**AGV-TASK ROUND**

Submitted By-

**Kaushal Jadhav**

Roll No -20EC30019

**Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Topic** | | **Page No** |
| 1 | Task 1 | Introduction | 4-40 |
| Result |
| Program |
| 2 | Task 3 | Introduction | 41-76 |
| Program |
| Result |
| 3 | Task 5 | Introduction | 77-116 |
| Program and Result using webcam |
| Program and Result using mobile phone camera |
| 4 | Task 4 | Introduction | 118-122 |
| Program |
| Explanation and Result |

**TASK 1**

**Task 1**

**Astar**

**Introduction**

A-star (also referred to as A\*) is one of the most successful search algorithms to find the shortest path between nodes or graphs. It is an informed search algorithm, as it uses information about path cost and also uses heuristics to find the solution. A\* algorithm is a searching algorithm that searches for the shortest path between the initial and the final state.

It is used in various applications, such as maps. In maps, the A\* algorithm is used to calculate the shortest distance between the source (initial state) and the destination (final state).

Example: Imagine a square grid which possesses many obstacles, scattered randomly. The initial and the final cell is provided. The aim is to reach the final cell in the shortest amount of time. Here A\* search Algorithm comes to the rescue.

**Explanation:**

A\* algorithm has 3 parameters.

1) g: It is the cost of moving from the initial cell to the current cell. Basically, it is the sum of all the cells that have been visited since leaving the first cell.

2) h: It is also known as the heuristic value, it is the estimated cost of moving from the current cell to the final cell. The actual cost cannot be calculated until the final cell is reached. Hence, h is the estimated cost. One must make sure that there is never an over estimation of the cost.

3) f: It is the sum of g and h.

So, f = g + h

The algorithm makes its decisions by taking the f-value into account. The algorithm selects the smallest f-valued cell and moves to that cell. This process continues until the algorithm reaches its goal cell.

Terminology:

* Node (also called State) — all potential positions or stops with a unique identification
* Transition — the act of moving between states or nodes.
* Starting Node — where to start searching
* Goal Node — the target to stop searching
* Search Space — a collection of nodes, like all board positions of a board game
* Cost — numerical value (say distance, time, or financial expense) for the path from a node to another node.
* g(n) —the exact cost of the path from the starting node to any node n
* h(n) —the heuristic estimated cost from node n to the goal node
* f(n) — lowest cost in the neighbouring node n

Each time A\* enters a node, it calculates the cost, f(n) (n being the neighbouring node), to travel to all of the neighbouring nodes and then enters the node with the lowest value of f(n). These values are calculated using following formula.

f(n) = g(n) + h(n)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Result**

Time taken (in seconds):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Diag-dist | Dijkstra | Euclidean | H1 | Manhattan Distance |
| Case1 | 0.347 | 0.524 | 0.291 | 0.227 | 0.031 |
| Case2 (1) | 0.25 | 0.315 | 0.268 | 0.141 | 0.19 |
| Case2 (2) | 0.238 | 0.292 | 0.244 | 0.4 | 0.219 |

Case 1- Allowed to move in any direction.

Case2-Allowed to move in only 4 directions.

The second cost function (in case of case-2):

In reality the bot takes some time to turn. So, to include that delay the cost function is changed accordingly.

**Cost:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Diag-dist | Dijkstra | Euclidean Distance | H1 | Manhattan Distance |
| Case1 | 147.61 | 147.61 | 147.61 | 162.92 | 147.61 |
| Case2 (1) | 181.0 | 181.0 | 181.0 | 191.0 | 181.0 |
| Case2 (2) | 185.0 | 184.0 | 185.0 | 224.0 | 185.0 |

**Hence here, Dijkstra always gave the shortest path. The difference becomes more evident in the Case-2(2).**

**The behavior of H1 is peculiar. It works with the best results in a maze with no obstacle on the line joining start to end. However, in other cases it may not provide optimum solution.**

**But Dijkstra also consumes a lot of time in searching.**

**Hence it can be stated that- The cost and time both compromised simultaneously.**

**The choice of the Heuristic depends on the maze, application etc.**

**Whenever time is not the constraint Dijkstra is the most-reliable.**

**H1 is non-admissible. It Can be proved as:**

****

As seen from the figure the highlighted green path is not the shortest path.



This is actually the shortest path.

****

**Case 1 -Diagonal Distance**

****

**Case 1 Dijkstra**

****

**Case 1 Euclidean Distance**

****

**Case 1 H1**

****

**Case 1 Manhattan Distance**

****

**Case 2 Diagonal Distance**

****

**Case 2 Dijkstra**

****

**Case 2 Euclidean Distance**

****

**Case 2 H1**

****

**Case2 Manhattan Distance**

****

**Case 2(2) Diagonal Distance**

****

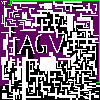
**Case 2(2) Dijkstra**

****

**Case 2(2) Euclidean Distance**

****

**Case 2(2) H1**

****

**Case 2(2) Manhattan Distance**

**Detailed Explanation of Result**

**1) Diagonal Distance case (1) -**

cost of traversing by distance=147.61

cost stored at end point=147.61

No. of nodes in path=125

No of nodes visited=3224

start point=(4, 7)

end point=(82, 96)

Nodes in path-

[(4, 7), (4, 8), (4, 9), (4, 10), (3, 11), (4, 12), (4, 13), (4, 14), (4, 15), (4, 16), (4, 17), (5, 18), (6, 19), (7, 20), (8, 21), (8, 22), (9, 23), (10, 24), (11, 25), (11, 26), (12, 27), (13, 27), (14, 28), (15, 29), (16, 30), (17, 31), (18, 32), (19, 33), (20, 33), (21, 33), (22, 32), (23, 31), (24, 31), (25, 31), (26, 30), (27, 30), (28, 31), (28, 32), (28, 33), (28, 34), (28, 35), (29, 36), (30, 37), (31, 37), (32, 37), (33, 38), (33, 39), (33, 40), (33, 41), (34, 42), (35, 42), (36, 43), (37, 43), (38, 43), (39, 43), (40, 44), (41, 45), (42, 46), (43, 47), (44, 48), (45, 49), (46, 50), (47, 51), (48, 52), (49, 53), (50, 54), (51, 55), (52, 55), (53, 56), (54, 57), (55, 57), (56, 57), (57, 57), (58, 57), (59, 57), (60, 58), (61, 59), (62, 60), (63, 60), (64, 61), (65, 62), (66, 63), (67, 63), (68, 64), (69, 65), (70, 65), (71, 65), (72, 66), (73, 67), (74, 67), (75, 67), (76, 67), (77, 67), (78, 68), (78, 69), (78, 70), (78, 71), (78, 72), (78, 73), (77, 74), (78, 75), (79, 76), (80, 76), (81, 76), (82, 76), (83, 77), (83, 78), (83, 79), (83, 80), (83, 81), (84, 82), (84, 83), (83, 84), (83, 85), (83, 86), (83, 87), (83, 88), (82, 89), (82, 90), (82, 91), (82, 92), (82, 93), (82, 94), (82, 95), (82, 96)]

time taken for traversing=0.347 second

**2) Dijkstra case (1) -**

cost of traversing by distance=147.61

cost stored at end point=147.61

No. of nodes in path=125

No of nodes visited=5637

start point= (4, 7)

end point=(82, 96)

Nodes in path-

[(4, 7), (4, 8), (4, 9), (4, 10), (3, 11), (4, 12), (4, 13), (4, 14), (4, 15), (4, 16), (4, 17), (5, 18), (6, 19), (7, 20), (8, 21), (8, 22), (8, 23), (9, 24), (10, 25), (11, 26), (12, 27), (13, 27), (14, 28), (15, 29), (16, 30), (17, 31), (18, 32), (19, 33), (20, 33), (21, 33), (22, 32), (23, 31), (24, 31), (25, 31), (26, 30), (27, 30), (28, 31), (28, 32), (28, 33), (28, 34), (28, 35), (29, 36), (30, 36), (31, 36), (32, 37), (33, 38), (33, 39), (33, 40), (33, 41), (34, 42), (35, 42), (36, 42), (37, 42), (38, 42), (39, 42), (40, 42), (41, 43), (42, 44), (43, 45), (44, 46), (45, 47), (46, 47), (47, 48), (48, 49), (49, 50), (50, 51), (51, 52), (52, 53), (53, 53), (54, 53), (55, 54), (56, 55), (57, 56), (58, 56), (59, 56), (60, 57), (61, 58), (62, 59), (63, 60), (64, 61), (65, 61), (66, 61), (67, 62), (68, 62), (69, 62), (70, 62), (71, 63), (72, 63), (73, 64), (74, 65), (75, 65), (76, 66), (77, 67), (78, 68), (78, 69), (78, 70), (78, 71), (78, 72), (78, 73), (77, 74), (78, 75), (79, 75), (80, 76), (81, 76), (82, 76), (83, 77), (83, 78), (83, 79), (83, 80), (83, 81), (84, 82), (84, 83), (83, 84), (83, 85), (83, 86), (83, 87), (83, 88), (82, 89), (82, 90), (82, 91), (82, 92), (82, 93), (82, 94), (82, 95), (82, 96)]

time taken for traversing=0.524 second

**3) Euclidean distance case (1) -**

cost of traversing by distance=147.61

cost stored at end point=147.61

No. of nodes in path=125

No of nodes visited=2797

start point=(4, 7)

end point=(82, 96)

Nodes in path-

[(4, 7), (4, 8), (4, 9), (4, 10), (3, 11), (4, 12), (4, 13), (4, 14), (4, 15), (4, 16), (4, 17), (5, 18), (6, 19), (7, 20), (8, 21), (8, 22), (8, 23), (9, 24), (10, 25), (11, 26), (12, 27), (13, 27), (14, 28), (15, 29), (16, 30), (17, 31), (18, 32), (19, 33), (20, 33), (21, 33), (22, 32), (23, 31), (24, 31), (25, 31), (26, 30), (27, 30), (28, 31), (28, 32), (28, 33), (28, 34), (28, 35), (29, 36), (30, 36), (31, 36), (32, 37), (33, 38), (33, 39), (33, 40), (33, 41), (34, 42), (35, 42), (36, 42), (37, 42), (38, 42), (39, 42), (40, 42), (41, 43), (42, 44), (43, 45), (44, 46), (45, 47), (46, 47), (47, 48), (48, 49), (49, 50), (50, 51), (51, 52), (52, 53), (53, 53), (54, 53), (55, 54), (56, 55), (57, 56), (58, 56), (59, 56), (60, 57), (61, 58), (62, 59), (63, 60), (64, 61), (65, 61), (66, 61), (67, 62), (68, 62), (69, 62), (70, 62), (71, 63), (72, 63), (73, 64), (74, 65), (75, 65), (76, 66), (77, 67), (78, 68), (78, 69), (78, 70), (78, 71), (78, 72), (78, 73), (77, 74), (78, 75), (79, 75), (80, 76), (81, 76), (82, 76), (83, 77), (83, 78), (83, 79), (83, 80), (83, 81), (84, 82), (84, 83), (83, 84), (83, 85), (83, 86), (83, 87), (83, 88), (82, 89), (82, 90), (82, 91), (82, 92), (82, 93), (82, 94), (82, 95), (82, 96)]

time taken for traversing=0.291 second

**4) H1 case (1) -**

cost of traversing by distance=162.92

cost stored at end point=152.92

No. of nodes in path=137

No of nodes visited=1325

start point=(4, 7)

end point=(82, 96)

Nodes in path-

[(4, 7), (4, 8), (4, 9), (4, 10), (3, 11), (4, 12), (5, 13), (5, 14), (5, 15), (5, 16), (6, 17), (7, 18), (8, 19), (8, 20), (8, 21), (9, 22), (10, 23), (11, 24), (11, 25), (11, 26), (12, 27), (13, 27), (14, 28), (15, 29), (16, 30), (17, 31), (18, 32), (19, 33), (20, 32), (21, 31), (22, 31), (23, 31), (24, 31), (25, 31), (26, 30), (27, 30), (28, 31), (28, 32), (28, 33), (28, 34), (28, 35), (29, 36), (30, 37), (31, 37), (32, 37), (33, 38), (33, 39), (33, 40), (33, 41), (34, 42), (35, 42), (36, 43), (37, 43), (38, 43), (39, 43), (40, 44), (41, 45), (42, 46), (43, 47), (44, 48), (45, 49), (46, 50), (47, 51), (48, 52), (49, 53), (50, 54), (51, 55), (52, 55), (53, 56), (54, 57), (55, 57), (56, 57), (57, 57), (58, 57), (59, 57), (60, 58), (60, 59), (60, 60), (60, 61), (60, 62), (60, 63), (61, 64), (62, 65), (63, 66), (64, 67), (65, 67), (66, 67), (67, 67), (68, 68), (68, 69), (67, 70), (66, 71), (66, 72), (66, 73), (66, 74), (67, 75), (68, 76), (68, 77), (68, 78), (68, 79), (67, 80), (66, 80), (65, 80), (64, 81), (63, 81), (62, 82), (61, 83), (61, 84), (62, 85), (63, 85), (64, 85), (65, 86), (66, 87), (66, 88), (67, 89), (68, 89), (69, 90), (69, 91), (70, 92), (71, 92), (72, 92), (73, 92), (74, 92), (75, 92), (76, 91), (76, 90), (76, 89), (77, 88), (78, 88), (79, 89), (79, 90), (79, 91), (79, 92), (79, 93), (80, 94), (81, 95), (82, 96)]

time taken for traversing=0.227 second

**5) Manhattan Distance case (1) -**

cost of traversing by distance=147.61

cost stored at end point=147.61

No. of nodes in path=125

No of nodes visited=511

start point=(4, 7)

end point=(82, 96)

Nodes in path-

[(4, 7), (4, 8), (4, 9), (4, 10), (3, 11), (4, 12), (4, 13), (4, 14), (4, 15), (4, 16), (4, 17), (5, 18), (6, 19), (7, 20), (8, 21), (8, 22), (8, 23), (9, 24), (10, 25), (11, 26), (12, 27), (13, 27), (14, 28), (15, 29), (16, 30), (17, 31), (18, 32), (19, 33), (20, 33), (21, 33), (22, 32), (23, 31), (24, 31), (25, 31), (26, 30), (27, 30), (28, 31), (28, 32), (28, 33), (28, 34), (28, 35), (29, 36), (30, 36), (31, 36), (32, 37), (33, 38), (33, 39), (33, 40), (33, 41), (34, 42), (35, 42), (36, 42), (37, 42), (38, 42), (39, 42), (40, 42), (41, 43), (42, 44), (43, 45), (44, 46), (45, 47), (46, 47), (47, 48), (48, 49), (49, 50), (50, 51), (51, 52), (52, 53), (53, 53), (54, 53), (55, 54), (56, 55), (57, 56), (58, 56), (59, 56), (60, 57), (61, 58), (62, 59), (63, 60), (64, 61), (65, 61), (66, 61), (67, 62), (68, 62), (69, 62), (70, 62), (71, 63), (72, 63), (73, 64), (74, 65), (75, 65), (76, 66), (77, 67), (78, 68), (78, 69), (78, 70), (78, 71), (78, 72), (78, 73), (77, 74), (78, 75), (79, 75), (80, 76), (81, 76), (82, 76), (83, 77), (83, 78), (83, 79), (83, 80), (83, 81), (84, 82), (84, 83), (83, 84), (83, 85), (83, 86), (83, 87), (83, 88), (82, 89), (82, 90), (82, 91), (82, 92), (82, 93), (82, 94), (82, 95), (82, 96)]

time taken for traversing=0.031 second

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1) Diagonal Distance case (2) -**

cost of traversing by distance=181.0

cost stored at end point=181.0

No. of nodes in path=182

No of nodes visited=3951

start point=(4, 7)

end point=(82, 96)

Nodes in path-

[(4, 7), (4, 8), (4, 9), (4, 10), (3, 10), (3, 11), (2, 11), (1, 11), (0, 11), (0, 12), (0, 13), (0, 14), (1, 14), (1, 15), (1, 16), (1, 17), (1, 18), (1, 19), (1, 20), (1, 21), (1, 22), (1, 23), (1, 24), (1, 25), (1, 26), (1, 27), (1, 28), (1, 29), (1, 30), (1, 31), (1, 32), (1, 33), (1, 34), (1, 35), (1, 36), (1, 37), (2, 37), (2, 38), (2, 39), (2, 40), (2, 41), (2, 42), (2, 43), (2, 44), (2, 45), (2, 46), (2, 47), (2, 48), (2, 49), (2, 50), (2, 51), (2, 52), (3, 52), (4, 52), (5, 52), (6, 52), (7, 52), (7, 53), (7, 54), (7, 55), (7, 56), (7, 57), (7, 58), (7, 59), (7, 60), (7, 61), (7, 62), (7, 63), (8, 63), (9, 63), (10, 63), (11, 63), (11, 64), (11, 65), (11, 66), (12, 66), (12, 67), (12, 68), (12, 69), (12, 70), (12, 71), (12, 72), (12, 73), (12, 74), (12, 75), (12, 76), (12, 77), (12, 78), (12, 79), (12, 80), (12, 81), (12, 82), (13, 82), (14, 82), (14, 83), (14, 84), (14, 85), (14, 86), (14, 87), (14, 88), (14, 89), (14, 90), (14, 91), (14, 92), (14, 93), (14, 94), (14, 95), (14, 96), (14, 97), (14, 98), (15, 98), (16, 98), (17, 98), (18, 98), (19, 98), (20, 98), (21, 98), (22, 98), (23, 98), (24, 98), (25, 98), (26, 98), (27, 98), (28, 98), (29, 98), (30, 98), (31, 98), (32, 98), (33, 98), (34, 98), (34, 99), (35, 99), (36, 99), (37, 99), (38, 99), (39, 99), (40, 99), (41, 99), (42, 99), (43, 99), (44, 99), (45, 99), (46, 99), (47, 99), (48, 99), (49, 99), (50, 99), (51, 99), (52, 99), (53, 99), (54, 99), (55, 99), (56, 99), (57, 99), (58, 99), (59, 99), (60, 99), (61, 99), (62, 99), (63, 99), (64, 99), (65, 99), (66, 99), (67, 99), (68, 99), (69, 99), (70, 99), (71, 99), (72, 99), (73, 99), (74, 99), (75, 99), (76, 99), (77, 99), (78, 99), (79, 99), (79, 98), (80, 98), (81, 98), (81, 97), (81, 96), (82, 96)]

time taken for traversing=0.25 second

**2) Dijkstra case (2) -**

cost of traversing by distance=181.0

cost stored at end point=181.0

No. of nodes in path=182

No of nodes visited=5176

start point=(4, 7)

end point=(82, 96)

Nodes in path-

[(4, 7), (4, 8), (4, 9), (4, 10), (3, 10), (3, 11), (2, 11), (1, 11), (0, 11), (0, 12), (0, 13), (0, 14), (1, 14), (1, 15), (1, 16), (1, 17), (1, 18), (1, 19), (1, 20), (1, 21), (1, 22), (1, 23), (1, 24), (1, 25), (1, 26), (1, 27), (1, 28), (1, 29), (1, 30), (1, 31), (1, 32), (1, 33), (1, 34), (1, 35), (1, 36), (1, 37), (2, 37), (2, 38), (2, 39), (2, 40), (2, 41), (2, 42), (2, 43), (2, 44), (2, 45), (2, 46), (2, 47), (2, 48), (2, 49), (2, 50), (2, 51), (2, 52), (3, 52), (4, 52), (5, 52), (6, 52), (7, 52), (7, 53), (7, 54), (7, 55), (7, 56), (7, 57), (7, 58), (7, 59), (7, 60), (7, 61), (7, 62), (7, 63), (8, 63), (9, 63), (10, 63), (11, 63), (11, 64), (11, 65), (11, 66), (12, 66), (12, 67), (12, 68), (12, 69), (12, 70), (12, 71), (12, 72), (12, 73), (12, 74), (12, 75), (12, 76), (12, 77), (12, 78), (12, 79), (12, 80), (12, 81), (12, 82), (13, 82), (14, 82), (14, 83), (14, 84), (14, 85), (14, 86), (14, 87), (14, 88), (14, 89), (14, 90), (14, 91), (14, 92), (14, 93), (14, 94), (14, 95), (14, 96), (14, 97), (14, 98), (15, 98), (16, 98), (17, 98), (18, 98), (19, 98), (20, 98), (21, 98), (22, 98), (23, 98), (24, 98), (25, 98), (26, 98), (27, 98), (28, 98), (29, 98), (30, 98), (31, 98), (32, 98), (33, 98), (34, 98), (34, 99), (35, 99), (36, 99), (37, 99), (38, 99), (39, 99), (40, 99), (41, 99), (42, 99), (43, 99), (44, 99), (45, 99), (46, 99), (47, 99), (48, 99), (49, 99), (50, 99), (51, 99), (52, 99), (53, 99), (54, 99), (55, 99), (56, 99), (57, 99), (58, 99), (59, 99), (60, 99), (61, 99), (62, 99), (63, 99), (64, 99), (65, 99), (66, 99), (67, 99), (68, 99), (69, 99), (70, 99), (71, 99), (72, 99), (73, 99), (74, 99), (75, 99), (76, 99), (77, 99), (78, 99), (79, 99), (79, 98), (80, 98), (81, 98), (81, 97), (81, 96), (82, 96)]

time taken for traversing=0.315 second

**3) Euclidean Distance case (2) -**

cost of traversing by distance=181.0

cost stored at end point=181.0

No. of nodes in path=182

No of nodes visited=3893

start point=(4, 7)

end point=(82, 96)

Nodes in path-

[(4, 7), (4, 8), (4, 9), (4, 10), (3, 10), (3, 11), (2, 11), (1, 11), (0, 11), (0, 12), (0, 13), (0, 14), (1, 14), (1, 15), (1, 16), (1, 17), (1, 18), (1, 19), (1, 20), (1, 21), (1, 22), (1, 23), (1, 24), (1, 25), (1, 26), (1, 27), (1, 28), (1, 29), (1, 30), (1, 31), (1, 32), (1, 33), (1, 34), (1, 35), (1, 36), (1, 37), (2, 37), (2, 38), (2, 39), (2, 40), (2, 41), (2, 42), (2, 43), (2, 44), (2, 45), (2, 46), (2, 47), (2, 48), (2, 49), (2, 50), (2, 51), (2, 52), (3, 52), (4, 52), (5, 52), (6, 52), (7, 52), (7, 53), (7, 54), (7, 55), (7, 56), (7, 57), (7, 58), (7, 59), (7, 60), (7, 61), (7, 62), (7, 63), (8, 63), (9, 63), (10, 63), (11, 63), (11, 64), (11, 65), (11, 66), (12, 66), (12, 67), (12, 68), (12, 69), (12, 70), (12, 71), (12, 72), (12, 73), (12, 74), (12, 75), (12, 76), (12, 77), (12, 78), (12, 79), (12, 80), (12, 81), (12, 82), (13, 82), (14, 82), (14, 83), (14, 84), (14, 85), (14, 86), (14, 87), (14, 88), (14, 89), (14, 90), (14, 91), (14, 92), (14, 93), (14, 94), (14, 95), (14, 96), (14, 97), (14, 98), (15, 98), (16, 98), (17, 98), (18, 98), (19, 98), (20, 98), (21, 98), (22, 98), (23, 98), (24, 98), (25, 98), (26, 98), (27, 98), (28, 98), (29, 98), (30, 98), (31, 98), (32, 98), (33, 98), (34, 98), (34, 99), (35, 99), (36, 99), (37, 99), (38, 99), (39, 99), (40, 99), (41, 99), (42, 99), (43, 99), (44, 99), (45, 99), (46, 99), (47, 99), (48, 99), (49, 99), (50, 99), (51, 99), (52, 99), (53, 99), (54, 99), (55, 99), (56, 99), (57, 99), (58, 99), (59, 99), (60, 99), (61, 99), (62, 99), (63, 99), (64, 99), (65, 99), (66, 99), (67, 99), (68, 99), (69, 99), (70, 99), (71, 99), (72, 99), (73, 99), (74, 99), (75, 99), (76, 99), (77, 99), (78, 99), (79, 99), (79, 98), (80, 98), (81, 98), (81, 97), (81, 96), (82, 96)]

time taken for traversing=0.268 second

**4) H1 case (2) -**

cost of traversing by distance=191.0

cost stored at end point=181.0

No. of nodes in path=192

No of nodes visited=1334

start point=(4, 7)

end point=(82, 96)

Nodes in path-

[(4, 7), (4, 8), (5, 8), (5, 9), (6, 9), (6, 10), (7, 10), (7, 11), (8, 11), (9, 11), (9, 10), (9, 9), (9, 8), (10, 8), (11, 8), (12, 8), (13, 8), (14, 8), (15, 8), (16, 8), (17, 8), (18, 8), (18, 9), (18, 10), (18, 11), (18, 12), (18, 13), (18, 14), (18, 15), (18, 16), (18, 17), (19, 17), (20, 17), (21, 17), (21, 18), (21, 19), (21, 20), (21, 21), (21, 22), (21, 23), (21, 24), (22, 24), (23, 24), (24, 24), (25, 24), (26, 24), (26, 25), (26, 26), (26, 27), (26, 28), (26, 29), (26, 30), (27, 30), (28, 30), (28, 31), (28, 32), (28, 33), (28, 34), (28, 35), (28, 36), (29, 36), (30, 36), (30, 37), (31, 37), (32, 37), (33, 37), (33, 38), (33, 39), (33, 40), (33, 41), (33, 42), (34, 42), (35, 42), (35, 43), (36, 43), (36, 44), (37, 44), (37, 45), (38, 45), (38, 46), (39, 46), (39, 47), (39, 48), (40, 48), (40, 49), (41, 49), (41, 50), (42, 50), (42, 51), (43, 51), (44, 51), (44, 52), (45, 52), (46, 52), (47, 52), (47, 53), (48, 53), (49, 53), (49, 54), (50, 54), (51, 54), (51, 55), (52, 55), (53, 55), (53, 56), (53, 57), (54, 57), (55, 57), (56, 57), (57, 57), (58, 57), (59, 57), (60, 57), (60, 58), (60, 59), (60, 60), (60, 61), (60, 62), (60, 63), (60, 64), (60, 65), (60, 66), (60, 67), (61, 67), (62, 67), (63, 67), (64, 67), (65, 67), (66, 67), (67, 67), (68, 67), (68, 68), (68, 69), (68, 70), (67, 70), (66, 70), (66, 71), (66, 72), (66, 73), (66, 74), (66, 75), (67, 75), (68, 75), (68, 76), (68, 77), (68, 78), (68, 79), (68, 80), (68, 81), (68, 82), (67, 82), (66, 82), (65, 82), (64, 82), (63, 82), (62, 82), (61, 82), (61, 83), (61, 84), (61, 85), (62, 85), (63, 85), (64, 85), (65, 85), (66, 85), (66, 86), (66, 87), (66, 88), (66, 89), (67, 89), (68, 89), (69, 89), (69, 90), (69, 91), (69, 92), (70, 92), (71, 92), (72, 92), (73, 92), (74, 92), (75, 92), (76, 92), (76, 93), (76, 94), (76, 95), (76, 96), (77, 96), (78, 96), (79, 96), (80, 96), (81, 96), (82, 96)]

time taken for traversing=0.141second

**5) Manhattan Distance case (2) -**

cost of traversing by distance=181.0

cost stored at end point=181.0

No. of nodes in path=182

No of nodes visited=2907

start point=(4, 7)

end point=(82, 96)

Nodes in path-

[(4, 7), (4, 8), (4, 9), (4, 10), (3, 10), (3, 11), (2, 11), (1, 11), (0, 11), (0, 12), (0, 13), (0, 14), (1, 14), (1, 15), (1, 16), (1, 17), (1, 18), (1, 19), (1, 20), (1, 21), (1, 22), (1, 23), (1, 24), (1, 25), (1, 26), (1, 27), (1, 28), (1, 29), (1, 30), (1, 31), (1, 32), (1, 33), (1, 34), (1, 35), (1, 36), (1, 37), (2, 37), (2, 38), (2, 39), (2, 40), (2, 41), (2, 42), (2, 43), (2, 44), (2, 45), (2, 46), (2, 47), (2, 48), (2, 49), (2, 50), (2, 51), (2, 52), (3, 52), (4, 52), (5, 52), (6, 52), (7, 52), (7, 53), (7, 54), (7, 55), (7, 56), (7, 57), (7, 58), (7, 59), (7, 60), (7, 61), (7, 62), (7, 63), (8, 63), (9, 63), (10, 63), (11, 63), (11, 64), (11, 65), (11, 66), (12, 66), (12, 67), (12, 68), (12, 69), (12, 70), (12, 71), (12, 72), (12, 73), (12, 74), (12, 75), (12, 76), (12, 77), (12, 78), (12, 79), (12, 80), (12, 81), (12, 82), (13, 82), (14, 82), (14, 83), (14, 84), (14, 85), (14, 86), (14, 87), (14, 88), (14, 89), (14, 90), (14, 91), (14, 92), (14, 93), (14, 94), (14, 95), (14, 96), (14, 97), (14, 98), (15, 98), (16, 98), (17, 98), (18, 98), (19, 98), (20, 98), (21, 98), (22, 98), (23, 98), (24, 98), (25, 98), (26, 98), (27, 98), (28, 98), (29, 98), (30, 98), (31, 98), (32, 98), (33, 98), (34, 98), (34, 99), (35, 99), (36, 99), (37, 99), (38, 99), (39, 99), (40, 99), (41, 99), (42, 99), (43, 99), (44, 99), (45, 99), (46, 99), (47, 99), (48, 99), (49, 99), (50, 99), (51, 99), (52, 99), (53, 99), (54, 99), (55, 99), (56, 99), (57, 99), (58, 99), (59, 99), (60, 99), (61, 99), (62, 99), (63, 99), (64, 99), (65, 99), (66, 99), (67, 99), (68, 99), (69, 99), (70, 99), (71, 99), (72, 99), (73, 99), (74, 99), (75, 99), (76, 99), (77, 99), (78, 99), (79, 99), (79, 98), (80, 98), (81, 98), (81, 97), (81, 96), (82, 96)]

time taken for traversing=0.19 second

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1) Diagonal Distance case (2) -**

Using a different cost function

cost of traversing by distance=185.0

cost stored at end point=185

No. of nodes in path=182

No of nodes visited=3711

start point=(4, 7)

end point=(82, 96)

Nodes in path-

[(4, 7), (4, 8), (4, 9), (4, 10), (3, 10), (2, 10), (1, 10), (0, 10), (0, 11), (0, 12), (0, 13), (0, 14), (0, 15), (0, 16), (0, 17), (0, 18), (0, 19), (0, 20), (0, 21), (0, 22), (0, 23), (0, 24), (0, 25), (0, 26), (0, 27), (0, 28), (0, 29), (0, 30), (0, 31), (0, 32), (0, 33), (0, 34), (0, 35), (0, 36), (0, 37), (0, 38), (0, 39), (0, 40), (0, 41), (0, 42), (0, 43), (0, 44), (0, 45), (0, 46), (0, 47), (0, 48), (0, 49), (0, 50), (0, 51), (0, 52), (0, 53), (0, 54), (0, 55), (0, 56), (0, 57), (0, 58), (0, 59), (0, 60), (0, 61), (0, 62), (0, 63), (0, 64), (0, 65), (0, 66), (0, 67), (0, 68), (0, 69), (0, 70), (0, 71), (0, 72), (0, 73), (0, 74), (0, 75), (0, 76), (0, 77), (0, 78), (0, 79), (0, 80), (0, 81), (0, 82), (0, 83), (0, 84), (0, 85), (0, 86), (0, 87), (0, 88), (0, 89), (0, 90), (0, 91), (0, 92), (0, 93), (0, 94), (0, 95), (0, 96), (0, 97), (0, 98), (0, 99), (1, 99), (2, 99), (3, 99), (4, 99), (5, 99), (6, 99), (7, 99), (8, 99), (9, 99), (10, 99), (11, 99), (12, 99), (13, 99), (14, 99), (15, 99), (16, 99), (17, 99), (18, 99), (19, 99), (20, 99), (21, 99), (22, 99), (23, 99), (24, 99), (25, 99), (26, 99), (27, 99), (28, 99), (29, 99), (30, 99), (31, 99), (32, 99), (33, 99), (34, 99), (35, 99), (36, 99), (37, 99), (38, 99), (39, 99), (40, 99), (41, 99), (42, 99), (43, 99), (44, 99), (45, 99), (46, 99), (47, 99), (48, 99), (49, 99), (50, 99), (51, 99), (52, 99), (53, 99), (54, 99), (55, 99), (56, 99), (57, 99), (58, 99), (59, 99), (60, 99), (61, 99), (62, 99), (63, 99), (64, 99), (65, 99), (66, 99), (67, 99), (68, 99), (69, 99), (70, 99), (71, 99), (72, 99), (73, 99), (74, 99), (75, 99), (76, 99), (77, 99), (78, 99), (79, 99), (80, 99), (81, 99), (82, 99), (82, 98), (82, 97), (82, 96)]

time taken for traversing=0.238 second

**2) Dijkstra case (2) -**

Using a different cost function

cost of traversing by distance=184.0

cost stored at end point=184

No. of nodes in path=182

No of nodes visited=4718

start point=(4, 7)

end point=(82, 96)

Nodes in path-

[(4, 7), (3, 7), (2, 7), (1, 7), (0, 7), (0, 8), (0, 9), (0, 10), (0, 11), (0, 12), (0, 13), (0, 14), (0, 15), (0, 16), (0, 17), (0, 18), (0, 19), (0, 20), (0, 21), (0, 22), (0, 23), (0, 24), (0, 25), (0, 26), (0, 27), (0, 28), (0, 29), (0, 30), (0, 31), (0, 32), (0, 33), (0, 34), (0, 35), (0, 36), (0, 37), (0, 38), (0, 39), (0, 40), (0, 41), (0, 42), (0, 43), (0, 44), (0, 45), (0, 46), (0, 47), (0, 48), (0, 49), (0, 50), (0, 51), (0, 52), (0, 53), (0, 54), (0, 55), (0, 56), (0, 57), (0, 58), (0, 59), (0, 60), (0, 61), (0, 62), (0, 63), (0, 64), (0, 65), (0, 66), (0, 67), (0, 68), (0, 69), (0, 70), (0, 71), (0, 72), (0, 73), (0, 74), (0, 75), (0, 76), (0, 77), (0, 78), (0, 79), (0, 80), (0, 81), (0, 82), (0, 83), (0, 84), (0, 85), (0, 86), (0, 87), (0, 88), (0, 89), (0, 90), (0, 91), (0, 92), (0, 93), (0, 94), (0, 95), (0, 96), (0, 97), (0, 98), (0, 99), (1, 99), (2, 99), (3, 99), (4, 99), (5, 99), (6, 99), (7, 99), (8, 99), (9, 99), (10, 99), (11, 99), (12, 99), (13, 99), (14, 99), (15, 99), (16, 99), (17, 99), (18, 99), (19, 99), (20, 99), (21, 99), (22, 99), (23, 99), (24, 99), (25, 99), (26, 99), (27, 99), (28, 99), (29, 99), (30, 99), (31, 99), (32, 99), (33, 99), (34, 99), (35, 99), (36, 99), (37, 99), (38, 99), (39, 99), (40, 99), (41, 99), (42, 99), (43, 99), (44, 99), (45, 99), (46, 99), (47, 99), (48, 99), (49, 99), (50, 99), (51, 99), (52, 99), (53, 99), (54, 99), (55, 99), (56, 99), (57, 99), (58, 99), (59, 99), (60, 99), (61, 99), (62, 99), (63, 99), (64, 99), (65, 99), (66, 99), (67, 99), (68, 99), (69, 99), (70, 99), (71, 99), (72, 99), (73, 99), (74, 99), (75, 99), (76, 99), (77, 99), (78, 99), (79, 99), (80, 99), (81, 99), (82, 99), (82, 98), (82, 97), (82, 96)]

time taken for traversing=0.292 second

**3) Euclidean Distance case(2) -**

Using a different cost function

cost of traversing by distance=185.0

cost stored at end point=185.0

No. of nodes in path=182

No of nodes visited=3573

start point=(4, 7)

end point=(82, 96)

Nodes in path-

[(4, 7), (4, 8), (4, 9), (4, 10), (3, 10), (2, 10), (1, 10), (0, 10), (0, 11), (0, 12), (0, 13), (0, 14), (0, 15), (0, 16), (0, 17), (0, 18), (0, 19), (0, 20), (0, 21), (0, 22), (0, 23), (0, 24), (0, 25), (0, 26), (0, 27), (0, 28), (0, 29), (0, 30), (0, 31), (0, 32), (0, 33), (0, 34), (0, 35), (0, 36), (0, 37), (0, 38), (0, 39), (0, 40), (0, 41), (0, 42), (0, 43), (0, 44), (0, 45), (0, 46), (0, 47), (0, 48), (0, 49), (0, 50), (0, 51), (0, 52), (0, 53), (0, 54), (0, 55), (0, 56), (0, 57), (0, 58), (0, 59), (0, 60), (0, 61), (0, 62), (0, 63), (0, 64), (0, 65), (0, 66), (0, 67), (0, 68), (0, 69), (0, 70), (0, 71), (0, 72), (0, 73), (0, 74), (0, 75), (0, 76), (0, 77), (0, 78), (0, 79), (0, 80), (0, 81), (0, 82), (0, 83), (0, 84), (0, 85), (0, 86), (0, 87), (0, 88), (0, 89), (0, 90), (0, 91), (0, 92), (0, 93), (0, 94), (0, 95), (0, 96), (0, 97), (0, 98), (0, 99), (1, 99), (2, 99), (3, 99), (4, 99), (5, 99), (6, 99), (7, 99), (8, 99), (9, 99), (10, 99), (11, 99), (12, 99), (13, 99), (14, 99), (15, 99), (16, 99), (17, 99), (18, 99), (19, 99), (20, 99), (21, 99), (22, 99), (23, 99), (24, 99), (25, 99), (26, 99), (27, 99), (28, 99), (29, 99), (30, 99), (31, 99), (32, 99), (33, 99), (34, 99), (35, 99), (36, 99), (37, 99), (38, 99), (39, 99), (40, 99), (41, 99), (42, 99), (43, 99), (44, 99), (45, 99), (46, 99), (47, 99), (48, 99), (49, 99), (50, 99), (51, 99), (52, 99), (53, 99), (54, 99), (55, 99), (56, 99), (57, 99), (58, 99), (59, 99), (60, 99), (61, 99), (62, 99), (63, 99), (64, 99), (65, 99), (66, 99), (67, 99), (68, 99), (69, 99), (70, 99), (71, 99), (72, 99), (73, 99), (74, 99), (75, 99), (76, 99), (77, 99), (78, 99), (79, 99), (80, 99), (81, 99), (82, 99), (82, 98), (82, 97), (82, 96)]

time taken for traversing=0.244 second

**4) H1 case(2) -**

Using a different cost function

cost of traversing by distance=224.0

cost stored at end point=214

No. of nodes in path=192

No of nodes visited=1337

start point=(4, 7)

end point=(82, 96)

Nodes in path-

[(4, 7), (4, 8), (4, 9), (4, 10), (5, 10), (6, 10), (7, 10), (7, 11), (8, 11), (9, 11), (10, 11), (11, 11), (11, 10), (11, 9), (11, 8), (12, 8), (13, 8), (14, 8), (15, 8), (16, 8), (17, 8), (18, 8), (18, 9), (18, 10), (18, 11), (18, 12), (18, 13), (18, 14), (18, 15), (18, 16), (18, 17), (19, 17), (20, 17), (21, 17), (21, 18), (21, 19), (21, 20), (21, 21), (21, 22), (21, 23), (21, 24), (22, 24), (23, 24), (24, 24), (25, 24), (26, 24), (27, 24), (28, 24), (28, 25), (28, 26), (28, 27), (28, 28), (28, 29), (28, 30), (28, 31), (28, 32), (28, 33), (28, 34), (28, 35), (28, 36), (28, 37), (29, 37), (30, 37), (31, 37), (32, 37), (33, 37), (33, 38), (33, 39), (33, 40), (33, 41), (33, 42), (33, 43), (33, 44), (33, 45), (33, 46), (33, 47), (33, 48), (34, 48), (35, 48), (36, 48), (37, 48), (38, 48), (39, 48), (40, 48), (41, 48), (42, 48), (43, 48), (44, 48), (45, 48), (46, 48), (47, 48), (48, 48), (49, 48), (50, 48), (51, 48), (51, 49), (51, 50), (51, 51), (51, 52), (51, 53), (51, 54), (51, 55), (52, 55), (53, 55), (54, 55), (55, 55), (56, 55), (56, 56), (56, 57), (57, 57), (58, 57), (59, 57), (60, 57), (60, 58), (60, 59), (60, 60), (60, 61), (60, 62), (60, 63), (60, 64), (60, 65), (60, 66), (60, 67), (61, 67), (62, 67), (63, 67), (64, 67), (65, 67), (66, 67), (67, 67), (68, 67), (68, 68), (68, 69), (68, 70), (68, 71), (67, 71), (66, 71), (66, 72), (66, 73), (66, 74), (66, 75), (67, 75), (68, 75), (68, 76), (68, 77), (68, 78), (68, 79), (68, 80), (68, 81), (68, 82), (67, 82), (66, 82), (65, 82), (64, 82), (63, 82), (62, 82), (61, 82), (61, 83), (61, 84), (61, 85), (62, 85), (63, 85), (64, 85), (65, 85), (66, 85), (66, 86), (66, 87), (66, 88), (66, 89), (67, 89), (68, 89), (69, 89), (69, 90), (69, 91), (69, 92), (70, 92), (71, 92), (72, 92), (73, 92), (74, 92), (75, 92), (76, 92), (76, 93), (76, 94), (76, 95), (76, 96), (77, 96), (78, 96), (79, 96), (80, 96), (81, 96), (82, 96)]

time taken for traversing=0.4 second

**5) Manhattan Distance case (2) -**

Using a different cost function

cost of traversing by distance=185.0

cost stored at end point=185

No. of nodes in path=182

No. of nodes visited=2677

start point=(4, 7)

end point=(82, 96)

Nodes in path-

[(4, 7), (4, 8), (4, 9), (4, 10), (3, 10), (2, 10), (1, 10), (0, 10), (0, 11), (0, 12), (0, 13), (0, 14), (0, 15), (0, 16), (0, 17), (0, 18), (0, 19), (0, 20), (0, 21), (0, 22), (0, 23), (0, 24), (0, 25), (0, 26), (0, 27), (0, 28), (0, 29), (0, 30), (0, 31), (0, 32), (0, 33), (0, 34), (0, 35), (0, 36), (0, 37), (0, 38), (0, 39), (0, 40), (0, 41), (0, 42), (0, 43), (0, 44), (0, 45), (0, 46), (0, 47), (0, 48), (0, 49), (0, 50), (0, 51), (0, 52), (0, 53), (0, 54), (0, 55), (0, 56), (0, 57), (0, 58), (0, 59), (0, 60), (0, 61), (0, 62), (0, 63), (0, 64), (0, 65), (0, 66), (0, 67), (0, 68), (0, 69), (0, 70), (0, 71), (0, 72), (0, 73), (0, 74), (0, 75), (0, 76), (0, 77), (0, 78), (0, 79), (0, 80), (0, 81), (0, 82), (0, 83), (0, 84), (0, 85), (0, 86), (0, 87), (0, 88), (0, 89), (0, 90), (0, 91), (0, 92), (0, 93), (0, 94), (0, 95), (0, 96), (0, 97), (0, 98), (0, 99), (1, 99), (2, 99), (3, 99), (4, 99), (5, 99), (6, 99), (7, 99), (8, 99), (9, 99), (10, 99), (11, 99), (12, 99), (13, 99), (14, 99), (15, 99), (16, 99), (17, 99), (18, 99), (19, 99), (20, 99), (21, 99), (22, 99), (23, 99), (24, 99), (25, 99), (26, 99), (27, 99), (28, 99), (29, 99), (30, 99), (31, 99), (32, 99), (33, 99), (34, 99), (35, 99), (36, 99), (37, 99), (38, 99), (39, 99), (40, 99), (41, 99), (42, 99), (43, 99), (44, 99), (45, 99), (46, 99), (47, 99), (48, 99), (49, 99), (50, 99), (51, 99), (52, 99), (53, 99), (54, 99), (55, 99), (56, 99), (57, 99), (58, 99), (59, 99), (60, 99), (61, 99), (62, 99), (63, 99), (64, 99), (65, 99), (66, 99), (67, 99), (68, 99), (69, 99), (70, 99), (71, 99), (72, 99), (73, 99), (74, 99), (75, 99), (76, 99), (77, 99), (78, 99), (79, 99), (80, 99), (81, 99), (82, 99), (82, 98), (82, 97), (82, 96)]

time taken for traversing=0.219 second

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Task 1: Program**

#TASK-1 ASTAR

#Done By Kaushal Jadhav

#Roll No.-20EC30019

#First Year Undergraduate,Dept. of Electronics and Electrical Communication Engineering, IIT Kharagpur

#ASTAR is a further modification and more advanced than Dijkstra

#It uses Heuristics for time-efficient path planning

#import necessary libraries

import numpy as np

# Numpy library for handling matrices efficiently

import cv2

# Open-CV to dislay image

import collections

#To use the default-dict

import heapq

#To take advantage of fast and efficient methods

'''

Why use heap?

The heap implementation ensures time complexity is logarithmic. Thus push/pop operations are

proportional to  the base-2 Logarithm of number of elements.

Implementation is through a binary tree (Just like a sieve!)

Heap property: Value of node is always smaller than both of its children unlike a binary search tree.

'''

import queue

import time

#to measure time

import pandas as pd

#  to use ToNumeric method

'''

Why Priority Queue?

Because here we have to decide the priority. So what if we had a queue that adjusts the priority for us!

Earlier attempt was to use a list/collection of nodes and find out the minimum. But the time complexity increases in that case.

'''

#Appendix:

#Diagonal Distance=1

#Dijkstra=2

#Euclidean=3

#H1=4   ...A non-admissible Heuristic.

#Manhattan=5

#Appendix:

choice=4 # Heuristic\_type

choice2=2 # Travel\_selection

'''

Choice2=1 means diagonal movement allowed

Choice2=2 means diagonal movement not allowed

'''

choice3=2  # Cost\_selection

'''

Choice3=1 means normal given cost

Choice3=2 valid for only Choice2=2

It is a cost method developed assuming that a bot takes some time to turn.

It is designed so that the bot will prefer to go straight (According to the path along which it has entered the current node)

'''

heuristic=['Diagonal Distance','Dijkstra','Euclidean','H1','Manhattan']

#For documentation

write\_path='a a astar.txt'

#Read Image Path

img\_path='Task\_1\_Low.png'

#Write Image Path

img\_write\_path='AStar\_'+str(choice)+'\_'+heuristic[choice-1]+'\_'+'Case-'+str(choice2)

if(choice2==2 and choice3==2):

    img\_write\_path=img\_write\_path+'\_2'

img\_write\_path=img\_write\_path+'.png'

#Written as variables so that making changes in code according to the need become easy

wait=10000

#Define color parameters

start\_color = [113,204,45]

end\_color = [60,76,231]

ob\_color = [255,255,255]

np\_color =[0,0,0]

visited\_color=[100,0,100]

path\_pointer=[0,255,0]

#Read Image

img = cv2.imread(img\_path,cv2.IMREAD\_COLOR)

#Calculate image size before search

h,w,c = img.shape

#Find start point and end point

#Here used a break\_point in finding end\_point

b=False

for i in range(h):

    for j in range(w):

        if(b):

            break

        if (img[i,j,0] == start\_color[0] and img[i,j,1]==start\_color[1] and img[i,j,2]==start\_color[2]):

          start = (i,j)

        if (img[i,j,0] == end\_color[0] and img[i,j,1] == end\_color[1] and img[i,j,2] == end\_color[2] ):

          end = (i,j)

          b=True

    if(b):

        break

#Calculate Heuristic Func

def calcHeuristic(point1,point2=None,startpt=None,endpt=None):

    '''

 Calculates Heuristic Function according to the global choice selected.\n

 Calculates the Heuristic between points point1 and point2

 choice==1 Diagonal Distance\n

 returns max(abs(point1[0] - point2[0]),abs(point1[1] - point2[1]))\n

 choice==2 Dijkstra\n

 returns 0 (No Heuristic)\n

 Choice==3 Euclidean=3\n

 returns ((point1[0] - point2[0])\*\*2 + (point1[1] - point2[1])\*\*2)\*\*0.5\n

 Can delete \*\*0.5 according to need\n

 Choice==4 H1   ...A non-admissible Heuristic.\n

 H1 is a non-Admissible Heuristic.\n Based on distance between point and a line.\nWorks well with no obstacles\n

 Choice==5 Manhattan\n

 returns abs(point1[0] - point2[0]) + abs(point1[1] - point2[1])\n

 @param startpt and endpt required for H1\n

    '''

    if(choice==1):

        return max(abs(point1[0] - point2[0]),abs(point1[1] - point2[1]))

    if(choice==2):

        return 0

    if(choice==3):

        return ((point1[0] - point2[0])\*\*2 + (point1[1] - point2[1])\*\*2)\*\*0.5

    if(choice==4):

        a=start[1]-end[1]

        b=end[0]-start[0]

        c=start[0]\*end[1]-start[1]\*end[0]

        if(a\*point1[0]+b\*point1[1]+c==0):

            return -10

        return abs(a\*point1[0]+b\*point1[1]+c)

    if(choice==5):

        return abs(point1[0] - point2[0]) + abs(point1[1] - point2[1])

def calcCost(point1,point2):

    '''

    Calculates cost

    '''

    if(abs(point1[0]-point2[0])==1 and abs(point1[1]-point2[1])==0):

        return 1.0

    if(abs(point1[0]-point2[0])==1 and abs(point1[1]-point2[1])==1):

        return 1.41421356237      #define square root of 2

    if(abs(point1[0]-point2[0])==0 and abs(point1[1]-point2[1])==1):

        return 1.0

    if(abs(point1[0]-point2[0])==0 and abs(point1[1]-point2[1])==0):

        return 0.0

    else:

      return np.inf

def calcCost2(child,current,start,parent=None):

    '''

    Special cost function\n

    based on real runtime situation.\n

    Accounts time taken for turning by the bot\n

    '''

    point1=child.position

    point2=current.position

    start\_point=start.position

    if not (point2[0]==start\_point[0] and point2[1]==start\_point[1]):

        parent\_point=parent.position

        if(float(point2[0])==(point1[0]+parent\_point[0])/2 and float(point2[1])==(point1[1]+parent\_point[1])/2 ):

            return 1

        if(parent\_point[0]==point1[0] and parent\_point[1]==point1[1]):

            return 3

        if(point1[0]==point2[0] and point1[1]==point2[1]):

            return 0

        else:

            return 2

    else:

     if(abs(point1[0]-point2[0])==1 and abs(point1[1]-point2[1])==0):

         return 1

     if(abs(point1[0]-point2[0])==0 and abs(point1[1]-point2[1])==1):

         return 1

     if(abs(point1[0]-point2[0])==0 and abs(point1[1]-point2[1])==0):

         return 0

#check for navigation path

def isinrange(img,position):

    '''

    Returns False if given node is not accessible

    \nElse returns True

    '''

    b=False

    x=position[0]

    y=position[1]

    ob\_color = [255,255,255]

    if(x>=0 and x<img.shape[0] and y >=0 and y<img.shape[1]):

        b=True

        if(img[x,y,0] ==ob\_color[0] and img[x,y,1] ==ob\_color[1] and img[x,y,2] ==ob\_color[2]):

           b=False

    return b

# class for handling priority queues

class PriorityQueue :

    '''

    creates a Queue Class

    '''

    def \_\_init\_\_(self):

        '''

        Creates a queue

        '''

        self.Queue=[]

    def isempty(self):

        '''

        Checks if given Queue is empty.

        \nReturns True if Empty

        '''

        if not self.Queue:

            return 1

        else:

             return 0

    def put(self,index):

        '''

        Puts given element onto the queue

        \nUses heapq.heappush for faster aproach.

        '''

        heapq.heappush(self.Queue,index)

    def get(self):

        '''

        Returns the smallest-in-priority element using heapq.heappop

        \n\nComparison based on \_\_lt\_\_ and \_\_gt\_\_ of node class

        '''

        #print(self.Queue)

        return heapq.heappop(self.Queue)

class node:

    '''

    class for handling nodes\n

    @param index=input default=None\nstands for position\n

    @param is\_in\_list and is\_current are check-points\n

    @param parent=input default=None\n

    @param f,g,h=cost\n

    @param h stands for Heuristic cost\n

    @param f stands for total cost

    '''

    #constructor

    def \_\_init\_\_(self,index=None,parent=None):

        # Initialise params

        self.position=index  #position/location

        self.is\_in\_list=False  # chkpt

        self.is\_current=False  #chkpt

        self.parent=parent     #parent node of the node

                                  # \_\_

        self.f=np.inf             #   |

        self.g=np.inf             #   | initialise cost

        self.h=np.inf             # \_\_|

        self.isvisted=False

    def \_\_lt\_\_(self,other):

     '''

     overload operator < (chk parameter=cost)

     '''

     return self.f<other.f

    def \_\_gt\_\_(self,other):

     '''

     overload operator > (chk parameter=cost)

     '''

     return self.f>other.f

#get neighbourhood according to choice2

def get\_nbd(current):

    '''

    returns list of neighbourhood positions

    \nchoice is made based on choice2

    '''

    l=[]

    if(choice2==1):

     for i in range(-1,2):

         for j in range(-1,2):

             position=(current.position[0]+i,current.position[1]+j)

             if (isinrange(img,position)==True):

                 l.append(position)

    if(choice2==2):

     for i in range(-1,2):

         position=(current.position[0]+i,current.position[1])

         if (isinrange(img,position)==True):

                l.append(position)

     for j in range(-1,2):

         if(j==0): #skip repetition

             continue

         position=(current.position[0],current.position[1]+j)

         if (isinrange(img,position)==True):

             l.append(position)

    return l

# create a matrix/dict of objects

node\_matrix=np.empty((h,w),dtype=object)

#node\_matrix=collections.defaultdict(node)

for i in range(h):

    for j in range(w):

        node\_matrix[i,j]=node()

        node\_matrix[i,j].position=(i,j)

#main function

def main\_func(img,startpt,endpt):

    '''

    The main traversing function\n

    returns time taken for traversing

    '''

    #create lists

    visited=[]

    path=[]

    h,w,c=img.shape

    #create priority queue

    pt\_list=PriorityQueue()

    #Initialise start node

    start=node(startpt)

    start.g=0

    start.f=start.h=calcHeuristic(start.position,endpt)

    start.isvisted=True

    node\_matrix[startpt]=start

    pt\_list.put(start)

    #Start Traversing

    beg=time.time()

    while(not pt\_list.isempty()):

        #while the list is not empty obtain the node with smallest f

        current=pt\_list.get()

        #Now that current node is not in the list change the corresponding chkpts

        current.is\_in\_list=False

        current.is\_current=True

        #Now to preserve the node use the matrix

        node\_matrix[current.position]=current

        #break condition

        if(current.position==endpt):

            break

        #get the neighbourhood points

        nbd\_list=get\_nbd(current)

        # searching operation

        for pos in nbd\_list:

            # earlier attempt was to use a list of objects. But finally dealing with position.

            nbd=node\_matrix[pos]

            #get the temp\_cost

            if(choice2==2 and choice3==2):

                g\_temp=current.g+calcCost2(nbd,current,start,current.parent)

            else:

                g\_temp=current.g+calcCost(pos,current.position)

            # if this newly calculated cost< the stored cost-

            if(g\_temp<nbd.g):

                #make note that it is visited

                nbd.isvisted=True

                #if the point is current and the new cost is less than its stored cost-

                #make the current as it's parent and put the nbd node in the pt\_list.

                if nbd.is\_current:

                    nbd.is\_current=False

                    nbd.parent=current

                    nbd.g=g\_temp

                    nbd.h=calcHeuristic(nbd.position,endpt)

                    nbd.f=nbd.g+nbd.h

                    nbd.is\_in\_list=True

                    pt\_list.put(nbd)

                # if the nbd node is not current- then put the nbd node in the pt\_list (if it is not there) along with setting the chk\_points.

                else:

                    nbd.parent=current

                    nbd.g=g\_temp

                    nbd.h=calcHeuristic(nbd.position,endpt)

                    nbd.f=nbd.g+nbd.h

                    if(not nbd.is\_in\_list):

                        pt\_list.put(nbd)

                    nbd.is\_in\_list=True

                    nbd.is\_current=False

        # to display progress

        #showPath(img,current,start=start,is\_end=False)

        #cv2.waitKey(1)

    # finish traversing

    finish=time.time()

    print(round(finish-beg,3))

    # display final path

    visited,path,cost=showPath(img,current,True,start,visited,path)

    path.reverse()

    #end with saving the results in a txt document

    documentation(visited,path,finish-beg,cost,start,current,1)

    return finish-beg

def showPath(img,current,is\_end,start,visited\_list=None,parent\_list=None):

    '''

    displays current progress as image if is\_end is disabled\n

    if is\_end is enabled-\n

    returns visited list and parent list and calculates cost

    '''

    cost=0.0

    visited\_color=[100,0,100]

    path\_pointer=[0,255,0]

    # to avoid errors

    if(parent\_list==None):

        parent\_list=[]

    if(visited\_list==None):

        visited\_list=[]

    img2=np.copy(img)

    # display/compile visited path

    for i in range(node\_matrix.shape[0]):

        for j in range(node\_matrix.shape[1]):

            if(node\_matrix[i,j].isvisted==True):

              img2[i,j,0]=visited\_color[0]

              img2[i,j,1]=visited\_color[1]

              img2[i,j,2]=visited\_color[2]

              if(is\_end):

                  visited\_list.append((i,j))

    while(current.position!=start.position):

        #display image

        temp=current.position

        img2[temp[0],temp[1],0]=path\_pointer[0]

        img2[temp[0],temp[1],1]=path\_pointer[1]

        img2[temp[0],temp[1],2]=path\_pointer[2]

        if(not is\_end):

            current=current.position

            continue

        else:

         parent\_list.append(temp)

         temp=current.parent

         #calculate cost

         if(choice2==2 and choice3==2):

             cost=cost+calcCost2(current,temp,start,temp.parent)

         else:

             cost=cost+calcCost(current.position,temp.position)

         current=temp

    if(is\_end):

     parent\_list.append(start.position)

    #imshow and imwrite

    cv2.resize(img2,(1000,1000))

    #cv2.namedWindow('path',cv2.WINDOW\_NORMAL)

    #cv2.imshow('path',img2)

    if(is\_end):

        cv2.imwrite(img\_write\_path,img2)

    #return the lists and the cost

    return (visited\_list,parent\_list,cost)

# display option changed according to option

def display(val,option,w=None):

    '''

    write data according to option specified

    '''

    if(option==0):

        print(val)

    else:

        w.write(val)

        w.write('\n')

# Documentation according to option

def documentation(visited\_list,parent\_list,val,cost,start,end,option):

    '''

    document the data

    '''

    if(option==1):

     #open desired file to write result-

     w=open(write\_path,'w')

     #write the data

    display(str(choice)+')'+heuristic[choice-1]+' case('+str(choice2)+') -',option,w)

    if(choice2==2 and choice3==2):

        display('Using a different cost function',option,w)

    display('cost of traversing by distance='+str(round(cost,2)),option,w)

    display('cost stored at end point='+str(round(end.f,2)),option,w)

    display('No. of nodes in path='+str(len(parent\_list)),option,w)

    display('No of nodes visited='+str(len(visited\_list)),option,w)

    display('start point='+str(start.position),option,w)

    display('end point='+str(end.position),option,w)

    display('Nodes in path-',option,w)

    display(str(parent\_list),option,w)

    #display('Nodes visited-',option,w)

    #display(str(visited\_list),option,w)

    display('time taken for traversing='+str(val),option,w)

    if(option==1):

     #close file

     w.close()

#         \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Driver Code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

main\_func(img,start,end)

#cv2.waitKey(wait)

#         \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'''

The heuristics-

diagonal distance- max(abs(point1[0] - point2[0]),abs(point1[1] - point2[1]))

So it is using max(|x1-x2|,|y1-y2|)

djikstra- Has no heuristic

euclidean- ((point1[0] - point2[0])\*\*2 + (point1[1] - point2[1])\*\*2)\*\*0.5  (yes, the very old equation form the school days!)

H1 - It is a non-admissible heuristic developed by me

What it does is that it checks how far you are from the line joining start point and end point. It forces to follow the line

Based on the fact that a line is the shortest path ( in plane-geom)

Now it may not be the case when there are exceptions in the form of obstacles

    a=startpt[1]-endpt[1]

    b=endpt[0]-startpt[0]

    c=startpt[0]\*endpt[1]-startpt[1]\*endpt[0]

    if(a\*point[0]+b\*point[1]+c==0):

        return -10

    return abs(a\*point[0]+b\*point[1]+c)

So as it is evident it will force to follow the line joining startpt and endpt.

Manhattan- also called city-block distance

abs(point1[0] - point2[0]) + abs(point1[1] - point2[1])

or |x1-x2|+|y1-y2|

'''

'''

Note about the cost function-

In real life a bot will take some time to turn. The actual time taken will depend on various parameters

I have assumed the change in state ( from travelling straight to the state of turning to the left/right) as taking 0-time delay

So

if deviation=90 degrees returns 2

if deviation=0 degrees either will return 1 or 0

if deviation is 180 degrees returns 3

'''

#references-

'''

1) http://robotics.caltech.edu/wiki/images/e/e0/Astar.pdf

2) https://cs.stanford.edu/people/eroberts/courses/soco/projects/2003-04/intelligent-search/astar.html

'''

**TASK 3**

**Task-3**

**Kalman Filter**

**Introduction**

In this AI-driven world, self-driving cars are one of the current hot topics that the world speaks of. For a self-driving car, it is very important to know where you are and how fast you are moving at all times with high precision. So, we get a state estimate with the help of the kinematics equations (which we are very much familiar with) and then we compare our calculations with the measured data. **But in practice the calculated results do not always match the actual ground measurements.**

To overcome this problem, Kalman filter is used. It is an algorithm that uses a series of time-variant measurements containing various uncertainties and produces estimates of the unknown variables. It uses a joint probability distribution over the variables for each timeframe. Each step is considered to be operating on a linear system. The filter algorithm uses a recursive approach.

The errors are considered as multivariate normal distribution with mean=0. Two types of noise are considered:

1. Process Noise

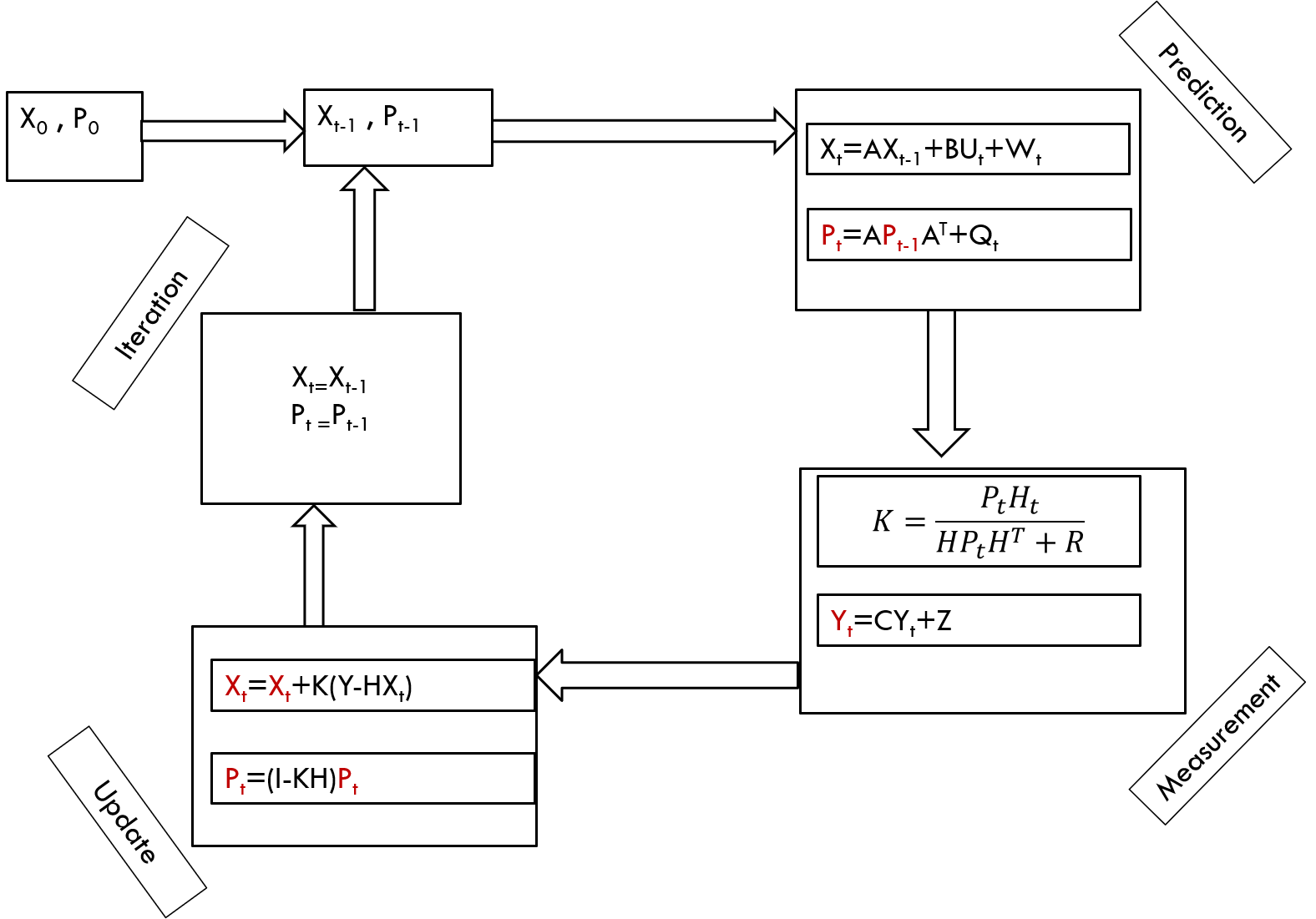
2. System Noise

**The general overview of the process-**

Here, predicted state is generated based on previous state

The final state is calculated using the prediction and the current ground measurement.

**Flowchart**

****

The flowchart shows main three steps in the implementation of Kalman filter are Prediction, Measurement and Update.

**Algorithm**

The Kalman filter algorithm can be divided into four parts:

1. The initial state is recorded at the time when the sensors begin to give the readings.
2. Prediction- Based on previous knowledge of a vehicle position and kinematic equations, the position of vehicle is predicted.
3. Measurement- Ground measurements of the position and the velocity of vehicle are obtained and further compared with the predicted state obtained in step1.
4. Update - Finally the knowledge about the position (or state) of vehicle is updated based on the prediction and sensor readings.

Again, repeat through the step 2.

**Drawback and further modifications**

The Kalman filter is a linear filter. So, it will fail for non-linear systems. In case if the prediction equation is not linear, then the other variants of Kalman Filter are used. The commonly used variants are-

1. Extended Kalman filter

2. Unscented Kalman filter

In the extended Kalman filter the linear function is replaced by a non-linear function and the covariance gets replaced by Jacobians.

The extended Kalman filter is found to give poor performance when the functions are highly non-linear. In that case the Unscented Kalman filter is used.

**Applications of Kalman Filter are as follows**

1. Kalman filters are extensively used in the autonomous world.
2. A common application is in navigation and control of vehicles, particularly aircraft, spacecraft and dynamically controlled ships.
3. The Kalman filter stands out of other techniques because of its dynamic structure. It considers errors, noise, external influence unlike other techniques.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Task 3- Program with given Input and Obtained Output**

# Kalman Filter implementation

# Done by - Kaushal Jadhav

# Roll No. 20EC30019

import numpy as np

import cv2 as cv

cv.namedWindow('plot',cv.WINDOW\_NORMAL)

img=np.full((1000,1000),fill\_value=0,dtype=np.uint8)

# create the image where path will be plotted

# The following is just for presentation purpose-

def inrange(img,x,y):

    if x>=0 and x<img.shape[0] and y>=0 and y<img.shape[1]:

        return True

    else:

        return False

#  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# define param

t=1

sigma\_x=1

sigma\_y=1

# create various matrices and perform the operations of the filter

def create\_A(m):

    calc\_m=m

    calc\_m[0,0]=calc\_m[0,0]+m[1,0]\*t

    calc\_m[1,0]=calc\_m[1,0]+m[1,1]\*t

    return calc\_m

def calc\_verror(v1,v2):

    return abs(v1-v2)

def calc\_perror(p\_measured,p\_prev):

    return abs(p\_measured-p\_prev)

def calc\_verror2(v\_prev,v\_current):

    v\_max\_error=abs(v\_prev-v\_current)

    return 1/(2\*(np.pi)\*(v\_max\_error\*\*2))

def calc\_perror2(p\_measured,p\_prev):

    p\_max\_error=abs(p\_measured-p\_prev)

    return 1/(2\*(np.pi)\*(p\_max\_error\*\*2))

# R is the co-varaince matrix of measurement

# For now thye state covariance matrix has been neglected

# For calculating R I have considered the following-

# Variance is calculated about the predicted path point

def create\_R(prev,curr):

    var\_px=calc\_perror2(curr[0][0],prev[0][0])

    var\_py=calc\_perror2(curr[0][1],prev[0][1])

    var\_vx=calc\_verror2(curr[1][0],prev[1][0])

    var\_vy=calc\_verror2(curr[1][1],prev[1][1])

    R=np.diag([var\_px,var\_py,var\_vx,var\_vy])

    #R=np.diag([0.8,0.8,0.1,0.1])

    return R

# Calculate Kalman Gain

def calc\_K(P,R):

    return (np.dot(P,np.linalg.inv(np.add(P,R))))

# Finally cary out the update step

def update\_X(prev,curr,P,R):  # update X

    t=np.subtract(curr,prev)

    K=calc\_K(P,R)

    t[0][0]=K[0][0]\*t[0][0]

    t[0][1]=K[1][1]\*t[0][1]

    t[1][0]=K[2][2]\*t[1][0]

    t[1][1]=K[3][3]\*t[1][1]

    t=np.add(t,curr)

    return t

def update\_P(P,R):      # update P

    K=calc\_K(P,R)

    I=np.identity(4,dtype=float)

    return np.dot(np.subtract(I,K),P)

count=0

# define P

P=np.identity(4,dtype=float)

# open file to read and write output

f=open('kalmann.txt','r')

w=open('output.txt','w')

for line in f:

    count=count+1

    l=[]

    if(count>1):

        # Because first line has only two values listed.

        prev\_state=m

    # get data

    m=np.full((2,2),fill\_value=0,dtype=float)

    for val in line.split():

     if(val[0]=='-' or val[0].isdigit()):

         l.append(float(val))

    m[0,0]=l[0]

    m[0,1]=l[1]

    if(count>1):

     m[1,0]=l[2]

     m[1,1]=l[3]

    #plotting the points and saving the output

    if(count==0):

      if inrange(img,int(m[0,0]+600),int(m[0,1]+600)):

         img[int(m[0,0]+600),int(m[0,1]+600)]=255

         cv.waitKey(1)

    if(count>1):

     curr\_predict=create\_A(prev\_state)

     R=create\_R(curr\_predict,m)

     m=update\_X(curr\_predict,m,P,R)

     P=update\_P(P,R)

     #np.savetxt(w,curr\_predict)

     np.savetxt(w,m)

     w.write("\n")

     if inrange(img,int(m[0,0]+600),int(m[0,1]+600)):

         img[int(m[0,0]+600),int(m[0,1]+600)]=255

         cv.imshow('plot',img)

         cv.waitKey(100)

     #print(curr\_predict,count,m)

# end the for loop

cv.waitKey(2000)

cv.destroyAllWindows()

#close files

w.close()

f.close()

Input-

372.99815102559614 , 3.686804471625727e-06

368.18931566716645 , 6.5966106132139 , -0.0990697000865573 , 6.361599147637872

369.50793174567275 , 12.687990358130962 , -0.29807896622275515 , 6.396538351574547

377.0563002520624 , 19.79319841043345 , -0.4985398435117239 , 6.339846883140003

365.68748510106127 , 25.654749805023815 , -0.695548210403849 , 6.348781312130009

370.0554474631758 , 32.0809839266665 , -0.90068969817821 , 6.373027003012441

370.4990715786451 , 39.15583085842088 , -1.134448100073856 , 6.640137849992708

368.1216053618028 , 45.15172616638304 , -1.2184168531207904 , 5.7592836433320596

364.07241650092124 , 51.04920725220644 , -1.4888522771789692 , 6.242348056950071

359.5012924871974 , 57.07419330241444 , -1.6405475563328016 , 6.080235126043206

351.7177370750413 , 64.21512717192309 , -1.8764722713558963 , 6.150399802235123

356.27112056343486 , 70.67440031169427 , -1.9615311806377491 , 6.341383463636228

355.54280165427815 , 76.72984007621992 , -2.2378142771541936 , 5.769883881985963

350.9476774269117 , 80.56630243748215 , -2.33259891598878 , 6.007404993276571

352.956328915756 , 88.81488331063812 , -2.382500818832718 , 5.9495813608045385

343.40166400655164 , 94.22217512643324 , -2.5495946556048454 , 5.815399704566555

344.0926383456345 , 98.06911329583517 , -2.6696662390820096 , 5.446505639637232

341.48838017954995 , 105.41733248321577 , -3.036801665851725 , 5.643404986277882

336.1901953882568 , 109.94035535806863 , -2.9006478265709386 , 5.60315189821029

334.77680425097776 , 115.95956298776869 , -3.184730617098442 , 5.64080254072384

332.345080893923 , 119.04750078347992 , -3.3645244983939593 , 5.166301914818789

332.0656841828804 , 126.74103497475056 , -3.381546486216495 , 5.330611335619468

328.9660415114842 , 129.0223133710013 , -3.802738749423398 , 4.897408032761449

325.25715735062863 , 136.92203275126678 , -3.6429512272556632 , 5.369895841004677

317.6562450683927 , 140.88738776325783 , -3.792707767704288 , 4.919735177094985

316.8770452344245 , 147.19281713550058 , -3.9104185856037175 , 4.923003632794093

307.26167997721296 , 150.3307040281144 , -4.079281877986015 , 4.489400869821912

304.598402472766 , 156.9254855394049 , -4.0061361919477765 , 4.889399722092298

300.9226208685748 , 160.73380018582478 , -4.490050671001453 , 4.141191906218092

296.96894186317417 , 164.5227805846082 , -4.258207574705038 , 4.330490221812888

291.19690616913067 , 168.75457939704066 , -4.451244073272296 , 4.483035911152267

287.95138412641063 , 179.37833249360668 , -4.312287065422556 , 4.2373475248352594

279.91806902368313 , 181.6397600170029 , -4.438155169213285 , 4.2757474763316345

285.10340807711594 , 179.5563286748867 , -4.436562030942301 , 4.178297722690968

275.16834815849865 , 186.71556703049845 , -4.753479558965729 , 3.9726402364334352

270.923402502827 , 188.0988549461731 , -4.681392212108236 , 3.8528147460704116

272.49936257756855 , 195.07345270439413 , -4.606134006605618 , 3.6989411533947973

265.8400733176275 , 200.19641309097827 , -4.756931444469781 , 3.5701448659975985

259.2150917479431 , 199.9929591277236 , -4.757435026160474 , 3.5941591188742104

250.9116891145804 , 207.09721047086688 , -4.931134574575293 , 3.529315590875577

247.86734957671317 , 206.335673708998 , -5.017640952633613 , 3.2115761944437384

242.14284431554057 , 212.71322889353942 , -4.855569368178842 , 3.250241934271156

239.83201209603425 , 218.4883115236451 , -4.9254807793508215 , 3.2631954137911494

234.08755906085176 , 215.2665630009311 , -5.159227507230827 , 3.1819445173301966

228.67907472447763 , 222.54982333561816 , -5.1085985869552015 , 3.0115575281365707

222.83555556161167 , 224.232560899945 , -5.0515214594122835 , 2.9856094935622663

218.03294110972715 , 229.59774357214022 , -4.768149884809362 , 2.795967618875425

211.85538112347487 , 231.31164968734774 , -5.014290514392959 , 2.635732088893518

205.56097452779977 , 232.97657738683915 , -4.964900966013858 , 2.6097435151715698

202.39434297985215 , 233.25259660940725 , -5.01484876104399 , 2.572063406021318

196.92230309107222 , 238.47807806098768 , -5.108425011158128 , 2.352633064175985

192.42091447096047 , 236.2395357530136 , -5.202941258902183 , 2.407682832464262

186.8362803398033 , 242.57703793479675 , -5.233524671092123 , 2.3622174182202205

183.64601018350535 , 242.59123942440675 , -5.189595925843711 , 2.129042820866552

176.46397080836803 , 245.2091043678949 , -4.939997697516189 , 2.1135936854342376

172.49988124479682 , 244.23709365562317 , -4.9470155793306825 , 2.0343034839291945

167.8100041878801 , 249.34689929290636 , -5.178962320021212 , 1.947306509904018

164.3933504242674 , 256.10379313428876 , -5.074460621328505 , 1.8906779364164072

159.86934175258136 , 248.34546316019316 , -5.030544978212277 , 1.7866312438575458

152.35637895859904 , 259.09665287378704 , -5.168725425982084 , 1.7774762229875247

147.27624398292846 , 253.30989486765753 , -5.155696474524861 , 1.5988782339767833

141.0066410162445 , 258.4267697496312 , -5.178209709358785 , 1.5917950311319182

141.0656254988835 , 258.8087974538563 , -4.824875796673689 , 1.5934209784973286

133.3123894788495 , 266.3528316523576 , -5.270112996799417 , 1.4227319746999738

126.6395324152328 , 259.54622832513235 , -4.698291788580124 , 1.4970024653138292

122.21099815885304 , 261.85537633561097 , -5.125549830238942 , 1.3844050731482416

117.7071011149903 , 265.71369997847614 , -4.818170444109786 , 1.3515317600545322

114.59905429199452 , 265.2776183142955 , -5.085450738661767 , 1.2451850673693088

107.23212201460755 , 261.01388771090217 , -4.733684552542296 , 1.1429069983976081

103.8593220894234 , 268.48693734529047 , -4.7852901939427905 , 1.133100441610077

98.13780489827172 , 271.41108270986314 , -4.9381553114820305 , 1.001921035411768

93.58328272729729 , 269.4237497018143 , -4.708973466800496 , 1.032182558134655

88.56315886964998 , 269.1685049489408 , -5.044302723127166 , 0.9221223850964603

83.93038239376776 , 270.5179987527689 , -4.88000817256307 , 0.9206292629204574

78.63024376336594 , 271.06559487211615 , -4.873245149456855 , 0.8291692634868766

72.86607530526709 , 279.04503689328794 , -4.923287982342755 , 0.7872341342495499

70.19759680262233 , 268.62583114426496 , -4.922425498027207 , 0.7078098505554188

64.43269092892483 , 273.9163540723164 , -4.851135520397761 , 0.6722891288230957

58.56483351197937 , 277.9999886453685 , -4.945773936937016 , 0.6274138200810446

54.159058053222644 , 278.09656724378016 , -4.846772920542428 , 0.5762213420233437

49.72326061619975 , 277.90113077824543 , -4.910493677930177 , 0.5338217241425367

44.53743570280951 , 273.2510567365842 , -5.019778356639428 , 0.49287263594698105

39.1128572446589 , 276.90694419692596 , -4.857129771618732 , 0.4108200633167163

34.497279475080106 , 272.86494442191474 , -4.782624768758776 , 0.3552816615785243

30.35662522343044 , 282.927124468495 , -4.872446043216092 , 0.3158706407024548

24.53380025607124 , 286.58939407485406 , -4.816210114470394 , 0.27899299834539015

19.97074918542979 , 276.4566802068186 , -4.810362791844236 , 0.22413386203649438

14.690627484440832 , 274.63832520781654 , -4.622714667179933 , 0.16787155816819282

10.008432239522188 , 279.9023309296582 , -4.9233198896691075 , 0.1276658716926528

5.062465441405342 , 281.15584380346 , -4.691519944855747 , 0.07419664050185709

-0.22121604724219368 , 280.3738136245879 , -5.4889944575616765 , 0.0022079733702266746

-4.766014513112098 , 284.0605100090169 , -4.351083965576753 , 0.002315312421696828

-9.498279435650778 , 279.42063446485633 , -5.044547294287567 , -0.06686090842997193

-14.386403832011 , 280.93986688707594 , -4.967388145517432 , -0.12484267293253103

-19.556675827058942 , 283.12831202870325 , -4.794986039823618 , -0.1687523582637663

-24.02079432916977 , 276.8615465662836 , -5.0502652533332215 , -0.22313092548730945

-29.048458889041566 , 274.8195833058524 , -5.006301817315884 , -0.26949707493735275

-33.822932394377865 , 278.11657048283257 , -4.958065810597869 , -0.31274539344017455

-38.720688646485925 , 279.58459877038007 , -4.894236608672674 , -0.38289972282874285

-43.99899682550826 , 274.66591641294804 , -4.819423979695893 , -0.429555018917355

-47.9830461868261 , 276.002729607535 , -4.9301353048121275 , -0.46526771200477046

-54.89912945876757 , 273.4474314742595 , -4.902126458765793 , -0.49646129205900574

-58.43383207815853 , 272.4226907825739 , -5.097665287247765 , -0.5675580853484118

-62.287315810112666 , 275.88306536784296 , -4.669742280410596 , -0.601872561546904

-67.73983698831131 , 274.86380719281203 , -4.746476145499201 , -0.6929391455927789

-73.92553936045523 , 276.5398995528243 , -4.7058070932636795 , -0.7013640388686374

-77.74909511517382 , 274.27005286036996 , -5.111068127197881 , -0.7785504604936607

-81.4277733679863 , 272.7807932816064 , -4.879220981712785 , -0.8585005515024755

-89.01049289120344 , 277.21582973372307 , -5.208823500860144 , -0.8352505798050384

-91.94398215955185 , 273.3258264436397 , -5.116198658821557 , -0.9326227942732215

-97.72390032790544 , 264.20703927049664 , -5.009618858384414 , -0.9825546095986575

-101.86236119977873 , 274.40386202872213 , -5.058360562147518 , -1.054518806299674

-108.09988436332809 , 265.4252131832601 , -4.965325351144753 , -1.1020562337900848

-112.38854911750498 , 265.53181181968586 , -5.04988862556382 , -1.1228520205635228

-117.432203244014 , 265.502197078812 , -4.93494674007568 , -1.2910453120923846

-121.56361610451292 , 262.64136979549 , -5.172364760399524 , -1.334845472310523

-127.21183771655438 , 258.15791146388347 , -4.957452357996271 , -1.3594256106506875

-132.97434724190612 , 262.1098538276993 , -5.188329575504482 , -1.4003604303585497

-137.0358051672655 , 257.8043660236414 , -5.031158490578554 , -1.4389313456636932

-142.63650000035997 , 256.94804032827085 , -4.791984433323407 , -1.5994375132436818

-148.45328386788103 , 255.6078774141165 , -4.938763046652047 , -1.5811015043397516

-152.4434407530426 , 254.47276028558758 , -4.89076507692769 , -1.6109836515530194

-158.90422172924266 , 253.98596814537797 , -5.175174696960881 , -1.7283680372099153

-160.9146579487741 , 251.9369375878722 , -4.970683593949931 , -1.7411609857542172

-169.18790237527938 , 250.13907398592033 , -4.763168140347535 , -1.9201835561714713

-172.5570134780878 , 244.54806361182676 , -4.9545910375088935 , -1.9723700383584704

-176.98321998903114 , 247.881107521971 , -5.087655430809131 , -1.9997162218505269

-182.7880611972589 , 239.74094292585133 , -5.119700858184481 , -2.1263621509526702

-185.68799616496548 , 244.54362993595717 , -4.9157675815573345 , -2.199340626736196

-194.21929538631488 , 241.90080036758826 , -4.9115197086825875 , -2.274731629866091

-195.83828880131338 , 238.74610090003247 , -5.173355356380073 , -2.3126621509004237

-201.94481610086228 , 230.9330050461983 , -5.013460923751694 , -2.4653975769547043

-209.0627541254782 , 234.52478064634104 , -5.010377043800726 , -2.569689608059972

-210.88868985967088 , 225.24608758351167 , -4.969387314578192 , -2.6071796516847234

-217.76674427103748 , 227.3858835794601 , -4.941153293690463 , -2.7230239456035

-220.44061129193122 , 222.7721448466413 , -4.976962367584715 , -2.8272350372554067

-232.92516488374116 , 217.2160370120031 , -5.0561971704834034 , -2.8216669454460024

-231.7968996873265 , 216.8593594079005 , -4.8464399722809635 , -3.04759151960218

-237.0845878725307 , 214.69030836180133 , -4.901729802181536 , -2.8980170567509997

-240.05104866913564 , 215.358874698012 , -4.778135301036684 , -3.2555338252819146

-249.01705367501978 , 206.9981310816798 , -4.636287495858939 , -3.365229085409339

-249.5874602093272 , 202.4545531686484 , -4.730973285446026 , -3.458572966360134

-258.38023459266753 , 198.32500112015762 , -4.802679684312743 , -3.373891746810144

-259.6328364104571 , 195.65183918441286 , -4.696028963890458 , -3.387803485901604

-267.36177493718367 , 193.63471714032156 , -4.8496471232814855 , -3.630350799844039

-273.9891487866909 , 192.83216454707585 , -4.662207622684885 , -3.6744100215627307

-271.7209247830324 , 187.60169315793652 , -4.407706767393152 , -3.843101369025829

-279.3722317682305 , 185.14515577754634 , -4.82950681616772 , -3.9911307456409384

-286.65207857952845 , 175.37010261182442 , -4.7196189038596055 , -4.042233491080853

-297.5963319965963 , 171.7988204070729 , -4.368191987029421 , -3.9815196544223967

-294.73495192860037 , 169.8610695298723 , -4.482208833985111 , -4.140734215938549

-294.60438122256227 , 166.16777028976267 , -4.3776039530286655 , -4.77548252094009

-298.2455191570674 , 162.25354042692712 , -4.435667563103276 , -4.503143363262419

-305.7466603763564 , 154.68040989551938 , -4.172075062155597 , -4.577461694212933

-314.7808218993326 , 152.66179171684388 , -3.9954841977099274 , -4.812837081222444

-318.36066431431254 , 146.7521848577679 , -3.997107113488987 , -4.819867286438338

-319.03435356112254 , 142.7791312752057 , -3.9808374516006046 , -4.880884299267241

-323.95314470567905 , 137.14246020695688 , -3.7198724814040642 , -5.1008439279747195

-322.37896148173405 , 131.18885926665206 , -3.6179584514309457 , -5.007661450129236

-327.3489217738244 , 128.0263938151378 , -3.542765045195785 , -5.203449570631336

-330.1834544384678 , 122.45116044274491 , -3.4575205857571087 , -5.160236207281302

-335.4347500164083 , 116.56248191222802 , -3.3052120947913535 , -5.075952835509601

-342.8264950325326 , 110.55411944351238 , -3.314304616716457 , -5.870989567614285

-342.4175423895798 , 105.27040567479843 , -3.2981841200325346 , -5.68704709239438

-346.72711591234247 , 98.64171664853637 , -2.9711977185357585 , -5.814012630481501

-346.84985052119896 , 95.12502533392185 , -2.6189652887402173 , -5.857525823574927

-347.70999097958145 , 86.7768566350309 , -2.6867816759037773 , -5.879244535289077

-354.8007237788312 , 81.34522483724984 , -2.475441322064059 , -6.016080303204727

-345.7309677383313 , 76.93984046833484 , -2.35079180860413 , -6.172932214994643

-358.21029077042044 , 69.27784393566071 , -2.1801537511698745 , -5.928932703582727

-356.263248204901 , 64.70860853595833 , -1.944280454059427 , -6.0698093043333365

-356.8992009158455 , 57.515611387214435 , -1.788647920893896 , -5.951510547234618

-366.8915977318734 , 50.92635925292543 , -1.7167432265594251 , -6.561921775508653

-369.30324977711456 , 44.31546359831127 , -1.526548868376773 , -6.0938440445805515

-363.4067741723748 , 39.39817618351954 , -1.2962442132493015 , -6.106641360298072

-364.9500609663309 , 33.28953895090312 , -1.1356366143104346 , -6.625882779855242

-372.04667214325093 , 26.816264324662765 , -0.9559312525626953 , -6.1919393185969644

-365.07414541158505 , 19.713982451787384 , -0.7190058107704282 , -6.382689745527245

-367.31413411455316 , 13.358324026741741 , -0.48292499901697483 , -6.662037191101488

-364.3033403334889 , 6.977346172909161 , -0.31305410221862273 , -6.129132156622972

-370.4031282847595 , 0.5895649888956205 , -0.12114799986966382 , -6.454009394658688

-365.94136670021214 , -5.814455377574818 , 0.08015976145479506 , -6.576624518055729

-368.11840787133394 , -12.697527062559416 , 0.276960120606932 , -6.792274090491824

-362.6222061369959 , -19.050292190702486 , 0.45679910477180374 , -6.773721650169993

-363.6898373930281 , -25.22588298722983 , 0.6368653551600117 , -6.566080511225335

-369.7111505021266 , -31.364281695685378 , 0.8495640781523456 , -6.488247114530613

-373.43464581029787 , -37.4110063549687 , 1.0287964778215137 , -6.5293105335346215

-364.2594366799094 , -43.562817904499234 , 1.2488248229704664 , -5.7539541969574906

-366.06459125604573 , -51.331212419508255 , 1.3724023333467192 , -6.573481204620788

-362.260474737826 , -57.286582319506664 , 1.5946713958663123 , -6.162914448140891

-362.7838924342159 , -62.495922884362734 , 1.7599149547002717 , -5.769342487252937

-362.0708779314181 , -70.38351504646637 , 1.8890511600554796 , -6.154123616715205

-354.39389739101466 , -74.84826507562168 , 2.2423221952085366 , -6.1295287906203555

-357.22749265883806 , -80.37058515403487 , 2.26120177594721 , -5.713819150477096

-356.5093894979105 , -86.01752306716273 , 2.5594624317766907 , -6.073879569176982

-350.5845824640225 , -93.18620550123079 , 2.5829451911028896 , -5.625500251754891

-349.9831772583633 , -98.81313644973875 , 2.7997231083737235 , -5.885804435555642

-342.36575345868124 , -103.12271732557717 , 2.890355410275968 , -5.315712023488834

-340.79053712878954 , -110.59519256998965 , 3.0610502881412462 , -5.5363795971556815

-341.3753771464991 , -115.71119459639038 , 3.2272809597913437 , -5.530073347330086

-331.7990397026131 , -121.41563105620867 , 3.3325798930850032 , -5.4074436890234345

-328.2146435723222 , -125.8673670472534 , 3.4209270402233636 , -5.385390985527265

-326.00626263287666 , -133.18112995153587 , 3.6226909743225466 , -5.369837477576973

-329.04130783965525 , -135.76227906635668 , 3.636184695400793 , -4.973954827635682

-318.6332439435952 , -139.670246389559 , 3.750690242201644 , -5.113408906717023

-317.8565803694601 , -145.6377083755344 , 3.9577411119580064 , -5.231476050052236

-311.7837974147916 , -150.94520918660322 , 4.12884116661851 , -4.620502783867295

-304.74703884498723 , -155.98369954476522 , 4.0550073863836 , -4.576335776104918

-301.3499026593618 , -158.31747045436862 , 4.185534106586107 , -4.691580445045387

-303.8358799019693 , -166.53625965829906 , 4.197118584882641 , -4.459843645214232

-294.0200436629427 , -168.2094684483585 , 4.180599349963571 , -4.519257125566065

-287.86205000231746 , -171.35033734859138 , 4.086687868157557 , -4.442097914689161

-286.65631366518664 , -180.14404269974972 , 4.424988606913149 , -4.144207529779565

-278.9212290277079 , -181.49798168659206 , 4.342036497438285 , -4.145995653088952

-282.2815752070535 , -187.07326011300114 , 4.908434866712435 , -4.057645262145983

-274.01357631037195 , -189.79207814969664 , 4.551758777343463 , -3.7942811050246847

-266.53425201879486 , -193.8488271274568 , 4.585405562771841 , -3.6473579146377526

-266.7979732997983 , -197.5772173887379 , 4.663005892107849 , -3.6969233932301737

-258.09057562927256 , -200.0076687455554 , 4.654008819119362 , -3.5550885593272925

-255.03848355536664 , -204.2448590834107 , 4.964270927920109 , -3.7016723384636636

-249.15240450052684 , -203.80777205740534 , 4.730509343015052 , -3.52177960965344

-241.89098583634652 , -209.29908596552914 , 4.825920960452296 , -3.2199701224703525

-237.75335317918248 , -214.6877556304161 , 4.895322226027058 , -3.132544659481972

-233.7062360471417 , -215.80872373752786 , 5.101430524332937 , -2.980591038374004

-226.4466432452746 , -213.58253166299883 , 4.9081318123826945 , -2.9871586863808557

-221.13362060178932 , -222.28021059266086 , 5.022497370742015 , -2.82173662626917

-219.99772276079023 , -220.44730384329932 , 5.29660564327428 , -2.793452675511025

-212.80233782448116 , -227.12800393494248 , 4.765395707428895 , -2.821973684166419

-207.31028859755418 , -233.2632953600602 , 4.990526634298781 , -2.6585264994882536

-201.49939651173267 , -237.24911673249042 , 5.126099862064318 , -2.358466317802712

-201.7202677156457 , -238.31935068436303 , 4.993068436868548 , -2.3925584072676616

-196.65314790398227 , -240.7538484983412 , 4.964108599842319 , -2.3621530135949427

-188.27732388095325 , -239.97186257073858 , 5.001405516865931 , -2.2351434792369442

-182.36086410442368 , -244.35016193568734 , 5.028118087849195 , -2.2358987530015058

-179.4289414050767 , -239.54184887233222 , 5.069811066652493 , -2.070304836149385

-171.1820235450768 , -245.40245965048896 , 5.21028632467528 , -2.0207411657802816

-167.28634054201362 , -249.44214731604467 , 4.94292790829465 , -2.1034976671623293

-165.65546374348386 , -252.98229028663525 , 5.003949628249042 , -1.9108097894077583

-159.63151047431074 , -249.94302735927195 , 5.0938856248858215 , -1.7822723368882256

-151.25384445751328 , -257.26019556909137 , 4.9451353620364245 , -1.755183415084611

-148.6018296615416 , -256.11585509046955 , 5.19699931944581 , -1.6825490808925645

-145.30886208238755 , -261.0924243164978 , 5.212817900245121 , -1.6108972709424645

-139.45824693174657 , -259.4374299406599 , 5.0811230724196825 , -1.5726914321209098

-137.296501133057 , -258.8531219051592 , 4.78691606154866 , -1.527251323392831

-127.50066440993834 , -262.7621614623594 , 4.971347985330398 , -1.4718557966280164

-125.15840204848249 , -262.732133878296 , 5.058067247733214 , -1.376394237988457

-119.89484078396711 , -268.73338080483273 , 4.97332556630521 , -1.289305811024373

-114.27139987024785 , -264.3879553406471 , 4.727409214716367 , -1.246931510971216

-110.26499012629749 , -266.083029523338 , 5.0144924050235575 , -1.159104205750868

-103.88900184751776 , -267.52088985939065 , 4.8991240917041345 , -1.1268259260315494

-99.03621752621903 , -270.1140123927995 , 5.011415639752068 , -1.0672036344832077

-93.58825964111917 , -269.08012213206894 , 5.006967649785361 , -1.0399514140187611

-91.11178432506976 , -269.6617594056495 , 4.746564231952626 , -0.9863181542750095

-84.20403874675279 , -276.89553388909974 , 4.822225840222633 , -0.9416912035626268

-79.11152943360848 , -272.02593105071065 , 4.8770086135134 , -0.8221097810523068

-74.09554345304903 , -278.2479822742726 , 4.903411471587326 , -0.7904888555333207

-70.04012783995319 , -271.58213213946857 , 5.031253196446898 , -0.7143617433278622

-64.70677020897898 , -274.8989647434442 , 4.8778740152846085 , -0.6371787867382502

-59.78625070249817 , -279.3580275585801 , 5.15596134434484 , -0.6324734488297254

-54.570202555027 , -281.73755631547243 , 4.957478307715819 , -0.5683622568405907

-49.36466027655856 , -278.5422491292803 , 4.771522472423449 , -0.49737902075857476

-45.00662906869062 , -279.6085628783593 , 5.106043570605254 , -0.48151052473788886

-39.76850921606369 , -277.6941150698374 , 5.2290673719893785 , -0.4330582063541467

-35.4280065313705 , -276.79496276916944 , 5.1391833623746574 , -0.35172223066898417

-30.322247275500214 , -280.7203021602167 , 5.119722068047627 , -0.33165563190589586

-25.25424799195582 , -280.18964443981037 , 4.823706222825992 , -0.28128630636720325

-20.42221153460546 , -284.4593271085567 , 4.876149489286618 , -0.2288815788368691

-15.233466773714074 , -276.0539478610912 , 4.9644701477521345 , -0.18249957577126763

-10.629436954655555 , -279.06617388320853 , 4.776900580077811 , -0.12821973216861293

-5.535405906570101 , -280.0957627095596 , 4.746778461299446 , -0.08185207263971613

-0.6696658926112462 , -275.4992678138366 , 4.5757264289467825 , -0.03325166833732439

4.174641839996032 , -280.2223986143559 , 4.65882345779549 , 0.01729628885987145

9.038569559573032 , -281.0760905246848 , 4.863551268841175 , 0.06386488879361822

13.95310393408665 , -284.50744974629197 , 4.8672200879617185 , 0.11822594701048166

18.714357872676864 , -280.8229144343459 , 4.981505385802481 , 0.16908627722584396

23.967263379920094 , -281.4161044685297 , 4.970686381570269 , 0.2171750950164791

28.858746253968434 , -275.44975425592446 , 4.8592594157804205 , 0.2563922880274252

33.62881548095634 , -276.97279481661667 , 4.803881886191522 , 0.30577485334262244

38.42281086602939 , -278.15668301408226 , 4.713995185245243 , 0.3517192692302723

43.406506271673095 , -271.5114891055532 , 4.814745070752921 , 0.4057320248872166

48.27596004626236 , -280.39040428360937 , 5.1160912289439615 , 0.47589607488422786

52.704307672247815 , -278.19891956315973 , 4.884017469908306 , 0.5146248457032423

57.26053268918517 , -283.8777687653422 , 4.720546638152938 , 0.5461205261171838

62.56457979051086 , -274.0284303632829 , 4.926599690748521 , 0.5858524726643971

66.91967770770557 , -280.28742207851667 , 5.024060028356988 , 0.6499267213299996

73.26022537405714 , -279.671532926141 , 4.771826630210032 , 0.7514003040836337

78.64469148100902 , -278.11609292402414 , 5.0353301942315065 , 0.8100458136580999

84.17395555444938 , -274.56183050426154 , 4.9850311616653356 , 0.7677046199778673

85.46156307966464 , -272.0363064538373 , 5.011535257386129 , 0.8851380049874781

94.02666294893534 , -268.70252280499653 , 4.825429658758691 , 0.9418217756229518

98.2244061788734 , -267.9571216462832 , 5.083214186376892 , 1.0137450864292252

101.02321338171208 , -266.01378934209936 , 4.887252185141858 , 1.0253871634620983

108.24780127387619 , -268.46134788150107 , 4.890843958842477 , 1.03937211123581

111.36828496070542 , -268.41563705601976 , 4.957879881196504 , 1.1586375592477303

118.96695707349458 , -263.8107003004691 , 5.148048305777498 , 1.1994433685649897

122.42461007772997 , -260.768152827037 , 4.968126254797745 , 1.2888753992610582

128.18962287686333 , -263.7967698343714 , 5.037947264872079 , 1.4019883917816

133.45643939043882 , -262.22106232115686 , 5.080407963478757 , 1.3643811503197367

137.99623338147765 , -259.10648186365506 , 5.07606089077243 , 1.4718724058934463

141.39537470151905 , -257.97499168861276 , 5.214770442115848 , 1.4733151695196252

146.5236573255499 , -257.83523328845797 , 5.076440291633034 , 1.6286961885432896

153.1018359845615 , -248.75316118241818 , 5.271166680221886 , 1.755845221734071

156.35604000342653 , -254.4601413336936 , 4.786055510482185 , 1.7790633561324272

159.41485695703332 , -249.95021079347927 , 5.057391612409649 , 1.8095120692952575

166.45620016085707 , -253.23843694898244 , 4.726345761604998 , 1.955145363020452

171.5818387573997 , -246.54358576256732 , 4.759574953515774 , 1.950371024296131

177.17341233279043 , -245.15659775501587 , 5.156522676970581 , 2.01761083696891

181.21393272786364 , -244.4604318672593 , 4.962015322006815 , 2.152117371061002

189.4073457581722 , -243.1794437518765 , 5.112464499247025 , 2.032640152605285

194.9383275492058 , -241.60164142643276 , 4.876194267159844 , 2.297990677259973

195.94145999647395 , -234.34979169308127 , 4.795303906168662 , 2.363559190461798

196.96647008916483 , -231.33393234997376 , 5.214002005564379 , 2.4101316453843094

209.09100671413182 , -230.27020242278746 , 5.13290531409814 , 2.5068470292409706

207.7137057047972 , -231.3416209900647 , 4.947385687580487 , 2.6239100568905944

216.18945099713648 , -226.05241595304275 , 4.99922184297209 , 2.623926292382488

223.05812904834602 , -218.94303846985798 , 4.963876240816417 , 2.8127640268260805

228.52787645804347 , -220.49554227430534 , 4.881057735284009 , 2.8920233506309283

233.57715038438144 , -215.51612940697404 , 4.89826360393499 , 2.9268771632854906

236.61505756538716 , -216.74422228398828 , 4.992851417655343 , 3.2010372311055155

243.7191966380538 , -209.84446660912985 , 4.76094178368198 , 3.139817772942324

246.3148081113992 , -207.67779076256738 , 4.808452117792637 , 3.2576272390307706

251.1280025309161 , -204.97555498187646 , 4.658343561333571 , 3.453357246464131

253.62834791756563 , -201.75243536735948 , 4.915819529384442 , 3.394739789672577

258.7352468048674 , -201.34646871824822 , 4.834263555466219 , 3.438433471954162

266.3652108578679 , -195.7097855349802 , 4.940366393640362 , 3.820299117656826

265.5897628348443 , -190.50001515710431 , 4.595988877812536 , 3.8798266982957004

276.40587818223986 , -186.98977534380847 , 4.696205643499028 , 3.767241691483227

273.9313049834683 , -183.0205482646663 , 4.560559890845051 , 3.958627504611413

283.0610591776159 , -180.47038808912865 , 4.62250660907368 , 4.018036493471795

288.71775821035 , -173.97850119251217 , 4.452863775902685 , 4.0874885599623845

295.11430885061975 , -168.91282635229783 , 4.595767671923398 , 4.228467919663384

298.2270361804951 , -167.5893405171054 , 4.272600815291079 , 4.16100250036129

298.79517497627126 , -159.9679118554535 , 4.129960612054945 , 4.3543335486154815

308.00120729579436 , -154.00946904391654 , 4.144796482800258 , 4.626627226596881

307.6225511057127 , -150.72570941065462 , 4.113826191400062 , 4.552980426542557

316.10282540434775 , -148.61741877488208 , 3.8923529598712907 , 4.780387929827005

322.51086718752794 , -141.31875306261074 , 3.797204863863165 , 4.8279279087162665

320.9758570836058 , -138.67968831521185 , 3.7890419716377175 , 5.132107325560893

323.24326184137846 , -132.0588457228941 , 3.6300513245990493 , 5.188984893046529

332.4319153419111 , -128.98855722108172 , 3.651831504186173 , 5.067161291117191

330.736430885717 , -123.26437204800868 , 3.3561961588917444 , 5.320625725588241

342.5568254640763 , -116.1490919824658 , 3.2941012074340668 , 5.596713489833324

339.31331755332735 , -109.99625985961676 , 3.180980910736268 , 5.498497915125203

344.29171971139937 , -104.1701087146859 , 3.0710511995305843 , 5.351324536357365

349.4033218818921 , -101.82352757648269 , 2.953186499116312 , 5.616686886704613

348.3847493022117 , -94.94186695457917 , 2.8690082438777136 , 5.594903118942787

347.2281431434594 , -90.25105737892835 , 2.662170866494579 , 6.037909616400762

352.7177007575522 , -83.2287942998338 , 2.5213591058670537 , 6.032575226874173

360.70546895924315 , -77.3486174602368 , 2.3459307579473276 , 5.925330936620123

357.4755326542439 , -70.4313227495917 , 2.183269138180665 , 5.6483119562073325

362.84931186022726 , -64.3059729327486 , 1.991462874179891 , 6.138723025964218

362.03353537500556 , -58.206902703924555 , 1.7551916841536002 , 6.102388722899329

367.4390278136758 , -51.828259466963445 , 1.6666537622812108 , 6.233575578351294

362.27953539990625 , -45.3338164878342 , 1.4844912567346509 , 6.231418808161022

367.21925743656465 , -39.5848220744104 , 1.246379571773039 , 6.370332330241954

360.24304915924546 , -33.56014644700366 , 1.0844149678536235 , 6.423988609081126

367.24947571182065 , -26.262005082479497 , 0.9029860370592354 , 6.327364361959193

368.3590653478102 , -20.579041014712022 , 0.7324730099733945 , 6.402460456626358

362.3760604593261 , -14.273293709765587 , 0.5307019457114759 , 6.544884033961958

372.54670854018156 , -7.564212315190238 , 0.3256042342012195 , 6.003952519402083

**Output-**

372.99815102559614 , 3.686804471625727e-06

3.679899848572620158e+02 6.744798277803844222e+00

-1.971765010062893064e-01 6.515001794897844256e+00

3.699294565477911760e+02 1.284804082812013171e+01

-2.995477380174936255e-01 6.279754304088255523e+00

3.771523916650156139e+02 1.992751002526199500e+01

-5.000393626905045297e-01 6.387500828585574553e+00

3.656227873901515295e+02 2.581311948196410100e+01

-6.970236490809764707e-01 6.323366938758113953e+00

3.701901105000993084e+02 3.222242807776098772e+01

-9.021774512189317807e-01 6.387190747995795448e+00

3.708868242242809288e+02 3.928138232505863670e+01

-1.135914754389205417e+00 6.642888323165754727e+00

3.678766965553390946e+02 4.529636019225146271e+01

-1.219860096949659800e+00 5.758496973870234292e+00

3.639267219576724983e+02 5.119310366791034284e+01

-1.490462198895070367e+00 6.243780413115914918e+00

3.593783725109378224e+02 5.721166174013945493e+01

-1.642064867338652423e+00 6.076104508773445545e+00

3.516583860259310654e+02 6.432869300993914408e+01

-1.878009249680280268e+00 6.158501006718154258e+00

3.563256913898822518e+02 7.079731005163345969e+01

-1.963084636351891454e+00 6.344616368614991231e+00

3.557820779100072173e+02 7.685845967411070490e+01

-2.239278856996028910e+00 5.768856995155648981e+00

3.508432399061563842e+02 8.076127205581208557e+01

-2.334252197104731863e+00 6.009853670354611488e+00

3.530164467212668455e+02 8.890365691499140155e+01

-2.384100208778795249e+00 5.941232368576684131e+00

3.433648757645250384e+02 9.435328742971645966e+01

-2.551180898345236425e+00 5.811523818089904481e+00

3.441718046198641332e+02 9.824775429873952248e+01

-2.671299302908221041e+00 5.445174378875546672e+00

3.414762619055543951e+02 1.055087375919041648e+02

-3.038467789629024640e+00 5.645826401550273488e+00

3.361901300156163188e+02 1.100834576107646825e+02

-2.902404944747956517e+00 5.594250243702498793e+00

3.347769029171399779e+02 1.160655408189281417e+02

-3.186377070203598727e+00 5.647745635246145923e+00

3.323452755567579402e+02 1.192426674916587785e+02

-3.366184801658471670e+00 5.165631500184162839e+00

3.320657317698280622e+02 1.268178539718784066e+02

-3.383412781849905127e+00 5.332544809863654756e+00

3.289665582629878600e+02 1.292559200790267937e+02

-3.804419322610759213e+00 4.896676199010637553e+00

3.252586757007660481e+02 1.369886249276164563e+02

-3.644991744607410933e+00 5.370567824050455741e+00

3.176562086130256262e+02 1.410140728257452452e+02

-3.794458168678839005e+00 4.919030927226108574e+00

3.168770930796960670e+02 1.472717339118316318e+02

-3.912336271225118178e+00 4.926788212591358018e+00

3.072616546808751536e+02 1.504822112292604857e+02

-4.081176743127538309e+00 4.489366497908869214e+00

3.045985042106252649e+02 1.569966200848343192e+02

-4.008321612857033145e+00 4.889437300039748635e+00

3.009230542379108329e+02 1.608525463988271440e+02

-4.491846172141941373e+00 4.141171816520254012e+00

2.969692096477342602e+02 1.646398364675815742e+02

-4.260674053534846628e+00 4.330569589278563392e+00

2.911968109158019047e+02 1.688564061040177364e+02

-4.453374798413405777e+00 4.483134398580872926e+00

2.879515033149239116e+02 1.794180030820115519e+02

-4.314504541847560048e+00 4.237286446372216631e+00

2.799180303103870529e+02 1.818129861133293161e+02

-4.440361922882795298e+00 4.276133879448624775e+00

2.851034230340760587e+02 1.793977650755547018e+02

-4.438813363059479045e+00 4.178146056639831407e+00

2.751683219641266192e+02 1.867641401694053513e+02

-4.755619390572263683e+00 3.972568057397847685e+00

2.709236842567129884e+02 1.883208680570694185e+02

-4.683856794849436511e+00 3.852691009322536608e+00

2.724993855650841397e+02 1.951170524376450146e+02

-4.608677421481336189e+00 3.698844837189652512e+00

2.658400031509474957e+02 2.002537213229321651e+02

-4.759425553942930875e+00 3.570029905793152825e+00

2.592150146204276098e+02 1.997815751365687049e+02

-4.760122176854315157e+00 3.594757095767978861e+00

2.509116485076418712e+02 2.071047373961569065e+02

-4.933678910198122658e+00 3.529095848640494637e+00

2.478674257248901540e+02 2.062701727212407548e+02

-5.020290883653221847e+00 3.211530911540685373e+00

2.421426400942464170e+02 2.127210378240252169e+02

-4.858708550290306682e+00 3.250609828628789888e+00

2.398320685416345839e+02 2.184970222920460685e+02

-4.928361419592705239e+00 3.264233643564972365e+00

2.340873828711457918e+02 2.152510863454251364e+02

-5.161959536404547855e+00 3.181786032680027354e+00

2.286784923912900069e+02 2.225566669616411843e+02

-5.111647855219058734e+00 3.011480950499673792e+00

2.228353594442023677e+02 2.242618449832273200e+02

-5.054750136743957789e+00 2.985115255155634362e+00

2.180335082050214055e+02 2.296069252625513286e+02

-4.771676460916498996e+00 2.795900041507964229e+00

2.118552793880675154e+02 2.313399125050243015e+02

-5.017419254012472685e+00 2.635652324286740011e+00

2.055608624791791783e+02 2.330054953564578852e+02

-4.968576034334021507e+00 2.609259620926060563e+00

2.023944223707951267e+02 2.333604965206051531e+02

-5.018418829217834443e+00 2.571729379060999410e+00

1.969219879765483370e+02 2.384832826849594483e+02

-5.111981999523596443e+00 2.352576370736715994e+00

1.924211484059218265e+02 2.362277274190487617e+02

-5.206809740877033299e+00 2.407907320162696152e+00

1.868359026993956888e+02 2.425812080744592265e+02

-5.237401133902801931e+00 2.361948259251228421e+00

1.836460799098238965e+02 2.426012327944762603e+02

-5.193667030912887839e+00 2.128990030946411149e+00

1.764638990134258734e+02 2.452091428074851365e+02

-4.945008013354772913e+00 2.112834492866019431e+00

1.725000267497183586e+02 2.442369905385325808e+02

-4.951449126187002392e+00 2.034154922502061869e+00

1.678105490499306143e+02 2.493469189086177948e+02

-5.183100458466417315e+00 1.947172383548589458e+00

1.643934310827191609e+02 2.561038079687036770e+02

-5.079537090552577183e+00 1.890472494971874751e+00

1.598695980668392167e+02 2.483454502406708855e+02

-5.035599409628371603e+00 1.786519594376453357e+00

1.523563214978520364e+02 2.590966621968390768e+02

-5.173561832420147333e+00 1.776353377707684977e+00

1.472777424575147620e+02 2.533098775464631558e+02

-5.160960312043821219e+00 1.598821037041368021e+00

1.410065148449413073e+02 2.584267893383452019e+02

-5.183917198421866601e+00 1.590596673130223015e+00

1.410656522138358469e+02 2.588090596591213739e+02

-4.832320002659491465e+00 1.594871282348246710e+00

1.333123415268858025e+02 2.663528449301135197e+02

-5.274613323463888115e+00 1.422708182720852310e+00

1.266394322469254377e+02 2.595462136094296284e+02

-4.708963407269314949e+00 1.497057550282221428e+00

1.222114965242139277e+02 2.618554197116206979e+02

-5.130258173338538619e+00 1.384368756335897510e+00

1.177073244166819279e+02 2.657137259386407777e+02

-4.826507843744743731e+00 1.351407640457523929e+00

1.145991356250935524e+02 2.652773887634599532e+02

-5.090983273833413314e+00 1.245146711651362592e+00

1.072320606156769713e+02 2.610138642305940948e+02

-4.743607141211519718e+00 1.142867164016946102e+00

1.038594240214657560e+02 2.684869507411367522e+02

-4.792681002101414123e+00 1.132700523165591111e+00

9.813765449771159410e+01 2.714111169446267695e+02

-4.944968243801562302e+00 1.001891175984187088e+00

9.358364009096415259e+01 2.694236993314411848e+02

-4.720176987811552927e+00 1.032310926397428608e+00

8.856269477605351881e+01 2.691681132836109782e+02

-5.050599297739654325e+00 9.220871073158726228e-01

8.393071531918090500e+01 2.705180727926130544e+02

-4.891204055654440275e+00 9.196867387216662859e-01

7.862990378361735111e+01 2.710657773524276308e+02

-4.882480189565132633e+00 8.291540860504315047e-01

7.286591745876803827e+01 2.790450494147308405e+02

-4.932756054616492136e+00 7.872013873123919758e-01

7.019765823690640616e+01 2.686258215550857926e+02

-4.932885357967849771e+00 7.077925634342254746e-01

6.443252377469030989e+01 2.739163729572573516e+02

-4.863851752959918784e+00 6.722505140128512968e-01

5.856469497692074810e+01 2.780000131116266289e+02

-4.956184043002568096e+00 6.273832642132359805e-01

5.415931053488258584e+01 2.780975910364346646e+02

-4.861499210893371448e+00 5.761945778335446633e-01

4.972358708609559841e+01 2.779006289012422144e+02

-4.922460934990944281e+00 5.337894330936757692e-01

4.453691001261178428e+01 2.732510355306683891e+02

-5.031414812942988313e+00 4.928392231030222503e-01

3.911250451524945504e+01 2.769069711663383941e+02

-4.878627658760365193e+00 4.108033980792847317e-01

3.449780421947482978e+01 2.728649200289258374e+02

-4.802972846280274766e+00 3.552570538654044041e-01

3.035683401267939630e+01 2.829271342672441847e+02

-4.887010421240297831e+00 3.158359825397041298e-01

2.453365265969451769e+01 2.865894209971091300e+02

-4.839097179279169936e+00 2.789559848035346734e-01

1.997124848681481879e+01 2.764566704763616372e+02

-4.830933009786066989e+00 2.241089985180147737e-01

1.469032102769318016e+01 2.746382709863989930e+02

-4.637862198193403174e+00 1.678473308271722364e-01

1.000550985952261307e+01 2.799023496591005369e+02

-4.923850266704675960e+00 1.276319920159641463e-01

5.057497274016097499e+00 2.811559224529866583e+02

-4.689272892898329381e+00 7.417117634613570876e-02

-2.213777829951080467e-01 2.803736875924669789e+02

-5.489263586853023291e+00 2.189070255383381534e-03

-4.765913601935747756e+00 2.840605367448095535e+02

-4.350876968011422541e+00 2.440092710431320585e-03

-9.498529144150063885e+00 2.794206132209459383e+02

-5.044884932135317968e+00 -6.686113573523499998e-02

-1.438579944901501761e+01 2.809399317646281133e+02

-4.965778175095213243e+00 -1.248429446122630015e-01

-1.955713746315250035e+01 2.831283570681472952e+02

-4.794201979959357374e+00 -1.687527170087022022e-01

-2.402050813901467663e+01 2.768615308384771083e+02

-5.052841750616582139e+00 -2.231312151657728060e-01

-2.904516179101717199e+01 2.748195350377889099e+02

-5.005470205572536635e+00 -2.694974146718153163e-01

-3.382257254144459324e+01 2.781166003763418075e+02

-4.957359718025873718e+00 -3.127457576640814141e-01

-3.871933712338177003e+01 2.795846659063200832e+02

-4.893642230716046093e+00 -3.828999473619868477e-01

-4.399920413765402571e+01 2.746658963763409815e+02

-4.818935769936739888e+00 -4.295553565394582551e-01

-4.798295036892424292e+01 2.760028033262410077e+02

-4.929435838304303985e+00 -4.652681530730964909e-01

-5.489916973337579265e+01 2.734473929091760738e+02

-4.901674803867362762e+00 -4.964617970204929231e-01

-5.843377354768968956e+01 2.724225946191349976e+02

-5.096926687019406543e+00 -5.675583068970346812e-01

-6.228725146336854124e+01 2.758830938431796085e+02

-4.669519229349329059e+00 -6.018730205711265713e-01

-6.773993914735204669e+01 2.748637105284197446e+02

-4.746053792497708201e+00 -6.929393185554930668e-01

-7.392559492949587252e+01 2.765399583321553223e+02

-4.705504839140408180e+00 -7.013659080760235032e-01

-7.774900442859032523e+01 2.742700094561899959e+02

-5.110320842914709338e+00 -7.785506645177473484e-01

-8.142771749654586699e+01 2.727807271266649423e+02

-4.879002093373961735e+00 -8.585007484692206425e-01

-8.901052247282250107e+01 2.772158519470796136e+02

-5.208405806989535591e+00 -8.352499025191867377e-01

-9.194394700234303741e+01 2.733258011178741071e+02

-5.115960621444254564e+00 -9.326229559915790723e-01

-9.772402076032689422e+01 2.642070284666442035e+02

-5.009406411997004760e+00 -9.825549249674373486e-01

-1.018622694067058205e+02 2.744038716903159525e+02

-5.058124199426238476e+00 -1.054519025115508990e+00

-1.080999521495541416e+02 2.654252022108477149e+02

-4.965133049221623018e+00 -1.102056565040646197e+00

-1.123884309578603364e+02 2.655327279747039597e+02

-5.049671792581859719e+00 -1.122852777755577902e+00

-1.174280931859614157e+02 2.654993014077854809e+02

-4.934768526926545995e+00 -1.291045405712146099e+00

-1.215635857679138780e+02 2.626413388617638702e+02

-5.172155439604237515e+00 -1.334845831808051608e+00

-1.272118886385833605e+02 2.581578917454678503e+02

-4.957310065895830675e+00 -1.359426251242510153e+00

-1.329743773517460568e+02 2.621098761977480649e+02

-5.188134211202168444e+00 -1.400360815011612470e+00

-1.370357836510739276e+02 2.578043454906129455e+02

-5.031016947402452644e+00 -1.438931753884437281e+00

-1.426365425502027904e+02 2.569479371004619566e+02

-4.791853075875819101e+00 -1.599437611341601562e+00

-1.484533075192994147e+02 2.556078114542524702e+02

-4.938611317987314564e+00 -1.581100645676363436e+00

-1.524434151969355469e+02 2.544726824176523508e+02

-4.890629792114118857e+00 -1.610984178416556389e+00

-1.589042371664660891e+02 2.539857866197230578e+02

-5.175008579912064199e+00 -1.728368171336091885e+00

-1.609146502892898241e+02 2.519368944664580852e+02

-4.970569591797302955e+00 -1.741162216346108016e+00

-1.691879097144821742e+02 2.501390248448333580e+02

-4.763055073272007611e+00 -1.920183644109177301e+00

-1.725569960895649047e+02 2.445480478101427195e+02

-4.954463595604261528e+00 -1.972370340020142443e+00

-1.769831741071159570e+02 2.478811340279168860e+02

-5.087535655700205162e+00 -1.999716797524174527e+00

-1.827880949819651448e+02 2.397409320728158946e+02

-5.119588904184446676e+00 -2.126362275254807255e+00

-1.856879852465880845e+02 2.445436483309190692e+02

-4.915673054097895012e+00 -2.199340842447310873e+00

-1.942193020892396476e+02 2.419007669398890528e+02

-4.911419750922434879e+00 -2.274731838674069806e+00

-1.958382814404076839e+02 2.387460728959453888e+02

-5.173245934290520154e+00 -2.312662565925367897e+00

-2.019448420677620959e+02 2.309329937390630789e+02

-5.013371847470801335e+00 -2.465397680022200166e+00

-2.090627656408116195e+02 2.345248052421235911e+02

-5.010287824409611623e+00 -2.569689759001589469e+00

-2.108886822492258091e+02 2.252460780625064558e+02

-4.969302957656795527e+00 -2.607180071579656921e+00

-2.177667569673822356e+02 2.273859248640600015e+02

-4.941069731329273296e+00 -2.723024081491544734e+00

-2.204406006029826415e+02 2.227721256995108376e+02

-4.976880418612784851e+00 -2.827235188312218206e+00

-2.329251681116525674e+02 2.172160211122577778e+02

-5.056117036650157992e+00 -2.821664119817369532e+00

-2.317968957687280636e+02 2.168591118929922743e+02

-4.846367332499522007e+00 -3.047591589243054511e+00

-2.370846427779194414e+02 2.146902676585144718e+02

-4.901656214719797511e+00 -2.898016951560829391e+00

-2.400510361482999429e+02 2.153590067038062159e+02

-4.778062429126675781e+00 -3.255533869290379023e+00

-2.490170594608788122e+02 2.069981205253821201e+02

-4.636222685431707013e+00 -3.365229228840826980e+00

-2.495874542496755453e+02 2.024545337434902592e+02

-4.730905963036213357e+00 -3.458573134916911762e+00

-2.583802405581023436e+02 1.983249797475070579e+02

-4.802614675361991381e+00 -3.373891561011522988e+00

-2.596328295849157826e+02 1.956518061678297329e+02

-4.695965705820546177e+00 -3.387804616759217513e+00

-2.673617829262700525e+02 1.936346733861530822e+02

-4.849579048063959341e+00 -3.630350864707570402e+00

-2.739891624163063852e+02 1.928320545869658247e+02

-4.662149954709605026e+00 -3.674410378634349605e+00

-2.717209212867577435e+02 1.876016762865067449e+02

-4.407650731409401779e+00 -3.843101462286682590e+00

-2.793722392383366468e+02 1.851451198558494013e+02

-4.829442467367959679e+00 -3.991130851918927025e+00

-2.866520884678795937e+02 1.753700935844946969e+02

-4.719565221197092875e+00 -4.042233798935005673e+00

-2.975963358892262818e+02 1.717987956983237154e+02

-4.368141880672723687e+00 -3.981519395304975451e+00

-2.947349485770170077e+02 1.698610239921586356e+02

-4.482151911655931187e+00 -4.140734314748965694e+00

-2.946043759696237316e+02 1.661677463978083722e+02

-4.377552097416299581e+00 -4.775482545724850425e+00

-2.982454862551729775e+02 1.622535178837750607e+02

-4.435620897340255553e+00 -4.503143305495928672e+00

-3.057466682800474587e+02 1.546803982439748779e+02

-4.172028879701716875e+00 -4.577461905897204808e+00

-3.147808268825780260e+02 1.526617480052573228e+02

-3.995437892503332922e+00 -4.812837148060544301e+00

-3.183606060233217931e+02 1.467521699266691542e+02

-3.997061358868271963e+00 -4.819869523524235255e+00

-3.190343462714981797e+02 1.427791090665675711e+02

-3.980791933230269475e+00 -4.880884557023772885e+00

-3.239531705317713204e+02 1.371424445530577430e+02

-3.719829669725804422e+00 -5.100843999474069079e+00

-3.223789569058284314e+02 1.311888444461180825e+02

-3.617916141317951251e+00 -5.007661281353821714e+00

-3.273489396908565254e+02 1.280263659144541464e+02

-3.542721737362087620e+00 -5.203449650957472805e+00

-3.301834202344414280e+02 1.224511446165507778e+02

-3.457478963825735718e+00 -5.160235843348618623e+00

-3.354347635199549700e+02 1.165624669285103181e+02

-3.305170660056008458e+00 -5.075952648915054155e+00

-3.428265009600872872e+02 1.105541047583024721e+02

-3.314261173232502689e+00 -5.870989587395441411e+00

-3.424175358835789780e+02 1.052703889755951394e+02

-3.298146729788470122e+00 -5.687047006896235324e+00

-3.467271398612248845e+02 9.864170333766753629e+01

-2.971161117660364059e+00 -5.814012754347522183e+00

-3.468498420174088324e+02 9.512500024411882293e+01

-2.618929591562345838e+00 -5.857526184997283281e+00

-3.477099772072924111e+02 8.677684606589710370e+01

-2.686743657390930906e+00 -5.879245259380523336e+00

-3.548007292789195048e+02 8.134520859304224416e+01

-2.475405181882962324e+00 -6.016080418131854124e+00

-3.457309656402757696e+02 7.693982044004680176e+01

-2.350755963760588152e+00 -6.172932315255199143e+00

-3.582102931619127162e+02 6.927783242011389575e+01

-2.180119052690149495e+00 -5.928932639131622828e+00

-3.562632423358925848e+02 6.470858922601019003e+01

-1.944244749615434475e+00 -6.069809415963121069e+00

-3.568991824015340626e+02 5.751559912090029769e+01

-1.788612564409411743e+00 -5.951510414300007490e+00

-3.668916006844076492e+02 5.092634586273957353e+01

-1.716706684426433593e+00 -6.561921801271629384e+00

-3.693032846300660594e+02 4.431545025198852272e+01

-1.526516270111677676e+00 -6.093844010983542780e+00

-3.634067709094492784e+02 3.939815824051999016e+01

-1.296209409196431572e+00 -6.106642589029274149e+00

-3.649501589552992300e+02 3.328952450730570689e+01

-1.135601494464843375e+00 -6.625882810138918444e+00

-3.720466762049049976e+02 2.681625069470668166e+01

-9.558989109456508260e-01 -6.191939282360679364e+00

-3.650741423578716649e+02 1.971397002898178741e+01

-7.189715741617663802e-01 -6.382689827962135354e+00

-3.673141500320924706e+02 1.335831014461159683e+01

-4.828917448882127927e-01 -6.662037247391585204e+00

-3.643033334036321662e+02 6.977332345889581511e+00

-3.130218856926729187e-01 -6.129132127115861195e+00

-3.704031324685823279e+02 5.895511766325776115e-01

-1.211131797695429030e-01 -6.454009443060003015e+00

-3.659413614173309384e+02 -5.814469154783263427e+00

8.019283247354813682e-02 -6.576624646298284027e+00

-3.681184185971308125e+02 -1.269753988086925034e+01

2.769926148010950495e-01 -6.792274163408613674e+00

-3.626222014982434985e+02 -1.905030607900974005e+01

4.568306728111102832e-01 -6.773720802643476979e+00

-3.636898532742660564e+02 -2.522589727395529380e+01

6.368970062462188109e-01 -6.566080435499467960e+00

-3.697111541383081885e+02 -3.136429606894049371e+01

8.495965462240502308e-01 -6.488246912513017328e+00

-3.734346511043954138e+02 -3.741102094610143070e+01

1.028829487724048253e+00 -6.529310916441763091e+00

-3.642594337079941056e+02 -4.356283224634882600e+01

1.248857432352659780e+00 -5.753954176678369059e+00

-3.660645991834146571e+02 -5.133122377683316984e+01

1.372439779427437445e+00 -6.573481223806918905e+00

-3.622604647816813213e+02 -5.728659713436584866e+01

1.594703782204755171e+00 -6.162914409843707197e+00

-3.627839038641362777e+02 -6.249593982081778165e+01

1.759949733884651257e+00 -5.769342447302043375e+00

-3.620709010558698537e+02 -7.038352623208218972e+01

1.889088472549909525e+00 -6.154123657578816520e+00

-3.543938932082440942e+02 -7.484828483646299446e+01

2.242356016081965997e+00 -6.129528151334948660e+00

-3.572274974282846642e+02 -8.037060113048227095e+01

2.261237571302564309e+00 -5.713819112654633159e+00

-3.565094051863218283e+02 -8.601753869098352823e+01

2.559499038609561072e+00 -6.073879612845026976e+00

-3.505845752703011726e+02 -9.318621780844874536e+01

2.582981285711341890e+00 -5.625500216688301691e+00

-3.499831894753878601e+02 -9.881315212903051304e+01

2.799760778700553399e+00 -5.885804495958547378e+00

-3.423657484336380321e+02 -1.031227377976377966e+02

2.890392234759456169e+00 -5.315711995908869447e+00

-3.407905555359051846e+02 -1.105952043767577493e+02

3.061090401403736561e+00 -5.536379668408214272e+00

-3.413753837864408069e+02 -1.157112118413888027e+02

3.227319551988157897e+00 -5.530070855082779957e+00

-3.317990358896448697e+02 -1.214156465222886823e+02

3.332618945481353734e+00 -5.407443560855298870e+00

-3.282145474565698464e+02 -1.258673868653463614e+02

3.420967084169615369e+00 -5.385390272849170934e+00

-3.260062825878060266e+02 -1.331811420143694420e+02

3.622730363178044311e+00 -5.369836467124124546e+00

-3.290413114743599294e+02 -1.357623132463694731e+02

3.636225575295533918e+00 -4.973954787938552613e+00

-3.186332403701268277e+02 -1.396702689648653291e+02

3.750733490698106021e+00 -5.113409019408940104e+00

-3.178565885060770597e+02 -1.456377231595251942e+02

3.957782474278816576e+00 -5.231476183157424131e+00

-3.117837859733414234e+02 -1.509452258088406609e+02

4.128881899927185906e+00 -4.620502758145497602e+00

-3.047470305231790917e+02 -1.559837170544300307e+02

4.055055787256996780e+00 -4.576335420291450795e+00

-3.013499394374691747e+02 -1.583175082563734577e+02

4.185580859783565089e+00 -4.691580581408657302e+00

-3.038358835289277522e+02 -1.665362703923434253e+02

4.197165374350510625e+00 -4.459843577399475123e+00

-2.940200393563554258e+02 -1.682095211723393220e+02

4.180648875105203111e+00 -4.519257390069942204e+00

-2.878620377650914293e+02 -1.713503654358363235e+02

4.086737593974403282e+00 -4.442097711019134643e+00

-2.866563220639990277e+02 -1.801440527315907900e+02

4.425034638985529512e+00 -4.144207477024990816e+00

-2.789212217175534079e+02 -1.814980468395878859e+02

4.342090679925447283e+00 -4.146004398431774263e+00

-2.822815783485275460e+02 -1.870732759352258370e+02

4.908481561951176353e+00 -4.057645085161666110e+00

-2.740135691078629065e+02 -1.897921105945814020e+02

4.551818232984333079e+00 -3.794281045646109796e+00

-2.665342437534087594e+02 -1.938488488718283804e+02

4.585463045783601643e+00 -3.647357808200009099e+00

-2.667979782896771326e+02 -1.975772410479313237e+02

4.663064963190222123e+00 -3.696923708732693825e+00

-2.580905696463776167e+02 -2.000077050391159048e+02

4.654068482420651520e+00 -3.555088449072148382e+00

-2.550384986595398118e+02 -2.042448799012616405e+02

4.964327851737555086e+00 -3.701672445146898305e+00

-2.491523782509989360e+02 -2.038075703503482998e+02

4.730572789484184248e+00 -3.521779522723708755e+00

-2.418909762758710258e+02 -2.092991020207469148e+02

4.825981787552239943e+00 -3.219970070656084538e+00

-2.377533883274327593e+02 -2.146877719920562413e+02

4.895389113878795939e+00 -3.132544480609479542e+00

-2.337062645695885408e+02 -2.158088023858958024e+02

5.101496423662181812e+00 -2.980590935460964541e+00

-2.264466320347722217e+02 -2.135824920633866952e+02

4.908210745456170976e+00 -2.987161066544716359e+00

-2.211335608437564986e+02 -2.222802207285780298e+02

5.022568305134692324e+00 -2.821736531770953960e+00

-2.199977289845445512e+02 -2.204472557467152569e+02

5.296676706041668403e+00 -2.793452122827938311e+00

-2.128023250843391736e+02 -2.271280171306002842e+02

4.765492952415827155e+00 -2.821974232232789426e+00

-2.073102553045963532e+02 -2.332633097288723718e+02

4.990598829578652307e+00 -2.658526403850955955e+00

-2.014993670236255525e+02 -2.372491388500117182e+02

5.126178591924913874e+00 -2.358466265707490184e+00

-2.017202722392167686e+02 -2.383194330506607628e+02

4.993167281102463662e+00 -2.392558865774689902e+00

-1.966528222906839858e+02 -2.407538847074753221e+02

4.964201663071289516e+00 -2.362152499508965775e+00

-1.882773168216581325e+02 -2.399717498688497699e+02

5.001497185848329252e+00 -2.235143356165314898e+00

-1.823608377854858134e+02 -2.443501820651236187e+02

5.028215329238624243e+00 -2.235918893985459288e+00

-1.794289528921292174e+02 -2.395418305427724306e+02

5.069907631465221165e+00 -2.070304744282623677e+00

-1.711820159655036662e+02 -2.454024746888752873e+02

5.210385804263132137e+00 -2.020740858814292551e+00

-1.672863588579357099e+02 -2.494421691331414195e+02

4.943053331372281534e+00 -2.103497851005802666e+00

-1.656554710136921358e+02 -2.529823151821981355e+02

5.004051223341288157e+00 -1.910809710449608234e+00

-1.596314868648366030e+02 -2.499429983615353024e+02

5.093995529362152830e+00 -1.782272218523594187e+00

-1.512538371243651909e+02 -2.572602076136379878e+02

4.945269964706902499e+00 -1.755182853451704661e+00

-1.486018401615862103e+02 -2.561157780806407231e+02

5.197108865654238130e+00 -1.682548871431882587e+00

-1.453088747281568658e+02 -2.610924420244227804e+02

5.212947346346743949e+00 -1.610897058611206800e+00

-1.394582091733788616e+02 -2.594373766943760984e+02

5.081271686321208847e+00 -1.572691033914748182e+00

-1.372965093800787599e+02 -2.588529711139275378e+02

4.787087989207478245e+00 -1.527250988584276881e+00

-1.275006596028211447e+02 -2.627621839989932369e+02

4.971476389253736805e+00 -1.471855521991756177e+00

-1.251584112061827199e+02 -2.627294627156059050e+02

5.058208250237067993e+00 -1.376394078620152728e+00

-1.198947236063036286e+02 -2.687333941741994181e+02

4.973495689385035057e+00 -1.289305636334109106e+00

-1.142713628412393660e+02 -2.643879368688093336e+02

4.727619785567757305e+00 -1.246931151946848892e+00

-1.102650234870919235e+02 -2.660830768753212396e+02

5.014635611762951584e+00 -1.159104032531665940e+00

-1.038889841735719415e+02 -2.675209456822852871e+02

4.899334525128283602e+00 -1.126825454718543673e+00

-9.903672854426706351e+01 -2.701140433453100513e+02

5.011592872914891750e+00 -1.067203379324505264e+00

-9.358820517702031339e+01 -2.690800445092309019e+02

5.007174253539668385e+00 -1.039950855794262496e+00

-9.111179372700156875e+01 -2.696618973372543451e+02

4.746845820323223997e+00 -9.863178706302594412e-01

-8.420402773571628074e+01 -2.768955449812725078e+02

4.822432500328947214e+00 -9.416908626785991387e-01

-7.911144135844408254e+01 -2.720259145737422841e+02

4.877228797573002872e+00 -8.221096538373525719e-01

-7.409537213368575692e+01 -2.782479951697138176e+02

4.903669985368773077e+00 -7.904883744489965558e-01

-7.004015583995303018e+01 -2.715821201025930236e+02

5.031491999410859783e+00 -7.143615434998883407e-01

-6.470669153978461452e+01 -2.748989889337777868e+02

4.878264720545604227e+00 -6.371785896451138687e-01

-5.978569501323011792e+01 -2.793580455524261197e+02

5.156200751997896070e+00 -6.324702179496215493e-01

-5.456980989295778528e+01 -2.817375900342178170e+02

4.957982687095364938e+00 -5.683620197141971975e-01

-4.936456611859409804e+01 -2.785422240197569863e+02

4.772094467008198393e+00 -4.973788065996356256e-01

-4.500668525403649056e+01 -2.796086381150680609e+02

5.106306254844399817e+00 -4.815095668064506840e-01

-3.976833299311791592e+01 -2.776940731658914387e+02

5.229428544923072408e+00 -4.330578926245115534e-01

-3.542803266762521730e+01 -2.767948735497481607e+02

5.139818816247122335e+00 -3.517220437818015522e-01

-3.032291663985364849e+01 -2.807203225956135952e+02

5.120377660411468135e+00 -3.316548744202527099e-01

-2.525468482702847695e+01 -2.801894933221362294e+02

4.828983429797763627e+00 -2.812860045931321329e-01

-2.041994418118908072e+01 -2.844593458902001544e+02

4.876710604492888734e+00 -2.288812887880136981e-01

-1.523344123159846752e+01 -2.760539383202694239e+02

4.965051125545957511e+00 -1.824992480615879875e-01

-1.062945886765469794e+01 -2.790662005058595128e+02

4.772085260878719204e+00 -1.282194521429411982e-01

-5.535381335734247621e+00 -2.800958405947317829e+02

4.747042209810545366e+00 -8.185174483274053336e-02

-6.695992678641081719e-01 -2.754992503688119996e+02

4.575424058051487108e+00 -3.325135559187774453e-02

4.174671249251018956e+00 -2.802224155918108863e+02

4.659054897786075955e+00 1.729658955365011991e-02

9.038608146217606887e+00 -2.810761844459324266e+02

4.863695376203746257e+00 6.386521517993536123e-02

1.395325905260327382e+01 -2.845074731130857799e+02

4.866776298844196269e+00 1.182262266089527669e-01

1.871428335795309295e+01 -2.808228926740206362e+02

4.979104674594239555e+00 1.690865760672335683e-01

2.396729212913615470e+01 -2.814162395955323746e+02

4.970639096186516781e+00 2.171754110792033787e-01

2.885864694438200218e+01 -2.754497408210885965e+02

4.859233871409005090e+00 2.563926755860296836e-01

3.362872727560826291e+01 -2.769728474450734552e+02

4.803854966697395312e+00 3.057751611205307141e-01

3.842206800356457563e+01 -2.781567507189217849e+02

4.713973974887514018e+00 3.517196000373021270e-01

4.340653310726147396e+01 -2.715114770441391556e+02

4.814711635843047510e+00 4.057323062771894251e-01

4.827609237696615452e+01 -2.803904133107325833e+02

5.116010886225131671e+00 4.758962914989869897e-01

5.270429714527751486e+01 -2.781988829900083147e+02

4.884005626480816709e+00 5.146252381360130190e-01

5.726051060078781063e+01 -2.838777828789617388e+02

4.720534274367981453e+00 5.461210086693986110e-01

6.256459219724386855e+01 -2.740284222257360511e+02

4.926575038662275752e+00 5.858528551817554497e-01

6.691966503975444880e+01 -2.802874348839816321e+02

5.024042863457866659e+00 6.499269585237152924e-01

7.326023087308874437e+01 -2.796714028202514442e+02

4.771817338227473115e+00 7.514004538559297952e-01

7.864470329772504442e+01 -2.781160414038621980e+02

5.035313013217595390e+00 8.100460728061186222e-01

8.417397021053679396e+01 -2.745618079593671155e+02

4.985021418509919755e+00 7.677042610459234862e-01

8.546156112175989961e+01 -2.720362747261576715e+02

5.011523948307479159e+00 8.851381344024376041e-01

9.402666498609590917e+01 -2.687024987696972858e+02

4.825421834014625766e+00 9.418220437357794905e-01

9.822439464581779589e+01 -2.679570141622082815e+02

5.083201932567592785e+00 1.013745297732391970e+00

1.010232102127754104e+02 -2.660137481145121114e+02

4.887245256187808806e+00 1.025388468739370840e+00

1.082478043710028857e+02 -2.684613806129143541e+02

4.890835755732239498e+00 1.039373197839815299e+00

1.113682808716731643e+02 -2.684139502776012591e+02

4.957871260884700426e+00 1.158637686650967336e+00

1.189669598147163043e+02 -2.638106835386238913e+02

5.148039651109961312e+00 1.199443740928012314e+00

1.224246057952767188e+02 -2.607681274671219285e+02

4.968120178294622491e+00 1.288875569161722856e+00

1.281896319607623695e+02 -2.637967953103643026e+02

5.037940389328736224e+00 1.401988526111906719e+00

1.334564710154850218e+02 -2.622210133564690864e+02

5.080401798378721523e+00 1.364380746294818980e+00

1.379962199940866014e+02 -2.591064570911779583e+02

5.076054767142291091e+00 1.471872547246710239e+00

1.413953703853805735e+02 -2.579749235026917518e+02

5.214764155154560932e+00 1.473325625596360711e+00

1.465235737112157892e+02 -2.578346831698688675e+02

5.076435091073820516e+00 1.628696285628345342e+00

1.531018407991711001e+02 -2.487531527206083695e+02

5.271160835277128420e+00 1.755845340367746976e+00

1.563560364184804428e+02 -2.544601547989264532e+02

4.786051770359486923e+00 1.779064005788443259e+00

1.594148527707488938e+02 -2.499501937543230952e+02

5.057386053435569906e+00 1.809512564679084168e+00

1.664562038054179709e+02 -2.532384603186329173e+02

4.726341846081870912e+00 1.955145466592364878e+00

1.715818568652505292e+02 -2.465435742843892228e+02

4.759570592564997149e+00 1.950367867174829684e+00

1.771734210231701354e+02 -2.451565423525694314e+02

5.156517281547039033e+00 2.017611061132763339e+00

1.812139262492020180e+02 -2.444603214972662784e+02

4.962011533176112543e+00 2.152117483125861153e+00

1.894073479956543054e+02 -2.431793837726068261e+02

5.112460312079732283e+00 2.032640026443989889e+00

1.949383448242037957e+02 -2.416015927342360214e+02

4.876190573181209409e+00 2.297990734065605167e+00

1.959414581297621396e+02 -2.343497810993504231e+02

4.795300382961398711e+00 2.363559420349147899e+00

1.969664681715630365e+02 -2.313339068768860898e+02

5.213997696118425118e+00 2.410131969037600630e+00

2.090910077603467130e+02 -2.302701302055812391e+02

5.132901949789243190e+00 2.506847185092909491e+00

2.077137045942468205e+02 -2.313416926776849607e+02

4.947382574613489403e+00 2.623910185652283289e+00

2.161894530462180626e+02 -2.260524014307521838e+02

4.999218584419659273e+00 2.623942126588401536e+00

2.230581329157197104e+02 -2.189430276654726697e+02

4.963873089134789751e+00 2.812764028176756970e+00

2.285278907496197291e+02 -2.204955917487828003e+02

4.881054840799512462e+00 2.892023353848685474e+00

2.335771933548524544e+02 -2.155161139816920013e+02

4.898260688546582919e+00 2.926877170602832035e+00

2.366150536802883551e+02 -2.167442848238306681e+02

4.992848458494062669e+00 3.201037232035764735e+00

2.437192000614570304e+02 -2.098444554775937263e+02

4.760939342282807907e+00 3.139817768776373974e+00

2.463148047734336501e+02 -2.076777553143881505e+02

4.808449407458732061e+00 3.257627241195597989e+00

2.511291551944831326e+02 -2.049755265592941669e+02

4.658341101876565027e+00 3.453357247767136151e+00

2.536283453826571019e+02 -2.017524115383894241e+02

4.915816906892597871e+00 3.394739785321703174e+00

2.587352754436262217e+02 -2.013462796099279046e+02

4.834261144518695374e+00 3.438433477791095427e+00

2.663652128153275385e+02 -1.957097719153474316e+02

4.940363878535778852e+00 3.820299118324697485e+00

2.655897618774270654e+02 -1.905000004219130574e+02

4.595986865374176666e+00 3.879826702580052800e+00

2.764058790620314880e+02 -1.869897534744746679e+02

4.696203426034888828e+00 3.767241689217942113e+00

2.739313042203129385e+02 -1.830205289242931315e+02

4.560557743423350630e+00 3.958627505943994151e+00

2.830610603752914471e+02 -1.804703579868681516e+02

4.622504458232779001e+00 4.018036497764699178e+00

2.887177635018050523e+02 -1.739784893677189928e+02

4.452861774523058536e+00 4.087488563634515515e+00

2.951143116660882697e+02 -1.689128111983813483e+02

4.595765547205021839e+00 4.228467921472422120e+00

2.982270324905322809e+02 -1.675892825171276002e+02

4.272598973946549528e+00 4.161002496581026833e+00

2.987951734990331829e+02 -1.599679017836210164e+02

4.129958664606175667e+00 4.354333549934654712e+00

3.080012083738640740e+02 -1.540094561611659856e+02

4.144794551443049535e+00 4.626627227533505859e+00

3.076225498959371407e+02 -1.507256860347787892e+02

4.113824391949472137e+00 4.552980423079580952e+00

3.161028266576217902e+02 -1.486173823662482505e+02

3.892351204463487147e+00 4.780387930948502451e+00

3.225108693628184824e+02 -1.413187425457017525e+02

3.797203144850104461e+00 4.827927914080952121e+00

3.209758560573242789e+02 -1.386796592297261270e+02

3.789040238604282074e+00 5.132107326399336422e+00

3.232432582450211385e+02 -1.320588341294487122e+02

3.630049740598079389e+00 5.188984897530493079e+00

3.324319163263915584e+02 -1.289885322209214848e+02

3.651829882206403699e+00 5.067161289023697002e+00

3.307364298623363652e+02 -1.232643586386643335e+02

3.356194596069282721e+00 5.320625726594444771e+00

3.425568261106038221e+02 -1.161490811948000470e+02

3.294099650396833390e+00 5.596713490757077913e+00

3.393133167162723112e+02 -1.099962473845327793e+02

3.180979442901209708e+00 5.498497912528496734e+00

3.442917227559398725e+02 -1.041700955401300490e+02

3.071049705155966869e+00 5.351324534624462892e+00

3.494033245636804281e+02 -1.018234948667037543e+02

2.953184966697287361e+00 5.616686887665703232e+00

3.483847479244045076e+02 -9.494185580088182519e+01

2.869006773734634308e+00 5.594903107235132289e+00

3.472281417840839026e+02 -9.025104101597256090e+01

2.662169421913381129e+00 6.037909616976457450e+00

3.527177026930174293e+02 -8.322878336954170209e+01

2.521357749424077532e+00 6.032575179064666671e+00

3.607054699603232848e+02 -7.734860440702732376e+01

2.345929407902787922e+00 5.925330934242050596e+00

3.574755316728158618e+02 -7.043131165348226830e+01

2.183267761523364214e+00 5.648311955286692942e+00

3.628493135754091554e+02 -6.430596040205263364e+01

1.991461439092939267e+00 6.138723026484260714e+00

3.620335334256510009e+02 -5.820689011925941259e+01

1.755190369473876411e+00 6.102388715880218584e+00

3.674390293128120675e+02 -5.182824743390208511e+01

1.666652408513966011e+00 6.233575580295349639e+00

3.622795345982383424e+02 -4.533380466935233244e+01

1.484489950404519254e+00 6.231418689919269838e+00

3.672192590203363238e+02 -3.958480872350267532e+01

1.246378276304938693e+00 6.370332332077774495e+00

3.602430484937251549e+02 -3.356013370704077659e+01

1.084413684833480662e+00 6.423988613833980565e+00

3.672494766358811376e+02 -2.626199456555561795e+01

9.029847682421096033e-01 6.327364359319891385e+00

3.683590918313182669e+02 -2.057902750876302989e+01

7.324717201575438041e-01 6.402460460022279420e+00

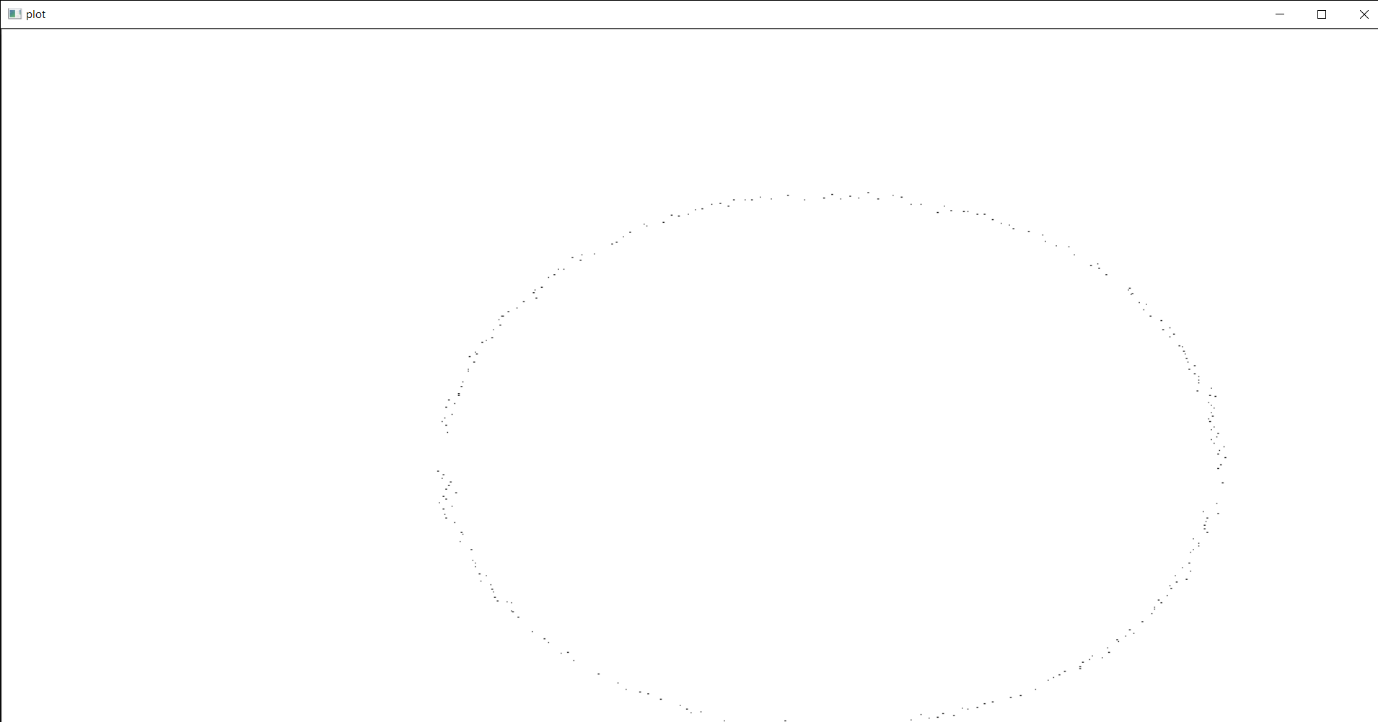
3.623760596445551982e+02 -1.427328153774325337e+01

5.307006766669840614e-01 6.544884035752535745e+00

3.725467091077775876e+02 -7.564200874944324582e+00

3.256029925590418483e-01 6.003952518930636195e+00

**Result- Plot of updated points**



**TASK 5**

**Task-5**

**Camera Calibration**

**Introduction**

Camera maps the information in the 3D world to the information in the 2D image. Thus the camera projects 3D world points to their corresponding 2D image points This mapping can be mathematically expressed as shown in equation 5.1

where, is a camera projection matrix of size 3 by 4

is a world point

is an image point

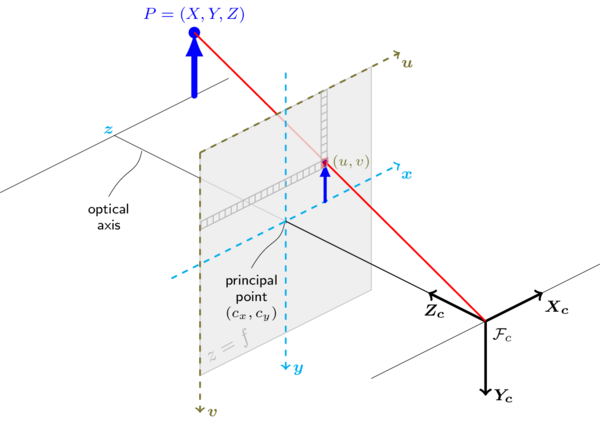
**Determining the camera projection matrix and its elements is termed as ‘camera calibration’.** The camera projection matrix can be expressed in the form of its elements as given in equation 5.2

**The plane which passes through camera projection centre and parallel to focal (image) plane is termed as the principal plane**. The ray orthogonal to the principal plane and originating from the camera projection centre passes through the **focal plane** of the camera. This point of intersection is the **principal point** of the camera. This point can be determined as follows-

The camera matrix can be decomposed as,

Where, M is a sub matrix with first three columns of the camera matrix and C4is fourth column of the camera matrix. Thus is an **augmented matrix**.

The graphical representation for the transformation expressed in the equation is- (The Pinhole model is the most widely used model which is also demonstrated here)



(Actually the image will form on the other end of a simple pin-hole camera but it is projected on the other side for ease in handling the situation.)

The mapping of object point P(X,Y,Z) to its corresponding image point Pc is described by equations 5.4 and 5.5

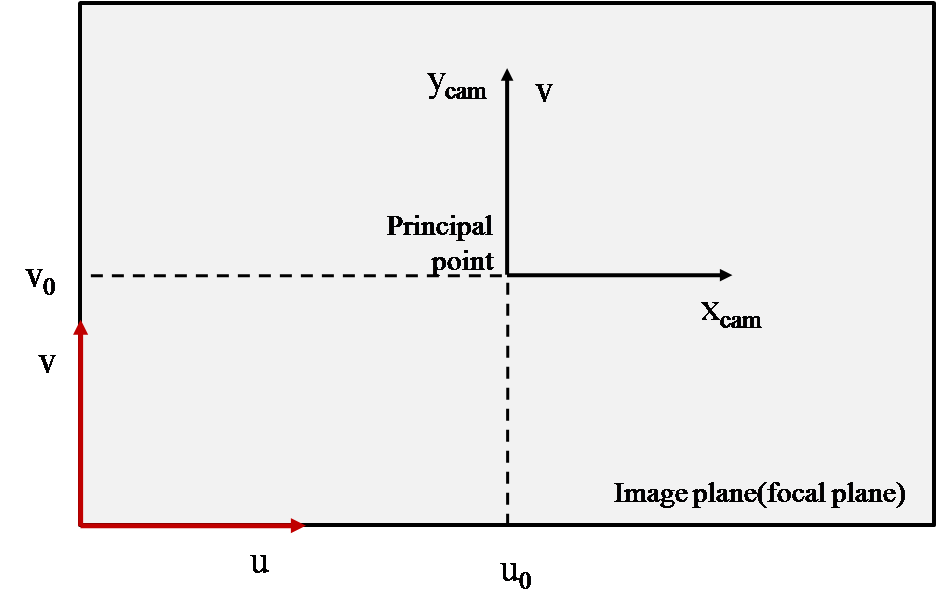
An intelligent approach is to utilise X,Y and Z to define u and v. Thus in **homogenous system** u and v are-

and

Hence,

OR

The pinhole model described above assumes that the origin of the focal (or the image) plane is at its centre which also happens to be the principal point of the camera. In practice, the origin of the focal (image) plane may not be the principal point. ( Usually it is at one corner of the image plane)



Hence,

where, (tu,tv) are the coordinates of the principal point of the camera with respect to the origin of the image plane. ( Multiplied by Z in the homogenous system).

Normally, the pixels in the image are square shaped. If the pixels of the image are not square shaped, the pixel resolution in the image in the direction of u axis will be different than pixel resolution in the direction of v axis. Let and represent camera pixel resolution along u and v axes. Hence,

Here we can write and

The more general form of the above equation is by considering skew parameter ‘s’

The skew parameter is zero in most of the normal cameras. Actually Open-CV does not consider skew. Thus the equation becomes-

where, is called as intrinsic parameter matrix of the camera.

Here, and represents the camera focal lengths (in pixels), and represent camera pixel resolution along u and v axes of the camera, whereas and represent the principal point coordinates in pixel units. All these elements of matrix K are called as intrinsic parameters of the camera. Thus the intrinsic parameters of camera need to be determined to relate the camera coordinate system and image coordinate system. The mapping equation expressed assumes that the camera centre is at the origin of the real-world coordinate system which may not be the case in practical situations.

Hence the rotation and translation matrices are used. The translation vector ‘t’ and the rotation matrix ‘R’ are the extrinsic parameters of the camera.

Thus final equation becomes-

Where R:t is called the **extrinsic matrix.**

Thus the final equation becomes-

Where the camera matrix is,

Thus to fully calibrate the camera, all the elements of camera matrix, need to be determined.

(Need of homogenous set of equation- To represent infinity)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**This task is implemented using webcam as well as mobile phone camera given as program 1 and program 2 respectively.**

**Task 5- Program 1 - using webcam**

#camera caliberation performed with laptop webcam

#program by - Kaushal Jadhav

#Roll No. 20EC30019

#import necessary libraries

import numpy as np

import cv2

from cv2 import aruco

cor\_list=[]

ids\_list=[]

count=[]

chk=True

#import aruco dictionary of standard 6X6 size and create the board

aruco\_dict=aruco.Dictionary\_get(aruco.DICT\_6X6\_250)

board=aruco.GridBoard\_create(3,4,0.018,0.004,aruco\_dict)

#get images for caliberation

for c in {'1','2','3','4','5','6','7','8','9'}:

 img=cv2.imread('aruco pic'+c+'.jpeg')

 # conversion to grayscale image is necessary.

 img=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

 img.astype(np.uint8)

 #detect aruco\_markers from the given image.

 corners,ids,rejected\_pts=aruco.detectMarkers(img, aruco\_dict)

 # refine the detection

 corners,ids,rejected\_pts,recovered\_ids=aruco.refineDetectedMarkers(img,board,corners,ids,rejected\_pts)

 # append the corners and ids into a stack

 if(chk):

     cor\_list=corners

     ids\_list=ids

     chk=False

 else:

     cor\_list=np.vstack((cor\_list, corners))

     ids\_list=np.vstack((ids\_list,ids))

 count.append(len(ids))

h=img.shape[0]

w=img.shape[1]

count=np.array(count)

#calibrate the camera

ret,cam\_mat,dist\_coeff,rvecs,tvecs=aruco.calibrateCameraAruco(cor\_list,ids\_list,count,board,(h,w),None,None)

# Some info about param

# @ cam\_mat is the intrinsic camera matrix

# @ dist\_coeff is for distorted images.

# @ rvecs is the rotation vector

# @ tvecs is the translation vector

print(cam\_mat)

print(rvecs)

print(tvecs)

w=open('a a aruco.txt','w')

w.write('corners:')

w.write(str(cor\_list))

w.write('\n')

w.write('ids\_list:')

w.write(str(ids\_list))

w.write('\n')

w.write('Intrinsic Camera Matrix')

w.write('\n')

w.write(str(cam\_mat))

w.write('\n')

w.write('Extrinsic Camera Parameters')

w.write('\n')

w.write('R\_vecs')

w.write('\n')

w.write(str(rvecs))

w.write('\n')

w.write('T\_vecs')

w.write('\n')

w.write(str(tvecs))

w.write('\n')

w.write('Shape of the image=')

w.write(str(img.shape[0])+','+str(img.shape[1]))

w.close()

print(img.shape[0],img.shape[1])

**Result-**

**For Program implementation using webcam**

corners:[[[[691. 343.]

[743. 343.]

[739. 401.]

[687. 399.]]]

[[[896. 423.]

[961. 426.]

[960. 492.]

[893. 487.]]]

[[[821. 420.]

[882. 423.]

[879. 487.]

[818. 481.]]]

[[[751. 417.]

[807. 419.]

[804. 481.]

[748. 476.]]]

[[[686. 414.]

[738. 416.]

[735. 475.]

[683. 471.]]]

[[[899. 344.]

[964. 344.]

[961. 410.]

[896. 406.]]]

[[[825. 343.]

[885. 344.]

[882. 406.]

[822. 404.]]]

[[[755. 343.]

[810. 343.]

[807. 404.]

[752. 402.]]]

[[[695. 275.]

[746. 272.]

[744. 328.]

[692. 329.]]]

[[[759. 272.]

[813. 269.]

[811. 329.]

[756. 329.]]]

[[[829. 269.]

[888. 268.]

[884. 329.]

[825. 328.]]]

[[[902. 267.]

[967. 265.]

[964. 329.]

[899. 328.]]]

[[[677. 294.]

[736. 295.]

[731. 356.]

[672. 354.]]]

[[[905. 377.]

[976. 380.]

[974. 448.]

[903. 445.]]]

[[[823. 375.]

[889. 376.]

[887. 443.]

[820. 440.]]]

[[[745. 372.]

[808. 375.]

[805. 439.]

[741. 435.]]]

[[[671. 370.]

[731. 371.]

[727. 435.]

[667. 432.]]]

[[[909. 294.]

[978. 295.]

[976. 362.]

[905. 359.]]]

[[[827. 294.]

[892. 294.]

[889. 360.]

[824. 358.]]]

[[[750. 294.]

[811. 294.]

[808. 358.]

[746. 356.]]]

[[[683. 222.]

[740. 220.]

[735. 280.]

[678. 280.]]]

[[[754. 220.]

[815. 217.]

[812. 279.]

[750. 279.]]]

[[[830. 218.]

[895. 216.]

[892. 279.]

[827. 279.]]]

[[[910. 216.]

[980. 215.]

[978. 279.]

[908. 279.]]]

[[[691. 343.]

[743. 343.]

[739. 401.]

[687. 399.]]]

[[[896. 423.]

[961. 426.]

[960. 492.]

[893. 487.]]]

[[[821. 420.]

[882. 423.]

[879. 487.]

[818. 481.]]]

[[[751. 417.]

[807. 419.]

[804. 481.]

[748. 476.]]]

[[[686. 414.]

[738. 416.]

[735. 475.]

[683. 471.]]]

[[[899. 344.]

[964. 344.]

[961. 410.]

[896. 406.]]]

[[[825. 343.]

[885. 344.]

[882. 406.]

[822. 404.]]]

[[[755. 343.]

[810. 343.]

[807. 404.]

[752. 402.]]]

[[[695. 275.]

[746. 272.]

[744. 328.]

[692. 329.]]]

[[[759. 272.]

[813. 269.]

[811. 329.]

[756. 329.]]]

[[[829. 269.]

[888. 268.]

[884. 329.]

[825. 328.]]]

[[[902. 267.]

[967. 265.]

[964. 329.]

[899. 328.]]]

[[[691. 343.]

[743. 343.]

[739. 401.]

[687. 399.]]]

[[[896. 423.]

[961. 426.]

[960. 492.]

[893. 487.]]]

[[[821. 420.]

[882. 423.]

[879. 487.]

[818. 481.]]]

[[[751. 417.]

[807. 419.]

[804. 481.]

[748. 476.]]]

[[[686. 414.]

[738. 416.]

[735. 475.]

[683. 471.]]]

[[[899. 344.]

[964. 344.]

[961. 410.]

[896. 406.]]]

[[[825. 343.]

[885. 344.]

[882. 406.]

[822. 404.]]]

[[[755. 343.]

[810. 343.]

[807. 404.]

[752. 402.]]]

[[[695. 275.]

[746. 272.]

[744. 328.]

[692. 329.]]]

[[[759. 272.]

[813. 269.]

[811. 329.]

[756. 329.]]]

[[[829. 269.]

[888. 268.]

[884. 329.]

[825. 328.]]]

[[[902. 267.]

[967. 265.]

[964. 329.]

[899. 328.]]]

[[[616. 368.]

[674. 365.]

[678. 423.]

[620. 425.]]]

[[[613. 298.]

[669. 295.]

[673. 350.]

[616. 354.]]]

[[[683. 295.]

[738. 292.]

[744. 346.]

[687. 350.]]]

[[[541. 237.]

[596. 233.]

[599. 286.]

[543. 289.]]]

[[[475. 376.]

[532. 372.]

[534. 429.]

[476. 433.]]]

[[[545. 372.]

[603. 369.]

[605. 426.]

[547. 429.]]]

[[[687. 365.]

[744. 361.]

[749. 418.]

[691. 421.]]]

[[[474. 306.]

[530. 303.]

[531. 358.]

[475. 362.]]]

[[[543. 302.]

[599. 299.]

[602. 354.]

[545. 358.]]]

[[[473. 241.]

[528. 237.]

[530. 289.]

[474. 293.]]]

[[[609. 234.]

[665. 230.]

[669. 282.]

[613. 285.]]]

[[[678. 230.]

[733. 227.]

[738. 279.]

[682. 282.]]]

[[[691. 343.]

[743. 343.]

[739. 401.]

[687. 399.]]]

[[[896. 423.]

[961. 426.]

[960. 492.]

[893. 487.]]]

[[[821. 420.]

[882. 423.]

[879. 487.]

[818. 481.]]]

[[[751. 417.]

[807. 419.]

[804. 481.]

[748. 476.]]]

[[[686. 414.]

[738. 416.]

[735. 475.]

[683. 471.]]]

[[[899. 344.]

[964. 344.]

[961. 410.]

[896. 406.]]]

[[[825. 343.]

[885. 344.]

[882. 406.]

[822. 404.]]]

[[[755. 343.]

[810. 343.]

[807. 404.]

[752. 402.]]]

[[[695. 275.]

[746. 272.]

[744. 328.]

[692. 329.]]]

[[[759. 272.]

[813. 269.]

[811. 329.]

[756. 329.]]]

[[[829. 269.]

[888. 268.]

[884. 329.]

[825. 328.]]]

[[[902. 267.]

[967. 265.]

[964. 329.]

[899. 328.]]]

[[[459. 383.]

[519. 383.]

[523. 441.]

[462. 443.]]]

[[[532. 382.]

[588. 381.]

[593. 439.]

[536. 441.]]]

[[[602. 381.]

[656. 380.]

[661. 437.]

[607. 439.]]]

[[[668. 380.]

[719. 380.]

[725. 434.]

[674. 436.]]]

[[[661. 313.]

[711. 314.]

[717. 366.]

[667. 367.]]]

[[[596. 311.]

[648. 312.]

[654. 367.]

[601. 367.]]]

[[[527. 311.]

[583. 311.]

[587. 367.]

[531. 367.]]]

[[[653. 250.]

[701. 252.]

[709. 301.]

[659. 300.]]]

[[[455. 311.]

[514. 311.]

[517. 367.]

[458. 368.]]]

[[[590. 246.]

[641. 249.]

[647. 300.]

[595. 299.]]]

[[[522. 244.]

[577. 246.]

[582. 298.]

[526. 297.]]]

[[[452. 242.]

[509. 243.]

[513. 297.]

[455. 297.]]]

[[[609. 404.]

[668. 407.]

[669. 469.]

[609. 468.]]]

[[[680. 333.]

[734. 337.]

[736. 396.]

[681. 392.]]]

[[[607. 328.]

[666. 332.]

[667. 392.]

[608. 388.]]]

[[[677. 263.]

[731. 269.]

[733. 324.]

[679. 319.]]]

[[[815. 414.]

[865. 417.]

[867. 474.]

[818. 472.]]]

[[[751. 411.]

[803. 414.]

[806. 472.]

[752. 470.]]]

[[[682. 408.]

[737. 410.]

[739. 471.]

[683. 470.]]]

[[[811. 344.]

[859. 348.]

[864. 402.]

[814. 399.]]]

[[[747. 338.]

[799. 342.]

[802. 399.]

[750. 396.]]]

[[[806. 277.]

[853. 283.]

[858. 335.]

[810. 330.]]]

[[[744. 270.]

[794. 276.]

[798. 329.]

[747. 325.]]]

[[[607. 256.]

[664. 262.]

[666. 318.]

[607. 313.]]]

[[[691. 343.]

[743. 343.]

[739. 401.]

[687. 399.]]]

[[[896. 423.]

[961. 426.]

[960. 492.]

[893. 487.]]]

[[[821. 420.]

[882. 423.]

[879. 487.]

[818. 481.]]]

[[[751. 417.]

[807. 419.]

[804. 481.]

[748. 476.]]]

[[[686. 414.]

[738. 416.]

[735. 475.]

[683. 471.]]]

[[[899. 344.]

[964. 344.]

[961. 410.]

[896. 406.]]]

[[[825. 343.]

[885. 344.]

[882. 406.]

[822. 404.]]]

[[[755. 343.]

[810. 343.]

[807. 404.]

[752. 402.]]]

[[[695. 275.]

[746. 272.]

[744. 328.]

[692. 329.]]]

[[[759. 272.]

[813. 269.]

[811. 329.]

[756. 329.]]]

[[[829. 269.]

[888. 268.]

[884. 329.]

[825. 328.]]]

[[[902. 267.]

[967. 265.]

[964. 329.]

[899. 328.]]]]

ids\_list:[[ 5]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 1]

[ 2]

[ 3]

[ 4]

[ 5]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 1]

[ 2]

[ 3]

[ 4]

[ 5]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 1]

[ 2]

[ 3]

[ 4]

[ 5]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 1]

[ 2]

[ 3]

[ 4]

[11]

[ 7]

[ 8]

[ 2]

[ 9]

[10]

[12]

[ 5]

[ 6]

[ 1]

[ 3]

[ 4]

[ 5]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 1]

[ 2]

[ 3]

[ 4]

[ 9]

[10]

[11]

[12]

[ 8]

[ 7]

[ 6]

[ 4]

[ 5]

[ 3]

[ 2]

[ 1]

[ 9]

[ 6]

[ 5]

[ 2]

[12]

[11]

[10]

[ 8]

[ 7]

[ 4]

[ 3]

[ 1]

[ 5]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 1]

[ 2]

[ 3]

[ 4]]

Intrinsic Camera Matrix

[[ 1.00300964e+03 0.00000000e+00 -4.57525034e+02]

[ 0.00000000e+00 4.42652128e+02 9.22497481e+02]

[ 0.00000000e+00 0.00000000e+00 1.00000000e+00]]

Extrinsic Camera Parameters

R\_vecs

[array([[-2.19737745],

[-0.99491132],

[ 1.60990966]]), array([[-2.16976798],

[-0.91647861],

[ 1.59443824]]), array([[-2.19737745],

[-0.99491132],

[ 1.60990966]]), array([[-2.19737745],

[-0.99491132],

[ 1.60990966]]), array([[-2.82536566],

[ 0.47579947],

[-0.87208236]]), array([[-2.19737745],

[-0.99491132],

[ 1.60990966]]), array([[ 3.03827445],

[-0.50706223],

[ 0.97975207]]), array([[ 3.14678772],

[-0.39490319],

[ 0.88266793]]), array([[-2.19737745],

[-0.99491132],

[ 1.60990966]])]

T\_vecs

[array([[ 0.1929214 ],

[-0.15460879],

[ 0.23531072]]), array([[ 0.20146714],

[-0.17480563],

[ 0.2443662 ]]), array([[ 0.1929214 ],

[-0.15460879],

[ 0.23531072]]), array([[ 0.1929214 ],

[-0.15460879],

[ 0.23531072]]), array([[ 0.20249637],

[-0.20151747],

[ 0.27564145]]), array([[ 0.1929214 ],

[-0.15460879],

[ 0.23531072]]), array([[ 0.19598563],

[-0.19311598],

[ 0.26960371]]), array([[ 0.2018271 ],

[-0.15522637],

[ 0.23719006]]), array([[ 0.1929214 ],

[-0.15460879],

[ 0.23531072]])]

Size of the image (using shape function) =720,1280 pixels

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Task 5- Program 2- using mobile phone camera**

#camera caliberation performed with mobile phone camera

#Program by - Kaushal Jadhav

#Roll No. 20EC30019

#import necessary libraries

import numpy as np

import cv2

from cv2 import aruco

cor\_list=[]

ids\_list=[]

count=[]

chk=True

#import aruco dictionary of standard 6X6 size and create the board

aruco\_dict=aruco.Dictionary\_get(aruco.DICT\_6X6\_250)

board=aruco.GridBoard\_create(3,4,0.018,0.004,aruco\_dict)

#get images for caliberation

for c in range(25):

 img=cv2.imread('mob\img'+str(c+1)+'.jpg')

 # conversion to grayscale image is necessary.

 img=cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

 img.astype(np.uint8)

 #detect aruco\_markers from the given image.

 corners,ids,rejected\_pts=aruco.detectMarkers(img,aruco\_dict)

 # refine the detection

 corners,ids,rejected\_pts,recovered\_ids=aruco.refineDetectedMarkers(img,board,corners,ids,rejected\_pts)

 # append the corners and ids into a stack

 if(chk):

     cor\_list=corners

     ids\_list=ids

     chk=False

 else:

     cor\_list=np.vstack((cor\_list, corners))

     ids\_list=np.vstack((ids\_list,ids))

 count.append(len(ids))

h=img.shape[0]

w=img.shape[1]

count=np.array(count)

#calibrate the camera

ret,cam\_mat,dist\_coeff,rvecs,tvecs=aruco.calibrateCameraAruco(cor\_list,ids\_list,count,board,(h,w),None,None)

# Some info about param

# @ cam\_mat is the intrinsic camera matrix

# @ dist\_coeff is for distorted images.

# @ rvecs is the rotation vector

# @ tvecs is the translation vector

print(cam\_mat)

print(rvecs)

print(tvecs)

w=open('a a aruco.txt','w')

w.write('corners:')

w.write(str(cor\_list))

w.write('\n')

w.write('ids\_list:')

w.write(str(ids\_list))

w.write('\n')

w.write('Intrinsic Camera Matrix')

w.write('\n')

w.write(str(cam\_mat))

w.write('\n')

w.write('Extrinsic Camera Parameters')

w.write('\n')

w.write('R\_vecs')

w.write('\n')

w.write(str(rvecs))

w.write('\n')

w.write('T\_vecs')

w.write('\n')

w.write(str(tvecs))

w.write('\n')

w.write('Shape of the image=')

w.write(str(img.shape[0])+','+str(img.shape[1]))

w.close()

print(img.shape[0],img.shape[1])

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Result –**

**For program implementation using mobile phone camera**

corners:[[[[442. 484.]

[485. 486.]

[482. 529.]

[440. 526.]]]

[[[391. 482.]

[433. 484.]

[431. 527.]

[389. 525.]]]

[[[338. 480.]

[381. 483.]

[379. 525.]

[337. 523.]]]

...

[[[341. 354.]

[385. 357.]

[382. 402.]

[339. 399.]]]

[[[287. 350.]

[331. 354.]

[329. 398.]

[285. 395.]]]

[[[233. 346.]

[277. 350.]

[275. 394.]

[231. 391.]]]]

ids\_list:[[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 3]

[ 2]

[ 1]

[12]

[ 6]

[ 5]

[ 4]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 2]

[ 1]

[ 3]

[12]

[11]

[10]

[ 9]

[ 7]

[ 8]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 3]

[ 2]

[ 1]

[ 4]

[12]

[11]

[10]

[ 9]

[ 7]

[ 6]

[ 8]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[ 6]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 5]

[ 3]

[ 6]

[ 4]

[ 2]

[ 1]

[12]

[11]

[10]

[ 9]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]

[ 8]

[12]

[11]

[10]

[ 9]

[ 8]

[ 7]

[ 6]

[ 5]

[ 4]

[ 3]

[ 2]

[ 1]]

Intrinsic Camera Matrix

[[257.34009637 0. 602.88111388]

[ 0. 243.54297745 189.3356733 ]

[ 0. 0. 1. ]]

Extrinsic Camera Parameters

R\_vecs

[array([[0.89147451],

[0.06381024],

[1.49432147]]), array([[0.90148458],

[0.09757184],

[1.15986976]]), array([[1.58858787],

[0.39499866],

[1.04526763]]), array([[ 0.79235602],

[-0.0465023 ],

[ 1.89863387]]), array([[ 2.9853125 ],

[-0.27744289],

[-0.47025134]]), array([[1.17680889],

[0.16157322],

[1.42015508]]), array([[1.52674243],

[0.36018961],

[1.29120238]]), array([[1.04894227],

[0.08757869],

[1.61986746]]), array([[1.04894227],

[0.08757869],

[1.61986746]]), array([[1.67364081],

[0.39833064],

[1.37585792]]), array([[1.77163662],

[0.4086198 ],

[1.39811629]]), array([[1.62639605],

[0.40628807],

[1.13360151]]), array([[1.61738773],

[0.39990955],

[1.2227614 ]]), array([[1.16674985],

[0.1717056 ],

[1.41222883]]), array([[1.39934564],

[0.24954635],

[1.47703297]]), array([[1.05516636],

[0.095713 ],

[1.57938887]]), array([[-2.57615219],

[-1.07646831],

[-0.45056633]]), array([[1.09057704],

[0.13445199],

[1.35421778]]), array([[1.28183151],

[0.20721364],

[1.42885188]]), array([[1.42638501],

[0.28049528],

[1.42284362]]), array([[1.01032564],

[0.1190258 ],

[1.32679621]]), array([[1.39679237],

[0.30555829],

[1.1764111 ]]), array([[1.05408869],

[0.10268914],

[1.52923848]]), array([[1.54291601],

[0.38587684],

[1.28999661]]), array([[0.81249503],

[0.02202119],

[1.7443075 ]])]

T\_vecs

[array([[-0.07172931],

[ 0.08846801],

[ 0.06245533]]), array([[-0.06095047],

[ 0.13016888],

[ 0.079389 ]]), array([[-0.00733045],

[ 0.08392621],

[ 0.05416222]]), array([[-0.03897786],

[ 0.02458459],

[ 0.02208502]]), array([[-0.08878069],

[ 0.10945081],

[ 0.08730928]]), array([[-0.05309204],

[ 0.07019583],

[ 0.04992457]]), array([[-0.02825442],

[ 0.06756412],

[ 0.04624193]]), array([[-0.05227693],

[ 0.04523824],

[ 0.03703573]]), array([[-0.05227693],

[ 0.04523824],

[ 0.03703573]]), array([[-0.02561965],

[ 0.04750632],

[ 0.03542198]]), array([[-0.02413637],

[ 0.04555146],

[ 0.03719789]]), array([[-0.01198064],

[ 0.06768113],

[ 0.04510582]]), array([[-0.01766266],

[ 0.06310427],

[ 0.04292311]]), array([[-0.06035143],

[ 0.08129877],

[ 0.0588886 ]]), array([[-0.04307903],

[ 0.05234001],

[ 0.03915726]]), array([[-0.05460975],

[ 0.05183633],

[ 0.04098759]]), array([[-0.0252532 ],

[ 0.04068522],

[ 0.03431308]]), array([[-0.05845196],

[ 0.09297847],

[ 0.06313379]]), array([[-0.04931874],

[ 0.06325316],

[ 0.04603052]]), array([[-0.04283303],

[ 0.05982784],

[ 0.04383079]]), array([[-0.06402543],

[ 0.10252613],

[ 0.06871603]]), array([[-0.03401152],

[ 0.10187975],

[ 0.06658995]]), array([[-0.05315655],

[ 0.05620141],

[ 0.04187131]]), array([[-0.02893322],

[ 0.06929272],

[ 0.04789309]]), array([[-0.07739113],

[ 0.06766236],

[ 0.0546507 ]])]

Size of the image (using shape function) =1160, 868 pixels

**TASK 4**

**Task-4**

**Object Detection**

**Introduction**

Obstacle segmentation is an important task in autonomous robotics. A robust algorithm is the one which detects the obstacles efficiently.

The main theme underlying object detection is **segmentation.** Various ways/algorithms/functions can be used to detect the objects.

Various functions/algorithms that are available in Open-CV which are handy in object detection. Some of them are-

1. Image Thresholding- (Open-CV has in-built modules for Binary Thresholding, Otsu’s Thresholding, Adaptive thresholding)
2. inRange Function- This function comes handy when color segmentation is required.
3. Contour Detection- Open-CV has in-built function, findContours which along with drawContours helps in detection of objects.
4. Histogram of Oriented Gradients (HOG),Histogram Equalization,

CLAHE (**Contrast Limited Adaptive Histogram Equalization )** are some Histogram related algorithms which deal with the frequency distribution or contrast of the image.

1. Finding Gradients in image (For e.g. Sobel filters)
2. Using masks- Open-CV has functions like bitwise\_not, bitwise\_add etc.
3. Using Hough Line Transform for detecting straight lines.

**Why is HSV used commonly for object detection** -

HSV stands for Hue-Saturation-Value.  It was designed so that it resembles the way human vision perceives colours.

Hence in varying illumination levels HSV colour space works better than normal BGR/RGB space.

So it is the most commonly used colour space.

**Task 4-Program -**

#Task-4

#Name-Kaushal Jadhav

#Roll No.- 20EC30019

import cv2

import numpy as np

import math

import statistics as stat

import matplotlib

cap=cv2.VideoCapture('sample\_output.mp4')

cv2.namedWindow('frame',cv2.WINDOW\_NORMAL)

print(cap.isOpened())

while(cap.isOpened()):

    ret,frame=cap.read()

    if(ret):

     frame1=cv2.cvtColor(frame,cv2.COLOR\_BGR2HSV)

     frame2=cv2.inRange(frame1,(110,120,70),(130,255,255))

     frame3=cv2.inRange(frame1,(0,120,70),(15,255,255))

     frame3=cv2.bitwise\_or(frame3,cv2.inRange(frame1,(165,120,70),(180,255,255)))

     frame4=cv2.bitwise\_or(frame2,frame3)

     frame1=cv2.cvtColor(frame1,cv2.COLOR\_BGR2GRAY)

     frame5=cv2.inRange(frame1,0,100)

     frame7=cv2.inRange(frame1,200,255)

     frame6=cv2.bitwise\_or(frame5,frame7)

     frame8=np.zeros(frame.shape,dtype=np.uint8)

     frame8[:,:,0]=frame8[:,:,1]=frame8[:,:,2]=frame6

     frame=cv2.bitwise\_and(frame,frame8)

     cv2.imshow('frame',frame)

     cv2.waitKey(1)

    if(not ret):

         break

cap.release()

cv2.destroyAllWindows()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Explanation and Result -**

The method used here is based on segmenting on the basis of colours.

First the RGB/BGR image is converted to HSV space. **Then this HSV space is considered as a BGR image and converted to Grayscale image.**

This type of transformation increases sharpness in the image

Fig 1- Original Image Fig 2 - Output image (HSV as

BGR-Grayscale)

Then the white and black portions of the grayscale image are segmented out.

From the original image, the red and blue coloured objects are segmented out using the HSV space.

Finally all these masks are combined into a single mask by cv2.bitwise\_or.

The combined mask then operates on the original image in the sample video.

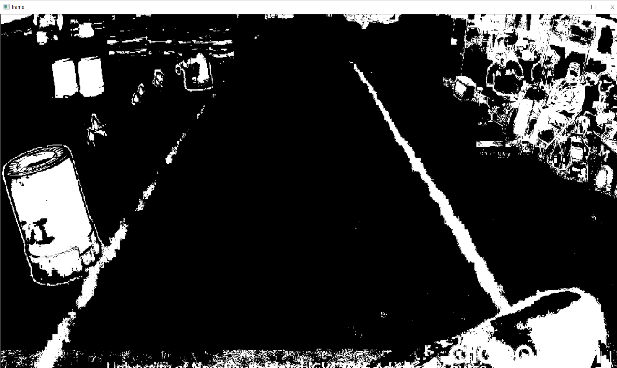
 

Fig 3- Mask Fig 4- Final Output