D layers generation* - The fifth stage concerns the generation of *GeoJSON layers*. For each storey of the building, the Toolkit generates a *GeoJSON* layer that can be used for the creation of a 2D map. Each layer contains only the children of a ‘level’ node of the HIJSON Tree. The presence of a specific element inside the layer can be finely tuned by means of a Boolean value. The geographical coordinates of every elements are calculated by a series of multiplications between transformation matrices obtained during the tree traversal to the local coordinates.
6. *marshalling* - The last stage is responsible for executing a serialization of the the transformed data. This stage, in which are performed tasks like breaking dependency-loops and stringification, is mainly useful server-side, as the output is there stored ready to be served to any requiring client.

810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863

864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917

5.1.1 Automatic generation of valid paths

The fourth stage of the processing pipeline is responsible for the generation of a graph of valid paths through the entire model represented by the input HIJSON document. The graph generated according to the algorithm described in the following, although non optimal, ensures a complete coverage of the surface while limiting the number of generated nodes. The resulting graph is weighted on the edges with nodes distances. Each graph node may represent either:

- a. a *standard path node*, i.e. a junction node or possibly an endpoint of a path;
- b. a *connection node*, used as subproblem composing element in the divide et impera approach adopted;
- c. an *element node* i.e. HIJSON Element (whose HIJSON Class explicitly grants his presence in the graph), typically an endpoint of a path.

The graph of paths allows for calculations of directions between any two given nodes. Although different approaches have been explored [3], a very classical solution has been selected in this case, so directions are actually computed client-side by applying the Dijkstra algorithm to the graph.

Taking advantage of the hierarchical structure of the HIJSON document, and according to the divide et impera approach, the problem of paths generation is split in several sub-problems, which consist in the computation of the sub-graphs relative to each individual space, more generally a single room. The sub-graphs are then linked together through the connection nodes (which in most cases represent doors). The resolution of each sub-problem (as depicted in Figure 4), is composed by four steps, as detailed below.

1. Computation of the walkable area of the space: this task is accomplished by subtracting the shape of the obstacles from the area of the space; the result is typically a surface with holes.
2. Triangulation of the walkable area: the computed surface is triangulated taking into account the presence of holes.
3. Identification of graph nodes: for each triangle side completely internal to the area, its midpoint is selected as standard path node.
4. Junction of nodes: nodes relative to the same triangle are then linked pairwise; both element nodes and connection nodes (i.e. doors) are linked to the nearest node of the space (i.e. room).

5.2. HIJSON Class definition

To make better use of the possibilities offered by the HIJSON Toolkit and by the HIJSON document format, some

custom dynamic behaviors can be described. These behaviors encapsulate the specificities relative to communication protocols with the sensors, as well as to features of user interaction. The interface for such behavior is the HIJSON Class.



Figure 5: HIJSON Element/Class/Node relationship

Every HIJSON Element of the input HIJSON document has a dynamic counterpart, a running instance called *HIJSON Node*, instantiated according to the corresponding HIJSON Class via reflection methods (see Figure 5).

To specify a new *HIJSON Class* means to extend the Toolkit to deal with a new class of HIJSON Element. To extend the toolkit in order to deal with a new class of HIJSON Element is required to specify a new HIJSON Class, by defining the following properties and methods:

- `in_graph`: a Boolean value to express if the element is an approachable point in the graph of paths;
- `in_2D_map`: a Boolean value to express if the element must be shown in the 2D map;
- `get2DStyle ()`: a method that returns the 2D map appearance of the element, essentially HTML and CSS code;
- `get3DModel ()`: a method that returns the 3D model appearance of the element, i.e. an instance of `Object3D` of the *THREE.js* framework;
- `getWidget ()`: a method that returns the information widget, a *React* component;
- `getProxy ()`: a method that returns the server-side proxy which encapsulate the IoT sensor communication protocol, i.e. a *Node.js* module.

User's needs for new indoor elements, greatly different sensor equipments, alternative representations of 2D or 3D viewports are accepted by the definition of new HIJSON Classes, that so provide single-point custom extensions of the Toolkit capabilities.

6. Virtual indoor mapping generation workflow

As case study of the discussed approach, we have taken into account the need of Sogei S.p.A. to support its maintenance service workflow, since its data center, one of biggest data centers in Europe, is subject to very strict access control policies.

918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971

972
973
974
975
976
977
978
979

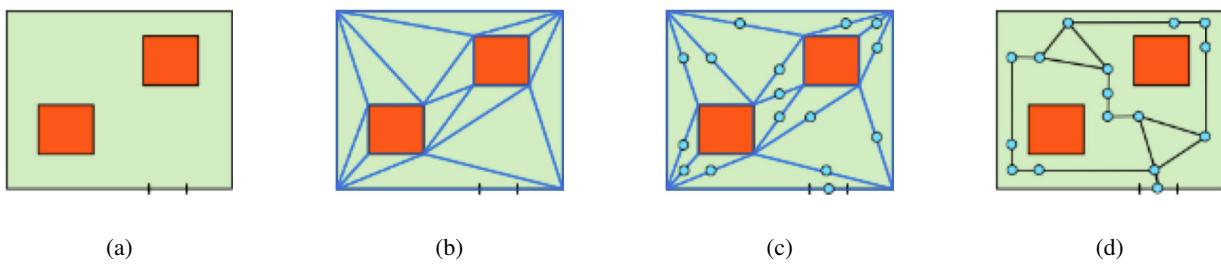


Figure 4: graph of paths generation: (a) detection of obstacles and computation of walkable area; (b) triangulation of walkable area; (c) identification of graph nodes; (d) junction of nodes.

984
985
986
987
988
989
990
991
992
993
994
995

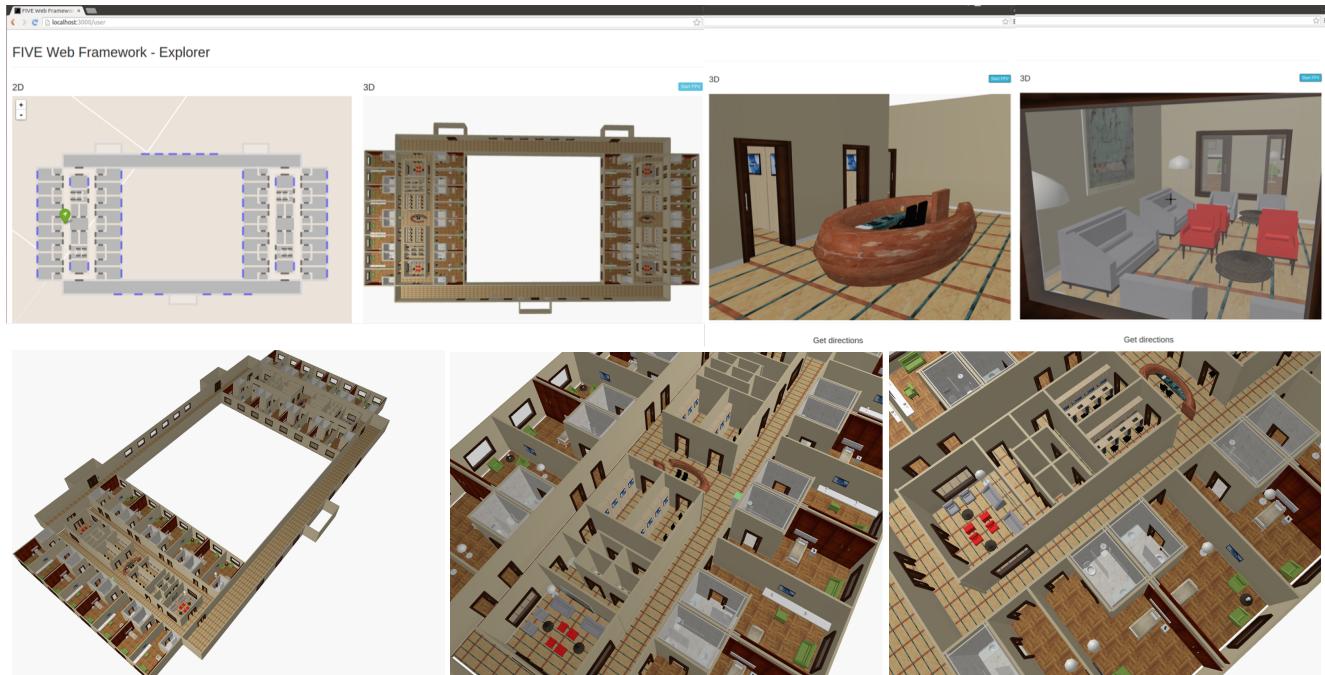


Figure 6: example caption

The overall state of the data center can be monitored through the *Supervisor* client. In this case the considered smart objects belong to a range of different devices, going from webcams, that provide on request the captured video streams, by way of alarm and antifire systems, till to individual servers, that can be monitored along several dimensions, including operating temperature, workload, etc.

1017 The most common maintenance scenario consists of an
1018 intervention by a technician that have to move across the
1019 environment, and locate within a huge data center the ma-
1020 chine on which to operate, a not trivial task due to the pres-
1021 ence of thousands of similar-looking machine racks. Thus
1022 the operator will be equipped with an *Explorer* client, which
1023 will drive him to the target machine on which operate, while
1024 continuously notifying to a security *Supervisor* his position,
1025 obtained by interacting with the indoor positioning system.

The maintenance workflow supervisor, using the *Supervisor* client, is able to monitor the operator position within the data center, verifying that he does not deviate on unauthorized paths, triggering some console alarm if this happens.

The purpose is to support the process of those in charge of carrying out the “ticket-maintenance” activities, as quickly as possible and without error. The “maintenance man” will be guided to the right sub-system among thousands of racks. The real-time awareness of the relative positions between the “maintenance man” and the rack — containing the sub-system — will help to reduce intervention times and to increase safety. By knowing when the maintenance process starts, the system can automatically move, in real time, services, which are hosted on virtual machines, to other systems, thus maintaining the continuity of services and, at the same time, reducing the global risk factor. When

1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079



Figure 7: example caption

the “ticket-maintenance” is over, and the technician goes away, it is possible to immediately restore the pre-existing conditions of services after a complete test has been performed.

7. Conclusions

In this paper a novel document format, named HIJSON, for indoor cartographical descriptions has been introduced. Utilization of local metric coordinate system, avoiding the manipulation of global geographical coordinates, really inconvenient when dealing with indoor spaces and objects, greatly enhances the modeling and rendering of the document content. Currently, we produce the HIJSON document from a python script using two libraries for geometric computing (`pyplasm` and `larcc` [7, 14, 6]). The modeling process can be further improved by implementing a LAR-based graphical editor to assist the user during the description of the indoor space. The realization of such an editor is already in our plans.

The HIJSON format focuses on a hierarchical representation of the indoor spaces that allows for completely capturing their topology. On the basis of this representation a virtual web environment can be rebuilt working as a unifying platform to run a bunch of different applications. The reference architecture of such a platform has been also implemented and described in this work.

The architecture supports a whole range of applications: IoT monitoring, realtime multi-person tracking and user cross-storey navigation are already implemented and de-

scribed. A very convenient way to extend the representation capabilities of smart objects is also mentioned as semantic extensions. These extensions, which affects both document format and its web framework, might be easily collected in a public repository. Community could both use public available extensions or contribute by mapping new (smart) objects inside the HIJSON document format.

Acknowledgments The authors acknowledge the inspiring cooperation on LAR from Antonio DiCarlo and Vadim Shapiro. Thanks are extended to SOGEI, the ICT company of the Italian Ministry of Economy and Finance, for the support provided through several grants. Giulia Clementi and Marco Grani have developed respectively the virtual environment of the ward department and the viral laboratory of the general hospital model.

References

- [1] B. Al Delail, L. Weruaga, M. Zemerly, and J. Ng. Indoor localization and navigation using smartphones augmented reality and inertial tracking. In *Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on*, pages 929–932, Dec 2013. [2](#)
- [2] T. Basak. Combinatorial cell complexes and Poincaré duality. *Geometriae Dedicata*, 147(1):357–387, 2010.
- [3] W. Bian, Y. Guo, and Q. Qiu. Research on personalized indoor routing algorithm. In *Distributed Computing and Applications to Business, Engineering and Science (DCABES), 2014 13th International Symposium on*, pages 275–277, Nov 2014. [9](#)

- 1188 [4] M. Boysen, C. De Haas, H. Lu, X. Xie, and A. Pilvinyte.
 1189 Constructing indoor navigation systems from digital building
 1190 information. In *Data Engineering (ICDE), 2014 IEEE*
 1191 *30th International Conference on*, pages 1194–1197, March
 1192 2014. 2
- 1193 [5] A. Buluç and J. R. Gilbert. Parallel sparse matrix-matrix
 1194 multiplication and indexing: Implementation and exper-
 1195 iments. *SIAM Journal of Scientific Computing (SISC)*,
 1196 34(4):170 – 191, 2012.
- 1197 [6] A. DiCarlo, M. Jirik, and A. Paoluzzi. Cad models from
 1198 medical images using the linear algebraic representation. In
 1199 *CAD'15*, London, UK, June 22-25 2015. Accepted for pub-
 1200 lication in Computer-Aided Design and Applications, Taylor
 1201 & Francis. 11
- 1202 [7] A. Dicarlo, A. Paoluzzi, and V. Shapiro. 11
- 1203 [8] C. Eastman, P. Teicholz, R. Sacks, and K. Liston. *BIM Hand-
 1204 book: A Guide to Building Information Modeling for Own-
 1205 ers, Managers, Designers, Engineers and Contractors*. Wi-
 1206 ley Publishing, 2008. 2
- 1207 [9] L. Faramondi, F. Inderst, S. Panzieri, and F. Pascucci. Hy-
 1208 brid map building for personal indoor navigation systems. In
 1209 *Advanced Intelligent Mechatronics (AIM), 2014 IEEE/ASME*
 1210 *International Conference on*, pages 646–651, July 2014. 2
- 1211 [10] GeoJSON contributors. Geojson. <http://geojson.org/>, 2015. Accessed: 2015-03-23. 2
- 1212 [11] Google, Inc. Go inside with Indoor Maps. <https://www.google.com/maps/about/partners/indoormaps>, 2014. Accessed: 2015-03-23. 2
- 1213 [12] D. Gotlib, M. Gnat, and J. Marciniak. The research on carto-
 1214 graphical indoor presentation and indoor route modeling for
 1215 navigation applications. In *Indoor Positioning and Indoor*
 1216 *Navigation (IPIN), 2012 International Conference on*, pages
 1217 1–7, Nov 2012. 2
- 1218 [13] indoor.io. Indoorjson specification and validator. <https://github.com/asaarinen/indoor-json>, 2013.
 1219 Accessed: 2015-03-23. 2
- 1220 [14] A. Paoluzzi, E. Marino, and F. Spini. LAR-ABC, a repre-
 1221 sentation of architectural geometry: From concept of spaces,
 1222 to design of building fabric, to construction simulation. In
 1223 Ph.Block, J.Knippers, W.Wang, and N.Mitra, editors, *Ad-*
 1224 *vances in Architectural Geometry*, LNCS (Lecture Notes in
 1225 Computer Science). Springer, 2014. To appear. 11
- 1226 [15] A. G. Requicha. Representations for rigid solids: Theory,
 1227 methods, and systems. *ACM Comput. Surv.*, 12(4):437–464,
 1228 Dec. 1980.

1231 A. Appendix

1232 A.1. Short summary of the LAR scheme

```

1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312 V = [[5.,0.],[7.,1.],[9.,0.],[13.,2.],[15.,4.],[17.,8.],[14.,9.],[13.,10.],[11.,11.],[9.,10.],[5.,9.],[7.,
1313 9.],[3.,8.],[0.,6.],[2.,3.],[2.,1.],[8.,3.],[10.,2.],[13.,4.],[14.,6.],[13.,7.],[12.,10.],[11.,9.],[9.,7.],
1314 [7.,7.],[4.,7.],[2.,6.],[3.,5.],[4.,2.],[6.,3.],[11.,4.],[12.,6.],[12.,7.],[10.,5.],[8.,5.],[7.,6.],[5.,5.]]
1315 FV = [[0,1,16,28,29],[0,15,28],[1,2,17],[1,16,17,33],[2,3,17],[3,4,18,19],[3,17,18,30],[4,5,19],[5,6,19],
1316 [6,7,20,21,22,32],[6,19,20],[7,8,21],[8,9,21,22],[9,11,23,24],[9,22,23],[10,11,24,25],[10,12,25],
1317 [12,13,25],[13,14,27],[13,26,27],[14,15,28],[14,27,28,29,36],[16,29,34],[16,33,34],[17,30,33],[18,19,31],[18,30,
1318 [31],[19,20,31,32],[22,23,32,33],[23,24,34,35],[23,33,34],[24,25,27,36],[24,35,36],[25,26,27],[29,34,35],[29,
1319 35,36],[30,31,32,33]]

```

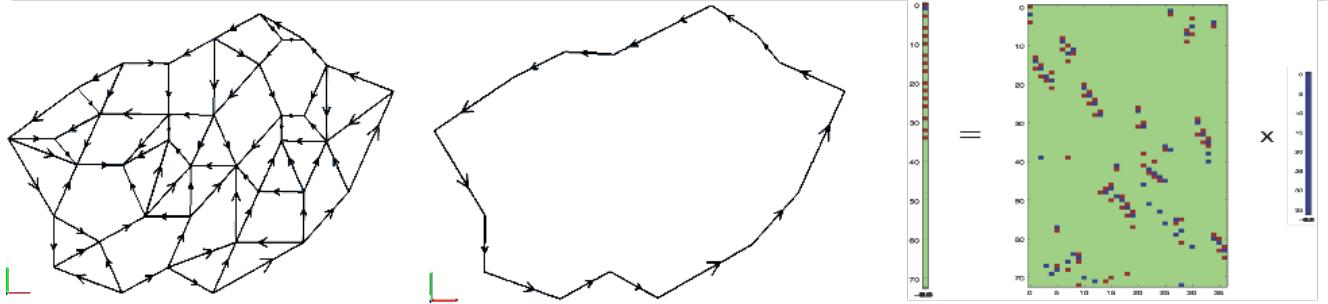


Figure 8: A toy example of the LAR scheme: (a) the bare minimum of data with *complete* information about topology; (b) the extracted boundary; (c) the extraction method $[e] = [\partial][f]$ giving the coordinate representation (in the discrete basis of the 1-cells) of the boundary edges $[e]$ by product of the sparse boundary operator matrix $[\partial]$ times the coordinate representation $[f]$ of the 2-cells.

```

1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403

```