FireRevit: Using Revit Files to Identify the Room Locations of Fires and Escape Routes.

LUHAN SHENG AND DENNIS SHASHA, Southwest Jiaotong University and New York University

New York University Computer Science Technical Report TR2020-995

A Revit file is a proprietary format used by Autodesk Revit to store a building model. It contains all the information that describes a building model, such as element and entity data, project location, etc[Fisher 2020]. Since 2010, to enable advanced users and third-party developers to integrate their applications into the Autodesk Revit family of products, Autodesk has permitted developers to use the API provided by Revit to obtain building data[Autodesk 2020]. In fact, one can now process large quantities of Revit files and extract building information automatically[Mason 2009].

Based on this, FireRevit consists of a parser for all the building model files in a given city to get the location of any window in any building and their corresponding rooms, and create a database to persist the data.

In this way, when a fire breaks out in the city and a drone sighting of the fire gives latitude,longitude and height, FireRevit can help firefighters determine the building and room where the fire occurred by retrieving records from the database.

FireRevit also combines the Revit file and information about which rooms are inaccessible due to fire to guide residents to the nearest exit.

Additional Key Words and Phrases: Autodesk, Revit, firefighting, fire rescue

Author's address: Luhan Sheng and Dennis Shasha, Southwest Jiaotong University and New York University, wc36170565@gmail.com,shasha@cims.nyu.edu.

2019. XXXX-XXXX/2019/8-ART1 $15.00
https://doi.org/

Contents

1 ARCHITECTURE

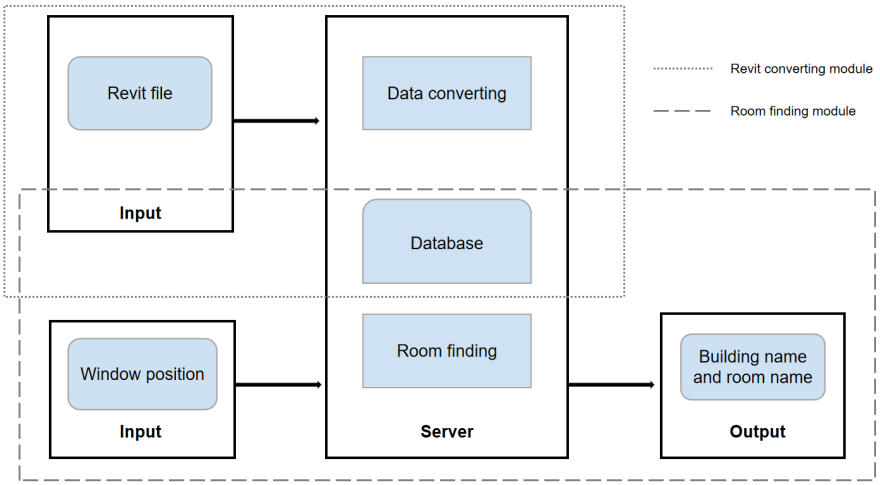

Fig. 1. Flow chart of this project. There are four major components and can be divided into two modules(Revit converting module and Room finding module).

Github repository: https://github.com/LuhanSheng/Revit_To_Database
Revit converting module: folder named *Data converting*: file RevitToDatabase.csproj is the file the user should open in Visual Studio.
Room finding module: folder named *Room finding*: file window_finder.py is the file to open in Python.

1.1 Revit converting module

This module contains two components and its function is to extract the data we need.

1. Revit file: All the Revit files should be provided in a folder. The file name of each file should be set to the building name.

2. Data conversion: This process is responsible for parsing the location of the windows and converting them into geographic locations (latitude, longitude, and height). To achieve this, the process will first find the Revit process on the machine and get authorization from it. So Revit should be installed on the machine before running it. After conversion, the window location and corresponding room information will be stored in a MySQL database. The file names of all converted files will be saved to ensure that a file will not be stored multiple times when FireRevit is run multiple times. In other words, the process will convert only newly added files.

1.2 Room finding module

This module contains two components whose purpose is to find a room corresponding to a window.

1. Window position: The Window position comes in from the drone/detector in the format of (latitude, longitude, height). The height here refers to the height from the window to the ground. In section 4, we will show how to convert this to a height with respect to the base point.

2. Room finding: This process is responsible for finding the window and its corresponding room. The process will start by ranking windows by their proximity to the point in three-space found by the drone. Finally, it will output the building name, room name and corresponding coordinates.

2 CORE CONVERTING AND FINDING STEPS

2.1 Part one: Revit file conversion

To parse the Revit file using the Revit api without opening Revit, acquire the window coordinates and convert them into geographic positions (latitude, longitude, height from the ground). The pseudocode of this part, see below:

Algorithm 1: Converting Revit file to find the window locations of every room

Input: F is a set containing all the Revit files
Output: $NULL$

for *each file* F_i *in the set* F **do**
  $FS \leftarrow$ get the set of all the files which have been converted ;
  if $F_i \in FS$ **then**
    delete F_i from F ;
Find the Revit installed on the machine and load dynamic link library from the Revit;
Get authorization from Revit;
for *each file* F_i *in the set* F **do**
  $f \leftarrow$ get file name ;
  $g_s \leftarrow$ get geographic position of survey point ;
  $b \leftarrow$ get xyz bias of base point ;
  $a \leftarrow$ get the angle to north of the coordinate system;
  $W \leftarrow$ select all the window elements among all elements ;
  for *each window* W_i *in the set* W **do**
    determine the room r the window belongs to ;
    if $r == null$ **then**
      delete W_i from W ;
  for *each window* W_i *in the set* W **do**
    determine the coordinate c of the window ;
    rotate the coordinate a degree around base point $(0, 0)$: $c_r = rotate(c, a)$;
    compute the coordinate of c_r using the survey point as origin point : $c_s = c_r + b$;
    compute the fraction of degrees corresponding to one foot (30.48 centimeters) equal to (the ratio of one foot and one degree of latitude/longitude, e.g. at latitude L1 the ratio is : $r_{long} = cos(abs(L1)/180) * 40076000/360 * 3.28083989501$
    $r_{lat} = 111322.222222222 * 3.28083989501$;
    compute the geographic position bias of the window: $g_b = c_s/r$;
    compute the geographic position of the window: $g_w = g_s + g_b$;
    create a connection to the database ;
    if *database does not exist* **then**
      create a new database and table ;
    store the window information (f, r, g_w) into database ;
    draw a picture to show the positions of windows and base point;
    add F_i into set FS ;

The pseudo-code described above consists of several C files:

(1) Line 1 to line 6 are in Program.cs.
(2) Line 7 to line 31 are in ProcessRevitFiles.cs.

Details about function rotate() used in line 21 of Algorithm 1:
Part of the code of function rotate() in room_finder.py:

```
1    y_xlxc = target_point[0] * (-1) + target_point[1] * 1 / math.tan(angle)
2    x_xlxc = target_point[0] * 1 + target_point[1] * math.tan(angle)
3    x = abs((1 / math.tan(angle) * target_point[0] + target_point[1]) / ((1 + (1 /
         math.tan(angle)) ** 2) ** 0.5))
4    y = abs((-math.tan(angle) * target_point[0] + target_point[1]) / ((1 + math.tan(angle) **
         2) ** 0.5))
```

target_point is the coordinate of the point we want to rotate
target_point[0] refers to x
target_point[1] refers to y
angle is the angle we want to rotate
y_xlxc indicates whether the point is to the left or the right of the y axis
x_xlxc indicates whether the point is to the above or below the x axis
x is the distance from the result point to the y-axis
y is the distance from the result point to the x-axis
We can know which quadrant the target point is in by looking at *y_xlxc* and *x_xlxc*.Also, we know the absolute values of the abscissa and ordinate. Thus, we can get the coordinates of the result point easily. Here is an example to show this:

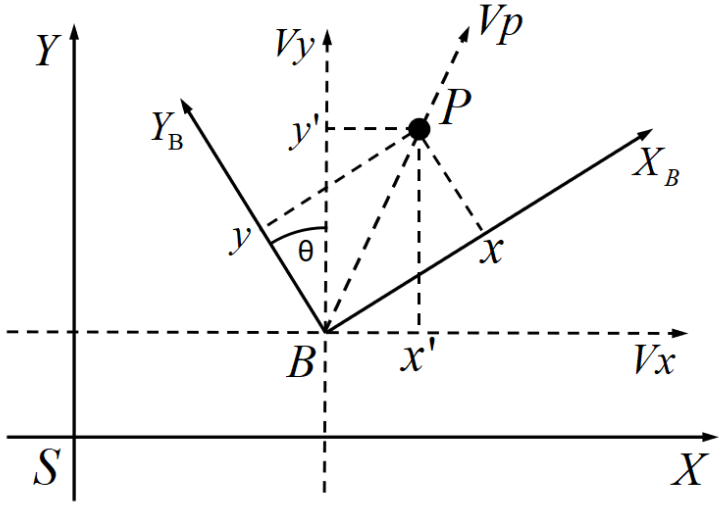

Fig. 2. Coordinate system showing how a point is rotated. S is the survey point, the X-axis is parallel to the weft and the Y-axis is parallel to the warp. B is the base point, coordinate system $Y_B B X_B$ is the coordinate system used in the Revit model. P is the point we want to rotate. θ is the angle we want to rotate, we need to get the value of x' and y'.

First, we multiply vector V_p and V_y. By doing this, we can know whether the angle between V_p and V_y is greater than 90 degrees or not. In the same way, we multiply vector V_p and V_x. Second,

we calculate the distance from the point P to vector V_x and vector V_y to get the absolute value of y' and x'. Third, using the value obtained in the first step, we can determine which quadrant point P is in, and then the sign of x' and y'. Here is an example of rotating multiple points:

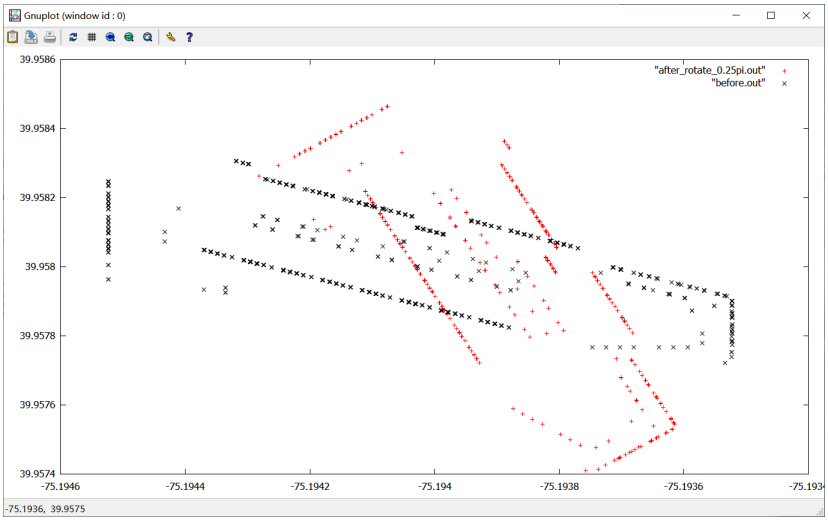

Fig. 3. Red points show point positions after rotating $\pi/4$, black points show the initial positions.

2.2 Part two: room finding

Part two retrieves data from the database to find out the most probable room. Here is the pseudocode of this part:

Algorithm 2: Finding the window closest to the input coordinates and its room

Input: P is the given point
m_t is the tolerance in meters
Output: *buildingname*, *roomname*, *windowposition*

compute the degree tolerance [Degree tolerance is the maximum degree error allowed] d_t: $d_t = 360 * (m_t/40000000)$;
compute the upper bound and lower bound of both latitude and longitude;
query the database by using *BETWEEN* to get a set of results R;
if $len(R) = 0$ **then**
    return *None*;
else
    for *each window* R_i *in the set* R **do**
        compute the distance between R_i and P;
    find the closest window and its corresponding room name and building name;
    return (*buildingname*, *roomname*, *windowposition*);

If we have a tolerance of 10(meters), then the degree tolerance here should be 10/40000000*360 = 0.00009, 4000000 meters is the perimeter of the earth[Wikipedia 2020], so this equation means 10

meters is equal to 0.00009 degree on earth and we will choose as candidates all the rooms within this range. So the tolerance is used to filter away windows that are distant from the calculated point.
The next step is to compute the upper bound and lower bound of both latitude and longitude and using it in the SQL to query the database. SQL used in this example is:

```
SELECT * FROM window.window WHERE (longitude BETWEEN 116.4329305 AND 116.4330694) AND
    (latitude BETWEEN 39.9159305 AND 39.9160694);
```

The query generates a list of possible rooms. The next step is to find out the best one by looking at their distances from the fire point.

3 INSTALLATION

Note: The IDE we used is Visual Studio(VS), So the installation shown here are based on VS. If you use another IDE, you can just regard the part related to VS as a reference.
RevitToDatabase requires python version >=3.5, packages of pymysql, ironPython (v2.7.10) and MySql.Data (v8.0.21).

(1) Clone code from github: *https://github.com/LuhanSheng/Revit_To_Database*
Module data converting is in folder *Data converting*, which is a C sharp project.
Module room finding is in folder *Room finding*, which is a python project.
(2) To use the function of converting the Revit file, please:
(a) Install Revit 2019 and MySQL on the machine.
Here is some guidance for this process (both text and videos):
Install Revit 2019: *https://www.youtube.com/watch?v=Wqd8N78i-eM*
Install MySQL: *https://www.youtube.com/watch?v=WuBcTJnIuzo*
(b) Install and import ironpython and mysql.data in the project. Here is some guidance for this process:
1. Open the C sharp project in VS, then find the solution explorer.

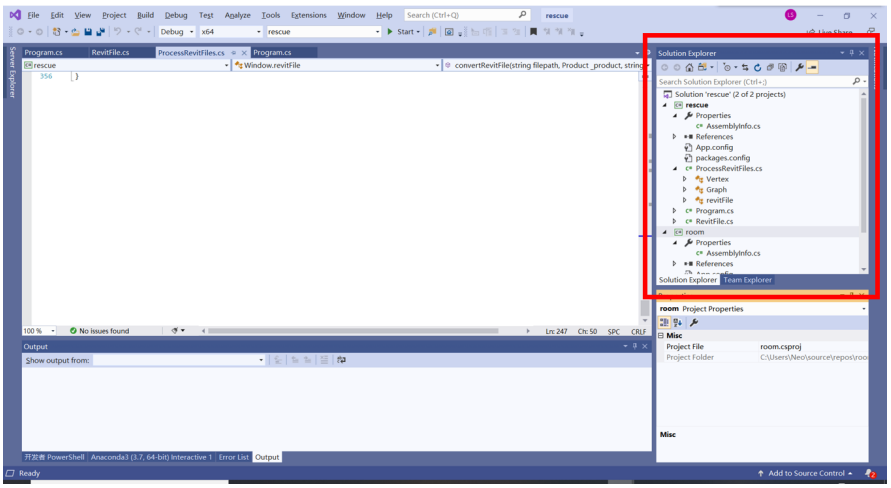

Fig. 4. Visual Studio interface. Red box shows the position of solution explorer.

2. Right click the project in Solution Explorer, then click Manage Nuget.

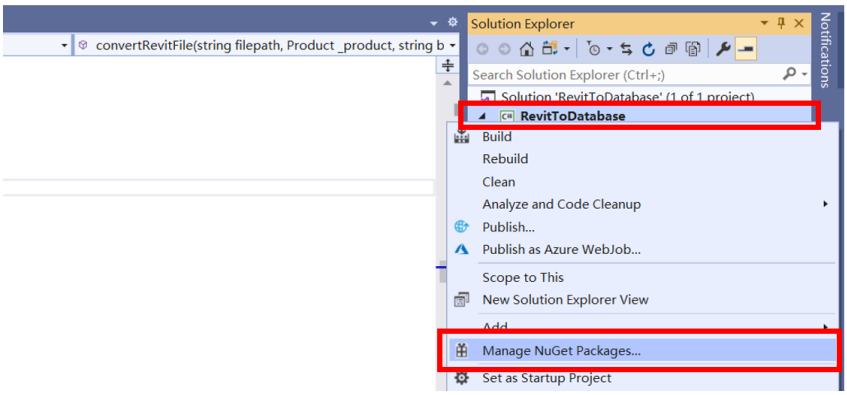

Fig. 5. Visual Studio interface. Red box shows the position of Manage Nuget.

3. Search for package IronPython and MySql.Data and click the button to install them

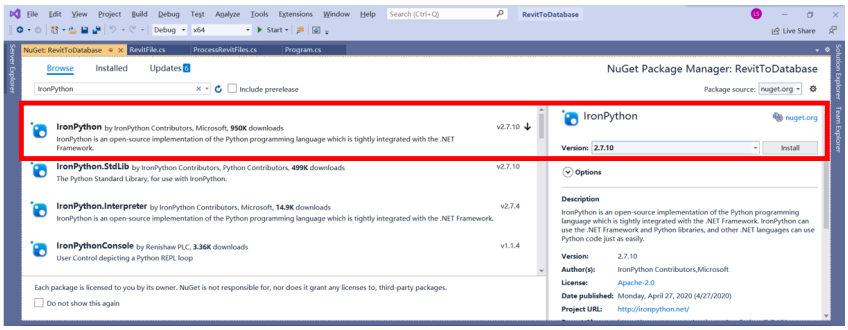

Fig. 6. Visual Studio interface. Red box shows the position of the package we want to install.

(c) Config the reference:

1. Right click the project in Solution Explorer, then click Add Reference...

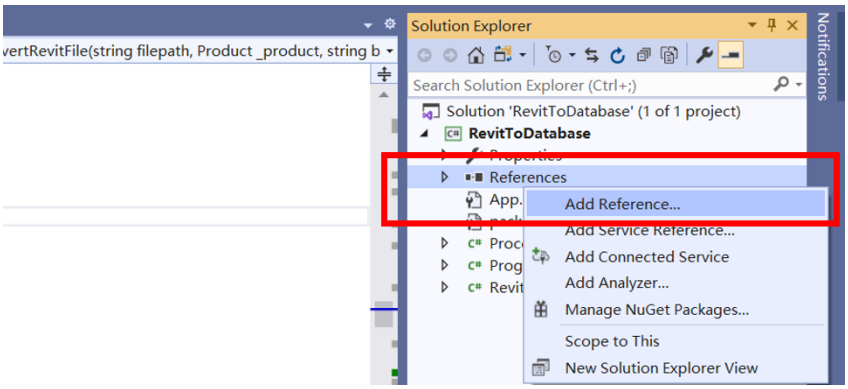

Fig. 7. Visual Studio interface. Red box shows the position of Add Reference

2. Add the references of:
- *RevitAPI.dll*
- *RevitAPIUI.dll*
- *RevitNET.dll*
- *RevitAddInUtility.dll*

(find these assemblies in the installation directory of Revit), as shown in the following figure:

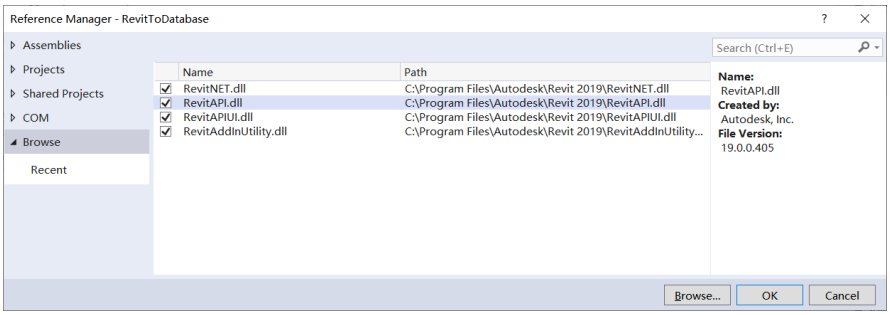

Fig. 8. Reference manager of Visual Studio after adding these references.

3. Click RevitAddInUtility and set the property of Copy Local to True.

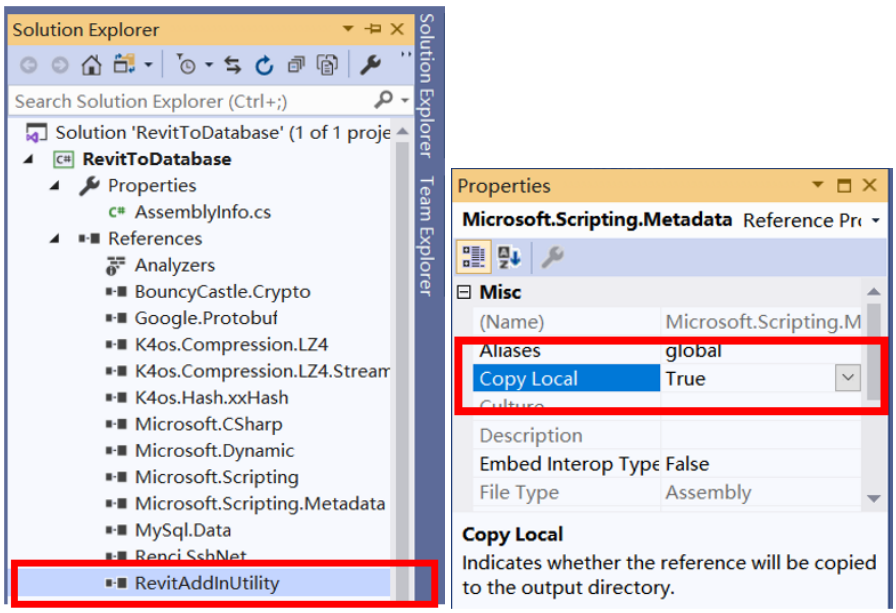

Fig. 9. Visual Studio interface. Red box shows the position of RevitAddInUtility and Copy Local

(3) To use the function of room finding, please run the following command:
pip install pymysql

4 RUNNING THE PROJECT

At present, this project can run only on windows.

4.1 Data conversion

(1) Project configuration
Solution configuration: Debug
Solution platform: x64
Target framework: .NET Framework 4.7.2.

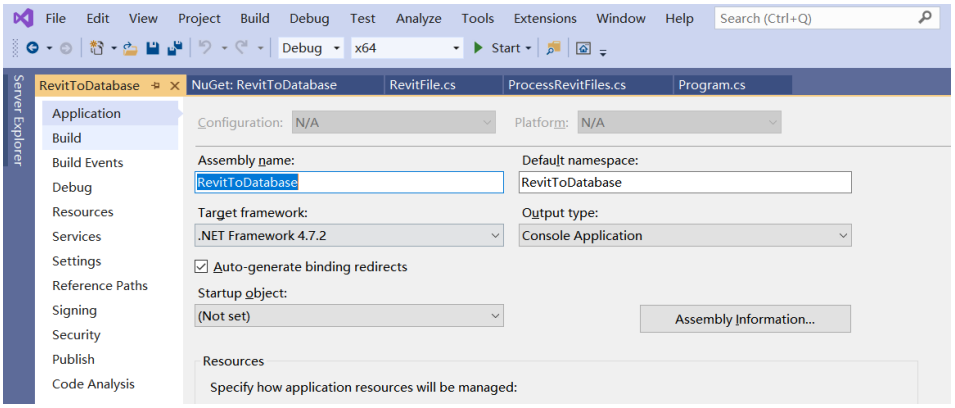

Fig. 10. Project configuration of this project.

(2) Running the project
Click the start button to compile and run.

4.2 Room finding

(1) Project configuration
Please set the tolerance in window_finder.py at line 13 first, the default value is 10(meters).
(2) Running the project
python window_finder.py latitude longitude height
For example:
python window_finder.py 30.5723038 114.2792084 8

4.3 Other requirements

(1) Database configuration
Please fill in your own database username and password in the code, both in file config.cs and in window_finder.py. The default ip address and port number is 127.0.0.1 and 3306, you can also choose to modify them.
(2) Adding Revit files
Please put the Revit file under the folder named rvtFiles. Then, please make sure that there is a file named converted_files.txt in the folder rvtFiles and it should be empty if you have never run FireRevit. File names of all converted files will be stored in converted_files.txt. Once a Revit file's name is recorded in there, this Revit file will not be converted by FireRevit again.
(3) Drone requirements
Drone will have to get the height of the burning window with respect to the first (ground)

floor, yielding DroneFireHeight. To convert this to a Revit height which is with respect to the base point, we simply add (FirstFloorHeight - basepoint) to DroneFireHeight. That is,

$$RevitHeight = DroneFireHeight + (FirstFloorHeight - basepoint). \quad (1)$$

(4) Revit file requirements
Please make sure all the file names are the building names. Then, make sure your Revit file version number is strictly greater than 2013. If not, open the Revit file in Revit to update it. Based on (1) mentioned above, FireRevit needs to know the height of the first floor, and this value is determined by the elevation of Revit level.
So it is necessary to check that the name of the level complies the naming rules, that is, the first floor level has digit "1" in its name. For instance, "LEVEL 01", "L1", "F1", "F01" comply with the naming rules. FireRevit will find out which level's name has number "1" and use its height as the height of the ground floor.

5 CHECKING FOR DATA ERRORS

Sometimes, though rarely, there is a problem in that the base point in the Revit file is not at (0,0). To see whether this is a problem, please follow these steps:

(1) Please open the Revit file in Revit to view the floor plan, click the lamp bulb to show the hidden base point.

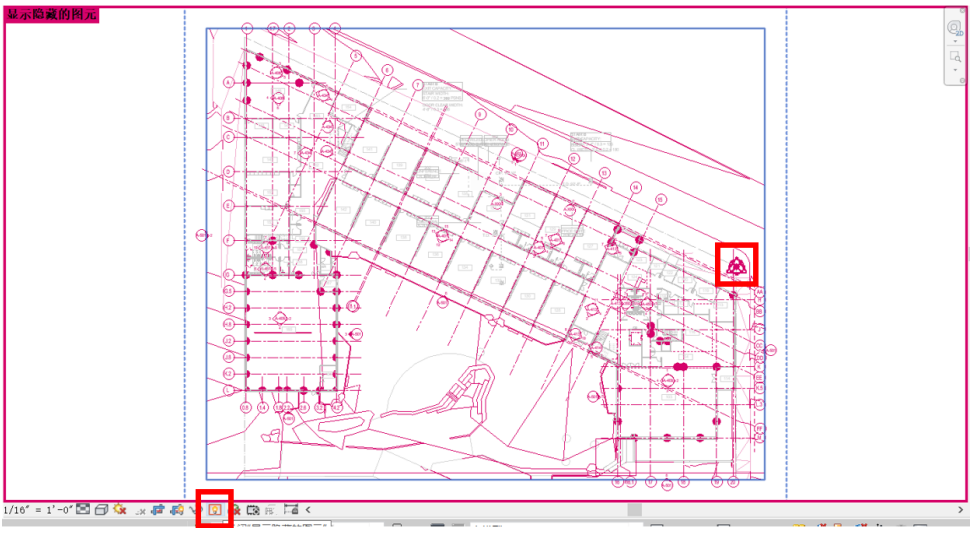

Fig. 11. Revit interface. Red box on the left shows the position of lamp bulb and red box on the right shows the position of base point.

(2) FireRevit produced a file called *filename*.jpg in the folder named images. Now, open it and compare the base points of the two pictures to see if they are the same.

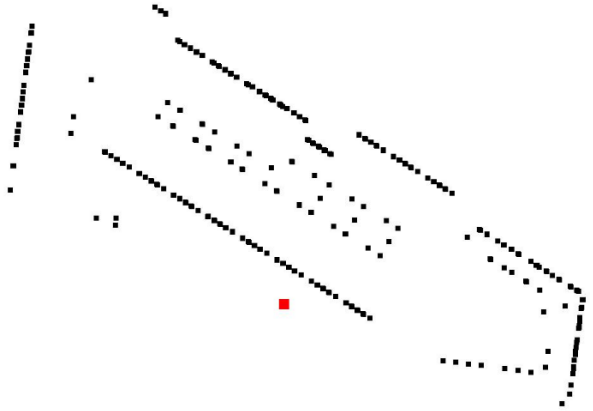

Fig. 12. Sample picture generated after running. Red square shows the position of base point.

Apparently, these two base points are not the same. So there must be something wrong with the Revit file. As of this writing, we can not process the data from this building, so we should simply delete this building model from the database.

(3) Please open the file converted_files.txt in the folder named revitFiles, check that whether all the names of the Revit files you want to convert are in this txt file. Note that there is one file name per line.
(4) After following the instructions mentioned above, the correct data should be stored in the database already. It might be good to check.

6 EXAMPLE

An example to show how FireRevit works.

6.1 First step: Preparing the Revit file

(1) The file name should be the building name. In this example, the building name is canteen.
(2) Put the file into the folder named rvtFiles and make sure converted_files.txt is empty.
(3) You can choose to open this file in Revit, these two pictures shows the design of example file.

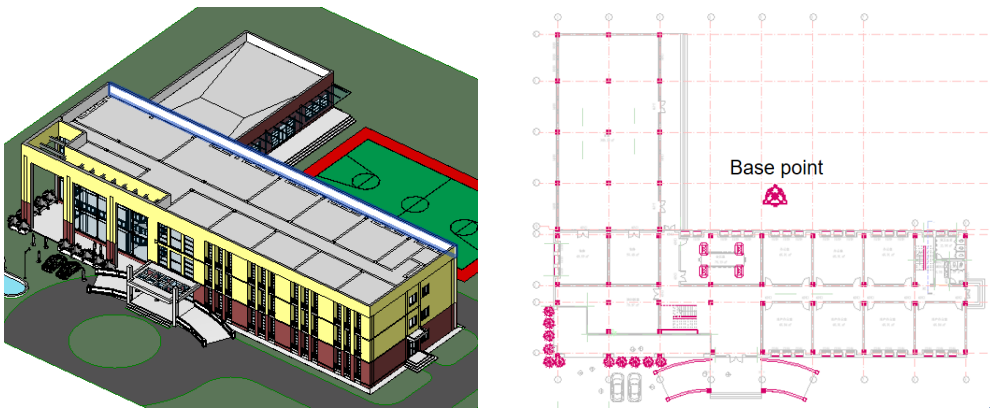

Fig. 13. 3D view and plan view of this building.

6.2 Second step: Running FireRevit

(1) Follow the steps mentioned in section 3 and 4 to install and configure.
(2) Compile and run.
(3) The result will be stored in the database:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

idwindow	building_name	room_name	latitude	longitude	height
690	canteen	食堂 110 | F1	30.5725491713391	114.279052070241	0.900...
691	canteen	食堂 110 | F1	30.5726758306864	114.279052070241	0.900...
692	canteen	食堂 110 | F1	30.5726906525249	114.278920645228	0.100...
693	canteen	食堂 110 | F1	30.5726758306864	114.278920645228	0.100...
694	canteen	食堂 110 | F1	30.5726610088479	114.278920645228	0.100...
695	canteen	资料室 210 | F2	30.5723038474545	114.27920053858	3.7
696	canteen	资料室 210 | F2	30.5723038474545	114.279216032513	3.7
697	canteen	资料室 210 | F2	30.5723038474545	114.279231526447	3.7
698	canteen	办公室 209 | F2	30.5723038474545	114.279297147812	3.7
699	canteen	办公室 209 | F2	30.5723038474545	114.279281653879	3.7
700	canteen	办公室 209 | F2	30.5723038474545	114.279266159945	3.7
701	canteen	办公室 207 | F2	30.5723038474545	114.279331781311	3.7
702	canteen	办公室 207 | F2	30.5723038474545	114.279347275244	3.7
703	canteen	办公室 207 | F2	30.5723038474545	114.279362769178	3.7
704	canteen	办公室 206 | F2	30.5723038474545	114.279428390543	3.7

Fig. 14. Database table and data.

(4) Open the folder whose name is images and check the points in the picture canteen.jpg.

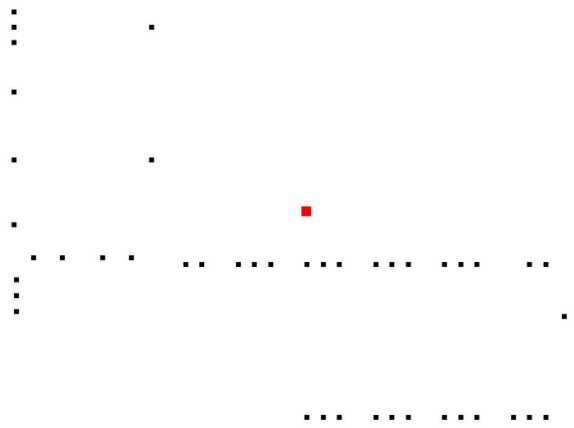

Fig. 15. 3D view and plan view of this building.

Note that it is important to check the position relationship between the window points and the base point. If we have no base point or an incorrect base point, then all the data will be incorrect. Please just delete all the wrong data FireRevit just stored into database.
These points are consistent with the windows in the model we just opened in Revit, see the picture Plan view in subsection 5.1. So we have all the data we want and they are in the database now.

6.3 Third step: Find the target window using given coordinates

After storing the data in the database, we can use the room finding module to find the corresponding room using the given coordinates.

(1) Adjust the variable tolerance in line 13 of window_finder.py to an appropriate value.

```
12  # tolerance unit: meter
13  tolerance = 10
14  degree_tolerance = tolerance / 40000000 * 360
```

Fig. 16. Tolerance. Default value is 10.

The tolerance is a kind of filter to help FireRevit faster. Regardless of the tolerance is, we will get one final result room.

(2) Follow the steps mentioned in section 3 and 4 to install and configure

(3) Run window_finder.py and the coordinates given by the drone is entered as a command line parameter into FireRevit. Now, we assume a fire broke out at the room whose room id is 210, which is also the last room in subsection 5.2, item 3. The drone finds that this room is on fire by looking at its window. The coordinates it gives is 30.5723038, 114.2792084, 8 (here, some error is added deliberately to simulate the actual situation).
The command in this example should be:
python window_finder.py 30.5723038 114.2792084 8

(4) The result will appear on the console:

```
Building name: canteen
Room name: 资料室 210 | F2
Window position: (30.5723038474545, 114.279216032513, 3.7)
```

Fig. 17. Result of running.

Building 'canteen', Room 'xxx 210' is the room we want, level name of this room is F2. This whole room name(room name | level name) can be used in the Escape route finding module, described in the next section.

7 ESCAPE ROUTES FOR PEOPLE INSIDE THE BUILDING

7.1 Architecture

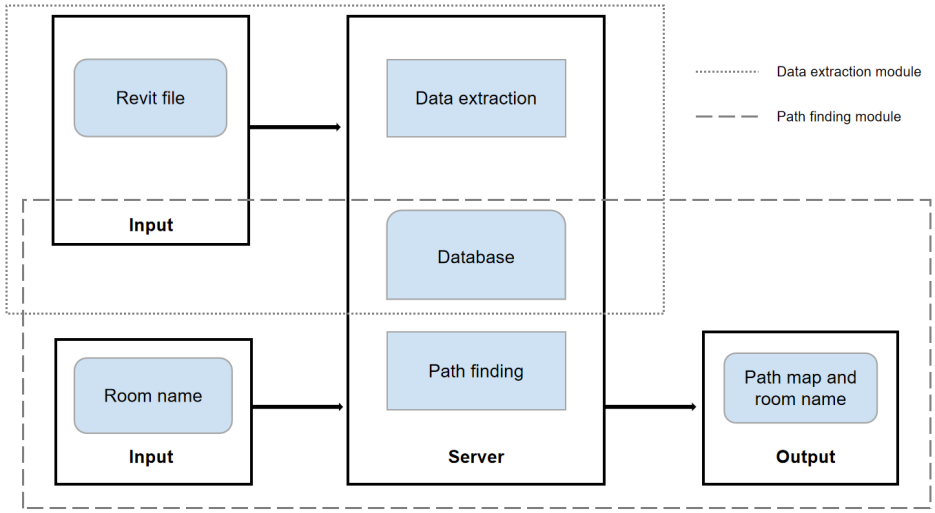

Fig. 18. Flow chart of this part. There are four major components and can be divided into two modules(Data extraction module and Path finding module).

7.2 Brief of Data extraction

This module contains two components and its function is to extract the data we need.
Revit file: All the Revit files should be provided in a folder. The file name of each file should be set to the building name.
Data extraction: This process is responsible for parsing the rooms in the building into nodes and the doors, empty spaces and stairs into edges. The net result is an undirected graph. To achieve this conversion to a graph, the process will first find the Revit process on the machine and get authorization from it. So Revit should be installed on the machine before running it. After conversion, the data of these element mentioned above will be stored in a MySQL database. The file names of all converted files will be saved to ensure that a file will not be stored multiple times when FireRevit is run multiple times.

7.3 Details about data extraction

This part will describe in detail how FireRevit obtains the graph model step by step. Note that this part helps you to understand FireRevit, but if you're just trying to use it, you can skip this section. In order to represent the building model as a graph, we want nodes to be rooms and edges to be connections between rooms. In addition, we need to obtain the information of walls and boundaries to visualize the escape path.

(1) Room(node) data
We represent each room (node) by its center point. The room's name and level is the unique identifier for the room and therefore the corresponding node. In addition, as we will give the shortest possible escape path, we need to know which rooms are exit rooms. Such exit rooms will be regarded as the end of each escape path. For each exit room, we need to save which

door of the room is the exit door. The reason is that if the exit room has many doors or the area of the exit room is large, we can give a more accurate escape path. It is easy for us to get the room location, name and level name by using the Revit API, room name and level will be used as a unique identifier so that the room name stored in the database will be in the form of *roomname* | *levelname*

How do we know which rooms are exit rooms?
The approach is to find the rooms on the ground floor that have boundaries to the outside. If such rooms have doors to the outside, they are exit rooms.
The algorithm is divided into the following steps:

(a) Find the boundaries of all the rooms on the ground floor. If two rooms r1 and r2 share a boundary (technically this means that a boundary line segment from r1 and a boundary line are close together and parallel), then that boundary cannot be a boundary to the outside. In addition, we will discard boundaries that are less than one meter in length, because such boundaries cannot include a door.
(b) Get the center point of all the doors on the ground floor, which are not directly on the outside boundary. If the distance between the center point of the door and the outside boundary is less than a certain value (about 0.6 meters or half the width of a door), then the door will be regarded as an exit door.
(c) Find all the rooms corresponding to exit doors, mark each such room as an exit room. If a room has several exit doors, associate just one with that room in order to be able to give an escapee a path to a specific door. A sample resulting database table might be as follows.

id_node	building_name	room_name	room_location	is_exit	exit_location
3	1507_DREXEL PSLAMS_CENTRAL_190327	WORK ROOM 147 | LEVEL 01	(-144.169516953, 152.545730714, 0.000000000)	false	null
4	1507_DREXEL PSLAMS_CENTRAL_190327	MUSIC 152 | LEVEL 01	(-156.652042226, 104.441519421, 0.000000000)	false	null
5	1507_DREXEL PSLAMS_CENTRAL_190327	VOCAL 153 | LEVEL 01	(-156.652042226, 82.800797931, 0.000000000)	false	null
6	1507_DREXEL PSLAMS_CENTRAL_190327	STOR 150 | LEVEL 01	(-139.115015877, 114.913389407, 0.000000000)	false	null
7	1507_DREXEL PSLAMS_CENTRAL_190327	STOR 151 | LEVEL 01	(-139.164230551, 106.883735224, 0.000000000)	false	null
8	1507_DREXEL PSLAMS_CENTRAL_190327	IT 155 | LEVEL 01	(-139.039230551, 84.329727249, 0.000000000)	false	null
9	1507_DREXEL PSLAMS_CENTRAL_190327	ELEC 156 | LEVEL 01	(-139.039230551, 78.010117838, 0.000000000)	false	null
10	1507_DREXEL PSLAMS_CENTRAL_190327	RESTROOM 160 | LEVEL 01	(-164.581739873, 63.439253514, 0.000000000)	false	null
11	1507_DREXEL PSLAMS_CENTRAL_190327	RESTROOM 159 | LEVEL 01	(-157.355203965, 46.746891689, 0.000000000)	false	null
12	1507_DREXEL PSLAMS_CENTRAL_190327	WC 158 | LEVEL 01	(-146.875607577, 53.132375345, 0.000000000)	false	null
13	1507_DREXEL PSLAMS_CENTRAL_190327	VESTIBULE 161 | LEVEL 01	(-118.296946490, 49.941352504, 0.000000000)	true	(-111.316856540, 53.504597005, 4.041666667)
14	1507_DREXEL PSLAMS_CENTRAL_190327	STORAGE 166 | LEVEL 01	(-115.167619992, -27.795906235, 0.000000000)	false	null
15	1507_DREXEL PSLAMS_CENTRAL_190327	GYM 165 | LEVEL 01	(-143.520439820, 4.541929349, 0.000000000)	true	(-124.181012855, -41.654347147, 3.958333333)
16	1507_DREXEL PSLAMS_CENTRAL_190327	CLASSROOM 142 | LEVEL 01	(-103.937412324, 91.479005087, 0.000000000)	false	null

Fig. 19. Sample database table of room data.

(2) Edge data
Each edge reflects the connection relationship between two rooms in the model.
There are three kinds of connections between rooms: door connection, boundary connection and stair connection. So we need to capture all three of these connections.
Edge data will include two different rooms (the two rooms connected by this edge), the length of the edge (from room center to room center), the type of edge.
The algorithm is divided into the following steps:

(a) Find all the doors in the model, get the room that the door faces, the room where the door is located and the room with the door. Remove the duplicate rooms in the three rooms obtained, and the remaining two rooms are the rooms connected by the door.
(b) Find all the stairs in the model and get the coordinates of the top and bottom of the stairs. Because a staircase may connect many floors, we start from the bottom coordinate, and increase the height of the coordinate by one meter until the added height is greater than the top coordinate, and get the room that this point belongs to. In this way, we can get the all the rooms connected by this staircase.

(c) In the Revit model, the boundaries between rooms are not always walls. Sometimes they can be boundaries without walls. In other words, a whole empty space may contain multiple rooms.So, first of all, We find all room boundaries and determine which boundaries contain no walls. Then, if two rooms r1 and r2 share a boundary and this boundary contains no walls (technically this means that a boundary line segment from room r1 and a boundary line from r2 are close together and parallel), then the two rooms to which the two boundaries belong are connected.
So using this method, we can get all the edges and store the edge data into the database. A sample database table is as follows.

id_edge	building_name	node1	node2	length	edge_type	edge_location
1	1507_DREXEL PSLAMS_CENTRAL_190327	IT 255 | LEVEL 02	CORRIDOR 269 | LEVEL 02	26.6174572912809	door	(-136.536757350768, 83.040541483947, 13)
2	1507_DREXEL PSLAMS_CENTRAL_190327	STAIR C | LEVEL 02	CORRIDOR 269 | LEVEL 02	46.5036367329697	door	(-136.53675735076, 71.6290831506284, 13)
3	1507_DREXEL PSLAMS_CENTRAL_190327	MUSIC 152 | LEVEL 01	LOBBY 162 | LEVEL 01	38.3653939191584	door	(-144.537966934009, 96.9301248172805, -1.4...
4	1507_DREXEL PSLAMS_CENTRAL_190327	VOCAL 153 | LEVEL 01	LOBBY 162 | LEVEL 01	52.9912890556613	door	(-144.537966934006, 92.0863748172805, -1.4...
5	1507_DREXEL PSLAMS_CENTRAL_190327	IT 155 | LEVEL 01	LOBBY 162 | LEVEL 01	42.9064065327169	door	(-136.552778133922, 83.040541483947, -1.47...
6	1507_DREXEL PSLAMS_CENTRAL_190327	ELEC 156 | LEVEL 01	LOBBY 162 | LEVEL 01	48.8725714648617	door	(-136.552778133922, 78.9676248172803, -1.47...
7	1507_DREXEL PSLAMS_CENTRAL_190327	JC 157 | LEVEL 01	LOBBY 162 | LEVEL 01	61.4806986046378	door	(-148.136890604233, 65.3274621957223, -1.4...
8	1507_DREXEL PSLAMS_CENTRAL_190327	ELEC 256 | LEVEL 02	CORRIDOR 269 | LEVEL 02	33.3505290678872	door	(-136.536757350765, 78.9676248172803, 13)
9	1507_DREXEL PSLAMS_CENTRAL_190327	PREP 249 | LEVEL 02	SCIENCE 250 | LEVEL 02	26.6959108000257	door	(-145.833169972319, 112.112305849032, 13)
10	1507_DREXEL PSLAMS_CENTRAL_190327	PREP 249 | LEVEL 02	SCIENCE 248 | LEVEL 02	23.149852840583	door	(-145.833169972318, 120.07725383767, 13)
11	1507_DREXEL PSLAMS_CENTRAL_190327	BLDG STOR 216 | LEVEL 02	CORRIDOR 268 | LEVEL 02	11.7934439650095	door	(146.2416320102669, 12.9550128443357, 13)
12	1507_DREXEL PSLAMS_CENTRAL_190327	JC 206 | LEVEL 02	CORRIDOR 268 | LEVEL 02	13.9557410262563	door	(121.400275541145, 12.9550128443358, 13)
13	1507_DREXEL PSLAMS_CENTRAL_190327	CNSLR 121 | LEVEL 01	CORRIDOR 105 | LEVEL 01	28.3312066816086	door	(124.853841856957, 44.8158936971465, -1.4...
14	1507_DREXEL PSLAMS_CENTRAL_190327	DEAN 122 | LEVEL 01	CORRIDOR 105 | LEVEL 01	24.7203091883265	door	(128.775883506012, 42.9723205447237, -1.4...
15	1507_DREXEL PSLAMS_CENTRAL_190327	PRINCIPAL 120 | LEVEL 01	CORRIDOR 105 | LEVEL 01	28.0285566033232	door	(148.384855469641, 33.7550359023O6, -1.47...
16	1507_DREXEL PSLAMS_CENTRAL_190327	CONFERENCE 119 | LEVE...	CORRIDOR 105 | LEVEL 01	37.9387962323467	door	(152.305865295392, 31.9119477630249, -1.4...
17	1507_DREXEL PSLAMS_CENTRAL_190327	VESTIBULE 102 | LEVEL 01	LOBBY 101 | LEVEL 01	12.377947328053	door	(96.4296162243405, 58.0449798653768, 0.16...

Fig. 20. Sample database table of edge data.

(3) Wall data
In addition to getting the information of all the nodes and edges, we also need the data of all the walls, so that we can draw the buildings in the path map to better visualize the path.
All the edges are obtained directly through the Revit API, and each edge is represented by its start and end point coordinates, and then data is stored in the database. A sample database table is as follows.

id_wall	building_name	room_name	start_point	end_point
1	1507_DREXEL PSLAMS_CENTRAL_190327	ART 149 | LEVEL 01	(-174.807524089, 143.838458151, 0.000000000)	(-174.807524089, 119.639499817, 0.000000000)
2	1507_DREXEL PSLAMS_CENTRAL_190327	ART 149 | LEVEL 01	(-174.807524089, 119.639499817, 0.000000000)	(-143.079171222, 119.639499817, 0.000000000)
3	1507_DREXEL PSLAMS_CENTRAL_190327	ART 149 | LEVEL 01	(-143.079171222, 119.639499817, 0.000000000)	(-135.401274089, 119.639499817, 0.000000000)
4	1507_DREXEL PSLAMS_CENTRAL_190327	ART 149 | LEVEL 01	(-135.401274089, 119.639499817, 0.000000000)	(-135.401274089, 143.838458151, 0.000000000)
5	1507_DREXEL PSLAMS_CENTRAL_190327	ART 149 | LEVEL 01	(-135.401274089, 143.838458151, 0.000000000)	(-155.390857420, 143.838458151, 0.000000000)
6	1507_DREXEL PSLAMS_CENTRAL_190327	ART 149 | LEVEL 01	(-155.390857420, 143.838458151, 0.000000000)	(-174.807524089, 143.838458151, 0.000000000)
7	1507_DREXEL PSLAMS_CENTRAL_190327	CONFERENCE 148 | LEVEL 01	(-174.807524089, 157.840544259, 0.000000000)	(-174.807524089, 144.348874817, 0.000000000)
8	1507_DREXEL PSLAMS_CENTRAL_190327	CONFERENCE 148 | LEVEL 01	(-174.807524089, 144.348874817, 0.000000000)	(-155.646065753, 144.348874817, 0.000000000)
9	1507_DREXEL PSLAMS_CENTRAL_190327	CONFERENCE 148 | LEVEL 01	(-155.646065753, 144.348874817, 0.000000000)	(-155.646065753, 157.840544259, 0.000000000)
10	1507_DREXEL PSLAMS_CENTRAL_190327	CONFERENCE 148 | LEVEL 01	(-155.646065753, 157.840544259, 0.000000000)	(-174.807524089, 157.840544259, 0.000000000)
11	1507_DREXEL PSLAMS_CENTRAL_190327	WORK ROOM 147 | LEVEL 01	(-155.135649086, 144.348874817, 0.000000000)	(-135.401274089, 144.348874817, 0.000000000)
12	1507_DREXEL PSLAMS_CENTRAL_190327	WORK ROOM 147 | LEVEL 01	(-135.401274089, 144.348874817, 0.000000000)	(-135.401274089, 149.926074708, 0.000000000)
13	1507_DREXEL PSLAMS_CENTRAL_190327	WORK ROOM 147 | LEVEL 01	(-135.401274089, 149.926074708, 0.000000000)	(-135.401274089, 157.840544259, 0.000000000)
14	1507_DREXEL PSLAMS_CENTRAL_190327	WORK ROOM 147 | LEVEL 01	(-135.401274089, 157.840544259, 0.000000000)	(-155.135649086, 157.840544259, 0.000000000)
15	1507_DREXEL PSLAMS_CENTRAL_190327	WORK ROOM 147 | LEVEL 01	(-155.135649086, 157.840544259, 0.000000000)	(-155.135649086, 144.348874817, 0.000000000)

Fig. 21. Sample database table of wall data.

7.4 Overview Path Finding

This module contains two components whose purpose is to find the shortest route to the exit.
Room name: The name of the room in which the person is.
Path finding This process is responsible for finding the shortest route to the exit. The process will use Dijkstra algorithm to find all the route to the exits and output the shortest one. Besides that, some pictures showing the path will be given.

7.5 Details of Path finding

This part will describe in detail how FireRevit reads the data in the database to build a graph, and finds the shortest escape path according to the entered room name and building name. Note that this part helps you to understand FireRevit, but if you're just trying to use it, you can skip this section. FireRevit takes room and building names as input and produces route as output. A simple list of room names is difficult for users to understand how to escape, so FireRevit will print a two dimensional escape path. If the user is on the first floor, it will print the path directly. If the user is on the second floor or above, FireRevit will print the path by floor. For example, if the user is on the second floor, FireRevit will print a path map to guide the user to the stairs leading to the first floor, and then print a map showing the path from the stairs on the first floor to the exit.

7.6 Requirements on the Information in the Revit File

The Revit model is used to represent the building, not for precise calculation, so many models may be unsuitable for the escape problem. In order to ensure that the data obtained is as accurate as possible, the following requirements on the Revit file should be met:

(1) Room names must be unique
The fire escape module uses the room name as the unique identification of the room. If many rooms have the same name, the path will be in error.
(2) All spaces on each floor of the building need corresponding rooms
The mistake in many models is that there are many spaces in the building that do not belong to any room, they are just empty spaces. This will lead FireRevit to mistakenly think that these spaces are outside the building and mark the rooms around such spaces as exit rooms.
(3) The file version should be greater than or equal to 2015, preferably 2019.
Any older version of the Revit file may not be compatible with the new API. Opening a file in Revit 2019 to upgrade the version of the file is helpful.

7.7 Installation

Note that most of the installation steps in this section are the same as those in Section 3. We repeat them here, but without figures.
The IDE we used is Visual Studio(VS), So the installation shown here are based on VS. If you use another IDE, you can just regard this description as a reference. RevitToDatabase requires python version >=3.5, packages of pymysql, ironPython (v2.7.10) and MySql.Data (v8.0.21)

(1) Clone code from github: https://github.com/LuhanSheng/Revit_To_Database
Module data extraction is in folder *rescue*, which is a C sharp project.
Module path finding is in folder *Room finding*, which is a python project.
(2) To use the function of converting the Revit file, please:
(a) Install Revit 2019 and MySQL on the machine.
Here is some guidance for this process (both text and videos):
Install Revit 2019: *https://www.youtube.com/watch?v=Wqd8N78i-eM*
Install MySQL: *https://www.youtube.com/watch?v=WuBcTJnIuzo*
(b) Install and import ironpython and mysql.data in the project. Here is some guidance for this process:
1. Open the C sharp project in VS, then find the solution explorer.
2. Right click the project in Solution Explorer, then click Manage Nuget.
3. Search for package IronPython and MySql.Data and click the button to install them

(c) Configure the reference:
1. Right click the project in Solution Explorer, then click Add Reference...
2. Add the references of:
- *RevitAPI.dll*
- *RevitAPIUI.dll*
- *RevitNET.dll*
- *RevitAddInUtility.dll*

Find these assemblies in the installation directory of Revit.
3. Click RevitAddInUtility and set the property of Copy Local to True.

(3) To use the function of path finding, please run the following command:
pip install pymysql
pip install matplotlib

7.8 Running the project

At present, this project can run only on Windows. We have not been able to run this on Linux.

(1) Data extraction
(a) Project configuration
Solution configuration: Debug
Solution platform: x64
Target framework: .NET Framework 4.7.2.

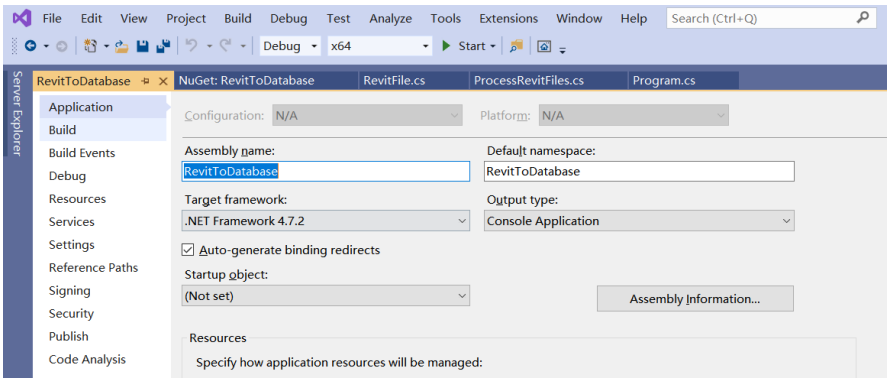

Fig. 22. Project configuration in Visual Studio.

(b) Running the project
Click start button to compile and run

(2) Room finding
(a) Project configuration
Please set the database configuration, room name and building name in line 7 through line 15 in the file rescue.py.
(b) Running the project
python rescue.py

(3) Other requirements
(a) Database configuration
Please fill in your own database username and password in the code, both in file config.cs and in rescue.py.

(b) Adding Revit files
Please put the Revit file under the folder: rvtFiles. Then, please make sure that there is a file named converted_files.txt in the folder rvtFiles and it should be empty if you have never run FireRevit. File names of all converted files will be stored in converted_files.txt. Once a Revit file's name is recorded in there, this Revit file will not be converted by FireRevit again.

(c) Revit file requirements
Please make sure all the file names are the building names and make sure your Revit file version number is strictly greater than 2015. If not, open the Revit file in Revit to update it. Based on item 3 mentioned above, FireRevit needs to know the height of the first floor, and this value is determined by the elevation of Revit level.
So it is necessary to check that the name of the level complies the naming rules, that is, the first floor level has digit "1" in its name. For instance, "LEVEL 01", "L1", "F1", "F01" comply with the naming rules. FireRevit will find out which level's name has number "1" and use its height as the height of the ground floor.

7.9 Example

Here is an example to show how the escape module of FireRevit works.

(1) First step: Preparing the Revit file
1. The file name should be the building name. Building name of this file is 1507_*DREXEL PSLAMS_CENTRAL*_190327.
2. Put the file into the folder named rvtFiles and make sure converted_files.txt is empty.

(2) Second step: Running FireRevit(data extraction)
1. Follow the steps mentioned to install and configure.
2. Compile and run.
3. The result will be stored in the database, Four tables will be created, including node, edge, wall and fire. The table named fire is used to store the fire room.

(3) Third step: Find route to the exit

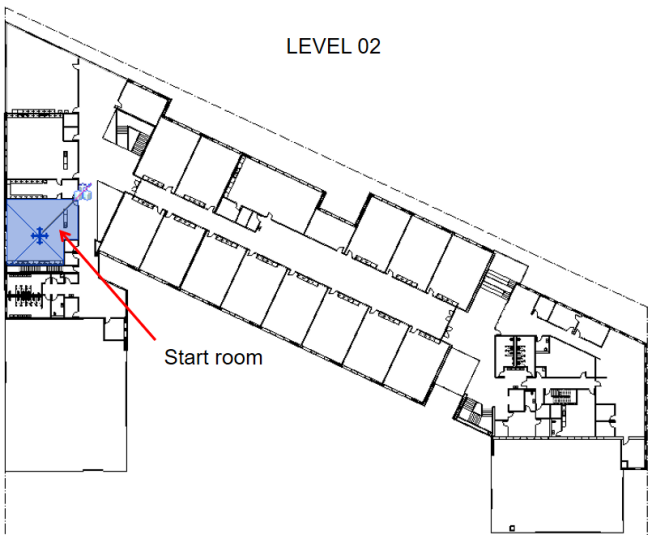

Fig. 23. Position of start room in the floor plan.

We set a starting room as the picture shown above which is on the first floor and its name is SCIENCE 250 and its level name is LEVEL 02. (Note: Usually, the room name and level name are given by room finding module). So the room name and building name should be set to SCIENCE 250 | LEVEL 02 and 1507_DREXEL PSLAMS_CENTRAL_190327.
After running FireRevit, the result will appear on the console:

```
step 1:
SCIENCE 250 --> CORRIDOR 269 --> SMALL GROUP 244 --> STAIR
step 2:
SPECIAL ED 154 --> SMALL GROUP 163 --> LOBBY 162 --> VESTIBULE 161 --> EXIT

Process finished with exit code 0
```

Fig. 24. Position of start room in the floor plan.

FireRevit gives the route to the exit in the form of a series of room names. We can know from the text that we start from room SCIENCE 250 and go to the stair, then go down stair and follow the path to get to the exit.
Alternatively, we may check the map in the folder named picture given by FireRevit:

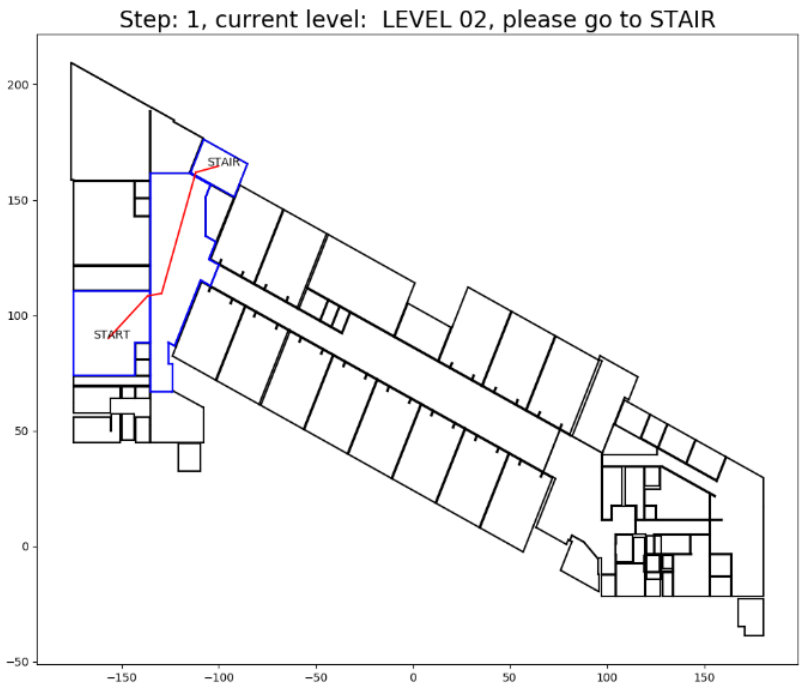

Fig. 25. First map generated by FireRevit.

Fig. 26. Second map generated by FireRevit.

The rooms passing by are marked in blue and the route is given in red. We should start from the start point and reach the end of the red line.
The first picture tells us that we start from the start room and go to the stair room to be able to descend the stairs. The second picture tells us that after following step 1, we are now at the ground floor(LEVEL 01), which means that we have gone downstairs. Then we should follow the red line to get to the exit room, which has already been marked on the map. After doing these two steps, we have reached our destination.

(4) Set a fire room
Now, we add a fire room, that is, store a fire room into the database table named fire. (The tool we use here to modify the database is Workbench. One can also use the command line interface to MySQL.) The building name is *1507_DREXEL PSLAMS_CENTRAL_190327* and the fire room name given by room finding module is *LOBBY 162 | LEVEL 01*.
So the database table of fire should be:

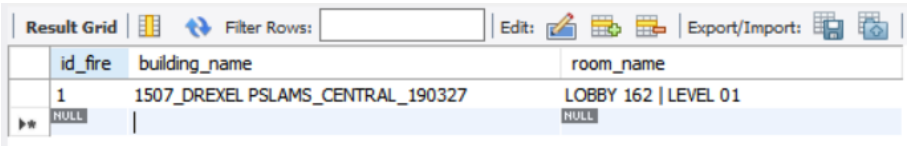

Fig. 27. First map generated by FireRevit.

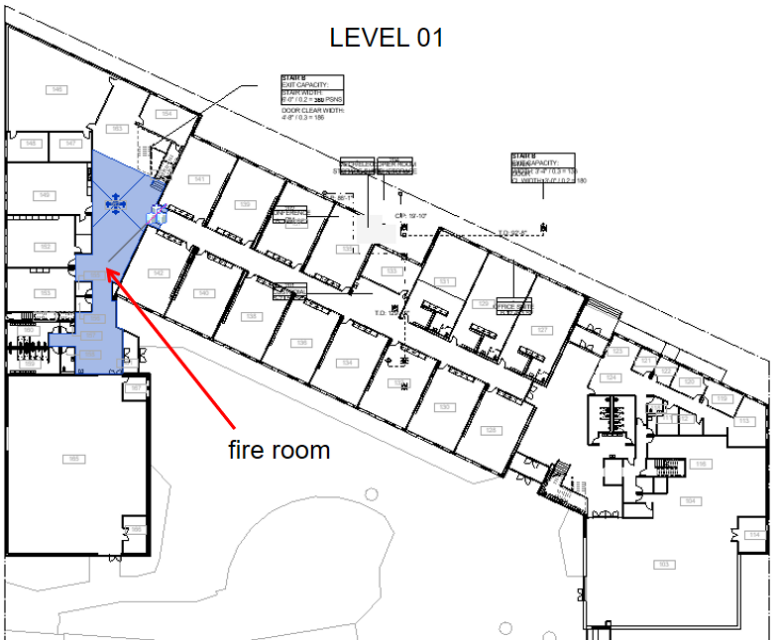

Fig. 28. The fire room we set is on the ground floor.

This fire room is just on the previous path FireRevit just gave us. Which means the former path is undesirable. Now we rerun FireRevit. The result is as follows.

```
step 1:
SCIENCE 250 --> CORRIDOR 269 --> CORRIDOR 226 --> CORRIDOR 245 --> STAIR
step 2:
CORRIDOR 166 --> LOBBY 101 --> VESTIBULE 102 --> EXIT

Process finished with exit code 0
```

Fig. 29. First map generated by FireRevit(after fire).

FireRevit gives us a new route that is different from the previous one. It tells us that we also start from room SCIENCE 250 and go to the stair then go down stair and reach the exit, but the room this path passed by is different. Also, we check the map generated this time.

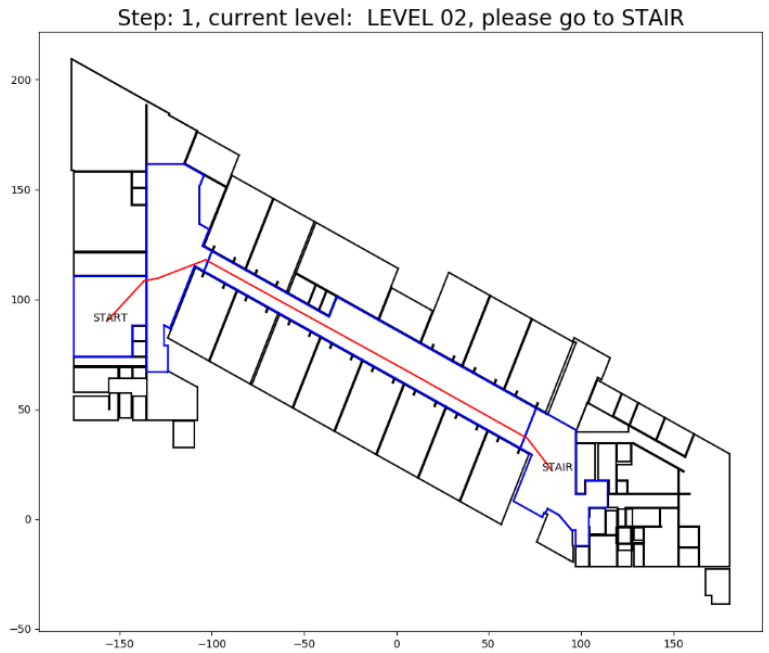

Fig. 30. First map generated by FireRevit(after fire).

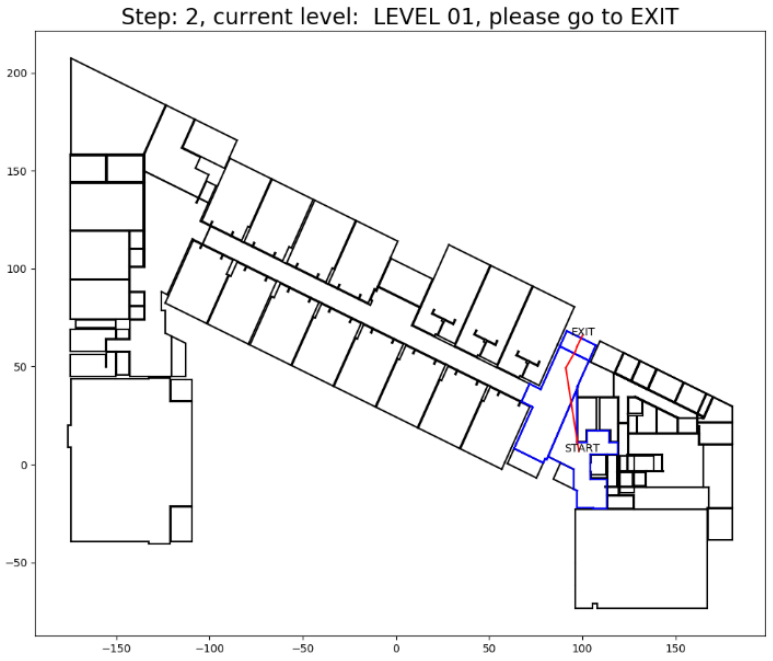

Fig. 31. First map generated by FireRevit(after fire).

FireRevit gave us a completely different path, bypassing the burning room and directing us to the exit at the other side of the building.
According to the map, we went from the second floor to the other side of the stairs to the first floor, and then go to the exit on the first floor. In this way, we managed to escape.

8 CONCLUSION

FireRevit is software to help firefighters determine the rooms where a fire is and to help residents escape. The software makes use of the Revit API and could be used for all Revit-designed buildings of a city.

REFERENCES

Autodesk. 2020. Revit API Docs. https://www.revitapidocs.com/
Tim Fisher. 2020. What Is an RVT File? https://www.lifewire.com/rvt-file-4175424
Matt Mason. 2009. Extending BIM Design Value Using the Revit Api. https://www.augi.com/articles/detail/extending-bim-design-value-using-the-revit-api
Wikipedia. 2020. Earth's circumference. https://en.wikipedia.org/wiki/Earth%27s_circumference