*Optimal Path Planning for Mobile Robots Using Memory Efficient A**

Iram Noreen
Department of Computer Science
COMSATS Institute of Information Technology
Lahore, Pakistan
iramnoreen@gmail.com

Amna Khan
Department of Computer Science
COMSATS Institute of Information Technology
Lahore, Pakistan
amna.cs@gmail.com

Zulfiqar Habib
Department of Computer Science
COMSATS Institute of Information Technology
Lahore, Pakistan
drzhabib@ciitlahore.edu.pk

***Abstract*— Optimal path for robot in collision free space is influenced by a number of factors such as path length, total number of turns, and execution time. A* is a well-known grid based path planning approach used for path planning of robots. In this paper, an efficient variation of A*, named Memory Efficient A* (MEA*), is proposed to find optimal path between start and goal point while avoiding obstacles. Performance of proposed algorithm is compared with popular grid based path planning approaches. Simulation results have shown that MEA* performs better with increased computational efficiency. It generates shorter paths in less time with less memory requirements. Moreover, proposed approach also outperforms other grid based planners in narrow passages and complex environments as well. These advantages of proposed approach make it feasible for mobile robots having limited memory and energy constraints.**

Keywords*— mobile robot; path planning; A*; grid based algorithm; memory efficient; Optimal path

I. INTRODUCTION

Path planning is a discipline that deals with planning algorithms to generate feasible path for mobile robots in an obstacle cluttered environment [1]. Its application areas include but are not limited to surveillance and security, agriculture, medicine, business industry and trajectory planning of unmanned aerial vehicle (UAV) [1, 2]. The main purpose of path planning algorithm is to efficiently navigate robot from an initial state to a final state by avoiding obstacles in its surroundings. In the category of stochastic path planners RRT [3] is most popular, whereas in grid based path planners A* [4] outperforms than others in most realistic scenarios. These planners generate geometric piecewise linear path comprising of straight line segments [1]. Though grid based algorithms like A* [4], wavefront [5], modified wavefront [6] and HPA* [7] give optimal solution but due to discretization of the state space their performance degrades in high dimensions. Consequently, memory requirements and execution time grows with the increase of problem complexity. These search algorithms suffer in real time scenarios of constrained limited memory and CPU requirements [7]. The need for memory efficient algorithm becomes evident with robots having limited memory and processing capability.

In this paper, a global grid based algorithm Memory Efficient A* (MEA*) for static environment is presented. Proposed approach aims to improve three main features; execution time, memory requirements and path length. Simulation results have shown that MEA* is time and memory efficient than prominent grid based planners such as wavefront [5], modified wavefront [6] and HPA* [7]. Moreover, proposed approach generates shorter path.

Rest of the paper is organized as follows. Related work is discussed in Section II. Methodology of the proposed approach is presented in Section III. Section IV describes datasets and experimentation setup. Simulation results and performance comparison are presented in Section V. Section VI presents conclusion and future directions.

II. RELATED WORK

Grid based path planning algorithms are generally suitable for low dimensional search space. They directly map the configuration space with the grid points by subdividing the configuration space into smaller cells. They use discrete methods to plan the path and the resulting path is a sequence of cells [5, 8]. Dijkstra [9], Extended Dijkstra [10], wavefront [5], and A* [4] are most popular grid based approaches. Dijkstra was first heuristic based approach, however it had poor search efficiency for high dimensions, hence was not practical for path planning applications [10]. Extended Dijkstra was able to solve path navigation problem within a specified time [10]. Wavefront [5] has also been used for path planning problems. It generates a cost matrix within grid and then performs backtracking from source to goal by selecting minimum cost. Thompson et al. [11] have efficiently used wavefront planner for Slocum glider with 48-hour ocean current predictions. Ansuategui et al. [12] have presented a trajectory comparison method using wavefront and other grid based planners. Recently, Tang et al. [6] have presented a performance comparison of wavefront with modified wavefront, claiming 19% improvement in results of modified wavefront than wavefront.

Originated with Dijkstra variations, A* [4] is a best first search algorithm commonly used in path planning algorithms in robotics and computer games [13, 14] . Working of A* algorithm is explained in the Fig. 1. A* search defines a heuristic cost function f-value as $f(s) = g(s) + h(s)$, where $g(s)$ is the length of the path from the current node to the start node and $h(s)$ is the length of the path from the current node to the goal node. Its cost function estimates the minimum cost from the starting node to the goal node. However, A* is highly dependent

978-1-5090-5300-1/16 $31.00 © 2016 IEEE
DOI 10.1109/FIT.2016.32

upon efficiency of its heuristic cost function and its search space expands large for high dimensional environment. Moreover, it is only suitable for static environment [15]. As the solution path given by A* search algorithm consists of adjacent tiles of the map therefore, solution path is long and jagged. Different variations of A* such as Theta* [14] and HPA* [7] address this issue by using a simplistic approach of global pruning as a post processing step.

Algorithm 1: A*(start, goal)

```
1  closeList  := φ
2  openList   := φ
3  parentList  := φ
4  openList := start // Initialize openList with the start.
5  g_cost [start] = 0
6  f_cost [start] = g_cost[start] + heuristic_cost( start, goal)
7  while openList ≠ φ
8  current := node in the openList with the minimum f_cost.
9  if current = goal
10     return pathFound
11 openList.remove(current)
12 closeList.add(current)
13 for each neighbor of current
14 if neighbor in closedList
15    continue// Ignore the neighbor which is already evaluated.
16    temp_cost := g_cost[current] + heuristic_cost (current,
       neighbor)
17    if neighbor not in openList// Discover a new node
18          openList.add(neighbor)
19    else if temp_cost >= g_cost[neighbor]
20          continue; // This is not a better path.
21          parentList[neighbor] := current // best path until now.
22          g_cost[neighbor] := temp_cost
23          f_cost[neighbor] := g_cost[neighbor] +
                                heuristic_cost(neighbor, goal)
24 end
25 end while
```

Fig. 1. Algorithm of the A*.

III. METHODOLOGY

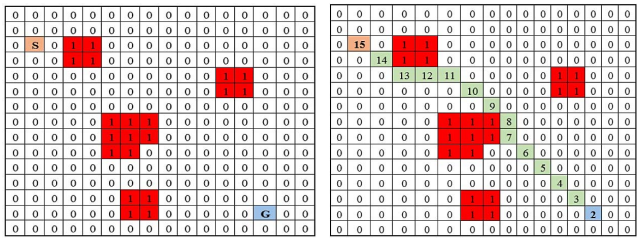
This section presents the proposed algorithm MEA*, which aims to improve time and memory efficiency of A*. Steps of proposed algorithm are described in Fig. 2. It maintains three lists, $openList$ containing the nodes to be explored, $closeList$ containing already explored nodes, and the $parentList$ containing parent node of the current node. Environment map E is provided to planner, where free space E_{Free} in environment is represented by value 0 and E_{Obs} is represented by 1, as shown in Fig. 3. MEA* starts by setting goal position E_G as current node with the cost value of 2. Then it propagates a wave towards the source position E_S, by creating cost matrix. For this purpose it uses minimum Euclidean distance criteria and assigns cost to neighbor, until source E_S is visited. At each iteration only node with the minimum f-value is added in the $openList$ and remaining neighbors are ignored. Hence, execution time and memory requirements of the proposed approach are reduced. Whereas in previous approaches such as A* [4], HPA* [7] and wavefront [5, 6], all the neighboring vertices are inserted in the $openList$ along with their cost, which causes large memory requirements with the increase of problem complexity.

Algorithm 2: MEA*(start, goal)

```
1  closeList  := φ
2  openList   := φ
3  parentList := φ
4  openList := start // Initialize openList with the start.
5  g_cost := map // Initialized with default value of infinity.
6  g_cost [start] = 0
7  f_cost := map // Initialized with default values of infinity.
8  f_cost [start] = g_cost[start] + heuristic_cost( start, goal)

9  while openList ≠ φ
10 current := node in the openList with the minimum f_cost.
11 if current == goal
12        return pathFound
13 openList.remove(current)
14 closeList.add(current)
15 for each neighbor of current
16        calculate fcost for each neighbor
17        minNeighbor := min_fcost(neighbour)
18         openList.add( minNeighbor)
19         closeList.add( allneighbors except minNeighbor)
20         parentList.add( current) // for constructing path
           when goal reached
21 end
22 end while
```

Fig. 2. Algorithm of the proposed MEA*.



(a) Initial Configuration. (b) Final Configuration.

Fig. 3. Cost matrix of MEA* algorithm.

Once source is visited, a piecewise linear path is generated by backtracking the grid cells from source to target position. This backtracking also requires traversing and processing less number of grid cells as compared to previous approaches. The proposed algorithm is not only efficient in finding a path as compared to previous grid based approaches [5, 6] but also eliminates the chances of multiple paths (see Fig. 3). However, the generated path may comprise of unnecessary waypoints. Therefore global path pruning [16] is performed to shorten the path. Pruning process is applied to the solution path. Pruning selects only ideal points, which are directly connectable using collision free straight lines as illustrated in Fig. 4. Moreover, a safe boundary distance according to robot size is also maintained during pruning.

The waypoints representing the final path obtained after pruning are much less than the original generated path as illustrated in Fig. 5. In modified wavefront and A*, every point on the path is saved as a waypoint, see Fig. 5 (a), whereas in proposed approach only pruned points are selected as final waypoints, see Fig. 5 (b). Hence final trajectory comprising of pruned points is more efficient with respect to storage and execution. The execution time efficiency is evident from simulation results in Section IV.

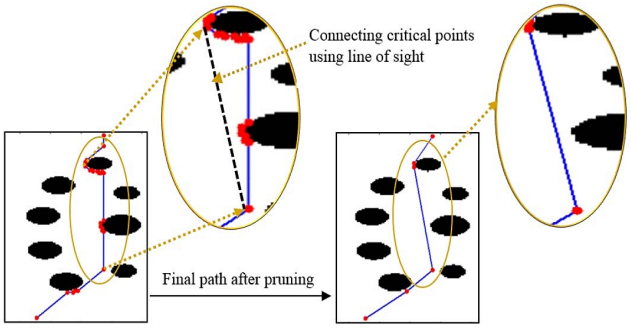
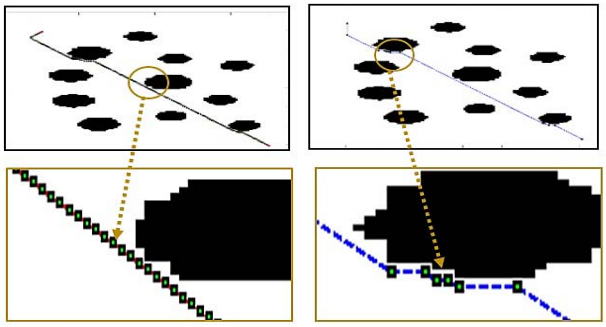


Fig. 4. Path pruning.



(a) Waypoints in Modified wavefront. (b) Waypoints in MEA*.

Fig. 5: Comparison of waypoints in path generated by the A* and MEA*.

A. Complexity Analysis

If n represents the size of the input, then complexity analysis of the proposed algorithm can be summarized with a few steps. During every iteration, the neighborhood with the minimum Euclidean distance is selected for expansion of the wave. Cost matrix expansion is dependent upon the heuristic function based on selective Euclidean, therefore it has complexity equal to $O\ (log\ h*\ (n))$. The resulting linear path is then pruned by first identifying the critical points and then selecting critical points directly connecting using line-of-sight principle. This yields our second heuristic function. The complexity of this procedure directly dependents on the size of the linear path that is further dependent on start position E_S, goal position E_G and selective Euclidean heuristic function. As a result, the complexity of the proposed algorithm could be expressed as $O\ (log\ h*\ (n))$.

IV. Data Sets and Experimentation Setup

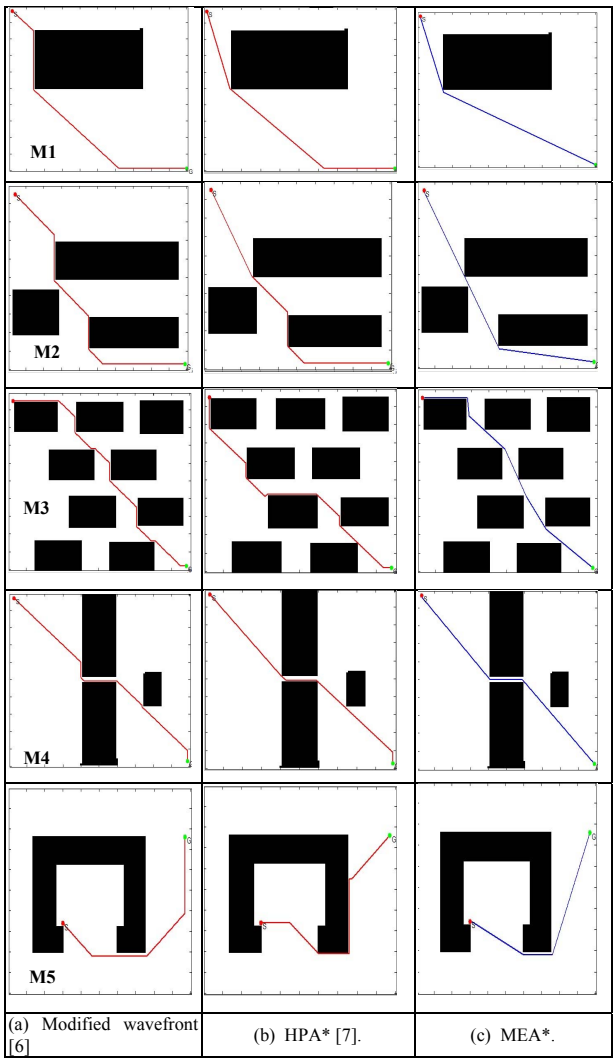
We have considered different environment map cases M1, M2, M3, M4 and M5. Environment map M1 represents a simple environment with a single obstacle. M2 is a dense environment case, M3 signifies highly dense environment. M1, M2 and M3 are adopted from datasets of previous approaches [6, 7]. However to test the robustness of proposed approach in exceptional cases two other scenarios of narrow passage and complex concave environment are also considered represented by maps M4 and M5 respectively. A simulation environment is developed using 64-bit version of MATLAB 15 to evaluate the proposed method for graphical and numerical comparative analysis with previous approaches. The proposed algorithm MEA*, modified wavefront [6] and another recent variation of A* algorithm i.e., HPA* [7] are implemented and tested in MATLAB. The operating system used for experimentation is 64-bit Windows 8.1 Pro on PC with an Intel i3-4010U@1.70 GHz CPU and 4GB internal RAM.

V. Simulation Results

In this section, simulation results for proposed algorithm are presented. Visual comparison of MEA* algorithm with previous approaches, i.e., modified wavefront [6] and HPA* [7] is shown in Fig. 6 (a) - (c). It is evident from Fig. 6 that proposed approach has generated shorter paths than both previous approaches. Main reason of getting shorter path is that proposed approach inserts neighbor with minimum Euclidean cost in $openList$, whereas previous approaches inserts all neighbors in $openList$. A short $openList$ comprising of only valuable neighbors thus, reducing execution time and total number of cells to be processed. Secondly, proposed approach further shortens the path using pruning technique. Proposed approach is flexible enough to generate safe and shorter path in much less time with less number of turns for narrow passages case M4 and for complex concave shape case M5, as shown in Fig. 6. This is quite possible because proposed approach has to traverse and process less number of grid cells to backtrack the path as explained in Section III. Further pruning process also eliminates extra waypoints in the path, which reduces the path length.

Fig. 7 (a) and Fig. 7 (b) presents execution time plot and path length plot respectively. Fig. 7 (c) shows total number of grid cells processed and Fig. 7 (d) shows total number of turns in final path for all three approaches. It is evident from plots in Fig. 7 that MEA* outperforms other approaches because of its short $openList$ and pruning criteria. Summary presented in Table I concludes that execution time, number of turns, path length and number of processed cells in proposed approach are less than previous approaches with remarkable percentage difference.

Proposed algorithm generates piecewise linear path like other grid based planners. Such a path is not feasible with dynamic and kinematic constraints of mobile robots. Such a path makes robot to make complicated motions to follow sharp turns resulting in high energy consumption, controller exertion and premature aging of robotic parts. Use of tangent continuous curves [17] for slow moving mobile robots and curvature continuous spiral transitions [18, 19] for high speed objects like auto-drive cars and aerial vehicles would be of great interest.



(a) Modified wavefront [6] (b) HPA* [7]. (c) MEA*.

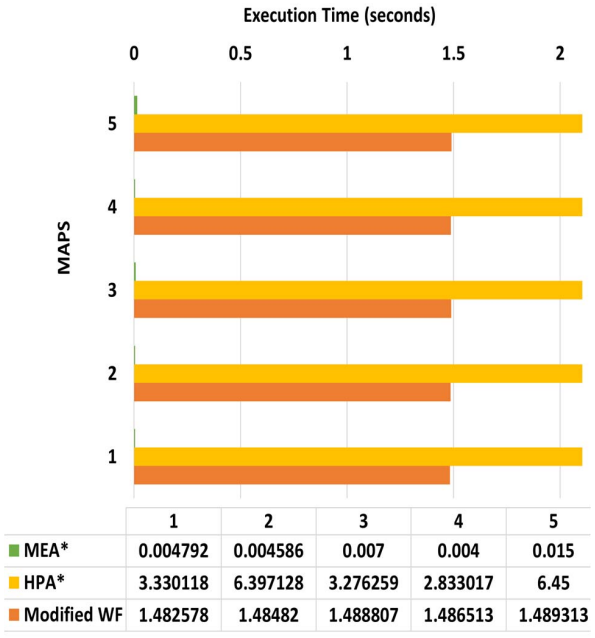
Fig. 6. Comparison of paths in different environments of proposed approach with popular grid based approaches.

TABLE I. SUMMARY OF COMPARATIVE ANALYSIS.

Algorithms	Execution time	Total number of turns	Path length	Processed grid cells
MEA* vs. Modified Wavefront [6]	97.45 % less	63.33 % less	5.89 % less	98.35 % less
MEA* vs. HPA* [7]	96.81 % less	54.16 % less	4.30 % less	97.84 % less

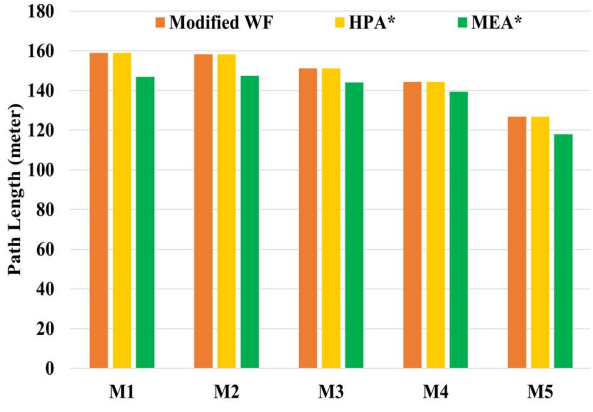
VI. CONCLUSION

In this paper, a grid based algorithm MEA* is presented to find collision free and optimal in cluttered environment. Performance of MEA* is compared with prominent grid based algorithms for static environment. MEA* is proved to generate shorter path, and effective for narrow passages as well. Further, computational efficiency and less memory requirements make it suitable to program for small robots with constraints of limited energy and memory. Desired future directions are experiments in high dimensions and to smooth sharp turns by replacing straight lines with smooth curves to improve the path feasibility for car-like robots.

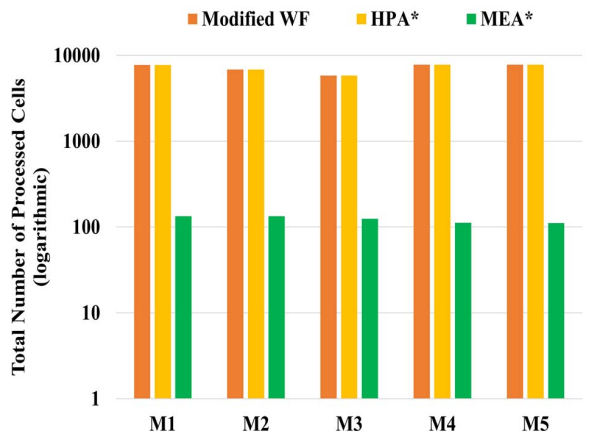


	1	2	3	4	5
MEA*	0.004792	0.004586	0.007	0.004	0.015
HPA*	3.330118	6.397128	3.276259	2.833017	6.45
Modified WF	1.482578	1.48482	1.488807	1.486513	1.489313

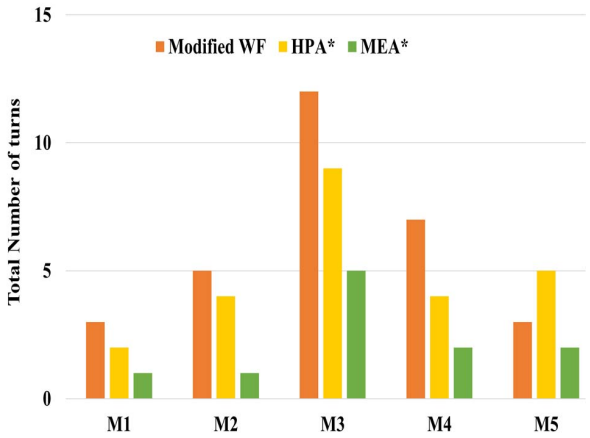
(a) Execution time comparison.



(b) Comparison for path length.



(c) Comparison for total number of processed cells.



(d) Comparison for total number of turns in path.

Fig. 7: Comparison of proposed approach MEA* with modified wave front [6] and HPA* [7] .

REFERENCES

[1] S. M. Lavalle, *Planning Algorithms*: Cambridge University Press, 2006.

[2] M. Yao, and M. Zhaoa, "Unmanned aerial vehicle dynamic path planning in an uncertain environment", *Robotica,* vol. 33 pp. 611-621, 2015.

[3] S. M. Lavalle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning", 1998.

[4] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *IEEE TRANSACTIONS OF SYSTEMS SCIENCE AND CYBERNETICS,* vol. 4, pp. 100-107, 1968.

[5] A. Zelinsky, R. A. Jarvis, J. C. Byrne, and S. Yuta, "Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot", in *Proceedings of International Conference on Advanced Robotics*, 1993.

[6] S. H. Tang, C. F. Yeong, and E. L. M. Su, "Comparison between Normal Waveform and Modified Wavefront Path Planning Algorithm for Mobile Robot", *Applied Mechanics and Materials,* vol. 607, pp. 778-781, 2014.

[7] A. Botea, M. Muller, and J. Schaeffer, "Near Optimal Hierarchical Path-Finding", *Journal of Game Development,* vol. 1, pp. 1-30, 2004.

[8] C. Goerzen, Z. Kong, and B. Mettler, "A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance", *Journal of Intelligent and Robotic Systems,* vol. 57, pp. 65-100, 2009.

[9] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs", *NUMERISCHE MATHEMATIK,* vol. 1, pp. 269-271, 1959.

[10] M. Noto, K. Univ., Yokohama., and H. Sato, "A Method for the Shortest Path Search by Extended Dijkstra Algorithm", presented at the IEEE International Conference on Systems, Man, and Cybernetics Nashville, TN, 2000.

[11] D. R. Thompson *et al.*, "Spatiotemporal path planning in strong, dynamic, uncertain currents", presented at the International Conference on Robotics and Automation Anchorage Convention District, Alaska, USA, 2010.

[12] A. Ansuategui *et al.*, "Robot trajectories comparison: a statistical approach", *The Scientific World Journal,* vol. 2014, pp. 1-13, 2014.

[13] X. Cui, and H. Shi, "A*-based Pathfinding in Modern Computer Games", *International Journal of Computer Science and Network Security,* vol. 11, pp. 125-130, 2011.

[14] K. Daniel, A. Nash, and S. Koenig, "Theta Any-Angle Path Planning on Grids", *Journal of Artificial Intelligence Research* vol. 39, pp. 533-579, 2010.

[15] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong Planning A*", *Artificial Intelligence* vol. 155, pp. 93-146s, 2004.

[16] K. Yang, "Anytime Synchronized-Biased-Greedy Rapidly-exploring Random Tree Path Planning in Two Dimensional Complex Environments", *International Journal of Control, Automation and Systems,* vol. 9, pp. 750-758, 2011.

[17] Z. Habib, M. Sarfraz, and M. Sakai, "Rational cubic spline interpolation with shape control", *Computers & Graphics,* vol. 29, pp. 594-605, 2005.

[18] Sarpono, Z. Habib, and M. Sakai, "Fair cubic transition between two circles with one circle inside or tangent to the other", *Numerical Algorithms,* vol. 51, pp. 461-476, 2009.

[19] Z. Habib, and M. Sakai, "Admissible regions for rational cubic spirals matching G^2 Hermite data", *Computer Aided Design,* vol. 42, pp. 1117-1124, 2010.