	Description	Category	Applicability	Advantages	Disadvantages	Comments
Potential Field[1]	This algorithm was proposed as a real-time collision avoidance algorithm. Navigation is achieved by applying a force towards the goal and applying other forces away from any obstacles. It is typically used as a heuristic algorithm, often after a best-first algorithm.	Probablistic	- Deals with a robot A - L is a local planner that constructs paths for A. - L must be symmetrical; that is L(a,b) == L(b,a)	- Can be used in real-time rather than waiting to determine the best path to the goal	- Will not determine on its own that a goal is unreachable.	- Could be used in a hybrid path planning sequence as a heuristic algorithm when real-time detection is needed to avoid moving obstacles.
Voronoi Diagram[2]	Introduced by computational geometry researchers. It will generate paths that maximize clearance from objects by seperating cells into geometric figures.	Roadmap	- Applicable to 2-D configuration spaces (C-Space)	-Maximizes clearance from objects - determines a fast possible path	- Cannot be used as a realtime avoidance system. - Takes time to determine the best path while the robot is not moving towards the goal	Not the Best Possible or simplest algorithm to implment for this project.
Cell Decomposition[3]	Decomposes the free space into cells in an attempt to find a path between two points. Further breaks down mixed cells (cells containing some free space and an obstacle) until a free path is found to the goal.	Roadmap	- 2-D C-Spaces - Known maps	- Can be used as both realtime or synchronously.	- Will not avoid unknown ostacles.	- Could be used in a hybrid path planning sequence as the primary algorithm for determining the path with the Potential Field Algorithm as the algorithm for use when unknown obstacles are found.
A*[4]	the A* algorithm plans a path from start to finish by storing estimates of g(s) of the path cost from the initial state to each new state.	Deterministic	-Planning an initial path through a known graph or planning space.	- Optimal Paths are found between a start and end point.	- Time delay for finding paths between points	Possible Algorithm to use if obstacles are static.
D*/ D* Lite[5]	D* Lite constructs an initial path in exactly the same way backwards A* would find an initial path. When a change is made in the planning graph, only the portions of the path being affected by the change will be updated. It uses a Heuristic to limit the states processed to only those states whose change in path could have a bearing on the path cost of the initial state.	Deterministic	- Applcable where changes in the path state occur. - Used for changing the path rather than replanning from scratch where A* would need to replan the entire path.	- Is more effiecient than planning from scrath using A* when an object moves into the path of the robot.	- State changes that are not neccissarily close to the robot can cause D* Lite to be less efficient than A* due to the possibility of D* Lite to process every state change twice.	Possible Algorithm to use if obstacles are moving or dynamic.

1) Regli, W., *Robot Lab: Robot Path Planning*, Department of Computer Science, Drexel University. Slide 58-61.
2) Ibid, Slide 38-39.
3) Ibid, Slide 42-50
4) Fergnson, D.; Likhachev, M.; and Stentz, A.; *A Guide to Heuristic-based Path Planning*. Department of Computer Science, Carnegie Mellon University. Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling (ICAPS), June, 2005. pg. 1-2
5) Ibid, pg. 2-4

Delayed D*[6]	Delayed D* is an algorithm that ignores underconsistent states when changes to edge costs occur. After values for the overconsistant stateshave been propagated, Delayed D* looks for underconsistant states, where if found are added to an "OPEN" List. The new path is then check again for underconsistant states. This repeats until a consistant state is found.	Heuristic	- Large distance between the start state and the goal state - changes are being observed in arbitrary locations in the graph (I.E. a satellite uplink to the robot.)	- Ignores the edge cost of increases that do not involve its current solution path, leading to decrease in overall computation	- Processing of underconsistant state changes soltuions of the path several times prducing a new path containing underconsitant states, resulting in more processing that in D* Lite	If map is monitored by a sensory network rather than simply by the robots, This algorithm may prove to be the best algorithm. Also for Dynamic or moving obstacles.
ARA*[7]	Anytime algorithms construct an initial, possibly suboptimal, solution quickly and then improves the quality of the path as time permits. ARA* begins by performing an A* search with an inflation factor, but during the search it will expand each state once. Once a state has been expanded by the search, if it become inconsistant, it is placed into an INCONS list, which contains all inconsistant states already expanded.	Heuristic	- High dimensioned planning (I.E. in a robot that tracks the (x, y) position, Orientation, and velocity). - Applicable in static planning domains.	- Provides substantial speed increases over the A* algorithm.	- Paths are not as optimal as in the A* algorithm.	Improves efficiency; by only expanding each state at most onc,e a solution is reached much more quickly; by only reconsidering states from the previous search that were inconsistent, much of the previous search effort can be reused. Possible Algoriothm to use if we want a speed boost over the normal A* algorithm
AD* (Anytime Dynamic A*)[8]	Combines the anytime performance of ARA* with the replanning capabilities of D* Lite. AD* performs a series of searches with decreasing inflation factors to generate series of solutions with improved bounds. When changes in the environment occur affecting the cost of edges in the graph, locally affected states are placed on the OPEN queue to propogate the changes through the rest of the graph.	Heurisitc	- Planning in complex, dynamic state spaces (3 DOF robotic arm operating in dynamic environments. -Path-planning for outdoor mobile robots those operatingin in dynamic or partially known outdoor environments)	- Perfomance increases over the A* search with the dynamic changes that can be found in D* algorithms.	- Paths may not be as optimal as in the A* or D* algorithms.	Improves efficiency; by only expanding each state at most onc,e a solution is reached much more quickly; by only reconsidering states from the previous search that were inconsistent, much of the previous search effort can be reused. Possible Algoriothm to use if a spped boost is needed over the A* algorithm and we have dynamic or moving objects.

6) Ibid, pg. 4-5
7) Ibid, pg. 5-6
8) Ibid, pg. 6-7

Tarry and Tremaux Algorithms[9]	Constructs a cyclic path passing through each edge once and only once in each direction. As the algorithm moves to each point in the graph, it will mark the direction that was traversed and continue moving around to the next point repeating paths as the line returns to the origin. Similar to the depth-first search algorithm used in graphs	Non-Heuristic	-Unknown Maze	- very little computation is needed	- Movement around entire graph twice	N/A to research, deals with unknown mazes. If obstacles are not connected to one another, the algorithm will fail.
Fraenkel's Algorithm[10]	Rather than repeating the paths as in the Tarry or Tremeaux Algorithms, Fraenkel's algorithm touches each point always moving forward, and never working its way back around.	Non-Heuristic	- Unknown Maze	- very little computation is needed	- movement around entire graph once.	N/A to research, deals with unknown mazes. If obstacles are not connected to one another, the algorithm will fail.
Pledge Algorithm[11]	Begins by choosing a direction to move. Walks following the the direction until an obstacle is found by the front sensor. Turn left and follow the obstacle boundary, keeping the obstacle on the right side. Follow the obstacle around until the turning angle is zero and go back to step 2.	Non-Heuristic	- Touch Sensing - Unknown Maze	- very little computation is needed	-Will continue the algortihm until battery dies or until the end of the maze is found.	N/A to research, deals with unknown mazes. If obstacles are not connected to one another, the algorithm will fail.
Lumelsky's Algorithm *Bug1*[12]	A Robot move towards the endpoint. When an object is reached, Robot moves around the object until the point at which is was reached is found, Robot will move back around and move forward on the other side of the robot. If distance travelled around the object is shorter to continue around object again, Robot will take shorter path (reverse or continue)	Non-Heuristic	- Unknown environment - Touch Sensing	- Will find a path to the finish.	-Moves around entire Object taking time.	N/A to research, deals with touch sensors.
Lumelsky's Algorithm *Bug2*[13]	A Robot move towards the endpoint. When an object is reached, Robot moves around the object measureing distance Up and when distance is reached coming back down on the other side of the object, robot will continue moving towards the goal.	Non-Heuristic	- Unknown environment - Touch Sensing	-Does not move the entire boundary of the object, as in Bug 1.	-May Visit the same obstacle in complex terrains	N/A to research, deals with touch sensors.

9) N. Rao, S. Kareti, and W. Shi. Robot Navigationin Unknown Terrains: Introductory Survey of Nonheuristic Algorithms. Department of Computer Science, Old Dominion University. July 2003. pg. 8
10) Ibid, pg. 9
11) Ibid, pg. 10
12) Ibid, pg. 12
13) Ibid, pg. 12-13

Sankaranarayanan's Algorithm[14]	Uses a similar algorithm as in Bug 2 however, when the finish is on the other side of an object that has on entrance, Algorithm willmove around the boundary of the entire obstacle until the robot can reach the finish inside of the walls of the obstacle.	Non-Heuristic	- Unknown environment - Touch Sensing	- Gets rid of the problem found in Bug2 because the paths in the algorithm are upperbound.	- Takes time to move around the entirity of a large obstacle to find opening	N/A to research, deals with touch sensors.
Cox and Yap's Algorithm[15]	As the Ladder or rod collides with an obstacle, the ladder searches the environment keeping in contact with the obstacle. As the ladder moves, a roadmap is created.	Roadmap	- Rod or Ladder Movement in Unknown 2-D terrains. -Touch Sensing	- Maps the area as algorithm is being initiated	- Only applicable to Rod/Ladder movement in 2-D terrains	N/A to research, deals with touch sensors.
Sutherland's Algorithm[16]	Robot scans the horizon, if no spurs are found, the robot can reach target directly, When Spur is in Vision. If a spur is located in the direction of the finish point, robot will move to a location to scan the found spur, this step is repeated until there is a clear path to the finish.	Non-Heuristic	- Vision Sensing - Unknown Terrains	- Will navigate through most simple terrains without many problems.	- Slow response since robot has to scan then move if a spur is found, rescan, etc. - Complex terrains (w/ small openings) may create a potential problem for the robot as it may not detec the opening as a spur.	Could be used if the map of the warehouse is unknown.

14) Ibid, pg. 13-14
15) Ibid, pg. 17
16) Ibid, pg. 21-22

References

D. Ferguson, M. Likhachev, and A. Stentz. *A Guide to Heuristic-based Path Planning*. Department of Computer Science, Carnegie Mellon University. Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling (ICAPS), June, 2005.

N. Rao, S. Kareti, and W. Shi. *Robot Navigationin Unknown Terrains: Introductory Survey of Nonheuristic Algorithms*. Department of Computer Science, Old Dominion University. July 2003.

W. Regli. *Robot Lab: Robot Path Planning*. Deptartment of Computer Science, Drexel University.