SC627-Assignment3
Robot Chasing Target using grid based planner

Krisha Shah, 21d070039

April 2024

1 Planner Summary

To solve the planning problem, I implemented a forward weighted A* multi-goal planner with an informative heuristic. For the graph nodes, I created a struct with the following structure:

Node
1. map index (output of GETMAPINDEX)
2. time
3. g-value
4. h-value
5. f-value
6. shared pointer to parent node

My robot state is 3-dimensional - $<$ x, y, t $>$. The grid is actually 9-connected instead of 8. At any point, the robot has the possibility to move to 9 other states in the next time layer - 8 neighbors + staying in its current cell.

To save on memory, I only store one copy of my graph nodes, and use shared pointers to the nodes for all other maps/lists/queues. Since it is a smart pointer, this also takes care of automatically deleting the memory allocated on the heap once all pointer instances are destroyed.

I used the following data-structures:

- An unordered map to map shared pointer and store my graph of nodes
- A priority queue of shared pointers for the open list, prioritized based on the f-values using a custom comparator
- An unordered set of int for the closed list

- A stack of int to store the computed path. Once the entire path is computed, the algorithm backtracks stores all the nodes in the stack. The planner keeps popping a node at every time-step and returns it for the next move

To calculate the heuristics, I used a lower dimensional (2D) planning problem, and performed a backward Dijkstra search from all possible goal cells. I used the 2 nd half of the target trajectory as goals. The heuristic cost for a grid cell in the map is basically the Dijkstra cost to the closest goal.

In the main path computation loop, I add a time factor to the heuristic for each state to make it more informed. So for each state, the final heuristic is a sum of the heuristic cost calculated by the backward 2D Dijkstra approach, and the time difference between the current time, and the time of the goal which is closest the state. So, for every state being explored which is not a goal index,
h_s = 2D Dijkstra cost to closest goal + (time of closest goal current time)

This is to take care of the fact that if the algorithm, while exploring, reaches a goal index in the map, but the target has not reached there yet, there is a possibility for the robot to just wait at that cell until the target arrives. In this case, the cost incurred will be the cost of that grid cell multiplied by the time remaining for the target to arrive. In this case, where a goal index is being explored, the heuristic perfectly estimates the cost to the goal.

Finally, I also weighted my A* search with $\epsilon = 1.8$

2 How to Compile my code

```
g++ runtest.cpp planner.cpp
```
To run the planner:
```
./a.out map3.txt
```

3 Results

Map	Target caught	Time Taken	Moves Made	Path cost
Map1	1	2871	2867	2871
Map2	1	4684	4609	1615823
Map3	1	345	343	345
Map4	1	381	340	381
Map5	1	179	167	1845
Map6	1	140	139	558
Map7	0	300	-1	300
Map8	1	431	430	431
Map9	1	368	367	368

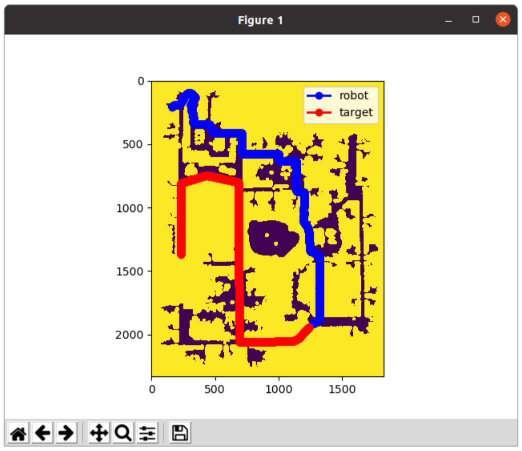
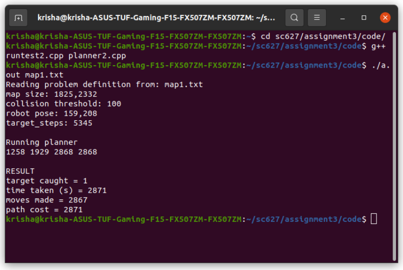


Figure 1: Map1 trajectory



```
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~$ cd sc627/assignment3/code/
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ g++
runtest2.cpp planner2.cpp
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ ./a.
out map1.txt
Reading problem definition from: map1.txt
map size: 1825,2332
collision threshold: 100
robot pose: 159,208
target_steps: 5345

Running planner
1258 1929 2868 2868

RESULT
target caught = 1
time taken (s) = 2871
moves made = 2867
path cost = 2871
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ 
```

Figure 2: Map1 Result

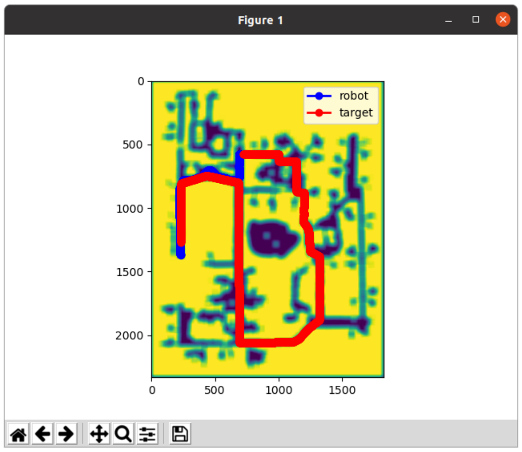
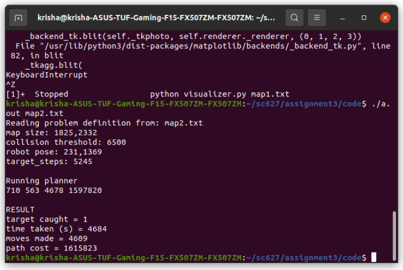


Figure 3: Map2 trajectory



```
    _backend_tk.blit(self._tkphoto, self.renderer._renderer, (0, 1, 2, 3))
  File "/usr/lib/python3/dist-packages/matplotlib/backends/_backend_tk.py", line
 82, in blit
    _tkagg.blit(
KeyboardInterrupt
^Z
[1]+  Stopped                 python visualizer.py map1.txt
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ ./a.
out map2.txt
Reading problem definition from: map2.txt
map size: 1825,2332
collision threshold: 6500
robot pose: 231,1369
target_steps: 5245

Running planner
710 563 4678 1597820

RESULT
target caught = 1
time taken (s) = 4684
moves made = 4609
path cost = 1615823
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ 
```

Figure 4: Map2 result

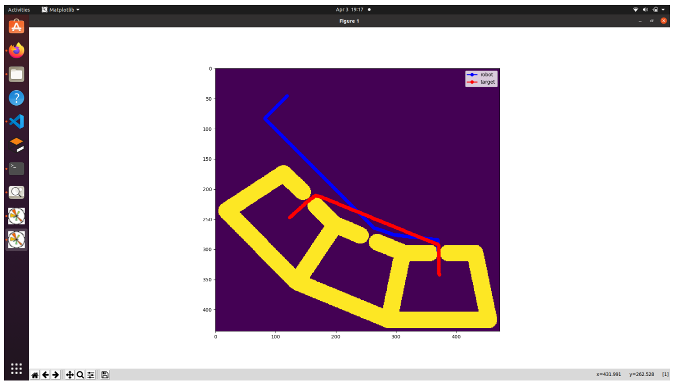
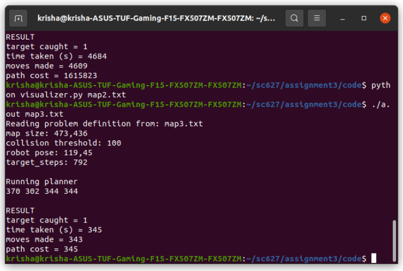


Figure 5: Map3 trajectory



```
RESULT
target caught = 1
time taken (s) = 4684
moves made = 4609
path cost = 1615823
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ pyth
on visualizer.py map2.txt
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ ./a.
out map3.txt
Reading problem definition from: map3.txt
map size: 473,436
collision threshold: 100
robot pose: 119,45
target_steps: 792

Running planner
370 302 344 344

RESULT
target caught = 1
time taken (s) = 345
moves made = 343
path cost = 345
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$
```

Figure 6: Map3 result

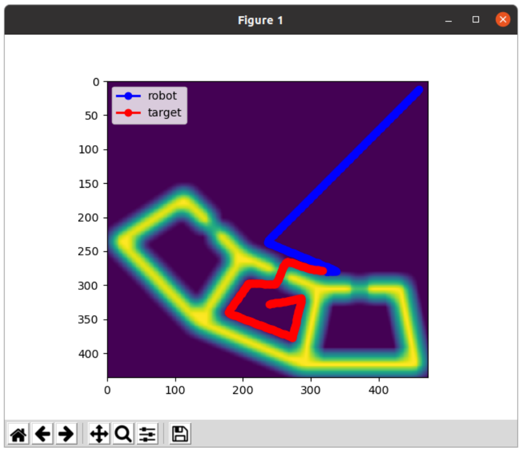
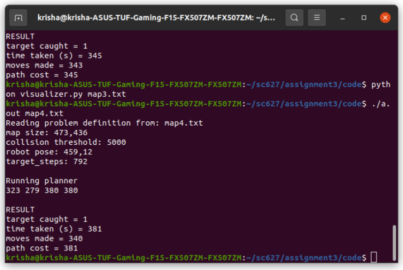


Figure 7: Map4 trajectory



```
RESULT
target caught = 1
time taken (s) = 345
moves made = 343
path cost = 345
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ pyth
on visualizer.py map3.txt
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ ./a.
out map4.txt
Reading problem definition from: map4.txt
map size: 473,436
collision threshold: 5000
robot pose: 459,12
target_steps: 792

Running planner
323 279 380 380

RESULT
target caught = 1
time taken (s) = 381
moves made = 340
path cost = 381
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$
```

Figure 8: Map4 result

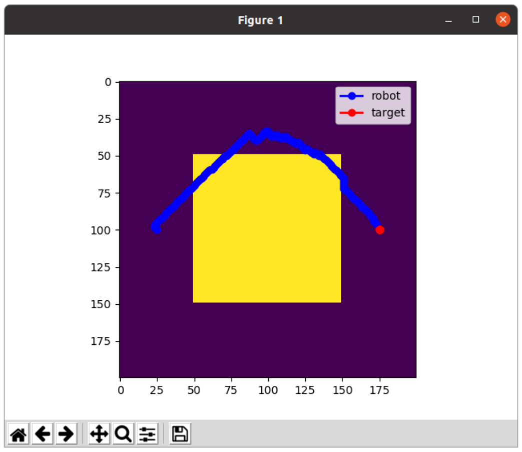
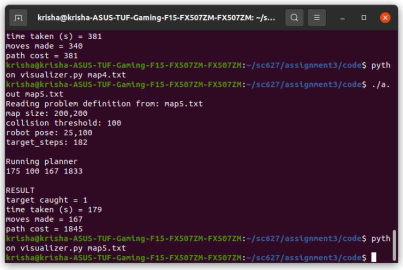


Figure 9: Map5 trajectory



```
time taken (s) = 381
moves made = 340
path cost = 381
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ pyth
on visualizer.py map4.txt
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ ./a.
out map5.txt
Reading problem definition from: map5.txt
map size: 200,200
collision threshold: 100
robot pose: 25,100
target_steps: 182

Running planner
175 100 167 1833

RESULT
target caught = 1
time taken (s) = 179
moves made = 167
path cost = 1845
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ pyth
on visualizer.py map5.txt
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$
```

Figure 10: Map5 result

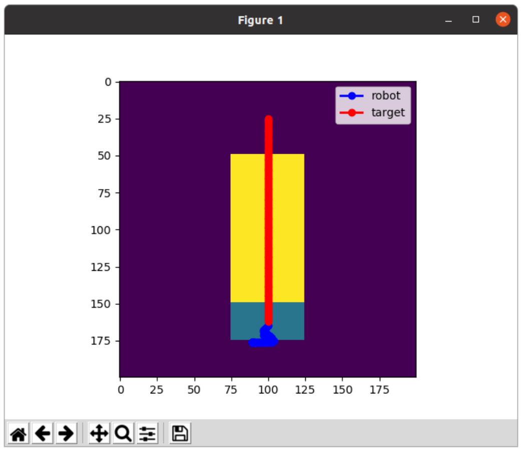
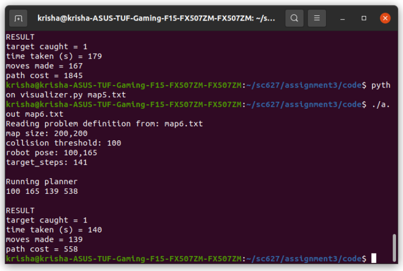


Figure 11: Map6 trajectory

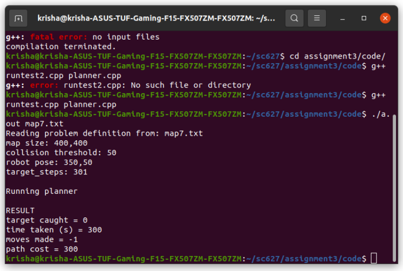


```
RESULT
target caught = 1
time taken (s) = 179
moves made = 167
path cost = 1845
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ pyth
on visualizer.py map5.txt
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ ./a.
out map6.txt
Reading problem definition from: map6.txt
map size: 200,200
collision threshold: 100
robot pose: 100,165
target_steps: 141

Running planner
100 165 139 538

RESULT
target caught = 1
time taken (s) = 140
moves made = 139
path cost = 558
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$
```

Figure 12: Map6 result



```
g++: fatal error: no input files
compilation terminated.
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627$ cd assignment3/code/
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ g++
runtest2.cpp planner.cpp
g++: error: runtest2.cpp: No such file or directory
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ g++
runtest.cpp planner.cpp
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ ./a.
out map7.txt
Reading problem definition from: map7.txt
map size: 400,400
collision threshold: 50
robot pose: 350,50
target_steps: 301

Running planner

RESULT
target caught = 0
time taken (s) = 300
moves made = -1
path cost = 300
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$
```

Figure 13: Map7 result

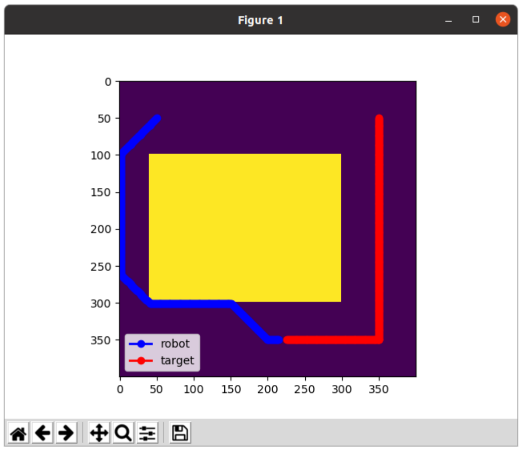
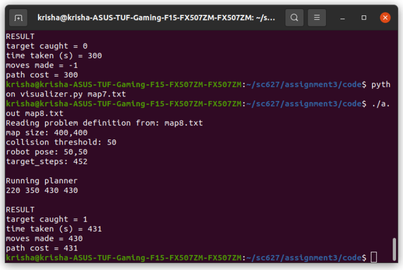


Figure 14: Map8 trajectory



```
RESULT
target caught = 0
time taken (s) = 300
moves made = -1
path cost = 300
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ pyth
on visualizer.py map7.txt
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ ./a.
out map8.txt
Reading problem definition from: map8.txt
map size: 400,400
collision threshold: 50
robot pose: 50,50
target_steps: 452

Running planner
220 350 430 430

RESULT
target caught = 1
time taken (s) = 431
moves made = 430
path cost = 431
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$
```

Figure 15: Map8 result

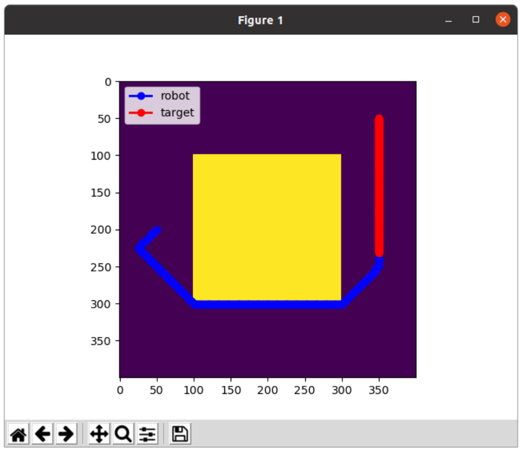
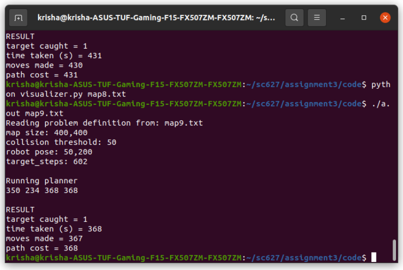


Figure 16: Map9 trajectory



```
RESULT
target caught = 1
time taken (s) = 431
moves made = 430
path cost = 431
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ pyth
on visualizer.py map8.txt
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$ ./a.
out map9.txt
Reading problem definition from: map9.txt
map size: 400,400
collision threshold: 50
robot pose: 50,200
target_steps: 602

Running planner
350 234 368 368

RESULT
target caught = 1
time taken (s) = 368
moves made = 367
path cost = 368
krisha@krisha-ASUS-TUF-Gaming-F15-FX507ZM-FX507ZM:~/sc627/assignment3/code$
```

Figure 17: Map9 result