COMS W4735 Assignment 3

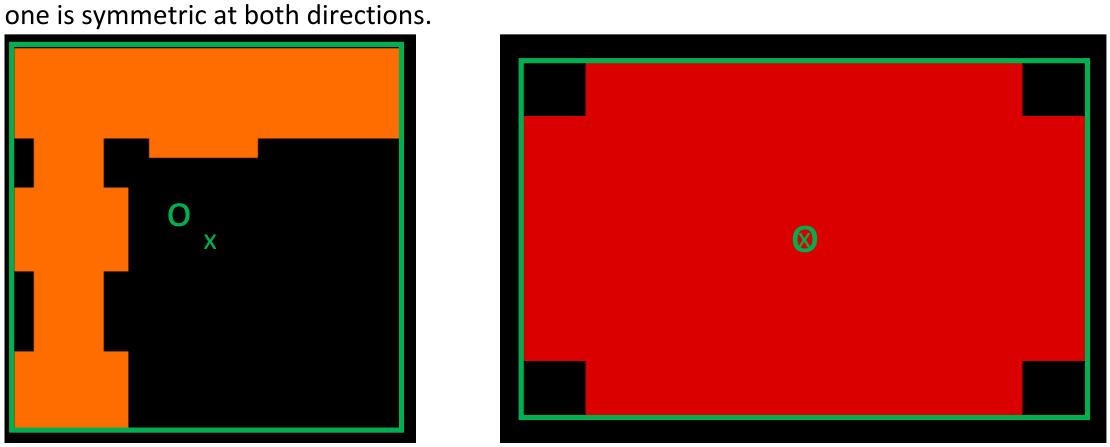
Wentao Jiang (wj2227)

Step 1. Basic infrastructure and building features and descriptions: the "what"

In this step, besides commonly seen descriptor of regions (including centroid, area, bounding box), additional descriptors of buildings are listed below:

(1) Small/medium/large: 27 buildings with area from small to large in order are divided into three equal division, each has 9 buildings. The smallest 1/3 of buildings are labeled small, and largest 1/3 buildings are labeled large, the rest labeled medium.

(2) SymmetricNorthSouth/SymmetricEastWest: a building is labeled symmetricNorthSouth if it is symmetric with vertical axis in the middle, and labeled symmetricEastWest if it is symmetric with horizontal axis in the middle on the map. Note that a building can be symmetric on both directions, and if a building is not symmetric in these two directions, it is said to be nonSymmetric. It is calculated by if the centroid of the building overlap with the centroid of its filled bounding box. If yes, it is symmetric at that direction. If not, it is not symmetric at that direction, as shown in figure below, left building is not symmetric at both directions, and right one is symmetric at both directions.



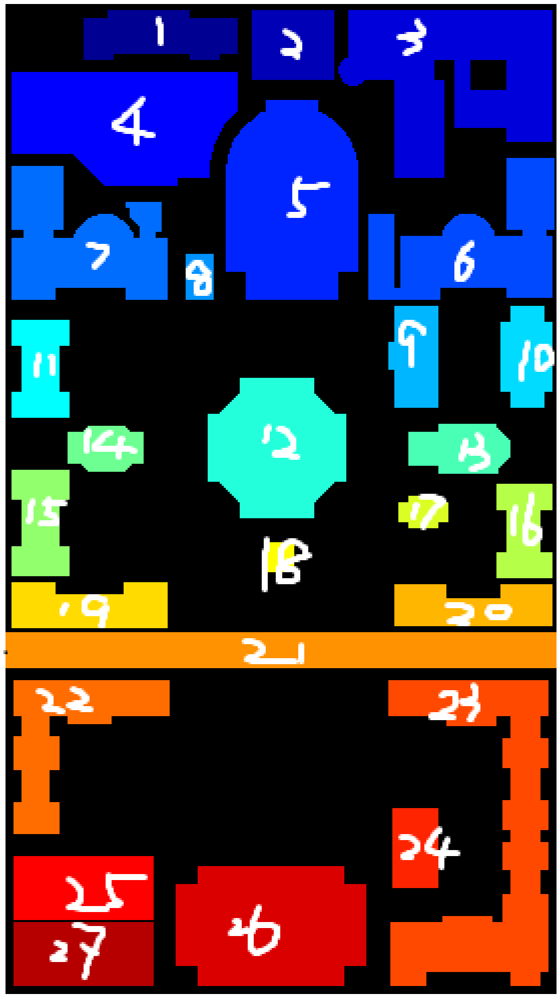
(3) orientedEastWest/orientedNorthSouth: a building is labeled orientedEastWest if the ratio of horizontal width over vertical height of its bounding box is greater than 1.2, or orientedNorthSouth if the ratio of vertical height over horizontal width of its bounding box is greater than 1.2. The reason for choosing 1.2 is because when the ratio is greater than 1.2, it is fairly easy to recognize the orientation with eyes and without any tools.

(4) Rectangle/square/nonRectangle: a building is labeled square if it is a rectangle and the ratio between two adjacent edges is less than 1.05 and greater than 1/1.05. The threshold 1.05 is used because it is found that a few rectangle buildings have very similar height and width (difference of 1). If a building is rectangle but not satisfies the condition of square defined above, it is labeled rectangle. Buildings with other shapes are labeled nonRectangle.

(5) UpperCampus/lowerCampus/WesterCampus/EasterCampus: a building is said to be at upperCampus if its centroid is within upper 1/d of the map, and similar to the definition of other concepts.

Since these selected descriptors can differentiate most of the buildings uniquely, I believe the design is fair for our purpose.

The descriptions of each building are listed below. The numbering is the same as in the text file given, and a figure with number labels are shown below for convenience.



1.
Name: Pupin
Center of Mass: [77,16]
Area: 1640
Bounding Box: [40,4], [117,29]
Description: medium nonSymmetricorientedEastWest nonRectangle upperCampus westerCampus

2.
Name: Schapiro CEPSR
Center of Mass: [144,21]
Area: 1435
Bounding Box: [124,4], [165,39]

Description: medium symmetricNorthSouth symmetricEastWest rectangle upperCampus

3.
Name: Mudd, Engineering Terrace, Fairchild & Computer Science
Center of Mass: [224,36]
Area: 5831
Bounding Box: [167,4], [274,88]
Description: large nonSymmetricorientedEastWest nonRectangle upperCampus easterCampus

4.
Name: Physical Fitness Center
Center of Mass: [60,59]
Area: 5368
Bounding Box: [4,35], [117,92]
Description: large nonSymmetricorientedEastWest nonRectangle upperCampus westerCampus

5.
Name: Gymnasium & Uris
Center of Mass: [144,100]
Area: 5753
Bounding Box: [111,49], [177,149]
Description: large nonSymmetricorientedNorthSouth nonRectangle upperCampus

6.
Name: Schermerhorn
Center of Mass: [234,122]
Area: 3911
Bounding Box: [182,78], [275,149]
Description: large nonSymmetricorientedEastWest nonRectangle upperCampus easterCampus

7.
Name: Chandler & Havemeyer
Center of Mass: [39,121]
Area: 3613
Bounding Box: [4,82], [82,149]
Description: large nonSymmetricnonRectangle upperCampus westerCampus

8.
Name: Computer Center
Center of Mass: [98,137]
Area: 322
Bounding Box: [91,126], [105,149]
Description: small symmetricNorthSouth symmetricEastWest orientedNorthSouth rectangle upperCampus

9.
Name: Avery
Center of Mass: [205,177]

Area: 1164
Bounding Box: [192,152], [217,203]
Description: small nonSymmetricorientedNorthSouth nonRectangle easterCampus

10.
Name: Fayerweather
Center of Mass: [261,177]
Area: 1182
Bounding Box: [248,152], [274,203]
Description: small symmetricNorthSouth orientedNorthSouth nonRectangle easterCampus

11.
Name: Mathematics
Center of Mass: [18,183]
Area: 1191
Bounding Box: [4,159], [33,208]
Description: medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus

12.
Name: Low Library
Center of Mass: [136,223]
Area: 3898
Bounding Box: [102,188], [171,258]
Description: large symmetricNorthSouth symmetricEastWest nonRectangle

13.
Name: St. Paul's Chapel
Center of Mass: [228,223]
Area: 1087
Bounding Box: [202,211], [253,236]
Description: small nonSymmetricorientedEastWest nonRectangle easterCampus

14.
Name: Earl Hall
Center of Mass: [51,223]
Area: 759
Bounding Box: [32,212], [70,235]
Description: small nonSymmetricorientedEastWest nonRectangle westerCampus

15.
Name: Lewisohn
Center of Mass: [18,260]
Area: 1307
Bounding Box: [4,234], [33,287]
Description: medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus

16.
Name: Philosophy
Center of Mass: [259,264]
Area: 1085
Bounding Box: [246,241], [274,288]
Description: small symmetricNorthSouth orientedNorthSouth nonRectangle easterCampus

17.
Name: Buell & Maison Francaise
Center of Mass: [209,255]
Area: 340
Bounding Box: [197,247], [222,263]
Description: small symmetricNorthSouth symmetricEastWest orientedEastWest nonRectangle easterCampus

18.
Name: Alma Mater
Center of Mass: [137,277]
Area: 225
Bounding Box: [130,270], [145,285]
Description: small symmetricNorthSouth symmetricEastWest square

19.
Name: Dodge
Center of Mass: [43,302]
Area: 1590
Bounding Box: [4,290], [82,313]
Description: medium symmetricEastWest orientedEastWest nonRectangle westerCampus

20.
Name: Kent
Center of Mass: [234,302]
Area: 1470
Bounding Box: [195,291], [274,312]
Description: medium symmetricEastWest orientedEastWest nonRectangle easterCampus

21.
Name: College Walk
Center of Mass: [138,324]
Area: 4950
Bounding Box: [1,315], [276,333]
Description: large symmetricNorthSouth symmetricEastWest orientedEastWest rectangle

22.
Name: Journalism & Furnald
Center of Mass: [32,365]
Area: 2615
Bounding Box: [5,339], [83,416]

Description: medium nonSymmetricnonRectangle lowerCampus westerCampus

23.
Name: Hamilton, Hartley, Wallach & John Jay
Center of Mass: [241,418]
Area: 5855
Bounding Box: [192,339], [272,492]
Description: large nonSymmetricorientedNorthSouth nonRectangle lowerCampus easterCampus

24.
Name: Lion's Court
Center of Mass: [205,423]
Area: 920
Bounding Box: [194,403], [217,443]
Description: small symmetricNorthSouth symmetricEastWest orientedNorthSouth rectangle lowerCampus easterCampus

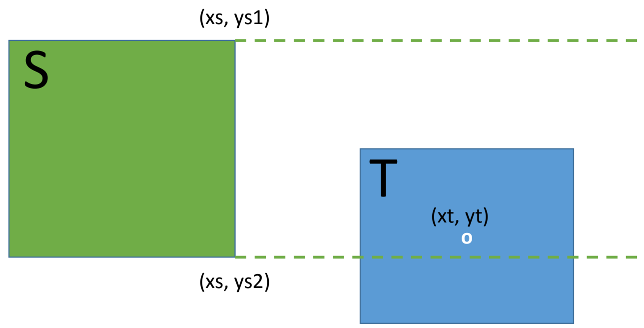
25.
Name: Lerner Hall
Center of Mass: [40,443]
Area: 2240
Bounding Box: [5,427], [75,459]
Description: medium symmetricNorthSouth symmetricEastWest orientedEastWest rectangle lowerCampus westerCampus

26.
Name: Butler Library
Center of Mass: [133,461]
Area: 5282
Bounding Box: [86,432], [181,492]
Description: large symmetricEastWest orientedEastWest nonRectangle lowerCampus

27.
Name: Carman
Center of Mass: [40,476]
Area: 2240
Bounding Box: [5,460], [75,492]
Description: medium symmetricNorthSouth symmetricEastWest orientedEastWest rectangle lowerCampus westerCampus

Step 2. Describing compact spatial relations: the "where".

For this part, East(S, T) is defined to be true if the following conditions are met: assume building T has centroid (xt, yt), building S has bounding box with upper right corner (xs, ys1) and lower right corner (xs, ys2). So East(S, T) is true if xt > xs and yt > ys1 and yt < ys2. As shown below:

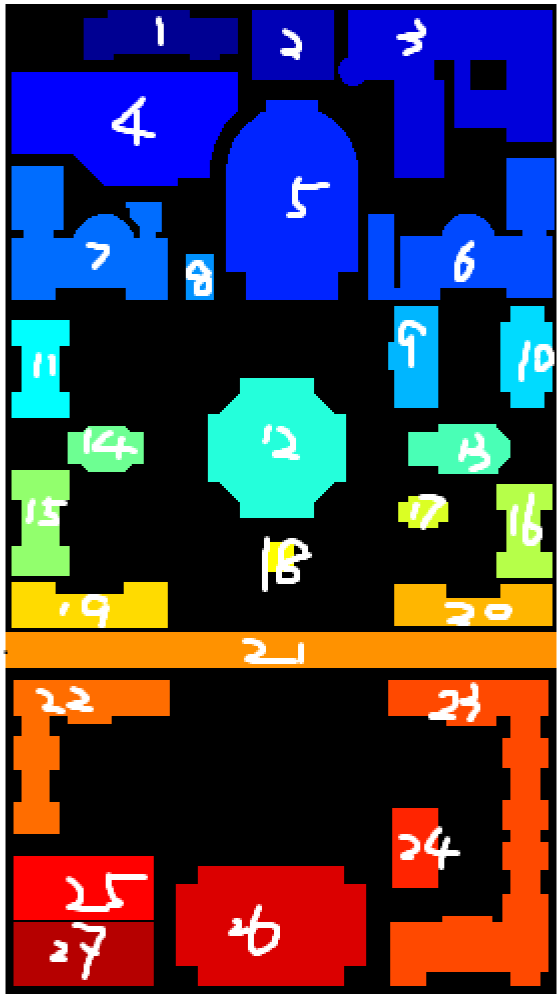


Similar definition apply to West(S, T), North(S, T), South(S, T). Note that it may not be symmetric for cases, i.e. it is possible that East(S, T) is true and East(T, S) is false or vice versa. The purpose to make relatively strict definition of direction is to prevent the ambiguity of jumping relationships between buildings or points, which were seen during the initially less strict design during experiments.

For the calculation of Near(S, T) function, each building is firstly dilated with half of the square root of its area. All other buildings that have overlap with the dilated area of a building is considered to be near to it. Since it is relevant to size (area) of buildings, it is also non-symmetric between buildings.

In the filtering step, transitive reduction was used to remove some of the relations of East, West, North and South. For example, if East(A, B) is true, East(B, C) is true, then we remove East(A, C) (set to false) since it can be inferred from the former two cases and is considered redundant. Near(S, T) relations were not filtered since it is not transitive relationship.

After filtering, the results of each building relating to other buildings are output as below. As reference, a map is attached:



1. Pupin is: west of Schapiro CEPSR, north of Physical Fitness Center, near Schapiro CEPSR, Physical Fitness Center, Gymnasium & Uris,

2. Schapiro CEPSR is: east of Pupin, west of Mudd, Engineering Terrace, Fairchild & Computer Science, north of Gymnasium & Uris, near Pupin, Mudd, Engineering Terrace, Fairchild & Computer Science, Physical Fitness Center, Gymnasium & Uris,

3. Mudd, Engineering Terrace, Fairchild & Computer Science is: east of Schapiro CEPSR, Physical Fitness Center, north of Schermerhorn, near Schapiro CEPSR, Gymnasium & Uris, Schermerhorn,

4. Physical Fitness Center is: west of Gymnasium & Uris, north of Chandler & Havemeyer, south of Pupin, near Pupin, Schapiro CEPSR, Gymnasium & Uris, Chandler & Havemeyer,

5. Gymnasium & Uris is: east of Chandler & Havemeyer, west of Schermerhorn, north of Low Library, south of Schapiro CEPSR, near Schapiro CEPSR, Mudd, Engineering Terrace, Fairchild & Computer Science, Physical Fitness Center, Schermerhorn, Chandler & Havemeyer, Computer Center,

6. Schermerhorn is: east of Gymnasium & Uris, north of St. Paul's Chapel, south of Mudd, Engineering Terrace, Fairchild & Computer Science, near Mudd, Engineering Terrace, Fairchild & Computer Science, Gymnasium & Uris, Avery, Fayerweather,

7. Chandler & Havemeyer is: west of Gymnasium & Uris, north of Earl Hall, south of Physical Fitness Center, near Physical Fitness Center, Gymnasium & Uris, Mathematics,

8. Computer Center is: east of Chandler & Havemeyer, west of Gymnasium & Uris, north of College Walk, south of Physical Fitness Center, near Physical Fitness Center, Gymnasium & Uris, Chandler & Havemeyer,

9. Avery is: east of Mathematics, west of Fayerweather, north of St. Paul's Chapel, Buell & Maison Francaise, south of Schermerhorn, near Gymnasium & Uris, Schermerhorn, Low Library, St. Paul's Chapel,

10. Fayerweather is: east of Avery, north of Philosophy, south of Schermerhorn, near Schermerhorn, St. Paul's Chapel,

11. Mathematics is: west of Avery, north of Lewisohn, south of Chandler & Havemeyer, near Chandler & Havemeyer, Earl Hall,

12. Low Library is: east of Earl Hall, west of St. Paul's Chapel, north of Alma Mater, south of Gymnasium & Uris, near

13. St. Paul's Chapel is: east of Low Library, north of Kent, south of Schermerhorn, near Avery, Fayerweather, Low Library, Philosophy,

14. Earl Hall is: west of Low Library, north of Dodge, south of Chandler & Havemeyer, near Mathematics, Lewisohn,

15. Lewisohn is: west of Buell & Maison Francaise, north of Dodge, south of Mathematics, near Earl Hall, Dodge, College Walk,

16. Philosophy is: east of Lewisohn, north of Kent, south of Fayerweather, near St. Paul's Chapel, Kent, College Walk,

17. Buell & Maison Francaise is: east of Low Library, west of Philosophy, north of Kent, Lion's Court, south of Avery, St. Paul's Chapel, near Low Library, St. Paul's Chapel,

18. Alma Mater is: east of Lewisohn, west of Philosophy, north of College Walk, south of Low Library, near Low Library, College Walk,

19. Dodge is: west of Kent, north of Journalism & Furnald, south of Earl Hall, near Lewisohn, College Walk,

20. Kent is: east of Dodge, north of College Walk, Hamilton, Hartley, Wallach & John Jay, south of St. Paul's Chapel, near Philosophy, College Walk, Hamilton, Hartley, Wallach & John Jay,

21. College Walk is: north of Butler Library, south of Alma Mater, near Dodge, Kent, Journalism & Furnald, Hamilton, Hartley, Wallach & John Jay,

22. Journalism & Furnald is: west of Hamilton, Hartley, Wallach & John Jay, north of Lerner Hall, south of Lewisohn, Dodge, College Walk, near College Walk, Lerner Hall,

23. Hamilton, Hartley, Wallach & John Jay is: east of Lion's Court, south of Kent, near College Walk, Lion's Court, Butler Library,

24. Lion's Court is: south of Buell & Maison Francaise, College Walk, near Hamilton, Hartley, Wallach & John Jay, Butler Library,

25. Lerner Hall is: west of Butler Library, north of Carman, south of Journalism & Furnald, near Journalism & Furnald, Butler Library, Carman,

26. Butler Library is: east of Carman, west of Hamilton, Hartley, Wallach & John Jay, south of College Walk, near Hamilton, Hartley, Wallach & John Jay, Lion's Court, Lerner Hall, Carman,

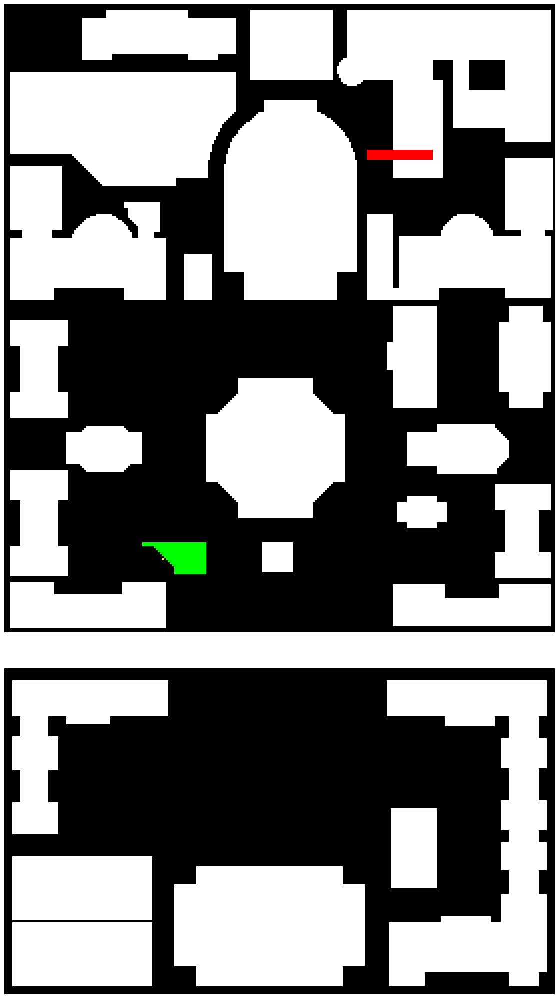
27. Carman is: west of Butler Library, south of Lerner Hall, near Lerner Hall, Butler Library,

Step 3. Source and Target Description and User Interface

In case of the user added points (either start point or target point), they are treated as a one-pixel virtual building with a new label, and all relations are recomputed. After the acquiring of relations between buildings and user added points, the method used to calculate cloud for a certain point on the map is to find the joint of all relationships the point have to other buildings. For example, if East(S, A), South(S, B), Near(S, C) are true, then the cloud is points that also satisfies these constrains. So we find the point set S’ that for S’’ in S’, East(S’’, A) is true, and same for South(S’’, B) and Near(S’’, C). Finally we find the joint area which is the cloud of S point.

Four cases, including the one with experimental largest and smallest cloud, are listed below:

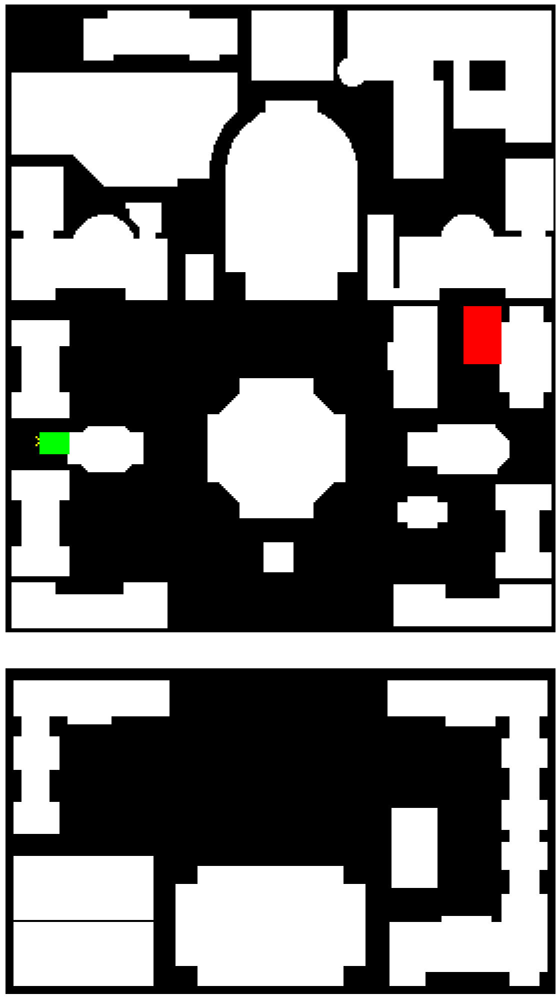
Case 1:



start point is: east of building (medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus), west of building (small symmetricNorthSouth symmetricEastWest square) , north of building (large symmetricNorthSouth symmetricEastWest orientedEastWest rectangle), south of building (large nonSymmetricorientedEastWest nonRectangle upperCampus westerCampus), near building (large symmetricNorthSouth symmetricEastWest nonRectangle), near building (medium symmetricEastWest orientedEastWest nonRectangle westerCampus),

target point is: east of building (large nonSymmetric orientedEastWest nonRectangle upperCampus westerCampus), east of building (large nonSymmetric orientedNorthSouth nonRectangle upperCampus), north of building (large nonSymmetric orientedEastWest nonRectangle upperCampus easterCampus), near building (large nonSymmetric orientedEastWest nonRectangle upperCampus easterCampus), near building (large nonSymmetric orientedNorthSouth nonRectangle upperCampus), near building (large nonSymmetric orientedEastWest nonRectangle upperCampus easterCampus),

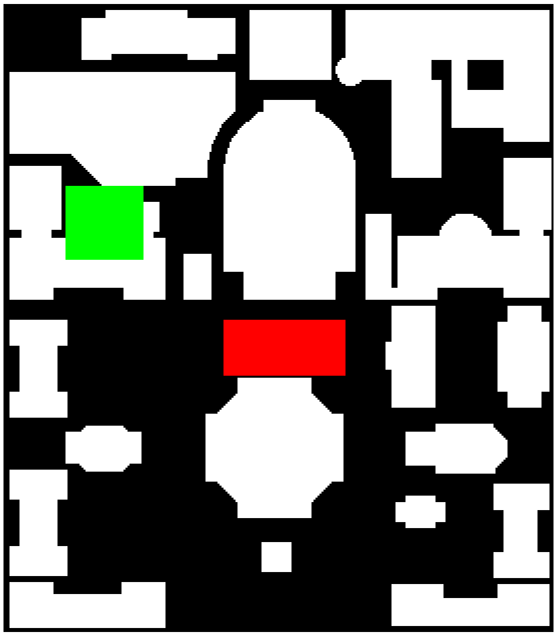
Case 2:

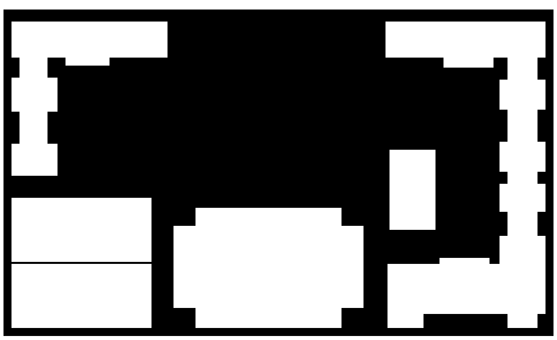


start point is: west of building (small nonSymmetric orientedEastWest nonRectangle westerCampus),north of building (medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus),south of building (medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus),near building (medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus),near building (small nonSymmetricorientedEastWest nonRectangle westerCampus),near building (medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus),

target point is: east of building (small nonSymmetric orientedNorthSouth nonRectangle easterCampus),west of building (small symmetricNorthSouth orientedNorthSouth nonRectangle easterCampus),north of building (small nonSymmetric orientedEastWest nonRectangle easterCampus),south of building (large nonSymmetric orientedEastWest nonRectangle upperCampus easterCampus),near building (large nonSymmetric orientedEastWest nonRectangle upperCampus easterCampus),near building (small symmetricNorthSouth orientedNorthSouth nonRectangle easterCampus),

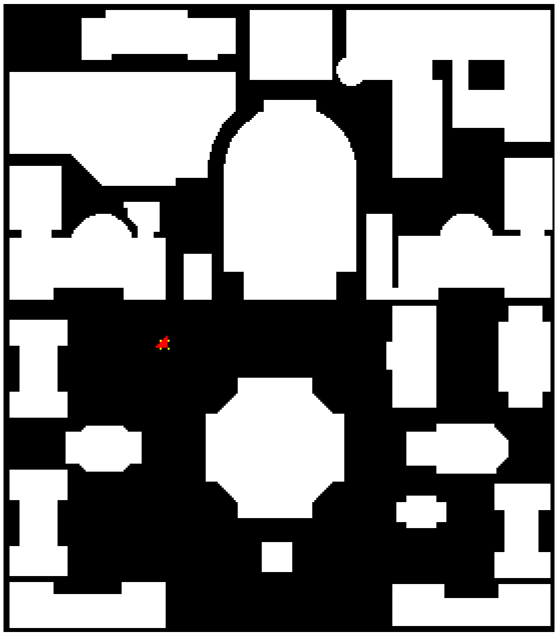
Case 3:

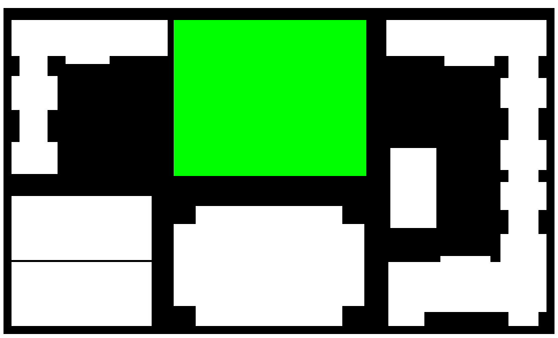




start point is: west of building (large nonSymmetric orientedNorthSouth nonRectangle upperCampus),north of building (small nonSymmetric orientedEastWest nonRectangle westerCampus),south of building (large nonSymmetric orientedEastWest nonRectangle upperCampus westerCampus),near building (large nonSymmetric orientedEastWest nonRectangle upperCampus westerCampus),near building (large nonSymmetric nonRectangle upperCampus westerCampus),

target point is: east of building (medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus),west of building (small nonSymmetric orientedNorthSouth nonRectangle easterCampus),north of building (large symmetricNorthSouth symmetricEastWest nonRectangle),south of building (large nonSymmetricorientedNorthSouth nonRectangle upperCampus),near building (large nonSymmetricorientedNorthSouth nonRectangle upperCampus),near building (large symmetricNorthSouth symmetricEastWest nonRectangle), Case 4: Largest and smallest cloud (Experimental)





start point is: east of building (medium nonSymmetric nonRectangle lowerCampus westerCampus),west of building (large nonSymmetric orientedNorthSouth nonRectangle lowerCampus easterCampus),north of building (large symmetricEastWest orientedEastWest nonRectangle lowerCampus),south of building (large symmetricNorthSouth symmetricEastWest orientedEastWest rectangle),

target point is: east of building (medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus),west of building (small nonSymmetricorientedNorthSouth nonRectangle easterCampus),north of building (medium symmetricEastWest orientedEastWest nonRectangle westerCampus),south of building (large nonSymmetricnonRectangle upperCampus westerCampus),near building (large nonSymmetric orientedNorthSouth nonRectangle upperCampus),near building (large nonSymmetric nonRectangle upperCampus westerCampus),near building (large symmetricNorthSouth symmetricEastWest nonRectangle),

Step 4. Creativity: Path Generation

Algorithm used to calculate the shortest path from start point to end point is Dijkstra's algorithm. By setting length of edges of related buildings or virtual buildings (any of East, West, South, North or Near function) to be the distance between centroids, we build a graph with 27+2 vertices. Since the user added point has area of 1, it is nearly impossible to have any of the functions (East(start, P), West(start, P), North(start, P), South(start, P), Near(start, P)) equal true due to the definition of these functions, the relations between buildings are made symmetric to prevent the generation of non-connected graph.

To find the path from start point to the target, at the beginning the distance of all centroids are set to very large number except the start point. Then each time a point with smallest distance value is chosen to discover and update the distance of connected point as normal Dijkstra's algorithm. After iterating all nodes, we back track from the target point and recover the shortest possible path from the start point to the target point.

The path generated are compared with human response according to the text generated by the system, and cases are listed below:

Case 1:

Description:

start point is: east of building (large nonSymmetric orientedNorthSouth nonRectangle upperCampus),north of building (small nonSymmetric orientedNorthSouth nonRectangle easterCampus),south of building (large nonSymmetric orientedEastWest nonRectangle upperCampus easterCampus),near building (large nonSymmetric orientedEastWest nonRectangle upperCampus easterCampus),near building (large nonSymmetric orientedNorthSouth nonRectangle upperCampus),near building (large nonSymmetric orientedEastWest nonRectangle upperCampus easterCampus),

target point is: west of building (large nonSymmetric orientedNorthSouth nonRectangle lowerCampus easterCampus),north of building (medium symmetricNorthSouth symmetricEastWest orientedEastWest rectangle lowerCampus westerCampus),south of building (medium symmetricEastWest orientedEastWest nonRectangle westerCampus),south of building (large symmetricNorthSouth symmetricEastWest orientedEastWest rectangle),near building (medium nonSymmetric nonRectangle lowerCampus westerCampus),

go to south (small nonSymmetric orientedNorthSouth nonRectangle easterCampus)

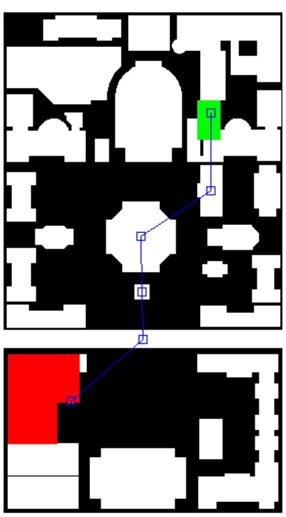
go to near (large symmetricNorthSouth symmetricEastWest nonRectangle)

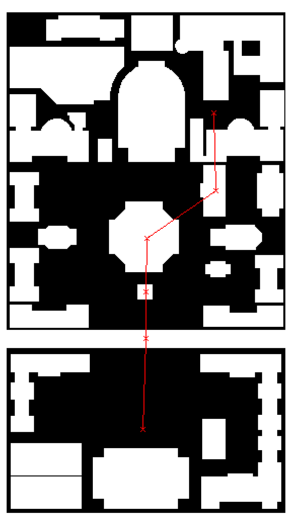
go to near south (small symmetricNorthSouth symmetricEastWest square)

go to near south (large symmetricNorthSouth symmetricEastWest orientedEastWest rectangle)

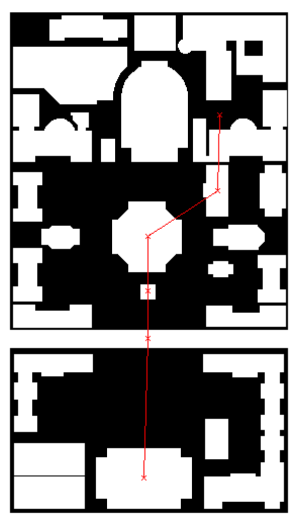
go to south

Planned Path/Friend1/Friend2:





X



X

Case 2:

Description:

start point is: east of building (medium symmetricNorthSouth symmetricEastWest orientedEastWest rectangle lowerCampus westerCampus),east of building (large symmetricEastWest orientedEastWest nonRectangle lowerCampus),south of building (small symmetricNorthSouth symmetricEastWest orientedNorthSouth rectangle lowerCampus easterCampus),near building (large nonSymmetric orientedNorthSouth nonRectangle lowerCampus easterCampus),near building (small symmetricNorthSouth symmetricEastWest orientedNorthSouth rectangle lowerCampus easterCampus),near building (large symmetricEastWest orientedEastWest nonRectangle lowerCampus),

target point is: west of building (small nonSymmetric orientedEastWest nonRectangle westerCampus),north of building (medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus),south of building (medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus),near building (medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus),near building (medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus),

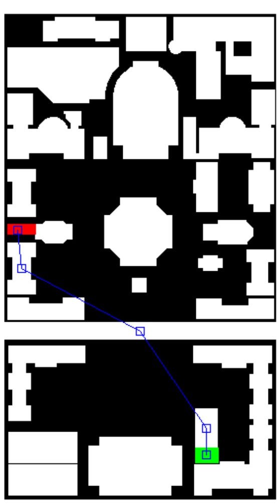
go to near north (small symmetricNorthSouth symmetricEastWest orientedNorthSouth rectangle lowerCampus easterCampus)

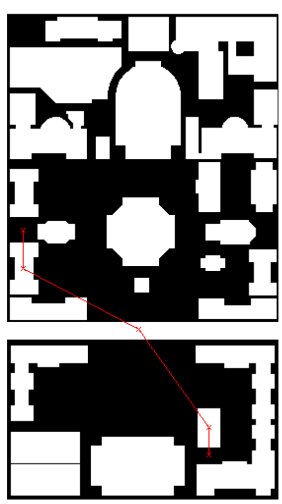
go to north (large symmetricNorthSouth symmetricEastWest orientedEastWest rectangle)

go to near (medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus)

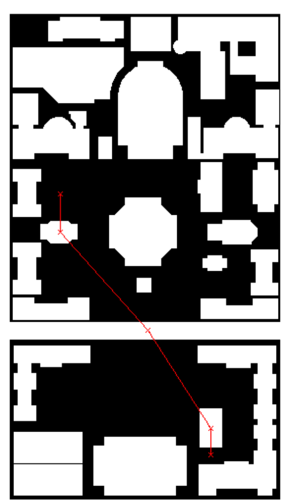
go to near north

Planned Path/Friend1/Friend2:





√



X

Case 3:

Description:

start point is: west of building (medium nonSymmetric orientedEastWest nonRectangle upperCampus westerCampus),north of building (large nonSymmetric orientedEastWest nonRectangle upperCampus westerCampus),north of building (medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus),near building (large nonSymmetric orientedEastWest nonRectangle upperCampus westerCampus),

target point is: east of building (small nonSymmetric orientedNorthSouth nonRectangle easterCampus),west of building (small symmetricNorthSouth orientedNorthSouth nonRectangle easterCampus),north of building (small nonSymmetric orientedEastWest nonRectangle easterCampus),south of building (large nonSymmetric orientedEastWest nonRectangle upperCampus easterCampus),near building (large nonSymmetric orientedEastWest nonRectangle upperCampus easterCampus),near building (small nonSymmetric orientedNorthSouth nonRectangle easterCampus),near building (small symmetricNorthSouth orientedNorthSouth nonRectangle easterCampus),

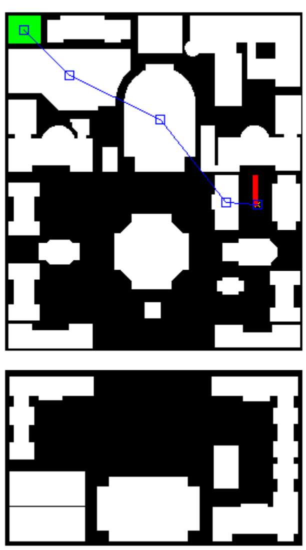
go to near south (large nonSymmetric orientedEastWest nonRectangle upperCampus westerCampus)

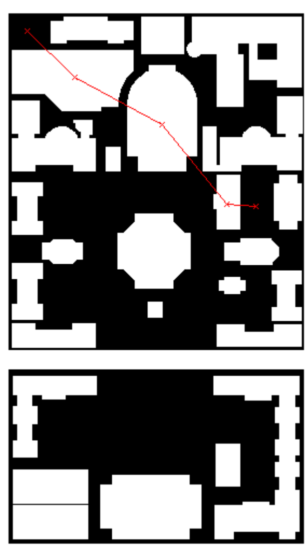
go to near east (large nonSymmetric orientedNorthSouth nonRectangle upperCampus)

go to near (small nonSymmetric orientedNorthSouth nonRectangle easterCampus)

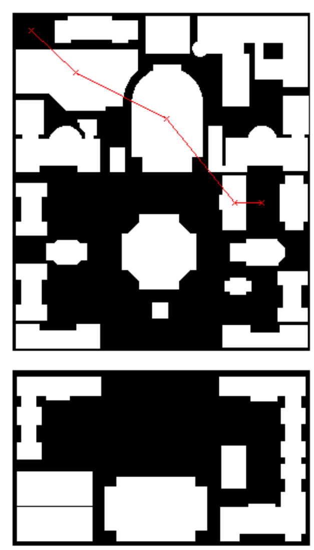
go to near east

Planned Path/Friend1/Friend2:





√



√

Case 4:

Description:

start point is: east of building (medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus),west of building (small symmetricNorthSouth symmetricEastWest orientedEastWest nonRectangle easterCampus),north of building (medium symmetricEastWest orientedEastWest nonRectangle westerCampus),south of building (small nonSymmetric orientedEastWest nonRectangle westerCampus),

target point is: east of building (small symmetricNorthSouth symmetricEastWest square),west of building (small symmetricNorthSouth orientedNorthSouth nonRectangle easterCampus),north of building (medium symmetricEastWest orientedEastWest nonRectangle easterCampus),south of building (small nonSymmetric orientedEastWest nonRectangle easterCampus),near building (small symmetricNorthSouth orientedNorthSouth nonRectangle easterCampus),near building (medium symmetricEastWest orientedEastWest nonRectangle easterCampus),near building (large symmetricNorthSouth symmetricEastWest orientedEastWest rectangle),

go to east (small symmetricNorthSouth symmetricEastWest orientedEastWest nonRectangle easterCampus)

go to south (medium symmetricEastWest orientedEastWest nonRectangle easterCampus)

go to near north

Planned Path/Friend1/Friend2:

√ √

Case 5:

Description:

start point is: east of building (small symmetricNorthSouth symmetricEastWest orientedNorthSouth rectangle lowerCampus easterCampus),south of building (medium symmetricEastWest orientedEastWest nonRectangle easterCampus),near building (large nonSymmetric orientedNorthSouth nonRectangle lowerCampus easterCampus),near building (small symmetricNorthSouth symmetricEastWest orientedNorthSouth rectangle lowerCampus easterCampus),

target point is: east of building (medium symmetricNorthSouth symmetricEastWest orientedNorthSouth nonRectangle westerCampus),west of building (small nonSymmetric orientedNorthSouth nonRectangle easterCampus),north of building (small nonSymmetric orientedEastWest nonRectangle westerCampus),south of building (large nonSymmetric nonRectangle upperCampus westerCampus),

go to near west (small symmetricNorthSouth symmetricEastWest orientedNorthSouth rectangle lowerCampus easterCampus)

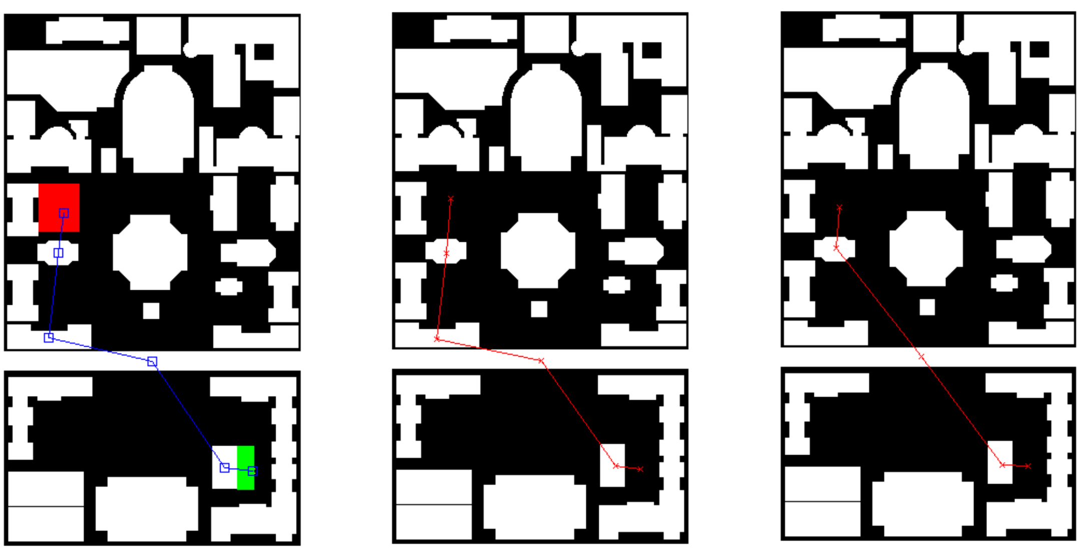
go to north (large symmetricNorthSouth symmetricEastWest orientedEastWest rectangle)

go to near (medium symmetricEastWest orientedEastWest nonRectangle westerCampus)

go to north (small nonSymmetric orientedEastWest nonRectangle westerCampus)

go to north

Planned Path/Friend1/Friend2:



√ √

Case 6:

Description:

start point is: east of building (medium symmetricEastWest symmetricNorthSouth orientedEastWest rectangle lowerCampus westerCampus),west of building (small symmetricEastWest symmetricNorthSouth orientedNorthSouth rectangle lowerCampus easterCampus),west of building (large symmetricEastWest orientedEastWest nonRectangle lowerCampus),south of building (medium nonSymmetric nonRectangle lowerCampus westerCampus),near building (medium symmetricEastWest symmetricNorthSouth orientedEastWest rectangle lowerCampus westerCampus),near building (large symmetricEastWest orientedEastWest nonRectangle lowerCampus),

target point is: east of building (medium symmetricEastWest symmetricNorthSouth orientedNorthSouth nonRectangle westerCampus),west of building (small nonSymmetric orientedNorthSouth nonRectangle easterCampus),north of building (large symmetricEastWest symmetricNorthSouth orientedEastWest rectangle),south of building (large nonSymmetric orientedNorthSouth nonRectangle upperCampus),near building (large nonSymmetric orientedNorthSouth nonRectangle upperCampus),near building (large nonSymmetric orientedEastWest nonRectangle upperCampus easterCampus),near building (small nonSymmetric orientedNorthSouth nonRectangle easterCampus),near building (large symmetricEastWest symmetricNorthSouth nonRectangle),

go to near east (large symmetricEastWest orientedEastWest nonRectangle lowerCampus)

go to north (large symmetricEastWest symmetricNorthSouth orientedEastWest rectangle)

go to north

Planned Path/Friend1/Friend2:

Case 7:

Description:

start point is: east of building (medium nonSymmetric nonRectangle lowerCampus westerCampus),west of building (large nonSymmetric orientedNorthSouth nonRectangle lowerCampus easterCampus),north of building (large symmetricEastWest orientedEastWest nonRectangle lowerCampus),south of building (large symmetricEastWest symmetricNorthSouth orientedEastWest rectangle),

target point is: east of building (large nonSymmetric orientedNorthSouth nonRectangle upperCampus),north of building (small nonSymmetric orientedNorthSouth nonRectangle easterCampus),south of building (large nonSymmetric orientedEastWest nonRectangle upperCampus easterCampus),near building (large nonSymmetric orientedEastWest nonRectangle upperCampus easterCampus),near building (large nonSymmetric orientedNorthSouth nonRectangle upperCampus),near building (large nonSymmetric orientedEastWest nonRectangle upperCampus easterCampus),

go to north (large symmetricEastWest symmetricNorthSouth orientedEastWest rectangle)

go to near north (small symmetricEastWest symmetricNorthSouth square)

go to near north (large symmetricEastWest symmetricNorthSouth nonRectangle)

go to near (small nonSymmetric orientedNorthSouth nonRectangle easterCampus)

go to north

Planned Path/Friend1/Friend2:

√ x

Case 8:

Description:

start point is: east of building (medium symmetricEastWest orientedEastWest nonRectangle westerCampus),west of building (medium symmetricEastWest orientedEastWest nonRectangle easterCampus),north of building (large symmetricEastWest symmetricNorthSouth orientedEastWest rectangle),south of building (small symmetricEastWest symmetricNorthSouth orientedNorthSouth rectangle upperCampus),near building (medium symmetricEastWest orientedEastWest nonRectangle westerCampus),near building (large symmetricEastWest symmetricNorthSouth orientedEastWest rectangle),

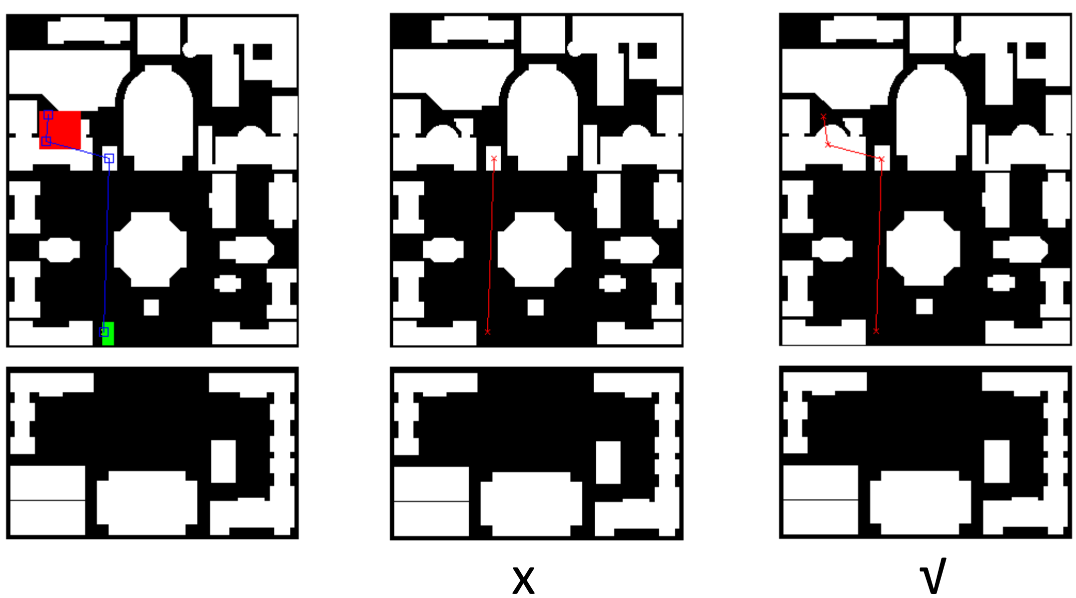
target point is: west of building (large nonSymmetric orientedNorthSouth nonRectangle upperCampus),north of building (small nonSymmetric orientedEastWest nonRectangle westerCampus),south of building (large nonSymmetric orientedEastWest nonRectangle upperCampus westerCampus),near building (large nonSymmetric orientedEastWest nonRectangle upperCampus westerCampus),near building (large nonSymmetric nonRectangle upperCampus westerCampus),

go to north (small symmetricEastWest symmetricNorthSouth orientedNorthSouth rectangle upperCampus)

go to near west (large nonSymmetric nonRectangle upperCampus westerCampus)

go to near

Planned Path/Friend1/Friend2:



We could see that the accuracy is 9/16 = 56.25%. Sometimes near doesn't provide too much useful information. The overall performance is good, and it can usually give the right direction combining the 'what' and 'where' information.

Appendix:

Hw3.m

```
function hw3
    close all;
    map = imread('ass3-campus.pgm');
    global map_labeled;
    map_labeled = imread('ass3-labeled.pgm');
    map_labeled = relabel(map_labeled);
    map2 = label2rgb(map_labeled, 'jet', 'k');
    figure;imshow(map2);
    %figure;imshow(map_labeled);

    % part 1
    global database;
    database = gen_prop();
    print_part1();

    % part 2
    relation = gen_spacial();
    print_part2(relation);

    % part 3
    fig = figure(); imshow(map)
    % get points
    disp('set start point (double click):');
    [y, x] = getpts(fig);
    start = [round(x), round(y)];
    hold on;
    plot(y, x, 'yx');
    disp('target point (double click):');
    [y, x] = getpts(fig);
    target = [round(x), round(y)];
    plot(y, x, 'yx');
    % plot start cloud
    map_labeled_rep = map_labeled;
    map_labeled(start(1), start(2)) = 28;
    database_rep = database;
    database = gen_prop_re();
    relation = gen_spacial();
    database = database_rep;
    list = print_info(relation, 'start point', 28);
    mask = gen_cloud(list);
    [row, col] = find(mask==1);
    plot(col, row, 'gs','MarkerSize',1);
    % plot target cloud
    map_labeled = map_labeled_rep;
    map_labeled(target(1), target(2)) = 28;
    database = gen_prop_re();
    relation = gen_spacial();
    database = database_rep;
    list = print_info(relation, 'target point', 28);
    mask2 = gen_cloud(list);
```

```
    [row, col] = find(mask2==1);
    plot(col, row, 'rs','MarkerSize',1);

    % part 4
    map_labeled = map_labeled_rep;
    map_labeled(start(1), start(2)) = 28;
    map_labeled(target(1), target(2)) = 29;
    database = gen_prop_re();
    relation = gen_spacial();
    database_rep2 = database;
    database = database_rep;
    print_info(relation, 'start point', 28);
    print_info(relation, 'target point', 29);
    database = database_rep2;
    %%%% generate path
    path = gen_path(relation);
    cent = [];
    for i = 1:length(path)
        cent = [cent; database.Centroid(path(i),:)];
    end
    plot(cent(:,1), cent(:,2),'bs-');
    database = database_rep;
    print_description(relation, path);

    %%%% user path
    fig2 = figure(); imshow(map);
    hold on;
    plot(start(2), start(1), 'yx');
    [y,x] = getpts(fig2);
    y = [start(2);y];
    x = [start(1);x];
    plot(y, x, 'rx-');

    fig3 = figure(); imshow(map);
    hold on;
    plot(start(2), start(1), 'yx');
    [y,x] = getpts(fig3);
    y = [start(2);y];
    x = [start(1);x];
    plot(y, x, 'rx-');
    a=1;
end

function print_description(rel, path)
    for i=1:length(path)-1
        step = 'go to ';
        if rel.rel_near(path(i),path(i+1))==1 ||
rel.rel_near(path(i+1),path(i))==1
            step = [step, 'near '];
        end
        if rel.rel_east(path(i),path(i+1))==1 ||
rel.rel_west(path(i+1),path(i))==1
            step = [step, 'east '];
        end
        if rel.rel_west(path(i),path(i+1))==1 ||
rel.rel_east(path(i+1),path(i))==1
```

```
            step = [step, 'west '];
        end
        if rel.rel_north(path(i),path(i+1))==1 ||
rel.rel_south(path(i+1),path(i))==1
            step = [step, 'north '];
        end
        if rel.rel_south(path(i),path(i+1))==1 ||
rel.rel_north(path(i+1),path(i))==1
            step = [step, 'south '];
        end
        if i+1<length(path) && i>=1
            step = [step, '(', print_building(path(i+1)), ')'];
        end
        disp (step)
    end
end


function path = gen_path(relation)
    global database;
    rel = (relation.rel_east | relation.rel_west | relation.rel_north |
relation.rel_south | relation.rel_near);
    rel = rel | rel';
    len = size(rel,1);
    edge = ones(len,len)*100000;
    for i = 1:len
        for j = 1:len
            if rel(i,j)==1
                edge(i,j) = norm(database.Centroid(i,:) -
database.Centroid(j,:));
            end
        end
    end
    dist = ones(len,1)*100000;
    visited = zeros(len, 1);
    track = zeros(len, 1);
    ind = 28;
    dist(ind) = 0;
    for i = 1:len-1
        for j = 1:len
            if edge(ind, j)+dist(ind)<dist(j)
                dist(j) = edge(ind, j)+dist(ind);
                track(j) = ind;
            end
        end
        visited(ind) = 1;
        ind = find(dist==min(dist(~visited)));
    end
    ind = 29;
    path = ind;
    while ind~=28
        path = [track(ind);path];
        ind = (track(ind));
    end
    a = 1;
```

```
end


function mask = gen_cloud(list)
    global database;
    global bw_dilate;
    mask = ones(495,275);
    if length(list.e)>0
        for i = 1:length(list.e)
            m = zeros(495, 275);
            bb = uint16(database.BoundingBox(list.e(i),:));
            m(bb(2):bb(2)+bb(4), min(bb(1)+bb(3),275):275) = 1;
            mask = mask & m;
        end
    end
    if length(list.w)>0
        for i = 1:length(list.w)
            m = zeros(495, 275);
            bb = uint16(database.BoundingBox(list.w(i),:));
            m(bb(2):bb(2)+bb(4), 1:bb(1)) = 1;
            mask = mask & m;
        end
    end
    if length(list.n)>0
        for i = 1:length(list.n)
            m = zeros(495, 275);
            bb = uint16(database.BoundingBox(list.n(i),:));
            m(1:bb(2), bb(1):min(bb(1)+bb(3),275)) = 1;
            mask = mask & m;
        end
    end
    if length(list.s)>0
        for i = 1:length(list.s)
            m = zeros(495, 275);
            bb = uint16(database.BoundingBox(list.s(i),:));
            m(bb(2)+bb(4):495, bb(1):min(bb(1)+bb(3),275)) = 1;
            mask = mask & m;
        end
    end
    if length(list.near)>0
        for i = 1:length(list.near)
            mask = mask & bw_dilate{list.near(i)};
        end
    end
end

function list = print_info(relation, name, num)
    list = struct('e',[], 'w',[], 'n',[], 's',[], 'near',[]);
    for j=1:size(relation.rel_east,1)
        if relation.rel_east(j,num)==1
            list.e = [list.e, j];
        end
        if relation.rel_west(j,num)==1
            list.w = [list.w, j];
        end
        if relation.rel_north(j,num)==1
```

```
            list.n = [list.n, j];
        end
        if relation.rel_south(j,num)==1
            list.s = [list.s, j];
        end
        if relation.rel_near(j,num)==1
            list.near = [list.near, j];
        end
    end
    str = sprintf('%s is: ', name);
    if (length(list.e)~=0)
        for i = 1:length(list.e)
            str = [str, 'east of building (',
print_building(list.e(i)),'),'];
        end
    end
    if (length(list.w)~=0)
        for i = 1:length(list.w)
            str = [str, 'west of building (',
print_building(list.w(i)),'),'];
        end
    end
    if (length(list.n)~=0)
        for i = 1:length(list.n)
            str = [str, 'north of building (',
print_building(list.n(i)),'),'];
        end
    end
    if (length(list.s)~=0)
        for i = 1:length(list.s)
            str = [str, 'south of building (',
print_building(list.s(i)),'),'];
        end
    end
    if (length(list.near)~=0)
        for i = 1:length(list.near)
            str = [str, 'near building (',
print_building(list.near(i)),'),'];
        end
    end
    disp(str);
end

function str = add_name(lst)
    global database;
    str = '';
    for i = 1:length(lst)
        str = [str, database.Name{lst(i)}, ', '];
    end
end

function database = gen_prop_re()
    global map_labeled;
    database = regionprops(map_labeled, 'centroid','area', 'boundingbox');
    database = struct2dataset(database);
    database = getname(database);
```

```
end

function map_labeled = relabel(map_labeled)
    count = 1;
    for i=1:255
        ind = find(map_labeled==i);
        if ~isempty(ind)
            map_labeled(ind)=count;
            count = count+1;
        end
    end
end

function database = getname(database)
    name = {'Pupin';
            'Schapiro CEPSR';
            'Mudd, Engineering Terrace, Fairchild & Computer Science';
            'Physical Fitness Center';
            'Gymnasium & Uris';
            'Schermerhorn';
            'Chandler & Havemeyer';
            'Computer Center';
            'Avery';
            'Fayerweather';
            'Mathematics';
            'Low Library';
            'St. Paul''s Chapel';
            'Earl Hall';
            'Lewisohn';
            'Philosophy';
            'Buell & Maison Francaise';
            'Alma Mater';
            'Dodge';
            'Kent';
            'College Walk';
            'Journalism & Furnald';
            'Hamilton, Hartley, Wallach & John Jay';
            'Lion''s Court';
            'Lerner Hall';
            'Butler Library';
            'Carman'};
    database.Name = name;
end
```

gen_prop.m

```
function database = gen_prop()
    global map_labeled;

    database = regionprops(map_labeled, 'centroid','area', 'boundingbox');
    database = struct2dataset(database);
```

```
    database = addprops(map_labeled, database);
    database = getname(database);
end

function database = getname(database)
    name = {'Pupin';
            'Schapiro CEPSR';
            'Mudd, Engineering Terrace, Fairchild & Computer Science';
            'Physical Fitness Center';
            'Gymnasium & Uris';
            'Schermerhorn';
            'Chandler & Havemeyer';
            'Computer Center';
            'Avery';
            'Fayerweather';
            'Mathematics';
            'Low Library';
            'St. Paul''s Chapel';
            'Earl Hall';
            'Lewisohn';
            'Philosophy';
            'Buell & Maison Francaise';
            'Alma Mater';
            'Dodge';
            'Kent';
            'College Walk';
            'Journalism & Furnald';
            'Hamilton, Hartley, Wallach & John Jay';
            'Lion''s Court';
            'Lerner Hall';
            'Butler Library';
            'Carman'};
    database.Name = name;
end

function database = addprops(map_labeled, database)
    % database = [Area, Centroid, BoundingBox, Small, Medium, Large,
    % SymmetricEW, SymmetricNS, OrientedEW, OrientedNS, Rect, Square,
    % Upper, Lower, Easter, Wester
    len = size(database,1);
    % small/medium/large
    area_sorted = sort(database.Area);
    area_small = area_sorted(len/3);
    area_large = area_sorted(len/3*2);
    small = [];
    medium = [];
    large = [];
    sym_EW = [];
    sym_NS = [];
    orient_EW = [];
    orient_NS = [];
    rect = [];
    sqr = [];
    upper = [];
    lower = [];
    easter = [];
```

```
wester = [];
for i=1:len
    % small/medium/large
    if database.Area(i) <= area_small
        small = [small; 1];
        medium = [medium; 0];
        large = [large; 0];
    elseif database.Area(i) > area_large
        small = [small; 0];
        medium = [medium; 0];
        large = [large; 1];
    else
        small = [small; 0];
        medium = [medium; 1];
        large = [large; 0];
    end
    % SymmetricEW/SymmetricNS/notSymmetric
    boundbox = database.BoundingBox(i,:);
    cent = database.Centroid(i,:);
    box_x = boundbox(1)+boundbox(3)/2;
    if cent(1) == box_x
        sym_EW = [sym_EW; 1];
    else
        sym_EW = [sym_EW; 0];
    end
    box_y = boundbox(2)+boundbox(4)/2;
    if cent(2) == box_y
        sym_NS = [sym_NS; 1];
    else
        sym_NS = [sym_NS; 0];
    end
    % OrientedEW(x/y>1.2)/OrientedNS(x/y<1/1.2)
    if boundbox(3)/boundbox(4)>=1.2
        orient_EW = [orient_EW; 1];
        orient_NS = [orient_NS; 0];
    elseif boundbox(3)/boundbox(4)<=1/1.2
        orient_EW = [orient_EW; 0];
        orient_NS = [orient_NS; 1];
    else
        orient_EW = [orient_EW; 0];
        orient_NS = [orient_NS; 0];
    end
    % rect / square(1/1.05 <=x/y <= 1.05)
    if boundbox(3)*boundbox(4) == database.Area(i)
        if boundbox(3)/boundbox(4)>=1.05 || boundbox(4)/boundbox(3)>=1.05
            rect = [rect; 1];
            sqr = [sqr; 0];
        else
            rect = [rect; 0];
            sqr = [sqr;1];
        end
    else
        rect = [rect; 0];
        sqr = [sqr; 0];
    end
    % upper/lower/ester/wester
    width = size(map_labeled, 2);
```

```
        length = size(map_labeled, 1);
        upper_line = length/3;
        lower_line = length/3*2;
        wester_line = width/3;
        easter_line = width/3*2;
        if cent(1)>easter_line
            easter = [easter; 1];
            wester = [wester; 0];
        elseif cent(1)<wester_line
            easter = [easter; 0];
            wester = [wester; 1];
        else
            easter = [easter; 0];
            wester = [wester; 0];
        end
        if cent(2)<upper_line
            upper = [upper; 1];
            lower = [lower; 0];
        elseif cent(2)>lower_line
            upper = [upper; 0];
            lower = [lower; 1];
        else
            upper = [upper; 0];
            lower = [lower; 0];
        end
    end
    database.Small = small;
    database.Medium = medium;
    database.Large = large;
    database.SymmetricEW = sym_EW;
    database.SymmetricNS = sym_NS;
    database.OrientedEW = orient_EW;
    database.OrientedNS = orient_NS;
    database.Rectangle = rect;
    database.Square = sqr;
    database.Upper = upper;
    database.Lower = lower;
    database.Easter = easter;
    database.Wester = wester;
end
```

gen_spacial.m

```
function rel = gen_spacial()
    global database;
    len = length(database);
    rel_east = zeros(len, len);
    rel_west = zeros(len, len);
    rel_north = zeros(len, len);
```

```
    rel_south = zeros(len, len);
    rel_near = zeros(len, len);
    global bw_dilate;
    bw_dilate = cell(len, 1);
    gen_bw_dilate();
    for i = 1:length(database)
        cent_s = database.Centroid(i,:);
        for j = 1:length(database)
            cent_t = database.Centroid(j,:);
            rel_east(i,j) = east([cent_s, i], [cent_t,j]);
            rel_west(i,j) = west([cent_s, i], [cent_t,j]);
            rel_north(i,j) = north([cent_s, i], [cent_t,j]);
            rel_south(i,j) = south([cent_s, i], [cent_t,j]);
            rel_near(i,j) = near([cent_s, i], [cent_t,j]);
            if i==j
                rel_east(i,j) = 0;
                rel_west(i,j) = 0;
                rel_north(i,j) = 0;
                rel_south(i,j) = 0;
                rel_near(i,j) = 0;
            end
        end
    end
    %%%%%%%%%%  filter %%%%%%%%
    rel_east = filter(rel_east);
    rel_west = filter(rel_west);
    rel_north = filter(rel_north);
    rel_south = filter(rel_south);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    rel.rel_east = rel_east;
    rel.rel_west = rel_west;
    rel.rel_north = rel_north;
    rel.rel_south = rel_south;
    rel.rel_near = rel_near;
end

function relation = filter(relation)
    global database;
    for i = 1:size(relation,1)
        for j = 1:size(relation,1)
            if relation(i,j)==1
                for k = 1:size(relation,1)
                    if relation(j,k)==1
                        relation(i,k) = 0;
                    end
                end
            end
        end
    end

    relation_1 = relation;
    for i = 1:size(relation, 1)
        min_dist = 100000;
        ind = 0;
        for j = 1:size(relation, 1)
            if relation_1(i,j) == 1
```

```
                dist = norm(database.Centroid(i,:)-database.Centroid(j,:));
                if dist<min_dist
                    min_dist = dist;
                    ind = j;
                end
            end
        end
        if ind~=0
            relation_1(i,:) = 0;
            relation_1(i, ind) = 1;
        end
    end

    relation_2 = relation;
    for j = 1:size(relation, 1)
        min_dist = 100000;
        ind = 0;
        for i = 1:size(relation, 1)
            if relation_2(i,j)==1
                dist = norm(database.Centroid(i,:)-database.Centroid(j,:));
                if dist<min_dist
                    min_dist = dist;
                    ind = i;
                end
            end
        end
        if ind~=0
            relation_2(:,j) = 0;
            relation_2(ind, j) = 1;
        end
    end

    relation = (relation_1 | relation_2);
end

function gen_bw_dilate()
    global map_labeled;
    global bw_dilate;
    global database;
    for i = 1:length(database)
        bw_img = (map_labeled == i);
        area = database.Area(i);
        bw_dilate{i} = bwmorph(bw_img,'dilate', sqrt(area)/2);
    end
end
```

east.m

```
function flag = east(s, t)
    global database;
```

```
    labeled_s = s(3);
    labeled_t = t(3);
    % find bound box and centroid of s and t
    if labeled_s==0
        boundbox_s = [s(1), s(2), 1, 1];
        cent_s = [s(1), s(2)];
    else
        boundbox_s = database.BoundingBox(labeled_s, :);
        cent_s = database.Centroid(labeled_s, :);
    end
    if labeled_t==0
        boundbox_t = [t(1), t(2), 1, 1];
        cent_t = [t(1), t(2)];
    else
        boundbox_t = database.BoundingBox(labeled_t, :);
        cent_t = database.Centroid(labeled_t, :);
    end
    % compute if t at east of s
    if cent_t(1) > boundbox_s(1)+boundbox_s(3) && cent_t(2)>boundbox_s(2) &&
cent_t(2)<boundbox_s(2)+boundbox_s(4)
        flag = 1;
    else
        flag = 0;
    end
end
```

west.m

```
function flag = west(s, t)
    global database;
    labeled_s = s(3);
    labeled_t = t(3);
    % find bound box and centroid of s and t
    if labeled_s==0
        boundbox_s = [s(1), s(2), 1, 1];
        cent_s = [s(1), s(2)];
    else
        boundbox_s = database.BoundingBox(labeled_s, :);
        cent_s = database.Centroid(labeled_s, :);
    end
    if labeled_t==0
        boundbox_t = [t(1), t(2), 1, 1];
        cent_t = [t(1), t(2)];
    else
        boundbox_t = database.BoundingBox(labeled_t, :);
        cent_t = database.Centroid(labeled_t, :);
    end
    % compute if t at west of s
    if cent_t(1) < boundbox_s(1) && cent_t(2)>boundbox_s(2) &&
cent_t(2)<boundbox_s(2)+boundbox_s(4)
        flag = 1;
    else
```

```
        flag = 0;
    end
end
```

north.m

```
function flag = north(s, t)
    global database;
    labeled_s = s(3);
    labeled_t = t(3);
    % find bound box and centroid of s and t
    if labeled_s==0
        boundbox_s = [s(1), s(2), 1, 1];
        cent_s = [s(1), s(2)];
    else
        boundbox_s = database.BoundingBox(labeled_s, :);
        cent_s = database.Centroid(labeled_s, :);
    end
    if labeled_t==0
        boundbox_t = [t(1), t(2), 1, 1];
        cent_t = [t(1), t(2)];
    else
        boundbox_t = database.BoundingBox(labeled_t, :);
        cent_t = database.Centroid(labeled_t, :);
    end
    % compute if t at north of s
    if cent_t(2) < boundbox_s(2) && cent_t(1)>boundbox_s(1) &&
cent_t(1)<boundbox_s(1)+boundbox_s(3)
        flag = 1;
    else
        flag = 0;
    end
end
```

south.m

```
function flag = south(s, t)
    global database;
    labeled_s = s(3);
    labeled_t = t(3);
    % find bound box and centroid of s and t
    if labeled_s==0
        boundbox_s = [s(1), s(2), 1, 1];
        cent_s = [s(1), s(2)];
    else
        boundbox_s = database.BoundingBox(labeled_s, :);
        cent_s = database.Centroid(labeled_s, :);
```

```
    end
    if labeled_t==0
        boundbox_t = [t(1), t(2), 1, 1];
        cent_t = [t(1), t(2)];
    else
        boundbox_t = database.BoundingBox(labeled_t, :);
        cent_t = database.Centroid(labeled_t, :);
    end
    % compute if t at south of s
    if cent_t(2) > boundbox_s(2)+boundbox_s(4) && cent_t(1)>boundbox_s(1) &&
cent_t(1)<boundbox_s(1)+boundbox_s(3)
        flag = 1;
    else
        flag = 0;
    end
end
```

near.m

```
function flag = near(s, t)
    global database;
    global map_labeled;
    global bw_dilate;
    [y,x] = size(map_labeled);
    labeled_s = s(3);
    labeled_t = t(3);
    if labeled_s==0
        bw_s = zeros([y,x]);
        bw_s(round(s(2)), round(s(1))) = 1;
    else
        bw_s = bw_dilate{labeled_s};
    end
    if labeled_t==0
        bw_t = zeros([y,x]);
        bw_t(round(t(2)), round(t(1))) = 1;
    else
        bw_t = (map_labeled == labeled_t);
    end
    if ~(bw_s & bw_t)
        flag = 0;
    else
        flag = 1;
    end
end
```

print_building.m

```
function str2 = print_building(num)
    global database;
    str2 = [];
    if database.Small(num) == 1
        str2 = [str2, 'small '];
    end
    if database.Medium(num) == 1
        str2 = [str2, 'medium '];
    end
    if database.Large(num) == 1
        str2 = [str2, 'large '];
    end
    if database.SymmetricEW(num) == 1
        str2 = [str2, 'symmetricEastWest '];
    end
    if database.SymmetricNS(num) == 1
        str2 = [str2, 'symmetricNorthSouth '];
    end
    if database.SymmetricEW(num) == 0 && database.SymmetricNS(num) == 0
        str2 = [str2, 'nonSymmetric '];
    end
    if database.OrientedEW(num) == 1
        str2 = [str2, 'orientedEastWest '];
    end
    if database.OrientedNS(num) == 1
        str2 = [str2, 'orientedNorthSouth '];
    end
    if database.Rectangle(num) == 1
        str2 = [str2, 'rectangle '];
    end
    if database.Square(num) == 1
        str2 = [str2, 'square '];
    end
    if database.Rectangle(num) == 0 && database.Square(num) == 0
        str2 = [str2, 'nonRectangle '];
    end
    if database.Upper(num) == 1
        str2 = [str2, 'upperCampus '];
    end
    if database.Lower(num) == 1
        str2 = [str2, 'lowerCampus '];
    end
    if database.Easter(num) == 1
        str2 = [str2, 'easterCampus '];
    end
    if database.Wester(num) == 1
        str2 = [str2, 'westerCampus '];
    end
end
```

print_part1.m

```
function print_part1()
    global database;
    for i = 1:length(database)
        nam = database.Name{i};
        cent = database.Centroid(i,:);
        area = database.Area(i);
        box = database.BoundingBox(i,:);
        str1 = sprintf('%d.\n Name: %s\n Center of Mass: [%d,%d]\n Area: %d\n
Bounding Box: [%d,%d], [%d,%d]\n'...
        , i, nam,
round(cent(1)),round(cent(2)),round(area),round(box(1)),round(box(2)),...
        round(box(1)+box(3)), round(box(2)+box(4)));
        str2 = [' Description: ', print_building(i)];
        str2 = sprintf('%s\n', str2);
        disp([str1,str2]);
    end
end
```

print_part2.m

```
function print_part2(relation)
    global database;
    for i=1:length(database)
        list = struct('e',[], 'w',[], 'n',[], 's',[], 'near',[]);
        name = database.Name{i};
        for j=1:length(database)
            if relation.rel_east(j,i)==1
                list.e = [list.e, j];
            end
            if relation.rel_west(j,i)==1
                list.w = [list.w, j];
            end
            if relation.rel_north(j,i)==1
                list.n = [list.n, j];
            end
            if relation.rel_south(j,i)==1
                list.s = [list.s, j];
            end
            if relation.rel_near(j,i)==1
                list.near = [list.near, j];
            end
        end
        str = sprintf('%d. %s is: ', i, name);
        if (length(list.e)~=0)
            str = [str, 'east of ', add_name(list.e)];
        end
        if (length(list.w)~=0)
            str = [str, 'west of ', add_name(list.w)];
        end
        if (length(list.n)~=0)
            str = [str, 'north of ', add_name(list.n)];
        end
```

```
        if (length(list.s)~=0)
            str = [str, 'south of ', add_name(list.s)];
        end
        str = [str, 'near ', add_name(list.near)];
        disp(str);
    end
end

function str = add_name(lst)
    global database;
    str = '';
    for i = 1:length(lst)
        str = [str, database.Name{lst(i)}, ', '];
    end
end
```