Realistic cities in Robocup Rescue - An Open Street Map to Rescue Converter

Moritz Göbelbecker and Christian Dornhege

Institut für Informatik**
Universität Freiburg
79110 Freiburg
{goebelbe, dornhege}@informatik.uni-freiburg.de

Abstract. In this Infrastructure Competition contribution, we tackle the problem of map generation from public sources. Usually the major problem is not only the data conversion itself, but to get access to the data at all. We solve this problem by using the website OpenStreetMap.org, that provides mapping data for the whole world in a wiki-style concept, as our source of data, thus being able to generate maps for almost any city. Additionally we present not only a data converter, but a plug-in to the JOSM map editor, gaining a fully functional map editor and converter for Rescue maps.

1 Introduction

The Robocup Rescue League was founded with the idea to provide solutions for large scale disaster events as the Great Hanshi-Awaji earthquake in Kobe [1, 2]. Since then the Rescue Simulation League has aimed not only at developing more and more sophisticated multi-agent systems in the agent competition, but members of the community have always been active in the development of realistic simulators for the simulation league [3]. This has been strongly encouraged by the introduction of the infrastructure competition rewarding such development.

(a) (b) (c)

Fig. 1. The figure shows the conversion of OpenStreetMap data to a rescue map for the city of Graz. (a) The raw data in the JOSM editor. (b) The map in JOSM adapted to Rescue specific needs. (c) The converted map in the Morimoto Viewer.

Besides the development of new simulators one major issue is the creation of realistic maps. Although the Kobe map, along with other maps, has served as

** This research was partially supported by DFG as part of the collaborative research center SFB/TR-8 Spatial Cognition R7

one of the default maps for years, it is obvious, that agent development should not focus on a very limited subset of maps. Manually creating maps is a very time consuming process and therefore the addition of new maps has been quite rare. One solution to this problem was the construction of random maps, that eliminates at least the possibility of tuning to each specific map [3]. The problem with random maps is, that although each map is different, they usually follow one construction pattern and thus their structure is still quite similar and uniform.

Realistic cities look very different than random maps, as they have grown over the years and were formed during their development. Almost any city is recognizable by its unique structure showing major roads, rivers, a coastline, as well as parks, open places, downtown, industrial or residential areas. It is very well possible to create such realistic city structures in a randomized way, but the overall goal of the Rescue League is providing solutions for the real world. Thus maps based on real world cities are favourable over random maps.

There have been previous approaches to generate maps based on GIS data [4]. The recurring issue is the acquiration of such official GIS data, that is not generally available for any city. We provide a solution to this problem by using OpenStreetMap [5] as our source of GIS data. OpenStreetMap is a wiki-style website dedicated to provide an open mapping resource for everybody. Almost any big city and even many small cities are accurately mapped, providing GIS data for the whole world. The data is licensed under the Creative Commons licence making it available for reuse in the Rescue League for free. OpenStreetMap provides several editors, one of which is JOSM, that incorporates a plug-in interface. Our contribution is such a plug-in for the Rescue Simulation league, that allows us to access OpenStreetMap data, process it, and finally write maps in the rescue format. As this is a plug-in for the JOSM map editor, we not only provide a converter for GIS data, but also gain a fully functional map editor for Rescue Maps.

In the remainder of this paper we describe the structure of the GIS data in section 2, the processing steps are shown in section 3 and 4, and in section 5 we present examples of converted maps for different cities. Finally we conclude in section 6.

2 GIS Data in the OpenStreetMap World Model

Compared to Robocup Rescue, OpenStreetMap uses a very simple, but at the same time flexible data model. There are only three primitive types [6]:

- **Nodes** have an id and a location, given in longitude and latitude coordinates.
- **Ways** consist of an ordered list of nodes. Depending on its context, it may represent a line (e.g. a street) or an area.
- **Relations** can contain an arbitrary number of other primitives (including other relations) and can be used to model higher level structures.

The underlaying data model is XML, so all semantic information is encoded in *tags*, key-value pairs of which each primitive can have an unlimited amount. Although tags are in principle arbitrary, standardized tags have emerged to represent commonly used information. Thus, the presence of a “highway” tag states that a way represents a road, whose type is further specified by the value of the tag. Table 1 lists a number of common tags that are relevant for rescue maps [7].

tag	semantics
highway	a road of any kind
oneway	true if the road is a one-way street
lanes	number of driving lanes
area	treat the way as an outline of an area
building	for a way: the outline of a building
shop, amenity, building	for nodes: indicate buildings
landuse	general type of a larger area

Table 1. Commonly used tags in OpenStreetMap and their meanings.

There are several tags that are not of direct interest for the rescue simulation, but which might help us to interpolate missing data. For example, the "shop" tag on a node indicates the presence of a buiding at that point, even if there is no outline present.

In the same way, the "landuse" tag on a way can give the automatic building generator hints which kinds of buildings (if any at all) should be placed in that area (e.g. no buildings should be placed in areas with "landuse"="forest") .

3 Map conversion

The map converter was implemented as a plug-in to the OpenStreetMap map editor JOSM [8]. We perform the conversion in two seperate steps. First, we extract the relevant information into an *intermediate representation*. All processing is executed on this intermediate representation. In the second step, this data is saved as a rescue map.

3.1 Intermediate representation

The intermediate representation is a map layer in the OpenStreetMap data model, which contains all the information required by the kernel and simulators in an easy to extract fashion. We chose this approach, because the conversion of data between the different models requires some amount of interpretation of the source data. All the processing will now take part when converting to the intermediate representation.

As this representation is valid OpenStreetMap data, it can be edited, saved and loaded after the lossy parts of the conversion have finished. It is also easy to *import* existing Rescue maps, thus gaining a well maintained map editor with very little effort. Conversion to Rescue maps is then a simple task, as is only requires writing of the already present data.

As all other semantics in OpenStreetMap, we model the Rescue specific data using a set of custom tags. The neccessary information is commonly stored in the form of properties, so we simply create a tag "rcr:*property-name*" = "*property value*" for each property. We introduce a variety of new tags to express different aspects of Rescue maps. These are listed in table 2.

3.2 Roads

Roads in OpenStreetMap are generally identified by the "highway" tag. Unfortunately, simply mapping all ways, containing such a tag, to rescue roads is problematic due to the following reasons:

tag	data type	semantics
rcr:type	{node, road, building}	The basic type of the object
rcr:outline	id	The id of the building outline
rcr:entrances	ids	A list of entrance nodes
rcr:refuge	bool	This building is a refuge
rcr:fire	bool	This building is an ignition point
rcr:ambulances	number	Number of ambulances on this position
rcr:firebrigades	number	Number of fire brigades on this position
rcr:policeforces	number	Number of police forces on this position
rcr:civilians	number	Number of civilians on this position

Table 2. Tags used in the OpenStreetMap representation of Rescue maps

- Roads are mapped for the general public, not for rescue services. There is, e.g., no distinction, if a way tagged as "highway"="pedestrian" is merely reserved for pedestrians, or if it is physically impossible to reach by rescue vehicles.
- Sometimes, especially at larger junctions, each lane is mapped seperately, creating a very complex road graph.
- In OpenStreetMap, ways can also represent areas, that cannot be represented in the Robocup Rescue domain.

Therefore we employ postprocessing on the raw data. First, we add all roads that are certainly passable by rescue forces. Next, among possibly passable roads (like pedestrian zones, cycleways, footways) we select those, that are required to reach all buildings already on the map. The intuition being, that each building will have at least one road connection in the real world.

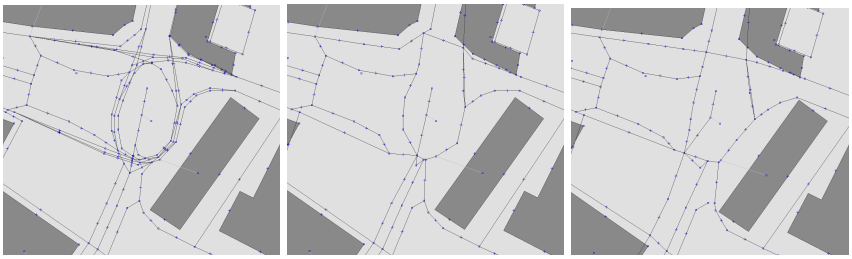
Finally, we perform a reachability check of all roads and remove those, that cannot be reached from the main part of the map, thus eliminating small isolated clusters, that might have been introduced as a map only contains a rectangular section of the world.

To remove redundant roads, we only regard those, that have the "oneway" tag set. We then iterate through each way, and at each crossing decide for each set of way segments with the same orientation (i.e. outgoing or incoming), if they can be merged into one segment. We merge the largest possible set of segments for which the average angular difference is below a certain threshold and for which the average direction of ways on the opposing ends of the segments is roughly similar, thus avoiding merging opposite lanes.

This simplification can introduce some irregularity into the roads, so afterwards we apply a simple smoothing algorithm to the modified roads. Figure 2 shows the different stages of our method.

3.3 Buildings

Buildings can be mapping in OpenStreetMap as ways, that have a "building" tag. The Rescue map format doesn't represent the building itself as a polygon from nodes and edges, but only stores the outline in the building itself. Therefore, when converting a building we construct a *building node*, that is located at the centroid of the building polygon and holds an "rcr:outline" tag, pointing to the corresponding building outline. This building node consequently receives



(a) (b) (c)

Fig. 2. This figure shows the steps of our road simplification algorithm. (a) a complex junction with multiple lanes (b) the same junction after lane merging (c) the final result after smoothing.

all rescue specific tags as, e.g., "rcr:area". When saving a map, the outline is automatically written to the map.

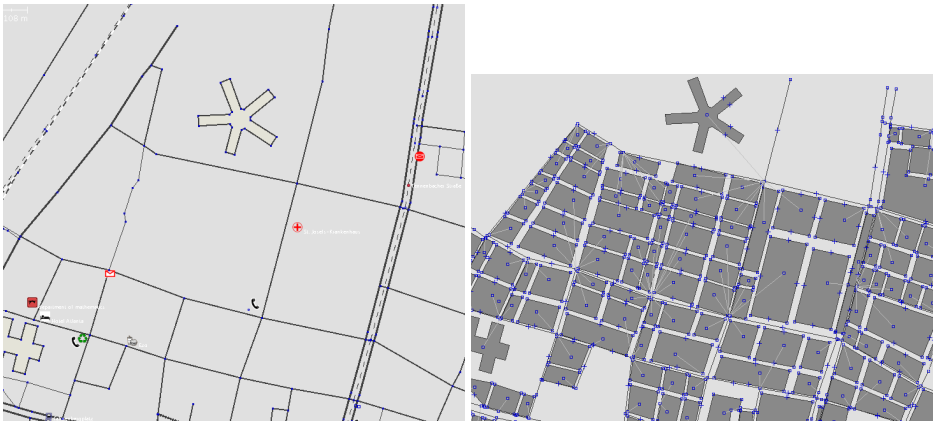
Building entrances are usually not modelled in GIS data, but necessary for rescue operations. Fortunately buildings in OpenStreetMap data can have an address tag, that we can use to select the correct street to connect a building to. If no adress is given we fall back to selecting the nearest road.

One general issue with GIS data, that also applies to OpenStreetMap data, is, that in contrast to roads, not all buildings are modelled. The Rescue League relies on the presence of buildings, so we implemented a building generation algorithm to fill unmapped areas. Basically, the algorithm follows the general idea, that can, for example, also be seen in the rescuecore's random map generator [3]. First we identify city blocks, that are surrounded by streets. Secondly, depending on the size of the block, we split it horizontally and vertically to gain the final buildings. An example of this algorithm can be seen in figure 3.

During this autogeneration, we try to use as much information as the base GIS data allows us to. We obviously retain every already manually mapped building. Sometimes buildings are only mapped as nodes or addresses hinting the location of a building. In such a case, we also place a building at that node. Additionally important information can be gained from marked areas or landuse tags. Those might mark parking areas, parks, forests, and other similar things prohibiting us from placing buildings, thus keeping open places, that naturally appear in the structure of most cities. Finally, when choosing parameters for constructing buildings as their size or the spacing between buildings, we obey "landuse" tags, that might mark areas as "industrial", "residential", and others. Residential areas, for example, are usually formed from many small buildings, that stand farther apart, which will especially influence the spreading of fires in those areas. Although we do not always have the original building data, by using those geographic annotations, we still think, that we can reproduce the non-uniform structure of a city in a representative manner for the Rescue Simulation League.

4 Scenario generation

Rescue maps not only consist of pure GIS data, but also describe a rescue scenario. This consists of marking specific buildings as the stations and refuges,



(a) (b)

Fig. 3. An example of our building generation algorithm. (a) shows the original data. (b) shows the same scene with added buildings. Note, that the already mapped building in the top is kept during the conversion.

placing agents and civilians as well as choosing fire points. Our plug-in can also edit those Rescue specific entities and save them to a map, thus providing a full rescue scenario.

We provide a simple algorithm, that places a given number of those entities on a converted map. As in previous sections we propose the use of semantic information, if applicable. In this case, we can place refuges at buildings, that have been tagged as hospitals. Additionally the probability for fire points can be significantly higher in industrial areas. A simple example is presented in figure 6.

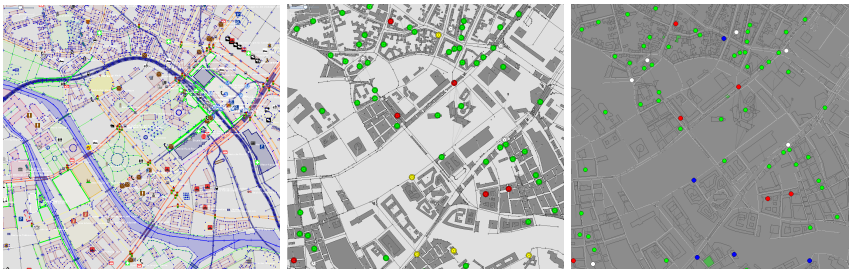
5 Results

In this section we present exemplary conversions of different cities as well as an insight to our implemented tools and the conversion procedure.

The conversion procedure is presented in figure 4 and figure 5. The process of generating a rescue map of any city follows the following few simple steps:

- Start the JOSM-editor and enable the Rescue plug-in.
- Via the download feature in JOSM, choose an excerpt from the world and download the GIS data of this area.
- Click the *Make Rescue Map* button to create the intermediate representation.
- Optionally, the intermediate representation can be edited manually, or a scenario can be generated.
- Save the map to the Rescue format and run it.

The scenario generation is shown in figure 6 during the conversion of London. Scenarios can be adapted by simple parameters shown in the plug-in's dialog. As this data is present in the JOSM map editor all relevant data can also still be edited, so that we not only have a simple conversion program, but a full editor.



(a) (b) (c)

Fig. 4. The figure shows the conversion of OpenStreetMap data to a rescue map for the city of Berlin. (a) The raw data in the JOSM editor. (b) The map in JOSM adapted to Rescue specific needs. (c) The converted map in the Morimoto Viewer.

(a) (b) (c)

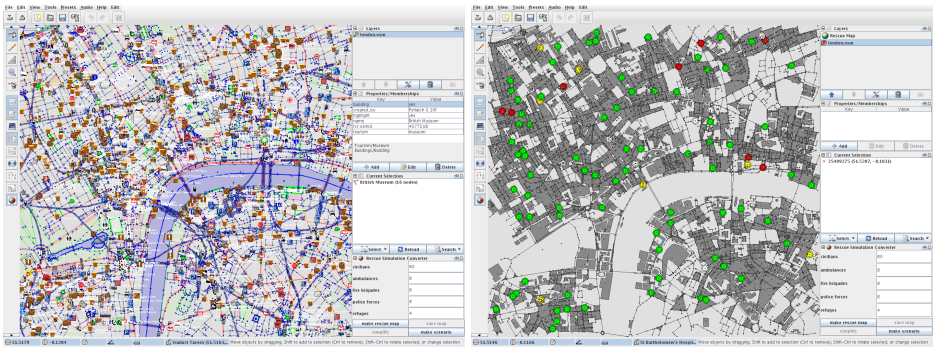
Fig. 5. The process of map conversion for the city of Freiburg. (a) shows the downloaded data in the JOSM editor. (b) shows the intermediate representation in a Rescue layer, that can be saved to Rescue format. (c) is a screenshot of the same map running in the rescue kernel's viewer.

6 Conclusion

We created a converter from OpenStreetMap GIS data to Robocup Rescue Simulation maps. By using OpenStreetMap as our source, we give the possibility to create rescue maps for almost any city on the world, solving the common problem of getting the actual GIS data. Our implementation as a plug-in to the map editor JOSM additionally enables us to not only convert maps, but also inspect and edit converted maps, and finally gains us a fully functional map editor.

References

1. Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, T., Shimada, S.: Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agent research. In: IEEE Conference on Man, Systems and Cybernetics. (1999)
2. RescueSystems: Robocup rescue. `http://www.rescuesystem.org/robocuprescue/` (2008)
3. Robocup: The rescue kernel. `http://sourceforge.net/projects/roborescue/` (2008)



(a) (b)

Fig. 6. This figure shows screenshots from within the JOSM map editor for the city of London. (a) shows the raw GIS data of London. (b) presents the intermediate Rescue layer, that can still be edited before being saved. On the lower right side, the options dialog of our Rescue plug-in is visible.

4. Takahashi, H., Tanigawa, M., Takahashi, T.: Tool kits for using open source gis data as robocup rescue gis maps. In: Robocup 2005. (2005)
5. Openstreetmap.org: Openstreetmap. `http://www.openstreetmap.org/` (2009) (date; 28. February 2009).
6. Openstreetmap.org: Osm protocol version 0.5. `http://wiki.openstreetmap.org/wiki/OSM_Protocol_Version_0.5` (2009) (date; 28. February 2009).
7. Openstreetmap.org: Map features. `http://wiki.openstreetmap.org/wiki/Map_Features` (2009) (date: 28. February 2009).
8. Scholz, I., Stoecker, D.: Josm. `http://josm.OpenStreetMap.de` (2009) (date: 28. February 2009).