Autonomous Smart Factory: Assembly and Warehousing

Tor Istvan Stadler Kjetså, Tim Allen, Sangeetha Reji and Khushdeep Singh.

***Abstract*—The Smart Factory describes a production environment that organizes itself and is the main focus of Industry 4.0. It is based on cyber-physical systems and the smart networking of products and machines.The Scenario explained here includes a warehouse with a package generator, input and output trays for the packages to move around and robots to perform the storing and transportation. Our objective is to develop and implement a scalable and unique architecture for the system also considering the existing state of the art systems. The architecture developed in this paper deals with a centralized task planner that deals with all the packages the agents are transporting and a decentralized path planner inside each agent to create it's own path to the target trays or charging units. The system deals with multiple robots, so the main objective is to implement the multi mobile agent coordination and navigation without collision. Once it was implemented, the system performance with variable parameters like time taken by an agent to deliver a package, number of agents used to complete the task, and the total time the system runs without collision is analyzed and recorded. By studying these parameters, we are able to provide a conclusion about the ideal number of required resources to run the system.**

***Keywords*—Agent/Robot, Task Planner, Path Planner , Motion Planner, Heat Map, PID.**

I. Introduction

An increase in the retail sales business demands optimal logistics in warehouses. To cope with the increasing economy of these businesses, the warehouse logistics processes become complex. This complexity enables the deployment of robots for logistics. Meanwhile, it saves time, money, and increases employee productivity. These changes in industry structure improve the work space quality for employees. As the warehouse economy grows, deployment of robots for logistics automation seems to be a scalable solution. This technology holds the potential for the future, demanding optimality on safety, and robot technology parameters. An example of a state-of-art logistic distribution center is Amazon's Kiva systems. Kiva systems attain high accuracy and smooth workflow with mobile agents[1].

This project aims to implement a similar multi-agent solution for manipulating commodities in warehouses. It aims to tackle multi-agents challenges such as task scheduling and path navigation in a real factory alike environment. Taking inspiration from Kiva systems, the project uses some novel state-of-art solutions, while striving to make multi-agent navigation and scheduling better. With a goal to attain scalability for agent architecture in a factory environment, the project targets optimal solutions with robot technology. Some of these existing solutions are derived from either a centralized or decentralized planning entity to deliver the task to agents and plan its course to the target. This project deals with a hybrid approach with a centralized task planner and a decentralized path planner for the robots, which will be discussed in detail later[2].

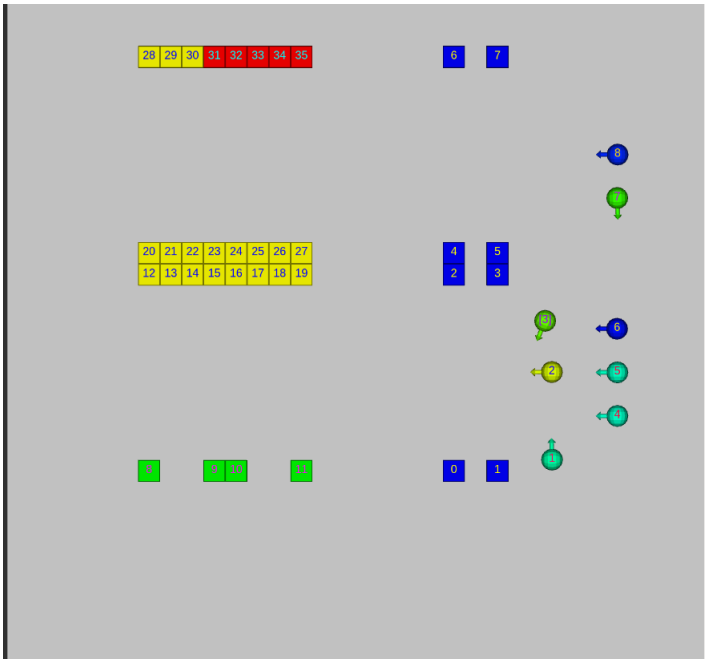

Fig. 1. Simulation Environment

The environment consists of a team of mobile robots taking the products and storing them in the warehouse. The system should deliver an optimal robot for the load and store of each task and an optimal path for the robot to follow. The centralized Task Planner deals with the tasks that are assigned to the robot based on the closest distance, shortest time and battery of the robots. Each robot is equipped with a Motion Planner to control and coordinate its movements with a PID controller and a Path Planner to generate its path to the target. For the purpose of this project, we have also assumed that each robot is identical, the static objects are known and that the robots are fully charged in the beginning of the simulation.

A non operating simulation environment is shown in Figure 1. The robots are numbered from 1 to 8 and the charging stations in blue colour are numbered from 0 to 7. The input trays where the packages appear are shown in green colour and are numbered from 8 to 11. The source trays to which the robots should deliver packages appearing in source trays are shown in yellow and red colour and are numbered from 12 to

35.

A fully automated system will also result in collisions and disruptions during its performance. If this system is not monitored manually, this could result in disruptions in the simulation environment. So the major part of our goal is collision detection, which should result in a smart factory environment that runs for an indefinite point of time. The collision detection is implemented with a centralized traffic management system which detects a possible collision based on the estimated position of robots in its future and warns one of the robots involved in the collision. This is also assisted with the laser detector each robot is equipped with which detects obstacles in its near vicinity.

The result of the implemented architecture was analyzed by running the simulation environment with the developed Task Planner, Motion Planner, Path Planner and Traffic Management with different number of agents. An ideal number of agents with which the system gives maximum performance is then identified. The system that uses least battery on each of its robots and also finishes early is considered as the ideal system with an ideal number of resources.

This paper continues with few related works that the project is inspired from, the methodology in detail, the result we have obtained and the conclusion we have derived from the results.

II. RELATED WORKS

A. Amazon robotics system

An example of a scalable and efficient automation warehouse for picking and placing goods is the Amazon robotics system [3] . Features of this system include-

- Robots work in an orderly manner avoiding a collision, and sensors are used to avoid a collision.
- The best route and nearest pickup location are calculated automatically and later resources are dispatched as per a scheduling algorithm.A control system assists the robot to make path selection.
- The robot uses a camera to read the bar code while following a path and automatically goes to the charging
- Robots are considered into one of the states- Performing tasks, Released into next round, or Waiting for performing the task.
- Robots continuously broadcast their location while following a path, at the same time they receive directions from a centralized computer station.

Drawbacks of this system-

- The collision detection system can only stop the agent and cannot follow a different path.
- Agents only follow a fixed trajectory in the Amazon warehouse.
- Robots may line up in a queue for charging.

B. Multi-agent path planning

Autonomous agents have a fundamental capability of navigation [6] . Firstly, path planning for multi-agents includes a Coupled centralized approach. Wherein, the path is planned in the configuration space of the entire robot team. In the dynamic environment, this approach does not return optimal paths for a single robot in polynomial time. As the number of robots increases, centralized techniques are impractical due to exponential increase in computation time, thus the approach is limited to small problems.

Secondly, the decoupled path planning approaches are considered [6] . These can be centralized or distributed. The idea is to break the problem into parts with each robot following individual planned paths and then coordinating to avoid collisions. Thus, a trade off between the solution quality and efficiency is observed. Collision avoidance can be done by assigning priorities to agents. Thus, plan for a first robot using a single path planning approach and successive agents treat high priority agents as moving obstacles.

Thirdly, a trade-off exists between Global and Local control for multi-agent systems. Local control assists the agent in have emergent decisions, while faces limitations when desired group behavior is expected. On the other hand, Global control consisting of a centralised controller allowing more coherent coordination. Meanwhile, increasing the inter-agent communication and thus affecting the system efficiency.

C. Multi-agent motion planning

At first, centralized methods for motion planning has many industrial applications such as product assembly, warehouse management, controlling robots in a structured environment and transportation applications. Safety and completeness are assured with this method. Meanwhile, decentralized methods assure collision and deadlock avoidance while increasing information exchange between the agents [8] . Algorithms for path planning and scheduling are viewed in [9] and [10].

The project proceeds with a hybrid approach for planning. Centralized control provides static maps through a Roadmap generator and the Heatmap generator provides information about the likelihood of collision following a specific path. It also assists with collision avoidance. Decentralized control uses static maps for optimal path generation, collisions are avoided uses Heatmap information. In order to have optimality, inter-agent communication is avoided. This is achieved with an obstacle detector and motion planner. Sensors guide the agent through the navigation and motion planner smooth the path by providing low-level signals to actuators.

III. METHODOLOGY

A. General Framework

As mentioned earlier, this project utilises a hybrid approach combining both centralised and decentralised aspects. Figure 2 shows a visual representation of the system with components split between centralised and decentralised. The different

components which are either of centralised or decentralised nature are outlined below. To understand the control flow of the different centralized and decentralized nodes, a control flow chart of the system is also attached to the appendix. It depicts the communication and control transfer between the different nodes of the architecture.

1) Centralised: The system uses a centralised task planner which creates a queue of tasks for each robot. It uses information such as proximity of each robot to the tray, as well as the current battery level and number of tasks in the robot's queue to calculate a score on which the task Planner bases its decision. Another centralised element of the system is the generation of a Heatmap used for obstacle avoidance. This heat map shows the probability of certain nodes being occupied at a given time, so that a new path can be created prior to a potential collision. The charging management uses the output of the battery monitor to sent the robots for charging if it falls below a certain threshold. For the purpose of this report, since the agents can be studied without running the system for a longer period of time, the results are obtained without implementing this functionality.

2) Decentralised: Once the robots are assigned a task, the decentralised path planner will compute the two paths in order to complete the task. First it computes a path to the input tray, where the package is. Then a second path from that tray to the destination tray is computed. Other decentralised elements of the system are the motion planner and obstacle detector. Each robot has its own identical motion planner to move along its own unique path. The motion planner is in charge of making the robot correctly move along a defined path the Path Planner provides. The main functionality will include smooth motion of the robot in a linear and angular direction. Obstacle detection and collision avoidance is developed using the centralised Heatmap/collision-monitoring functionality as well as the laser sensors of the robot.

Each robot/ agent is equipped with three main sensors, the pose sensor for providing the actual position of the robot in the warehouse, the laser sensor which detects static or dynamic objects in its near vicinity, the radius of the sensor can also be defined by the user, the battery of the robot is initialized as 100%, which reduces as the robot starts moving and this will help us to determine when the robot should be sent for charging. A battery monitor in each agent is set up for this functionality. The gripper of the robot is initialized when it nears a package, the functionality of the gripper is to correctly pick and drop a package to its defined trays. The storage management is in charge of produces the packages the robot transfers around in the warehouse environment. It generates packages in the source trays. The methodologies implemented will be further discussed in their own sections below.

B. Task Planner

The system begins its operation from the Task Planner node.The system has a package generator that generates packages, and as each package is generated, task planner is supposed to consider it as a task and do the needed. In this case, the needed accounts for an ideal tray and an ideal robot to finish the task, or move the package to a desirable location. Each package is considered as a request for the task planner to allocate required resources. Each robot is expected to have a queue with the number of tasks assigned to it by the task planner and the robot does the tasks one by one that is in its queue. Before the task planner assigns a task to the robot, it compares all the existing robots based on a score each robot calculates. The score is calculated based on the proximity of the robot to the source tray, where package is generated, the battery level of the robot and the number of tasks assigned to the robot.

A robot with high battery level, closer to the source tray and with less tasks in its queue will be the ideal robot for a task announced.The distance to the tray can be requested from the path planner as the path that the agent will take to reach the tray, but for the purpose of this report, euclidean distance between the trays and agents is considered. Each agent, when the project begins is assigned with a battery level of 100%, which decreases gradually during its motion though the environment. Thus, in the beginning of the simulation, each robot will have equal battery level to add to the score.

C. Path Planner

The Path Planner is decentralised and computes the optimal path from A to B using an A*-algorithm. The path computation does not account for dynamic obstacles. A predefined road map (including edges and nodes) is read by the path planner upon initialization. This road map is used to set necessary variables and initialize a graph (from boost library[5]) in the path planner node. This graph represents the road map in a way that is easier to apply a shortest path algorithm to. The A*-algorithm traverses this graph when computing the optimal path from a given start point to a given end point. Only nodes and edges that do not collide with static obstacles are present in the graph.

Whenever a new task is active in the agent, the path finding-service will be called twice; once to compute path from current position to input tray and again to compute path from input tray to storage tray.

(Extra: The path finding algorithm has also a currently unused functionality; when given a vector of nodes that are marked as occupied/obstacled and a flag indicating that an alternative path has to be computed, the graph is updated and excludes edges (sets their weight to infinity) to the marked nodes and computes an optimal path concerning these nodes as unavailable. At the end the graph is restored to normal. This is not perfected, and therefore not used. The efficiency of this compared to stopping and resuming path when necessary is also uncertain, which is why the ladder has been pursued)

D. Motion Planner

The Motion Planner is the part of the project concerned with the physical movement of the robot from point A to

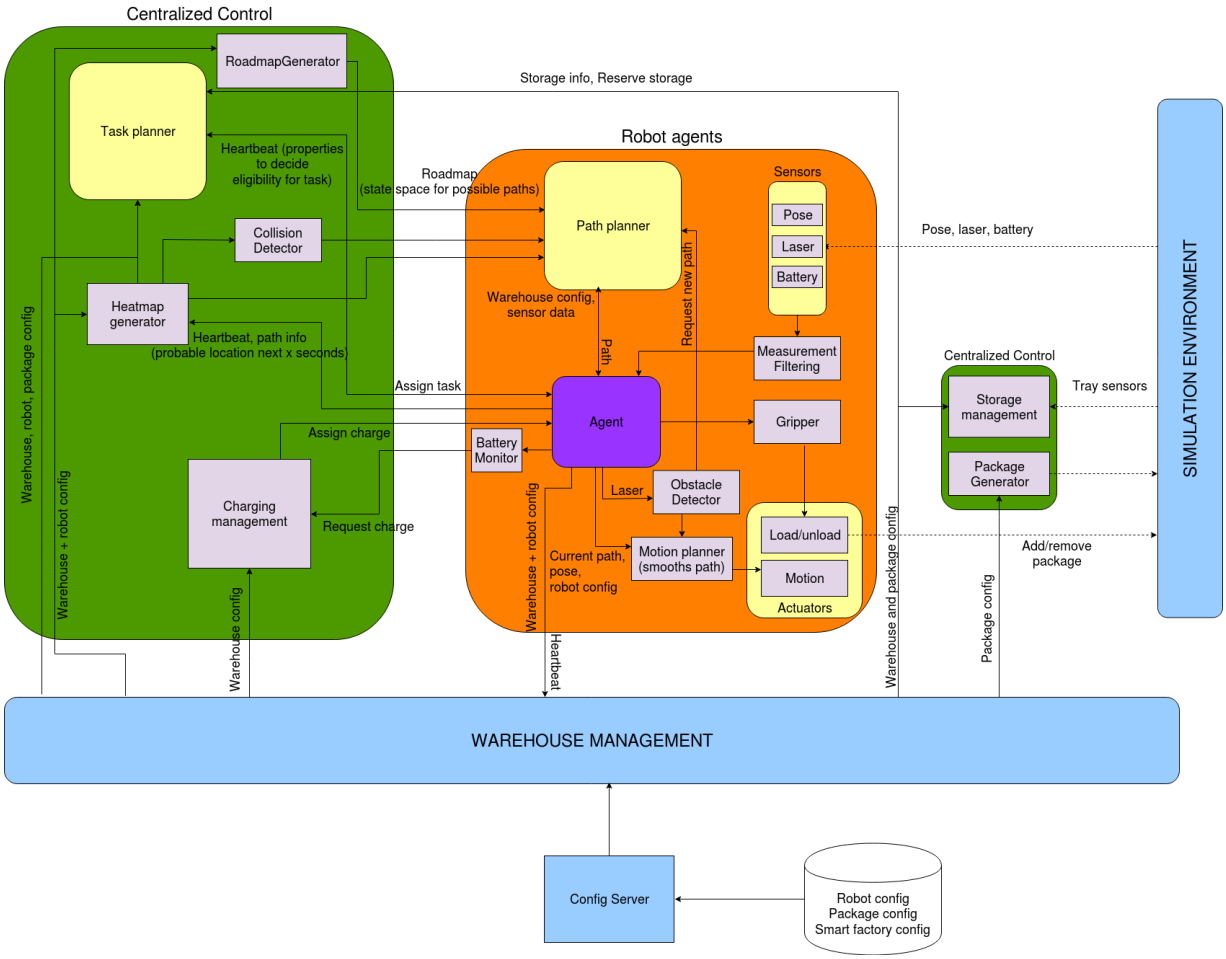

Fig. 2. System Architecture

point B. The Task Planner has already determined points A and B and the Path Planner has provided a vector of Points for the robot to follow. Now, the Motion planner will smooth the existing path and trigger a drive function to approach the tray quickly and smoothly.

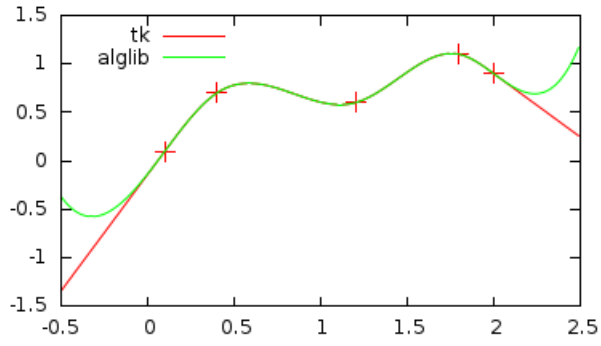

Fig. 3. The cubic spline algorithm

1) Path Smoothing: In order to create a smooth path for the individual robots to follow our system implemented the widely used cubic spline interpolation algorithm. This method gives a interpolated polynomial from a given path where each point on the new path does indeed cross the original data points however the path between original points is curved. Figure 3 below shows an example of how the algorithm changes a path. The data points of the original path are the red '+' symbols and the green and red lines show different versions of spline algorithms. The algorithm used in this project is better represented by the red line since the beginning and end of the new path is also linear.

2) PID control: The PID controller deals with both linear and angular control together so that the robot is able to move along its 2 dimensional path. Our system has split the PID controller into two, with the first part being from the starting point to a drive point perpendicular to the tray, and the other part being the short distance from the drive point to the tray. This is for the simple reason that precision of the final orientation and position of the robot has more importance in the final stages of the robot's approach to the tray. The

proportional gain (Kp) for this last section is reduced from 0.5 to 0.3 and an integral gain (Ki) is introduced to reduce overshoot. Since the path is non-linear the error must be calculated between the current node and the next node (as opposed to just the start and end points).

E. Traffic Management

1) Heatmap: At a fixed time interval, when in motion, the robots will publish an estimation of their location with timestamp up until 4 seconds into the future. This estimated path is subscribed to by a centralised unit called the Heatmap. The Heatmap continuously monitors whether a collision between robots is likely to occur based on their estimated locations and times. The Heatmap will classify a collision if two or more robots have estimated that they will be at locations within 0.5 m of each other within a time frame of 0.5 seconds.

Upon a collision one of the robots involved will be selected to handle the collision. The priority is based on the agent ID. The coordinates of where the collision is likely to occur are published to the robot that has been tasked with handling the collision. These coordinates make up a vector of 'dangerous points'. Each individual motion planner will check if the robot is heading into one of these 'dangerous points', and if it is, it will stop. Once the point is no longer dangerous or 5 seconds have passed the robot will resume its motion.

The Heatmap is continuously updating each agent's 'dangerous points', so if there is no collision, an empty vector will be published and stored in each agent.

(Extra: Like the path planner, the Heatmap also has an unused functionality - which is connected to that in the path planner. A service called VerifyPath takes a request containing a path. This path is compared with the knowledge the Heatmap possesses concerning dynamic obstacles. The response of the service is a vector containing all the nodes the requested path will collide with and a flag indicating whether it will collide or not. This was intended to be used together with the functionality of creating an alternative path based on unavailable nodes in the path planner.)

2) Obstacle Detection: In addition to the Heatmap that attempts to predict collisions, a decentralised obstacle detection based on laser sensors is implemented.

Each robot has laser scanners installed, and by analyzing the signals from these and using the information of the agents position and orientation it is possible to compute the coordinates of detected obstacles relative to the warehouse map. With this information and the information regarding the coordinates of all static obstacles in the environment it is possible to classify a detected obstacle as a static obstacle (e.g. a tray) and thus ignore it or as a dynamic obstacle and thus handle it accordingly.

The functionality for handling detected obstacles works as follows: First the detected obstacles are classified as static or dynamic. For each dynamic obstacle, the angle of the obstacle relative to the orientation of the robot is computed, and if it is within 30 degrees left or right of the robots orientation, the robot will stop. To implement prioritized maneuvering out of a potential deadlock situation (e.g. two robots facing each other): if obstacles persists after 2 s, the robot with an orientation between -90 degrees and 90 degrees relative to the environment will insert an intermediate point into its current path that will lead it away from the obstacle and resumes its motion. This will lead the robot out of collision range of the other robot, which then will resume its motion as well. The orientation criteria is a simple way to implement priorities which works quite well in most cases. If this does not resolve the collision a predefined maneuver of backing up and turning will be triggered after obstacles have persisted for 10 s.

IV. RESULTS

We tested performance of program with the number of robots varying from 1 to 7, and for each number of robots we did 5 tests. Each test consisted of the program running for 5 minutes as depicted in Figure 4.

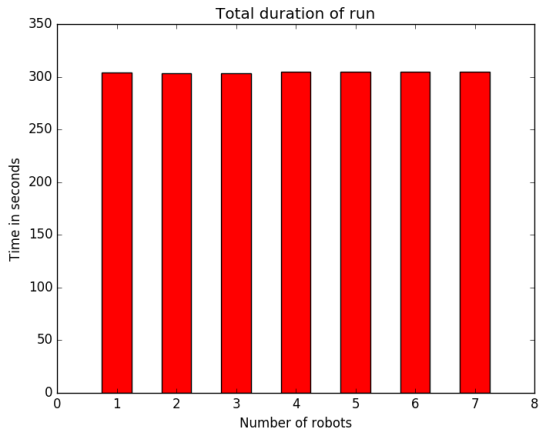

Fig. 4. Duration of simulation runs

This implied the results consists of equal time granted for trial runs with specific number of robots. Accordingly, the robots either completed all the tasks in the given time interval or could not fully complete those due to time spent in collision avoidance. Certainly, the performance of different no. of robots were evaluated on the following criteria:

- **Average battery consumption** - This emphasize the average battery consumed throughout the simulation run as in Figure5. The figures show an expected trend: more deployed robots performing tasks effectively with less overall battery consumption. So, generally the battery consumption is expected to decrease if we increase the number of robots. However, robots also spend time if they encounter collision. For example comparing robot number 4 and 5, we say that battery consumption is more with 5, due to robots spending time and energy in collision avoidance. .
- **Average time per task** - In order to better understand the efficiency of the system, consider the Figure 6 describing the average time to complete one task (from

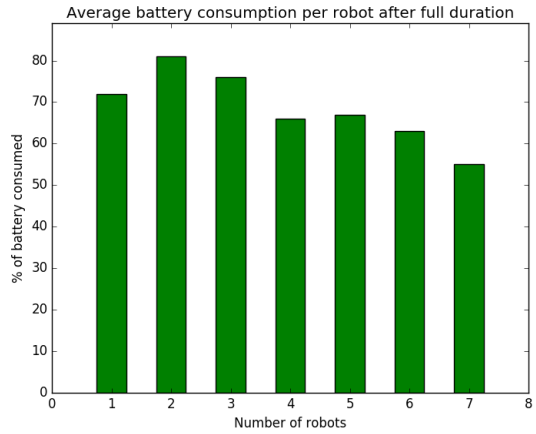

Fig. 5. Average battery consumed to complete all tasks

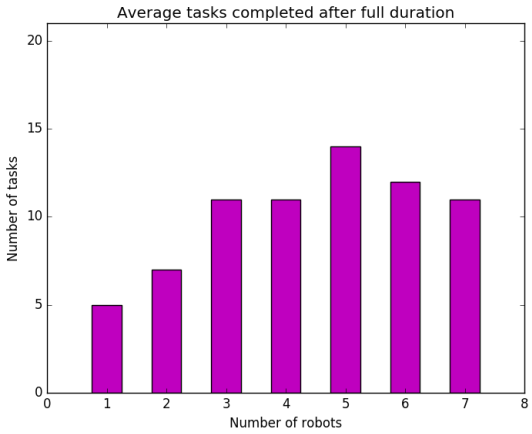

Fig. 7. Total completed tasks

start to finish) with different no. of robots. We do not expect a trend here, because this parameter will be high depending on how short the task was and time spent in collision avoidance. Basically, if we have robots with multiple detected obstacles (happens often with more than 4 robots) trying to find a safe path, then as per the current algorithm it is a bit time consuming for them to safely maneuver.

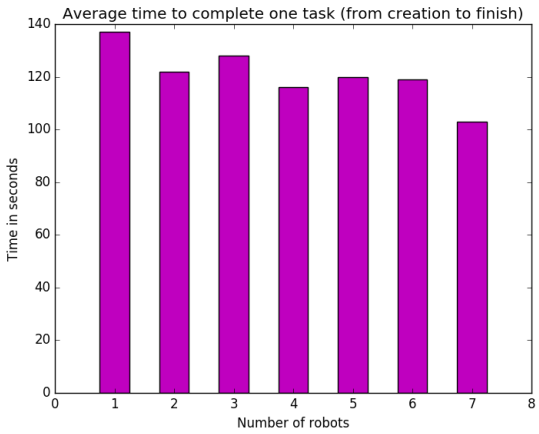

Fig. 6. Average time to complete one task

- **Average task completed** - This is an vital point to examine in order to test the effectiveness of algorithms and efficiency of the overall system. As seen in Figure 7, 5-robot system is the most efficient in terms of loading and storing packages (based on 5 simulations). This implies that the methodology developed was most suitable for 5-robot system. Also, it is interesting to see 6-robot system performing better than 4 robots.
- **Highest task completed in one run** - Comparing the best simulation performance for different number of robots in Figure 8, the system performs best with 5-robots. Followed equally by 4-robot and 6-robot systems. The analysis helps to understand the possible best case scenario for different robots. Although, increasing the number of robots adds to collision complexity and it takes several trial runs to identify the best possible scenario.

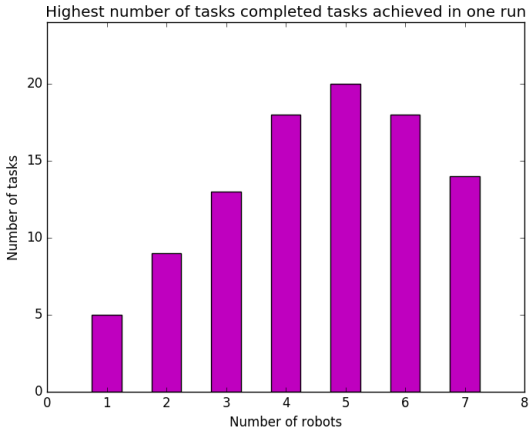

Fig. 8. Most tasks completed in one simulation run

V. Conclusion

The system we have studied contained of a variable number of robots, a package generator generating packages to input trays and robots to transport the packages to output trays. The system was also equipped with a a static road map for collision avoidance and a charging management to assist the robots in case the robot dies. The centralized task planner and heat map and the decentralized path planner and obstacle detector assists the robots in smooth movement along the warehouse environment.

The goal of this project was to allocate resources to each input package, provide path to each robots, have a smooth

motion of the robot and to transport packages properly through the warehouse. Each robot present in the simulation had to be given a path to its target, which will make the robot reach its destination in a lesser amount of time without running into collisions with any static or dynamic objects in its path. the static objects will be the charging stations and other input and output trays. Dynamic objects are the other robots present in the system doing tasks. A major goal of this project was collision avoidance, which can be achieved using proper Traffic management or collision avoidance algorithms.

The task assignment should be done considering the number of robots present in the system and it should divide the tasks equally among all the existing robots. A task is announced when a package is waiting to be pick up by some robot, the ideal robot for a task is considered by its proximity to the package, battery level of the robot and the number of tasks already assigned to the robot. This will help in reducing the overall time the system takes to finish the tasks. The robot once it has a destination, requires a path to its destination. This path has to be the shortest and also should not be in the vicinity of any static or dynamic obstacle. This will help the robots to reach the destination in a short period of time. Once it runs into a collision, collision avoidance will take some time which will reduce the overall performance of the system. The traffic management along with the laser detector is in charge of reducing the number of collisions the system has.

During the simulation, few observations have shown that the task assignment is not proper sometimes, where the task planner does not choose the closest robot. Also if a robot loses its heartbeat, the task planner doesn't consider the robot anymore and distributes the tasks among other robots. This robot will then be a static obstacle that is not present in our static road map, which will create more collisions the traffic management cannot avoid.

A functionality where the robots drive straight backwards after delivering a package would probably have improved the performance of the program. A lot of collisions occur in the loading area where there is not much space, and the robots collide when turning to drive back. The amount of collisions then starts snowballing which again hinders performance quite significantly. When the robots starting positions are further away from each other they are less likely to appear in the same loading areas at the same time, and this also reduces the number of collisions. The improvement is more apparent with many robots.A more comprehensive detection of idleness/non-responsiveness would be useful. Some robots seems to just be in the way after a while, and a way to move them out of the way/restart them would probably be performance enhancing. for this, a robot can be given an idle point which should be away from the path of other robots, where it moves to that point if it does not have any task in its queue.

As mentioned in sections III-C and III-E an alternative approach for avoiding collisions was attempted, based on keeping the robots moving an changing their paths instead of stopping. A hybrid between this and the current stop-and-maneuver functionality might be a good solution as it could help loosening up the large heaps that are formed when robots stop repeatedly because of several detected obstacles. In future performance optimization of the system, a method to retrieve the heartbeat of robots along with fast and optimized collision avoidance can be implemented.

REFERENCES

[1] Wikipedia, *Amazon Robotics*, https://en.wikipedia.org/wiki/Amazon_Robotics

[2] Rosina Geiger, *Humans And Robots Collaborate In Today's Warehouses*, 01-03-2018. https://www.digitalistmag.com/digital-supply-networks/2018/03/01/humans-robots-collaborate-in-todays-warehouses-05922042.

[3] Jun-tao Li, Hong-jian Liu, *Design Optimization of Amazon Robotics*, Automation, Control and Intelligent Systems Vol. 4, No. 2, 2016, pp. 48-52. doi: 10.11648/j.acis.20160402.17 http://www.sciencepublishinggroup.com/journal/paperinfo?journalid=134&doi=10.11648/j.acis.20160402.17

[4] MWPVL International- Leadership in supply chain and Logistics consulting, *Is Kiva systems good fit for your distribution center?*, https://mwpvl.com/html/kiva_systems.html

[5] *Boost C++ libraries* https://www.boost.org/users/index.html

[6] Dr. Lynne E. Parker, University of Tennessee, Knoxville USA *Multi-Robot Path Planning and Motion Coordination*, http://web.eecs.utk.edu/~leparker/Courses/CS494-529-fall14/Lectures/16-Nov-11-Multiple-Robots.pdf

[7] *Multi-Robot Path Planning*, https://www.cpp.edu/~ftang/courses/CS599-DI/notes/path\%20planning.pdf

[8] Kostas E. Bekris, Department of Computer Science, Rutgers University *Motion Planning in Multi-Robot Systems*, http://multirobotsystems.org/sites/default/files/slides/2015_RSS_MRS_Bekris.pdf

[9] Damjan Miklic, Stjepan Bogdan, University of Zagreb *A control architecture for warehouse automation - Performance evaluation in USARSim*, IEEE International Conference on Robotics and Automation, June 2011, DOI: 10.1109/ICRA.2011.5979972 http://multirobotsystems.org/sites/default/files/slides/2015_RSS_MRS_Bekris.pdf

[10] Damjan Miklic, Tamara Petrovi c, Mirko Cori c, Zvonimir Pi skovi c, and Stjepan Bogdan *A Modular Control System for Warehouse Automation - Algorithms and Simulations in USARSim*, IEEE International Conference on Robotics and Automation RiverCentre, Saint Paul, Minnesota, USA May 14-18, 2012 http://repozitorij.fsb.hr/7758/1/Modular\%20Control\%20System\%20for\%20Warehouse\%20Automation\%20-\%20Algorithms\%20and\%20Simulations\%20in\%20USARSim.pdf

APPENDIX
CLASS DIAGRAM

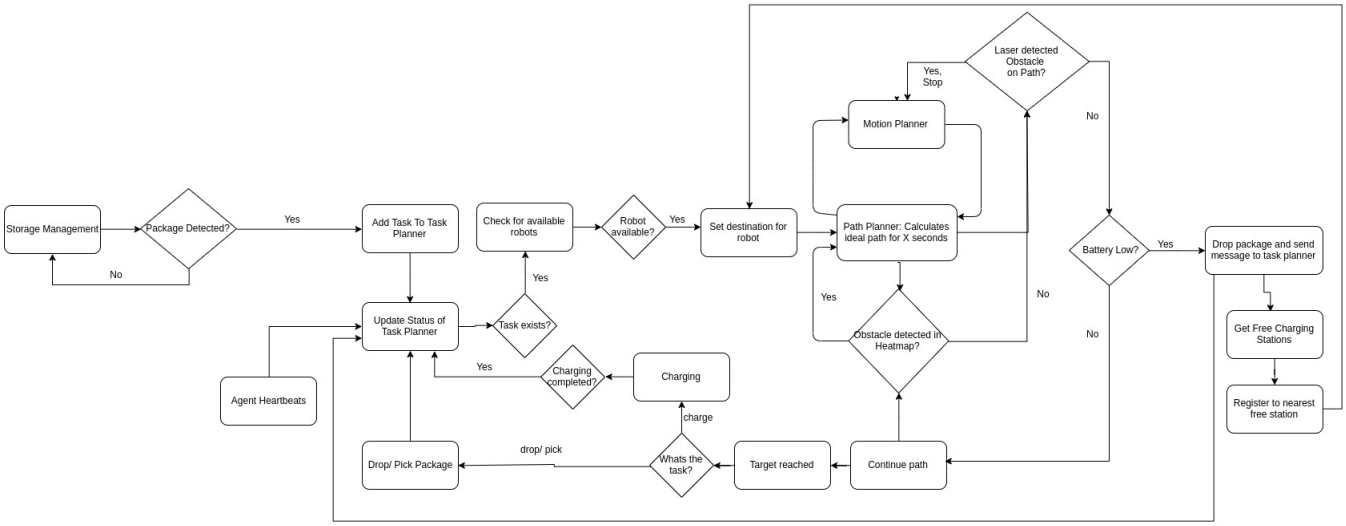

Fig. 9. System Architecture Control Flow Chart