**Fast Motion Planning for Multiple moving robots**

Abstract: This paper presents an algorithm that provides a method to assign priorities for the concurrent motion of robots in a plane. By careful priority planning, it is possible to reduce the average running time of the path planner. It also attempts to maximise the number of robots that are able to move in a straight line. Like needed in our case, this paper proposes a method to resolve collisions where the moving robots themselves are the obstacles.

Summary: Obstacles in space time are effectively higher priority robots in motion. It says that for example, if there are two robots, they are assigned priorities as robot 1 and robot 2. But assigning priorities in this manner does not prohibit the parallel operation of the two robots. It just means that the path of robot 1 is planned first and the path of robot 2 is planned next and in the event of a collision situation robot 2 must take the responsibility of avoiding the occurrence of that. This paper does not deal with the algorithm the planner must employ to find the shortest path for a robot and that will be subject to size of the environment and can take large amounts of time. This merely aims to deal with the collision avoidance.

Prioritization rules: There may be many collisions between two robots moving in straight lines but the most critical ones occur at the start and goal points. Thus the rules for the start and end point conflicts are resolved in a manner as follows; If Robot 1 (R1) is to move from start point 1(S1) to goal point 1(G1) and intercepts R2 in position S2, then it is to give priority to R2 to move first. If R1 intercepts R2 in position G2, it is to give priority to R1 to move through that position before R2 can move into G2. Prioritization is done based on priority graphs. In the priority graphs the robots are arranged in a hierarchical manner and thus as we move from the root node downwards, we can delete the node corresponding to a robot from the graph thus revealing a new root node. Cycles are formed when two or more robots have priority over each other and the only way to resolve this is by making one robot wait for the other one to complete its transition. The nodes/robots through which the most cycles are encountered are termed as complex robots and are deleted from the graph and assigned the lowest priority. The remainder of the robots in the priority graph can now have linear motions. If there are 2 robots where one has a start conflict and the other has a goal conflict (thus creating a cycle) , we can resolve it by using a via point where the assigned priorities can be interchanged. LINEAR PLANNER: This planner has to ensure the straight line path of the two robots and that they don’t intersect/collide. Speed control is assigned depending on the priority of the robots approaching the collision point. Once either one of the robots has crossed the collision point, the other robot can also start moving at max velocity. The COMPLEX PLANNER is not proposed in this paper. Though a brief algorithm has been proposed where the complex robots are first positioned at safe via points (intermediate points) and then the robots with higher priorities are moved and then the complex robots are moved.

**HOW WE CAN INCORPORATE THE IDEAS FROM THIS PAPER INTO OUR PROJECT:** This paper gives us an idea as to how a collision can be resolved between multiple mobile robots in an environment. We have learned that we can prioritize the motion of two robots depending on hierarchy (these can be dynamically assigned based on the distance a robot has to move or the distance a robot is from a collision point etc.) and speed control can be employed. Most importantly, this paper has also given us a way to solve the start point and end point collision problem while maintaining a straight line motion of the robots.

**Dead-Lock Free and Collision-Free Co-ordination of Two Robot Manipulators**

Abstract: This paper discusses a method for coordinating the motion of two robot manipulators so as to avoid collisions between them. It aims to be able to plan the paths of the two manipulators nearly independently and so that the motion of the other robot isn’t taken into consideration. In keeping with this idea, it thus has to come up with a way to avoid deadlocks and collisions along the path. It thus gives us a method to provide explicit coordination commands. This paper is strictly interested in the problem of coordinating the trajectories of the two manipulators.

Goals set for trajectory coordinator: plan the path for each manipulator independently, trajectories must guarantee that the manipulators reach their goals, should be possible to execute the trajectories without precise time coordination between the manipulators, safety of manipulators must not depend on accurate trajectory control of the manipulator.

Method: The approach presented in this method is such that it first assumes that the path for a manipulator has already been planned and that the path is composed of smaller path segments. The path segments are of fixed sizes and it is assumed that the time required to cover each path segment takes roughly the same time and so this trajectory coordination problem is broken down into a scheduling problem where a common region is the shared resource. This paper defines a collision region as the intersection of the path segments being executed by the two manipulators.

Proposed procedure for the Scheduler:

Begin  
i = 0, j = 0  
while i<m, j<n  
 begin

If Ri,j is collision free

Then begin

If i<m: execute Ai; i=i+1; end

If j<n: execute Bj; j=j+1; end

End

Else

If i<m and Ri,j-1 is collision free then

Then begin: execute Ai; I = i+1; end

Elseif j<n and Ri-1,j is collision free then

Then begin: execute Bj; j = j+1; end

Wait for completion

End

The schedule is constructed by a central controller as is in our project. Basically the algorithm keeps track of the next path segment to be executed by each manipulator and checks whether the simultaneous execution of the two path segments could cause a collision. Deadlock situations can be avoided by filling in the non-convexities in the collision region so as to prevent the manipulators from even going close to the collision region. Keeping these points in mind, a schedule can be constructed from this algorithm. Thus this method broadly uses a time scheduling scheme to avoid collisions and does not attempt to provide alternate paths to overcome the collision regions.