BAHRIA UNIVERSITY KARACHI CAMPUS

DEPARTMENT OF COMPUTER SCIENCES

Intruder Detection & Localization Artificial Intelligence

AIL 201

Muneeza Iftikhar – 012

Hafsa Hafeez Siddiqui – 026

Haris Aamir – 034

Acknowledgements

We are utmost grateful for the opportunity that was provided to us and it wouldn't have been possible without our course teacher Dr.Samabia Tehsin and our lab engineer Laila Nadeem. Both of them helped clarify our concepts throughout the course. Through their guidance we were able to execute our project to the best of our ability. Each group member participated and performed its duty in the right way. This project is the outcome of each group member and its instructors. The project has been assigned to each group member to play its role in accordance to their understanding and feasibility.

Abstract

Intruder Detection and Localization is built on Java and Prolog connectivity. It utilizes concepts of artificial intelligence to detect and locate intruders using searching methods and knowledge base built in prolog.

Contents

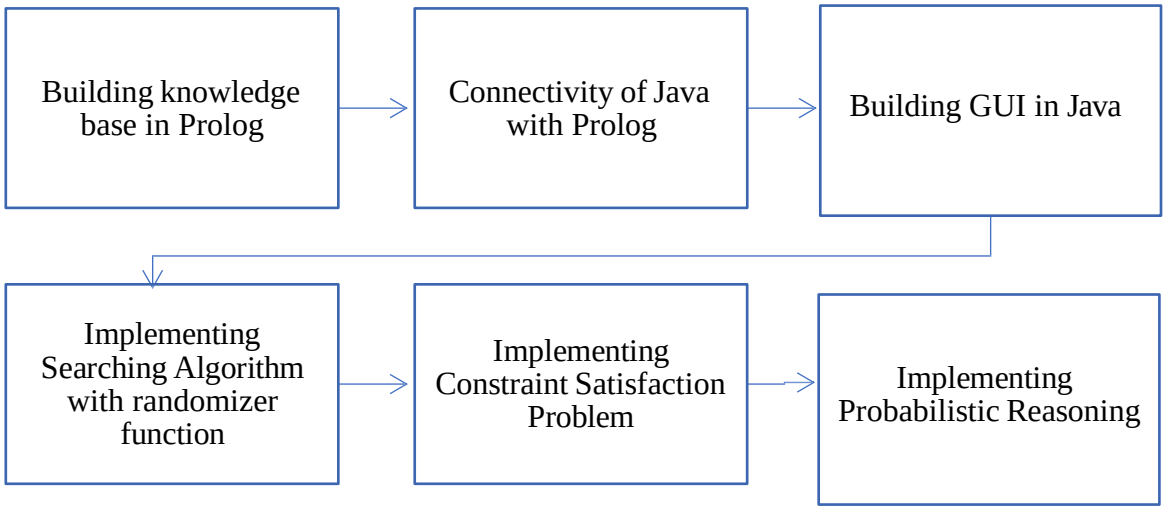
Introduction

Intruder Detection and Localization tries to detect intruders simulated in an apartment complex. However, it is only a simulation and doesn't accurately depict real life scenario. This project was complied on Java and connected through Prolog. Java provides the overall GUI of the project and Prolog implements the agent, searching and its knowledge base of the apartment complex. The apartment complex comprises of 6 floors. On those 6 floors, the agent will try to locate the intruder whether it is an animal or a human being. It will try to provide the shortest path to catch the intruder and an evacuation route for the bystanders.

Problem & Solution

Unfortunately, a lot of people have chosen this path of crime since the Covid and its impact on the job market. In the last few years, there has been a noticeable rise in crime. Furthermore, even if the police were to arrive on time, it would take them a long time to locate the intruder and track it down. This is partly a result of the slow human reaction speed and our inherent limitations. Therefore, an artificial agent could make this duty easier and lower the likelihood of errors in our suggested project, "Intruder Detection and Localization System."

Workflow



Overview of Project

Concepts:

- Building a knowledge base with prolog: The agent requires to have prior knowledge of the entire apartment complex. It needs to know the entire layout of the apartments, what number floors are present, how many apartments are on a single floor etc.
- Connectivity of Java with prolog.
- This program uses informed searching method which is Best First Search. The searching algorithm tries to find the best and shortest path to detect the intruder. Based on the heuristic function, this search strategy prioritizes the states that are closest to the goal state when exploring the possible states, and it stops when it locates the desired state which is the intruder's location in the scenario. The heuristic(X, Y, H) predicate defines a heuristic function which calculates the estimated distance between the current location and the goal location.
- Another concept implemented is from machine learning; Constraint Satisfaction Problem. The goal of CSP here is to find the adjacent rooms from where the intruder is present for evacuation of residents and bystanders. The constraint in this program is that the intruder cannot move diagonally. The only ways possible is to move vertically or horizontally.
- An intruder detection system is installed in a building to detect the presence of intruders. The system has a 90% chance of correctly identifying an intruder when one is present (true positive rate) and a 99% chance of correctly identifying the absence of an intruder when one is not present (true negative rate). The prior probability of an intruder being present in the building is 0.001, or 0.1%.

  Using Bayes' theorem, we can calculate the probability that an intruder is present given a positive detection from the system.

  P(intruder present | positive detection) = P(positive detection | intruder present) * P(intruder present) / P(positive detection)

  Where:

1. • P(intruder present | positive detection) is the probability of an intruder being present given a positive detection from the system, also known as the posterior probability.
2. • P(positive detection | intruder present) is the probability of a positive detection from the system given that an intruder is present, which is 0.9.
3. • P(intruder present) is the prior probability of an intruder being present, which is 0.001.
4. • P(positive detection) is the probability of a positive detection from the system, which can be calculated using the law of total probability.
5. P(positive detection) = P(positive detection | intruder present) * P(intruder present) + P(positive detection | intruder not present) * P(intruder not present)

6. where P(intruder not present) = 1 - P(intruder present) and P(positive detection | intruder not present) = 1 - P(true negative rate) = 1 - 0.99 = 0.01

- By substituting all values, we get P(intruder present | positive detection) = 0.9 * 0.001 / (0.9 * 0.001 + 0.01 * 0.999)
- So, the probability that an intruder is present given a positive detection from the system is approximately 0.081, or 8.1%.

Source Code:

Searching & Knowledge Base in Prolog

```
% facts
apartmentno(1,	1).
apartmentno(1,	2).
apartmentno(1,	3).
apartmentno(2,	1).
apartmentno(2,	2).
apartmentno(2,	3).
apartmentno(3,	1).
apartmentno(3,	2).
apartmentno(3,	3).
apartmentno(4,	1).
apartmentno(4,	2).
apartmentno(4,	3).
apartmentno(5,	1).
apartmentno(5,	2).
apartmentno(5,	3).
apartmentno(6,	1).
apartmentno(6,	2).
apartmentno(6,	3).

% define the goal state
burglar(X, Y) :- random(1, 7, X), random(1, 4, Y).
goal(X, Y) :- apartmentno(X, Y), burglar(X, Y).
% define the heuristic function
heuristic(X, Y, H) :- goal(Xg, Yg), H is abs(X - Xg) + abs(Y - Yg).

% define the possible moves
move(X, Y, X1, Y1) :- apartmentno(X1, Y1), X1 is X+1, Y1 is Y, apartmentno(X1, Y1).
move(X, Y, X1, Y1) :- apartmentno(X1, Y1), X1 is X-1, Y1 is Y, apartmentno(X1, Y1).
move(X, Y, X1, Y1) :- apartmentno(X1, Y1), X1 is X, Y1 is Y+1, apartmentno(X1, Y1).
```

```
move(X, Y, X1, Y1) :- apartmentno(X1, Y1), X1 is X, Y1 is Y-1, apartmentno(X1, Y1).
% define the path-checking predicate
path(X, Y, X, Y, Path, Visited) :- Path = [(X, Y)|Visited]. path(X, Y, X1, Y1,
Path, Visited) :-
  setof((X2, Y2), (move(X, Y, X2, Y2), heuristic(X2, Y2, _), \+ member((X2, Y2),
Visited)), NextMoves),
  member((X2, Y2), NextMoves),
path(X2, Y2, X1, Y1, Path, [(X2, Y2)|Visited]).
% define the predicate to find the path
find_path(X, Y, Path) :- goal(X, Y), path(1, 1, X, Y, Path, []). draw:-
random_number(y, z, X),
assert(rnumber(X)), write(X).
```

Searching in Java (Connectivity)

```
Query q = new Query("consult('newestestest.pl')");
    q.hasSolution();
    String qry = "find_path(Floor, Apartment, Path).";
    q = new Query(qry);
    //mapping of the  solution when the query has been called
    Map<String, Term>[] res = q.allSolutions(); //output of the query has been added in the map.
    JOptionPane.showMessageDialog(this, res[0]);
```

Constraint Satisfaction Problem

```
Query q = new Query("consult('newestestest.pl')");
    q.hasSolution();
    String qry = "find_path(Floor, Apartment, Path).";
    q = new Query(qry);
    Term var1 = q.getSolution().get("Floor");
    Term var2 = q.getSolution().get("Apartment");
```

```
int javaVar1 = var1.intValue();
int javaVar2 = var2.intValue();
Map<String, int[]> adjM = new HashMap<String, int[]>() {{
   put("1,1", new int[] {0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0});
    put("1,2", new int[] {1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0});
    put("1,3", new int[] {0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0});
    put("2,1", new int[] {1,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0});
    put("2,2", new int[] {0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0});
    put("2,3", new int[] {0,0,1,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0});
    put("3,1", new int[] {0,0,0,1,0,0,0,1,0,1,0,0,0,0,0,0,0,0});
    put("3,2", new int[] {0,0,0,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0});
    put("3,3", new int[] {0,0,0,0,0,1,0,1,0,0,0,1,0,0,0,0,0,0});
    put("4,1", new int[] {0,0,0,0,0,0,1,0,0,0,1,0,1,0,0,0,0,0});
    put("4,2", new int[] {0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,0,0,0});
    put("4,3", new int[] {0,0,0,0,0,0,0,0,1,0,1,0,0,0,1,0,0,0});
    put("5,1", new int[] {0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,1,0,0});
    put("5,2", new int[] {0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0});
    put("5,3", new int[] {0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,1});
    put("6,1", new int[] {0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0});
    put("6,2", new int[] {0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1});
    put("6,3", new int[] {0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0});
}};
for (Map.Entry<String, int[]> entry : adjM.entrySet()) {
    String key = entry.getKey();
    int[] value = entry.getValue();
```

```
        String Floor=String.valueOf(javaVar1);
        String Apartment=String.valueOf(javaVar2);
        String location = Floor + ","+  Apartment;
        if(key.equals(location))}                }
        for (int i = 0; i < 18; i++) {
           if(key.equals(location)){
              if (value[i] == 1) {
                 int row = i / 3 + 1;
                 int col = i % 3 + 1;
              JOptionPane.showMessageDialog(this, "Please Evacuate: (" + row + "," + col + ") ");
}}}
        System.out.println();     }
```

Bayes Theorem:

```
 double posterior_probability = 0 ; //intruder present | positive detection
       double prior_probability = 0.1; //intruder present
       double true_positive_rate = 0.9; //positive detection | intruder present
       double true_negative_rate = 0.99; //positive detection | intruder not present
       double positive_detection = 0; //positive detection


       positive_detection = true_positive_rate * prior_probability + true_negative_rate * (1-
 true_negative_rate) ;
       posterior_probability = (true_positive_rate * prior_probability)/positive_detection;
       JOptionPane.showMessageDialog(this, "Probability of an intruder is present given a positive
 detection from the system is approximately "+posterior_probability);
```

Output

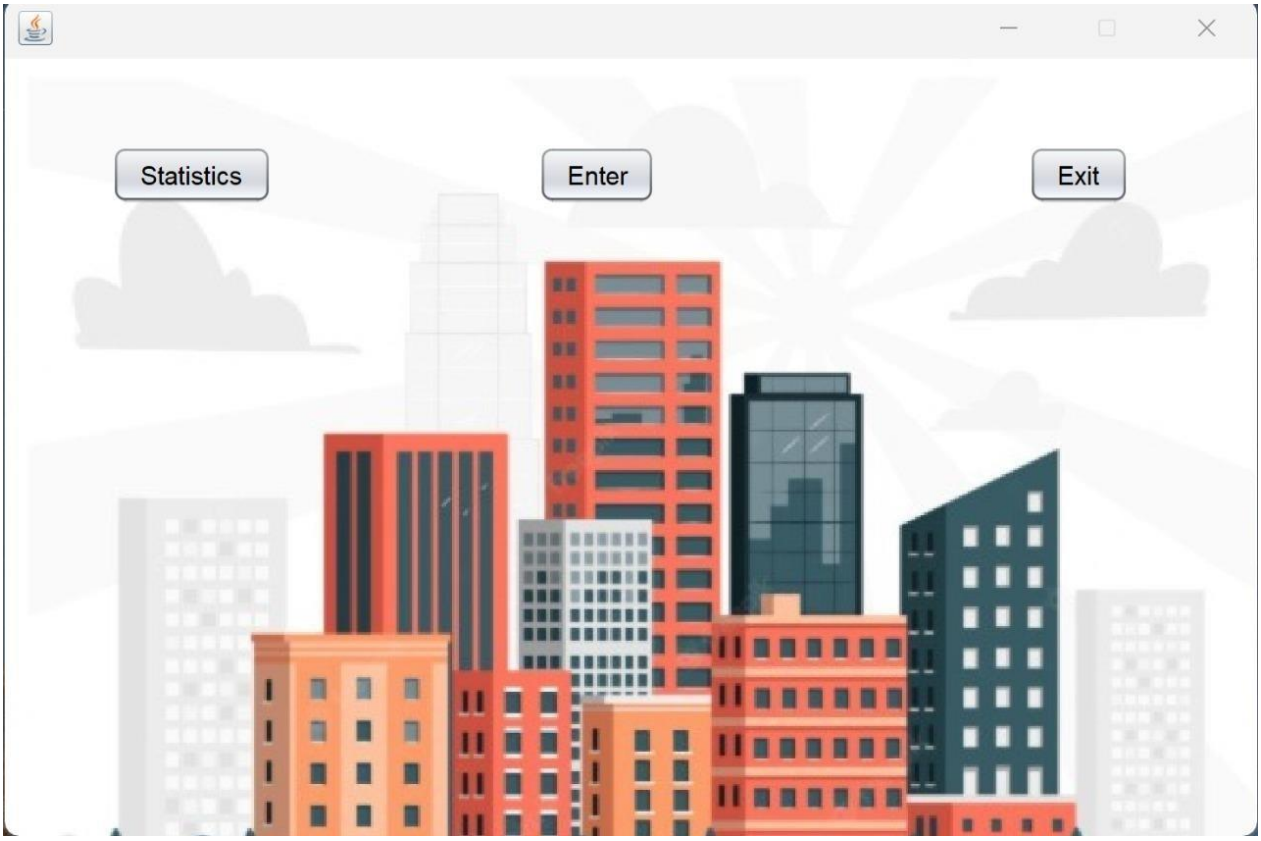


Figure i Home Page

Figure ii Main Page

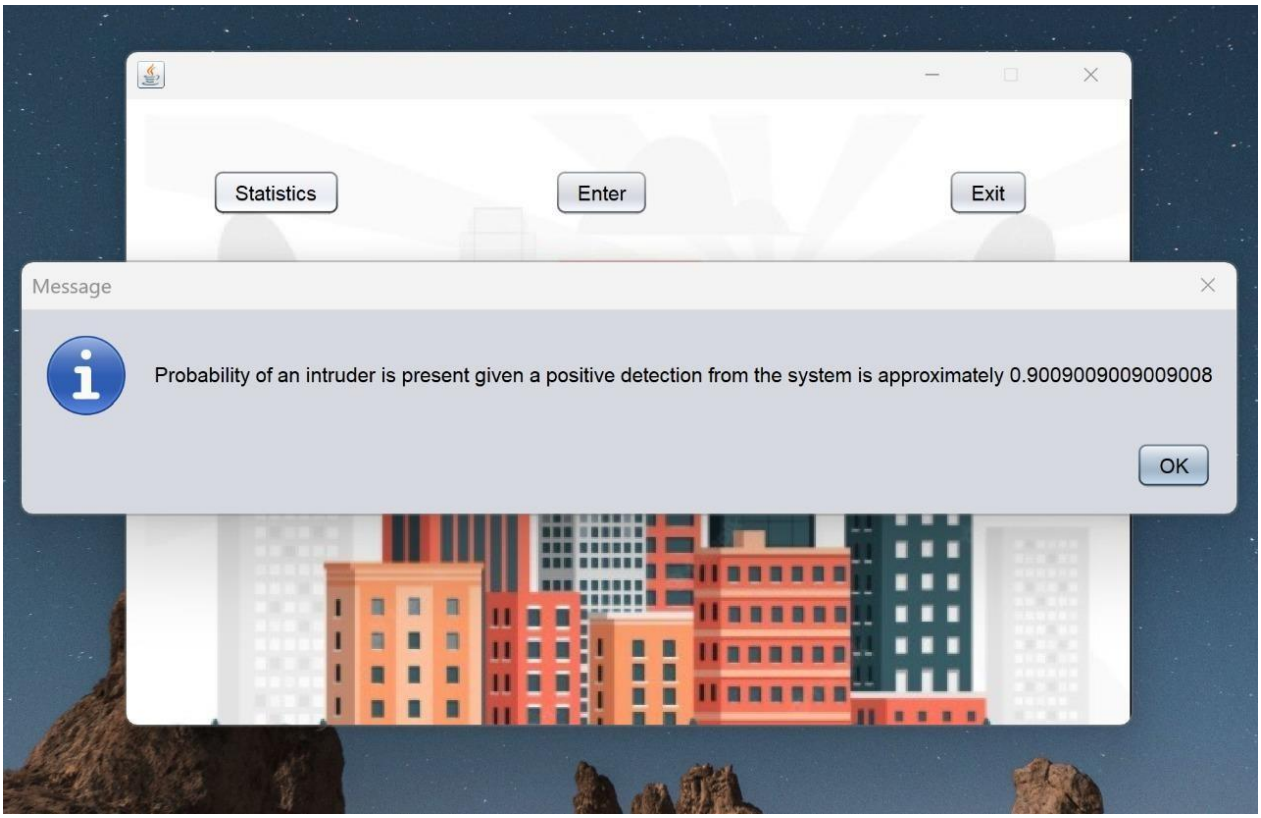


Figure iii Bayes Theorem

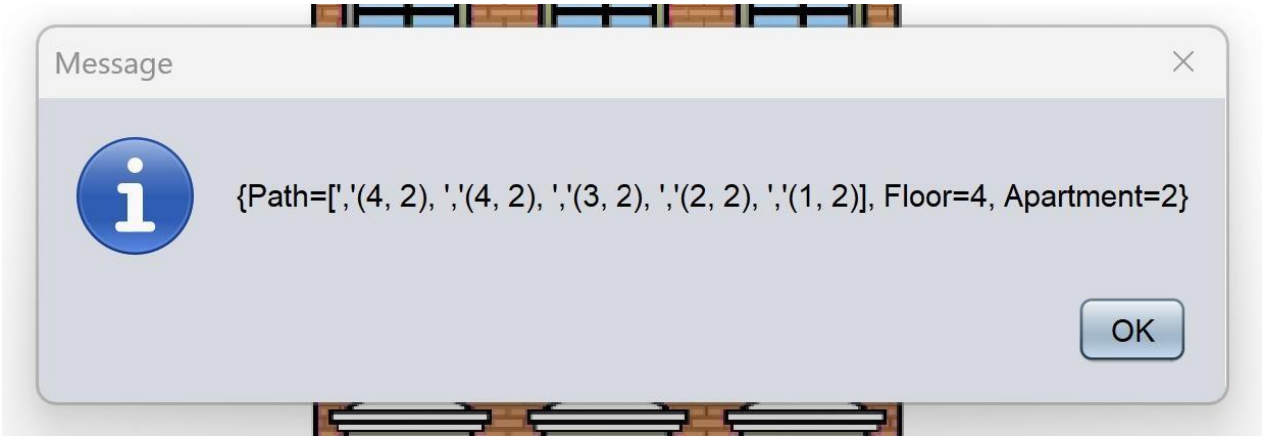


Figure iv Best First Search Output

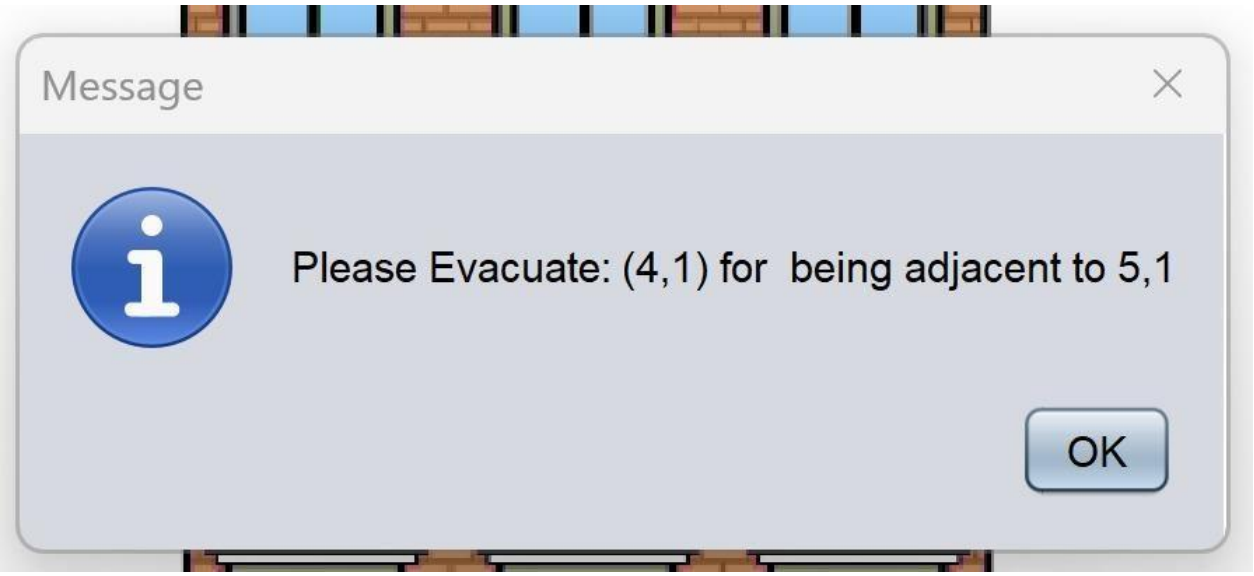


Figure v Constraint Satisfaction Problem

Conclusion

It is important to note that our program does not accurately depict the variables and constraints present in real life. It is only meant to be a simulation of what the agent can be used with the basic artificial intelligence and machine learning concepts that have been implemented in this project. Furthermore, more data and more complex concepts are required for it.