Routing people and vehicles inside buildings

Rui Casimiro, Renato Nunes
rui.filipe@ist.utl.pt, renato.nunes@ist.utl.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2016

Abstract

Most of the indoor routing systems found in our research lack the means to add new maps to the system and are not generic enough to fit all types of indoor spaces. In order to fill this gap, we propose an open and generic indoor routing system, in order to ease the addition of new maps and the disclosure of the system. The result is an Android smartphone application capable of answering the most challenging route searches. This work describes the application in great detail.The main application feature is generating instructions for the route returned by the route search between a start point and an end point, taking into account possible restrictions the user might choose to avoid such as stairs and elevators. This instructions help the user take the turns and also include reference points witch further enrich the instructions. A 2D map of the building is also shown. The application also includes other features like a point of interest directory, park near a target destination and providing instructions to return to the vehicle. For a building administrator the system will provide a mapping mechanism through the definition of a small set of concepts common to all kinds of buildings. This representation is done through a XML specification witch can be generated through JOSM.

Keywords: routing, XML map, JOSM, smartphone, Android

1. Introduction

Most of our time is spent inside buildings like schools and malls. Get to where we want is one of the major problems we face specially when we arrive for the first time at one of this places. It's always an option to ask someone else, search the directory, or even follow signposts, but all of this solutions will require us to memorize information witch is not ideal. One obvious solution in open spaces is the GPS, however GPS is unreliable indoors.

But more than a location problem, routing people inside buildings is a mapping problem. The task is easy for outside areas given the access to satellite images, but such help is not available inside buildings. Furthermore, routing inside buildings presents the vertical component of the problem where users need to travel between levels, witch suggests mapping techniques should be adapted.

Currently, if a user gets into a building for the first time and needs guidance there is no standard system available for the task. The same problem is faced by a building administrator as he doesn't know witch solution should be provided to his users for routing purposes.

For that reason, a generic solution for this problem must not only solve the routing problem for the user, but also provide the mechanisms that allow the building administrator to map his space. Also that mechanism should be simplified as much as possible contributing to the generalisation of the system. For this reason the two most important actors of our system are the user and the building administrator.

Currently, a document such as this is not known, where a solution for this problem is presented from start to finish, taking into account the perspectives from both the user and the building administrator. The result of this work is the following three main artefacts: a working prototype written for the Android platform; a XML specification to map buildings based on OSM (OpenStreetMap[1]) data types and; a modified version of the JOSM (OpenStreetMap Editor) that allows the creation of the XML specification using a GUI.

All software developed is open source and the prototype should be able to work in any kind of building. The administrator should create a XML specification for his building witch can be stored in a file server. Users of the system need only to know the map location and provide it to the prototype. In the end we test the software developed and prove it is able to map complex spaces and provides useful information with good performance.

[1] `https://www.openstreetmap.org`

2. Related work

Indoor navigation systems have greatly evolved in the past years. The best example found in our research is KAILOS [1]. KAILOS aims to provide a global indoor positioning system based on Wi-Fi fingerprints. This fingerprints are collected through crowdsourcing, where users contribute collecting data from the buildings using the tools provided for that purpose. There are also systems using different approaches. PINwI [3] enriches stationary maps with location information using a dead reckoning (DR) technique. DR techniques rely on sensor data gathered from the user mobile device. This means the performance of the system is strongly affected by accumulated errors from sensor readings. RoughMaps [5] relies on a similar approach but in their system the maps may be symbolic, that is, the factor scale is not important. This system is provided with location information through QR codes. iNavigation [4] presents a novel approach to the problem where user location and routing is provided through images. A web server capable of image processing receives queries from the user witch includes a picture taken from his current location. From that picture the system is able to determine user position and presents routing information also through images. Google also includes an indoor navigation system named Google Indoor Maps included in the Google Maps platform. However, this system is only available in some countries. There are also many systems used in malls witch don't rely on location systems; users must provide their location. Despite this type of systems being more close to be a generic solution as they don't rely on location systems, the fact that they are proprietary makes them useful only in some buildings. In the same way many of the other solutions lack the means to become generic as they either don't make the mapping process publicly available or rely on location technologies.

While we consider this to be the best approach aiming towards global indoor positioning systems, the truth is, user collaboration cannot be expected. Also, the precision of such systems may vary due to environmental dynamism. With that said, this technologies won't be further explored in this work as we want our system to be functional regardless. This also means our system will not be real time as that requires real time location systems.

3. Solution

In our solution we keep the maps separated from the application. That requires the introduction of more components other than the user device. In the next section we present this components. Then we present the features of this prototype and we finish the section giving some insight about the algorithms that support the prototype.

3.1. Components

To allow the building owner to share his map, our solution includes a file server as a component. We don't specify any requirement for this server as the prototype is able do download any file through HTTP. The other component of our solution is a computing device to help build the map. To build the map all we require is a text editor. Ideally the computing device should be able to run Java code as this means the XML map can be generated by our version of the JOSM editor. JOSM is a GUI tool and makes the mapping task way easier. Finally the last component of our system is the mobile device itself that runs the prototype presented in the next section.

3.2. Prototype

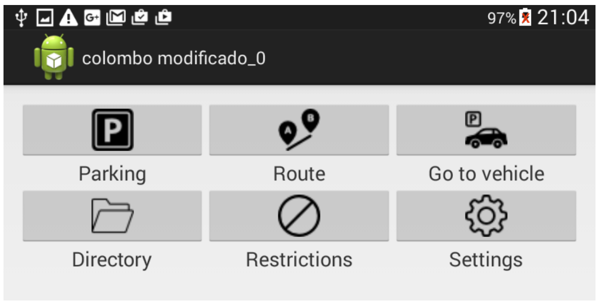


Figure 1: Main menu

The prototype is developed taking into account advanced routing techniques. Its primary feature is allowing the user to choose a starting point and an end point and delivers textual instructions and a 2D map view of the route. The starting and end points must be chosen from a set of pre-defined points, that is, either point must obey a certain criteria. First, the starting point must be unique in the context of a level. This allows the system to identify the location of the user with certainty. Second, the destination points are grouped by name. For example, all WC in the map will be considered as only one. When the user chooses one WC as end point, the route returned will choose the one considered in the best path.

The best path is one that: avoids useless turns, chooses the best vertical access and respects user restrictions. For example, in the presence of two routes with similar total distances, the one with higher distance might be chosen if that saves a turn. We have to remember the more turns in the route the more trouble to the user in following the suggested route. As an example of the best vertical access chosen, in the presence of two different routes to an end point, one through the stairs and one through the elevator, the system will possibly choose the stairs if the route has only one level change. However, if the route includes many level changes, then the route with the elevator is preferred. This is possible due to the concept of travelling costs added to the routing algorithms. Finally we have to refer the concept of restrictions. This is what allows the user to avoid stairs or elevators as he wishes or needs. A panel with a list of restrictions is included as option in the main menu of the application and the user is able to avoid each one of the restrictions included.

Other features of the prototype include parking assistance. The first is helping the user to park in the right place. The scenario witch best shows how useful this feature is starts even before the user reaches the pedestrian area of the building. While driving in the parking area, if the user has a target location on the pedestrian area, then, all he needs to do is type the target location and the result is a list of park-lots where the user should park. Note that this results are assured to be the best from the perspective of the walking effort; not while driving from the current position to the park-lot. It is included more than one result with total distance information as this might help the user avoid driving to another level only to save a few meters. We also assist the user getting back to his vehicle as the par-lot can be saved for later use the feature "Return to vehicle". This feature sets the park-lot as the destination and routes the user from the current position back to the vehicle.

We could not finish this section without mentioning the information provided during routing. We have chosen text and a 2D map (see figure 3) as mediums to deliver routing instructions. Additionally we also provide distance left and total distance of the route. The first textual instruction will always start with an orientation instruction so the user is able to start facing the right direction. Then an action follows witch might tell the user to keep going for a few meters or turn. Whenever possible reference points are also included in the instructions as we think this greatly enriches instructions value. The map is always useful as it provides a birds eye view of the whole level. The complete route is drawn but only the current instruction portion is highlighted. Reference points included in the text are also highlighted. The system tries to zoom the map always to the current instruction context but the user is always free to zoom/pan as he wishes.

This are the key feature of the prototype developed witch, we reinforce, are only useful if the map creation process is kept simple as we show in the next section.

3.3. Map creation

The map creation process is done writing a XML file with the definition of all the concepts present in the building. The list of concepts is kept short to make the whole process as simple as possible. Here is the list of the essential concepts one should include:

- Level
- Vertical access
- Point of interest

There are other concepts available like categories to group a set of points of interest and restrictions to avoid possible obstacles (like a step is for baby strollers and wheelchairs). The level is represented as a group of the other concepts. All the other concepts are assigned a specific coordinate. To make the map creation process simple, it is suggested that the creator starts by drawing all points of interest in a sheet of paper(or blueprint if its available) assigning a coordinate to each one. Coordinates are non negative integers in the Cartesian plane. Ways (one of the data types of the OSM) allow the connection of points of interest between each other and with the vertical accesses. Connections between levels are inferred by a group tag that should give the same name to all accesses vertically connected. There is no need to worry about the real coordinates as the system provides a way to set a scale factor to each level. Figure 3.3 shows an example of the XML mapping of a small building.

This sample shows a building with two levels. Level 0 is defined by the relation in the line 26 and level 1 in the line 31. Each of this levels have one point of interest and one stair access. The stairs are vertically connected as they belong to the same group called "north end" (line 4 and 12). Two ways are defined, one in each level. Each way connects the two nodes that belong to each level. This means its possible to move between the stairs and the point of interest in each level, and between the two levels as they are vertically connected.

In order to ease the burden of XML manipulation, we have tweaked the JOSM editor to support XML generation. With JOSM we can place all nodes and ways over the blueprint, set the respective tags, and export the file as XML using the GUI. This are the basics of mapping. In the next section we present how the system uses this information to provide routing features.

3.4. Algorithms

The algorithms that follow allow the system to compute routes and generate textual instructions.

3.4.1 Graph creation

To provide routing features, the system starts parsing the XML file and builds a line graph. A line graph is a special kind of graph presented in [6]

```
<map name="example map">
  <node id="8" x="86" y="51">
    <tag k="vtype" v="stairs"/>
    <tag k="vgroup" v="north end"/>
  </node>
  <node id="9" x="108" y="57">
    <tag k="pi" v="yes"/>
    <tag k="name" v="Exit 1"/>
  </node>
  <node id="15" x="86" y="51">
    <tag k="vtype" v="stairs"/>
    <tag k="vgroup" v="north end"/>
  </node>
  <node id="16" x="108" y="57">
    <tag k="pi" v="yes"/>
    <tag k="name" v="Entrance1"/>
  </node>
  <way id="2810">
    <nd ref="8"/>
    <nd ref="9"/>
  </way>
  <way id="3000">
    <nd ref="15"/>
    <nd ref="16"/>
  </way>
  <relation id="101">
    <member type="way" ref="2810"/>
    <tag k="type" v="level"/>
    <tag k="level" v="0"/>
  </relation>
  <relation id="501">
    <member type="way" ref="3000"/>
    <tag k="type" v="level"/>
    <tag k="level" v="1"/>
  </relation>
</map>
```

Figure 2: Small building XML sample

that allows assigning turn costs to the graph. This is exactly what's needed in this context to avoid useless turns.

3.4.2 Route search

The concept of cost is a key factor to the route search algorithm. While building the line graph the system computes the distance between each node and assigns it to the respective connection. This value will not change and won't need to be computed again. The cost, on the other hand, is only computed while searching the graph for a route. This means we can change the cost function as needed for the purpose, yet the graph and algorithms don't need to be changed. For the search we use the Dijkstra's algorithm as presented in [2]. This algorithm perfectly fits our needs as our search might be for one of many possible end points (this happens with the WC in any building with many available). Dijkstra algorithm assures we always find the best path to some destination at each iteration; our implementation of the algorithm stops as one end point is found that matches our destination.

3.4.3 Generate instructions

After the route to the destination is found, the system generates textual instructions. The way we see the problem of instruction generation is very simple. First we assume that if the user is able to reach his destination going straight ahead, then all we need is one instruction telling the user to face the right direction and keep going for the total distance. However if the route includes turns then we split the route in segments and generate one instruction for each segment between turns. Also if the route includes level changes, then we need to add a level change instruction between each set of level instructions. In the following list we can see all types of segments and the kind of instruction they will generate:

- StraightSegment - "with stairs behind you keep going for 10m (pass by store 1)"
- TurnSegment - "turn left (follow room 1 direction)
- StraightUntilObject - "keep going straight for 10m until you see kiosk"
- ChangeLevelSegment - "go to level 3"
- FinishSegment - "you have reached your destination"

There are also segments to group other segments but this type of segments don't generate instructions. An example is the StraightUntilTurn that groups both StraightSegment and TurnSegment.

3.4.4 Turn detection and reference point search

To correctly generate instructions some kind of turn detection technique is required. We also include reference points in the instructions as they give confidence to the user that the right path is being followed. Turn detection is done via vector angle determination. We compute the angle between the way segments before and after the next move of the user. If that angle is over a threshold, a turn is considered to take place. This mechanism is required as we have freed the building administrator from explicit cross placement.

Reference point search is done with the help of Dijkstra's algorithm. We want to choose one reference point as close as possible from the beginning of a straight segment and one from the end. We use the Dijkstra's algorithm to search for all points of interest witch distance is less than the total length of the segment. This gives us a set of candidates in a close range. Then, from there we exclude the ones witch route will turn more than once. This excludes all nodes away from the user eye sight. Finally we score each candidate left according to the distance from the segment and the ones closer score best. We choose the ones with the best score, one for each end of the straight segment.

4. Evaluation

As it should be clear, we are trying to provide the best possible solution to both the end user and the building owner. For that reason we cannot finish this document without providing a good evaluation of our proposal. To start with the building owner, we have built a map of one of the biggest malls available in Portugal named Colombo. This map took us less than 4 hours to built with JOSM and includes 285 points of interest distributed by three levels. This seems like a good result considering we only relied on on-line maps.

From the end user stand point, we have shown the prototype provides good features. We tested all the features creating a set of scenarios to prove the correct action was taken by our prototype. The scenarios included maze like route queries that required the algorithm to return routes changing levels more than once in order to reach the destination. They also included searches where the shortest path was restricted to prove restrictions were respected. The prototype was also able to reach the best destination when more than one location was a possibility like the WC example detailed before.

We finish our evaluation testing the performance of our prototype running on a Samsung Galaxy Tab 4 Tablet. First we tested the running time of the algorithms. Considering the challenge of parsing a map with the Colombo's dimensions and the time required for the creation of the line graph (witch takes place as soon as the users selects one of the maps), we got quite good results. We registered a maximum parsing time of 600ms multiple times. The route queries were answered in less than 60ms and same happened with the textual instruction generation witch also took less than 60ms every time. All this execution times are barely noticeable by the user.

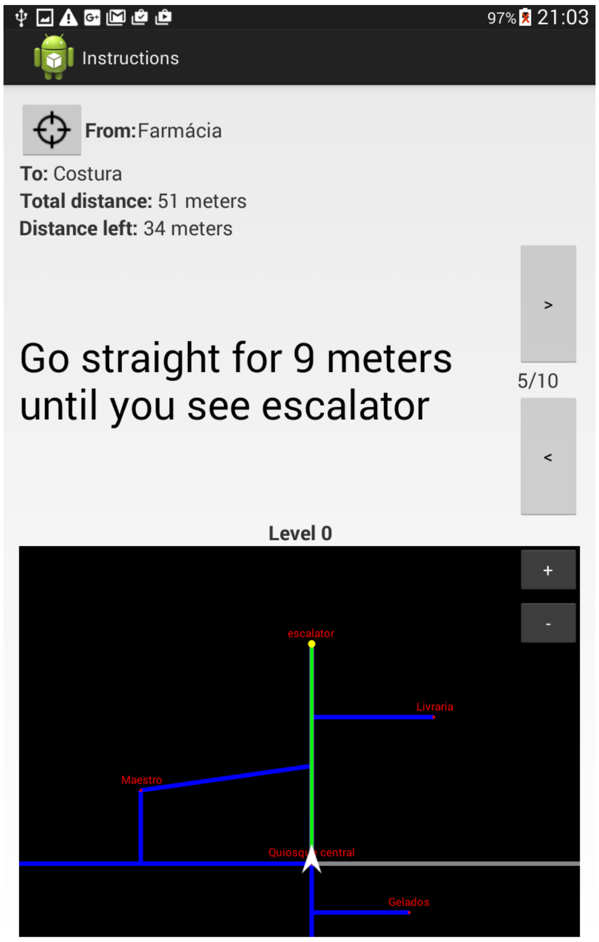
We also measured maximum CPU usage with the "top" command, executed in a remote shell through the ADB (Android Debug Bridge) application. Over the course of multiple queries the values never reached over 12% usage witch were never close to max out the CPU usage.

Finally, we also measured RAM usage through DDMS memory readings. The total maximum memory used were 14,460MB. Considering we added pictures (figure 1) to the main menu, this can also be considered a good result as the pictures take a considerable portion of the total memory.

5. Conclusions

We proposed a general solution for the problem from the point of view of both the building owner and the end user. As a result we created a system witch domain is an aggregation of concepts every building contains. We have also shown how the process of mapping a new building through our specification takes place. To demonstrate the usefulness of the system we also enumerated numerous features included for the routing context. This features were all tested and performance issues were not found. Right now the maps can be created, published and the prototype is ready to download a new map and start to guide people inside buildings. Being an open solution, there are no costs attached. There are also no infrastructure or device requirements hard to comply.

As future work, this system could be tested in a real environment in order to get user feedback. The interface and the instructions can be tweaked if that brings more value to its users. Also, a module to check for new versions of downloaded maps can be included as indoor spaces are very dynamic. In general, the system should be used as much as possible in all kinds of indoor environments in order to get the most feedback possible. However we think this is by now one valid solution for the presented problem.



From: Farmácia
To: Costura
Total distance: 51 meters
Distance left: 34 meters

Go straight for 9 meters until you see escalator

Level 0

Figure 3: Instruction screen

References

[1] D. Han, S. Lee, and S. Kim. Kailos: Kaist indoor locating system. In *Indoor Positioning and Indoor Navigation (IPIN), 2014 International Conference on*, pages 615–619, 2014.

[2] J. F. Kurose and K. W. Ross. *Computer networking : a top-down approach featuring the Internet.* Addison-Wesley, 1st edition, 2001. ISBN:0-201-47711-4.

[3] M. Löchtefeld, S. Gehring, J. Schöning, and A. Krüger. Pinwi: Pedestrian indoor navigation without infrastructure. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, pages 731–734. ACM, 2010.

[4] E. Wang and W. Yan. inavigation: an image based indoor navigation system. *Multimedia Tools and Applications*, 73:1597–1615, 2014.

[5] R. Wasinger, K. Gubi, J. Kay, B. Kummerfeld, M. Fry, and T. Kuflik. Roughmaps a generic platform to support symbolic map use in indoor environments. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, pages 1–10, 2012.

[6] S. Winter. Modeling costs of turns in route planning. *GeoInformatica*, 6:345–361, 2002.